NeMo-TMS (Neuron Modeling for TMS) Tutorial

Correspondence: aopitz[at]umn.edu (Prof. Alexander Opitz)

Table of Contents

Section 1. Short introduction and overview 2
Section 2. Software requirements 2
Section 3. Using NeMo-TMS 4
3.1. **Compilation of .mod files** 4
3.2. **Model generation** 5
3.3. **Electric field modeling** 6
3.4. **Export neuron segment location** 6
3.5. **Electric field coupling to the neuron model** 6
3.6. **TMS waveform Generation** 9
3.7. **Running the NEURON simulation** 10
3.8. **Running Calcium modeling** 11
3.9. **Results visualization** 12
Section 4. NeMo-TMS file structure 14
Contents of NeMo-TMS after downloading from GitHub 14
Contents of NeMo-TMS after generating neuron model(s) 15
Contents of the model folder 15
Contents of the Code folder 15
Contents of the Results folder 16
Section 5. Advanced parameters for calcium simulation 17

This symbol means there is important information given.

This symbol means some tips are given for a better use of the toolbox.

For a better visibility, paths/files will be shown in **orange**, and functions/commands will be shown in **green**.

Section 1. Short introduction and overview

Neuron Modeling for TMS (NeMo-TMS) toolbox is for multi-scale modeling of the effects of Transcranial Magnetic Stimulation on single neuron activity. The instructions and open-source codes provided here enable users with a different level of expertise to investigate the neuronal behavior under TMS. This paradigm incorporates modeling at three scales:

- TMS-induced electric field calculation at the macroscopic/mesoscopic scale
- Simulation of neuronal activity under the external electric field from the previous step
- Simulation of subcellular calcium concentrations based on the membrane voltages calculated in the previous step.

The procedures for running simulations at each scale and the intermediate steps are given in the instructions. This pipeline has been tested on **Windows 10**, and **Linux (Ubuntu 18.04/20.04)**. We have tested all the steps except the model generation (step 1) on **macOS Catalina**. However, the codes are developed to run on the native operating system's configurations and may possibly not work if there are any modifications (for example, using other forms of shell).

To cite the toolbox or for more information, please use the following reference:

Shirinpour et al., 2020. Multi-scale Modeling Toolbox for Single Neuron and Subcellular Activity under (repetitive) Transcranial Magnetic Stimulation. bioRxiv, https://doi.org/10.1101/2020.09.23.310219

Throughout this document, neuron refers to the biological neural cells, while NEURON refers to the NEURON simulation environment used for the computational modeling of neurons.

Section 2. Software requirements

Several software packages and external toolboxes are required before using this modeling toolbox. Here you can find a short description and explanations about their functions and how to download them.

MATLAB (Tested on versions 2019a & 2020a)

MATLAB is a programming language and a numeric computing environment ideally made for scientific computation. You can download it from here: https://www.mathworks.com/downloads/. Because MATLAB is a licensed product, it is not free. But most universities provide free licenses for academic use. If you want to get started with MATLAB, you can go to: https://www.mathworks.com/help/matlab/getting-started-with-matlab.html.

SimNIBS/Gmsh (Tested on version 3.2.2)

SimNIBS is a free and open source software package for the Simulation of Non-invasive Brain Stimulation. SimNIBS allows for realistic simulation of the electric field induced by transcranial magnetic stimulation (TMS) and transcranial electric stimulation (TES). You can download SimNIBS here: https://simnibs.github.io/simnibs/build/html/installation/simnibs_installer.html. If you are a beginner, you can follow this tutorial: https://simnibs.github.io/simnibs/build/html/tutorial/tutorial.html.

SimNIBS contains some MATLAB functions for interactivity. These MATLAB functions are used in NeMo-TMS, so be sure to add the SimNIBS path to MATLAB. To do so, go to home>Set Path in the MATLAB menu and select Add Folder. Browse to SimNIBS Directory\matlab and click on Save. For more information: https://simnibs.github.io/simnibs/build/html/tutorial/scripting.html#matlab

Note 1: If you only want to use a spatially uniform electric field, instead of realistic distribution of the electric fields in your macroscopic model (head, *in vitro*, etc), you can skip installing SimNIBS/Gmsh.

Note 2: Gmsh is a powerful program for 3D visualization of mesh files. Since Gmsh is distributed together with SimNIBS, there is no need to install it separately.

NEURON (Tested on versions 7.5 & 7.8.1)

NEURON is a simulation environment for modeling individual and networks of neurons. You can download NEURON for free from here: https://www.neuron.yale.edu/neuron/download. For Windows, install NEURON in the default directory, and select 'Set DOS environment' during installation. Although, all the NEURON scripts are generated automatically in NeMo-TMS, you can find the tutorials here: https://www.neuron.yale.edu/neuron/docs
For more examples visit: https://www.neuron.yale.edu/neuron/docs/neuron-course-exercises.

UG4

UG4 is a free simulation software for solving differential equations on unstructured finite element grids. UG4 is used in NeMo-TMS for running the calcium simulations. For manual installation of UG4, follow the instructions here: https://github.com/UG4/ughub. Alternatively, you can download the static build for your operating system: http://doi.org/10.5281/zenodo.3995132
Make sure UG4-Build\ug4\bin\ugshell is set as executable for Linux and macOS.

TREES toolbox / T2N

TREES toolbox is a MATLAB toolbox for automatically reconstructing neuronal branching from microscopy image stacks and to generate synthetic axonal and dendritic trees. You can find more information here: https://www.treestoolbox.org/index.html. T2N is an extension of the TREES toolbox providing an interface between MATLAB and compartmental modeling environment NEURON. For more information on T2N refer to: https://www.treestoolbox.org/T2N.html. These two toolboxes are used to automatically generate the NEURON models based on the desired neuron morphology.

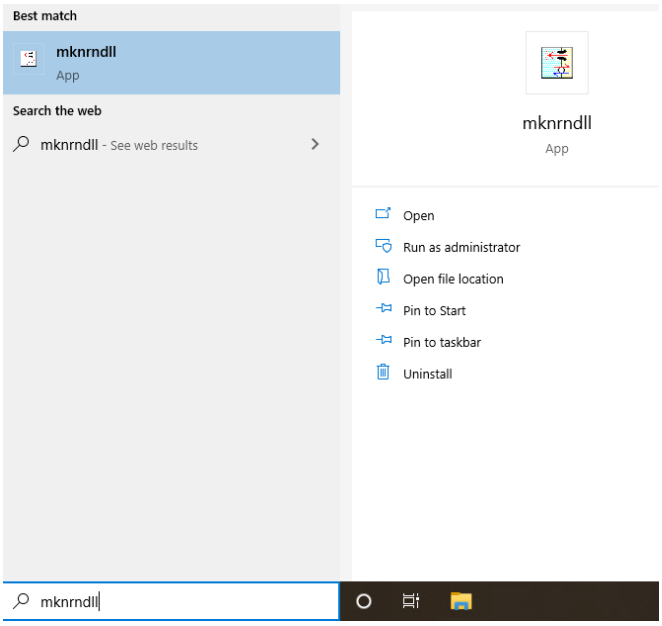
Note: There is no need to download TREES and T2N since they are distributed with NeMo.

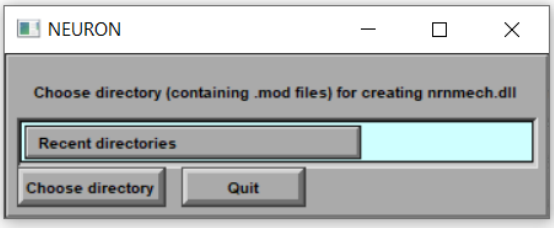
Section 3. Using NeMo-TMS

3.1. Compilation of .mod files

NeMo-TMS toolbox models realistic morphologies of neurons exhibiting various ionic channels dynamics. These dynamic mechanisms are implemented in multiple .mod files in the following folders Model_Generation\Jarsky_files\lib_mech and Model_Generation\Aberra_files\lib_mech. You need to compile these files before generating the models. You only need to do this once on each computer. For more information on .mod files and how to use them, see: https://www.neuron.yale.edu/phpBB/viewtopic.php?t=3263. To compile the .mod files, follow these instructions depending on your operating system:

- **Windows:** In the Windows search bar, search for mknrndll and select the icon. A NEURON window should appear where you can choose the directory that includes the .mod files and compile them.





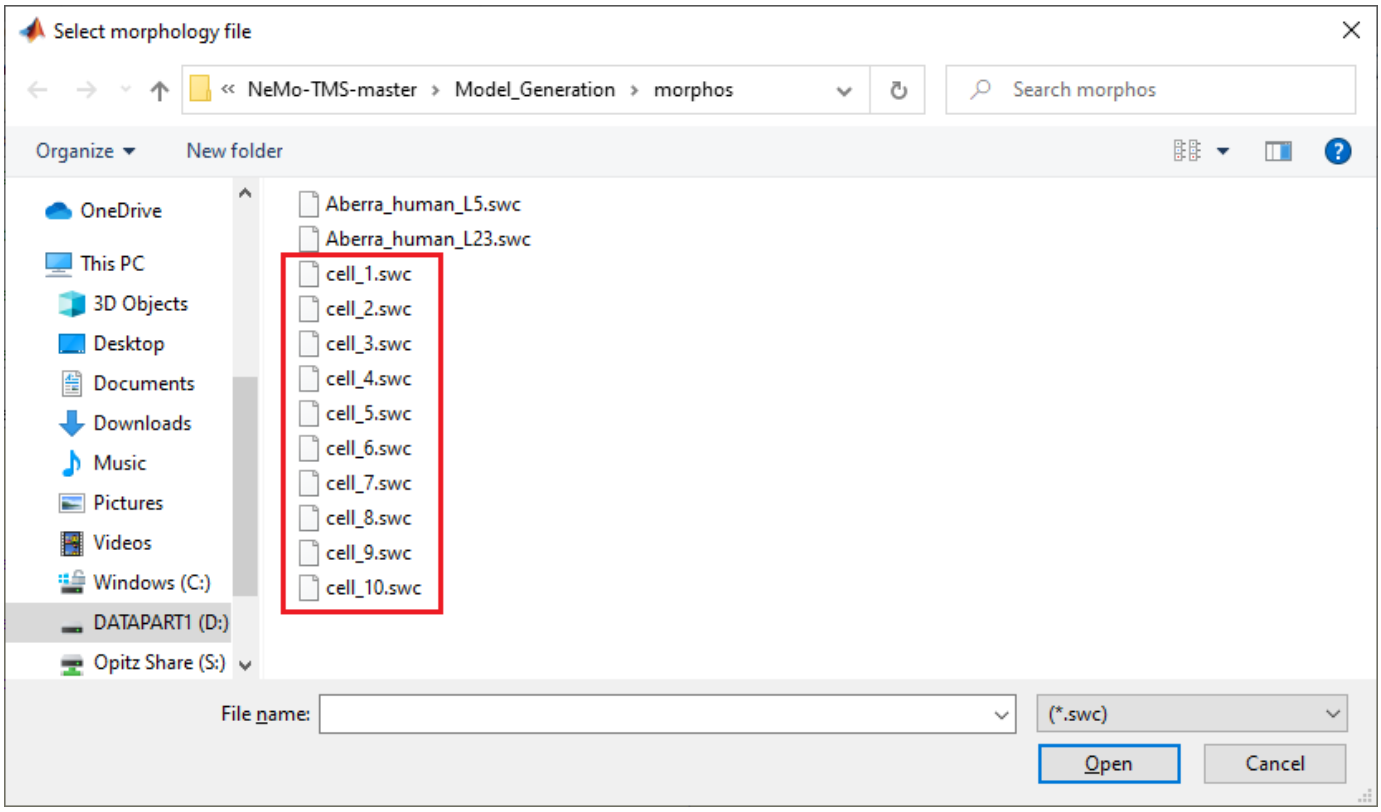
- **Linux:** In the terminal, use cd command to change your directory to where the directory that holds the .mod files. Execute the nrnivmodl command.
- **macOS:** Drag and drop the directory that holds the .mod files onto the mknrndll icon.

Note: Compile the .mod files for both directories mentioned above.

3.2. Model generation

In this subsection, you are going to generate the model for each model type.

Jarsky CA1 pyramidal cell: Change your MATLAB directory to Model_Generation folder and run Jarsky_model() in MATLAB. The prompt asks for the desired input morphology file (.swc). You can use the sample morphology files provided in Model_Generation\morphos\cell_#.swc as shown below:



Aberra cortical pyramidal cell: Change your MATLAB directory to Model_Generation folder and run Aberra_L5_model() for the L5 pyramidal cell or Aberra_L23_model() for the L2/3 pyramidal cell in MATLAB with no input arguments.

Note 1: The input morphology must be in standard SWC format, with soma as region 1, axon as region 2, basal dendrites as region 3, and apical dendrite as region 4. The units should be in µm.

Note 2: In both cases, you can choose the name of your neuron model and a folder will be created in Models which includes all the corresponding scripts and data for this neuron model. All the next steps are model-specific and therefore operate within this folder.

Note 3: Currently this step is not supported in macOS. However, models can be generated on other computers and copied to the macOS computer. In this case, the .mod files for the generated model (Models\model_name\lib_mech) should be recompiled in the new computer.

3.3. Electric field modeling

If you are interested in using a spatially uniform electric field, skip to **TMS waveform Generation**.

In this step the aim is to calculate the electric fields generated by TMS at the macro- and mesoscopic scale. This includes computing the spatial distribution and time course of the TMS electric field. Since the stimulation frequency is relatively low, we can use the quasi-static approximation to separate the spatial and temporal components of the electric field. Here, we only focus on the spatial distribution of the electric field since the temporal component is calculated in a different step. To this end, we numerically calculate the distribution of electric fields in any sophisticated geometry using Finite Element Method (FEM) modeling. For NeMo-TMS, we use the open-source software SimNIBS which is a versatile simulation platform that can simulate TMS electric fields for various geometries and a variety of TMS coils. For a quick start on how to use SimNIBS, visit:

https://simnibs.github.io/simnibs/build/html/tutorial/gui.html. you can skip the section describing setting up a tDCS Simulation.

For more details, refer to the full tutorial: https://simnibs.github.io/simnibs/build/html/tutorial/tutorial.html

You can use prebuilt FEM models or generate your own models to use in SimNIBS. You can find example human head models in https://simnibs.github.io/simnibs/build/html/dataset.html or a high-resolution human head FEM model in https://zenodo.org/record/5209082 and some animal and *in vitro* models in the https://zenodo.org/record/4009465. Alternatively, you can generate your own FEM models as explained here: https://simnibs.github.io/simnibs/build/html/tutorial/head_meshing.html

3.4. Export neuron segment location

Export the segment coordinates of the neuron model by running Code\NEURON\save_locations.hoc in NEURON environment. These data are used in multiple subsequent steps.

You can run NEURON scripts by double-clicking on the file on Windows or running the command nrniv file_name.hoc in the terminal on macOS and Linux (cd to the correct directory).

Note: For the Aberra_L23_model, the stack size must be increased to simulate the complicated myelinated axon, otherwise, the simulation fails. To do this on windows, right-click on Code\NEURON\save_locations.ps1 and select 'Run with PowerShell'. On mac/Linux, run Code\NEURON\save_locations.sh in terminal. Alternatively, run this command in terminal:
nrniv -NSTACK 100000 -NFRAME 20000 -Py_NoSiteFlag save_locations.hoc

3.5. Electric field coupling to the neuron model

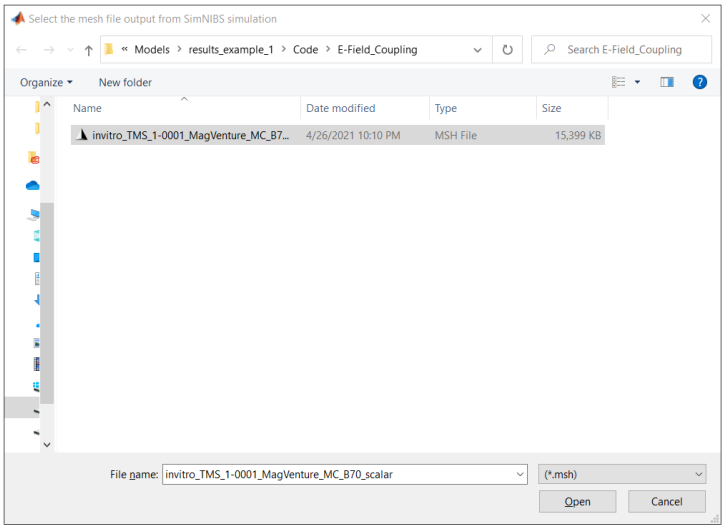
Since NEURON cannot directly work with the calculated electric fields, in this intermediate step, the quasipotentials at the location of the neuron model are generated based on the electric fields. These quasipotentials can then be imported in the NEURON environment as the extracellular potentials for

the simulation. In this step, we couple the SimNIBS simulation electric fields with the NEURON model by writing the quasipotentials in a file which will be imported in the NEURON simulation later.

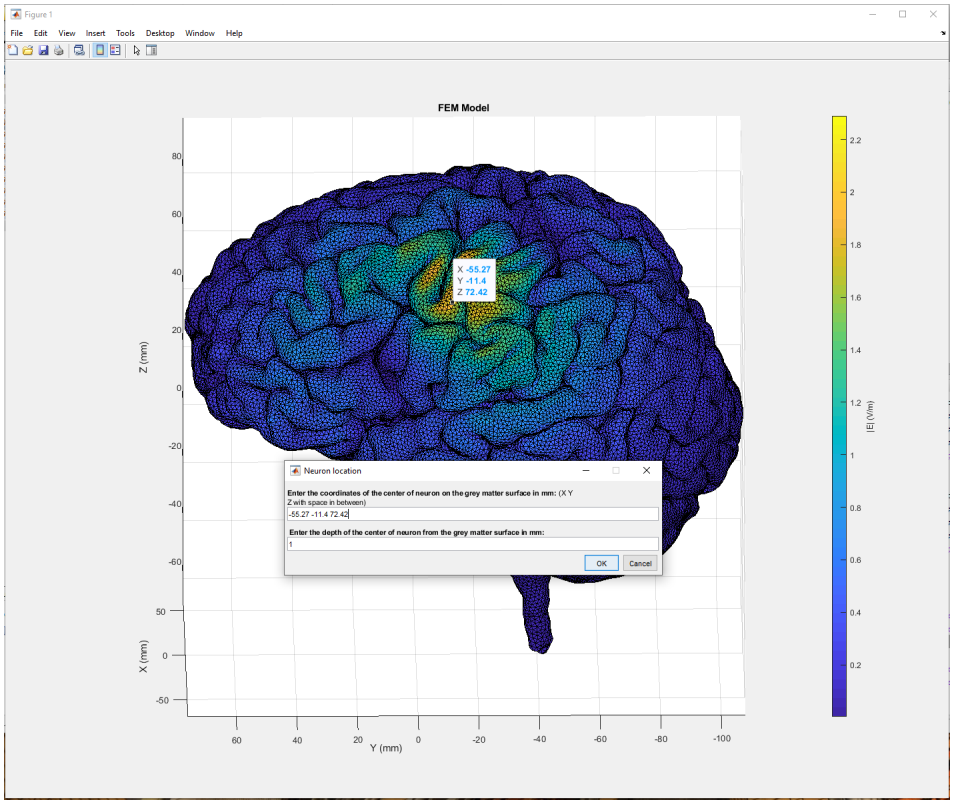
Note: Before running this step, be sure to add SimNIBS MATLAB functions to the MATLAB set path as explained earlier.

Change your MATLAB directory to Code\E-Field coupling folder and run couple_gui() in MATLAB. The Graphical User Interface (GUI) walks you through the procedure step by. In summary, the following steps will run:

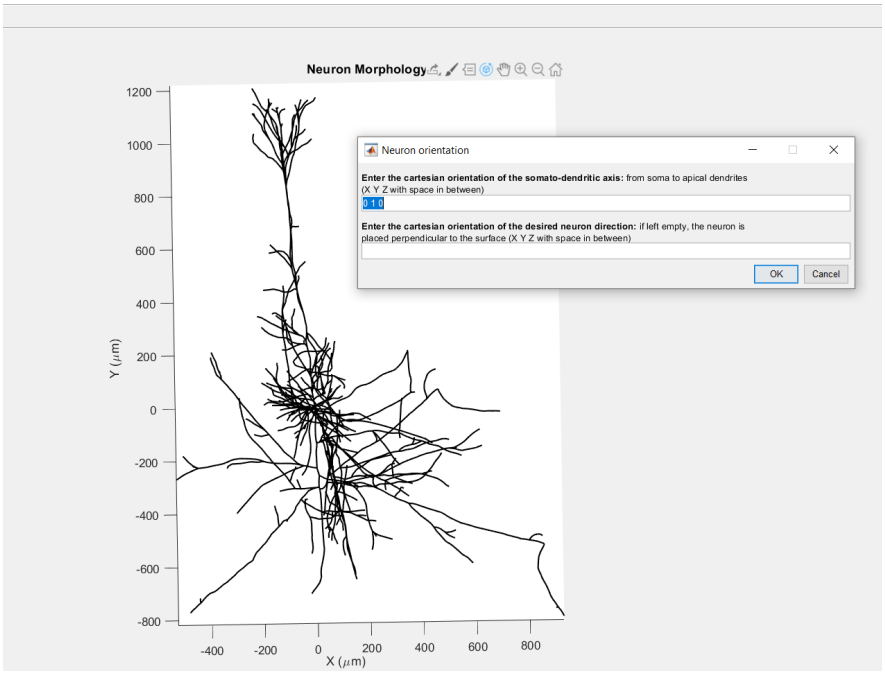
1. Load the simulated SimNIBS mesh file generated in the earlier step. Make sure to use the mesh file resulted from the SimNIBS simulation, and not the raw mesh file you used as the input to SimNIBS.



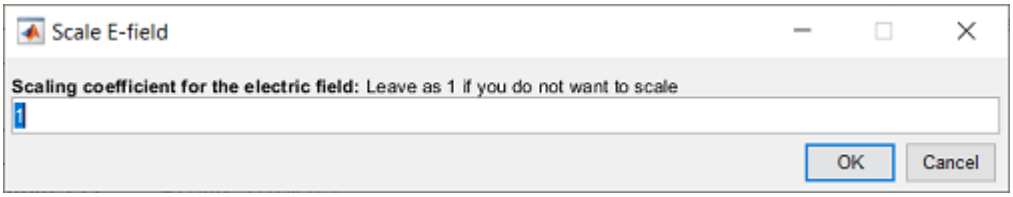
2. The selected mesh file with the electric field magnitude is plotted. Enter the desired location and depth (from the grey matter surface) for the neuron's center (soma) from the user. Set depth as 0 if you wish the neuron to be placed at the exact location you enter.



3. The neuron is plotted based on the exported segment locations from the previous step. Enter the neuron's somato-dendritic axis (mainly defined for pyramidal cells). Leave it to default if this axis is not defined for the neuron. **Optional:** You can also specify the desired orientation of the neuron here. If left blank, the neuron will be placed perpendicular to the grey matter surface by default as this is the typical orientation of pyramidal cells.



4. Scale the electric fields if necessary. Since the magnitude of induced electric fields (and quasipotentials) are proportional to the current rate of change in the TMS coil (dI/dt), you don't need to rerun the SimNIBS simulation to calculate the E-field for a different TMS intensity. Instead, you can run the simulation with the default dI/dt (10^6 *A/s*) and scale the E-field here. **Note:** If you change any other parameter (e.g. Coil position/orientation, head model, ...) you have to rerun the simulation to get the updated E-fields as there is no linear relationship between those parameters and the E-field.

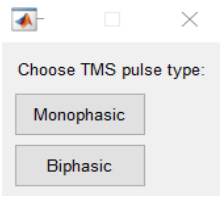


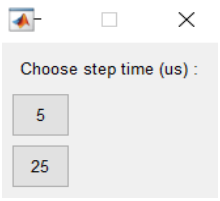
Then, the electric fields are interpolated at the location of the neuron and the quasipotentials are calculated and exported to a file.

For advanced users: You can run this step as a script without the GUI to run batch simulations. Run couple_script('parameter_file.txt') in MATLAB with the location of the parameter file as the argument. The parameter file needs to include all the necessary variables in the correct format (see the parameter file generated by couple_gui()).

3.6. TMS waveform Generation

In this step, the TMS waveform is generated. This corresponds to the temporal component of the electric field in the quasi-static approximation. Select the TMS pulse type (monophasic or biphasic) and the time step of the waveform (5 or 25 µs). The subsequent NEURON and calcium simulations will automatically run at the select time step and the length of time as the waveform generated in this step. First, change your MATLAB directory to Code\TMS_Waveform. Run TMS_Waveform() in MATLAB. A GUI pops where you can choose the TMS pulse type and the time step. Then, enter the number of pulses and the inter-pulse interval (in *ms*) to generate the desired (r)TMS waveform.





Note: The waveforms are normalized and therefore unitless. The amplitude of the electric field is accounted for during calculation of the spatial distribution of the Electric fields.

For advanced users: You can create any custom waveform (e.g. theta burst stimulation or cTMS) by creating the output files with the same format. Make sure you use a time steps of 5 or 25 *μs* in generating the TMS waveforms. You can also change the delay before and after the pulse train by adjusting the values of **delay_start** and **delay_end** in the MATALB scripts (units are in *ms*).

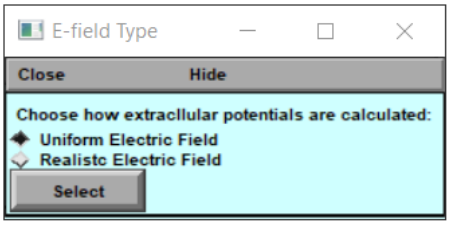
3.7. Running the NEURON simulation

In this section, you will run the NEURON simulation. First, go to Code\NEURON and run GUI_params.hoc in the NEURON environment. Select the desired simulation parameters in the GUI.

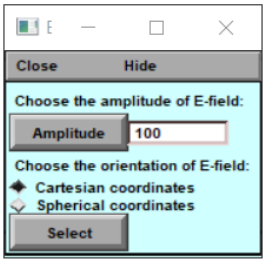
You can run NEURON scripts by double-clicking on the file on Windows or running the command nrniv file_name.hoc in the terminal on macOS and Linux (cd to the correct directory). For NEURON scripts that require graphical interaction with the user (e.g. GUI_params.hoc) use nrngui file_name.hoc in the terminal for macOS and Linux instead.

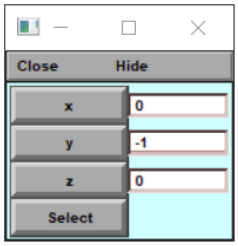
The following parameters need to be selected:

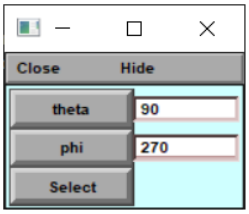
- Select the external electric field type:



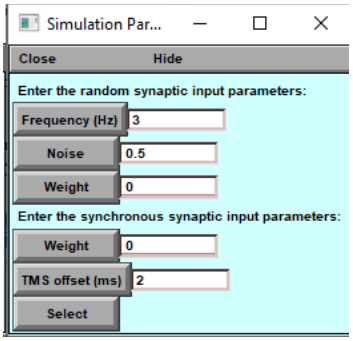
- In the case of spatially realistic electric fields, the quasipotentials file from the coupling step is automatically imported. In the case of spatially uniform electric field, you will be asked to enter the amplitude of the electric field (*V/m*) as well as its orientation (in Cartesian or spherical coordinates). In the spherical coordinates, θ and φ denote the polar and azimuthal angles respectively. The Cartesian coordinates are normalized since the amplitude is entered separately.







- Finally, enter the parameters for the synaptic inputs:
  - Frequency (*Hz*): The average frequency of the random presynaptic spikes
  - Noise: The fractional randomness (between 0 and 1)
  - Weight: Strength of the synaptic connection
  - TMS offset (*ms*): The delay between the synchronous synaptic input and the TMS pulse (positive values mean the synaptic input occur before the TMS pulse).



Afterward, start the simulation by running Code\NEURON\TMS_script.hoc in the NEURON environment. Two plots pop up on the screen. The first one displays the TMS waveform used as the input to the simulation/ The second one plots the voltage of the somatic membrane potential in real-time. These windows will automatically close upon successful completion of the stimulation.

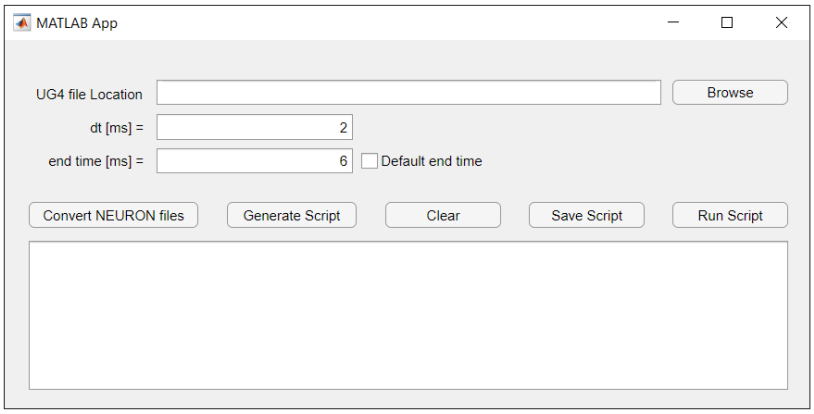
Note: After the simulation finishes, it takes some time to write the large results in files. Do not close the simulation window prematurely, or the output results will be incomplete.

Note: For the Aberra_L23_model, the stack size must be increased to simulate the complicated myelinated axon, otherwise, the simulation fails. To do this on windows, right-click on Code\NEURON\TMS_script.ps1 and select 'Run with PowerShell'. On mac/Linux, run Code\NEURON\TMS_script.sh in terminal. Alternatively, run this command in terminal: nrniv -NSTACK 100000 -NFRAME 20000 -Py_NoSiteFlag TMS_script.hoc

For advanced users: You can manually edit or programmatically generate the simulation parameters, saved in result\NEURON\ params.txt for each simulation. You can scale the electric fields by modifying the value of **TMSAMP** (even in the case of realistic fields). This is especially useful for finding the firing threshold of the neuron.

3.8. Running Calcium modeling

After running the NEURON simulation, the resulting voltage traces are used to run the calcium simulation. You can skip this step if you do not need to run the calcium simulation. First, change your MATLAB directory to Code\Calcium and then run calcium_simulation_setup.mlapp to start the MATLAB GUI for calcium simulation:



There are multiple ways to run .mlapp files. You can use the file_name without the extension in the command prompt, or run('filename.mlapp'). Alternatively, you can open the file in the MATLAB App Designer.

Select the parameters and inputs in the GUI. The first input is the location of the UG4 execution file. Click on 'Browse' and then go to your UG4 folder (which depends on where you installed UG4 or downloaded the static build). Then go to UG4-Build\ug4\bin and select the ugshell file (the GUI remembers this location for future use). Choose the step time and the end time of the calcium simulation. If you want to have the full length of the voltage data, check '**Default end time**'.

Note: The step time for the calcium modeling does not need to be the same as the NEURON simulation.

Since UG4 requires a specific format for the input files, the NEURON voltage traces need to be converted first. To do this, click on '**Convert NEURON files**' button. The conversion can take a long time if the simulation is long. A message is displayed upon successful conversion. Then, click on '**Generate script**' to generate the UG4 script responsible for running the calcium simulations. Advanced users can edit additional parameters in the textbox as explained in **Advanced parameters for calcium simulation**. Click on '**Save Script**'. Finally, start the calcium simulation by clicking on '**Run script**'. This is the most computationally demanding step. The simulations can take a long time.

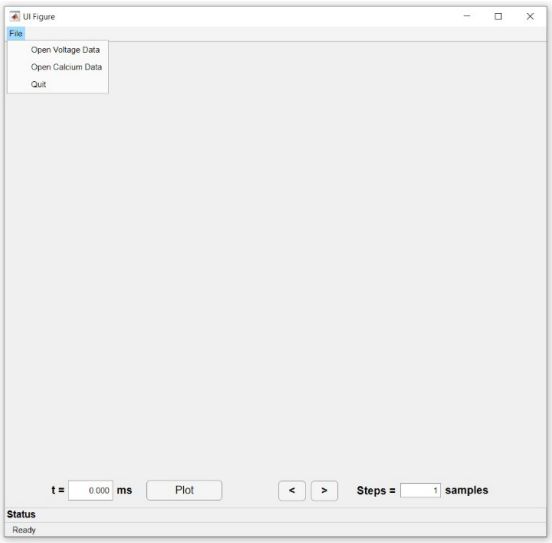
Note 1: You can also run the calcium simulation by running the saved shell script (Results\Calcium\calciumshellscript) in the terminal (powershell in Windows). This way, MATLAB will not be occupied while running the simulation.

Note 2: The '**Clear**' button just clears the generated script in the textbox and does not erase the converted NEURON files.

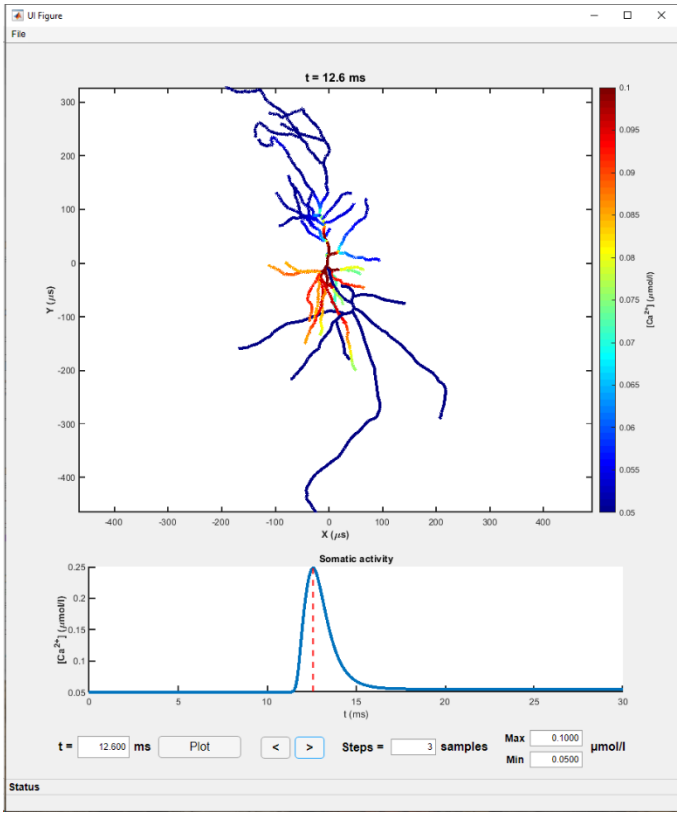
3.9. Results visualization

In the final step, you can visualize the simulation results in a MATLAB GUI. First, change your MATLAB directory to Code\Visualization and run visualization.mlapp in MATLAB.

There are multiple ways to run .mlapp files. You can use the file_name without the extension in the command prompt, or run('filename.mlapp'). You can also right click on file in matlab browser and select run. Alternatively, you can open the file in the MATLAB App Designer.



In the GUI, click on '**File**', and select '**Open Voltage Data**' or '**Open Calcium Data**' to visualize the 3D voltage and calcium results across time respectively. Please be patient while the large data loads. After the data is loaded, the top figure shows the spatial distribution of the membrane voltage/calcium concentrations in the neuron, while the bottom figure shows the temporal pattern of the somatic membrane voltage/calcium concentration. You can choose a specific time instance in the t = ms textbox and clicking on '**Plot**'. You can also go forward or backward (< > arrows) a certain number of steps. Enter the step size in the Steps = samples textbox. At the bottom of the GUI, the status bar indicates when the GUI is busy ('Wait' or 'Loading' appears). You can enter the minimum and maximum values for visualization. All the data beyond the specified range is saturated. This is useful for visualizing the spread of the calcium since most of the calcium is concentrated around the soma.

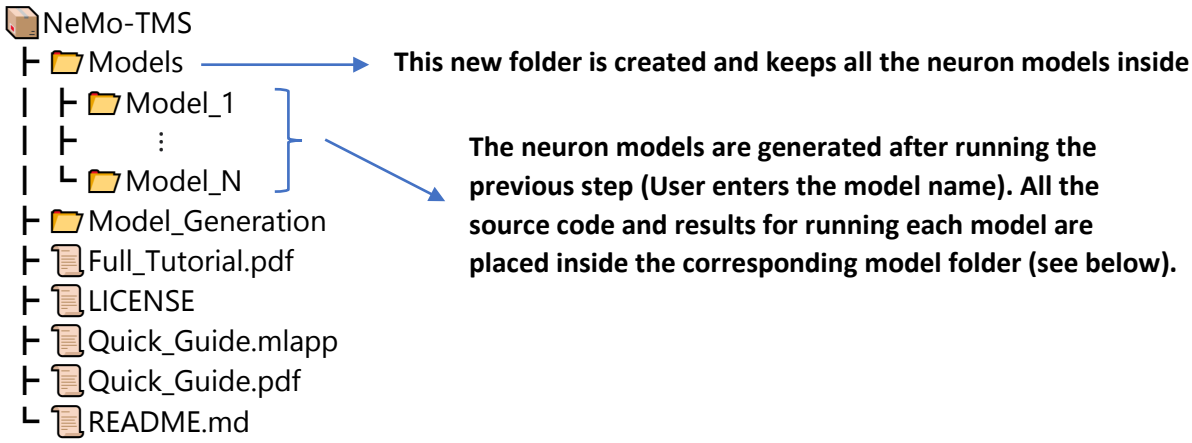


Section 4. NeMo-TMS file structure

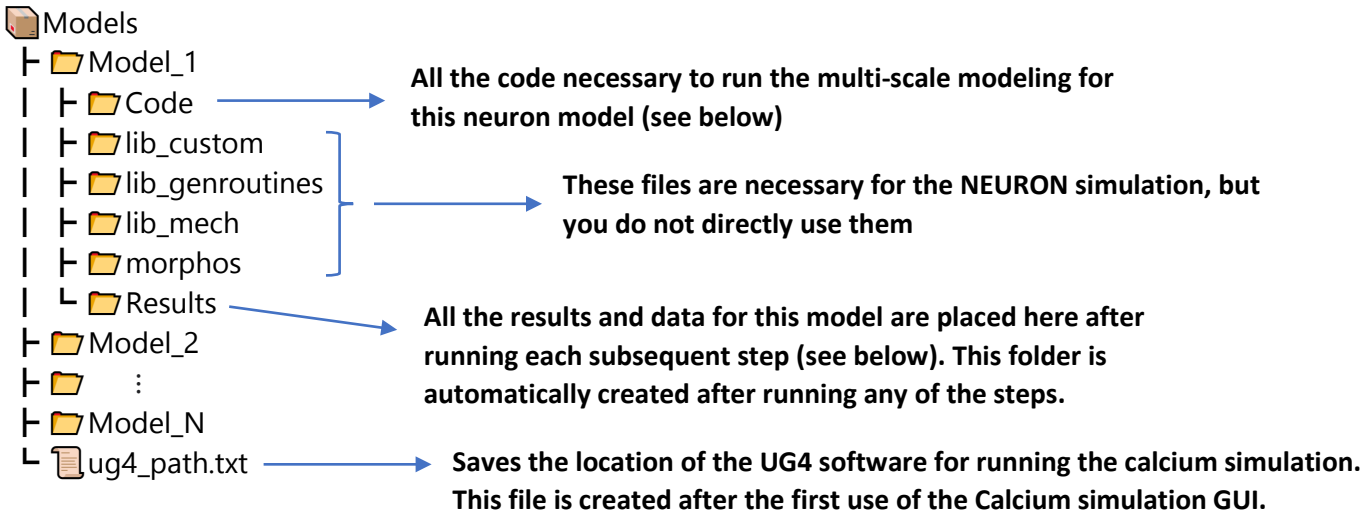
In this section, we briefly discuss the file structure of NeMo-TMS. This can help with finding the code or simulation data. The tree structures below are trimmed to only show the important files.

Contents of NeMo-TMS after downloading from GitHub

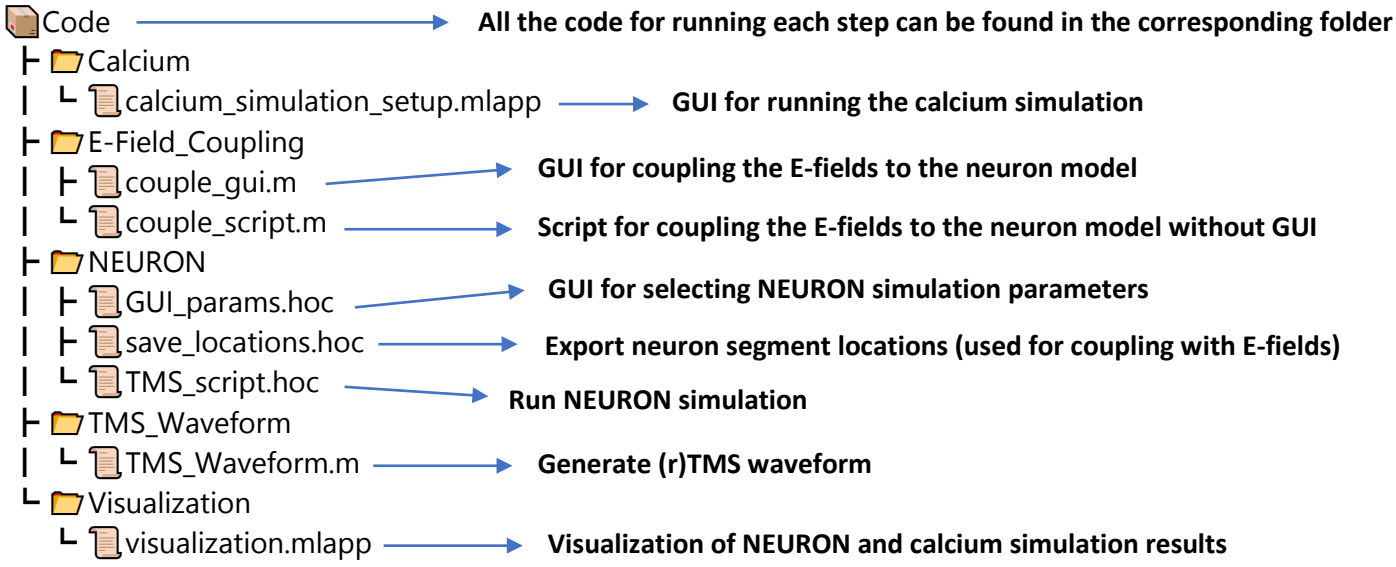
Contents of NeMo-TMS after generating neuron model(s)



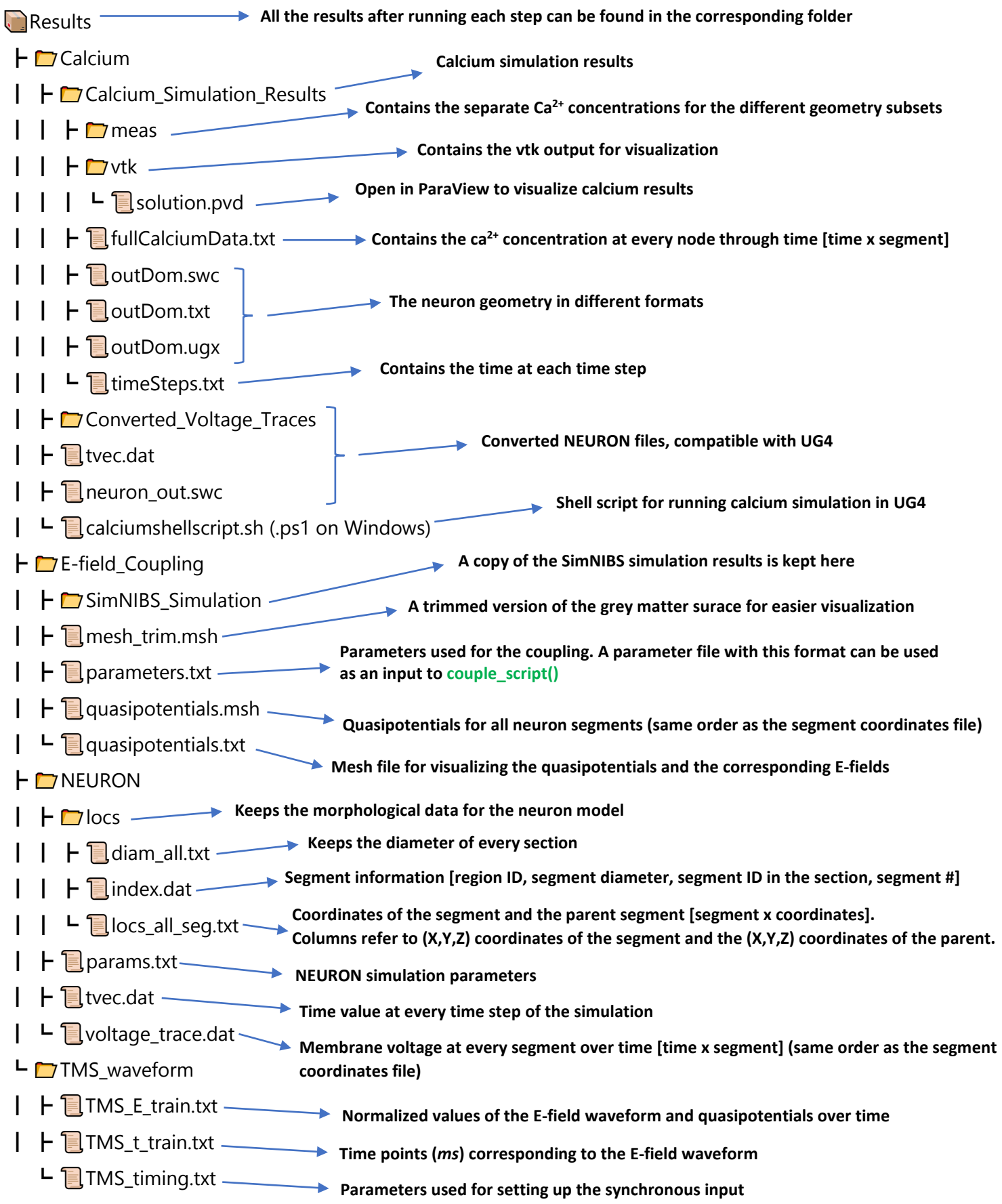
Contents of the model folder



Contents of the Code folder



Contents of the Results folder



Section 5. Advanced parameters for calcium simulation

Parameters available in the GUI	
PARAMETER CALL	DESCRIPTION OF INPUT
-grid	Use this command to specify the path to the .swc file, where the .swc file is the neuron geometry
-setting	Use this command to specify the type of calcium simulation, it is set to 'none' to not include intracellular calcium mechanisms due to the endoplasmic reticulum. The other options (currently in the prototype phase) are 'ryr', 'ip3r', and 'all' corresponding to only ryanodine receptors, only ip3 receptors, and all receptors, respectively
-dt	Use this command to specify the time step size for running the calcium simulation, in seconds
-endTime	Use this command to specify the end time of the simulation, this is set automatically when checking "Default end time"
-vtk	Use this command to output vtk files for use in ParaView, vtk output can be used to visualize the output data and make videos
-pstep	Use this command to specify how often to output vtk data, i.e. -pstep 0.01 would mean to output vtk data every 0.01 seconds Note: this is not the same as -dt parameter call
-vmData	Use this command to specify the path to the voltage data
-outName	Use this command to specify the output path of the calcium simulation
-solver	Use this command to specify the type of numerical solver GS = Gauss Seidel ILU= Incomplete LU decomposition GMG = Geometric Multi-Grid
-minDef	Use this command to specify the expected numerical accuracy
-numNewton	Use this command to specify the number of Newton iterations for numerically solving the nonlinear equations
-vSampleRate	Use this command to specify the sampling rate of the input voltage data, this is set automatically

Parameters available in the .lua script for advanced users (not recommended)	
VARIABLE	DESCRIPTION OF INPUT
totalBuffer	This variables sets the total calcium buffering that is available in the dendrite
verbose	This variable will output linear solver convergence progress
D_cac	This variable sets the diffusion constant for the cytosolic calcium
ca_cyt_init	This variable sets the initial calcium concentration in the cytosol
pmcaDensity/ ncxDensity/ vdccDensity	These variables are the densities of the plasma membrane calcium exchangers