Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 221 (2007) 781–798
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Numerical methods for solving moment equations in kinetic theory of neuronal network dynamics

Aaditya V. Rangan [a,*], David Cai [a], Louis Tao [b]

[a] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, United States
[b] Department of Mathematical Sciences, New Jersey Institute of Technology, New York, NJ 07102, United States

Received 17 November 2005; received in revised form 13 June 2006; accepted 28 June 2006
Available online 14 August 2006

Abstract

Recently developed kinetic theory and related closures for neuronal network dynamics have been demonstrated to be a powerful theoretical framework for investigating coarse-grained dynamical properties of neuronal networks. The moment equations arising from the kinetic theory are a system of $(1+1)$-dimensional nonlinear partial differential equations (PDE) on a bounded domain with nonlinear boundary conditions. The PDEs themselves are self-consistently specified by parameters which are functions of the boundary values of the solution. The moment equations can be stiff in space and time. Numerical methods are presented here for efficiently and accurately solving these moment equations. The essential ingredients in our numerical methods include: (i) the system is discretized in time with an implicit Euler method within a spectral deferred correction framework, therefore, the PDEs of the kinetic theory are reduced to a sequence, in time, of boundary value problems (BVPs) with nonlinear boundary conditions; (ii) a set of auxiliary parameters is introduced to recast the original BVP with nonlinear boundary conditions as BVPs with linear boundary conditions – with additional algebraic constraints on the auxiliary parameters; (iii) a careful combination of two Newton's iterates for the nonlinear BVP with linear boundary condition, interlaced with a Newton's iterate for solving the associated algebraic constraints is constructed to achieve quadratic convergence for obtaining the solutions with self-consistent parameters. It is shown that a simple fixed-point iteration can only achieve a linear convergence for the self-consistent parameters. The practicability and efficiency of our numerical methods for solving the moment equations of the kinetic theory are illustrated with numerical examples. It is further demonstrated that the moment equations derived from the kinetic theory of neuronal network dynamics can very well capture the coarse-grained dynamical properties of integrate-and-fire neuronal networks.

Keywords: Newton's method; Divide and conquer

* Corresponding author. Tel.: +1 212 998 3303; fax: +1 212 995 4121.
E-mail address: rangan@cims.nyu.edu (A.V. Rangan).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.06.036

1. Introduction

Neuronal networks with statistically homogeneous couplings, such as all-to-all couplings with or without synaptic failures, have served as protypical theoretical models for providing basic insight into the fascinating network dynamics of large numbers of neurons in the brain. Applications of fundamental concepts and theoretical tools from nonequilibrium statistical physics to this type of neuronal network dynamics have proven to be quite fruitful. Recently, starting with "microscopic" networks of homogeneously coupled, conductance-based integrate-and-fire (I&F) neurons, $(1+1)$-dimensional moment equations have been derived via the maximum-entropy principle (without any new parameters introduced) from a $(2+1)$-dimensional Boltzmann-like kinetic equation that governs the evolution of a one-particle (i.e., one-neuron) probability density function (pdf) [1–3]. The approach is reminiscent of derivations of hydrodynamic equations from the Boltzmann equation for molecular motion in fluids [4,5]. This closure theory provides a coarse-grained theoretical framework for describing the statistics of a neuronal network of conductance-based integrate-and-fire neurons. Importantly, in contrast to the mean-field approach [6–8], it can capture many of the features of integrate-and-fire networks, such as fluctuation effects when these networks are within a fluctuation-dominated regime. For example, the moment equations can consistently describe intrinsic dynamic synaptic fluctuations arising from network interactions. Specifically, the kinetic moment equations can accurately describe the dynamic evolution of firing rate and voltage distributions for a neuronal network where the mean in the input to the network is not sufficient to cause any firing events but the fluctuations in the input can. The kinetic theory with its associated closures has been demonstrated to be a powerful approach [1] in that it can serve as (i) an analytical tool for examining the coarse-grained network dynamics, (ii) a computational tool for efficiently evaluating statistical information of integrate-and-fire neuronal networks and (iii) a useful theoretical basis for constructing coarse-grained sub-networks within a larger heterogeneously structured integrate-and-fire network [9], such that the entire system can be described by a kinetic theory of interacting populations of coarse-grained subsystems, each of which contains neurons with similar connectivity and response properties.

In this paper, we present numerical methods for efficiently and accurately solving the moment equations derived from the kinetic theory for neuronal network dynamics. The moment equations themselves are $(1+1)$-dimensional *nonlinear* partial differential equations (PDE), which are self-consistently specified by parameters that are functions of the boundary values of the solution on a bounded domain, with *nonlinear* boundary conditions. Our numerical method for solving these equations has three basic ingredients. The first step is to discretize time and apply an implicit Euler method within a spectral deferred correction framework [10–14]. This process reduces the PDE of the closure theory to a sequence of boundary value problems (BVPs) with nonlinear boundary conditions. The second step is to introduce auxiliary parameters which recast the original BVPs with nonlinear boundary conditions as BVPs with *linear* boundary conditions but with additional algebraic constraints on the auxiliary parameters. The third step is to actually determine the solution to each boundary value problem as well as the corresponding (constrained) auxiliary parameters using a modified version of Newton's method.

Each main step within our numerical algorithm has distinct computational advantages. First, discretizing in time (without discretizing in space) allows us to naturally incorporate the correct boundary conditions into our scheme with ease. These nonlinear boundary conditions (of the kinetic equations) are difficult to encapsulate correctly using standard finite difference schemes, which discretize both space and time simultaneously. Second, the use of the implicit Euler method avoids the restrictive CFL conditions associated with explicit time-stepping (since the "wave-speeds" associated with the moment equations can be quite large, especially when the voltage distribution becomes small (see below)), while the spectral deferred correction framework allows our algorithm to remain accurate even when taking large numerical time-steps ($\Delta t \approx 1$ ms even when the smallest time-scale of the moment equations is $\sim$0.5 ms). Third, the introduction of auxiliary parameters linearizes the boundary conditions of each BVP, and allows us to employ the fast spectral integral methods of [15,16], which use a divide-and-conquer strategy to adaptively discretize space as necessary and generate a highly accurate differentiable solution for each linear BVP. Finally, at each time-step, a careful choice of Newton's method in enforcing the associated algebraic constraints related to each BVP allows us to determine the correct parameters with a quadratic error rate. In contrast, a simple fixed-point iteration has only a linear convergence in error rate, as will be seen below.

The most practicable implementation of our algorithm involves one pass of spectral deferred correction (which yields a formally second order accurate method, as well as an useable error estimate) with each substep involving 2–3 iterates of Newton's method for solving the BVP and associated algebraic constraints – here, the relevant Jacobians need only be updated about once or twice every correction step. Using our method, high relative accuracy (errors of 10^{-4} or less) can typically be achieved with large time-steps ($\Delta t \approx 0.5$ ms).

The outline of the paper is as follows. In Section 2, the kinetic theory formulation for neuronal network dynamics will be briefly reviewed. In Section 3, the basic implicit Euler scheme will be described, along with the approach of Newton's method for solving the algebraic constraint associated with each BVP. In Section 4, the spectral deferred correction framework will be detailed, which provides a systematic way of improving the accuracy of the initial approximate solution. In Section 5, numerical examples are presented to illustrate the convergence rate and efficiency of our numerical methods, and to demonstrate the efficacy of our algorithm for networks that operate in a fluctuation-dominated regime. There, we also provide a simple geometric construction underlying the quadratic convergence of our combination of Newton's schemes. In Section 6, we present our conclusions.

2. Kinetic theory

To motivate, we briefly review the kinetic theory formulation for neuronal network dynamics [1,3,2]. For conceptual simplicity, we consider an all-to-all coupled network of N excitatory integrate-and-fire neurons [17]. The ith model neuron's state is determined by its membrane voltage $V_i(t)$ and excitatory conductance $G_i(t)$. Their dynamics evolve according to the integrate-and-fire equations:

$$\tau_v \frac{d}{dt} V_i(t) = -(V_i(t) - \epsilon_r) - G_i(t)(V_i(t) - \epsilon_E), \tag{1a}$$

$$\tau_g \frac{d}{dt} G_i(t) = -G_i(t) + f \sum_k \delta(t - T^{ext}_{i,k}) + \frac{S}{N} \sum_{j,k} \delta(t - T_{j,k}), \tag{1b}$$

where τ_v and τ_g are the decay time constants for the voltage and conductance, respectively. The voltage evolves continuously until it crosses the threshold ϵ_θ, at which point it resets to the reset voltage ϵ_r and the time at which it crosses ϵ_θ is recorded as the firing (spiking) time of the neuron. (From physiology, typical values are $\epsilon_r = -65$ mV, $\epsilon_\theta = -55$ mV and $\epsilon_E = 0$ mV. We rescale voltage such that $\epsilon_r = 0$, $\epsilon_\theta = 1$, and the excitatory reversal potential $\epsilon_E = \frac{14}{3}$ in our rescaled units [18].) Physiologically, when a neuron fires, it emits a spike which affects the conductance of all the neurons in the network. Here, the conductance G_i has a jump of size f/τ_g upon receiving a spike from the feedforward (external) input, as signified by $\delta(t - T^{ext}_{i,k})$, where $T^{ext}_{i,k}$ is the kth spike in the feedforward (external) input to the ith neuron. Similarly, G_i has a jump of size $S/(N\tau_g)$ upon receiving a spike from a neuron in the network, as signified by $\delta(t - T_{j,k})$, where the times $T_{j,k}$ record the kth spike of the jth neuron in the network. The parameters f and S determine the size of conductance increase associated with the spike from the feedforward input and from other neurons in the network, respectively. S/N describes the strength of network couplings and the factor $1/N$ ensures that there is a well-defined network coupling in the limit of $N \to \infty$.

For a fixed neuron j, the output spike statistics of $\{T_{j,k}\}$ is, in general, not Poisson. However, the input to the ith neuron is a spike train summed over output spike trains from many neurons in the network. If we assume that each neuronal firing event has a very low rate and is statistically independent from the other neurons in the network, then the spike train obtained by summing over a large number of output spike trains of neurons in the network asymptotically tends to be a Poisson spike process [19]. Therefore, it can be assumed that the input spike train summed from all other neurons to the ith neuron is Poisson with rate $Nm(t)$, where $m(t)$ is the population-averaged firing rate per neuron. Further, it can be assumed that the feedforward spikes are described by a Poisson process with rate ν. With these assumptions, a $(2+1)$-dimensional Boltzmann-like kinetic equation governing the evolution of a one-particle (i.e., one neuron) probability distribution function $\rho(v, g, t)$ can be derived [2]:

$$\partial_t \rho(v,g,t) = \partial_v\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\rho(v,g,t) + g\left(\frac{v-\epsilon_E}{\tau_v}\right)\rho(v,g,t)\right] + \partial_g\left[g\frac{1}{\tau_g}\rho(v,g,t)\right] + v\left[\rho\left(v,g-\frac{f}{\tau_g},t\right) - \rho(v,g,t)\right] + Nm\left[\rho\left(v,g-\frac{S}{N\tau_g},t\right) - \rho(v,g,t)\right], \tag{2}$$

where $\rho(v,g,t)\mathrm{d}v\,\mathrm{d}g$ is the probability of finding a neuron whose voltage is in $(v, v+\mathrm{d}v)$ and whose conductance is in $(g, g+\mathrm{d}g)$ at time t. If the jumps f and S/N are small, a Taylor expansion yields a diffusion approximation of Eq. (2),

$$\partial_t \rho(v,g,t) = \partial_v\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\rho(v,g,t) + g\left(\frac{v-\epsilon_E}{\tau_v}\right)\rho(v,g,t)\right] + \partial_g\left[g\frac{1}{\tau_g}\rho(v,g,t)\right] + \frac{v}{\tau_g}\left[-f\partial_g\rho(v,g,t) + \frac{f^2}{2\tau_g}\partial_g^2\rho(v,g,t)\right] + \frac{Nm}{\tau_g}\left[-\frac{S}{N}\partial_g\rho(v,g,t) + \frac{S^2}{2N^2\tau_g}\partial_g^2\rho(v,g,t)\right],$$

which can be rewritten as

$$\partial_t \rho(v,g,t) = \partial_v\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\rho(v,g,t) + g\left(\frac{v-\epsilon_E}{\tau_v}\right)\rho(v,g,t)\right] + \frac{1}{\tau_g}\partial_g[(g-\bar{g})\rho(v,g,t) + \hat{g}\partial_g\rho(v,g,t)] \tag{3}$$

with

$$\bar{g} = fv + Sm, \tag{4a}$$

$$\hat{g} = \frac{1}{2\tau_g}\left(f^2 v + \frac{S^2 m}{N}\right). \tag{4b}$$

Eq. (3) expresses the conservation of probability. The flux along the v-direction is

$$J_V(v,g,t) = -\left[\left(\frac{v-\epsilon_r}{\tau_v}\right) + g\left(\frac{v-\epsilon_E}{\tau_v}\right)\right]\rho(v,g,t),$$

and the flux along the g-direction is

$$J_G(v,g,t) = -\frac{1}{\tau_g}[(g-\bar{g})\rho(v,g,t) + \hat{g}\partial_g\rho(v,g,t)].$$

We have the boundary condition

$$J_V(\epsilon_\theta,g,t) = J_V(\epsilon_r,g,t), \tag{5}$$

reflecting that neurons crossing ϵ_θ have just fired, and all re-enter through the reset voltage. The firing rate (m in our model) is one of the most important quantities measured in physiological experiments to describe neuronal network properties. The dynamics of the network (1) is characterized by $m(t)$ as determined by the total probability flux across the threshold ϵ_θ *regardless* of the values of conductance, i.e.,

$$m(t) = \int J_V(\epsilon_\theta,g,t)\mathrm{d}g. \tag{6}$$

Thus, for a given $\rho(v,g,t)$, we can determine the firing rate. However, Eq. (3) depends on the parameters $\bar{g}(t)$ and $\hat{g}(t)$, which are functions of $m(t)$, which, in turn, depends on the boundary value of $\rho(\epsilon_\theta,g,t)$ through $J_V(\epsilon_\theta,g,t)$. Therefore, Eq. (3) is a nonlinear $(2+1)$-dimensional equation.

Clearly, analytical insights and computational advantage can be achieved if the $(2+1)$-dimensional dynamics [Eq. (3)] can be reduced to a $(1+1)$-dimensional effective dynamics. Now we sketch this reduction to $(1+1)$-dimensional moment equations for the integrate-and-fire neuronal network dynamics.

Integration of Eq. (3) with respect to v yields,

$$\partial_t \rho^{(g)}(g) = \partial_g\left\{\frac{1}{\tau_g}[(g-\bar{g}) + \hat{g}\partial_g]\rho^{(g)}(g)\right\} \tag{7}$$

with $\rho^{(g)}(g) = \int_{\epsilon_r}^{\epsilon_\theta}\rho(v,g,t)\mathrm{d}v$. Defining the zeroth conductance moment

$$\rho(v,t) = \int_0^\infty \rho(v,g,t)\mathrm{d}g,$$

i.e., the voltage probability density function, and the moments

$$\mathscr{X}(v,t) = \int_0^\infty g\rho(v,g,t)\mathrm{d}g, \quad \mathscr{Y}(v,t) = \int_0^\infty g^2\rho(v,g,t)\mathrm{d}g, \ldots,$$

it can be shown that Eq. (3) generates an infinite hierarchy of equations governing the evolution of moments. The first and second equations in this hierarchy are

$$\partial_t\rho(v,t) = \partial_v\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\rho(v,t) + \left(\frac{v-\epsilon_E}{\tau_v}\right)\mathscr{X}(v,t)\right], \tag{8}$$

$$\partial_t\mathscr{X}(v,t) = \partial_v\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\mathscr{X}(v,t) + \left(\frac{v-\epsilon_E}{\tau_v}\right)\mathscr{Y}(v,t)\right] - \frac{1}{\tau_g}[\mathscr{X}(v,t) - \bar{g}\rho(v,t)]. \tag{9}$$

Note that, in this hierarchy, the evolution of $\mathscr{X}(v,t)$ depends on the evolution of $\mathscr{Y}(v,t)$ (as seen in Eq. (9)), i.e., Eqs. (8) and (9) are not closed with respect to $\rho(v,t)$ and $\mathscr{X}(v,t)$. We need to address the closure issues, namely, how to reduce this hierarchy to a closed effective dynamics that involve only lower order moments. Based on the maximum-entropy principle, a closure condition can be derived [3]:

$$\mathscr{Y}(v,t) = \frac{\mathscr{X}^2(v,t)}{\rho(v,t)} + \hat{g}\rho(v,t). \tag{10}$$

Under this closure (10), Eq. (3) leads to closed equations with respect to $\rho(v,t)$ and $\mathscr{X}(v,t)$, which is a conservation law with the following form:

$$\partial_t\rho(v,t) = -\partial_v J^\rho(v,t), \tag{11a}$$

$$\partial_t\mathscr{X}(v,t) = -\partial_v J^{\mathscr{X}}(v,t) - \frac{1}{\tau_g}[\mathscr{X}(v,t) - \bar{g}\rho(v,t)] \tag{11b}$$

with

$$J^\rho(v,t) = -\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\rho(v,t) + \left(\frac{v-\epsilon_E}{\tau_v}\right)\mathscr{X}(v,t)\right],$$

$$J^{\mathscr{X}}(v,t) = -\left[\left(\frac{v-\epsilon_r}{\tau_v}\right)\mathscr{X}(v,t) + \left(\frac{v-\epsilon_E}{\tau_v}\right)\left(\hat{g}\rho(v,t) + \frac{\mathscr{X}(v,t)^2}{\rho(v,t)}\right)\right].$$

Eq. (11) constitute the *closed* moment equations in kinetic theory of the neuronal network dynamics (1). The population-averaged firing rate per neuron in the system is determined via Eq. (6) [1,2]:

$$m(t) = J^\rho(\epsilon_\theta,t) = -\left[\left(\frac{\epsilon_\theta-\epsilon_r}{\tau_v}\right)\rho(\epsilon_\theta,t) + \left(\frac{\epsilon_\theta-\epsilon_E}{\tau_v}\right)\mathscr{X}(\epsilon_\theta,t)\right]. \tag{12}$$

The boundary conditions for this conservation law can be derived from Eq. (5) to yield

$$J^\rho(\epsilon_\theta,t) = J^\rho(\epsilon_r,t),$$

$$J^{\mathscr{X}}(\epsilon_\theta,t) = J^{\mathscr{X}}(\epsilon_r,t).$$

It is convenient to define

$$a(v) \equiv \left(\frac{v-\epsilon_r}{\tau_v}\right) \quad \text{and} \quad b(v) \equiv \left(\frac{v-\epsilon_E}{\tau_v}\right),$$

$$c \equiv \partial_v a = \partial_v b = \frac{1}{\tau_v},$$

and to use the subscripts r, θ to refer to function evaluation at ϵ_r, ϵ_θ, respectively (e.g., $\phi_r = \phi(v)|_{v=\epsilon_r} = \phi(\epsilon_r)$). With this notation we can cast the conservation law as

$$\partial_t \rho = \partial_v [a\rho + b\mathscr{X}], \tag{13a}$$

$$\partial_t \mathscr{X} = \partial_v \left[a\mathscr{X} + b\left(\hat{g}\rho + \frac{\mathscr{X}^2}{\rho} \right) \right] - \frac{1}{\tau_g}[\mathscr{X} - \bar{g}\rho] \tag{13b}$$

with boundary conditions

$$\begin{bmatrix} a_r & b_r \\ b_r\hat{g} & a_r + b_r\mu_r \end{bmatrix} \cdot \begin{bmatrix} \rho_r \\ \mathscr{X}_r \end{bmatrix} - \begin{bmatrix} a_\theta & b_\theta \\ b_\theta\hat{g} & a_\theta + b_\theta\mu_\theta \end{bmatrix} \cdot \begin{bmatrix} \rho_\theta \\ \mathscr{X}_\theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{14}$$

where

$$\mu_r = \frac{\mathscr{X}_r}{\rho_r} \quad \text{and} \quad \mu_\theta = \frac{\mathscr{X}_\theta}{\rho_\theta}. \tag{15}$$

Note that the nonlinear PDEs (11) of the kinetic theory depend on the firing rate parameter m, which is a function of the boundary value of the solution (Eq. (12)). In addition, the boundary conditions (14) are nonlinear (Eq. (15)) and depend on the firing rate m via $\hat{g}$. We remark that, in certain settings, for example, if the initial data is compactly supported within a sufficiently small region in the interior of $[\epsilon_r, \epsilon_\theta]$, then the boundary conditions are automatically satisfied, Eq. (13) reduces to a hyperbolic system (with a source term). It can easily be seen that the wave speeds in this situation can become arbitrarily large when $\rho \ll \mathscr{X}$. This indicates that the convective term $J^{\mathscr{X}}$ in Eq. (11) gives rise to stiff dynamics in general and is difficult to handle with a simple explicit scheme. We further note that in the work of [1,2], a further reduction to a Fokker–Planck equation in the limit of $\tau_g \to 0$ can be achieved [20–22]. However, we will focus on the numerical methods of solving the nonlinear PDE (13) with boundary conditions (14) in this work.

3. Basic first order method and Newton's method for the algebraic constraint

Our goal is to numerically approximate the solution of the nonlinear PDE (13) with boundary conditions (14). At a first glance, Eq. (13) might appear similar to a hyperbolic conservation law with stiff relaxation terms. Efficient splitting methods for accurately resolving these types of stiff conservation laws have been discussed in [23–26]. These splitting methods treat parts of the PDE explicitly, and parts of the PDE implicitly, while ensuring that as the stiffness parameter increases, the discrete numerical method converges to a numerical method which handles the appropriate limiting dissipative PDE (e.g., some sort of diffusion equation). We have implemented a hybrid upwind–downwind numerical method with a similar flavor (i.e., as $\tau_g \to 0$, the numerical method converged to a stable scheme for solving the diffusive, i.e., Fokker–Planck, equation derived from Eq. (13) in the $\tau_g \to 0$ limit) [2]. However, both the methods of [23–26] as well as the hybrid upwind–downwind scheme [2] face two obstacles when tackling Eqs. (13) and (14): (i) the boundary conditions (14) are difficult to correctly incorporate with any sort of finite difference method, and the transient dynamics of PDE (13) are critically dependent on the details of the boundary conditions in our experience of solving these equations; (ii) in a low firing rate regime with moderate τ_g and small f, the system (13) and (14) can develop a very sharp spatial boundary layer near either ϵ_θ or ϵ_r. Resolving this layer with a uniform spatial grid (or, indeed, any *apriori* spatial grid) is impractical. To circumvent these two issues, we discretize time first, and apply the implicit Euler scheme to turn the PDE into a sequence, in time, of boundary value problems (BVPs). The implicit Euler method is only first order, but allows us to naturally and conveniently incorporate the boundary conditions (Eq. (14)) and to take relatively large time-steps ($\Delta t \approx 1$ ms) (the issue of accuracy will be addressed below). In addition, this approach allows us to adaptively discretize space as necessary when solving the individual BVPs, and thus resolve spatial boundary layers as they manifest. We remark that by viewing m, μ_r, μ_θ as auxiliary parameters, we can treat the nonlinear boundary conditions (14) as linear boundary conditions for fixed values of m, μ_r, μ_θ, while simultaneously fully specifying the PDE (13) with this fixed, explicit auxiliary parameter m. Upon this "linearizaton," we can readily solve Eqs. (13) and (14) using existing methods, such as *spectral integral methods* [15,16], for solving standard one-dimensional BVPs with *linear* boundary conditions. The methods of [15,16] are advantageous because they efficiently and adaptively discretize space. In particular, they can easily resolve aforementioned boundary layers with a minimal computational

overhead. After all the treatment, the problem reduces to solving Eq. (13) with auxiliary parameters and simultaneously finding the correct parameters m, μ_r, μ_θ as solutions of the algebraic constraint (Eqs. (12) and (15)).

For clarity of notation, we define

$$Y(v,t) = \begin{bmatrix} \rho(v,t) \\ \mathscr{X}(v,t) \end{bmatrix},$$

$$W(t) = \{m(t), \mu_r(t), \mu_\theta(t)\},$$

$$F(W,Y,\partial_v Y) = \begin{bmatrix} c\rho + c\mathscr{X} + a\partial_v\rho + b\partial_v\mathscr{X} \\ c\left(\mathscr{X} + \hat{g}\rho + \frac{\mathscr{X}^2}{\rho}\right) + a\partial_v\mathscr{X} + b\hat{g}\partial_v\rho + 2b\left(\frac{\mathscr{X}\partial_v\mathscr{X}}{\rho}\right) - b\left(\frac{\mathscr{X}^2}{\rho^2}\right)\partial_v\rho - \frac{1}{\tau_g}[\mathscr{X} - \bar{g}\rho] \end{bmatrix},$$

$$A(W,t) = \begin{bmatrix} a_r & b_r \\ b_r\hat{g}(t) & a_r + b_r\mu_r(t) \end{bmatrix},$$

$$C(W,t) = -\begin{bmatrix} a_\theta & b_\theta \\ b_\theta\hat{g}(t) & a_\theta + b_\theta\mu_\theta(t) \end{bmatrix}.$$

With this notation, the original PDE (13) can be rewritten as

$$\partial_t Y(v,t) = F(W(t), Y(v,t), \partial_v Y(v,t)), \tag{16}$$

with boundary conditions

$$A(W(t),t)Y(\epsilon_r,t) + C(W(t),t)Y(\epsilon_\theta,t) = 0. \tag{17}$$

As mentioned above, to solve Eqs. (16) and (17), we first discretize time as $t_0 < t_1 < \cdots < t_F$, with time-steps $\Delta_n = t_n - t_{n-1}$, and write the numerical solution $Y(t_n), W(t_n)$, as Y^n, W^n. Using the implicit Euler scheme, each time-step becomes a BVP which can be formally written as

$$Y^n - Y^{n-1} = \Delta_n F(W^n, Y^n, \partial_v Y^n) \tag{18}$$

with boundary conditions

$$A(W^n, t_n)Y^n_r + C(W^n, t_n)Y^n_\theta = 0. \tag{19}$$

Note that Eq. (18) with boundary conditions (19) cannot be straightforwardly solved using standard BVP solvers, since the parameter set W^n is a function of the desired solution Y^n.

3.1. Newton's method for BVP

We now outline a Newton's method approach to solve Eqs. (18) and (19). First, we view $W = \{m, \mu_r, \mu_\theta\}$ as auxiliary fixed parameters, computed via Eqs. (12) and (15) from the boundary values of Y^n. Thus, with the fixed parameters W^n, the nonlinear boundary conditions (14) are turned into linear boundary conditions and the BVP (18) is explicitly specified since W^n in Eq. (18) is no longer a function of Y^n. Later, we address the question of how to obtain a consistent set of auxiliary parameters, i.e., those consistent with the solution Y^n.

We begin with $Y^{n,[0]}$, which denotes the old solution from the previous step Y^{n-1}, and use it as the 0th approximation to the new solution at the current step. The parameter $W^{n,[0]}$ is computed directly using the solution $Y^{n,[0]}$ via Eqs. (12) and (15).

By denoting the first Newton's iterate as $Y^{n,[1]}$, we apply one step of Newton's method to the BVP (Eqs. (18) and (19)) in an attempt to solve for Y^n:

$$\begin{aligned}&[I - \Delta_n H(W^{n,[0]}, Y^{n,[0]}, \partial_v Y^{n,[0]}) - \Delta_n K(W^{n,[0]}, Y^{n,[0]}, \partial_v Y^{n,[0]})\partial_v] \cdot [Y^{n,[1]} - Y^{n,[0]}] \\ &\quad = \Delta_n F(W^{n,[0]}, Y^{n,[0]}, \partial_v Y^{n,[0]}) + Y^{n-1} - Y^{n,[0]},\end{aligned} \tag{20}$$

with corresponding boundary conditions

$$A(W^{n,[0]}, t_n)Y^{n,[1]}_r + C(W^{n,[0]}, t_n)Y^{n,[1]}_\theta = 0, \tag{21}$$

where we have defined

$$K = \partial_{\partial_v Y} F = \begin{bmatrix} a & b \\ b\hat{g} - b\frac{\mathscr{X}^2}{\rho^2} & a + 2b\frac{\mathscr{X}}{\rho} \end{bmatrix},$$
$$H = \partial_Y F = \begin{bmatrix} c & c \\ H_{21} & H_{22} \end{bmatrix}$$

with

$$H_{21} = c\hat{g} - c\frac{\mathscr{X}^2}{\rho^2} - 2b\frac{\mathscr{X}\partial_v \mathscr{X}}{\rho^2} + 2b\left(\frac{\mathscr{X}^2}{\rho^3}\right)\partial_v \rho + \frac{\bar{g}}{\tau_g},$$
$$H_{22} = c + 2c\frac{\mathscr{X}}{\rho} + 2b\frac{\partial_v \mathscr{X}}{\rho} - 2b\left(\frac{\mathscr{X}}{\rho^2}\right)\partial_v \rho - \frac{1}{\tau_g}.$$

Note that H and K both depend on the parameters W through $\bar{g}$ and $\hat{g}$ (which are functions of m (Eq. (4)). For fixed parameters W, Eqs. (20) and (21) constitute a linear BVP with linear boundary conditions, therefore, they can be solved efficiently using the domain splitting methods of [16].

Before we address the issue of how to incorporate consistent parameters W^n, we first recast BVP (20) and (21) to obtain a compact expression for $Y^{n,[1]}$. For ease of notation, we define the linear operators

$$L_1(w,y) \equiv [I - \Delta_n H(w,y,\partial_v y) - \Delta_n K(w,y,\partial_v y)\partial_v],$$
$$L_2(w) \equiv A(w,t_n)\cdot|_{v=\epsilon_r} + C(w,t_n)\cdot|_{v=\epsilon_\theta},$$

where the linear operator L_1 represents the ordinary differential operator (see Eq. (20)) and the linear operator L_2 represents the boundary conditions (see Eq. (21)). Therefore, Eqs. (20) and (21) can be cast as

$$L_1(W^{n,[0]}, Y^{n,[0]})\cdot Y^{n,[1]} = R_1(W^{n,[0]}, Y^{n,[0]}), \tag{22a}$$
$$L_2(W^{n,[0]})\cdot Y^{n,[1]} = R_2 \tag{22b}$$

with

$$R_1(W^{n,[0]}, Y^{n,[0]}) \equiv \Delta_n F(W^{n,[0]}, Y^{n,[0]}, \partial_v Y^{n,[0]}) + Y^{n-1} - Y^{n,[0]} + L_1(W^{n,[0]}, Y^{n,[0]})\cdot Y^{n,[0]},$$
$$R_2 \equiv 0.$$

By defining $L(w,y) \equiv [L_1(w,y), L_2(w)]^T$, Eq. (22) combines into a compact form:

$$L(W^{n,[0]}, Y^{n,[0]})\cdot Y^{n,[1]} = R(W^{n,[0]}, Y^{n,[0]}), \tag{23}$$

where the right-hand side $R(w,y) \equiv [R_1(w,y), R_2]^T$. Now the Newton iterate $Y^{n,[1]}$ is simply the solution of

$$Y^{n,[1]} = L^{-1}(W^{n,[0]}, Y^{n,[0]})\cdot R(W^{n,[0]}, Y^{n,[0]}). \tag{24}$$

3.2. Algebraic constraint

Next, we discuss two methods to deal with how to solve for the correct parameters W^n – i.e., those consistent with the solution Y^n. Clearly, the consistent parameters W^n are determined by solving the algebraic constraint:

$$G(W^n, Y^n) = \begin{bmatrix} m^n + a_r\rho_r^n + b_r\mathscr{X}_r^n \\ \mu_r^n - \frac{\mathscr{X}_r^n}{\rho_r^n} \\ \mu_\theta^n - \frac{\mathscr{X}_\theta^n}{\rho_\theta^n} \end{bmatrix} = 0. \tag{25}$$

3.2.1. Fixed-point iteration

As a first attempt, we employ a basic fixed-point iteration to obtain a parameter set W^n consistent with the solution Y^n. Each BVP (24) with these linear boundary conditions is solved using the fast divide-and-conquer strategy of [16], which automatically discretizes space adaptively, and efficiently generates a highly accurate differentiable solution. We repeatedly update the parameter values using the algebraic relations (Eqs. (12) and (15)), and repeat Newton's iteration until the solution is consistent with the firing rate m, and other parameters $\{\mu_r, \mu_\theta\}$ at each time-step. The fixed-point iteration can be simply illustrated with the following pseudocode:

- Set $v = 0$, maximum iteration $v_{\max}$ and tolerance ε.
- Evaluate $W^{n,[0]}$ using the solution for the last step Y^{n-1} via Eqs. (12) and (15).
- do
  {
  set $v := v + 1$.
  Solve the linear BVP (24) to obtain:
  $Y^{n,[v]} = L^{-1}(W^{n,[v-1]}, Y^{n,[v-1]}) \cdot R(W^{n,[v-1]}, Y^{n,[v-1]})$.
  Evaluate $W^{n,[v]}$ using the solution $Y^{n,[v]}$ via Eqs. (12) and (15).
  }
  while ($v \leqslant v_{\max}$) and ($|W^{n,[v]} - W^{n,[v-1]}| \geqslant \varepsilon$).
- Set $Y^{n,[v]}, W^{n,[v]}$ as the solution Y^n, W^n for the current step.

However, as will be described in Section 5, this fixed-point iterative process only converges linearly in the parameter values, and can be time-consuming if high accuracy is desired.

Incidentally, we point out that the above fixed-point iteration with no iterations (i.e., $v_{\max} = 0$) can be viewed as a fully explicit treatment of the parameter W. This method performs poorly, unless the time-step is very small, since it is essentially an explicit method for time-evolution.

3.2.2. Newton's method for the algebraic constraint

We now outline an approach based on Newton's method for the algebraic constraint (25) on the auxiliary parameters. As will be seen below, this approach yields quadratic convergence for the parameter values, and is therefore more efficient than the fixed-point iteration.

Since the correct choice of parameters W^n are the root of Eq. (25), at first glance, it would appear to be natural to compute $W^{n,[1]}$ directly by using the function values of $Y^{n,[1]}$ to iteratively satisfy the algebraic constraint (as is precisely the case in fixed point iteration). However, in order to achieve a quadratic convergence for parameters, we use, instead, the following procedure: First, under the Newton iterate (24), the algebraic constraint (25) is replaced by the following associated constraint:

$$G[W^n, L^{-1}(W^n, Y^{n,[0]}) \cdot R(W^n, Y^{n,[0]})] = 0. \tag{26}$$

Then, a second Newton's iterate is used to find the root W^n of Eq. (26) with its corresponding Newton step as

$$[\partial_W G(W^{n,[0]}, Y^{n,[1]}) + \partial_Y G(W^{n,[0]}, Y^{n,[1]}) \partial_W Y^{n,[1]}] \cdot [W^{n,[1]} - W^{n,[0]}] = -G(W^{n,[0]}, Y^{n,[1]}). \tag{27}$$

Suppose we have solved Eq. (27) to obtain $W^{n,[1]}$ (details see below). Then, the iterative process for our problem proceeds by using this better approximation $W^{n,[1]}$ to the parameter set to obtain a new solution $\overline{Y}^{n,[1]}$ by solving the BVP (23) to obtain

$$\overline{Y}^{n,[1]} = L^{-1}(W^{n,[1]}, Y^{n,[1]}) \cdot R(W^{n,[1]}, Y^{n,[1]}). \tag{28}$$

In summary, a single iteration in our method consists of two Newton's iterates for the linear BVP similar to Eq. (23) with an interlacing Newton's iterate for the algebraic constraint, e.g., Eq. (27). Then, using $(W^{n,[1]}, \overline{Y}^{n,[1]})$, we start the second iterate in our method to find the corresponding $Y^{n,[2]}$, then $W^{n,[2]}, \overline{Y}^{n,[2]}$. For example, $Y^{n,[2]}$ is determined via the BVP similar to Eq. (23), i.e.,

$$L(W^{n,[1]}, \overline{Y}^{n,[1]}) \cdot Y^{n,[2]} = R(W^{n,[1]}, \overline{Y}^{n,[1]})$$

or

$$[I - \Delta_n H(W^{n,[1]}, \overline{Y}^{n,[1]}, \partial_v \overline{Y}^{n,[1]}) - \Delta_n K(W^{n,[1]}, \overline{Y}^{n,[1]}, \partial_v \overline{Y}^{n,[1]})\partial_v] \cdot [Y^{n,[2]} - \overline{Y}^{n,[1]}]$$
$$= \Delta_n F(W^{n,[1]}, \overline{Y}^{n,[1]}, \partial_v \overline{Y}^{n,[1]}) + Y^{n-1} - \overline{Y}^{n,[1]}$$

with boundary conditions

$$A(W^{n,[1]}, t_n) Y_r^{n,[2]} + C(W^{n,[1]}, t_n) Y_\theta^{n,[2]} = 0$$

(compare with Eqs. (20) and (21)). We continue the iteration process (one step of Newton's method for obtaining Y (essentially Eq. (24)), one step for obtaining W (essentially Eq. (27)), and another step of Newton's method for $\overline{Y}$ (again, essentially Eq. (24)) until the parameter constraint (Eq. (25)) is below a specified tolerance (usually single digit precision 10^{-7}). As will be demonstrated in Section 5, this process converges much faster than fixed point iteration, and usually obtains consistent parameter values after only a few iterations.

We emphasize that the above associated constraint (26) with its corresponding Newton iterate (27) is a crucial construction for achieving quadratic convergence for parameters, as will be discussed below.

Evaluation of $W^{n,[1]}$: To complete, we now turn to discussion of the details of how to solve Eq. (27) to obtain $W^{n,[1]}$. Note that Eq. (27) can be expanded as

$$\begin{bmatrix} 1 + a_r \partial_m \rho_r^{n,[1]} + b_r \partial_m \mathscr{X}_r^{n,[1]} & a_r \partial_{\mu_r} \rho_r^{n,[1]} + b_r \partial_{\mu_r} \mathscr{X}_r^{n,[1]} & a_r \partial_{\mu_\theta} \rho_r^{n,[1]} + b_r \partial_{\mu_\theta} \mathscr{X}_r^{n,[1]} \\ -\partial_m \left(\frac{\mathscr{X}_r^{n,[1]}}{\rho_r^{n,[1]}} \right) & 1 - \partial_{\mu_r} \left(\frac{\mathscr{X}_r^{n,[1]}}{\rho_r^{n,[1]}} \right) & -\partial_{\mu_\theta} \left(\frac{\mathscr{X}_r^{n,[1]}}{\rho_r^{n,[1]}} \right) \\ -\partial_m \left(\frac{\mathscr{X}_\theta^{n,[1]}}{\rho_\theta^{n,[1]}} \right) & -\partial_{\mu_r} \left(\frac{\mathscr{X}_\theta^{n,[1]}}{\rho_\theta^{n,[1]}} \right) & 1 - \partial_{\mu_\theta} \left(\frac{\mathscr{X}_\theta^{n,[1]}}{\rho_\theta^{n,[1]}} \right) \end{bmatrix} \cdot \begin{bmatrix} m^{n,[1]} - m^{n,[0]} \\ \mu_r^{n,[1]} - \mu_r^{n,[0]} \\ \mu_\theta^{n,[1]} - \mu_\theta^{n,[0]} \end{bmatrix}$$
$$= - \begin{bmatrix} m^{n,[0]} + a_r \rho_r^{n,[1]} + b_r \mathscr{X}_r^{n,[1]} \\ \mu_r^{n,[0]} - \frac{\mathscr{X}_r^{n,[1]}}{\rho_r^{n,[1]}} \\ \mu_\theta^{n,[0]} - \frac{\mathscr{X}_\theta^{n,[1]}}{\rho_\theta^{n,[1]}} \end{bmatrix}.$$

Note that the various components of $\partial_W Y^{n,[1]}$ (e.g., $\partial_m \rho^{n,[1]}, \ldots, \partial_{\mu_\theta} \mathscr{X}^{n,[1]}$) can be computed by simply differentiating Eq. (24) and using the identity

$$\partial_W [L^{-1}] = -L^{-1} [\partial_W L] L^{-1}. \tag{29}$$

For example, the components of $\partial_m Y^{n,[1]}$ can be computed as follows. First, using Eqs. (24) and (29), we obtain

$$\partial_m Y^{n,[1]} = -L^{-1} [\partial_m L] L^{-1} R + L^{-1} \partial_m R, \tag{30}$$

where L and R are all evaluated at $(W^{n,[0]}, Y^{n,[0]})$. Using Eq. (24), the right-hand side of Eq. (30) (denoted by $\mathscr{A}$) can be written as

$$\mathscr{A} = L^{-1} [[-\partial_m L] L^{-1} R + \partial_m R] = L^{-1} [[-\partial_m L] Y^{n,[1]} + \partial_m R],$$

which is the solution to the linear BVP

$$L_1[\mathscr{A}] = \Delta_n (\partial_m H + \partial_m K \partial_v) Y^{n,[1]} + \Delta_n (\partial_m F - \partial_m H Y^{n,[0]} - \partial_m K \partial_v Y^{n,[0]}) \tag{31}$$

(where $\partial_m H$ and $\partial_m K$ are both evaluated at $(W^{n,[0]}, Y^{n,[0]})$) with boundary conditions

$$A(W^{n,[0]}, t_n) \mathscr{A}_r + C(W^{n,[0]}, t_n) \mathscr{A}_\theta = - \begin{bmatrix} 0 & 0 \\ b_r \partial_m \hat{g} & 0 \end{bmatrix} Y_r^{n,[1]} + \begin{bmatrix} 0 & 0 \\ b_\theta \partial_m \hat{g} & 0 \end{bmatrix} Y_\theta^{n,[1]}. \tag{32}$$

Once we solve this BVP (Eqs. (31) and (32)), we obtain $\partial_m Y^{n,[1]} = \mathscr{A}$. The other components of $\partial_W Y^{n,[1]}$, namely, $\partial_{\mu_r} Y^{n,[1]}$, and $\partial_{\mu_\theta} Y^{n,[1]}$, are computed similarly, but are slightly easier to evaluate since the operators $\partial_{\mu_r} L_1$, $\partial_{\mu_\theta} L_1$, $\partial_{\mu_r} R$ and $\partial_{\mu_\theta} R$ are all zero, and the operators $\partial_{\mu_r} L_2$ and $\partial_{\mu_\theta} L_2$ only act on the boundary conditions. Thus we only have to solve one BVP in each case.

After the components of $\partial_W Y^{n,[1]}$ are all computed, the new parameter $W^{n,[1]}$ can be obtained by solving Eq. (27). Each step of Newton's iteration for the parameters $W^{n,[\nu]}$ requires the Jacobian $\partial_W Y^{n,[\nu]}$. Evaluating this Jacobian requires the solution of three BVPs. This is somewhat costly, and the standard way to minimize this cost is to simply keep using the old Jacobian $\partial_W Y^{n,[0]}$ until the iterations for W stop converging. Using this approach we only have to update (refresh) the Jacobian once in a while (about once per time-step).

This completes the description of alternate Newton's methods for solving the BVP along with the algebraic constraint at each time-step.

4. Spectral deferred correction framework

In this section, we address the accuracy issue in numerically solving the kinetic moment equations for neuronal network dynamics. Recall that we are trying to numerically approximate a solution to the PDE

$$\partial_t Y(v,t) = F(W(t), Y(v,t), \partial_v Y(v,t)),$$

subject to boundary conditions

$$A(W(t),t)Y(\epsilon_r,t) + C(W(t),t)Y(\epsilon_\theta,t) = 0,$$

and constraints

$$G(W(t), Y(t)) = 0.$$

Given initial conditions to this PDE at some initial time T_0, the task is to approximate the exact solution $Y(v,t)$ and the exact parameter set $W(t)$. The basic implicit Euler method (Eq. (18)) only produces an approximate solution Y^n of the original PDE (16) with boundary conditions (17), which is first order accurate in the time-step Δt (regardless of how the constraint is solved, e.g., using Newton's method or fixed point iteration). Now we discuss how to improve the accuracy of this numerical solution by applying the implicit Euler method within a spectral deferred correction framework [10–14].

We note that essentially the same spectral deferred correction scheme used in [13,14] can be applied here with the following procedures:

1. First, a numerical method, such as those in Section 3, is selected to numerically solve the PDE (16) and (17), whose exact solution is denoted by $Y(v,t)$.
2. Once an approximate solution $\widetilde{Y}(v,t)$ is obtained, we construct a PDE for the error $e(v,t) = Y(v,t) - \widetilde{Y}(v,t)$.
3. Using a numerical method (e.g. again using the method of Section 3), we solve this PDE to obtain an approximation $\tilde{e}(x,t)$ to the error.
4. The original approximation is updated by $\widetilde{Y}(v,t) := \widetilde{Y}(v,t) + \tilde{e}(v,t)$.
5. Return to step 2 and repeat as necessary.

We now explain this procedure for solving PDE (16) and (17) in more detail (closely following [13,14]). Assuming that the exact solution (i.e., initial conditions) at time T_0 is known, one can use the methods of Section 3 to obtain a numerical solution $Y^n(v), W^n$ at a sequence of q time points $\{t_1, \ldots, t_n, \ldots, t_q\}$ within the time interval $[T_0, T_0 + \Delta T]$. These time points are chosen to be Chebyshev nodes within the time interval $[T_0, T_0 + \Delta T]$, which will allow us to construct an accurate qth order polynomial interpolant $\widetilde{Y}(v,t), \widetilde{W}(t)$ through the points $Y^n(v), W^n$ [27]. Data about the approximate solution at these Chebyshev nodes will also allow us to accurately compute time-wise integrals over the time interval $[T_0, T_0 + \Delta T]$ [27].

Using the interpolants $\widetilde{Y}(v,t), \widetilde{W}(t)$, the error in the solution is $Z(v,t) = Y(v,t) - \widetilde{Y}(v,t)$, and the error in the parameters is $X(t) = W(t) - \widetilde{W}(t)$, where $Y(v,t)$ and $W(t)$ are the exact solution and the exact

parameter set for the PDE (16) and (17). For $Z(v,t)$ and $X(t)$, a PDE can be formally constructed as follows:

$$\begin{aligned}\partial_t Z(v,t) &= \partial_t Y(v,t) - \partial_t \widetilde{Y}(v,t) = F[W(t), Y(v,t), \partial_v Y(v,t)] - \partial_t \widetilde{Y}(v,t)\\ &= F[\widetilde{W}(t) + X(t), \widetilde{Y}(v,t) + Z(v,t), \partial_v \widetilde{Y}(v,t) + \partial_v Z(v,t)] - \partial_t \widetilde{Y}(v,t)\\ &\approx \partial_W F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) \cdot X(t) + \partial_Y F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) \cdot Z(v,t)\\ &\quad + \partial_{\partial_v Y} F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) \cdot \partial_v Z(v,t) + F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) - \partial_t \widetilde{Y}(v,t)\end{aligned} \tag{33}$$

with boundary conditions

$$\begin{aligned}&A(\widetilde{W}(t) + X(t), t) Z(\epsilon_r, t) + C(\widetilde{W}(t) + X(t), t) Z(\epsilon_\theta, t)\\ &\quad = -A(\widetilde{W}(t) + X(t), t) \widetilde{Y}(\epsilon_r, t) - C(\widetilde{W}(t) + X(t), t) \widetilde{Y}(\epsilon_\theta, t).\end{aligned} \tag{34}$$

In order to compute an approximate $Z^n(v), X^n$ to the error (in both the solution and the parameters) at the time points $\{t_1, \ldots, t_n, \ldots, t_q\}$ within the time interval $[T_0, T_0 + \Delta T]$, we apply a version of implicit Euler to Eqs. (33) and (34):

$$\begin{aligned}Z^n(v) &= Z^{n-1}(v) + \Delta_n[(\partial_W F)^n \cdot X^n + (\partial_Y F)^n Z^n(v) + (\partial_{\partial_v Y} F)^n \partial_v Z^n(v)]\\ &\quad + \int_{t_{n-1}}^{t_n} [F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) - \partial_t \widetilde{Y}(v,t)] \mathrm{d}t\end{aligned} \tag{35}$$

with boundary conditions

$$A(W^n + X^n, t_n) Z_r^n + C(W^n + X^n, t_n) Z_\theta^n = -A(W^n + X^n, t_n) Y_r^n - C(W^n + X^n, t_n) Y_\theta^n, \tag{36}$$

where we have defined

$$\begin{aligned}&\Delta_n = t_n - t_{n-1},\\ &F^n = F(W^n, Y^n(v), \partial_v Y^n(v)),\\ &(\partial_W F)^n = \partial_W F(W^n, Y^n(v), \partial_v Y^n(v)),\\ &(\partial_Y F)^n = \partial_Y F(W^n, Y^n(v), \partial_v Y^n(v)),\\ &(\partial_{\partial_v Y} F)^n = \partial_{\partial_v Y} F(W^n, Y^n(v), \partial_v Y^n(v)).\end{aligned}$$

Note that under the appropriate Gaussian quadrature for the time nodes $\{t_1, \ldots, t_n, \ldots, t_q\}$ [27], in general,

$$\int_{T_0}^{t_n} f(t) \mathrm{d}t \approx \sum_{i=1}^{m} w_{n,m} f(t_m),$$

where $w_{n,m}$ are the corresponding weights in the quadrature. Therefore, the integral in the second line of Eq. (35) can be accurately approximated (to qth order accuracy) by using the previous solution values $Y^n(v)$, i.e.,

$$\int_{t_{n-1}}^{t_n} [F(\widetilde{W}(t), \widetilde{Y}(v,t), \partial_v \widetilde{Y}(v,t)) - \partial_t \widetilde{Y}(v,t)] \mathrm{d}t \approx \sum_{i=1}^{m} (w_{n,m} - w_{n-1,m}) F^m - (\widetilde{Y}^n - \widetilde{Y}^{n-1}), \tag{37}$$

which is completely known, i.e., does not depend on the error Z^n, X^n.

Thus, Eqs. (35) and (36) constitute a boundary value problem with a form similar to Eqs. (18) and (19), and which can be solved in a very similar manner (using either fixed point iteration or, preferably, Newton's method for solving the algebraic constraint as above). Once we obtain an approximation to the errors Z^n, X^n at the Chebyshev points $\{t_1, \ldots, t_n, \ldots, t_q\}$, we can form $Y^n + Z^n$ and $W^n + X^n$ to update our approximate solution. This process can be repeated as necessary, but the improvement in accuracy is limited by the ability to compute the integral Eq. (37) accurately.

5. Numerical examples

We illustrate the efficacy of the kinetic theory (Eqs. (13) and (14)), and the practicality of our numerical methods with the following numerical examples.

5.1. Rate of convergence

First, we illustrate the efficiency of the Newton's method as described in Section 3 for finding parameters (e.g., firing rate) consistent with the numerical solution. We evolve the kinetic theory equations (Eqs. (13) and (14)) with parameters

$$\begin{aligned} &N = 100, \quad \tau_v = 20 \text{ ms}, \quad \tau_g = 0.1 \text{ ms}, \\ &f = 0.5, \quad v = 0.5 \text{ spikes/ms}, \quad S = 0.125 \end{aligned} \tag{38}$$

and initial conditions $\rho_0 = 1/(\epsilon_\theta - \epsilon_r), \mathscr{X}_0 = 0$, for 16 ms with a variety of time-steps ($\Delta T = 0.0625$–1 ms) and measure the rates of convergence of the parameter values for each step. (In order to make Newton's method appreciably faster than fixed point iteration, we recompute the Jacobian of the parameter constraint once every four time-steps (as opposed to every time-step).) Fig. 1A compares the correction size of Newton's iterates to the correction size of fixed point iterates (averaged over all time-steps). As is expected, the Newton's scheme converges much faster than fixed point iteration.

The reason that our Newton's methods can achieve a quadratic convergence of error rate can be intuited from a geometric construction as illustrated in Fig. 2. In Fig. 2, the two surfaces intersect along a curve passing through the point x in the plane $W-Y$. The surface $Z = B(W, Y)$ can be viewed as the surface associated with the problem of finding roots of $B(W, Y) = 0$, which corresponds to the BVP (18) with the fixed auxiliary parameter W, while the surface $Z = G(W, Y)$ can be viewed as the surface associated with the problem of finding roots of $G(W, Y) = 0$, which corresponds to the algebraic constraint (25). Therefore, the problem of finding a consistent set of auxiliary parameters W which solve the BVP (18) with the boundary condition linearized by the auxiliary parameterization, as discussed in Section 3.1, now becomes the problem of finding the common root (W, Y) that solves both $B(W, Y) = 0$ and $G(W, Y) = 0$ simultaneously, i.e., finding the coordinates of the point x, which is the intersection of the curve $u'u$ (i.e., $B(W, Y) = 0$) and the curve $p'p$ (i.e., $G(W, Y) = 0$).

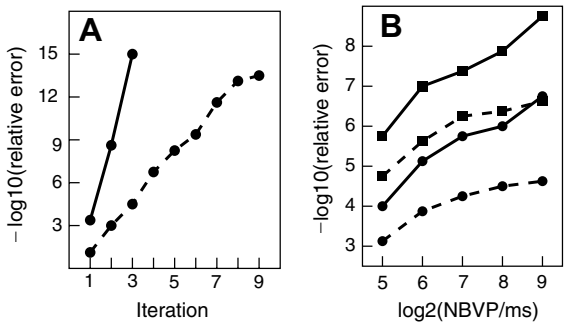

Fig. 1. Error plots. (A): average relative L^∞ error in boundary data W as measured using a sample network (see text) over several time-steps. In all cases the m-component of W contributed the most to the error. Error is plotted as a function of iteration, with the dashed line indicating fixed-point iteration and the solid line indicating Newton's Method. Newton's method converges much faster than fixed-point iteration. (B): average relative L^∞ error in boundary data W (solid lines) and average relative L^2 error in probability $\rho(v)$ (dashed lines) obtained using the implicit Euler scheme of Section 3 (data indicated by circles) and compared against the errors obtained using the spectral deferred correction scheme of Section 4 (data indicated by squares). The methods are both tested on a specific sample network (see text) using a variety of time-steps. The errors are plotted as a function of the total number of boundary value problems that are solved during a millisecond of time-evolution, which is an indication of the total time taken by the numerical method. The spectral deferred correction method is more accurate than the simple implicit Euler scheme (by about one extra digit for the same amount of operations). Notice that the L^2 errors in $\rho(v)$ are consistently two digits larger than the L^∞ errors in the W. Again, in all cases the m-component of W contributed the most to the parameter error.

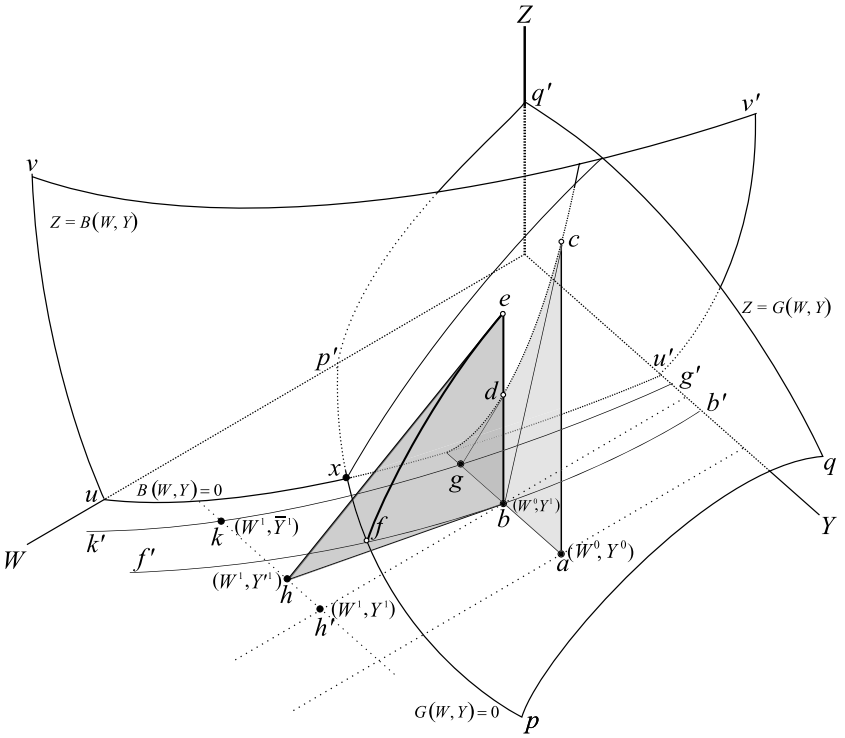

Fig. 2. A schematic illustrating the spirit of the Newton's scheme of Section 3. The curve uxu' corresponds to the nonlinear boundary value problem Eq. (18) (with linear boundary conditions given by Eq. (19). The curve pxp' corresponds to the algebraic constraint relating the solution Y to the boundary conditions W given by Eq. (25). The goal is to find the intersection point $x = \{W^{\text{perfect}}, Y^{\text{perfect}}\}$ of these two curves. Assuming that our best guess for x is the point $a = \{W^0, Y^0\}$, the scheme of Section 3 uses one step of Newton's iteration (Eq. (24)) to find the point $b = \{W^0, Y^1\}$, and then uses Newton's method for the algebraic constraint (Eq. (27)) to find the updated $W^1 \in h = \{W^1, Y'^1\}$, and finally corrects once more (using Eq. (24) to find the point $k = \{W^1, \overline{Y}^1\}$. We can approach the point x quadratically by repeating this entire procedure over and over again (see text for more details).

The first Newton's iterate (23) can be viewed as a Newton's step here for solving $B(W, Y) = 0$ for a fixed value of W: starting from the point $a = (W^{[0]}, Y^{[0]})$, find the Newton's iterate $b = (W^{[0]}, Y^{[1]})$ along the Y-direction only. This point b corresponds to the solution (24). The solution (24) for all possible values of $W^{[0]}$ with a given $Y^{[0]}$ yields $Y^{[1]}(W^{[0]})$ as a function of $W^{[0]}$. Geometrically, this corresponds to the locus (the curve $f'fbb'$) of the Newton's iterate, which is the set of the Newton iterates along the Y-direction for all possible values $W^{[0]}$ for a given $Y^{[0]}$ – clearly, the point b is on this locus. By construction, the curve $f'fbb'$ deviates from the true solution $B(W, Y) = 0$ in the Y-direction (i.e., with a fixed W) by an error of quadratic order in the sense of Newton's method. In this sense, we say that the point b is "quadratically" close to the curve $B(W, Y) = 0$ along the Y-direction. If this construction is repeated starting from the points on the line $h'b$ (i.e., varying W for the fixed $Y^{[1]}$), then, a new locus of the Newton's iterate is obtained, i.e., the curve $k'kgg'$ in Fig. 2. For example, the point g is the Newton's iterate obtained by starting from the initial point $b = (W^{[0]}, Y^{[1]})$ along the Y-direction, and the point k is the Newton's iterate obtained by starting from the initial point $h' = (W^{[1]}, Y^{[1]})$ along the Y-direction. (The meaning of $W^{[1]}$ will become clear momentarily, $W^{[1]}$ can be viewed, at the moment, as just another parameter W used in the Newton's step for a fixed $Y^{[1]}$.) The curves thus constructed, $f'fbb'$, or $k'kgg'$, approach the true solution $B(W, Y) = 0$ with a quadratic rate for a fixed W.

Next, we note that the curve $f'fbb'$ corresponds to the function $Y^{[1]} = Y^{[1]}(W^{[0]})$ and that the problem (26) corresponds to solving $G(W, Y^{[1]}(W)) = 0$, whose root is the point f, which is the intersection between the curve $p'xp$ (i.e., $G(W, Y) = 0$) and the curve $f'fbb'$. In addition, we note that the point f and the point x are "quadratically" close since the point f is "quadratically" close to the curve $u'xu$ along the Y-direction. The Newton's iterate (27) constructed for the problem (26) can now be viewed as the following geometric construction:

First, the locus $f'fbb'$ is vertically lifted to the surface $Z = G(W, Y)$ (For example, the portion fb of the locus is vertically lifted to the curve fe on the surface $Z = G(W, Y)$.) Then, a Newton's iterate, which corresponds to the Newton's iterate (27), is used to find the point $h = (W^{[1]}, Y'^{[1]})$, as an approximation to the point f. The point h is the intersection with the (W, Y)-plane of the tangent to the curve fe at the point e, which is vertically lifted from the point b. Therefore, the point h is quadratically close to the point f and the value of $W^{[1]}$ thus obtained differs from the W-coordinate of the point f by a quadratic error in the sense of Newton's method. Since the points f and x are "quadratically" close, the point h is quadratically close to the point x. Incidentally, by construction, the straight line connecting the point h and the point b is the tangent to the curve $f'fbb'$ at the point b since the line segment bh is simply the projection of the tangent eh to the (W, Y)-plane.

One could simply compute $Y'^{[1]}$ by evaluating all the derivatives at the point e along the curve fe, just as what we have carried out for the evaluation of $W^{[1]}$ by using all possible derivatives at the point e along the curve fe (see Section 3.2.2). Therefore, by iterating these two Newton's steps, i.e., the construction of $a \to b$ and $b \to h$, a sequence of the points of the h-kind can be obtained, approaching the true solution (i.e., the point x) with a quadratic rate. Then, the W-coordinates of these points possess a quadratic rate in approaching the true value, i.e., the W-coordinate of the point x. Clearly, the curve $Y^{[1]} = Y^{[1]}(W)$ $(f'fbb')$ plays an important role here, allowing us to construct or approximate the point f, which is quadratically close to x and simultaneously satisfies $G(W, Y^{[1]}[W]) = 0$.

In our method as described in Section 3, instead of constructing $Y'^{[1]}$ directly by using all possible derivatives, we actually used the point $k = (W^{[1]}, \overline{Y}^{[1]})$, which is the Newton's iterate starting from the point $h' = (W^{[1]}, Y^{[1]})$ along the Y-direction. The point $k = (W^{[1]}, \overline{Y}^{[1]})$ is still quadratically close to $B(W, Y) = 0$ along the Y-direction, but with $W = W^{[1]}$, which is quadratically close to the true value, i.e., the W-coordinate of the point x. Therefore, the quadratic rate for the convergence of the auxiliary parameter W in our method (i.e., the two Newton's steps (23) interlaced with the Newton's step (27) for solving the constraint (25)) can be intuited from the corresponding geometric construction ($a \to b$, $b \to h$, $h' \to k$ and repeat) as summarized in Fig. 2.

5.2. Spectral deferred correction

We now demonstrate the practicability of the spectral deferred correction framework. We evolve the same system (with parameters given by Eq. (38)) for 16 ms using both the basic method (as described in Section 3) as well as the spectral deferred correction (SDC) method (as described in Section 4) with a variety of time-steps ($\Delta T = 0.0625$–1 ms) to obtain solutions $\{Y^{\Delta T}_{\text{Euler}}, W^{\Delta T}_{\text{Euler}}\}$ and $\{Y^{\Delta T}_{\text{sdc}}, W^{\Delta T}_{\text{sdc}}\}$, respectively. The exact solution is estimated by evolving the system with a sufficiently small time time-step ($\Delta T = 2^{-10}$ ms $\approx 1 \times 10^{-3}$ ms), and this estimate is used to measure the errors,

$$Z^{\Delta T}_{\text{Euler}} = \|\rho^{\text{exact}} - \rho^{\Delta T}_{\text{Euler}}\|_2,\ X^{\Delta T}_{\text{Euler}} = \|W^{\text{exact}} - W^{\Delta T}_{\text{Euler}}\|_\infty,$$
$$Z^{\Delta T}_{\text{sdc}} = \|\rho^{\text{exact}} - \rho^{\Delta T}_{\text{sdc}}\|_2,\ X^{\Delta T}_{\text{sdc}} = \|W^{\text{exact}} - W^{\Delta T}_{\text{sdc}}\|_\infty,$$

respectively. In Fig. 1B, we plot the accuracy of the numerical solution as a function of the amount of time taken to compute that solution. Here, time is measured in terms of the number of boundary value problems solved, since each application of the BVP solver takes approximately the same amount of time. It is clear that the spectral deferred correction framework allows for higher accuracy at a lower computational cost. We observe that, in our simulations, the greatest gain in efficiency is achieved with one or two passes of deferred correction. More passes of deferred correction do increase the overall accuracy of the solution, but require detailed spatial resolution of the residual near the boundary, and are not significantly faster than taking a smaller time-step with fewer overall correction steps.

5.3. Predictions of kinetic theory

Finally, we demonstrate the usefulness of the kinetic moment equations for neuronal network dynamics as an approximation to an ensemble of integrate-and-fire networks. We evolve an ensemble of 10^4 integrate-and-fire neuronal networks each under independent Poisson input drive with the same rate, and compare the firing

rate statistics of the integrate-and-fire network ensemble with those calculated using the kinetic theory. Each integrate-and-fire neuronal network within the ensemble obeys Eq. (1), with parameters given by Eq. (38), except for the input rate ν, which is given by

$$\nu = \nu_0 \exp\left[\frac{1}{4}\sin\left(\frac{2\pi t}{T} + \left(\frac{2\pi t}{T}\right)^2\right)\right], \tag{39}$$

where $\nu_0 = 0.5$ spikes/ms and $T = 100$ ms. The integrate-and-fire networks are evolved over 100 ms using an efficient integrate-and-fire neuronal solver [28]. The ensemble-averaged firing rate per neuron of the integrate-and-fire networks is compared with the firing rate for the corresponding kinetic theory (Eqs. (13) and (14)), which is solved using the methods of Sections 3 and 4. As shown in Fig. 3, the comparison demonstrates that the kinetic theory is dynamically accurate in capturing the firing rate of the integrate-and-fire neuronal networks under time varying drive. Note that the fast oscillations in the input (39) are approaching the timescales of fast input to a cortical network. For reference, we compare these two firing rate curves with the firing rate obtained using the mean rate model [7,8]. The firing rate of the mean rate model is obtained by solving the following algebraic equation:

$$m = \begin{cases} \dfrac{1+\bar{g}}{\tau_v \log\left|\dfrac{\bar{g}(\epsilon_r-\epsilon_E)}{(\epsilon_\theta-\epsilon_r)+\bar{g}(\epsilon_\theta-\epsilon_E)}\right|}, & \text{if } \bar{g} > \dfrac{\epsilon_\theta-\epsilon_r}{\epsilon_E-\epsilon_\theta} \\ 0, \quad \text{otherwise} \end{cases} \tag{40}$$

with

$$\bar{g} = f\nu + Sm.$$

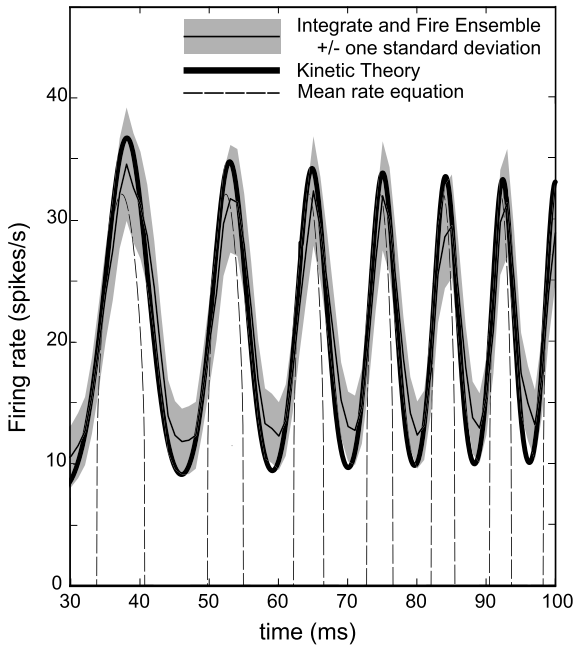

Fig. 3. Dynamical accuracy of kinetic theory – the firing rate $m(t)$. (Thin solid line: simulation result averaged over an ensemble of 10^4 identically structured networks. The firing rate is measured using a bin size of 1 ms. The upper and lower boundaries of the gray area mark the values one-standard deviation away from the mean, measured from the ensemble of the 10^4 networks. Thick solid line: kinetic theory (11). Dashed line: mean-driven limit (40).) The parameters for the network (1): $N = 100$ purely excitatory I& F neurons driven by a time-varying input rate $\nu(t) = \nu_0\exp[0.25\sin(2\pi t/T + (2\pi t/T)^2)]$, where $\nu_0 = 0.5$ spikes/ms, $T = 100$ ms. $f = 0.5$ ms, $\tau_v = 20$ ms and $\tau_g = 0.1$ ms, $S = 0.125$.

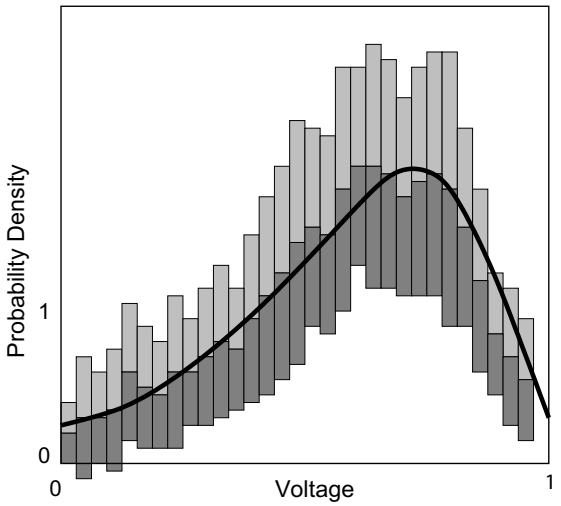

Fig. 4. Dynamical accuracy of kinetic theory – instantaneous probability density function of the membrane potential. Plotted is the pdf at time $t = 29.375$ ms. (The upper and lower boundaries of the gray area mark the values one-standard deviation away from the mean, measured from the ensemble of the 10^4 networks. Thick solid line: kinetic theory (11).) See Fig. 3 for network parameters.

Note that the rate equation (40) is derived under the mean-driven approximation [6–8]. As is demonstrated in Fig. 3, the kinetic theory clearly provides a much better approximation of the actual ensemble firing rate than the mean rate theory does. The mean-driven approximation fails dramatically for small firing rates (<15 spikes/s). As shown in Fig. 4, it can be seen that the kinetic theory also accurately predicts instantaneous voltage distributions of individual neurons within the ensemble. In general, the kinetic theory well captures the corresponding integrate-and-fire ensemble for a wide range of parameter values. However, it can become less accurate if the small jump approximation becomes invalid or the firing statistics from the other neurons in the network is far from Poisson, which can arise from the limit of very large S and very small N.

It should be noted that accurately evolving the 10^4 integrate-and-fire ensembles took about 500 times longer than accurately evolving the kinetic theory equations for the sample network given by Eq. (38). To obtain an extra significant digit in the integrate-and-fire ensemble firing rate (and thus substantially differentiate the ensemble response from the response calculated using kinetic theory), we would need approximately 10^6 individual ensembles, and the computation would take about 50,000 times as long as the analogous kinetic theory computation. Clearly, the computational advantage of the kinetic theory equations (for computing statistical information) becomes even more pronounced as the number of neurons N within each ensemble increases.

6. Conclusion

The algorithms we describe in Sections 3 and 4 can successfully evolve the kinetic theory equations within a variety of regimes, and are useful for efficiently determining the statistical properties of ensembles of integrate-and-fire neuronal networks. Moreover, the essential elements in our algorithm are rather general, therefore, our algorithms can be extended to other types of PDEs with nonlinear boundary conditions.

Acknowledgments

The work was performed under the support of NSF Grant DMS-0506396 for A.V.R., D.C. and L.T., NSF Grant DMS-0211655 for A.V.R. and D.C, and a Sloan Fellowship for D.C.

References

[1] D. Cai, L. Tao, M. Shelley, D.W. Mclaughlin, An effective representation of fluctuation-driven neuronal networks with application to simple & complex cells in visual cortex, Proc. Natl. Acad. Sci. (USA) 101 (2004) 7757–7762.

[2] D. Cai, L. Tao, A.V. Rangan, D.W. McLaughlin, Kinetic theory for neuronal network dynamics, Commun. Math. Sci. 4 (2006) 97–127.
[3] A.V. Rangan, D. Cai, Maximum-entropy closures for kinetic theories of neuronal network dynamics, Phys. Rev. Lett. 96 (2006) 178101.
[4] H. Grad, On the kinetic theory of rarefied gases, Commun. Pure Appl. Math. 2 (1949) 331–407.
[5] C.D. Levermore, Moment closure hierarchies for kinetic theories, J. Stat. Phys. 83 (1996) 1021–1065.
[6] H. Wilson, J. Cowan, A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue, Kybernetik 13 (1973) 55–80.
[7] A. Treves, Mean field analysis of neuronal spike dynamics, Network 4 (1993) 259–284.
[8] M. Shelley, D. McLaughlin, Coarse-grained reduction and analysis of a network model of cortical response. I. Drifting grating stimuli, J. Comp. Neurosci. 12 (2002) 97–122.
[9] D. Cai, L. Tao, D.W. McLaughlin, An embedded network approach for scale-up of fluctuation-driven systems with preservation of spike information, Proc. Natl. Acad. Sci. (USA) 101 (2004) 14288–14293.
[10] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, Bit 40 (2000) 241–266.
[11] A. Bourlioux, A. Layton, M. Minion, Multi-implicit spectral deferred correction methods for problems of reactive flow, J. Comp. Phys. 189 (2003) 351–376.
[12] A. Layton, M. Minion, Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics, J. Comp. Phys. 194 (2) (2004) 697–714.
[13] A.V. Rangan, Adaptive Solvers for Partial-differential and Differential–Algebraic Equations, PhD thesis, University of California, Berkeley, 2003.
[14] A.V. Rangan, Spectral deferred correction methods for partial differential equations.
[15] P. Starr, V. Rokhlin, On the numerical solution of two-point boundary value problems ii, Appl. Math. 79 (1988) 271–289.
[16] J.Y. Lee, L. Greengard, A fast adaptive numerical method for stiff two-point boundary value problems, SIAM J. Sci. Comput. 18 (1997).
[17] C. Koch, Biophysics of Computation, Oxford University Press, Oxford, 1999.
[18] D. McLaughlin, R. Shapley, M. Shelley, J. Wielaard, A neuronal network model of macaque primary visual cortex (V1): Orientation selectivity and dynamics in the input layer 4Cα, Proc. Natl. Acad. Sci. USA 97 (2000) 8087–8092.
[19] E. Cinlar, Superposition of point processes, in: P.A.W. Lewis (Ed.), Stochastic Point Processes: Statistical Analysis, Theory, and Applications, Wiley, New York, NY, 1972, pp. 549–606.
[20] G.Q. Chen, C.D. Levermore, T.P. Liu, Hyperbolic conservation laws with stiff relaxation terms and entropy, Commun. Pure Appl. Math. 47 (1994) 787–830.
[21] S. Jin, Z.P. Xin, The relaxation schemes for systems of conservation laws in arbitrary space dimensions, Commun. Pure Appl. Math. 48 (1995) 235–276.
[22] R. Natalini, Convergence to equilibrium for the relaxation approximations of conservation laws, Commun. Pure Appl. Math. 49 (1996) 795–823.
[23] S. Jin, Runge–Kutta methods for hyperbolic conservation laws with stiff relaxation terms, J. Comp. Phys. 122 (1995) 51–67.
[24] R. Caflisch, S. Jin, G. Russo, Uniformly accurate schemes for hyperbolic systems with relaxations, SIAM J. Num. Anal. 34 (1998) 246–281.
[25] S. Jin, L. Pareschi, G. Toscani, Diffusive relaxation schemes for discrete-velocity kinetic equations, SIAM J. Num. Anal. 35 (1998) 2405–2439.
[26] L. Pareschi, G. Russo, Implicit-explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation, J. Sci. Comp. 25 (2005) 129–155.
[27] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, New York, 1998.
[28] A.V. Rangan, D. Cai, Fast numerical methods for simulating large-scale integrate-and-fire neuronal networks, J. Comp. Neurosci., in press.