By Boshuo Wang, February 27, 2018

Summary

This repository contains MATLAB code and data for the article titled “Coupling magnetically induced electric fields to neurons: longitudinal and transverse activation”. The code simulates magnetic stimulation of long peripheral nerve and investigates how much the electric field (E-field) component transverse to the nerve trunk affects activation thresholds.

Model set up

The model includes a total of 12 combinations of magnetic stimulation set-ups: three coil configurations, two electric field waveforms, and two types of axon models (see article for details). The models are coded by strings in the format of `COIL_wvf_AXON`:

- `COIL` is the coil configuration, and determines the number of parameters (axon position relative to center of coil) tested:
  - **`SC`**: single circular coil. 615 = 41 × 15 (41 lateral and 15 vertical positions)
  - **`F8Ca`**: Figure-8 coil, maximum E-field aligned with nerve. 465 = 31 × 15 (31 lateral positions)
  - **`F8Cp`**: Figure-8 coil, maximum E-field perpendicular to nerve. 615 = 41 × 15 (41 lateral positions)
- `wvf` is the stimulation waveform of E-field:
  - **`mp`**: Monophasic stimulation waveform
  - **`hs`**: Half-sine stimulation waveform
- `AXON` is the axon model
  - **`HH`**: unmyelinated axon with Hudgkin-Huxley ion channels adjusted to room temperature (Hudgkin-Huxley, 1952)
  - **`RMG`**: human myelinated axon at body temperature with properties given by Richardson, McIntyre & Grill, 2000

Units

The following unit system is used in the simulations (sometimes units are converted, but only for text display and in figures):

• Temperature	Celsius	
• Length	centimeter	1 cm = 10 mm = 10^4 μm
• Time and frequency	millisecond and kilohertz	1 ms = 1 kΩ·μF = 1 kHz^{-1}
• Potentials	millivolt	1 mV = 1 kΩ·μA = 1 ms·μA·$μF^{-1}$

• Currents	microampere	1 μA = 1 mS·mV
• Resistance and conductance	kiloohm and millisiemens	1 kΩ = 1 mV·μA^{-1} = 1 mS^{-1}
• Capacitance	microfarad	1 μF = 1 ms·μA·mV^{-1}
• Charge	nanocoulomb	1 nC = 1 ms·μA = 1 mV·μF

Math of cable equation solver

`simulate_cable_HH.m` and **`simulate_cable_RMG.m`**

For cable with linear topology (i.e., no branching), the transversely-averaged membrane potential along the cable is discretized in space and time:

$$\bar{\varphi}_{\mathrm{m}}(z,t) \to \bar{\varphi}_{\mathrm{m}}(z_n,t_k) \to \bar{\varphi}_{\mathrm{m}}^{k}[n]$$

The discretization is not necessary uniform, i.e., compartment length Δz_n and time step Δt_kmay vary. The cable has N compartments, each with radius R_n, surface area $A_n = 2\pi R_n \Delta z_n$, and specific membrane capacitance c_n. The compartments have the following time-and location-dependent parameters at each time step:

- Equivalent channel conductance and reversal potential $\bar{g}_n^k$ and $\bar{\mathcal{E}}_n^k$ (includes averaging for transverse dimension in modified cable equation (see Wang et al., 2018, *J. Neural Eng.*) and for different ion channels indexed by j)
$$\bar{g}_n^k = \sum_j g_n^{(j),k}, \bar{\mathcal{E}}_n^k = \frac{1}{\bar{g}_n^k}\sum_j \mathcal{E}_n^{(j),k} g_n^{(j),k}$$
- Equivalent time constant $\bar{\tau}_n^k$
$$\bar{\tau}_n^k = c_n/\bar{g}_n^k$$
- Left and right axial resistance $R_{\mathrm{i},-}[n]$ and $R_{\mathrm{i},+}[n]$. (For the two terminals, the axial resistance outward contains only one term.)
$$R_{\mathrm{i},-}[n] = \rho_{\mathrm{i}}\left(\frac{\Delta z_{n-1}}{2\pi R_{n-1}^2} + \frac{\Delta z_n}{2\pi R_n^2}\right), R_{\mathrm{i},+}[n] = \rho_{\mathrm{i}}\left(\frac{\Delta z_n}{2\pi R_n^2} + \frac{\Delta z_{n+1}}{2\pi R_{n+1}^2}\right)$$
- Left and right length constants $\bar{\lambda}_-^k[n]$ and $\bar{\lambda}_+^k[n]$. (For the two terminals, the length constant outward can be calculated, but is irrelevant in the backward Euler calculation.)
$$\bar{\lambda}_-^k[n] = \left(A_n \bar{g}_n^k R_{\mathrm{i},-}[n]\right)^{-\frac{1}{2}}, \bar{\lambda}_+^k[n] = \left(A_n \bar{g}_n^k R_{\mathrm{i},+}[n]\right)^{-\frac{1}{2}}$$

The backward Euler equation for the entire cable is given in matrix form

$$\left(\bar{\boldsymbol{\tau}}^{\boldsymbol{k}} + \Delta t_k\left(\boldsymbol{I}_N - \boldsymbol{\Lambda}^k\right)\right)\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k+1} = \bar{\boldsymbol{\tau}}^{\boldsymbol{k}}\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k} + \Delta t_k\left(\bar{\boldsymbol{\mathcal{E}}}^k + \boldsymbol{\Lambda}^k\bar{\boldsymbol{\varphi}}_{\mathrm{e}}^{k}\right)$$

The column vectors are

$$\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k} = \left[\bar{\varphi}_{\mathrm{m}}^{k}[n]\right]_{N\times 1}, \bar{\boldsymbol{\varphi}}_{\mathrm{e}}^{k} = \left[\bar{\varphi}_{\mathrm{e}}^{k}[n]\right]_{N\times 1}, \bar{\boldsymbol{\mathcal{E}}}^k = [\bar{\mathcal{E}}_n^k]_{N\times 1}$$

and the square matrices are

$$\bar{\boldsymbol{\tau}}^{\boldsymbol{k}} = \mathrm{diag}\left(\bar{\tau}_n^k\right), \boldsymbol{I}_N = \mathrm{diag}(1,1,\ldots,1)$$

$$\boldsymbol{\Lambda}^k = \left[\Lambda_{i,j}^k\right]_{N\times N^-} = \begin{cases} \left(\bar{\lambda}_-^k[n]\right)^2 & i = j+1 = n \\ -\left(\bar{\lambda}_-^k[n]\right)^2 - \left(\bar{\lambda}_+^k[n]\right)^2 & i = j = n \\ \left(\bar{\lambda}_+^k[n]\right)^2 & i = j-1 = n \end{cases}$$

The $\bar{\boldsymbol{\mathcal{E}}}^k$ term represents the ionic current due to nonlinear channels, the $\boldsymbol{\Lambda}^k\bar{\boldsymbol{\varphi}}_{\mathrm{e}}^k$ term represents the activating function of the extracellular field, and the remaining terms containing $\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^k$ and $\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k+1}$ represent the repolarization and propagation of membrane potentials.

The matrix on the left-hand side is tridiagonal and $\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k+1}$ can be solved with the tridiagonal algorithm with linear time complexity. If branching is included, off-diagonal entries in the matrix would require a more general Gaussian elimination algorithm. Due to the tree-like topology, the matrix is sparse and the backward Euler step still requires linear time complexity.

File organization

- **`main_MS_HH.m`**: main function to obtain thresholds for unmyelinated HH model axon with both conventional and modified cable equations. `main_MS_HH(mod_prmtr, out_ctrl)`. Call function in a loop (parallelized) or run on computer cluster to cover all parameters `id` (see below). Input arguments:
  - `mod_prmtr`: structure specifying model parameter. Fields:
    - `model_name`, a string with format `COIL_wvf`: `'SC_mp'`, `'SC_hs'`, `'F8Ca_mp'`, `'F8Ca_hs'`, `'F8Cp_mp'`, `'F8Cp_hs'`.
    - `id`: integer, parameter ID. Range for the 3 coil configurations: `SC` & `F8Cp`: 1-615. `F8Ca`: 1-465.
  - `out_ctrl`: structure specifying outputs. Fields of logical or 0/1:
    - `if_save_data`: whether to save the output `results` in a .mat file in a subfolder for each simulation.
    - `if_write_log`: whether to write threshold finding process in a .txt log in a subfolder for examining search process.
    - `if_plot`: whether to plot threshold finding process in a subfolder for visual confirmation of threshold values.

  Output arguments:

  - `results`: structure containing simulation results. Fields:
    - `th_CE`: threshold obtained with conventional cable equation.
    - `th_MCE`: threshold obtained with modified cable equation.
    - `th_per_diff_MCE`: percentage different of thresholds comparing `th_MCE` versus `th_CE`.
- **`specify_model_MS_HH.m`**: specifies model, generates cable and stimulation waveform, calculates E-field, and sets parameters for solver and threshold search. Called by **`main_MS_HH.m`**.
- **`simulate_cable_HH.m`**: runs simulation for HH axon, returns whether stimulation is suprathreshold or subthreshold. Called by **`threshold_finding.m`** (located in subfolder).

- **HH.m**: ion channel gating variables. Called by **simulate_cable_HH.m**.
- **compile_HH.m**: compile data from individual simulations for all of the 6 models with HH axon. Generates COIL_wvf**_HH_compiled_result.mat** in each of the COIL_wvf**_HH** folders. Run after all simulations complete.
- **plot_MS_HH_combined.m**: generates figure for all 6 models with HH axon (Figure 3 in article). Called by **compile_HH.m**.
- **main_MS_RMG.m**: main function to obtain thresholds for myelinated axon. Five threshold values: 2 for straight axons with both conventional and modified cable equations, and 3 for undulating axons with conventional cable equation. `main_MS_RMG(mod_prmtr, out_ctrl)`. Call using a loop (parallelized) or run on cluster to cover all parameter id. Input arguments are the same as **main_MS_HH.** Output arguments:
  - `results`: structure containing simulation results. Fields:
    - `th_CE`: threshold of straight axon obtained with conventional cable equation.
    - `th_MCE`: threshold of straight axon obtained with modified cable equation.
    - `th_per_diff_MCE`: percentage different of thresholds comparing `th_MCE` versus `th_CE`.
    - `th_UA`: threshold of fiber with axonal undulation obtained with conventional cable equation.
    - `th_per_diff_UA`: percentage different of thresholds comparing `th_UA` versus `th_CE`.
    - `th_UF`: threshold of fiber with fascicle undulation obtained with conventional cable equation.
    - `th_per_diff_UF`: percentage different of thresholds comparing `th_UF` versus `th_CE`;
    - `th_UAF`: threshold of fiber with axonal and fascicle undulation obtained with conventional cable equation.
    - `th_per_diff_UAF`: percentage different of thresholds comparing `th_UAF` versus `th_CE`.
- **specify_model_MS_RMG.m**: specifies model, generates cable and stimulation waveform, calculates E-field, and sets parameters for solver and threshold search. Called by **main_MS_RMG.m**.
- **simulate_cable_RMG.m**: runs simulation for RMG axon, returns whether stimulation is suprathreshold or subthreshold. Called by **threshold_finding.m** (located in subfolder).
- **RMG.m**: ion channel gating variables. Called by **simulate_cable_RMG.m**.
- **compile_RMG.m**: compile data from individual simulations for each of the 6 models with RMG axon. Generates COIL_wvf**_RMG_compiled_result.mat** in the each of the COIL_wvf**_RMG** folders. Run after all simulations complete.
- **plot_MS_RMG_combined.m**: generates figure for all 6 models with straight RMG axon (Figure 4 in article). Called by **compile_RMG.m**.
- **plot_MS_RMG_UND_combined.m**: generates figure for all 6 models with undulating RMG axon (Figure 5 in article). Called by **compile_RMG.m**.
- **Shared functions and data**: folder containing code and data shared for both axon models.
  - **create_folders.m**: creates subfolders for saving data, logs, and figures. Called by **main_MS_HH.m** & **main_MS_RMG.m**.

- **write_fun.m**: output function for writing log files or displaying in MATLAB command lines. Called by many functions.
- **threshold_finding.m**: finds threshold for a specific model set-up. Called by **main_MS_HH.m** & **main_MS_RMG.m**.
- **tridiag.m**: solves backward Euler iteration using the tridiagonal matrix algorithm for linear neuronal topology. See solver descript below. Called by **simulate_cable_HH.m** & **simulate_cable_RMG.m**.
- **TMS_wave.m**: loads and processes E-field waveforms for magnetic stimulation (resampling according to the simulation time step, rescaling the peak amplitude at onset, and ensuring a zero time integral). Called by **specify_model_MS_HH.m** & **specify_model_MS_RMG.m**.
- **MagproX100_TMS_waves.mat**: normalized E-field waveforms for magnetic stimulation recorded from a MagProX100 device. Includes monophasic, half-sine, and biphasic waveforms.

- **Plot functions**: folder containing code related to visualization for both axon models. The functions are called by **plot_MS_HH_combined.m**, **plot_MS_RMG_combined.m**, and **plot_MS_RMG_UND_combined.m**.
- **SC_mp_HH**, **SC_hs_HH**, **SC_mp_RMG**, **SC_hs_RMG**, **F8Ca_mp_HH**, **F8Ca_hs_HH**, **F8Ca_mp_RMG**, **F8Ca_hs_RMG**, **F8Cp_mp_HH**, **F8Cp_hs_HH**, **F8Cp_mp_RMG**, **F8Cp_hs_RMG**: Folders containing simulation results and outputs.
  - **Results**: subfolder containing simulation results generated by **main_MS_HH.m** or **main_MS_RMG.m** and saved in **results_**`id`**.mat**.
  - `COIL_wvf_AXON`**_compiled_result.mat**: all simulation results for the given model set-up compiled from the **Results** subfolder by **compile_HH.m** or **compile_RMG.m**.
  - **Logs**: subfolder containing log files generated by **main_MS_HH.m** or **main_MS_RMG.m** documenting the threshold search process in **log_**`id`**.txt**.
  - **Figures**: subfolder for figures. Examples figures provided in .tif format. Original .fig files not included due to the file size (total of several hundred GBs for all simulations) but can be generated and examined by re-running the simulations of interest (typically less than half an hour for RMG models and between half an hour up to 8 hours for HH models depending on parameter).
- **SC_mp_HH.slurm**: Example slurm script to run **main_MS_HH.m** on a computer cluster for coil model `'SC_mp'` (all parameters `id` 1-615, with all 3 types of outputs).

Note on simulations with HH-model axon: When the modified cable equation is used, the unmyelinated axon could have two "thresholds". The multiple locations of hyper- and depolarization along the axon could generate an action potential at one location that may or may not pass through another location depending on the stimulation amplitude. As the amplitude increases, the stimulation first generates an action potential that passes through unobstructed to the detection site (near one of the axon terminals) and the simulation is determined as suprathreshold. Slightly larger stimulation amplitude could result in an obstruction of the action potential due to various polarization behavior in between the action potential initiation site and the detection site, and the search

algorithm determines the amplitude subthreshold. For even larger amplitude, the location that previously obstructed the action potential depolarizes sufficiently to become the action potential initiation site. The amplitude windows for passing or blocking action potentials could be very small, and for different model and search parameters (the initial amplitude and factor for increasing/decreasing the amplitude in the search), the algorithm could miss the lower threshold and determines the threshold as the larger one. Because the two thresholds are close, practically there is no difference. However, when a parameter has two thresholds, only the lower threshold is used for consistency as well as for smooth visualization.

The simulations were first run with the default parameters in the code, and upon inspection of the figures, simulations that resulted in the upper thresholds were rerun using slightly adjusted search parameters (line 46 in **`main_MS_HH.m`**, adjustment highlighted):

```
solver.thresh_find.factor = (solver.thresh_find.factor)^(1/6); % or ^(1/9)
```

The smaller factor for changing the stimulation amplitude (or a slightly different initial search point, line 45) allows the search to find the lower threshold.