By Boshuo Wang, March 1, 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 a 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 (lateral and vertical positions of the nerve relative to center of coil):
  - **`SC`**: single circular coil. 615 = 41 × 15
  - **`F8Ca`**: Figure-8 coil, maximum E-field aligned with nerve. 465 = 31 × 15
  - **`F8Cp`**: Figure-8 coil, maximum E-field perpendicular to nerve. 615 = 41 × 15
- `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 and Huxley, 1952)
  - **`RMG`**: myelinated axon of human peripheral nerve at body temperature (Richardson, McIntyre, and Grill, 2000)

Units

The unit system is used in the simulations is given as follows. (Sometimes units are converted, but only for text display and in figures.)

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

• Current	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 $\bar{\varphi}_\mathrm{m}$ along the cable (z direction) 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^k may vary. The cable has N compartments, each with radius R_n, surface area $A_n = 2\pi R_n \Delta z_n$, specific membrane capacitance c_n, and axial resistances to its previous/left (−) and next/right (+) neighbor $R_{\mathrm{i},-}[n]$ and $R_{\mathrm{i},+}[n]$

$$R_{\mathrm{i},\pm}[n] = \rho_\mathrm{i}\left(\frac{\Delta z_n}{2\pi R_n^2} + \frac{\Delta z_{n\pm1}}{2\pi R_{n\pm1}^2}\right)$$

The outward axial resistances ($R_{\mathrm{i},-}[1]$ and $R_{\mathrm{i},+}[N]$) would contain only one term, but can be set to 0, NaN, or $+\infty$ since they are irrelevant in the cable equation calculation.

The compartments have the following time- and/or location-dependent parameters at each time step:

- Equivalent channel conductance and reversal potential $\bar{g}_n^k$ and $\bar{\mathcal{E}}_n^k$

  $$\bar{g}_n^k = \sum_{b,j} g_{b,n}^{(j),k}\,,\ \bar{\mathcal{E}}_n^k = \frac{1}{\bar{g}_n^k}\sum_{b,j} \mathcal{E}_{b,n}^{(j),k}\cdot g_{b,n}^{(j),k}$$

  The equivalent channels takes into account the averaging of different ion channels (indexed by j), and, for the modified cable equation (see Wang et al., 2018, *J. Neural Eng.*), the discretization of the membrane in the transverse dimension (indexed by b).
- Equivalent time "constant" $\bar{\tau}_n^k$

  $$\bar{\tau}_n^k = c_n/\bar{g}_n^k$$
- Left and right length "constants" $\bar{\lambda}_-^k[n]$ and $\bar{\lambda}_+^k[n]$

  $$\bar{\lambda}_\pm^k[n] = \left(A_n \bar{g}_n^k R_{\mathrm{i},\pm}[n]\right)^{-\frac{1}{2}}$$

  The length constants $\bar{\lambda}_-^k[1]$ and $\bar{\lambda}_+^k[N]$ do not appear in the backward Euler calculation of the cable equation.

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 $\bar{\boldsymbol{\varphi}}_{\mathrm{m,e}}^k = \left[\bar{\varphi}_{\mathrm{m,e}}^k[n]\right]_{N\times1}$ and $\bar{\boldsymbol{\mathcal{E}}}^k = [\bar{\mathcal{E}}_n^k]_{N\times1}$, the diagonal matrices are $\bar{\boldsymbol{\tau}}^{\boldsymbol{k}} = \mathrm{diag}\left(\bar{\tau}_n^k\right)$ and $\boldsymbol{I}_N = \mathrm{diag}(1,1,\ldots,1)$, and the tridiagonal matrix $\boldsymbol{\Lambda}^k$ is given by

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

The $\bar{\boldsymbol{\mathcal{E}}}^k$ term represents the ionic currents, the $\bar{\boldsymbol{\varphi}}_{\mathrm{e}}^k$ 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 first and last terms of the main diagonal of $\boldsymbol{\Lambda}^k$ are different to other terms, which reflect sealed boundary condition at the cable's terminals.

The matrix $\bar{\boldsymbol{\tau}}^{\boldsymbol{k}} + \Delta t^k\left(\boldsymbol{I}_N - \boldsymbol{\Lambda}^k\right)$ on the left-hand side is tridiagonal and $\bar{\boldsymbol{\varphi}}_{\mathrm{m}}^{k+1}$ can be solved using the tridiagonal algorithm with $O(N)$ run-time complexity. If branching is included, the $\boldsymbol{\Lambda}^k$ matrix contains off-diagonal terms for branch compartments, but the total number of non-zero terms is unchanged due to the tree-like topology. The backward Euler step would require a more general Gaussian elimination algorithm but does not need to calculate the inverse of the left-hand side matrix, and therefore run times still scales linearly with the number of compartments.

File organization

- **`main_MS_HH.m`**: main function to obtain thresholds for unmyelinated HH model axon with both conventional and modified cable equations. `results = 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 the simulation results. Fields:
    - `th_CE`: threshold obtained with the conventional cable equation.
    - `th_MCE`: threshold obtained with the 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. `results = 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 the simulation results. Fields:
    - `th_CE`: threshold of straight axon obtained with the conventional cable equation.
    - `th_MCE`: threshold of straight axon obtained with the 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 and addition figure for half-sine waveform). 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`**: function to find threshold for a specific model set-up. Called by **`main_MS_HH.m`** & **`main_MS_RMG.m`**, and calls **`simulate_cable_HH.m`** or **`simulate_cable_RMG.m`** various times.
  - **`tridiag.m`**: solves backward Euler of the cable equation using the tridiagonal matrix algorithm for linear neuronal topology. See solver descript below. Called by **`simulate_cable_HH.m`** & **`simulate_cable_RMG.m`** for each backward Euler step.
  - **`TMS_wave.m`**: loads and processes E-field waveforms for magnetic stimulation (resampling according to the simulation time step, rescaling the peak amplitude at pulse onset, and ensuring a zero time integral for the waveform). 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, i.e., .mat file, .txt log, and .fig figure).

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 amplitude. 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.