**Running forward solver**

See ‘D:\Depth electrode github example\Rat\_055\Running forward’ on harddrive or ‘eit-nas/shared/Mayo Depth electrode github example\Rat\_055\Running forward’ on eit-nas for example of output from various steps

See PEITs repository on github for usage of solver

1. **Run forward for depth protocol**

Overview

Runs the forward for each protocol line. Automatically updates the relevant lines in the parameter and standard parameters file for each protocol line. Also reads in the boundary voltages and Jacobian and saves them into specified folder.

Code/files required

1. run\_forward\_for\_each\_protocol.m
2. load\_electrode\_voltages\_binary.m
3. load\_sigma\_vector\_binary.m
4. load\_jacobian\_binary\_in\_steps.m
5. concatenate\_protocol.m
6. Mesh\_rods\_RatXX\_hex.mat
7. prtN\_inj\_idx.mat

Input

In ‘run\_forward\_for\_each\_protocol.m’

1. Lines 1 – 9 – change so code is pointing to correct files
2. Line 24, 38,39,40,46,52 – mesh\_Rat\_056 to the name of your mesh (change this so you define the name at the top, it will be much easier)

In parameters in ‘PEITS\_root/PEITS/data/’

1. Line 8 – this should be false as we have to make new partitions for each protocol as node assignment changes each time
2. Line 17 – change this to mesh\_RatXX.dgf
3. Line 48 – ‘current.protocol: ../data/protocol\_mesh\_RatXX\_prt1’

In standard parameters in ‘PEITS\_root/PEITS/data’

1. Line 25 – change contact.impedance to 10000 (or whatever the contact impedance of your electrodes are)
2. Line 26 – change input.current to 10e-6 (or whatever you injected in experiments)

In param\_mesh\_RatXX

1. Line 8 – change to ‘electrode.positions: ../data/electrode\_positions\_mesh\_RatXX\_prt1’
2. Line 10 – change to ‘electrode.nodes: ../data/electrode\_nodes\_mesh\_RatXX\_prt1’

In load\_jacobian\_binary\_in\_steps.m

1. Line 29 – change to the number of measured electrodes (73 when dealing with shunted depth and cortical)

Output

1. BV\_prtN.mat
   * This will be saved wherever you have specified save path in ‘run\_forward\_for\_each\_protocol.m’
   * Electrode voltages for each protocol line
2. J\_hex\_prtN.mat
   * Jacobian for each protocol line
3. Use concatenate\_protocols.m to put individual BV and J\_hex from each protocol together (into normal format when output from forward). Get out BV\_depth.mat and J\_hex\_depth.mat

Notes

1. The final output is equal to number of electrodes (32 depth and 47 cortical). The change in numbering required for the shunting protocol is readjusted such that the same numbering as experimental data is used. The BV and Jacobian for the electrodes that were incorporated into a single one when shunting are set to zero (see lines 88-95 in run\_forward\_for\_each\_protocol.m)
2. run\_forward\_for\_each\_protocol.m it is set to use one processor (line 65). I found that when using more the BV changed compared to single processor case. This is something to investigate more carefully. Ideally want to use more processors as much faster
3. **Run forward for cortical protocol**

Overview

For this case forward can be run in the normal mode from the terminal

Code/Files required

1. load\_electrode\_voltages\_binary.m
2. load\_sigma\_vector\_binary.m
3. load\_jacobian\_binary\_in\_steps.m
4. Mesh\_rods\_RatXX\_hex.mat

Input

1. Change relevant lines in parameter and standard\_parameters to point to correct files
2. I set contact impedance to 1000 for cortical injection and current to 50e-6, change as necessary if these vary
3. In load\_jacobian\_binary\_in\_steps change line 29 to number of depth electrodes and cortical electrode (79)

Output

1. BV for all protocol lines. Save as BV\_cortex.mat
2. Jacobian for all protocol line. Save as J\_hex\_cortex.mat