NFTsim: Theory and Simulation of Multiscale Neural Field Dynamics

P. Sanz-Leon [1,2,*,¤], P. A. Robinson[1,2], S. A. Knock [1,2], P. M. Drysdale[1], R. G. Abeysuriya[1,2,3], P. K. Fung[1,2,3], C. J. Rennie[1], X. Zhao[1,2]

1 School of Physics, University of Sydney, Sydney, New South Wales, Australia
2 Center for Integrative Brain Function, University of Sydney, Sydney, New South Wales, Australia
3 Oxford Centre for Human Brain Activity, Wellcome Centre for Integrative Neuroimaging, Department of Psychiatry, University of Oxford, Oxford, United Kingdom
4 Downstate Medical Center, State University of New York, Brooklyn, United States

¤Current Address: School of Physics, University of Sydney, New South Wales, Australia
* `paula.sanz-leon@sydney.edu.au`

Abstract

A user ready, portable, documented software package, *NFTsim*, is presented to facilitate numerical simulations of a wide range of brain systems using continuum neural field modeling. *NFTsim* enables users to simulate key aspects of brain activity at multiple scales. At the microscopic scale, it incorporates to define characteristics of local interactions between cells, neurotransmitter effects, synaptodendritic delays and feedbacks. At the mesoscopic scale, it incorporates information about medium to large scale axonal ranges of fibers, which are essential to model dissipative wave transmission and to produce synchronous oscillations and associated cross-correlation patterns as observed in local field potential recordings of active tissue. At the scale of the whole brain, *NFTsim* allows for the inclusion of long range pathways, such as thalamocortical projections, when generating macroscopic activity fields. The multiscale nature of the neural activity produced by *NFTsim* enables the modeling of resulting quantities measurable via various neuroimaging techniques. In this work, we give a comprehensive description of the design and implementation of the software. Due to its modularity and flexibility, *NFTsim* enables the systematic study of an unlimited number of neural systems with multiple neural populations under a unified framework and allows for direct comparison with analytic and experimental predictions. The code is written in C++ and bundled with Matlab routines for a rapid quantitative analysis and visualization of the outputs. The output of *NFTsim* is stored in plain text file enabling users to select from a broad range of tools for offline analysis. This software enables a wide and convenient use of powerful physiologically-based neural field approaches to brain modeling. *NFTsim* is distributed under the Apache 2.0 license.

Introduction

The brain is a multiscale physical system, with structures ranging from the size of ion channels to the whole brain, and timescales running from sub-millisecond to multi-year durations. When modeling brain structure and dynamics, it is thus necessary to choose models that are appropriate to the scales of the phenomena involved. These range from microscale models of individual neurons and their substructures, through network-level models of discrete neurons, to population-level mesoscale and macroscale neural mass and neural field models that average over microstructure and apply from local brain areas up to the whole brain. Many useful results can be obtained analytically from models at various scales, either generally or when applied to specific brain systems and phenomena. However, in order to minimize approximations and make realistic predictions in complex situations, numerical simulations are usually necessary. The purpose of this paper is to present a neural field software package, *NFTsim*, that can simulate scales from a few tenths of a millimeter and a few milliseconds upward, thereby making contact with experiments [1–6] and other classes of simulations over this range [7, 8].

No one type of brain model is optimal at all scales. For example, single neuron models abound in neuroscience, and can include a large number of biophysical effects with relatively few approximations. Many such models have also been used to study networks of interconnected neurons with varying degrees of idealization, thereby revealing a huge number of insights [9–11]. However, several key problems arise as network size grows: (i) the computational resources required become prohibitive, meaning that simulations can often only be carried out in physiologically unrealistic scenarios, typically with idealized neurons, which may be quantitatively and/or qualitatively inappropriate for the real brain; (ii) it is increasingly difficult to measure and assign biophysical parameters to the individual neurons — e.g., individual connectivities, synaptic strengths, or morphological features, so large groups of neurons are typically assigned identical parameters, thereby partly removing the specificity of such simulations; (iii) analysis and interpretation of results, such as large collections of timeseries of individual soma voltages, becomes increasingly difficult and demanding on storage and postprocessing; (iv) emergence of collective network-level phenomena can be difficult to recognize, sometimes leading to single-neuron dynamics being overemphasized; (v) the scales of these simulations are well suited to relate to single-neuron measurements, and microscopic pieces of brain tissue, but are distant from those of noninvasive imaging modalities such as functional magnetic resonance imaging (fMRI), electroencephalography (EEG), and magnetoencephalography (MEG) [12–14], which detect signals that result from the aggregate activity of large numbers of neurons; and (vi), inputs from other parts of the brain are neglected, meaning that such models tend to represent isolated pieces of neural tissue.

At the level of neurons and neuronal networks [15, 16], software is abundant, including BRIAN, NEURON, GENESIS, and NeoCortical Simulator [17–22]. A detailed review of tools and implementation strategies for spiking neural network simulations can be found in [9].

At the largest scales, neural mass models average the properties of huge numbers of neurons into those of a single population, without taking account of its spatial aspects. This enables the temporal dynamics of whole neural populations to be approximated, but information on individual neurons and spatial dynamics and patterns is not tracked. This scale can be used to study whole-brain phenomena such as generalized seizures, if time delays within each mass can be neglected. This approach has been used to treat relatively coarse-grained networks of interacting brain regions, each modeled as a neural mass. However, it is rare to see careful attention paid to the need for these representations to approach the correct spatiotemporal continuum limit as the size of the regions decreases [10, 23, 24], thereby throwing some such discretizations into question. Of course, neural structure is not truly continuous, but its granularity is at a far finer scale than that of the discretizations just mentioned.

Above the single-neuron scale and extending to encompass the neural-mass limit as a special case, neural field approaches retain spatial information in a continuum limit in which properties such as firing rate and soma voltage are viewed as local averages over many neurons, and can vary from point to point, and as functions of time; when correctly discretized, neural mass models are a limiting case of the more general neural fields and should not be viewed as a separate category. Neural fields approximate rate-based models of single neurons from the small scale, while retaining relative timings between neural inputs and outputs. Simultaneously, they self-consistently add spatial structure that is neglected in neural mass models. Hybrid models with features of both neural fields and spiking neurons have also been developed and used to clarify the relationship between these approaches [3], or to enable single-neuron dynamics to be influenced by average neural fields [25], but we do not discuss these classes of models further here.

The issues discussed in the preceding paragraphs are closely analogous to ones that arise in other branches of physics. Specifically, no single model can cover all scales at once. Rather, a hierarchy of models is needed, from the microscale to the macroscale, each relating predictions to measurements at its operational scale. This yields tractable models that can be interpreted in terms of concepts and measurements that apply at the appropriate scales for a given phenomenon. Importantly, each model needs to be related to the ones at nearby scales, especially by making complementary predictions at overlapping scales of common applicability. By analogy, molecular dynamics approaches and statistical mechanics (akin to single neuron approaches) are widely used to track molecules at the microscopic scale, but large-scale theories like thermodynamics and fluid mechanics (akin to neural mass and neural field methods) are more useful and tractable for macroscopic phenomena, and their predictions can be more easily interpreted. At intermediate scales, nonequilibrium thermodynamics and fluctuation theory meet with statistical mechanics and molecular approaches to make complementary predictions of the same phenomena; so that consistency of the various approaches in their common domain can be established. Although molecular-level and spiking-neuron approaches are more fundamental, they are not practical at large scales, and yield results that have to be reinterpreted in terms of larger-scale observables in any case. Conversely, thermodynamic and neural-field approaches fail at spatial and temporal scales that are too short to justify the relevant averaging over a system's microscopic constituents.

Because of the wide range of scales that can be incorporated in neural field theory, it provides useful macroscopic predictions and can reach down to mesoscopic scales that now overlap with those that can be simulated with neuron-level methods. This provides a range of common applicability on scales of around 1 mm, or slightly less, where complementary predictions can be made and tested – an overlap that will increase as microscopic simulations increase in scale. Equally significantly, quantitative neural field predictions can readily be made of quantities observable by EEG, MEG, fMRI, electrocorticography (ECoG), and other imaging technologies, by adding the biophysics of these signals, measurement procedures, and postprocessing [26–29]. This enables predictions of a single brain model to be tested against multiple phenomena in order to better determine the relevant physiological parameters. The importance of this point cannot be overemphasized — underdetermination of theories and models is rife in neuroscience, and ones that are tested against only one or a few phenomena, and/or whose predictions and parameters are not expressed in quantitative physical units, must at best be treated as being highly provisional, subject to further verification and parameter constraint.

As an illustration of the versatility of NFT approaches, we note that the particular NFT on which the present *NFTsim* software is based has been extensively applied and quantitatively tested against experiments, including EEG, evoked response potentials

(ERPs), ECoG, age-related changes to the physiology of the brain, sleep and arousal dynamics, seizures, Parkinson's disease, and other disorders, transcranial magnetic stimulation (TMS), synaptic plasticity phenomena [1, 6, 26–38]. Indeed, one of the major strengths of this NFT is its versatility: within the same framework we can express different models to study purely cortical phenomena, the corticothalamic system, basal ganglia, sleep dynamics, or the visual cortex, among an essentially unlimited number of other applications [1, 26–28, 30, 32, 34–37, 39–42]. This NFT has also been clearly linked to its neural mass limit [35], to hybrid spiking-field approaches [3, 34, 43], and to network and connection-matrix representations of spatial structure in the brain [44], usually obtained via fMRI.

We stress that the NFT embodied in *NFTsim* is not the only possibility. Other NFTs have been developed and applied by numerous authors [45–53], each of which has been applied to one or more physical situations in these and subsequent publications. This list is not exhaustive, since the present work is not intended as a review, but more examples can be found in [10], [24], and [54]. Notably, most of these NFTs can be expressed in the notation of the present paper, and can thus be simulated with the *NFTsim* software described below. Some of these previous neural field models leave out physical effects that are included in *NFTsim*, while others include additional features that remain to be incorporated in a future version of the code.

A few software packages are available to model neural masses and neural fields: [7] developed a neuroinformatics platform for large-scale brain modeling in terms of a network of linked neural masses with anatomically specified cortical geometry [54], long-range connectivity, and local short-range connectivity that approximates the continuum limit when it is Gaussian and homogeneous [23]. While the mathematical framework described in [54] allows for neural field models to be treated using realistic geometry on nonregular grids, a user-ready implementation is not currently available. Similarly, the Brain Dynamics Toolbox [55] provides tools for network-based and continuum models of brain dynamics. The most recent simulation tool for spatiotemporal neural dynamics is the Neural Field Simulator [8], which allows for study of a range of 2D neural field models on a square grid. However, this software does not allow for either the simulation of neural field models with heterogeneous parameters or with multiple populations.

To address the need for research-ready NFT simulation tools with direct application to the study of large-scale brain phenomena, this paper introduces and describes *NFTsim*, a software package that solves neural field equations expressed in differential form for simulating spatially extended systems containing arbitrary numbers of neural populations.

Neural Field Theory

Neural field theory (NFT) treats multiscale brain activity by averaging neural quantities such as firing rate, soma voltage, and incoming and outgoing activity over multiple neurons. The scales over which neural field models average must be sufficient to represent large numbers of neurons and spikes, but can still be small enough to resolve quite fine structure in the brain and its activity. *NFTsim* allows an arbitrary number p of spatially extended populations of neurons to be simulated. Each of these can be distinguished by its location (e.g., belonging to the cortex or a particular nucleus) and its neural type (e.g., pyramidal excitatory, interneuron). To model a particular system, we must specify the neural populations and the connections between them, including self-connections within a population. If we introduce position and time coordinates $\mathbf{r}$ and t, the main macroscopic variables that describe the activity of neural populations a and their interaction with other populations b are: the incoming, axonal spike-rate fields

$\phi_{ab}(\mathbf{r},t)$ arriving at population a at $(\mathbf{r},t)$ from population b, the dendritic potentials V_{ab}, the mean soma potential $V_a(\mathbf{r},t)$, the mean firing rate $Q_a(\mathbf{r},t)$, and the axonal fields $\phi_{ca}(\mathbf{r},t)$ propagating to other populations c from population a. Figure 1 illustrates the interactions of these quantities: (i) synaptodendritic dynamics involving the incoming axonal fields ϕ_{ab} to yield the potentials V_{ab}; (ii) dendritic summation and soma charging processes to yield the soma potential V_a; (iii) generation of pulses Q_a at the axonal hillock, and (iv) axonal propagation of pulses ϕ_{ac} within and between neural populations [1]. The following subsections present a review of the equations describing these physiological processes, while Table 1 summarizes the quantities and symbols used in NFT and their SI units.

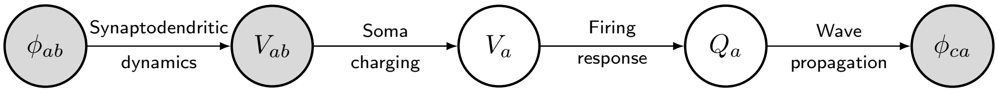

Fig 1. Schematic of the dynamical processes that occur within and between neural populations. Gray circles are quantities associated with interactions between populations (i.e., a and b), while white circles are quantities associated with a population (i.e., a or b). Spike-rate fields ϕ_{ab} arriving at neurons of type a from ones of type b are modulated by the synaptic dynamics, and undergo dendritic dynamics to produce postsynaptic subpotentials V_{ab}. These contributions are linearly summed in the dendritic tree, eventually resulting in charging currents at the soma that give rise to the soma potential V_a, after allowing for capacitive effects and leakage. Action potentials generated at the axonal hillock are averaged over a population of neurons. Then, when the mean soma voltage exceeds a threshold, the mean firing rate Q_a of the population is obtained via a nonlinear response function. Finally, the pulses propagate away across the axonal tree and the dendrites of the receiving population c as the set of average spike-rate fields ϕ_{ca}. Note that self-connections with $b = a$ or $c = a$ are included.

Synaptodendritic Dynamics and the Soma Potential

When spikes arrive at synapses on neurons of population a from a neural population b, they initiate neurotransmitter release and consequent synaptic dynamics, like transmembrane potential changes, followed by dendritic propagation of currents that result in soma charging and consequent modifications of the soma potential. Each of these processes involves its own dynamics and time delays and results in low pass filtering and temporal smoothing of the original spike until the soma response is spread over a time interval that is typically tens of ms, exhibiting a fast rise and an approximately exponential decay [3, 56].

If the overall synaptodendritic and soma responses are linear, which is the most common approximation in the literature [2, 30, 57], the total soma potential V_a is the sum of subpotential contributions V_{ab}, which are components of perturbation to the dendritic transmembrane potential, arriving at each type of dendritic synapse ab. The subscript a denotes the receiving population and b denotes the neural population from which the incoming spikes originate, distinguished by its source and the neurotransmitter type. The subpotentials V_{ab} at a particular location comprise contributions from both the wave fields ϕ_{ab} from other internal populations b and inputs ϕ_{ax} from external populations x [58]; the external inputs are often split into a uniform mean nonspecific excitation and a specific excitation due to structured stimuli. Thus we write the total mean cell body potential as the sum of postsynaptic subpotentials

$$V_a(\mathbf{r},t) = \sum_b V_{ab}(\mathbf{r},t), \tag{1}$$

where the subscript b distinguishes the different combinations of afferent neural type and synaptic receptor and all the potentials are measured relative to resting [2].

The overall effect of synaptodendritic dynamics and soma charging in response to an incoming weighted pulse-rate field ϕ_{ab} are well described by an impulse response kernel $L_{ab}(t-t')$

$$V_{ab}(\mathbf{r},t) = \int_{-\infty}^{t} L_{ab}(\mathbf{r},t-t')\nu_{ab}(\mathbf{r},t)\phi_{ab}(\mathbf{r},t'-\tau_{ab})dt', \tag{2}$$

$$\nu_{ab}(\mathbf{r},t) = N_{ab}(\mathbf{r},t)s_{ab}(\mathbf{r},t), \tag{3}$$

where ϕ_{ab} is the average rate of spikes arriving at a from population b; the time delay τ_{ab} is nonzero when a and b are in anatomical structures that are separated by a nonzero distance [2]. In Eq. (3), N_{ab} is the mean number of connections of mean time-integrated synaptic strength s_{ab} to a cell of type a from cells of type b. In [2], L_{ab} is a nonnegative response kernel with

$$\int_{-\infty}^{\infty} L_{ab}(\mathbf{r},u)du = 1, \tag{4}$$

and $L_{ab}(\mathbf{r},u) = 0$ for $u < 0$ to express causality. Note that τ_{ab} are not the only time delays in the system. Propagation delays within a single structure, such as the cortex, are handled by accounting for axonal propagation, as described in section *Propagation of Axonal Pulse-rate Fields.* In *NFTsim* $L_{ab}(\mathbf{r},t)$ is defined as

$$L_{ab}(\mathbf{r},t) = \begin{cases} \dfrac{\alpha_{ab}\beta_{ab}}{\beta_{ab}-\alpha_{ab}}\left\{\exp[-\alpha_{ab}t] - \exp[-\beta_{ab}t]\right\}, & \alpha \neq \beta, \\ \alpha_{ab}^2 t \exp[-\alpha_{ab}t], & \alpha = \beta, \end{cases} \tag{5}$$

for $t \geq 0$, with $L_{ab}(\mathbf{r},t) = 0$ for $t < 0$ and the $\mathbf{r}$-dependence of the positive constants α and β has been omitted for compactness. These quantities parametrize the decay rate and rise rate of the soma response, respectively, and $\beta \geq \alpha$ is assumed without loss of generality. The temporal profile of the dendritic response function is illustrated in Fig. 2. This function peaks at $t = \ln(\beta/\alpha)/(\beta-\alpha)$ for $\alpha \neq \beta$; if $\alpha = \beta$, the peak is at $t = 1/\alpha$. In addition, there are two special cases of Eq. (5): (i) if either $\alpha \to \infty$ or $\beta \to \infty$, then L_{ab} becomes a single exponential function in which only one of the characteristic timescales dominates; and, (ii) if $\alpha = \beta = \infty$, then the kernel reduces to the impulse $L_{ab}(\mathbf{r},t) = \delta(\mathbf{r},t)$.

The convolution in Eq. (2) can be re-expressed as

$$D_{ab}V_{ab}(\mathbf{r},t) = \nu_{ab}\phi_{ab}(\mathbf{r},t-\tau_{ab}), \tag{6}$$

where the differential operator D_{ab} is given by

$$D_{ab}(\mathbf{r},t) = \frac{1}{\alpha_{ab}\beta_{ab}}\frac{d^2}{dt^2} + \left(\frac{1}{\alpha_{ab}} + \frac{1}{\beta_{ab}}\right)\frac{d}{dt} + 1. \tag{7}$$

In some previous work [56] a special approximation has been used where α_{ab} and β_{ab} are independent of b and are thus treated as effective values, representing an average over different receptor time constants. Under this approximation Eq. (6) becomes

$$D_a V_a(\mathbf{r}, t) = \sum_b \nu_{ab} \phi_{ab}(\mathbf{r}, t - \tau_{ab}) \tag{8}$$

All the aforementioned cases and forms of the operators (differential and integral) are implemented in *NFTsim*.

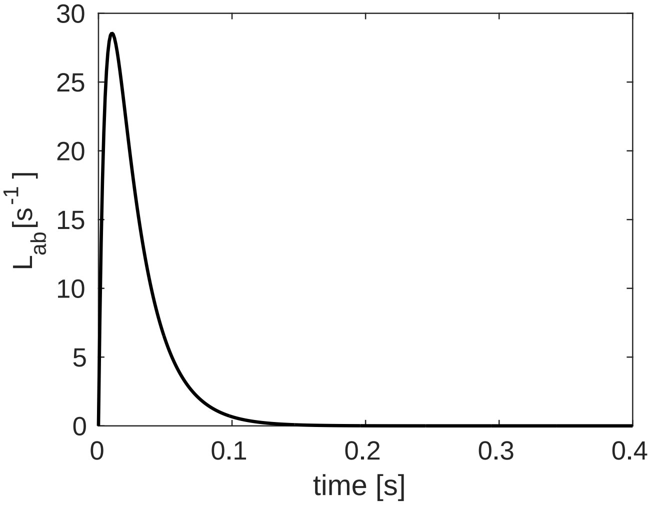

Fig 2. Dendritic response function. The response to a delta-function input, via L_{ab} as defined in Eq. (5), for decay rate parameter $\alpha_{ab} = 45$ s^{-1} and rise rate parameter $\beta_{ab} = 185$ s^{-1}. This function peaks at $t = \ln(\beta/\alpha)/(\beta - \alpha)$ for $\alpha \neq \beta$.

Generation of Pulses

Pulses (i.e., spikes or action potentials) are produced at the axonal hillock when the soma potential exceeds a threshold potential $\theta_a(\mathbf{r}, t)$. When we consider the mean response of a population of neurons to a mean soma potential we must bear in mind that each neuron has slightly different morphology and environment. Hence, they respond slightly differently in the same mean environment. This has the effect of blurring the firing threshold and the resulting overall population response function is widely approximated by the nonlinear form [48]

$$Q_a(\mathbf{r}, t) = S[V_a(\mathbf{r}, t) - \theta_a(\mathbf{r}, t)], \tag{9}$$

where θ_a is the mean threshold potential of population a and S_a is a function that increases monotonically from zero at large negative V_a to a maximum firing rate $Q_a^{\max}$ at large positive V_a, with the steepest increase concentrated around the mean threshold θ_a. *NFTsim* employs by default the nonlinear sigmoid response function

$$S_a[V_a(\mathbf{r}, t), \theta_a(\mathbf{r}, t)] = \frac{Q_a^{\max}}{1 + \exp[-\{V_a(\mathbf{r}, t) - \theta_a(\mathbf{r}, t)\}/\sigma'_a(\mathbf{r}, t)]}, \tag{10}$$

where $\sigma_a = \sigma'_a \pi/\sqrt{3}$ is the population standard deviation of the soma voltage relative to the threshold. If the function in Eq. (10) is linearized to consider small perturbations around a steady state of the system [2, 31], one finds the linear response function

$$Q_a = Q_a^{(0)} + \rho_a [V_a - V_a^{(0)}] \tag{11}$$

where $Q_a^{(0)}$ and $V_a^{(0)}$ are the relevant steady-state values and $\rho_a = dQ_a/dV_a$, is the slope of the sigmoid function, evaluated at $V_a^{(0)}$ [2]. This linear population response function is also implemented in *NFTsim* and other functional forms can be defined as well.

Propagation of Axonal Pulse-rate Fields

The propagation of the pulses $Q_b(\mathbf{r}, t)$ in each population b generates an outgoing mean field ϕ_{ab} that propagates via axons to the population a at other locations. In general, this propagation can depend on both the initial and final populations, and can incorporate arbitrary nonuniformities and a range of propagation velocities via propagator methods, for example [59, 60]. However, considerable theoretical and experimental work has shown that, to a good approximation, the mean field of axonal signals in a smoothly structured neural population propagates approximately as if governed by an isotropic damped wave equation [2, 47, 49, 52, 53, 61–66]. In *NFTsim* we implement the widely used equation

$$\mathcal{D}_{ab}\phi_{ab}(\mathbf{r}, t) = Q_b(\mathbf{r}, t), \tag{12}$$

with

$$\mathcal{D}_{ab} = \left[\frac{1}{\gamma_{ab}^2}\frac{\partial^2}{\partial t^2} + \frac{2}{\gamma_{ab}}\frac{\partial}{\partial t} + 1 - r_{ab}^2\nabla^2\right], \tag{13}$$

where $\gamma_{ab} = v_{ab}/r_{ab}$ is a temporal damping coefficient, r_{ab} is the spatial effective axonal range, v_{ab} is the axonal velocity [2, 53, 62–66], and ∇^2 is the Laplacian operator. Equations (12) and (13) constitute the two-dimensional generalization of the telegrapher's equation [2, 53, 67]. More generally, γ_{ab}, r_{ab}, and v_{ab} can be functions of position. If the special case of spatially uniform activity is considered, the Laplacian operator has no effect and can be omitted from (13). This special case results in the harmonic operator

$$\mathcal{D}_{ab} = \left[\frac{1}{\gamma_{ab}^2}\frac{\partial^2}{\partial t^2} + \frac{2}{\gamma_{ab}}\frac{\partial}{\partial t} + 1\right]. \tag{14}$$

We stress that this is not the same as using a local neural mass model because the damping parameter γ_{ab} depends on spatial propagation. To obtain the neural mass limit, one also needs to set the spatial ranges $r_{ab} = 0$ so γ_{ab} becomes infinite and

$$\mathcal{D}_{ab}(\mathbf{r}, t) = 1. \tag{15}$$

This yields

$$\phi_{ab}(\mathbf{r}, t) = Q_b(\mathbf{r}, t), \tag{16}$$

which is termed the *local interaction approximation* [2, 50].

The parameter r_{ab} in the propagators in Eqs (13) and (14) encompasses coordinate divergence of axons traveling to the target population a from the source population b and the extent of dendritic arborization of the target population a, and thus $r_{ab} \neq r_{ba}$ in general [79].

Design and Implementation of *NFTsim*

This section presents a comprehensive description of *NFTsim*. The subsection *General Workflow* gives an overview of the typical usage workflow of *NFTsim*. The subsection *Classes and their Interactions* describes the main *NFTsim* classes, which represent the biophysical processes and quantities introduced in *Neural Field Theory*. Next, subsection *Input-Output* illustrates with examples how to specify a model in the

Table 1. NFT quantities and associated SI units.

Symbol	Description	Units
$Q_a^{\max}$	Maximum firing rate	s^{-1}
γ_{ab}	Damping rate	s^{-1}
v_{ab}	Wave velocity	$\mathrm{m\ s}^{-1}$
r_{ab}	Mean axonal range	m
θ_a	Mean neural firing threshold	V
σ'_a	Standard deviation of the firing threshold	V
α_{ab}	Mean dendritic response decay rate	s^{-1}
β_{ab}	Mean dendritic response rise rate	s^{-1}
ν_{ab}	Synaptic coupling strength	V s
τ_{ab}	Long range time delay	s
ϕ_{ab}	Axonal field	s^{-1}
Q_a	Mean firing rate	s^{-1}
V_{ab}	Subpotential	V
V_a	Mean soma potential	V

Symbols used in NFT, associated physical quantities and their SI units. Double subscripts ab mean that the target population is a and the source population is b.

input configuration file to *NFTsim* and how to interpret the output file. In addition, subsection *Numerical Methods, Considerations, and Constraints* elaborates on the numerical approaches and constraints used to correctly solve the equations of neural field models while attaining numerical accuracy and stability. Lastly, subsection *Analysis and Visualization* presents a simple example of how to run a simulation, and analyze and visualize the results using the auxiliary Matlab module `+nf`. A list of the available functions in this module is presented in Table 3.

The typographic conventions used in the remainder of this text are that: (i) all computer code is typeset in a typewriter font; and (ii) code snippets are framed by horizontal lines with line numbers on the left.

General Workflow

A typical *NFTsim* workflow consists of three broad phases: configuration; simulation; and postprocessing. The first phase involves writing a configuration file that specifies the neural field model as well as other parameters required to run a simulation. This file is a human readable plain text file with the extension `.conf`. Once a configuration is specified the simulation can be launched by invoking the `nftsim` executable, either directly via a shell (eg. *bash*) terminal

```
user@host$ nftsim -i <my-model.conf> -o <my-model.output>
```

or indirectly via the `nf.run` Matlab function. In the simulation phase, *NFTsim* reads the configuration file, specified after the flag `-i`, builds the objects of the specified model, runs the simulation and writes the output file, which contains the timeseries of the neural quantities requested in the configuration file. The name of the output file can be specified using the flag `-o` and must have the extension `.output`. In the absence of an output file name, *NFTsim* uses the input file name with the extension `.output`. For autogenerated output file names, the flag `-t` can be used to append a string to the output file name of the form `_YYYY-MM-DDTHHMMSS`, which follows the standard ISO 8601 [73] to represent date and time. In the postprocessing phase, the simulation results can be analyzed offline and visualized with the functions provided in the Matlab module

`+nf`.

Code Architecture

Neural field models can be decomposed into a small number of objects, that represent their various parts. Each object has intrinsic properties that, in turn, can be well represented as classes, each of which is a set of elements having common attributes different from other sets, using object oriented programming. *NFTsim* classes have been implemented in C++ (C++11 standard) [74, 75].

The most prominent components of neural field models are populations, synaptic connections, and propagators. Each of these components (or objects) is described by a main base class with properties specific to a group of objects. Derived classes are defined via the mechanism of class inheritance which allows for: (i) the definition of class in terms of another class; (ii) the customization of different parts of the system being modeled; and (iii) the extension of the functionalities of the library. For instance, a base class describing propagators has properties such as axonal range and axonal velocity. These properties are common to different propagators (derived classes) such as the wave propagator in Eq. (13) or the harmonic propagator in Eq. (14), and are inherited from the base class. However, the optimal method to solve each form of propagation may vary and thus each propagator-derived class can have its own solver. Furthermore, there are auxiliary base classes that define additional properties of the main classes described above. These auxiliary classes embody processes like dendritic dynamics, soma charging, firing response, external stimuli, and anatomical time delays.

Thanks to this modular architecture, *NFTsim* allows for the specification of models with (i) an arbitrary number of neural populations, of different types and with different parameter sets; (ii) different types of connections between pairs of populations; and (iii) different types of activity propagation, with or without propagation time delays between and within neural populations.

Classes and their Interactions

An overview of *NFTsim*'s calling interactions between classes, is illustrated in Fig. 3. In this diagram main and auxiliary base classes are positioned so that, in a simulation, their position corresponds to being initialized and stepped forward in time from top to bottom and from left to right within each row. In the first row, we see the high-level class `Solver` which coordinates how the other classes interact during a simulation. In the second row, the main base class `Propagator` computes each of the axonal pulse-rate fields ϕ_{ab} generated by the firing rate Q_b. In any given neural field model there are as many `Propagator` objects as there are connections. These can be any of three derived `Propagator` classes (`Wave`, `Harmonic`, `Map`) implemented to accommodate the operators defined in Eqs (13), (14), or (15), respectively. The `Wave` class uses an explicit time stepping method based on second order central difference schemes in space and time (see *Explicit Difference Method and Boundary Conditions for the 2D Wave Equation*). The `Harmonic` class implements Eq. (14), where for spatially homogeneous models the Laplacian term is zero and one finds a damped oscillator response. This class uses a standard fourth-order Runge-Kutta (RK4) explicit forward time stepping method with a fixed time step [83]. Lastly, the `Map` class, where the propagator is simply a direct mapping as in Eq. (15). Below `Propagator`, there is the auxiliary class `Tau`, which handles the activity history and retrieves the appropriate delayed activity for use in Eq. (6) when the discrete time delay τ_{ab} is nonzero. Then, to the right of `Propagator`, the `Coupling` class handles synaptic connections and their dynamics. The base `Coupling` class assumes that the synaptic strengths are constant over space and time. Thus, the output signal is a product of incoming activity and synaptic weights. Other

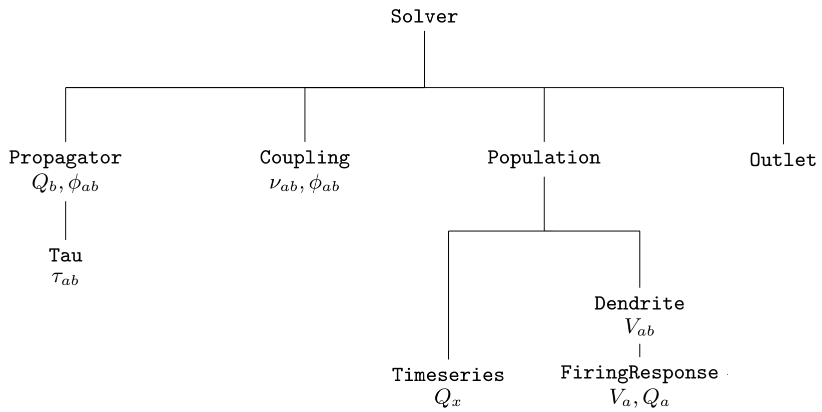

Fig 3. Simplified diagram of *NFTsim*'s call graph. The execution of a simulation is controlled by the class `Solver`. Initial conditions are given in terms of firing rates Q_b which are then propagated to other populations via `Propagator`. Synaptic connections are handled via `Coupling`. The incoming activity to postsynaptic `Population` undergoes dendritic dynamics via `Dendrite`. The sum of individual contributions V_{ab} and the resulting firing response are handled by `FiringResponse`. The class `Timeseries` is used to represent external inputs Q_x from a stimulus `Population`. Lastly, the class `Outlet` stores the variables that are written to the output file.

derived `Coupling` classes implement temporally varying synaptic strengths as in [35], or modulation by pre- or postsynaptic activity, as in [39]. To the right of `Coupling`, the `Population` class describes neural population activity and its parameters define the type.

In the third and fourth rows, below `Population`, we see that each `Population` uses two subsidiary classes: an array of `Dendrite` objects (one for every population connected via a `Coupling`); and, a `FiringResponse`. The signal from a `Coupling` object is passed to a corresponding `Dendrite` object which implements the synaptodendritic effects defined in Eq. (6). The contributions V_{ab} are then summed to yield the soma potential V_a of the population. Then, the population's `FiringResponse` object implements Eq. (9) to calculate the resulting population firing rate Q_a. Different forms of the activation function are specified within the base `FiringResponse` class. Other types of activation function that involve modulation of parameters due to presynaptic or postsynaptic activity are implemented in classes derived from the `FiringResponse` class. Such is the case of `BurstingResponse` that implements modulation of firing threshold θ_a [68]. External or stimulus populations are also objects of the `Population` class. However, their activity is represented by a predefined spatiotemporal profile of firing rate Q_x, that represents a chosen input and is contained in an object of the class `Timeseries`. In *NFTsim* the external inputs may include noisy and/or coherent components which may or may not be spatially localized (e.g., afferent to the visual thalamus in response to a visual stimulus). Currently, *NFTsim* supports a number of different external driving signals (ϕ_{ax}) to stimulate any population a of a system. These signals include: a constant value equivalent to applying DC voltage; sine waves; square pulse trains; and, white noise to simulate random perturbations. These basic functions can be combined additively to generate more complex stimulation signals.

Lastly, to the right of `Population`, the class `Outlet`, stores the variables that are written to the output file.

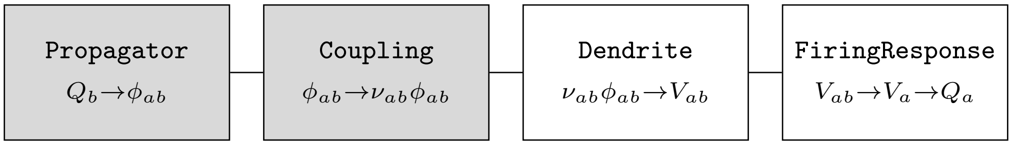

Fig 4. *NFTsim* classes associated with biophysical processes. This diagram illustrates the relationship of the classes in the library and the biophysical transformations they represent. Input variables are on the left, while output variables are on the right. Gray boxes are classes associated with interactions between populations, while white boxes are classes associated with internal mechanisms of a population.

In summary, a compact representation of the neural field equations with the label of the associated *NFTsim* classes is

$$\mathcal{D}_{ab}\phi_{ab} = Q_b, \qquad \texttt{Propagator} \tag{17}$$

$$P_{ab} = \nu_{ab}\phi_{ab}, \qquad \texttt{Coupling} \tag{18}$$

$$D_{ab}V_{ab} = \nu_{ab}\phi_{ab}, \qquad \texttt{Dendrite} \tag{19}$$

$$Q_a = S_a\left[\sum_b V_{ab}\right], \qquad \texttt{FiringResponse.} \tag{20}$$

where the auxiliary variable P_{ab} in Eq. (18) is only defined inside the `Coupling` class and assigned the presynaptic inputs weighted by the local synaptic coupling strength. Figure 4, which is analogous to the diagram presented in Fig. 1, illustrates the input and output variables of each class and the direction in which they flow within a simulation.

Input-Output

The main routine of *NFTsim* takes a plain text configuration file as input, where all the model description and simulation parameters are specified, and writes the simulation result to an output file. Both the configuration file and output file are plain text files, so launching simulations and reading the results with other programming languages is also possible. Note that all the parameters in the configuration and output files are specified directly in SI units without prefixes (e.g., s, s^{-1}, V); e.g., a value of 1 mV is written as `1e-3` (where V is implicit).

Configuration and Output Files

The following listing shows an exemplar configuration file, named `e-erps.conf`, which is included with other examples in the `configs/` directory of *NFTsim*. This file specifies a neural field model with a single cortical excitatory population that receives inputs from an external population which is the source of a stimulus to the cortex. In this example, parameters were taken from [32], with the exception of the axonal propagation parameters, which are tuned to emphasize wave propagation properties (i.e., by decreasing the damping rate γ_{ab}) for illustrative purposes. The cortical population is initially in a steady state of low firing rate around 10 s^{-1} and is driven by two pulses applied toward the center of the grid. The first pulse occurs at $t \approx 32$ ms and has a positive amplitude of $\phi_{ex_1} = 2\ \mathrm{s}^{-1}$. The onset of the second pulse is $t \approx 60$ ms and has a negative amplitude of $\phi_{ex_2} = 2\ \mathrm{s}^{-1}$.

```
 1 e-erps.conf - configuration file for a single-population neural field
       model.
 2 All parameters are in SI units
 3
 4 Time: 0.25 Deltat: 2.4414e-4
 5 Nodes: 4096
 6
 7     Connection matrix:
 8 From:  1  2
 9 To 1:  1  2
10 To 2:  0  0
11
12 Population 1: Excitatory
13 Length: 0.5
14 Q: 10
15 Firing: Function: Sigmoid Theta: 0.01292 Sigma: 0.0038 Qmax: 340
16  Dendrite 1: alpha: 83 beta: 769
17  Dendrite 2: alpha: 83 beta: 769
18
19 Population 2: Stimulation
20 Length: 0.5
21  Stimulus: Superimpose: 2
22    Stimulus: Pulse - Onset: 0.03125 Node: 2000 Amplitude:  2
23                      Width: 0.001953125 Frequency: 1 Pulses: 1
24    Stimulus: Pulse - Onset: 0.06250 Node: 2097 Amplitude: -2
25                      Width: 0.001953125 Frequency: 1 Pulses: 1
26
27 Propagator 1: Wave - Tau: 0 Range: 0.2 gamma: 30
28 Propagator 2: Map -
29
30 Coupling 1:  Map - nu: 0
31 Coupling 2:  Map - nu: 1e-4
32
33 Output: Node: 2000 Start: 0 Interval: 9.7656e-4
34 Population: 2.Q
35 Dendrite:
36 Propagator: 1.phi
37 Coupling:
```

The above file starts with a brief description of the model to be simulated. This comment is optional and can span multiple lines. In lines 4-5, global parameters for the simulation are defined: simulation duration (`Time`), time step size (`Deltat`), and the total number of nodes in the two dimensional grid (`Nodes`).

The aforementioned parameters are followed by the specification of a square connection matrix in lines 7-10, where the rows are the target populations and columns indicate the source populations. In this matrix, a positive integer indicates there is a connection between two populations and it also serves as an identifier of that connection. In the case presented above, there are only two nonzero connections, connection `1` to `Population 1` from itself and connection `2` to `Population 1` from `Population 2`. The couplings, dendrites and propagators are labeled by these consecutive positive integers. The two populations of this example are defined in lines 12-25. Each population in the model is specified separately, indicating its type (e.g., excitatory, inhibitory, or external), the physical size of its longest side (`Length`), its initial condition in terms of firing rate `Q`, and its type of dendritic and firing responses. The next step, in lines 27-28, is to define the type of propagation and coupling between each pair of connected populations. In line 27, the axonal propagation of the excitatory-excitatory connection follows a damped wave equation, with zero long-range time delay (`Tau`), characteristic spatial range of 0.2 m (`Range`) and a damping coefficient of 30 s^{-1} (`gamma`). Finally, at the end of the configuration file, from line 33 onwards in this example, we specify which timeseries are written to the output file.

There are three global output parameters: `Node` which specifies the labels of the grid nodes whose activity will be written to the output file; `Start`, sets the time (in seconds) from which the output timeseries will be written, and cannot be larger than the total simulation duration `Time`; and, `Interval` is the sampling interval between points in the timeseries. In lines 34 and 36 we see that *NFTsim* has to write the firing rate (`Q`) of `Population 2`, and the axonal field `phi` of `Propagator 1`, respectively. *NFTsim* first writes the configuration file at the top of the output file to ensure full reproducibility of the results, then it writes a line filled with the symbol `=`, and finally, it writes the requested timeseries. Below we show part of the output file `e-erps.output`.

```
1 ========================================================================
2
3                  Time                    Pop.2.Q           Propagator.1.phi
4                                             2000                       2000
5  9.76560000000000e-04    0.00000000000000e+00    1.00003146139049e+01
6  1.95312000000000e-03    0.00000000000000e+00    1.00014242188480e+01
```

Here, the first column is the time vector. The second column is the firing rate `Q` of the second population at node 2000. The third columns is the excitatory field of `Propagator 1` from `Population 1` to itself at node 2000. Line 3 provides the label of each timeseries, while line 4 shows the node index.

Numerical Methods, Considerations, and Constraints

This section focuses on considerations and constraints regarding the numerical methods implemented in *NFTsim*. In *Initial Conditions* we give a general overview and strategies to set initial conditions for neural field simulations. Furthermore, *Discretization of the Spatial Domain* and *Courant Condition* describe the way space is discretized in *NFTsim* and the maximum grid ratio for correctly solving the 2D damped wave equation, respectively. Lastly, in *Explicit Difference Method and Boundary Conditions for the 2D Wave Equation* we explain the stepping method used to solve the wave equation on a finite grid.

Initial Conditions

Neural field equations are partial delay differential equations (PDDEs), thus at the start of a simulation activity from previous times is required for initialization. *NFTsim* assumes the system is initialized at a stable fixed point and then fills a history array, which stores the past activity of the system, with the values of firing rate at equilibrium. The size of the history array depends on the time step size Δt and the longest time delay τ_{ab} in the model. For example, if the maximum time delay is 16 ms and the time step is 0.125 ms, then the length of the history array will be 128.

In a steady state, the fields in the system do not change, so *NFTsim* sets all the temporal derivatives to zero. Furthermore, *NFTsim* currently assumes that the initial activity is uniform spatially. If the stationary state used as initial conditions is unstable, or if it is close to a stable state, one can expect to see transient activity until the system settles into the closest stable attractor (either a fixed point or another manifold).

We now present an example of how to find the steady states of a cortical model with two cortical populations with one external input. The equation for the cortical steady state $\phi_e^{(0)}$ is

$$S_e^{-1}\left[\phi_{ee}^{(0)}\right] = (\nu_{ee} + \nu_{ei})\phi_{ee}^{(0)} + \nu_{es}\phi_{es}^{(0)}, \tag{21}$$

where the system is also assumed to be driven by a constant, spatially uniform input $\phi_{es}^{(0)}$ from subcortical structures. The structure of the solution can be interpreted as follows: the left-hand side of Eq. (21) is a monotonic increasing function of $\phi_{ee}^{(0)}$, while the right-hand side is linear in $\phi_{ee}^{(0)}$ (although it could more generally be nonlinear). Hence, either one or three solutions exist [31]. When three solutions are found, the middle one represents an unstable equilibrium, the lower corresponds to normal activity, and the upper to a high firing rate seizure-like state [31]. When five solutions are found, three of which are stable, the additional middle stable steady state has been found to be consistent with a waking state of increased cortical and thalamic activity [76].

The form of the steady-state equation strongly depends on the number of populations and the interconnections within the model. Also, due to the transcendental nature of the steady-state equation, fixed point solutions have to be computed numerically. To find the roots, the steady-state equation is usually rearranged so as to have all the terms to the right hand side, and then evaluated for a range of $S_e^{-1}\left[\phi_{ee}^{(0)}\right] \equiv V_e^{(0)}$ values. Solutions are identified by a change in sign between consecutive test values of $V_e^{(0)}$. A standard root finding algorithm (e.g., Newton-Raphson) can then be used to refine the roots. An initial scan over $V_e^{(0)}$ is required because the root finding algorithm will most likely converge to the root nearest the initial guess for $V_e^{(0)}$.

For these reasons, *NFTsim* does not currently provide a general method to find the stable steady-state solutions of the system. Nevertheless, it is possible to use *NFTsim* to find them. One strategy consists of running auxiliary simulations to give the system enough time to reach a stable state. The end state of this auxiliary simulation can then be used to provide the initial conditions for other simulations. The second strategy consists of combining *NFTsim* with Monte Carlo methods to run numerous simulations with randomly sampled initial conditions in order to find the stable states.

The first approach mentioned above is best suited for scenarios in which one already has an initial estimate of the initial stable state of the system; and for nonuniform situations [77, 78], in which case the auxiliary simulations are run for the uniform case and the nonuniformities in the parameters are introduced in the main simulations. The second approach is more general and does not require any a priori knowledge of the initial conditions. This approach is best suited for neural field models with several populations and for which finding the steady states of the systems analytically is not possible or is too cumbersome. If multiple steady states are found [76], the user must decide which one is to be used for the main simulations. Usually the linearly stable fixed point that represents the lowest firing rates is selected as the initial condition on the basis that represents a normal brain state [2, 28].

Discretization of the Spatial Domain

Each population is modeled as a 2D rectangular sheet. In *NFTsim*, the physical spatial domain of each population, whatever its extent, is divided into a finite number N of uniform grid cells (or nodes), which remain invariant throughout the simulations for all times.

In a configuration file, the parameter `Length` corresponds to the physical length of the x-axis. By default, the domains are assumed to be square with $L_x = L_y$. In this case, the value of the parameter `Nodes` must be a perfect square so that the spatial resolutions

$$\Delta x = \frac{L_x}{\sqrt{N}}, \tag{22}$$

and

$$\Delta y = \frac{L_y}{\sqrt{N}} \tag{23}$$

are the same.

To define a rectangular domain, in addition to the parameter `Nodes` (N), in the configuration file one can specify the number of nodes along the x-axis via the parameter `Longside nodes` (N_x). In this case, the number of nodes along the x and y axes are different, but the spacing remains the same for both axes (i.e., $\Delta x = \Delta y$)

$$\Delta x = \frac{L_x}{N_x}. \tag{24}$$

The number of nodes and physical length of the y-axis can be obtained as $N_y = N/N_x$ and $L_y = N_y \Delta y$, respectively. Table 2 summarizes the symbols and configuration length and size parameters used in this section and in the remainder of the text.

As an example, we show part of a configuration file for a neural field model with two populations. The physical length of the first population L_x^1 is larger than the length of second population L_x^2.

```
Time: 0.15 Deltat: 0.0001
Nodes: 12 Longside nodes: 4

    Connection matrix:
From:  1  2  3
To 1:  1  2  0
To 2:  0  0  3
To 3:  0  0  0

Population 1: Big population
Length: 0.8
Q: 10
Firing: Function: Sigmoid Theta: 0.01292 Sigma: 0.0038 Qmax: 340
 Dendrite 1: alpha: 83 beta: 769
 Dendrite 2: alpha: 83 beta: 769


Population 2: Small population
Length: 0.08
Q: 10
Firing: Function: Sigmoid Theta: 0.01292 Sigma: 0.0038 Qmax: 340
 Dendrite 3: alpha: 83 beta: 769
```

In the above file the two internal populations are modeled as rectangular grids with a total of 12 nodes or grid cells, and with the number of nodes of the longest side specified by `Longside nodes`. The resulting 2D grid has a size of 4× 3 nodes as shown in the schematic of Fig. 5. For illustrative purposes, the parameter values used in this configuration file have been exaggerated so the link between the input parameters and the discretization of the space shown in the schematic is clear. However, this configuration file will not produce accurate results because the spatial resolution is too coarse.

Figure 5 illustrates that *NFTsim* populations are linked via a primary topographic one-to-one map, which implies that all the populations must have the same number of grid points N, even if they have different physical spatial dimensions. We assign the same map coordinate $\mathbf{r}_n$ to homologous grid cells in different populations. In this example, $\mathbf{r}_1$ is assumed to be the actual physical position in `Population 1`, but in `Population 2`, $\mathbf{r}_1$ denotes a rescaled physical dimension. Also, any physical position $\mathbf{r}_n$, for $n = 1, ..., N$ is assumed to be at the center of a grid cell, which is also labeled with integers $n = 1, ..., N$. For instance, in `Population 1`, $\mathbf{r}_1$ corresponds to position $(\Delta x_1/2, \Delta y_1/2) = (0.1,\ 0.1)$ m; and, in `Population 2`, $\mathbf{r}_1$ corresponds to position $(\Delta x_2/2, \Delta y_2/2) = (0.01,\ 0.01)$ m. Lastly, the borders of the grid are depicted with dashed lines to denote periodic boundary conditions (PBCs), which represent structures with planar geometry and toroidal topology.

Table 2. Symbols, configuration parameters and units.

Symbol	Parameter name in configuration file	Units	Parameter exposure
N	`Nodes`	-	required
N_x	`Longside nodes`	-	optional ($\sqrt(N)$)
L_x	`Length`	m	required
Δx	-	m	none (L_x/N_x)
Δy	-	m	none (Δx)
N_y	-	-	none (N/N_x)
L_y	-	m	none ($N_y \Delta y$)

First column: symbols used in this work to identify the parameters specified in a configuration file. Second column: parameter names used in configuration files to determine the physical size and spatial resolution of the 2D sheets for each population. The symbol − means the parameter is not specified directly in a configuration file. Third column: SI units of each parameter. Here, the symbol − means the parameter is dimensionless. Fourth column: shows whether the exposure [81] of each parameter in the configuration file is (i) required, (ii) optional (with its default value); or, (iii) not required (none). In the latter case, the parameter is derived internally in the code and we provide the equation used to calculate its value).

Courant Condition

The interval Δx is used to evaluate whether the current parameters satisfy the Courant condition, a necessary condition for obtaining stable solutions when solving hyperbolic partial differential equations on a regular discrete grid. For the wave equation in 1D the dimensionless number

$$p_{ab} = \frac{v_{ab}\Delta t}{\Delta x} \leq 1, \tag{25}$$

is called the Courant number [82]; Δt is the integration time step size and $v_{ab} = \gamma_{ab} r_{ab}$ is the magnitude of the wave velocity. In the continuum wave equation, activity propagates at maximum speed v_{ab} and the method is stable when $\Delta x/\Delta t \geq v_{ab}$. Unstable schemes arise when $\Delta x/\Delta t < v_{ab}$ because waves propagate more than one grid spacing in a period Δt. However, for the 2D case one finds the stability criterion to be [83]

$$\Delta t \leq \frac{1}{v_{ab}} \left[\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right]^{-\frac{1}{2}}, \tag{26}$$

so, because $\Delta x = \Delta y$

$$\begin{aligned} \Delta t &\leq \frac{1}{v_{ab}} \left[\frac{2}{(\Delta x)^2} \right]^{-\frac{1}{2}}, && (27) \\ &\leq \frac{\Delta x}{v_{ab}} \frac{1}{\sqrt{2}}. && (28) \\ & && (29) \end{aligned}$$

Hence, considering all wave-type propagators, the maximum value of the Courant number $p_{\max}$ must satisfy

$$p_{\max} = \max(v_{ab}) \frac{\Delta t}{\Delta x_1} \leq \frac{1}{\sqrt{2}}. \tag{30}$$

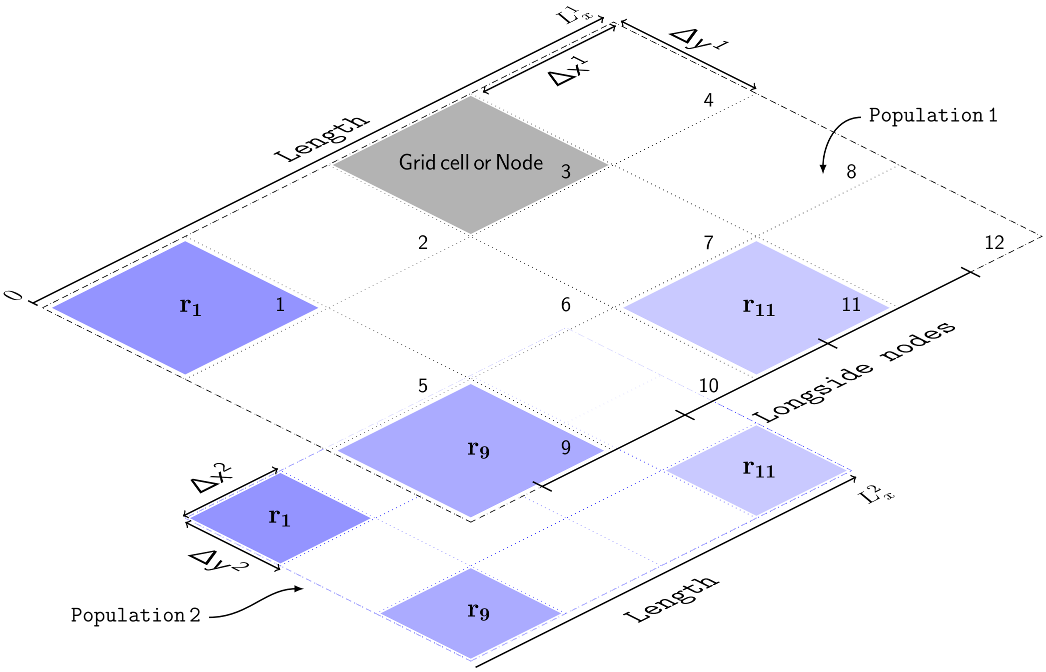

Fig 5. Schematic of the discretized spatial domain. The model has two populations: `Population 1` and `Population 2`. Geometrically, each population is represented by a grid of 12 nodes, which are labeled with integers. The grid is rectangular with dimensions 4 × 3 nodes. The number of nodes of the longest side is specified by `Longside nodes`. The physical size, L_x, of each population is different. Thus, each node in `Population 1` has a linear size of Δx^1, and of Δx^2 in `Population 2`. Each spatial point (e.g., $\mathbf{r}_1$, $\mathbf{r}_9$, $\mathbf{r}_{11}$) is at the center of a grid cell. The subscript denotes the node index on this grid. Also, $\mathbf{r}_n$ denotes the actual position in the largest population; in the smallest population $\mathbf{r}_n$ denotes a rescaled physical dimension.

This condition is checked internally by *NFTsim* and if it is not satisfied, an error message is returned. Note that, in practice, one usually imposes a stricter condition to ensure the system has a margin of stability; e.g., in [2], the grid ratio was chosen so that $p_{\max} = 0.1$.

Explicit Difference Method and Boundary Conditions for the 2D Wave Equation

NFTsim uses an explicit central difference method [84] to solve Eq. (13), which represents axonal propagation of activity through the cortex or other structures with a significant spatial extent. Here, we present the explicit time stepping formula currently implemented to compute the next value of ϕ_{ab} from past values of ϕ_{ab} and Q_b. The full derivation is in *Appendix S1.*

Equation (13) is the inhomogeneous damped wave equation, which can be simplified by making the substitutions

$$u = \phi_{ab} e^{\gamma_{ab} t}, \tag{31}$$

$$w = Q_b e^{\gamma_{ab} t}. \tag{32}$$

We then obtain the undamped wave equation

$$\left[\frac{1}{\gamma_{ab}^2}\frac{\partial^2}{\partial t^2} - r_{ab}^2\nabla^2\right] u(\mathbf{r},t) = w(\mathbf{r},t). \tag{33}$$

Note that this simplification only works for small values of Δt because the exponential factors introduced in Eqs (31) and (32) diverge as $\Delta t \to \infty$. Then, the final time-stepping formula using an explicit central difference method for the 2D wave equation is

$$\begin{aligned}\phi_{m,l}^{n+1} = e^{-\gamma_{ab}\Delta t}\Big\{&(2-4p^2)\phi_{m,l}^{n} + p^2(\phi_{m,l+1}^{n} + \phi_{m,l-1}^{n} + \phi_{m+1,l}^{n} + \phi_{m-1,l}^{n}) - \phi_{m,l}^{n-1}e^{-\gamma_{ab}\Delta t} \\ &+ \frac{\Delta t^2\gamma_{ab}^2}{12}\Big[(10-4p^2)Q_{m,l}^{n} + (Q_{m,l}^{n+1}e^{\gamma_{ab}\Delta t} + Q_{m,l}^{n-1}e^{-\gamma_{ab}\Delta t}) \\ &+ p^2\left(Q_{m,l+1}^{n} + Q_{m,l-1}^{n} + Q_{m+1,l}^{n} + Q_{m-1,l}^{n}\right)\Big]\Big\}\end{aligned} \tag{34}$$

where the superscript n indexes time step; the first and second subscripts index space along the orthogonal x and y directions, respectively, except for the subscripts on γ_{ab};and p is the Courant number.

Note that in Eq. (34), only five spatial points are required: the central point m, l; its horizontal neighbors $m+1, l$, $m-1, l$; and, its vertical neighbors $m, l+1$ and $m, l-1$. This pattern is often referred to as a five-point stencil. There are alternative finite difference methods that use higher-order terms to approximate the derivatives and would require larger stencils (e.g., more neighboring points) [86]. It is usually better to increase the spatial resolution rather than the stencil complexity to obtain higher accuracy.

The finite difference scheme presented above is second-order accurate in space and time. This means that the rate at which the error between the discretized approximation and the exact continuum solution decreases to zero is $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2) + \mathcal{O}(\Delta t^2)$. For instance, halving Δx, Δy, or Δt, subject to Eq. (30) leads to a decrease of the error by a factor of four.

When solving partial differential equations on a finite spatial domain, one must specify boundary conditions for the simulations. *NFTsim* uses periodic boundary conditions (PBCs). This type of condition avoids boundary effects stemming from the finite size of a grid and avoids the perturbing influence of an artificial boundary like a reflective wall. In PBCs, opposite boundaries are treated as if they were physically connected, that is, the top of the grid is wrapped on to the bottom and the left of the grid on to the right.

The class `Stencil` has two main functions: (i) retrieving the five-point stencil pattern for every node in the grid; and, (ii) correctly copying the activity close to the boundaries of the domain at every time step to implement periodicity. To achieve this, `Stencil` operates on a grid of size $(N_x + 2) \times (N_y + 2)$. The additional ghost cells are used to store copies of the top and bottom rows and left and right columns of the grid.

Figure 6 illustrates a 4×4 grid with the additional ghost cells shaded in light blue and five-point stencil pattern consisting of a central grid point `c` and its 4 neighbors labeled as `n`, `s`, `e`, `w` (i.e., north, south, east, west). The number in each grid cell represents its linear index – because the class `Stencil` accesses the elements of the two dimensional grid using a single subscript instead of two. The grid cells with prime, double-prime, and triple-prime indices are copies of the original cells with the same indices. These copies are used to implement PBCs along the vertical, horizontal, and diagonal directions, respectively. For instance, the cell $1'$ is the vertical copy of cell 1; cell $1''$ is the horizontal copy, and cell $1'''$ is the diagonal copy. The diagonal copies are not used by the 5-point stencil, but would be used by a 9-point stencil [86].

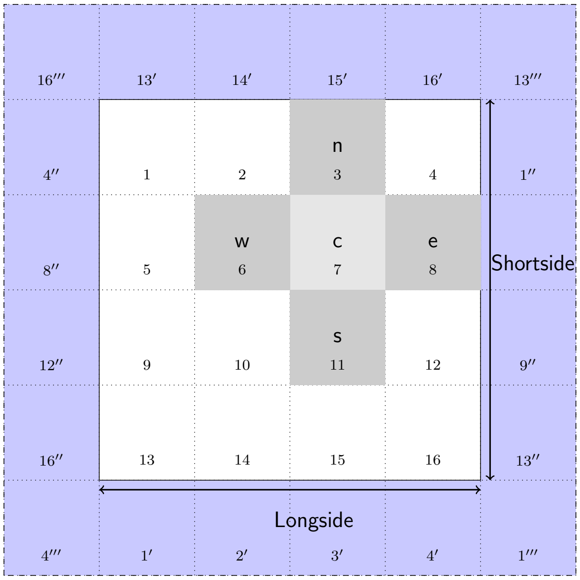

Fig 6. Schematic of the grid used by the class `Stencil`. This class retrieves the four nearest neighbors (labeled `n`, `s`, `e`, `w`) of a central point `c`. These five points define the pattern known as a five-point stencil. The cells in light blue are the ghost cells required to implement periodic boundary conditions. The prime, double-prime and triple prime indices represent copies of the corresponding indices in the vertical, horizontal and diagonal directions, respectively.

Analysis and Visualization

NFTsim includes a Matlab module which provides ancillary tools to assist with running, analyzing and visualizing models. This package folder is called `+nf`. The available functions and a description of their functionality are summarized in Table 3.

The code snippet below uses some basic `nf` functions as an example of how users can interact with *NFTsim* directly from Matlab. The model is the same as the one specified in the configuration file `e-erps.conf` presented earlier, except that the timeseries of all the nodes in the grid are written to the output file. The simulation is executed via `nf.run()`. Once the output file is available `nf.read()` loads the simulated data into a Matlab structure.

Spatial patterns of activity and propagation of waves of activity across space can be visualized using the function `nf.movie()`

```
nf.movie(nf_struct, ‘Propagator.1.phi’, 1)
```

Representative frames from the movie of waves propagating from stimulation sites are shown in Fig. 7(a) to Fig. 7(f). In each panel the mean spatial value of $\phi_{ee}(x, y, t)$ at time t has been subtracted, so red and blue reflect positive and negative deviations, respectively, from the mean. The file used in this example is included in *NFTsim* and is also available in *Appendix S2*.

Furthermore, extracting and plotting the timeseries of a few nodes enable users to directly inspect the type of activity (e.g., healthy neural activity, evoked responses, or seizures). In this example, `nf.extract()` is used internally by `nf.plot_timeseries()` to select the timeseries `Propagator.1.phi`.

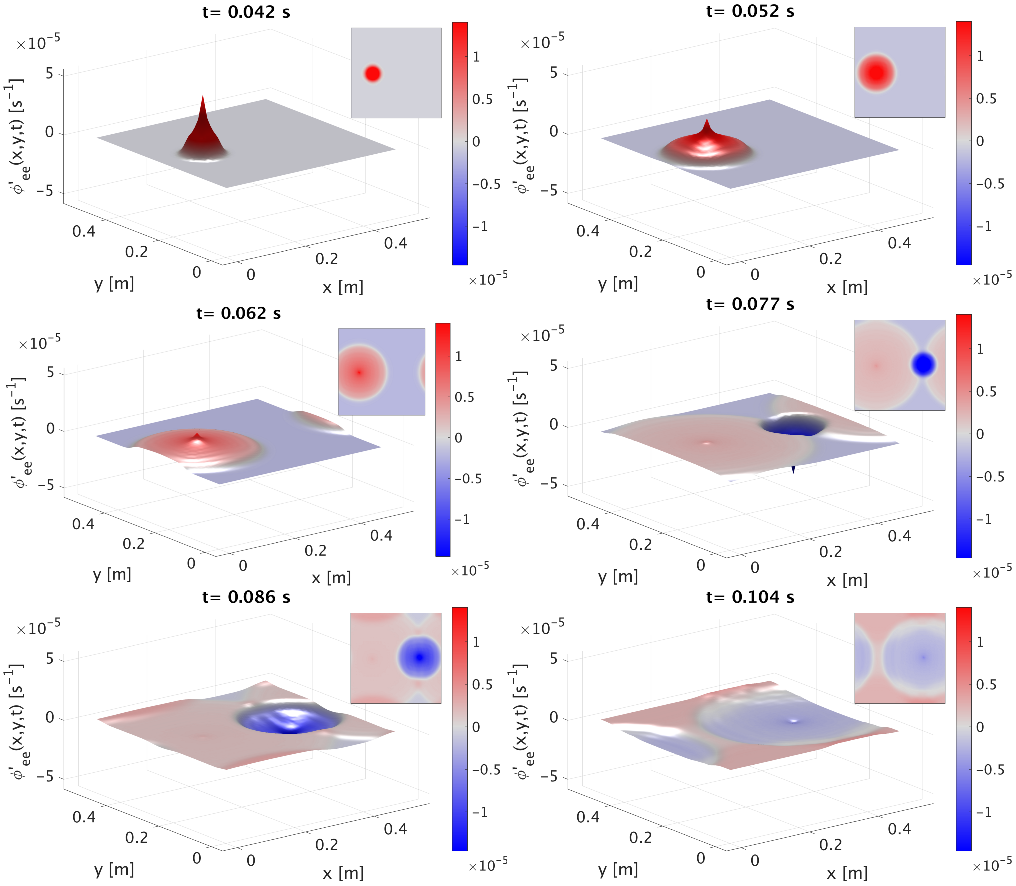

Fig 7. Neural activity of the model described in `e-erps.conf`. The cortical population is driven by two square pulses. The first pulse is positive, while the second pulse is negative. For illustrative purposes, in each panel the mean spatial value of $\phi_{ee}(x,y,t)$ has been subtracted, so the color reflects deviations from the mean at that specific time. Each panel shows a surface plot of $\phi'_{ee}(x,y,z) = \phi_{ee}(x,y,t) - \langle\phi_{ee}(x,y,t)\rangle$ s^{-1} propagating radially outwards from the stimulation sites, and an inset with a planar view of the same quantity, at different times: **(a)** 42 ms; **(b)** 52 ms; **(c)** 62 ms; **(d)** 77 ms; **(e)** 86 ms; **(f)** 104 ms.

```
nf_struct = nf.run(‘configs/e-erps-all-nodes.conf’)
these_nodes = {[1992:2008],[2089:2105]};
these_traces = {‘Propagator.1.phi’, ‘Propagator.1.phi’};
nf.plot_timeseries(nf_struct, these_traces, these_nodes, true)
```

Figure 8 shows the resulting plots generated with the code shown above. Each set of timeseries is centered around one of the stimulation sites. In Fig.8(a) the red curve is the axonal field at the site that received positive stimulation; and, in Fig.8(b), the blue line is the axonal field at the site that received negative inputs. The timeseries in gray above and below the colored curves are the axonal fields from neighboring sites along the x-direction. In these plots, the distance between the stimulation sites and neighboring sites increases vertically from the center to the top and bottom edges. The vertical dashed lines are not automatically produced by `nf.plot_timseries`, but have been added to mark the onset time of the positive (red dashed) and negative (blue dashed) inputs, respectively.

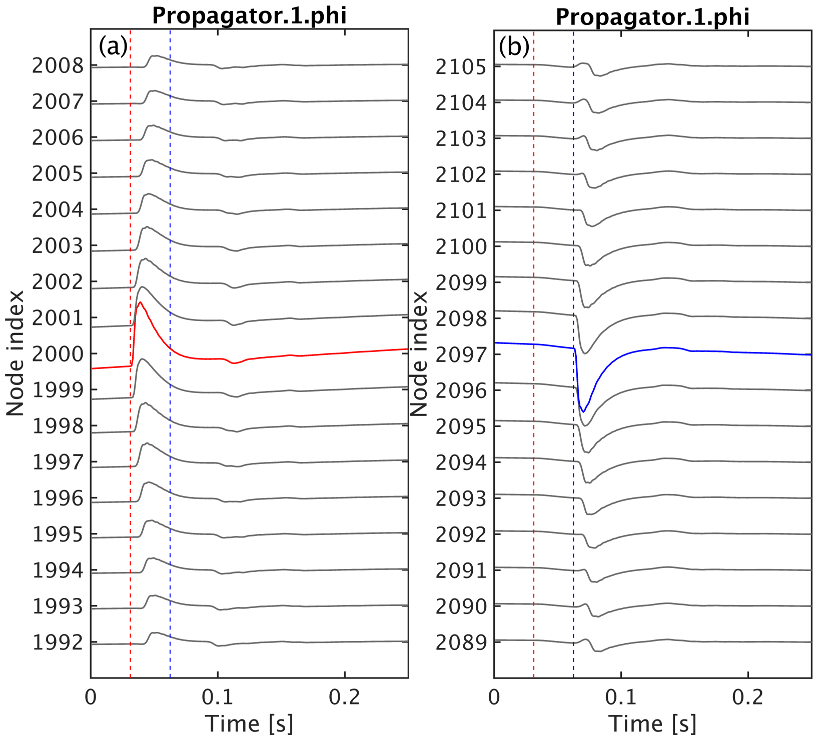

Fig 8. Timeseries of neural activity of the model described in `e-erps.conf`. The cortical population is driven by two square pulses applied at the center of the grid as shown in Fig. 7. Here, we illustrate the timeseries of ϕ_{ee} from a few nodes close to the vicinity of (black lines) and at the stimulation sites. The vertical dashed lines mark the onset time of the positive (red dashed) and negative (blue dashed) stimulation inputs, respectively. **(a)** the axonal field at the site receiving the positive stimulus is highlighted in red while the time evolution of the same axonal field at neighbouring locations is shown as black lines. **(b)** the axonal field at the site receiving the negative input is highlighted in blue.

Another important step is the calculation of the temporal power spectrum for a range of frequencies (in Hz), which is often compared to the power spectrum of experimental data. The power spectrum may also include multiple spatial modes for a range wave numbers (in m^{-1}) and incorporate volume conduction or hemodynamic effects [97, 98] on measurement. A comparison between the linear analytic power spectrum and the numerical nonlinear power spectrum calculated with `nf.spatial_spectrum()` is given as an example in *Standard Tests and Reproducibility.*

Table 3. Auxiliary functions available in the module `+nf`.

Function	Description
`extract`	extracts time-series from an output structure
`get_frequencies`	returns the spatial frequencies
`grid`	reshapes output into a 3D array of shape (L_{out}, N_x, N_y)
`movie`	generates a movie from a 3D array produced by `nf.grid`
`plot_timeseries`	plots vertically spaced timeseries of specific traces and nodes
`read`	reads an output file and returns a structure
`report`	prints information about a structure
`run`	runs a simulation from a configuration file
`spatial_spectrum`	computes spatiotemporal spectrum
`spectrum`	computes temporal spectrum
`wavelet_spectrogram`	computes wavelet spectrogram using a Morlet wavelet

First column: names of the available Matlab functions. Second column: brief descriptions of what each function does. The variable L_out is the number of time points in the output file.

Results and Applications

In this section we first present four exemplar systems that can be simulated using *NFTsim* and that have been previously studied in detail. Then, section *Observables and Diagnostics* briefly discusses the main observables that can be currently computed in *NFTsim* and how these have been used to predict a range of brain phenomena and compare to experimental results. In section *Standard Tests and Reproducibility*, we discuss how *NFTsim* could be used as a validation tool for neural field models and neural field simulators. Lastly, section *Benchmarks* presents performance metrics and practical information for users regarding average run times, memory usage, and storage required for typical simulations based on a neural field model of the corticothalamic system [28].

Exemplar Systems

The versatility of neural field theory and its concomitant implementation in *NFTsim* allow for the investigation of an unlimited variety of specific models and parameter sets. In this section we present a few illustrative cases, which have been thoroughly described elsewhere, along with some of their applications [1, 2, 6, 26–30, 32–38, 68]. Their respective configuration files are included in *NFTsim.*

The most general corticothalamic model considered here includes populations with long-, mid-, and short-range connections in the cortex, the specific and reticular nuclei in the thalamus, and external inputs. We indicate how components of this model can be successively deleted to obtain a family of models suited to simpler applications in corticothalamic and cortical systems. In what follows we label specific models according to the internal populations they include. The first system, called EMIRS, includes five different populations of neurons: cortical excitatory pyramidal (e), excitatory mid-range

(m) and inhibitory (i) populations; internal thalamic reticular nucleus (r), relay specific nucleus (s), whereas the simplest case is of a system with a single excitatory population (e). There is also always at least one external population that provides inputs (often labeled either x or n). *NFTsim* provides a number of external input types such as sinusoids (in space and time), pulses, and white noise. For instance, these inputs could be from excitatory neurons of the auditory pathway, which transmit signals from the cochlea to the thalamus [70]; or, they could be artificial external stimulation like Transcranial Magnetic Stimulation [37].

Figure 9 shows schematics of the illustrative neural field models described here. The EMIRS corticothalamic model displayed in Fig. 9(a) includes three cortical populations (e, m, and i) and two thalamic populations (r and s), with intracortical, intrathalamic, and corticothalamic connections.

The EIRS corticothalamic model is obtained by deleting the population m from Fig. 9(a) to obtain Fig. 9(b). Physically, this deletion corresponds to describing the effects of the mid- range population, whose axonal range is of the order of a few millimeters, as part of the short-range i population [1]. In this case, the excitatory effect partly cancels inhibition to give a weaker, net effect from this compound population, which includes the effects of both short-range excitatory and inhibitory interneurons. This model has been successfully applied to investigate a wide range of phenomena [30, 42, 69] (see *Introduction*). The model has five distinct populations of neurons: four internal and one external.

In the purely cortical EI model of Fig. 9(c), thalamic dynamics are deleted and $\phi_{es} = \phi_{is}$ is assumed to replace ϕ_{sn} as the external input to an isolated cortex. The basic EI model includes external inputs to two cortical populations (e and i), and both intracortical and corticocortical feedback are represented. This model is a starting point for understanding more elaborate neural fields models of the cortex (e.g., modeling distinct layers within the gray matter [34, 70]). Delays in the propagation of signals within neurons are due to synaptodendritic, soma, and axonal dynamics. However, in this model there are no long-range delays like those from the thalamus to the cortex. An extensive description and analysis of this model are given elsewhere [2, 31, 71], including emergence of gamma rhythm [69] and integration of cholinergic modulation [72]. Finally, the excitatory-only E model in Fig. 9(d) omits cortical inhibitory effects. This neural field model is the simplest system we consider that can be simulated in *NFTsim* and has been used as the central example throughout this work.

Observables and Diagnostics

Brain phenomena including the alpha rhythm [30, 33], age-related changes to the physiology of the brain [26], evoked response potentials [27, 34, 70], and seizure dynamics [1, 5, 35, 68], can be measured noninvasively via EEG. In these studies, the axonal fields of excitatory cortical population ϕ_{ee} have been used to approximate EEG signals measured from the surface of the scalp [49, 87] and constitute one of the main biophysical observables comparable to experimental EEG data. Furthermore, another tool traditionally used to detect various waking and sleep stages [6, 28] is the EEG power spectrum [49]. In calculating scalp EEG spectra (rather than intracranial ones), one must consider filtering due to volume conduction by the cerebrospinal fluid, skull, and scalp [49, 87]. The calculation of the power spectrum including volume conduction filtering is implemented in the module `+nf`.

It is extremely important to note that EEG is only one type of output. The neural activity of different cortical and subcortical populations can be used to predict other relevant electrical signals such as local field potentials (LFPs), eECoG and stereoencephalography (SEEG); magnetic signals such as MEG; metabolic-related signals like fMRI [88] or indirect fluorescence signals like those recorded via voltage

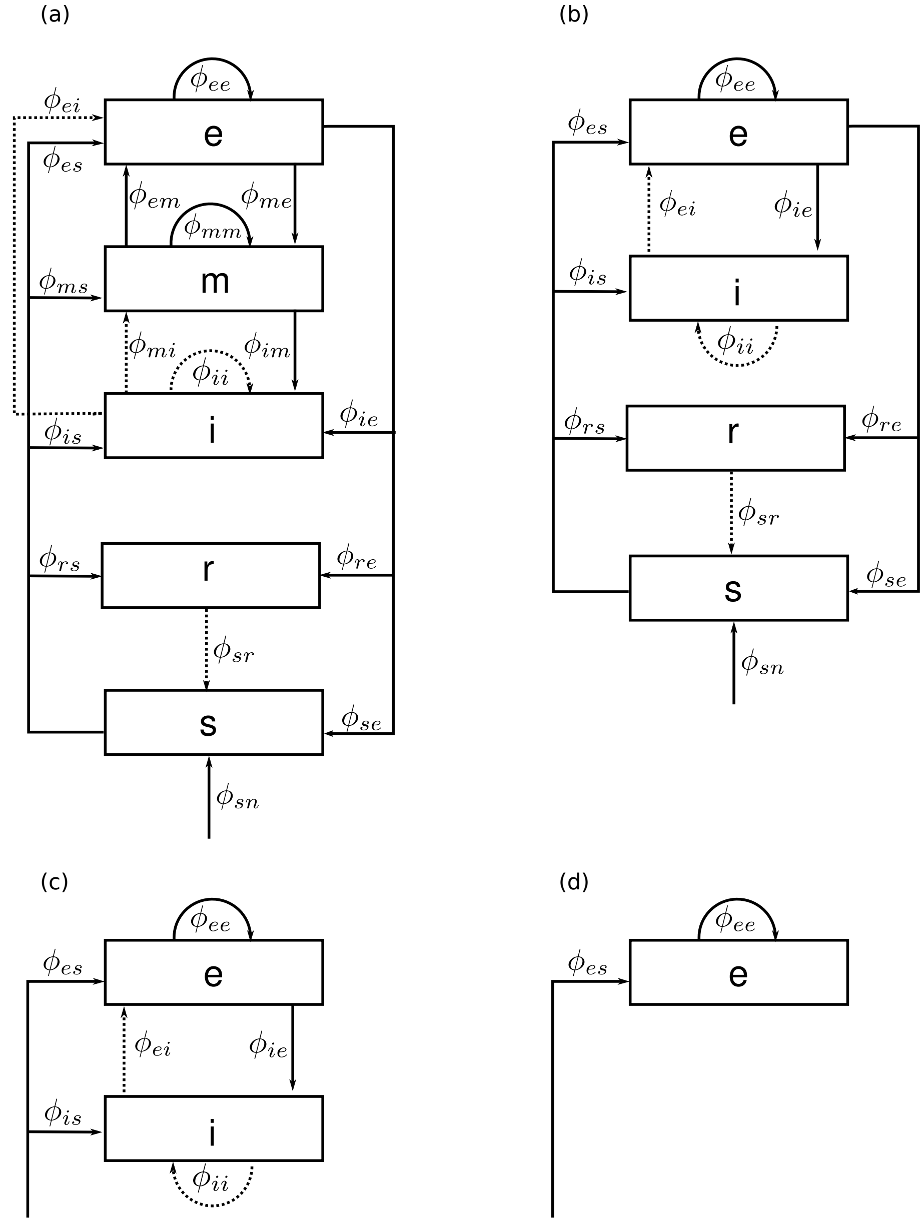

Fig 9. Schematic of four representative neural field models. The quantities ϕ_{ab} are the fields propagating to population a from population b. Dashed lines represent inhibitory connections. **(a)** Corticothalamic model including excitatory (e), mid-range (m), inhibitory (i), reticular (r), specific relay (s) and external non-specific (n) populations. **(b)** Corticothalamic model including excitatory (e), inhibitory (i), reticular (r), specific relay (s), and external (n) populations. **(c)** Cortical model comprising excitatory (e) and inhibitory (i) cortical populations plus an external input field from a subcortical population (s). **(d)** Purely excitatory (e) cortical model with input from a subcortical population (s).

sensitive dyes imaging (VSDI) [89]. Note that conversion of *NFTsim* outputs to the desired neuroimaging modality signals requires an additional modeling step, including a description of the causal relationship and physiological couplings between the sources (i.e., the spatiotemporal fields of neural activity stemming from multiple populations) and the effective biophysical quantity measured in experiments [90, 92, 93, 96–98].

Standard Tests and Reproducibility

Standard tests are a set of benchmarks used evaluate and compare disparate numerical implementations of similar neurobiological models [94]. There are very few such tests in computational neuroscience [95] and the ones currently available are only for single-cell models. To the best of the authors' knowledge, there are no published standard tests for mesoscopic models such as neural fields. Thus, there is a huge void regarding quality assessment of scientific software for continuum models of brain dynamics.

NFTsim provides a reference framework for standard tests for implementations of neural field models because its methods have been verified with analytic results; and, the linear analytic closed form solutions upon which the code is based have been extensively validated with experiments, as discussed in the Introduction. For example, in Fig. 10 we reproduce the results presented in Fig. 2 from [6]. This plot shows a comparison between the linear analytic power spectrum (dashed line) and the spectrum computed from *NFTsim* simulations. Both spectra agree to within agree to *NFTsim*'s default `eirs-corticothalamic.conf` is used with the parameters from [6], which we do not repeat here because the original configuration files are also included as part of the library package. The power spectrum is calculated using the `nf.spatial_spectrum()` function.

Furthermore, the *NFTsim*'s methods and implementation have also been directly validated by experimental data for nonlinear dynamics, notably neural activity corresponding to seizures [35] and sleep spindles [6].

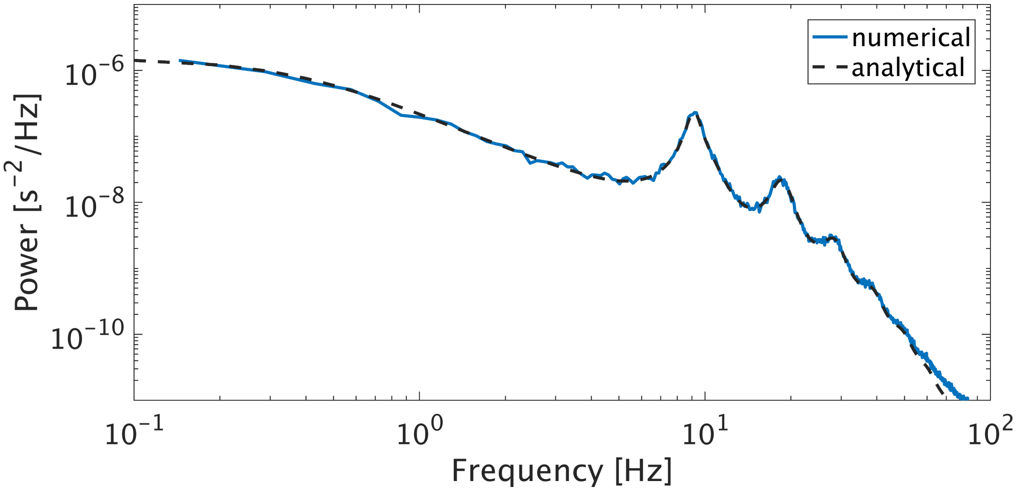

Fig 10. Comparison of analytic and numeric EEG power spectra in the corticothalamic system. The dynamics of the EIRS model were simulated using the wake parameters from [6] for their Figure 2. The linear analytic spectrum (black dashed solid) is compared against the spectra computed from simulations (solid line).

Benchmarks

NFTsim provides a tool for semi-automated benchmarking. Timing and configuration information for simulation runs are stored in a comma-separated value (csv) file that can be processed at a later stage.

Invoking the script

```
nf_benchmarks
```

without any arguments will run all the configuration files in the `benchmarks/` directory once. We provide ten default configuration files that run in a total of under 700 s on a desktop computer. Example results for specific hardware are given below. These files are based on the corticothalamic model and are representative of typical simulation scenarios. With `nf_benchmarks` users can also:

(i) benchmark a specific configuration file

```
nf_benchmarks <config_filename>
```

(ii) benchmark a specific configuration file multiple times (e.g., 8 times in this example)

```
nf_benchmarks --num-trials 8 <config_filename>
```

(iii) benchmark a specific configuration file with output written to memory instead of disk (this only works under Linux)

```
nf_benchmarks --to-mem <config_filename>
```

(iv) benchmark a specific configuration file using a non-default compiler

```
nf_benchmarks --clang <config_filename>
```

In *NFTsim* propagating fields are followed via partial differential equations, so the main contributions to the runtime T are (i) the number of grid cells N; (ii) advancing a maximum of P^2 fields, between P populations, on the N cells; (iii) the length of the simulation in integration steps $L = T_{\rm sim}/\Delta t$; and, (iv) the size of the output O written to file. So,

$$T \propto k_{\rm sim} P^2 N L + k_{\rm out} O \tag{35}$$

where the coefficients $k_{\rm sim}$ and $k_{\rm out}$ depend on the hardware architecture. The output size O depends on the product of the total number of variables (W), the number of grid cells ($N_{\rm out}$) and the total output time points [$L_{\rm out} = (T_{\rm sim} - T_{\rm start})/\Delta t_{\rm out}$] requested in the configuration file.

For large O, the runtime is dominated by writing operations. This overhead is expected for two reasons: (i) a simulated data sample is written to disk every $\Delta t_{\rm out}$, which takes additional time; and, (ii) writing the output to a text file requires conversion of numbers to text. Despite the runtime overhead this last point entails, text files are a convenient format to store the output because they are easier to debug than binary files.

The required memory, M, used by a *NFTsim* process is dominated by the number of grid points N and the history arrays of P internal populations with delay depth $D = \max(\tau_{ab})/\Delta t$, which is the number of integration steps for a signal to propagate to the target population from the source population. So,

$$M \propto NPD. \tag{36}$$

Table 4 summarizes the simulation parameters that determine runtime and memory usage of a *NFTsim* process, including those which are not directly specified in a configuration file.

To assess *NFTsim*'s performance, we select the corticothalamic model presented in earlier sections, with the parameters taken from previously published work [6] and thus considered a typical simulation use case.

The simulation length and integration time step are held constant at 16 s and $\Delta t = 2^{-14}\,\text{s} \approx 10^{-4}$ s, respectively. So, the only varying parameter that affects the

Table 4. *NFTsim* simulation and output size parameters, and runtime and memory usage symbols.

Symbol	Description	Parameter in configuration file	Units
T_{sim}	Simulation length	`Time`	s
Δt	Integration time step	`Deltat`	s
L	Number of time steps	-	-
P	Number of populations	-	-
D	Delay depth	-	-
O	Output size	-	-
N_{out}	Number of output nodes	`Node`	-
Δt_{out}	Output sampling period	`Interval`	s
T_{start}	Output start time	`Start`	s
W	Number of output variables	-	-
L_{out}	Number of output time points	-	-
M	Memory used	-	byte
T	Runtime	-	s

First column: symbols used in this work to identify either the parameters specified in a configuration file, or variables associated with runtime and memory usage. Second column: description of the variable or parameter. Third column: parameter names used in configuration files. The symbol – means the parameter is not specified directly in a configuration file. Fourth column: SI units. Here, the symbol – means the quantity is dimensionless.

runtime and storage is `Nodes` (N). The choice of this integration time step size is such that is sufficiently small to resolve high frequency oscillations and to satisfy the Courant condition for numerical stability for a range of discretization values between $3\,\text{mm} < \Delta x < 50\,\text{mm}$. The Courant number ranges between $0.014 < p < 0.15$ for a fixed velocity $v_{ab} = 10$ m s^{-1}.

Two groups of simulations were ran. The first group, G_{no}, runs the simulation and only writes a copy of the configuration file to the output file. The subscript no means `no output`. In this case the runtime represents the effective time spent executing a simulation without the time overhead due to writing operations. From Eq. (35), the group G_{no} has $k_{\text{out}} \approx 0$. The second group of simulations G_{wo} consists of identical simulations to those of G_{no}, except that all the model variables (firing rate, voltages, fields, coupling strengths), for all the nodes, sampled at 512 Hz, are written to a file in the hard disk.

Approximate runtimes and memory usage are measured using tools available on Linux systems. The computer used for the benchmarks has Red Hat Enterprise Linux (RHEL) 6.9 as operating system, GNU Compiler collection (gcc) 4.9.2 as the default compiler, a 3.50 GHz Intel i5-4690 processor and 8GB of RAM.

Table 5 presents the benRHchmark results for different grid sizes and shows that the runtimes scale linearly as a function of the number of nodes with $k_{\text{sim}} \approx 0.15$ s for the simulation group G_{no} and and $k_{\text{sim}} \approx k_{\text{out}} \approx 0.15$ s for group G_{wo}. From these results, we conclude that in order to produce one minute worth of data sampled at a rate typically used in clinical EEG recordings, *NFTsim* takes about four minutes to run the simulation and write the output to disk. Thus, *NFTsim*'s simulation length to real-time data length ratio ($T_{\text{sim}}/T_{\text{real}}$) for EEG-compatible outputs is approximately 4. To reduce this ratio users can decrease the size of the output O, by writing only a few relevant variables to disk.

While these benchmarks offer a narrow view of *NFTsim*'s performance, they are a

Table 5. Benchmarks for different grid sizes using *NFTsim* v.0.1.5.

`Nodes`	Storage [GB]	Memory [MB]	Runtime G_{no} [s]	Runtime G_{wo} [s]
144	1.1	5	24	44
256	1.9	6	40	79
1024	7.3	16	161	302
4096	29	51	646	1208
16384	116	200	2601	4908

The value of `Nodes` is reported on the first column. The second column informs the storage size of the output file with all the model variables for each node (firing rate, voltages, fields, coupling strengths) sampled at $1/\Delta t_{out}$ =512 Hz for the group of simulations G_{wo}. The third column presents the total memory usage of the process. Reported runtimes on the fourth column and fifth columns are the time elapsed in seconds. The values on the fourth column correspond to simulations with no output (no) written to the output file. The fifth column corresponds to the running times of simulations for which all the model variables for every node of the grid are written to the output file (wo). The same corticothalamic model is used in every simulation with T_{sim} =16 s, and $\Delta t = 2^{-14}$ s.

valuable practical tool for users and provide: (i) estimates of resources required to run simulations; and, (ii) a guide to make informed decisions between the execution runtimes and accuracy (i.e., decreasing the spatial resolution and/or the time step).

Conclusions, Availability, and Future Directions

We have introduced *NFTsim*, a user-ready, extensible and portable suite for numerical simulations of neural activity based on neural field models. *NFTsim* is based on the well established framework of neural field theory [2] and has been validated with both analytic solutions and experimental data. Thus, when working with new models and simulations users can use analytic solutions as a way to validate their results. This feature is unmatched by other tools currently available to simulate dynamical models of brain activity.

Written in C++, *NFTsim* has been tested on a range of Linux distributions (RHEL 6.9, RHEL 7.4, OpenSUSE 13.2, OpenSUSE 42.2). The output of *NFTsim* is written to a plain text file and ancillary modules written in Matlab contain functions to assist in simulation execution, quick analysis and visualization of the results. *NFTsim* thus provides an efficient solution to simulating various continuum spatiotemporal models including spatially uniform (homogeneous) and nonuniform (inhomogeneous) neural field models [77]; systems with heterogeneous time delays between populations [33]; and, the selected format for data storage is simple enough that enables users to choose from a broad selection of tools to perform further analysis and visualization. The development of *NFTsim* follows essential practices of modern open-source scientific software development [95] such as:

(i) The code is licensed under the Apache 2.0 license.

(ii) Our code sources are hosted on Github:
`https://github.com/BrainDynamicsUSYD/nftsim`.

(iii) We use pull requests to review new features and bug fixes.

(iv) Our users can open issues reporting bugs and/or other problems they encounter.

(v) The developer documentation is produced using Doxygen [100].

(vi) A separate manual is provided for end-users.

(vii) Releases are tagged, so users can refer to and download continuously improved versions of the code that are considered stable and tested. For instance, for this paper, we have used v.0.1.5.

Neural field models simulated with *NFTsim* include spatial propagation of signals and have been shown to have ample physiological applications. Most notably, the activity from neural populations can be used to calculate biophysical signals such as LFP, ECoG, or EEG signals, the latter being the most commonly found in previous studies. Other forms of biophysical observables, such as fMRI and VSDI may also be implemented, but require additional modeling work to define how the electrical activity relates to the corresponding measurements (e.g., oxygen consumption, blood flow changes or fluorescence). Further physical effects such as spatial smoothing due to volume conduction in EEG can be implemented as a part of postprocessing modules like `+nf`.

Due to its flexibility and generality, *NFTsim* allows for a systematic study of both healthy and unhealthy brain function. For instance, in [6] the authors used simulations of a full nonlinear EIRS model showed that for parameter values representing typical sleep spindle oscillations. They found that the numerical nonlinear power spectrum had an additional harmonic peak that was neither present in the linear EIRS model nor it was predicted by the analytic linearized power spectrum. This study clearly demonstrated that *NFTsim*'s flexibility allowed for the investigation of nonlinearities, introducing them one at the time in different neural populations. This enabled the authors to determine which anatomical structures and physiological mechanisms were responsible for the dynamics observed in experiments.

Due to its modularity, *NFTsim* is extensible and can accommodate new features presented in theoretical work on neural fields. In fact, a tool like *NFTsim* is essential for the study of nonlinearities and connectivities configurations that do not necessarily follow the random connectivity approximation or are not spatially homogeneous or constant over time. For instance, [68] explored the mechanisms of seizures by incorporating slow currents modulating the bursting behavior of the reticular nucleus in the corticothalamic (EIRS) model; while [37] incorporated a model of synaptic plasticity to the purely excitatory subsystem. These two mechanisms are already implemented in the current version of *NFTsim*. However, further investigation and development work is required before implementing a general mechanism of parameter modulation, which would allow for the study different types and functional forms of neural feedbacks [58, 99].

We remind potential users that *NFTsim*, as any scientific software, should not be used blindly. As a minimal requirement, users should: check that the integration time step is small enough to resolve the simulated dynamics, for example by running the simulation with increased or decreased time steps to check for stability and convergence of the solutions; run simulations for different values of the Courant number; and compare numerical results with known analytic solutions. Artifacts of periodicity introduced by PBCs, illustrated in Fig. 7 can be avoided if the grid's area is larger than that of the actual physical system. In this scenario, waves propagating from the region of interest towards the right edge of the grid would die off before being reintroduced on the left edge. This approximation would be close to the solution in the absence of artificial boundaries in which the region of interest has infinite size. Lastly, the parameter `Interval`, which effectively downsamples the timeseries written to disk, needs to be carefully selected so as to avoid temporal aliasing if there are signals with high-frequency content.

In the present work we have concentrated mainly on a high-level description of the software and presented examples for which model parameters are assumed to be spatially uniform. *NFTsim* already accepts spatial variations in many parameters, although more development work needs to be done to provide general mechanisms of parameter variation.

As mentioned in *Classes and their Interactions*, *NFTsim* currently has white noise in its collection of external driving signals because in the literature [1, 2, 4, 8, 27–30, 38, 62, 101–104] neural field models are typically either initialized or driven by random fields using a white noise process. However, there are several limitations that make white noise a poor choice. The first limitation is that idealized continuous noise is not physically realistic because it has an infinite bandwidth and infinite power. The second limitation is that in computer simulations, where continuous models are inevitably discretized, the bandwidth of a white noise signal depends on the size of the discretization. This dependence implies that if either the time step or the spatial step are reduced, the bandwidth increases and as a result a white noise signal has additional modes (i.e., frequency components). One can use a scaling parameter to adjust the overall power of the discretized driving signal [29, 84]. This scaling has no effect on the resulting spectral shape that is often compared to EEG [29]. The third limitation is that white noise introduces discontinuities in the derivatives of the system, which are continuous and differentiable. For these reasons, it is necessary to incorporate a new type of random stimuli that has similar spectral characteristics to white noise (i.e, a flat power spectrum over a range of relevant frequencies) but that is differentiable in time and space; and its spectral profile does not change under changes of the discretization.

Future work will extend *NFTsim* scientific features by including (i) a new iterative bandlimited noise generation to render the inputs even more biologically realistic; (ii) generalized mechanisms of spatiotemporal variations for different model parameters and variables; (iii) generalized mechanisms of neuromodulation; and, (iv) spherical topology. In addition, a number of technical enhancements will be made such as (i) implement support for output binary files; and (ii) extend and automate unit test coverage to ensure that new additions to the code do not break previous functionality.

Supporting information

Appendix S1. Discretization of the wave equation

In this section we describe the discretization of the wave equation. This method allows us to obtain an equation to advance each field ϕ_{ab} from t to $t + \Delta t$. We remind the reader that the equation relating the field $\phi_{ab}(\mathbf{r}, t)$ to the driving signal $Q_b(\mathbf{r}, t)$ is

$$\left[\frac{1}{\gamma_{ab}^2}\frac{\partial^2}{\partial t^2} + \frac{2}{\gamma_{ab}}\frac{\partial}{\partial t} + 1 - r_{ab}^2\nabla^2\right]\phi_{ab}(\mathbf{r}, t) = Q_b(\mathbf{r}, t). \tag{37}$$

This equation is a damped wave equation for $\phi_{ab}(\mathbf{r}, t)$ with source $Q_b(\mathbf{r}, t)$. The damping is introduced via the first-order derivative term in the same way friction forces enter a vibrating mechanical system; and, by the third term in Eq. (37). This equation can be simplified by making the following substitutions

$$u = \phi_{ab}\exp(\gamma_{ab}t), \tag{38}$$

and

$$w = Q_b\exp(\gamma_{ab}t). \tag{39}$$

We then obtain the undamped wave equation

$$\left[\frac{1}{\gamma_{ab}^2}\frac{\partial^2}{\partial t^2} - r_{ab}^2\nabla^2\right]u(\mathbf{r}, t) = w(\mathbf{r}, t), \tag{40}$$

To solve this differential equation numerically, we replace the temporal and spatial derivatives with finite central difference approximations on a discretized domain. The derivation presented in the following paragraphs solves the Eq. (40) by using explicit methods, that is, the next value of ϕ_{ab} is computed from known past values of u and w and all future time terms appear on the same side of the time stepping equation.

Consider first the term $\partial^2/\partial t^2$ in Eq. (40), and let the superscripts n index time in units of $k = \Delta t$. We can use a Taylor expansion to write

$$\begin{aligned}
\delta_t^2 u^n &= u^{n+1} - 2u^n + u^{n-1}, && (41)\\
&= 2\left(\frac{k^2}{2!}\frac{\partial^2 u}{\partial t^2} + \frac{k^4}{4!}\frac{\partial^4 u}{\partial t^4} + \dots\right), && (42)\\
&= k^2\left(\frac{\partial^2 u}{\partial t^2} + \frac{k^2}{12}\frac{\partial^4 u}{\partial t^4} + \dots\right), && (43)
\end{aligned}$$

where δ_t^2 is the second order central difference operator in time; and, u^{n+1} is the future term we are interested in calculating. Combining Eqs. (41) and (43) yields

$$u^{n+1} = 2u^n - u^{n-1} + k^2\left(\frac{\partial^2 u}{\partial t^2} + \frac{k^2}{12}\frac{\partial^4 u}{\partial t^4} + \dots\right). \tag{44}$$

Note that this approximation is $\mathcal{O}(k^2)$ accurate in time because we use a second order central difference formula to approximate the second order derivative. So, the error is proportional to the square of k. In a similar way, the second order centered finite difference approximation for the second order spatial derivatives are

$$\delta_{x_1}^2 u_{l,m} \equiv u_{l+1,m} - 2u_{l,m} + u_{l-1,m} = h^2\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial x_1^4} + \dots\right), \tag{45}$$

$$\delta_{x_2}^2 u_{l,m} \equiv u_{l,m+1} - 2u_{l,m} + u_{l,m-1} = h^2\left(\frac{\partial^2 u}{\partial x_2^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial x_2^4} + \dots\right), \tag{46}$$

where $h = \Delta x_1 = \Delta x_2$ is the grid spacing and the subscripts m and l index grid points in the orthogonal x_1 and x_2 directions, respectively. The error of the centered difference scheme used here is $\mathcal{O}(h^2)$. We also use:

$$\begin{aligned}
\nabla^2 &= \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}, && (47)\\
&= \frac{1}{h^2}(\delta_{x_1}^2 + \delta_{x_2}^2), && (48)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial^2 u}{\partial t^2} &= \gamma_{ab}^2 r_{ab}^2 \nabla^2 u + \gamma_{ab}^2 w, && (49)\\
\frac{\partial^4 u}{\partial t^4} &= \gamma_{ab}^2 r_{ab}^2 \nabla^2 \frac{\partial^2 u}{\partial t^2} + \gamma_{ab}^2 \frac{\partial^2 w}{\partial t^2}. && (50)
\end{aligned}$$

We then substitute Eqs (49) and (50) into Eq. (44) and obtain

$$\begin{aligned}
u^{n+1} &= 2u^n - u^{n-1}\\
&\quad + k^2\left[\gamma_{ab}^2 r_{ab}^2 \nabla^2 u + \gamma_{ab}^2 w + \frac{k^2}{12}\left(\gamma_{ab}^2 r_{ab}^2 \nabla^2\left(\frac{\partial^2 u}{\partial t^2}\right) + \gamma_{ab}^2 \frac{\partial^2 w}{\partial t^2}\right)\right], && (51)
\end{aligned}$$

and further substitute the term $\dfrac{\partial^2 u}{\partial t^2}$ in Eq. (51) for the right hand side of Eq. (49)

$$\begin{aligned} u^{n+1} &= 2u^n - u^{n-1} \\ &+ k^2 \left[\gamma_{ab}^2 r_{ab}^2 \nabla^2 u + \gamma_{ab}^2 w + \frac{k^2}{12} \left(\gamma_{ab}^2 r_{ab}^2 \nabla^2 \left(\gamma_{ab}^2 r_{ab}^2 \nabla^2 u + \gamma_{ab}^2 w \right) + \gamma_{ab}^2 \frac{\partial^2 w}{\partial t^2} \right) \right]. \end{aligned} \tag{52}$$

By rearranging the terms in Eq. (52) we can express u^{n+1} in terms of u and w

$$\begin{aligned} u^{n+1} &= 2u^n - u^{n-1} \\ &+ k^2 \left[\gamma_{ab}^2 r_{ab}^2 \nabla^2 u + \gamma_{ab}^2 w + \frac{k^2}{12} \left(\gamma_{ab}^4 r_{ab}^4 \nabla^4 u + \gamma_{ab}^4 r_{ab}^2 \nabla^2 w + \gamma_{ab}^2 \frac{\partial^2 w}{\partial t^2} \right) \right], \end{aligned} \tag{53}$$

$$\begin{aligned} u^{n+1} &= 2u^n - u^{n-1} + k^2 \gamma_{ab}^2 r_{ab}^2 \left[\nabla^2 + \frac{k^2 \gamma_{ab}^2 r_{ab}^2}{12} \nabla^4 \right] u \\ &\quad + k^2 \gamma_{ab}^2 \left[1 + \frac{k^2}{12} \frac{\partial^2}{\partial t^2} + \frac{k^2 \gamma_{ab}^2 r_{ab}^2}{12} \nabla^2 \right] w. \end{aligned} \tag{54}$$

We now omit the terms involving ∇^4 since a second order approximation is enough, giving

$$u^{n+1} = 2u^n - u^{n-1} + k^2 \gamma_{ab}^2 r_{ab}^2 \nabla^2 u^n + k^2 \gamma_{ab}^2 \left[1 + \frac{k^2}{12} \frac{\partial^2}{\partial t^2} + \frac{k^2 \gamma_{ab}^2 r_{ab}^2}{12} \nabla^2 \right] w. \tag{55}$$

Next we replace ∇^2 by the approximations defined in Eq. (48) to obtain

$$\begin{aligned} u^{n+1} &= 2u^n - u^{n-1} \\ &+ p^2 (\delta_{x_1}^2 + \delta_{x_2}^2) u^n + k^2 \gamma_{ab}^2 \left[1 + \frac{1}{12} \delta_t^2 + \frac{p^2}{12} \left(\delta_{x_1}^2 + \delta_{x_2}^2 \right) \right] w, \end{aligned} \tag{56}$$

where $p \equiv p_{ab} = k \gamma_{ab} r_{ab} / h$ is the Courant number and is equivalent to Eq. (30). Next, we replace the second order difference operators δ_t^2, $\delta_{x_1}^2$, and $\delta_{x_2}^2$ to obtain an explicit solution to compute the next value in time of $u_{m,l}$:

$$\begin{aligned} u_{m,l}^{n+1} &= 2u_{m,l}^n - u_{m,l}^{n-1} + p^2 (\delta_{x_1}^2 + \delta_{x_2}^2) u_{m,l}^n \\ &\qquad + k^2 \gamma_{ab}^2 \left[1 + \frac{1}{12} \delta_t^2 + \frac{p^2}{12} \left(\delta_{x_1}^2 + \delta_{x_2}^2 \right) \right] w_{m,l}^n, \end{aligned} \tag{57}$$

$$\begin{aligned} u_{m,l}^{n+1} &= 2u_{m,l}^n - u_{m,l}^{n-1} + p^2 (u_{m,l+1}^n + u_{m,l-1}^n + u_{m+1,l}^n + u_{m-1,l}^n - 4u_{m,l}^n) \\ &+ k^2 \gamma_{ab}^2 \left[w_{m,l}^n + \frac{1}{12} (w_{m,l}^{n+1} - 2w_{m,l}^n + w_{m,l}^{n-1}) \right. \\ &\left. + \frac{p^2}{12} \left(w_{m,l+1}^n + w_{m,l-1}^n + w_{m+1,l}^n + w_{m-1,l}^n - 4w_{m,l}^n \right) \right], \end{aligned} \tag{58}$$

$$\begin{aligned} u_{m,l}^{n+1} &= 2u_{m,l}^n - 4p^2 u_{m,l}^n + p^2 (u_{m,l+1}^n + u_{m,l-1}^n + u_{m+1,l}^n + u_{m-1,l}^n) - u_{m,l}^{n-1} \\ &+ \frac{k^2 \gamma_{ab}^2}{12} \left[12 w_{m,l}^n + w_{m,l}^{n+1} - 2w_{m,l}^n + w_{m,l}^{n-1} \right. \\ &\left. + p^2 \left(w_{m,l+1}^n + w_{m,l-1}^n + w_{m+1,l}^n + w_{m-1,l}^n - 4w_{m,l}^n \right) \right], \end{aligned} \tag{59}$$

$$\begin{aligned} u_{m,l}^{n+1} &= (2 - 4p^2) u_{m,l}^n + p^2 (u_{m,l+1}^n + u_{m,l-1}^n + u_{m+1,l}^n + u_{m-1,l}^n) - u_{m,l}^{n-1} \\ &+ \frac{k^2 \gamma_{ab}^2}{12} \left[(10 - 4p^2) w_{m,l}^n + (w_{m,l}^{n+1} + w_{m,l}^{n-1}) \right. \\ &\left. + p^2 \left(w_{m,l+1}^n + w_{m,l-1}^n + w_{m+1,l}^n + w_{m-1,l}^n \right) \right]. \end{aligned} \tag{60}$$

From Eqs (38) and (39), $u^n = \phi^n e^{\gamma_{ab} n \Delta t}$ and $w^n = \phi^n e^{\gamma_{ab} n \Delta t}$. Also, for a single simulation step, the current state is centered at $t = 0$ and thus indexed by $n = 0$; the next and previous states are ± 1 step away, or equivalently $\pm \Delta t$. Then, $n + 1$ denotes time Δt and $n - 1$ denotes time $-\Delta t$. Therefore we define the following substitutions

$$\begin{aligned} u^{n+1} &= \phi^{n+1} e^{\gamma_{ab}\Delta t}, && (61)\\ u^{n} &= \phi^{n}, && (62)\\ u^{n-1} &= \phi^{n-1} e^{-\gamma_{ab}\Delta t}, && (63)\\ w^{n+1} &= Q^{n+1} e^{\gamma_{ab}\Delta t}, && (64)\\ w^{n} &= Q^{n}, && (65)\\ w^{n-1} &= Q^{n-1} e^{-\gamma_{ab}\Delta t}, && (66) \end{aligned}$$

The spatial indices are omitted for compactness but can take the values $\{m, m \pm 1\}$ and $\{l, l \pm 1\}$. Hence, Eq. (60) can be expressed in terms of ϕ and Q as

$$\begin{aligned} \phi^{n+1}_{m,l} e^{\gamma_{ab}\Delta t} = (2-4p^2)\phi^n_{m,l} + p^2(\phi^n_{m,l+1} + \phi^n_{m,l-1} + \phi^n_{m+1,l} + \phi^n_{m-1,l}) - \phi^{n-1}_{m,l} e^{-\gamma_{ab}\Delta t} \\ + \frac{k^2\gamma^2_{ab}}{12}\Big[(10-4p^2)Q^n_{m,l} + (Q^{n+1}_{m,l} e^{\gamma_{ab}\Delta t} + Q^{n-1}_{m,l} e^{-\gamma_{ab}\Delta t}) \\ + p^2\left(Q^n_{m,l+1} + Q^n_{m,l-1} + Q^n_{m+1,l} + Q^n_{m-1,l}\right)\Big]. \qquad (67) \end{aligned}$$

Finally, upon multiplying both sides of Eq. (67) by $e^{-\gamma_{ab}\Delta t}$ one finds

$$\begin{aligned} \phi^{n+1}_{m,l} = e^{-\gamma_{ab}\Delta t}\Big\{(2-4p^2)\phi^n_{m,l} + p^2(\phi^n_{m,l+1} + \phi^n_{m,l-1} + \phi^n_{m+1,l} + \phi^n_{m-1,l}) - \phi^{n-1}_{m,l} e^{-\gamma_{ab}\Delta t} \\ + \frac{k^2\gamma^2_{ab}}{12}\Big[(10-4p^2)Q^n_{m,l} + (Q^{n+1}_{m,l} e^{\gamma_{ab}\Delta t} + Q^{n-1}_{m,l} e^{-\gamma_{ab}\Delta t}) \\ + p^2\left(Q^n_{m,l+1} + Q^n_{m,l-1} + Q^n_{m+1,l} + Q^n_{m-1,l}\right)\Big]\Big\}. \qquad (68) \end{aligned}$$

Eq. (68) is the formula to advance an axonal field ϕ_{ab} one time step based on its current state (n) and previous state ($n - 1$) when ϕ_{ab} is governed by Eq. (13).

Appendix S2. Configuration file used in Analysis and Visualization.

```
e-erps-all-nodes.conf - configuration file for one-population neural field model.
All parameters are in SI units.

Time: 0.25 Deltat: 2.4414e-4
Nodes: 4096

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

Population 1: Excitatory
Length: 0.5
Q: 10
Firing: Function: Sigmoid Theta: 0.01292 Sigma: 0.0038 Qmax: 340
 Dendrite 1: alpha: 83 beta: 769
 Dendrite 2: alpha: 83 beta: 769

Population 2: Stimulation
Length: 0.5
 Stimulus: Superimpose: 2
   Stimulus: Pulse - Onset: 0.03125 Node: 2000 Amplitude:  2
                     Width: 0.001953125 Frequency: 1 Pulses: 1
   Stimulus: Pulse - Onset: 0.06250 Node: 2097 Amplitude: -2
                     Width: 0.001953125 Frequency: 1 Pulses: 1

Propagator 1: Wave - Tau: 0 Range: 0.2 gamma: 30
Propagator 2: Map -

Coupling 1:  Map - nu: 0
Coupling 2:  Map - nu: 1e-4

Output: Node: All Start: 0 Interval: 9.7656e-4
Population:
Dendrite:
Propagator: 1.phi
Coupling:
```

Acknowledgments

The authors thank B. Fulcher, M. Prodanovic, J. A. Roberts and R. G. Townsend for useful discussions and feedback. This work was supported by an Australian Research Council Laureate Fellowship (grant number FL1401000025) and the Australian Research Council Center of Excellence for Integrative Brain Function (grant number CE140100007).

References

1. Robinson PA, Rennie CJ, Rowe DL, O'Connor SC, Gordon E. Multiscale brain modelling. Phil Trans R Soc B. 2005;360(1457):1043–1050. doi:10.1098/rstb.2005.1638.
2. Robinson PA, Rennie CJ, Wright JJ. Propagation and stability of waves of electrical activity in the cortex. Phys Rev E. 1997;56:826–840.
3. Robinson PA, Kim JW. Spike, rate, field, and hybrid methods for treating neuronal dynamics and interactions. Comp Neurosci. 2012;205:283–294.
4. Roberts JA, Robinson PA. Quantitative theory of driven nonlinear brain dynamics. Neuroimage. 2012;62(3):1947–1955. doi:10.1016/j.neuroimage.2012.05.054.
5. Roberts JA, Robinson PA. Modeling absence seizure dynamics: implications for basic mechanisms and measurement of thalamocortical and corticothalamic latencies. J Theor Biol. 2008;253(1):189–201. doi:10.1016/j.jtbi.2008.03.005.
6. Abeysuriya RG, Rennie CJ, Robinson PA. Prediction and verification of nonlinear sleep spindle harmonic oscillations. J Theor Biol. 2014;344:70–77. doi:10.1016/j.jtbi.2013.11.013.
7. Sanz-Leon P, Knock SA, Woodman MM, Domide L, Mersmann J, McIntosh AR, et al. The Virtual Brain: A simulator of primate brain network dynamics. Front Neuroinform. 2013;7(10). doi:10.3389/fninf.2013.00010.
8. Nichols EJ, Hutt A. Neural Field Simulator: Two-dimensional spatio-temporal dynamics involving finite transmission speed. Front Neuroinf. 2015;9(25). doi:10.3389/fninf.2015.00025.
9. Brette R, Rudolph M, Carnevale T, Hines M, Beeman D, Bower JM, et al. Simulation of networks of spiking neurons: A review of tools and strategies. J Comput Neurosci. 2007;23(3):349–398. doi:10.1007/s10827-007-0038-6.
10. 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(8):e1000092. doi:10.1371/journal.pcbi.1000092.
11. Wu H, Robinson PA, Kim JW. Firing responses of bursting neurons with delayed feedback. J Comput Neurosci. 2011;31(1):61–71.
12. Hämäläinen MS, Sarvas J. Realistic conductivity geometry model of the human head for interpretation of neuromagnetic data. IEEE Trans Biomed Eng. 1989;36(2):165–171.
13. Hämäläinen MS. Magnetoencephalography: A tool for functional brain imaging. Brain Topogr. 1992;5(2):95–102.

14. Hämäläinen MS, Hari R, Ilmoniemi RJ, Knuutila J, Lounasmaa OV. Magnetoencephalography-theory, instrumentation, and applications to noninvasive studies of the working human brain. Rev Mod Phys. 1993;65(2):413–497.
15. Gerstner W, Kistler WM. Spiking neuron models. Single neurons, populations, plasticity. Cambridge: Cambridge U. Press; 2002.
16. Izhikevich EM. Simple model of spiking neurons. IEEE Trans Neural Netw. 2003;14(6):1569 – 1572.
17. Goodman DFM, Brette R. Brian: A simulator for spiking neural networks in python. Front Neuroinform. 2008;2:5. doi:10.3389/neuro.11.005.2008.
18. Brette R, Goodman DFM. Vectorized algorithms for spiking neural network simulation. Neural Comput. 2011;23(6):1503–1535.
19. Hines ML, Davison AP, Muller E. NEURON and Python. Front Neuroinf. 2009;3(1). doi:10.3389/neuro.11.001.2009.
20. Gewaltig M, Diesmann M. NEST (Neural Simulation Tool). Scholarpedia. 2007;2(4):1430.
21. Pecevski D, Natschläger T, Schuch K. PCSIM: A Parallel Simulation Environment for Neural Circuits Fully Integrated with Python. Front Neuroinform. 2009;3:11. doi:10.3389/neuro.11.011.2009.
22. Hoang RV, Tanna D, Jayet Bray LC, Dascalu SM, Harris FC Jr. A novel CPU/GPU simulation environment for large-scale biologically realistic neural modeling. Front Neuroinf. 2013;7:19. doi:10.3389/fninf.2013.00019.
23. Spiegler A, Jirsa VK. Systematic approximations of neural fields through networks of neural masses in the virtual brain. Neuroimage. 2013;83C:704–725. doi:10.1016/j.neuroimage.2013.06.018.
24. Breakspear M. Dynamic models of large-scale brain activity. Nature Rev Neurosci. 2017;20(3):340–352.
25. Kerr CC, Van Albada SJ, Neymotin SA, Chadderdon GL, Robinson PA, Lytton WW. Cortical information flow in Parkinson's disease: A composite network/field model. Front Comput Neurosci. 2013;7:39. doi:10.3389/fncom.2013.00039.
26. 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(1):21–38. doi:10.1016/j.clinph.2009.09.021.
27. Rennie CJ, Robinson PA, Wright JJ. Unified neurophysical model of EEG spectra and evoked potentials. Biol Cybern. 2002;86(6):457–471. doi:10.1007/s00422-002-0310-9.
28. Robinson PA, Rennie CJ, Rowe DL. Dynamics of large-scale brain activity in normal arousal states and epileptic seizures. Phys Rev E. 2002;65:041924.
29. Abeysuriya RG, Rennie CJ, Robinson PA, Kim JW. Experimental observation of a theoretically predicted nonlinear sleep spindle harmonic in human EEG. J Clin Neurophysiol. 2014;125(10):2016–2023.

30. 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(3):413–433. doi:DOI: 10.1016/j.jtbi.2004.07.004.

31. Robinson PA, Rennie CJ, Wright JJ, Bourke PD. Steady states and global dynamics of electrical activity in the cerebral cortex. Phys Rev E. 1998;58:3557–3571. doi:10.1103/PhysRevE.58.3557.

32. Robinson PA, Rennie CJ, Rowe DL, O'Connor SC. Estimation of multiscale neurophysiologic parameters by electroencephalographic means. Hum Brain Mapp. 2004;23(1):53–72. doi:10.1002/hbm.20032.

33. Robinson PA, Whitehouse RW, Rennie CJ. Nonuniform corticothalamic continuum model of electroencephalographic spectra with application to split-alpha peaks. Phys Rev E. 2003;68:021922.

34. Kerr CC, Rennie CJ, Robinson PA. Model-based analysis and quantification of age trends in auditory evoked potentials. Clin Neurophysiol. 2011;122:134–147. doi:10.10.16/j.clinph.2010.05.030.

35. 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(9):1296–1313. doi:10.1093/cercor/bhj072.

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

37. Fung PK, Robinson PA. Neural field theory of synaptic metaplasticity with applications to theta burst stimulation. J Theor Biol. 2014;340(7):164–176.

38. Abeysuriya RG, Rennie CJ, Robinson PA. Physiologically based arousal state estimation and dynamics. J Neurosci Meth. 2015;253:55–69. doi:10.1016/j.jneumeth.2015.06.002.

39. Fung PK, Haber AL, Robinson PA. Neural field theory of large-scale synaptic plasticity in the cerebral cortex. J Theor Biol. 2013;318:44–57.

40. Robinson PA. Neural Field Theory of Synaptic Plasticity. J Theor Biol. 2011;285:156–163.

41. Wilson MT, Goodwin DP, Brownjohn PW, Shemmell J, Reynolds JNJ. Numerical modelling of plasticity induced by transcranial magnetic stimulation. J Comp Neurosci. 2013;36(3):499–514. doi:10.1007/S10827-013-0485-1.

42. Robinson PA, Loxley PN, O'Connor SC, Rennie CJ. Modal analysis of corticothalamic dynamics, electroencephalographic spectra, and evoked potentials. Phys Rev E. 2001;63:041909. doi:10.1103/PhysRevE.63.041909.

43. Kim JW, Robinson PA. Compact dynamical model of brain activity. Phys Rev E. 2007;75:031907.

44. Robinson PA, Sarkar S, Pandejee GM, Henderson JA. Determination of effective brain connectivity from functional connectivity with application to resting state connectivities. Phys Rev E. 2014;90(1):012707. doi:10.1103/PhysRevE.90.012707.

45. Amari S. Homogeneous nets of neuron-like elements. Biol Cybern. 1975;17:211–220.

46. Amari S. Dynamics of pattern formation in lateral inhibition type neural fields. Biol Cybern. 1977;27:77–87.

47. Wilson HR, Cowan JD. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. Kybernetik. 1973;13:55–80.

48. Freeman WJ. Mass Action in the Nervous System. New York: Academic; 1975.

49. Nunez PL. Neocortical Dynamics and Human EEG Rhythms. Oxford: Oxford University Press; 1995.

50. Wright JJ, Liley DTJ. Dynamics of the brain at global and microscopic scales: Neural networks and the EEG. Behav Brain Sci. 1996;19:285–320.

51. Lopes da Silva FH, Hoeks A, Smits H, Zetterberg LH. Model of brain rhythmic activity. The alpha-rhythm of the thalamus. Kybernetik. 1974;15:27–37.

52. Ermentrout B. Neural networks as spatio-temporal pattern-forming systems. Rep Prog Phys. 1998;61(4):353.

53. Jirsa VK, Haken H. Field Theory of Electromagnetic Brain Activity. Phys Rev Lett. 1996;77:960–963.

54. Sanz-Leon P, Knock SA, Spiegler A, Jirsa VK. Mathematical framework for large-scale brain network modeling in The Virtual Brain. Neuroimage. 2015;111:385–430. doi:10.1016/j.neuroimage.2015.01.002.

55. Heitmann S, Breakspear M. Handbook for the Brain Dynamics Toolbox. 1st ed. Brisbane: QIMR Berghofer Medical Research Institute; 2017.

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

57. O'Connor SC, Robinson PA, Chiang AKI. Wave-number spectrum of electroencephalographic signals. Phys Rev E. 2002;66(6):1–12. doi:10.1103/PhysRevE.66.061905.

58. Robinson PA, Roy N. Neural field theory of nonlinear wave-wave and wave-neuron processes. Phys Rev E. 2015;91(6-1):062719.

59. Bojak I, Liley DTJ. Axonal velocity distributions in neural field equations. PLoS Comput Biol. 2010;6(1):e1000653. doi:10.1371/journal.pcbi.1000653.

60. Roberts JA, Robinson PA. Modeling distributed axonal delays in mean-field brain dynamics. Phys Rev E. 2008;78:051901.

61. Nunez PL. Wavelike properties of the alpha rhythm. IEEE Trans Biomed Eng. 1974;21:473–482.

62. Steyn-Ross ML, Steyn-Ross DA, Sligh JW, Liley DTJ. Theoretical electroencephalogram stationary spectrum for a white-noise-driven cortex: Evidence for a general anesthetic-induced phase transition. Phys Rev E. 1999;60:7299–7311.

63. Kandel ER, Schwartz JH, Jessell TM. Principles of Neural Science. 4th ed. New York: McGraw-Hill; 2000.

64. Rubino D, Robbins KA, Hatsopoulos NG. Propagating waves mediate information transfer in the motor cortex. Nat Neurosci. 2006;9:1549–1557.
65. Xu W, Huang X, Takagaki K, Wu JY. Compression and reflection of visually evoked cortical waves. Neuron. 2007;55:119–129.
66. Schiff SJ, Huang X, Wu JY. Dynamical evolution of spatiotemporal patterns in mammalian middle cortex. Phys Rev Lett. 2007;98:178102.
67. Beurle RL. Properties of a Mass of Cells Capable of Regenerating Pulses. Phil Trans R Soc B. 1956;240:55–94.
68. Zhao X, Robinson PA. Generalized seizures in a neural field model with bursting dynamics. J Comput Neurosci. 2015;39(2):197–216. doi:10.1007/s10827-015-0571-7.
69. Rennie CJ, Wright JJ, Robinson PA. Mechanisms of cortical electrical activity and emergence of gamma rhythm. J Theor Biol. 2000;205:17–35.
70. Kerr CC, Rennie CJ, Robinson PA. Physiology-based modeling of cortical auditory evoked potentials. Biol Cybern. 2008;98(2):171–184. doi:10.1007/s00422-007-0201-1.
71. Rennie CJ, Robinson PA, Wright JJ. Effects of local feedback on dispersion of electrical waves in the cerebral cortex. Phys Rev E. 1999;59:3320–3329. doi:10.1103/PhysRevE.59.3320.
72. Clearwater JM, Rennie CJ, Robinson PA. Mean field model of acetylcholine mediated dynamics in the thalamocortical system. J Theor Biol. 2008;255:287–298.
73. International Organization for Standardization. ISO International Standard 8601 – Date and time format. 2nd ed. Geneva: ISO; 2004. Available from: `https://www.iso.org/iso-8601-date-and-time-format.html`.
74. International Organization for Standardization. International Standard ISO/IEC 14882:2011 – Information Technology - Programming Languages - C++. 3rd ed. Geneva: ISO/IEC; 2011. Available from: `https://webstore.iec.ch/publication/21240`.
75. Meyers S. Overview of the new C++ (C++11/14). 1st ed. Walnut Creek: Aritm2015.
76. Sanz-Leon P, Robinson PA. Multistability in the corticothalamic system. J Theor Biol. 2017; 432:141-156
77. 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;70:110–119.
78. O'Connor, SC and Robinson, PA . Unifying and interpreting the spectral wavenumber content of EEGs, ECoGs, and ERPs. J Theor Biol. 2004;231:386–412.
79. Robinson PA. Propagator theory of brain dynamics. Phys Rev E. 2005;72:011904-1–011904-13.
80. Steriade M, Jones EG, McCormick D. Thalamus: Experimental and clinical aspects. 1st ed. Oxford: Elsevier; 1997.

81. List M, Ebert P, Albrecht F. Ten Simple Rules for Developing Usable Software in Computational Biology. PLoS Comput Biol. 2017;13(1):1-5. doi:10.1371/journal.pcbi.1005265.

82. Courant R, Friedrichs K, Lewy H. On the Partial Difference Equations of Mathematical Physics. IBM J Res Dev. 1967;11(2):215–234. doi:10.1147/rd.112.0215.

83. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes in C: The Art of Scientific Computing. 2nd ed. New York: Cambridge University Press; 1992.

84. Rennie C. Modeling the large-scale electrical activity of the brain. Ph.D. Thesis. The University of Sydney. 2001. Available from: `http://www.brain-dynamics.net/~chris_rennie/thesis/thesis.pdf`.

85. Roth G, Mellor-Crummey J, Kennedy K, Brickner RG. Compiling Stencils in High Performance Fortran. In: Proceedings of High Performance Networking and Computing; 1997. pp. 15–21.

86. Hamilton B, Bilbao S. On finite difference schemes for the 3D wave equation using non-cartesian grids. In: Proceedings of Stockholm Musical Acoustics Conference/Sound and Music Computing Conference; 2013. pp.592–599.

87. Srinivasan R, Nunez PL, Silberstein RB. Spatial Filtering and Neocortical Dynamics: Estimates of EEG coherence. IEEE Trans Biomed Eng. 1998;45(7):814–826. doi:10.1109/10.686789.

88. Ogawa S, Lee TM. Magnetic resonance imaging of blood vessels at high fields: in vivo and in vitro measurements and image simulation. Magn Reson Med. 1990;16(1):9–18.

89. Grinvald A, Hildesheim R. VSDI: A new era in functional imaging of cortical dynamics. Nat Rev Neurosci. 2004;5(11):874–875.

90. Chemla S, Chavane F. A biophysical cortical column model to study the multi-component origin of the VSDI signal. NeuroImage. 2010;53(2):420–438.

91. Markounikau V, Igel C, Grinvald A, Jancke D. A dynamic neural field model of mesoscopic activity captured with voltage-sensitive dye imaging. PLoS Comput Biol. 2010;6(9):1–14.

92. Erhardt EB, Allen EA, Yonghua W, Eichele T. StimTB, a simulation toolbox for fMRI data under a model of spatiotemporal separability. NeuroImage. 2010;59(4):4160–4167.

93. Rankin J, Chavane F. Neural field model to reconcile structure with function in primary visual cortex. PLoS Comput Biol. 2017;13(10):1–30.

94. Bhalla US, Bilitch DH, Bower JM. Rallpacks: A set of benchmarks for neuronal simulators. In: Computation and Neural Systems; 1993. pp. 133–140.

95. Gewaltig MO, Cannon R. Current practice in software development for computational neuroscience and how to improve it. PLoS Comput Biol. 2014;10(1):e1003376. doi:10.1371/journal.pcbi.1003376.

96. Friston KJ, Mechelli A, Turner R, Price CJ. Nonlinear responses in fMRI: The Balloon model, Volterra kernels, and other hemodynamics. NeuroImage. 2000;12(4):466–477.

97. Drysdale PM, Huber JP, Robinson PA, Aquino KM. Spatiotemporal BOLD dynamics from a poroelastic hemodynamic model. J Theor Biol. 2010;265:524-–534. doi:10.1016/j.jtbi.2010.05.026.

98. Aquino KM, Schira MM, Robinson PA, Drysdale PM, Breakspear M. Hemodynamic traveling waves in human visual cortex. PLoS Comput Biol. 2012;8(3):e1002435. doi:10.1371/journal.pcbi.1002435.

99. Roy N, Sanz-Leon P, Robinson PA. Spectral signatures of local neural feedback in the corticothalamic system. Phys Rev E. 2017 Oct 26:Forthcoming.

100. van Heesch, Dimitri. Doxygen Qt-interest Archive. 1997. Available from: `http://www.stack.nl/~dimitri/doxygen/`.

101. Kilpatrick ZP, Ermentrout B. Wandering bumps in stochastic neural fields. SIAM J Appl Dyn Syst.2013;12:61-–94.

102. Faugeras O, Inglis J. Stochastic neural field equations: a rigorous footing. J Math Biol. 2015;71:259–300.

103. Thul R and Coombes S and Laing, CR. Neural Field Models with Threshold Noise J Math Neurosci. 2016;6(1). doi:10.1186/s13408-016-0035-z.

104. Bressloff PC, Webber MA. ront propagation in stochastic neural fields. SIAM J Appl Dyn Syst.2012;11(2):708–740.