User Manual

This toolbox was developed as part of an ERASMUS+ Internship Project by Pedro Félix Martins Alves (pedrofelixalves@gmail.com), and under the supervision of Khaled Nasr, in the **Charité – Universitätsmedizin Berlin**.

The objective was to develop a flexible and user-friendly toolbox for simulating the response of several different neuron types to AM stimulation, adapted for parameter sweeps and for running large amounts of simulations with need for little user interaction. This was achieved through two different methodologies, and different levels of model complexity, which have been made available on Github, together with instructions for utilization:

1) for simplified ball-and-stick minimal Hodgkin-Huxley (HH) type models for different classes of cortical and thalamic neurons, based on minimal neuron models obtained from (Pospischil et al., 2008) and modified for extracellular stimulation, referred to henceforth as ST (simple toolbox) and available in: https://github.com/pedrofurrix/HH_simple

2) for detailed morphological models of cortical cells of different layers, obtained from the Blue Brain Project (Ramaswamy et al., 2015) and adapted to adult human and rat by (Aberra et al., 2018), referred to henceforth as DT (detailed toolbox) and available in: https://github.com/pedrofurrix/241165. This makes use of the original code from (Aberra et al., 2018), available in modelDB (https://modeldb.science/241165), which was augmented and adapted for GUI-free utilization and AM-stimulation.

Twenty-five different neuron models from five different cortical layers are available as part of the DT, and it is possible to easily add additional morphologically realistic cell models, as it involves only downloading them from the Blue Brain database (Erö et al., 2018), adding their identifiers to the list in the "cellChooser.hoc" script and removing the gui import in the initialization script.

The rationale behind making both toolboxes was similar, based on a modular approach with several auxiliary scripts and a couple of main scripts that run the simulation and are supported by wrappers that enable simple interaction to allow for parameter changes. The simulation has been optimized to enable recording the membrane potential in every neuronal compartment at every timestep.

Additionally, the toolboxes are prepared for both batch and parallel running. We avoided using NEURON's Parallel Context since each simulation is only of a single neuron, and this method adds significant complexity in the management of data. As such, the interest in parallelization is more to enable a set of simulations to run simultaneously, since parameter sweeps are associated with the need for a large number of simulations to be run. This significantly lowers the computation time needed for a large batch of simulations to run(Singhal, 2018).

We looked to optimize both the storage space and the run time for each simulation, through recording voltages and time in the memory and flushing to a HDF5 file after a certain number of timesteps. This file format is ideal for storing a large amount of structured data, while enabling fast I/O operations, partial data access and great compression (The HDF Group, n.d.). Additionally, it is very versatile and supports saving multiple datasets in the same file.

Using this strategy enabled us to lower the storage space needed for each run by around 11.5x and 3.75x, compared to csv and npz, respectively.

Console outputs are saved to a log file, to enable debugging after the simulation.

The toolboxes also contain important data processing and analysis scripts, that may serve as a starting point for anyone looking to evaluate neuronal response to AM-stimulation.

Main functionalities:

- **Simulation of uniform-field AM stimulation** (customizable parameters, compatible with batch and parallel running). This simulates the chosen neuron model's response to AM-stimulation with the specified parameters, including simulation time, integration timestep, electric-field orientation, amplitude, carrier frequency, modulation frequency and depth, recording the membrane potential, local field potential and stimulation current at every timestep.
- **Threshold detection** – detection of the minimum electric-field value that induces rhythmic firing at the modulation frequency for the specified simulation parameters. This follows a similar process to what we described in the methods. It is prepared to consider every compartment or only the soma, and to remove bursting from the spike count, returning the membrane potential over time, spike number and spike times in the compartments that were monitored for spiking.

The toolboxes also contain important data processing and analysis scripts. This can be ran through "post_processing.py", which filters the data and provides summary files containing the maximum depolarization, maximum hyperpolarization and Fourier power around the modulation frequency. These are prepared for var="cfreq", but can be adapted for different ones without much added complexity.

These can also be further optimized and interactively changed through the Jupyter scripts also available.

Extracellular stimulation of neuron models

Quasi-static approximation

The stimulation of model neurons is performed using the quasi-static approximation. This enables separation of the electric field into spatial and temporal components, significantly reducing the computational complexity of the task and enabling a model simplification that should reduce the ambiguity and variability of results (Bossetti et al., 2008; Wang et al., 2024). This is associated with several assumptions, namely that the tissue's properties are linear, that it is purely resistive (ignores capacitive properties) and nondispersive, and that there is no wave propagation or self-induction. This is discussed in detail in this review (Wang et al., 2024), and it is considered a reasonable approximation for frequencies under 100 kHz, so it should be adequate adequate for our project that looks to characterize carrier frequencies up to 35 kHz (Wang et al., 2024).

The extracellular potentials can thus be defined, for each neuronal compartment, by equation 6:

$$E(x,y,z,t) = Ve(x,y,z) \times i(t) \tag{1}$$

The spatial component of the extracellular potential Ve(x,y,z) is independent of frequency and does not change with time during the pulse waveform(Aberra et al., 2019). It can be assigned to each cellular compartment based on NEURON's xtra mechanism, then scaled uniformly in time based on a time-dependent waveform (i(t)) and applied to the model through the extracellular mechanism ($E(x,y,z,t)$).

Spatial component

- Uniform Electric Field Approximation

Since transcranial NIBS methods like tES and TMS generate electric fields with low spatial gradients at the scale of single neurons, it is viable to consider that the electric field affecting a single neuron is uniform (no spatial gradient) (Aberra et al., 2018; Bikson et al., 2012). This approximation also significantly simplifies the model and avoids computation of spatial E-field distributions (Joucla et al., 2014; Mirzakhalili et al., 2020). The results obtained can be put together with the E-field distributions in a head model to estimate the response in each brain region (Shirinpour et al., 2021).

The fundamental idea is to calculate the electric field in the region of interest and apply it to neurons as a constant, defined only by its amplitude and orientation(Bikson et al., 2012).

As such, for calculation of the extracellular potential at each location $(Ve(x,y,z))$, we followed the equation defined in (Aberra et al., 2018). Similarly to their strategy, we considered the origin (soma) to be the reference point, that forms the zero potential plane, perpendicular to the electric field. The potential at any point (x,y,z) is proportional to the distance of that point from reference point in the direction of the electric field (Cavarretta et al., 2014).

The orientation of the field is assigned based on spherical coordinates, with polar angle theta (θ) and azimuthal angle phi (φ), relative to the neuron's somatodendritic axis.

We considered a unitary electric field amplitude (1 V/m) for this calculation, defined in equation (7). This simplifies steady state calculation, as the amplitude can be defined by multiplying the temporal component by a scaling factor (A).

$$Ve(x,y,z) = -(x \sin\theta \cos\varphi + y \sin\theta \sin\varphi + z \cos\theta) \qquad (2)$$

The term $(x \sin\theta \cos\varphi + y \sin\theta \sin\varphi + z \cos\theta)$ corresponds to the dot product between the electric field's direction ($\sin\theta \cos\varphi$, $\sin\theta \sin\varphi$, $\cos\theta$) and the compartment's position vector (x,y,z), representing the projection of the compartment's position vector of (x,y,z) onto the direction of the electric field.

The minus sign in the equation is due to the electric field being positive towards the negative electrode, such that a positive dislocation in the direction of the field is always associated with a lower potential.

After calculating the spatial distribution of potentials, these are multiplied by a time-dependent waveform to perform stimulation.

- Point Electrode Stimulation

The toolbox can also calculate the spatial component of the extracellular potentials through point-electrode stimulation. Point electrodes serve as idealized sources of stimulation to study neuronal response to external electric fields. Unlike physical electrodes, which have finite size and spatial extent, point electrodes are treated as infinitesimally small sources of current or voltage, allowing for mathematically tractable simulations.

Despite advantages, point electrodes are associated with reduced physical realism, ignore tissue-interface effects and assume that the medium is homogenous(Aberra et al., 2018; Mirzakhalili et al., 2020).

For a microelectrode delivering current *I* centered at (x_0,y_0,z_0), the extracellular potential V_e was calculated at the location of each compartment (x,y,z) based on the equation:

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \tag{3}$$

$$Ve(x, y, z) = \frac{0.001}{4\pi\sigma r} \tag{4}$$

Where σ is the conductivity in a homogenous, isotropic medium (default is 0.276 S/m)(Aberra et al., 2018).

This is calculated for a current amplitude of 1 μA, and then divided by σ x r to obtain the potentials in mV that may then be multiplied by a factor to obtain the results of stimulation by different current amplitudes.

Temporal component (Stimulation waveform)

The stimulation waveform for AM-stimulation is constructed so that the amplitude oscillates between 1 and 1 – m, making interpretation of results more intuitive. ϕ was chosen as to align the modulation with the carrier signal, such that both are 0 in the beginning of stimulation.

$$m(t) = \frac{m\,[\sin(2\pi\, f_m\, t + \phi) + 1]}{2} + (1 - m) \qquad 5)$$

$$c(t) = \sin(2\pi\, f_c\, t) \tag{6}$$

Additionally, to attempt to mitigate the onset response, we added a ramping function to the stimulation amplitude, which is defined by equation (6):

$$r(t) = \begin{cases} \left(1 - e^{\frac{-t}{\tau}}\right), & t \leq ramp_{dur} \\ 1, t > ramp_{dur} \end{cases} \text{, where } \tau = \frac{ramp_{dur}}{3} \tag{7}$$

The full stimulation waveform is thus given by equation (11):

$$AM_{stim}(t) = A\, c(t) m(t) r(t) \tag{8}$$

At each timestep and at the center location of each axon compartment, the total extracellular voltage due to AM-stimulation was calculated using equation (6) and applied to the model neurons using NEURON's extracellular mechanism(Carnevale & Hines, 2006).

Instructions for using the toolbox

Interaction with the toolbox for running simulations of neuronal response to AM-stimulation can be done only through the Python wrapper scripts, through modifying some parameters.

Additionally, and similarly to the “runs.xlsx” and the example batch files, batch running can be implemented, to enable large parameter sweeps and running a massive amount of simulations.

1. Setting up the toolbox

After cloning the repository and unzipping, go into the “mechanisms” folder and run the command “*mknrndll*” (Windows), and copy the file “nrnmech.dll” into the main folder.

If you are working on Linux, first copy all the mod files inside “mechanisms” to the main directory and then run “nrnivmodl”.

After this, the mechanisms have been compiled and you are all set to start simulating neuronal response to stimulation.

2. Main Wrapper Files

There are a couple of main wrapper files, that have to be considered:

Simulation of neuronal response to specific stimulation conditions:

“run_steadystate.py” – basic script to obtain the resting state before the stimulation

“run_main.py” – the basic script that runs one stimulation with the specified parameters

“run_parser_multi.py” –batch running script compatible with parallel simulations

Calculating electric field strength threshold for rhythmic firing:

“run_ss_threshold.py” – basic script to obtain the resting state before the stimulation

“run_threshold.py” – the basic script that calculates the threshold for stimulation with some specific parameters

“run_multi_threshold.py” – batch running script compatible with parallel simulations

The main stimulation method that was the focus of the toolbox - uniform-field stimulation with AM-stimulation - is the standard, but both toolboxes are also prepared for point-electrode stimulation and for different stimulation waveforms described in “stim.py”.

Simulation of neuronal response to AM-stimulation

- **Obtaining Resting State**

Before running simulations of AM-stimulation, the neuron's resting state needs to be obtained, because it is loaded as the initial state for every run. This allows membrane potential to stabilize to the resting state while avoiding the computational time associated with simulating long resting periods for each run. It is especially important when working with low electric field amplitudes, which have a less pronounced effect on the membrane potential.
This is done through running "run_steadystate.py" for the selected neuron model, specified through "cell_id".

The script computes the evolution of membrane potentials and ionic currents without stimulation during the specified simulation time, as they drift towards the steady state. As it finishes, it evaluates whether "steady state" was reached. Steady state is defined as being reached when the difference between the membrane potential at $time = simtime$ and $time = simtime - time\_before\ ms$ (default is simtime-1000) is lower than a maximum variation value (default of 1^{-7} mV) for every compartment.

When it is reached, a binary file containing the state of every variable at the end of the stimulation time is saved through NEURON's SaveState method, to be read by following stimulation runs. If it is not reached, the simulation must be run again, for a longer simulation time, or relaxing the maximum variation value. The default value is very low since the focus of our work was subthreshold modulation, which is associated with low electric field amplitudes. A higher value used beforehand (1^{-3} mV) led to most of the observed variation in membrane potential during stimulation with 1 V/m fields being due to stabilization of membrane potential.

This can, in principle, be run only once for each cell, since the spatial component of the electric field can be changed between runs without affecting the steady state.

Files obtained:

- Plot of **membrane potential** over time
- **Parameters json file**, containing several simulation parameters, namely the simulation timestep(dt), temperature, total time(simtime), initial membrane potential (v_init), maximum difference within the same compartment (max_dif), maximum variation value (threshold) and whether steady state was reached (reached_ss=True or False).
- **Voltages csv file**, containing membrane potentials at $time = simtime$ and $time = simtime - interval\ ms$

If the resting state was reached, the binary file is saved containing every variable in the simulation at the final timepoint.

- **Running Simulations of AM-stimulation**

After that, through running “run_main.py” or using a batch file to run the “run_parser_multi.py” file, the stimulation of model cells can be simulated. This script runs several processes that initialize the cell and restore the steady state, set the stimulation parameters and run the simulation until $time = simtime$, recording, for every timestep, the evolution of membrane potential at all cell compartments.

Important Parameters:

There are several important parameters that must be specified for each run. Most of these are common to both toolboxes, but there are some that are specific to one of them.

Simulation Parameters:

Simulation time (simtime – ms) – the time over which the neuronal membrane dynamics will be calculated.

Integration timestep (dt-ms) – Integration timestep for fadvance().This value should be chosen so as to have a sufficient number of time samples during the stimulation pulses. A very small value for dt will give very accurate results, but at the price of an increased computation time. This should be adequate to the stimulation conditions, and since the toolbox is intended for kHz stimulation, it should be kept considerably low.   To determine a good value of dt, compute the solution of the cable equation for dt and dt/10 and test whether the relative difference of the membrane potential (Vm) at a given position (for instance, the location of the largest value of Vm) is smaller than 1%. If this is not the case, divide dt by 10 until this criterion is satisfied (Joucla et al., 2014).

Temperature (Celsius) – Simulation temperature. Since the ion channel dynamics are temperature-dependent, this should be kept at the temperature for which the ion channel kinetics were fit (37ºC for DT, 36ºC for ST). If a different temperature is used, unexpected dynamics may appear.

cell_id – selects the cell to initialize (between 1 and 5 for ST and 1 and 25 for DT).

Cell_id	ST	Cell_id	DT
1	Fast Spiking	1-5	L1 NGC-DA
2	Intrinsic Bursting	6-10	L2/3 PC
3	Repetitive Bursting	11-15	L4 LBC
4	Low Threshold	16-20	L5 TTPC
5	Regular Spiking	21-25	L6 TPC

Each cell type in DT contains 5 clones, to generate morphological diversity by adding random variation to the branch lengths and rotations of the exemplar model(Erö et al., 2018).

More about the different neuron models is available in the original papers (ST-(Pospischil et al., 2008)) and DT-(Aberra et al., 2018; Erö et al., 2018; Markram et al., 2015; Ramaswamy et al., 2015).

Variable (var) – string that indicates the variable we want to study, as a way to organize the data. The toolbox is prepared for "cfreq", "modfreq", "depth", "theta" and "phi". The secondary variable is by default the amplitude (or electric field strength), but this can be changed through modifying "saveparams" in "savedata.py".

Stimulation Parameters:

- Uniform Electric-field or Point Electrode Stimulation:

ufield: True for uniform electric field and False for Point electrode.

Direction of the Electric-field:

The membrane potentials at each neuronal compartment (mV) are calculated according to equation 2, for an unitary field (1 V/m)

Theta (theta-degrees) – Polar angle (between 0-180º) of the electric field direction. The z axis is aligned with the cell's main axon.

Phi (phi-degrees) = Azimuthal angle (between 0-360º) of rotation around the cell's main axon.

Reference point (ST) (x,y,z - μm) – zero potential point

Point electrode:

coordinates: $[x_0,y_0,z_0]$, - μm, position of the stimulation electrode.

Conductance (rho) – S/ μm, of the medium

AM-stimulation

Amplitude(amp) – Multiplier, to simulate different electric field strengths or current amplitudes. Since for uniform electric field, the spatial distribution was calculated for 1 V/m, the amplitude corresponds to the field strength in V/m. For point electrode stimulation, it is the current in μA.

Carrier frequency (freq-Hz) – Carrier frequency of the AM stimulation wave. It should be lower than 100 kHZ, for the quasi-static approximation to be valid.

Modulation Frequency (**modfreq-Hz) –** Low-frequency stimulation waveform.

Modulation Depth (depth – 0 to 1) – depth of amplitude modulation, where 0 is minimum (unmodulated) and 1 is maximum (full modulation, with amplitude from 0 – amp).

Delay (ton – ms) – Time until stimulation starts.

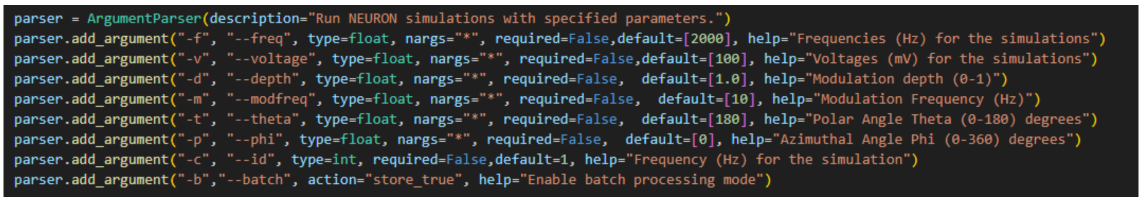
Duration (dur -ms) – Duration of stimulation.

Ramping stimulation (ramp) – True if you want to ramp stimulation amplitude to reduce onset effects, False if you do not

Ramp duration (ramp_duration – ms) – Time when stimulation amplitude is increasing (lower than maximum).

Time constant of exponential ramping amplitude (tau-ms) – if None, the ramping is linear, if 0, it is calculated as tau=ramp_duration/3, if another value, tau=value.

Batch Running

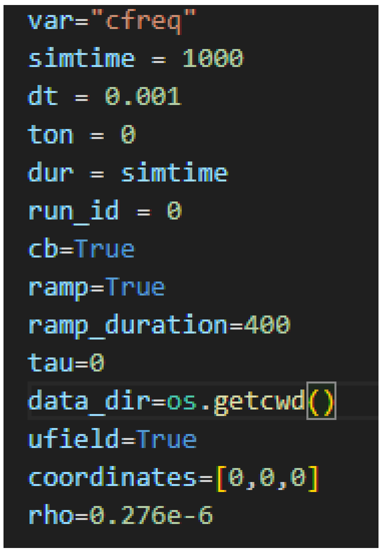


```
parser = ArgumentParser(description="Run NEURON simulations with specified parameters.")
parser.add_argument("-f", "--freq", type=float, nargs="*", required=False,default=[2000], help="Frequencies (Hz) for the simulations")
parser.add_argument("-v", "--voltage", type=float, nargs="*", required=False,default=[100], help="Voltages (mV) for the simulations")
parser.add_argument("-d", "--depth", type=float, nargs="*", required=False,  default=[1.0], help="Modulation depth (0-1)")
parser.add_argument("-m", "--modfreq", type=float, nargs="*", required=False,  default=[10], help="Modulation Frequency (Hz)")
parser.add_argument("-t", "--theta", type=float, nargs="*", required=False,  default=[180], help="Polar Angle Theta (0-180) degrees")
parser.add_argument("-p", "--phi", type=float, nargs="*", required=False,  default=[0], help="Azimuthal Angle Phi (0-360) degrees")
parser.add_argument("-c", "--id", type=int, required=False,default=1, help="Frequency (Hz) for the simulation")
parser.add_argument("-b","--batch", action="store_true", help="Enable batch processing mode")
```

This is what the commands to do batch parallel runs look like. It is possible to specify a list of carrier frequencies (-f), electric field strengths (-v), modulation frequencies (-m), depths (-d), thetas (-t), phis (-p) and one cell id (-c) to be simulated. Parallel running is specified through (-b).

python.exe run_parser_multi.py -f 10000 -v 1 2 3 -m 10 -d 1 -t 180 -p 0 -c 1 -b >> logs\10000.log 2>&1

If the amount of simulations that are run in parallel (number of parameter combinations) is larger than the number of CPUs available, a value error is raised warning that lesser simulations must be run at once.



```
var="cfreq"
simtime = 1000
dt = 0.001
ton = 0
dur = simtime
run_id = 0
cb=True
ramp=True
ramp_duration=400
tau=0
data_dir=os.getcwd()
ufield=True
coordinates=[0,0,0]
rho=0.276e-6
```

The rest of the parameters are specified in the wrapper script, and can be changed with ease.

Files obtained after simulation

- Plot of membrane potential over time in the soma, in one axon compartment and one dendritic compartment.
- Plot of the stimulation waveform.
- “run_voltages.h5” – containing the membrane potential at every compartment and over every simulation timestep. It is organized in datasets, with one dataset (“voltages”) having a size of (num_timesteps,num_segments) and another “time” as an unidimensional vector with all the time values.
- params.json file – Simulation parameters for debugging.
- Vrec.csv – Containing the local field potentials produced by the neuron model and calculated through the extracellular and xtra mechanism.

Calculating Field Strength Threshold

Another important functionality of the toolbox is threshold detection. This detects the “**minimum electric field amplitude needed to induce firing at the modulation frequency**” for the stimulation parameters considered(Wang et al., 2023).

This can easily be done by running the scripts “run_ss_threshold.py”, which has a similar function to “run_steadystate.py”, focused on obtaining the resting state for the cell before stimulation, followed by “run_threshold.py” or “run_multi_threshold.py” (with a batch file), based on the threshold detection script described in (https://modeldb.science/239006) (Patrick Reilly, 2016), and also taking inspiration from (Caldas-Martinez et al., 2024; Wang et al., 2023), such that a neuron is only considered to be spiking if the frequency of action potentials corresponds (at least) to the modulation frequency. Additionally, to avoid onset response playing too big of a role, a ramping stimulation amplitude is implemented.

We consider the neuron fires when it passes a certain *threshold* membrane potential value (binary threshold crossing), which may be defined in the script, and takes the value 0 mV if undefined, and monitor either every single compartment (if record_all is True), or only the center of the soma (if record_all is False), through adding either NEURON action potential counters or NetCon objects (which are also able to record spike times).

If the NetCon method is chosen (nc=True), we included a method to ensure that each spike burst does not get counted more than once, through implementing a simple criterion based on the inter-spike interval (ISI) to define whether spikes are part of a burst. This is defined in equation (12), such that the action potential only gets counted towards the number of spikes ($spike_{num}$) if the ISI to the previous spike is over 5 ms.

$$spiketime_i - spiketime_{i-1} > 5\ ms \tag{12}$$

Following, we defined the minimum number of spikes(min_{spikes}) for the threshold to have been reached, based on the simulation time ($simtime$), duration of amplitude ramping ($ramp_{duration}$) and modulation frequency (f_m). The threshold for rhythmic firing is considered to have been reached if $spike_{num}$ is at least equal to min_{spikes}.

$$spike_{num} \geq min_{spikes}$$

$$min_{spikes} = \left(\frac{simtime - (ramp_{duration} + 100)}{1000}\right) \times f_m$$

In the calculation of min_{spikes}, it is important to point out that it only considers the time after the stimulation has stabilized.

The function does successive runs to look to find the lowest electric-field value where rhythmic firing at the modulation frequency starts happening. It is associated with two different phases:

1. Finding upper (high) and lower (low) boundaries where spiking behavior changes. High corresponds, by definition, to the lowest electric field value associated with rhythmic firing, while low is the highest electric-field value that doesn't induce firing. These are initialized with values 1e6 and 0, respectively.

The stimulation starts with a certain amplitude and, if this is enough to induce firing, high takes the value of amplitude, and the amplitude of the next run is halved. If this is not enough to induce firing, low takes the value of the amplitude, and the amplitude is doubled. This goes on until both high and low have different values from their initialization, meaning that both a lower and upper bound have been found, or until amplitude is over 1e6, at which point it is considered that rhythmic spiking cannot be reached for the chosen parameters.

2. After defining the upper and lower bounds, the second phase is a simple binary search until high and low are within a defined tolerance.

   To start the search, amplitude is calculated through $amp = (high + low)/2$, and the default relative error is defined as being $\varepsilon = 0.01 \times amp$, but may be changed in the script according to the requirements of the study.

   The simulation is run subsequently while $(high - low) > \varepsilon$, and the low and high values are assigned similarly to the first phase. After each run, the amplitude is calculated through $amp = (high + low)/2$ again, and the ε also adequately assigned. After we reach a value compatible with the tolerance, the simulation is run for one last time, recording the voltage over time for every monitored compartment (either every compartment or only the soma) and the number of spikes registered by every action potential counter (APC), or the spike times and spike number (NetCon), enabling subsequent debugging and further analysis.

   The minimum electric field amplitude needed for firing at the modulation frequency ("threshold") is saved to a csv file, together with the key parameter that is being evaluated, expressed by the *variable*.

   By running "debug_threshold.py", it is possible to plot the voltage over time for the compartment with the highest number of spikes and also with the highest peak membrane potential (over time) overall.

Important Parameters:

Most parameters are similar to the simulation of AM stimulation.

Nc – True if spike times or excluding bursts is important, False otherwise.

Record_all – True if you want to record every single cellular compartment, False if you only want to record the soma.

Threshold (thresh – mV) – Membrane potential that we consider as the spiking threshold.

Amplitude (amp – V/m or μA) – Initial amplitude of stimulation. Should be close to the expected value to reduce the number of runs needed to find “high” and “low” for the binary search.

Files obtained after simulation

- “run_voltages.h5” – containing the membrane potential at every monitored compartment and over every simulation timestep. It is organized in datasets, with one dataset (“voltages”) having a size of (num_timesteps,num_segments), another “time” as an unidimensional vector with all the time values, and another “is_xtra” with the stimulation current over time.
- Spike_number.json – number of spikes recorded in every segment.
- Spike_times.json – only for the NetCon method, saves the timepoints at which the membrane potential crossed the spiking threshold.

Filing Structure

The filing structure of the “data” directory is simple.

```
/data
  ├── /cell_id
    ├── /var
      ├── /value
        ├── /E_field
```

The main variable that is in value is the variable we want to study, and the secondary one is by default electric field strength, but this can be changed for any other variable by simply modifying “savedata.py” accordingly.