**Modelling motor-evoked potentials**

*mep\_construction\_v\_1\_0.m*

**Updated documentation with note on NFTsim versions 28 October 2022 and including an ‘example.output’ file**

**28 October 2022. Note on NFTsim versions**

The MEP modelling code is written to call NFTsim version 0.1.4, then called ‘neurofield’. This is how the modelling in Wilson, Moezzi & Rogasch (2020) was done. However, to run with newer versions of NFTsim, the call to NFTsim from Matlab needs to be correct.

**For NFTsim version 1.1.0, line 395 in MEP\_construction\_v\_1\_0.m should be changed to replace ‘neurofield’ with ‘nftsim’.**

**Additionally, with NFTsim 1.1.0 and above one might have to edit the .conf file marcus\_template\_june2019.conf and change the second line from** Time: 40 Deltat: 1e-4 **to** Time: 40 Deltat: 2^-13 **to ensure that the total simulation time is divisible by Deltat accounting for precision errors. This might be dependent on the machine on which the code is running. Please refer to NFTsim documentation for further analysis of this issue.**

**Introduction**

This model is explained in detail in Wilson, Moezzi & Rogasch (2020). Modeling motor-evoked potentials from neural field simulations of transcranial magnetic stimulation. <https://www.biorxiv.org/content/10.1101/847830v3>

The software is written in Matlab. It provides a front-end and post-processing for the neural field model NFTsim. The NFTsim code is found at <https://github.com/BrainDynamicsUSYD/nftsim> and documented here: Sanz-Leon, P., Robinson, P.A., Knock, S.A., Drysdale, P.D., Abeysuriya, R.G., Fung, P.K., Rennie, C., Zhao, X.. NFTsim: Theory and simulation of multiscale neural field dynamics. PLoS Computational Biology 2017;14:e1006387.

**Installing**

1. Install Matlab (The Mathworks, Inc); licensing fees may be needed.

2. Install NFTsim from <https://github.com/BrainDynamicsUSYD/nftsim> (freely available)

3. Create a new directory ‘MEP\_model’ under the top-level NFTsim directory to contain the MEP model files (I suggest ‘MEP\_model’, although the name does not matter)

4. Copy all files (.m, .mat, .conf, .output) in this repository into the MEP\_model directory

**Running**

Run Matlab. Set working directory to MEP\_model

Run the model by running MEP\_construction\_v\_1\_0.m (e.g. by typing “MEP\_construction\_v\_1\_0” in the matlab command window, or running with the green arrow from a Matlab editor window).

This will write input to a NFTsim .conf file, run NFTsim, and process the NFTsim output file, giving plots of MEP against amplitude, or MEP against ISI (for paired-pulse protocols), and plots of various variables (notably synaptic weights \nu\_{ee}) against time.

**Selecting input data**

The kind of plot required (MEP against stimulation amplitude, MEP against ISI, etc) can be selected by the true/false flags at the top of the MEP\_construction\_v\_1\_0.m code

The parameters for the stimulation are set by changing the NFTsim template .conf file, marcus\_template\_june2019.conf . This file is set-up as follows, and is shown diagrammatically in Figure 1 in this readme file.

**Neural Populations**

*Populations 1 and 8* provide external TMS stimulation. Two populations are used to allow for combining pulse trains with differing patterns (e.g. two pulses of differing amplitudes). See below.

*Population 2* is the excitatory population. *Population 5* is a copy of the excitatory population, that receives exactly the same inputs as population 2 (this allows a slow and a fast stimulation to be made to the layer 5 neurons).

*Populations 3 and 4* are inhibitory populations. ‘3’ links with the slower GABAB timescales, whereas ‘4’ links with the faster GABAA timescales.

*Population 6* is the layer V cortical neuron population.

*Population 7* is defunct (essentially it is a delayed copy of population 6).

**Couplings and propagators**

These are set-up in the matrix at the top of the .conf file. Parameters can be changed (e.g. different values of excitatory-to-excitatory coupling) by editing these. In order, they correspond to:

*1. External 1 (x1) population -to-excitatory (e) population*

*2. e to e*

*3. inhibitory (gabab modulated, i\_b) population to excitatory population*

*4. inhibitory (gabaa modulated, i\_a) to excitatory*

*5. external 2 (x2) to excitatory*

*6. x1 to i\_b*

*7. e to i\_b*

*8. i\_b to i\_b*

*9. i\_a to i\_b*

*10. x2 to i\_b*

*11. x1 to i\_a*

*12. e to i\_a*

*13. i\_b to i\_a*

*14. i\_a to i\_a*

*15. x2 to i\_a*

*16. x1 to copy of exciatory (ec)*

*17. e to ec*

*18. i\_b to ec*

*19. i\_a to ec*

*20. x2 to ec*

*21. x1 to layer 5 (v)*

*22. e to v (fast)*

*23. i\_b to v (medium delay)*

*24. i\_a to v (medium delay)*

*25. x2 to v*

*26. copy of excitatory population to layer v (slow)*

*27. layer v to motoneurons (time delay added)*

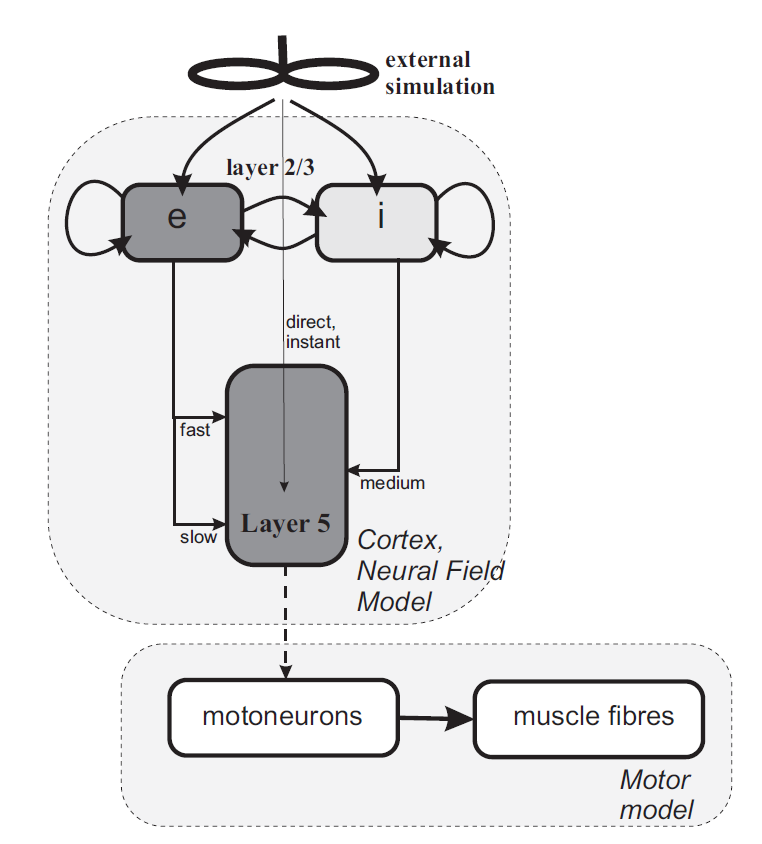
**Output**

Various Matlab figures are produced. Notably, Figure 8 (which corresponds to Fig. 3 in the Wilson, Moezzi, Rogasch manuscript) gives EMG as a function of time (part a), layer 5 firing rate as a function of time (part b) and firing of individual motounits against time (part c). Also, Figure 21 gives MEP size against either stimulation amplitude or ISI, which ever has been selected.

**Assistance**

Further assistance with implementing the software can be obtained from Marcus Wilson, [Marcus.wilson@waikato.ac.nz](mailto:Marcus.wilson@waikato.ac.nz), telephone +64 7 838 4834

**Summary of populations and propagators**



**27**

**21, 25**

**26**

**23,24**

**22**

**1, 5, 16, 20**

**3, 4, 18,19**

**2, 17**

**8, 9, 13, 14**

**7, 12**

**6, 10, 11, 15**

**1, 8**

**6**

**3 (B), 4 (A)**

**2, 5**

***Figure 1. A copy of Figure 1 from the article, but annotated with the population numbers (in blue underline) and propagators (in green).***

**Further note on multiple populations**

To implement the motor evoked potential modeling, I have needed to implement NFTsim in a more complicated manner than is often required. The implementation of NFTsim uses multiple populations in the model to represent the same population of cells, as described here:

**Two stimulating populations**

Two populations (1 and 8) are used to enable two pulses of differing amplitudes (e.g. for paired pulse protocols SICI, ICF). To achieve this, the user needs to specify the first population (1) as providing two positive pulses with a certain delay, and the second population (8) with a single negative pulse. This ensures the first pulse (1 + 8) has a lower amplitude than the second (which comes from just population 1. Some calculation is required to ensure it couples with the correct strength (ask me for help).

**Two excitatory populations**

Two excitatory populations (2 and 5) are used. Population 5 is simply a mirror of population 2. It receives the same inputs as 2. This is to allow a ‘fast’ (from population 2) and a ‘slow’ (from 5) input to the layer 5 neurons.

**Two inhibitory populations**

Two inhibitory populations (3 and 4) are used to allow the inhibitory cells to couple with both slow (GABAB, population 3) and fast (GABAA, population 4) synaptic responses. Both populations have identical inputs so they are mirrors of each other.

Marcus Wilson, 12 October 2020