NeuroField: A Neural Field Theory simulation toolbox

P.K. Fung*, R.G. Abeysuriya, X. Zhao, P. M. Drysdale, P.A. Robinson

School of Physics, University of Sydney, New South Wales, Australia

Abstract

Neural field theories are powerful, computationally efficient approach to modeling large-scale brain phenomena. In particular, the versatile neural field theory of Robinson et. al. addresses questions on brain physiology and matches them with experimental observables including the EEG spectra, evoked response potentials, age-related changes to the physiology of the brain, seizures, and synaptic plasticity phenomena. This paper presents NeuroField as a user-friendly, extensible, portable, well-documented software package applicable to simulate a wide range of neural field phenomena, packaged with MATLAB and Python routines for higher-level analysis. NeuroField is available for non-commercial use at `http://physics.usyd.edu.au/brain/neurofield`.

Keywords: EEG, neurophysiology, methods, modeling

1. Introduction

Neural field theory modeling is a powerful technique for constructing relatively simple, physiologically based models of the brain that are capable of predicting EEG and correlate well with experimental data Deco et al. (2008), Pinotsis et al. (2012). In particular, the neural field theory of Robinson et. al. is a well established theory that addresses the EEG spectra, evoked response potentials, age-related changes to the physiology of the brain, seizures, and synaptic plasticity phenomena (Breakspear et al., 2006, Kerr et al., 2011,

*Corresponding author. Tel. +61 9036 7274
Email address: `ffung@physics.usyd.edu.au` (P.K. Fung)

O'Connor and Robinson, 2004, Rennie et al., 2002, Robinson et al., 2002, 2004, 2005, 2001, 2003, Rowe et al., 2004, van Albada et al., 2010). In this neural field theory, we make a continuum approximation in which neural properties are averaged over spatial scales of an mm or so: sufficient to contain large numbers of neurons, but small enough to resolve fine structures in brain activities. Within this mean field context, rather than looking at individual neurons, we classify neural populations by the type of neuron (e.g. pyramidal excitatory neurons or inhibitory interneurons), and mean field quantities are label by their spatial position. In each modeling task, we specify the neural populations, stimulations, and synapto-dendritic connections between them (self-connection within a population is allowed), so that each neural population receives local dendritic voltage input and influences other populations via its action potential firing, measured via its instantaneous axonal action potential firing rate. Figure 8 is an example of such a "population model," where we have two cortical populations and two thalamic populations, with intracortical, intrathalamic, and corticothalamic connections; details about this particular population model can be found in Sec. 3.3.

The standard theory (Robinson et al., 2005) is represented by three equations, which are schematically illustrated in Fig. 2.:

$$D_{ab}V_{ab}(\mathbf{r},t) = \nu_{ab}\phi_{ab}(\mathbf{r},t), \tag{1}$$

$$Q_a(\mathbf{r},t) = S_a\big[\sum_b V_{ab}(\mathbf{r},t)\big], \tag{2}$$

$$\mathcal{D}_{ab}\phi_{ab}(\mathbf{r},t) = Q_b(\mathbf{r},t-\tau_{ab}). \tag{3}$$

whre the operators D_{ab}, S_a, and $\mathcal{D}_{ab}$ capture synapto-dendritic smoothing, soma response, and axonal propagation of action potentials, respectively, given by

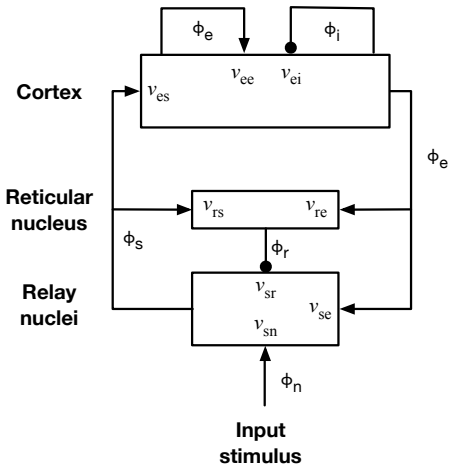

Figure 1: Schematic illustration of the corticothalamic model, which contain excitatory and inhibitory cortical populations (e and i, respectively), and reticular and relay thalamic populations (r and s, respectively), where the relay population receives subthalamic stimulation. Arrows indicate excitatory connections, while roundheaded arrows indicate inhibitory connections. This population model is capable of capturing the alpha rhythm, age-related changes to the physiology of the brain, evoked response potentials, seizures, and many other phenomena, as elaborated in Sec. 3.3.

$$D_\alpha(t) = \frac{1}{\alpha\beta}\frac{d^2}{dt^2} + \left(\frac{1}{\alpha} + \frac{1}{\beta}\right)\frac{d}{dt} + 1, \tag{4}$$

$$S(V_a) = \frac{Q_{\max}}{1 + \exp[-(V_a - \theta)/\sigma']}, \tag{5}$$

$$\mathcal{D}_a(\mathbf{r}, t) = \frac{1}{\gamma_a^2}\frac{\partial^2}{\partial t^2} + \frac{2}{\gamma_a}\frac{\partial}{\partial t} + 1 - r_a^2\nabla^2, \tag{6}$$

In these equations, all dynamical quantities and operations are subscripted with population indices a (if it is associated with population a) or ab (if it is associated with the connection from b to a). The mathematical symbol ϕ represents the axonal firing rate, ν represents local somato-dendritic coupling strength, V_{ab} denotes the connection ab's contribution to the soma voltage potential, which is summed via $\sum_b V_{ab}$ to give the population soma voltage potential V_a; Q is the population firing rate, which feeds into the axonal firing rate with a time delay

τ. All neural quantities carry spatial and temporal dependencies. The operator parameters α and β are the rise and decay rates of dendritic response, $Q_{\max}$ is the maximum firing rate, θ is the mean firing threshold, σ' is the population standard deviation of the soma voltage relative to threshold, γ is the temporal damping coefficient, and r is the spatial effective range of axonal propagation. This can be schematically summarized in Fig. 2.

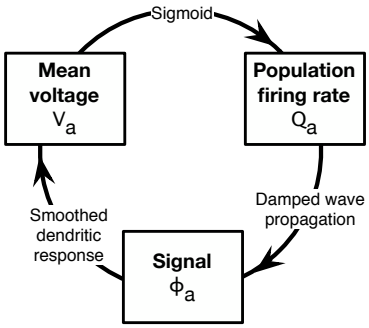

Figure 2: Schematic overview of the key dynamic quantities V_a, Q_a ϕ_{ab} in the neural field model, as governed by Eqs (1)–(3).

Of course, one of the major strengths of the neural field theory of Robinson et al. (2005) is its versatility: by choosing the appropriate neural population model, we may model cortical phenomena (), the corticothalamic system (), or even hemispheric interactions (). Moreover, we may replace the operators above with more sophisticated ones, so as to model additional phenomena. For example, by replacing the constant synapto-dendritic coupling strength ν with a dynamic one, we may model synaptic plasticity (see Sec. 3.2). Or, by replacing the sigmoidal soma firing response with differential equations, we can capture bursting firing rates (see Sec. 3.5).

refs
refs
refs

This versatility and extensibility of the theory presents a challenge in the implementation of the numerical solver. First, we need a program that takes an arbitrary number of neural populations, with possibly different population types and parameters for each population, and arbitrary connections between the populations, where again each connection may be of different types and is de-

scribed by different parameter values. Several factors contribute to making the numerical integration of equations difficult: propagation delays between neural populations result in delay-differential equations that require special handling of temporal history; propagation of action potentials according to a damped wave equation adds two spatial dimensions to the system; periodic boundary conditions must be correctly handled during the integration. The popularity and extensibility of the theory means that the program will involve many developers, who extend and reuse previous code independently; coding level structure needs to be in place to ensure proper collaborative effort and minimizing the chance of error.

We have developed NeuroField as a software package that solves the neural field equations for arbitrary neural population models that is user-friendly and easily extensible. Written in C++, this code has been tested on a range of Linux distributions, Microsoft Windows, and Mac OS X. Front-ends written in MATLAB and Python are also provided for analysis and visualization. In this paper, we present the software as a solution to quickly testing and analyzing neural field models. This paper is structured as follows: in Sec. 2, we present the usage of NeuroField, including the MATLAB and Python interfaces, as well as the numerical algorithm of NeuroField; in Sec. 3, we present example usages of NeuroField in published neural field theory as a showcase of the capabilities of NeuroField. Separate manuals that provide documentations to all user options, extension procedures and coding algorithms are bundled with the NeuroField package.

2. Method

We present NeuroField, a C++ program bundled with MATLAB and Python scripts as a flexible numerical package specialized in solving Eqs (1)–(6). The program reads a human readable configuration file (abbreviated as the config file), which specifies all the simulation information, including an arbitrary population model, (i.e., the number of populations and the connections between

them,) and the type and parameter values of each population and connection. Given the config file, NeuroField creates a human readable output file, which stores the time series of the neural quantities as requested by the user. Optional MATLAB and Python scripts are packaged with NeuroField to aid the launching and output analysis of the simulation. Figure 3 is a flow diagram summarizing the workflow of NeuroField usage. Ample documentation of all usage options, config file options and scripting interfaces are bundled with the software package, so that we will not exhaust these information in this paper. Below, we first explain the structure and algorithm of the program in Sec. 2.1, and in Sec. 2.2, we provide a brief overview of the interfaces of NeuroField.

2.1. Simulation algorithms

The NeuroField program uses standard C++ features including object-oriented programming, templates programming, and the C++ standard library, and has been tested on a range of Linux distributions, Microsoft Windows, and Mac OS X.

To solve Eqs (1)–(6), NeuroField creates objects to handle the time integration of neural dynamical quantities. Each population is associated with a firing response to handle Eqs (2) and (5), while each connection between two populations is associated with the axonal propagation of the instantaneous firing rate of the presynaptic neural population (to handle Eq. (3) and (6)), and the synaptic and dendritic coupling of the postsynaptic population (to handle Eqs (1) and (4)). We may rearrange Eqs (1)–(3) into this form to illustrate the division of the system of partial differential equations:

$$P = \nu_{ab}\phi_{ab}, \qquad \text{Synaptic coupling} \tag{7}$$

$$D_{ab}V_{ab} = P, \qquad \text{Dendritic response} \tag{8}$$

$$Q_a = S_a\big[\sum_b V_{ab}\big], \qquad \text{Firing response} \tag{9}$$

$$\mathcal{D}_{ab}\phi_{ab} = Q_b, \qquad \text{Axonal propagation} \tag{10}$$

which can be schematically summarized in Fig. 4.

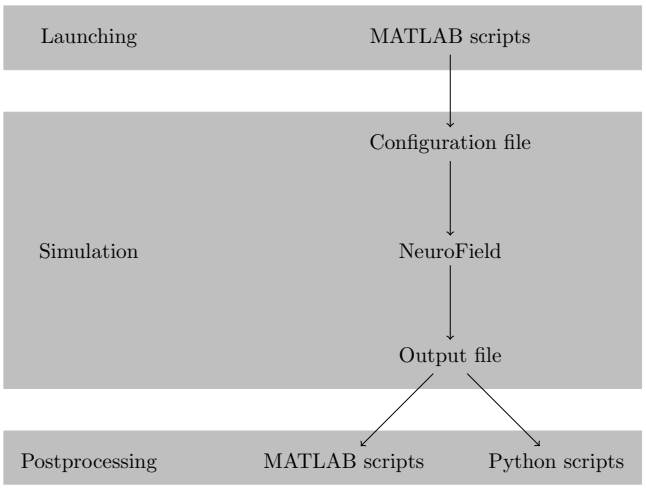

Figure 3: Flow diagram illustrating typical NeuroField usage workflow, which can be categorized into three broad phrases: launching, simulation, and postprocessing. The core is the simulation phase, where NeuroField reads the configuration file (or config file for short), and generates the output file, which stores the simulation results. In the postprocessing phase, the simulation results are further analyzed and useful data, plots, figures may be generated. MATLAB and Python routines are provided to aid postprocessing. Similarly, MATLAB and Python wrapper scripts are provided to aid the creating and running of config files. Since both the config file and output file are human readable, they may be processed manually, and the use of any MATLAB and Python scripts are optional. Section 2.1 elaborates on the simulation phase, while Sec. 2.2 elaborates on the launching and postprocessing phases. In Sec. 3 we present illustrative examples of NeuroField simulation results applied onto recent research.

All numerical integrations involves explicit direct integration, except for Eqs. (10). Much of the complexity of NeuroField lies in the solving of this delay-partial differential equation with spatial dependence. Correct, efficient implementation of this step tends to be the biggest hurdle to implementing a neural field model. NeuroField solves by storing the neural firing rate histories and uses explicit finite differences integration. By default, Euclidean geometry with toroidal topology is used. However, non-Euclidean geometry or alternative boundary conditions may be implemented, so that simulation may be performed on surfaces such those found in structural MRI in the future.

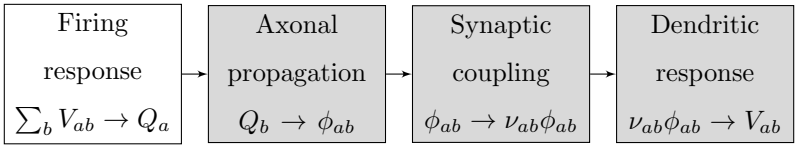

Figure 4: Flow diagram illustrating the NeuroField algorithm, where each equation of Robinson et al. (2005) is handled by a C++ object. Here, a firing response (white fill) is associated with each neural population, while each gray-filled objects are associated with each connection between two populations. Compare with Fig. 2. Importantly, each object may be extended or replaced via C++ inheritance, so that new object types provide extension to the standard Robinson et al. (2005) theory.

Importantly, NeuroField uses the object-oriented feature of C++, so that these classes of objects may be extended or replaced. For example, while in the standard theory of Robinson et al. (2005) the synaptic coupling strength ν_{ab} is constant, we may extend the synaptic coupling class via inheritance to implement synaptic plasticity, which will be elaborated in Sec. 3.2. Another example is to introduce a non-sigmoidal firing response to introduce burst firing, to be elaborated in Sec. 3.5. In addition, a generic 4^{th} order Runge-Kutta solver is implemented for all NeuroField objects, and a generic 9-point stencil for numerical integration involving spatial dependency is available. A developer's guideline is bundled with the NeuroField package, which provide a complete document on the algorithm and code structure of NeuroField, as well as guideline for code development. Thus, extension of the neural field theoretic equations is very simple in NeuroField, where the development time may be in the order of only a few hours.

2.2. User interfaces

As shown in Fig. 3, the core NeuroField C++ program takes in a plain text configuration file (abbreviated as the config file), where the user specifies all the simulation information, and write the simulation result into a output file. A separate user manual, that is bundled with the software package, exhaustively documents all the options for the user interface, so that it will not be repeated

Task	Action	File
Perform simulation	Run NeuroField manually, or via helper scripts	—
Parameter exploration	Change parameter value	In config file
Use alternative object type	Change object type	In config file
Use alternative population model	Change connection matrix and objects	In config file
Create new object type	Write new C++ class	In source code

Table 1: NeuroField tasks requirements, in roughly ascending order of difficulty, so that a task lower in the table often involve also doing the tasks above the entry. Most simulations require only editing of the config file; only the creation of new object types require code writing. Even for the latter, ample documentation, guidelines and working examples minimizes the coding workload. See also Fig. 3 for NeuroField usage workflow and interface.

here. Rather, we provide a brief overview of the interface below, and Sec. 3 presents illustrative results.

Table 1 tabulates typical NeuroField usage tasks, and the corresponding work required. Most tasks involve only editing the config file, and then postprocessing the simulations results, so that only the launching and postprocessing phase in Fig. 3 requires attention. Even so, ample helper scripts exist to aid both the launching and postprocessing phase.

2.2.1. Input

Figure 5 is an example config file for the neural field model to be discussed in Sec. 3.1. The config file starts with a descriptive comment. Then, simulation parameters including simulation duration and time step size, and the number of spatial nodes are specified. This is followed by the specification of a square connection matrix, where a non-zero entry indicate a connection between two populations, so that each connection consists of a axonal propagation from the presynaptic neural population, and a dendritic and firing response from the postsynaptic population. Then, each neural field object, which include firing re-

sponse (`Firing`), dendritic response (`Dendrite`), axonal propagation (`Propag`) and synaptic coupling (`Couple`), is specified via a `identifier: type -` syntax, followed by the designated parameter values, which follow a `parameter: value` syntax. In the example of Fig. 5, the axonal propagation of the excitatory-excitatory self-coupling (`Propag 1`) is a "wave" type propagation, that has a characteristic spatial range of 0.2 m and a damping coefficient of 30 s^{-1}.

2.2.2. Output

The outputting of simulation results are specified in the end of the config file. In Fig. 5, the excitatory population and the excitatory-excitatory axonal propagation is specified to be outputted, so that these two objects will output the time series of their neural dynamical quantities, which in this case will be population soma voltage and axonal firing rate, respectively. To ensure numerical stability, the equations need to be integrated with a very small time step (e.g., 1×10^{-4} s= 10000 Hz). This is much smaller than typically required for analysis; EEG is typically sampled at just 500 Hz or less. To reduce output file size and avoid unnecessarily long postprocessing computation, an output sampling interval can be specified, downsampling the output from NeuroField.

Figure 6 shows an example output file; given it is human readable, standard tools may be used to analyze these data, in addition to the provided MATLAB and Python scripts. In Sec. 2.2.3, we explain that users can manually edit the human-readable interfaces to use NeuroField, and in Sec. 2.2.4, we provide an overview on the MATLAB helper scripts that is bundled with NeuroField as additional interface.

2.2.3. Manual read/write

Given that both the config file and output file are human readable (see Fig. 5 and Fig. 6), manual usage of NeuroField without any helper scripts is possible, by writing the config file and reading the output file. This is particularly true as ample documentation and working example config files exist. Indeed, one convenient way of starting a simulation involves editing an existing config file

```
Example config file for one-population neural field model
Time: 0.15 Deltat: 1e-4
Nodes: 900

Connection matrix:
From:  1  2
To 1:  1  2
To 2:  0  0

Population 1: Excitatory
 Length: 0.5
 Q: 10.98
 Firing: Sigmoid - Theta: 0.013 Sigma: 0.0038 Qmax: 340
  Dendrite 1: alpha: 83 beta: 769
  Dendrite 2: alpha: 83 beta: 769

Population 2: Stimulation
 Length: 0.5
 Stimulus: Pulse - Onset: 0 Node: 465 Amplitude: 1 Width:
     1e-3

Propag 1: Wave - Range: 0.2 gamma: 30
Propag 2:  Map -
Couple 1:  Map - nu: 1e-4
Couple 2:  Map - nu: 1e-4

Output: Node: All Start: 0 Interval: 1e-4
Population: 1
Dendrite:
Propag: 1
Couple:
```

Figure 5: Example config file, for a one-population model receiving a pulse stimulus. Details and result of this system can be found in Sec. 3.1.

Time	Pop.1.V	Dendrite.1.V
	1	1
5.00e-03	-1.020386100923e-03	2.589927678839e-02
1.00e-02	-1.020386100890e-03	2.589927678842e-02

Figure 6: Example output file from NeuroField, where the time series of two neural dynamic quantities are outputted in columns. The first line provides the column heading, while the second line provides the spatial node index. Each subsequent line is outputted at a simulation time interval as the user dictated in the config file.

until it matches the intended simulation environment, and running NeuroField. Since the output file is plain-text, it is very easy to use standard tools to perform postprocessing analysis on the simulation results. For example, a simple PERL script that calls gnuplot exists for simple time series plotting.

2.2.4. MATLAB helper scripts

In addition to manually editing the plain-text files, NeuroField is packaged with MATLAB helper scripts that assists with both the launching of NeuroField and postprocessing of simulation results.

The usage of these MATLAB scripts involves first reading the simulated results from the NeuroField output file into an "NF" MATLAB object, via an `nf_read()` function, and then using the `nf_extract()` function to extract neural dynamic quantities. A number of functions are provided for visualization, including plotting routines and animation routines, and for the calculation of power spectra. In Sec. 3, we provide numerous examples of NeuroField simulations results, where the generation of these figures rely heavily on these helper scripts.

More from Romesh

Python scripts? John Griffiths

```
nf_object = nf_run('neurofield.conf')
time = nf.time;
V = nf_extract( nf, 'Pop.1.V' );
plot(time,V);
grid = nf_grid(nf,'Pop.1.V');
surf(grid(:,:,1));
```

Figure 7: Example usage of MATLAB helper scripts. The `nf_run()` function is used to execute a NeuroField configuration file and read the data into a MATLAB struct. A plot of the time series is generated using the `nf_extract()` function. A surface plot showing spatial variations in potential is generated using the `nf_grid()` function. Other functions including animation and spectral plots are also available.

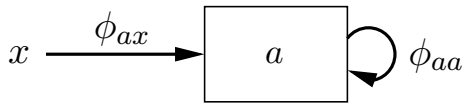

Figure 8: Schematic diagram of a single excitatory population model, which feeds back to itself and receive an external stimulus. This constitutes the simplest neural field model that conforms to Eqs (1)–(6).

3. Results

In this section, we present examples of NeuroField simulation results applied to recent research as a showcase of the capabilities of NeuroField. All plots and figures are simulation results of NeuroField, where the plots are generated by the helper scripts explained in Sec. 2.2.4.

3.1. Single excitatory population model

We first present the simplest neural field model possible, consisting of a single excitatory population that feeds back to itself, and receiving an external stimulus. All neural dynamics are as Eqs (1)–(6), so that this model conforms to the standard Robinson et al. (2005) theory. The config file is shown in Fig. 5, and no extension of NeuroField coding is required. All parameters are taken from

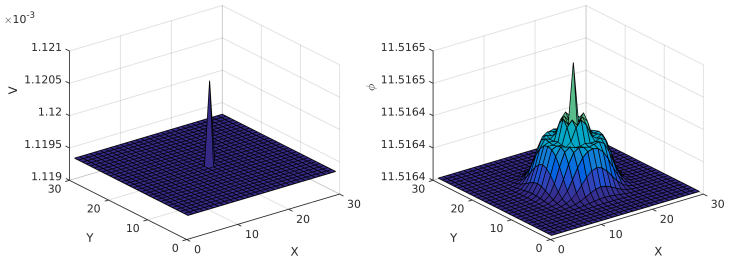

Figure 9: Two plots on the neural activity of the one-population model first introduced in Fig. **??**. The x and y coordinates are spatial coordinates, while the vertical axis is the axonal firing rate, which serves as a measure of neural activity. (a) Neural activity is in a steady state, where the firing rate is spatially homogeneous, while receiving a stimulus in the middle. (b) Neural activity propagates radially outwards, as governed by the damped wave equation (6).

Robinson et al. (2004), with the exception of the axonal propagation parameters, which are tuned to emphasize wave propagation properties for illustrative purposes.

Figure 9 presents the neural dynamics of the system in two snapshots, as generated by the MATLAB helper scripts presented in Sec. 2.2.4. Here, the x and y coordinates are spatial coordinates, and the vertical axis plots the axonal firing rate as a measure of neural activity. Figure 9 (a) show that the neural dynamics of the population is in a steady state and has achieved spatial homogeneity, while receiving a stimulus in the middle. Figure 9 (b) shows the neural dynamics at a later time instance, and we see that the stimulus spread according to damped wave propagation. In addition to generating plots, one MATLAB helper script (see Sec. 2.2.4) also generates movies, which would aid the analysis of wave propagation in this example.

Romesh please check transient vs stable state and stimulation onset (and why are we plotting V and phi?)

Put labels (a) and (b) into figure.

3.2. Single population with synaptic plasticity

The single-excitatory population model of Sec. 3.1 may be extended to model synaptic plasticity. Fung and Robinson (2013) and Fung and Robinson (2014) extended the standard theory of Robinson et al. (2005) by introducing time dependence on the synapto-dendritic coupling strength ν in Eq. (7), to model the experimental results found in transcranial magnetic stimulation (TMS). The coding implementation is done by inheriting the synaptic coupling class, and using the generic 4^{th} order Runge-Kutta solver provided in NeuroField.

Figure 10 shows a schematic diagram for the synaptic plasticity theory of Fung and Robinson (2013) and Fung and Robinson (2014), illustrating how neural activity leads to intraneuronal calcium influx, which signals synaptic plasticity expression, which in turn is modulated by metaplasticity on a longer timescale. Figure 11 presents a comparison between experimental results and NeuroField simulation results in paired associative stimulation (PAS), a variation of TMS experiment. The reproduction of experimental result via the introduction of mean field calcium level and synaptic plasticity into neural field theory demonstrates the flexibility and extensibility of NeuroField.

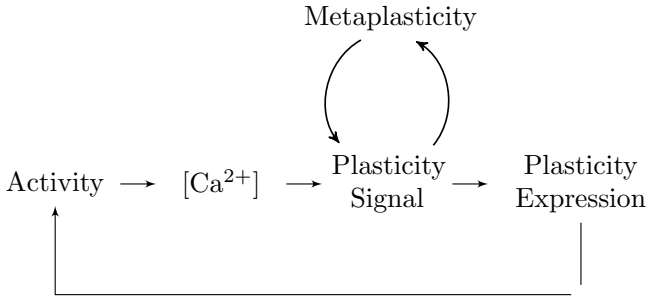

Figure 10: Schematic diagram illustrating the plasticity theory used in Fung and Robinson (2013) and Fung and Robinson (2014). Within the context of neural field theory, we introduce mean field intraneuronal calcium and non-static synaptic coupling strength, so that neural activity leads to intraneuronal calcium influx, which signals synaptic plasticity, which in turn is modulated by metaplasticity. Expression of synaptic plasticity feeds back into neural activity, introducing a loop. This is implemented into NeuroField via C++ inheritance on the synaptic coupling object, and using the generic 4^{th} order Runge-Kutta solver provided. Figure reproduced from Fung and Robinson (2014).

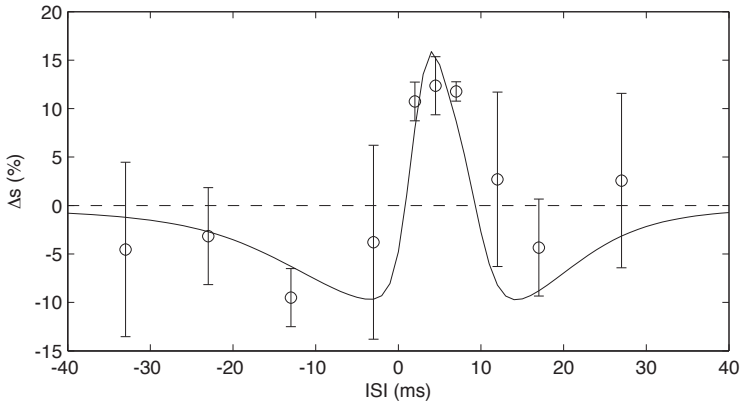

Figure 11: Experimental and simulation results of paired associative stimulation (PAS), where transcranial magnetic stimulation and peripheral electric stimulation is applied with an interstimulus interval (ISI) in between. Depending on the size and ordering of the ISI, different size and direction of synaptic plasticity may be induced. Dots and error bars: mean and standard deviation from experiments; line: NeuroField simulation result. Figure reproduced from Fung and Robinson (2013).

3.3. Corticothalamic model

NeuroField has been extensively tested with a recent neural field corticothalamic model of the brain (Robinson et al., 2002, 2004, 2005, 2001, Rowe et al., 2004) that we have previously used to investigate the alpha rhythm (O'Connor and Robinson, 2004, Robinson et al., 2003), age-related changes to the physiology of the brain (van Albada et al., 2010), evoked response potentials (Kerr et al., 2011, Rennie et al., 2002), seizures (Breakspear et al., 2006), and many other phenomena.

The structure of the model is shown in Fig. 8. Our model predictions are often tested against EEG measures, with $\phi_e(\mathbf{r}, t)$ (as marked in Fig. 8) being associated with the measured EEG signal at position $\mathbf{r}$. Much of our work is in the linear regime and utilizes analytic results for small perturbations to steady states. Steady states can be obtained by setting the differential operators (**??**) and (**??**) to unity, and solving for the the steady state firing rate $\phi_e^{(0)}$ Robinson et al. (2004). However, some phenomena are strongly nonlinear, and these must be solved by numerical integration. We have used NeuroField to model results relating to seizures (**?**), visually evoked potentials (Roberts and Robinson, 2012), and sleep spindles (Abeysuriya et al., 2014, **?**).

The EEG time series is directly produced by NeuroField. The power spectrum can be obtained by FFT, but correct normalization and calculation of the power spectrum including multiple spatial modes can be challenging to implement. With periodic boundary conditions, the EEG power spectrum is given by the summation

$$P(\omega) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \Delta k_x \Delta k_y |\phi_e(\mathbf{k}, \omega)|^2 |F(k) \tag{11}$$

$$k^2 = \left(\frac{2\pi m}{L_x}\right)^2 + \left(\frac{2\pi n}{L_y}\right)^2 \tag{12}$$

where the size of the two-dimensional rectangular cortex is $L_x \times L_y$. In practice, modes with large $\mathbf{k}$ are strongly damped, so only a small number of spatial modes need to be included (**?**). Volume conduction by the cerebrospinal fluid, skull and scalp serves as a low-pass spatial filter whose k dependence is well fitted by

the form (Robinson et al., 2001, Rowe et al., 2004, van Albada et al., 2010, **?**)

$$F(k) = e^{-k^2/k_0^2}, \tag{13}$$

with a low-pass cutoff $k_0 \approx 10\,\mathrm{m}^{-1}$ based on the spherical harmonic head transfer function developed by **?**.

Both summation over spatial modes and volume conduction filtering are implemented in MATLAB helper scripts in the NeuroField toolbox. Figure. **??** shows a three-way comparison between the linear analytic prediction, the nonlinear spectrum generated by NeuroField, and the linear spectrum generated by NeuroField. The model parameters are drawn from Abeysuriya et al. (2014) and correspond to sleep spindles. The linear NeuroField spectrum is computed by changing the population `qresponse` to a linear function, which can simply be selected in the configuration file. There is an excellent match between the linear analytic spectrum and the linear NeuroField spectrum, verifying our implementation of the numerical integrator and power spectrum helper scripts. The nonlinear spectrum shows an additional harmonic peak not present in the linear model.

Figure 12: EEG power spectrum for sleep spindles, for parameters in Abeysuriya et al. (2014). The linear analytic spectrum and linear NeuroField spectrum are an excellent match. The nonlinear NeuroField spectrum shows additional features that arise due to nonlinear effects in the model, in this case corresponding to a nonlinear sleep spindle harmonic (**?**).

3.4. Seizures

Content from Xuelong

3.5. Corticothalamic model with bursting dynamics

Content from Xuelong

3.6. Wilson-Cowan model

3.7. Jansen-Rit model

4. Discussion

We have developed NeuroField to provide an extensible, reliable framework for integrating nonlinear delay differential equations including spatial propagation. NeuroField is aimed for use by researchers who have constructed neural field models of the brain that require numerical integration. In this section, we review some usage and performance considerations.

Most notably, the signals from populations can be associated with local field potentials (LFP) or EEG depending, and these predictions can be directly compared against experimental data. The soma potential or firing rate of the neural populations can be compared to individual neuron data. Changes to synaptic strength can be monitored when simulating neural plasticity. When simulating spatially extended populations, spatial correlations and patterns of activity can also be analyzed.

4.1. White noise stimulus

- White noise requires stocastic DE integrator, effectively Euler (FELIX)
- Noise amplitude depends on grid resolution as this affects the possible bandwidth. Similar features depend on frequency domain power so noise needs to be normalized correctly (ROMESH)

Many neural simulations drive the system using spatially uncorrelated white noise, which is typically denoted in our model as $\phi_n(\mathbf{r}, t)$. This noise signal

can be efficiently generated by sampling from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ at each grid point and time step. However, the power spectrum of this noise signal $\phi_n(\mathbf{k}, \omega)$ depends on σ, the time step Δt, and grid resolution Δx and Δy. Intuitively, the standard deviation σ represents the sum of all of the frequency components $\phi_n(\mathbf{k}, \omega)$ at each point in space and time. In a discrete system, decreasing Δt, Δx or Δy increases the number of frequency components present in the Fourier transform. However, if σ is not adjusted accordingly, the effect will be a decrease in magnitude of each frequency component, changing the normalization of the power spectrum.

Typically we are interested in preserving the value of $|\phi_n(\mathbf{k}, \omega)|^2$, so that the normalization of the power spectrum is invariant when the numerical resolution is changed. From Parseval's theorem, we find that the sum of the discrete Fourier components must equal the expectation value of the original function in the time domain, which is σ^2 for white noise from the Wiener-Khinchin theorem. We therefore obtain

$$\sigma^2 = \sum_{M,N,P} |\phi_n(\mathbf{k}, f)|^2 \Delta f \Delta k_x \Delta k_y \tag{14}$$

$$= \frac{4\pi^2 |\phi_n(\mathbf{k}, f)|^2}{\Delta T \Delta x \Delta y} \tag{15}$$

where

$$\Delta k_x = \frac{2\pi}{L_x} = \frac{2\pi}{M \Delta x} \tag{16}$$

and there are M, N, and P grid points in each dimension x, y and t, respectively. In our model formulation, the analytic power spectrum is single-sided (positive frequencies only) and depends on $|\phi_n(\mathbf{k}, \omega)|^2$. This quantity is equal to $\pi |\phi_n(\mathbf{k}, f)|^2$ noting that Eq. (**??**) refers to both positive and negative frequencies. Thus we finally have the result

$$\sigma = \sqrt{\frac{4\pi^3 |\phi_n(\mathbf{k}, \omega)|^2}{\Delta T \Delta x \Delta y}} \tag{17}$$

which relates the standard deviation of the Gaussian distribution used to generate the noise in the time domain, to the power spectral density of the noise in

Platform	Processor	Clock speed	Runtime
Windows	i5-2500k	4.8 GHz	14.3 s
Arch Linux	i5-760	2.8 GHz	12.0 s
RHEL6	Xeon E5-2698	2.3 GHz	10 s
Mac OS 10.10	i5-4260U	1.4 GHz	11.4s

Table 2: Example NeuroField run times for a typical NeuroField run. The configuration file corresponds to the corticothalamic model, simulating 15 seconds sampled at 10000 Hz with 144 grid points and outputting ϕ_e at 200 Hz for all nodes. NeuroField was compiled natively on each platform using GCC 4.8 or higher with all applicable performance flags selected, with the exception of Windows where the MinGW was used for cross-compiling. TODO: Double check with John Palmer how to interpret these - to me these seem like they are memory-bound rather than CPU bound. Tests were run using eirs_eyes_closed.conf from the repository

the frequency domain. This enables the power spectral density to be correctly matched between the analytic spectrum and NeuroField, as shown in Fig. **??**.

For ease of use, users can specify the desired power spectral density of the white noise stimulus in the configuration file, as well as the physical dimensions of each population. NeuroField will then automatically calculate the appropriate Gaussian distribution to correctly match the desired power spectral density.

4.2. Performance

- Some numbers about the runtime and memory requirements of NeuroField (FELIX)
- Note that the memory requirements scale with the grid size, and the grid size depends on Lx and the propagator lengths (automatically enforced) (FELIX)
- Also that the delays in the system cause O(n) increases in memory usage (FELIX)

Additional considerations?

5. Acknowledgements

This work was supported by the Australian Research Council, National Health and Medical Research Council (through the Center for Integrated Research and Understanding of Sleep), and the Westmead Millennium Institute.

6. References

Abeysuriya RG, Rennie CJ, Robinson PA. Prediction and verification of nonlinear sleep spindle harmonic oscillations. J Theor Biol 2014; 344:70–77.

Breakspear M, Roberts JA, Terry JR, Rodrigues S, Mahant N, Robinson PA. A unifying explanation of primary generalized seizures through nonlinear brain modeling and bifurcation analysis. Cereb Cortex 2006; 16:1296–1313.

Deco G, Jirsa VK, Robinson PA, Breakspear M, Friston KJ. The dynamic brain: from spiking neurons to neural masses and cortical fields. PLoS Comput Biol 2008; 4:e1000092.

Fung PK, Robinson PA. Neural field theory of calcium dependent plasticity with applications to transcranial magnetic stimulation. J Theor Biol 2013; 324:72–83.

Fung PK, Robinson PA. Neural field theory of synaptic metaplasticity with applications to theta burst stimulation. J Theor Biol 2014; 340:164–176.

Kerr CC, Rennie CJ, Robinson PA. Model-based analysis and quantification of age trends in auditory evoked potentials. Clin Neurophysiol 2011; 122:134–147.

O'Connor SC, Robinson PA. Spatially uniform and nonuniform analyses of electroencephalographic dynamics, with application to the topography of the alpha rhythm. Phys Rev E 2004; 70:11911.

Pinotsis DA, Moran RJ, Friston KJ. Dynamic causal modeling with neural fields. NeuroImage 2012; 59:1261–1274.

Rennie CJ, Robinson PA, Wright JJ. Unified neurophysical model of EEG spectra and evoked potentials. Biol Cybern 2002; 86:457–471.

Roberts JA, Robinson PA. Quantitative theory of driven nonlinear brain dynamics. NeuroImage 2012; 62:1947–1955.

Robinson PA, Rennie CJ, Rowe DL. Dynamics of large-scale brain activity in normal arousal states and epileptic seizures. Phys Rev E 2002; 65:41924.

Robinson PA, Rennie CJ, Rowe DL, O'Connor SC. Estimation of multiscale neurophysiologic parameters by electroencephalographic means. Hum Brain Mapp 2004; 23:53–72.

Robinson PA, Rennie CJ, Rowe DL, O'Connor SC, Gordon E. Multiscale brain modelling. Phil Trans Roy Soc B 2005; 360:1043–1050.

Robinson PA, Rennie CJ, Wright JJ, Bahramali H, Gordon E, Rowe DL. Prediction of electroencephalographic spectra from neurophysiology. Phys Rev E 2001; 63:21903.

Robinson PA, Whitehouse RW, Rennie CJ. Nonuniform corticothalamic continuum model of electroencephalographic spectra with application to split-alpha peaks. Phys Rev E 2003; 68:21922.

Rowe DL, Robinson PA, Rennie CJ. Estimation of neurophysiological parameters from the waking EEG using a biophysical model of brain dynamics. J Theor Biol 2004; 231:413–433.

van Albada SJ, Kerr CC, Chiang AKI, Rennie CJ, Robinson PA. Neurophysiological changes with age probed by inverse modeling of EEG spectra. Clin Neurophysiol 2010; 121:21–38.