LETTER Communicated by Bruce Knight

A Simple and Stable Numerical Solution for the Population Density Equation

M. de Kamps
kamps@in.tum.de
Section Cognitive Psychology, Faculty of Social Sciences, Leiden University, 2333 AK Leiden, The Netherlands

A population density description of large populations of neurons has generated considerable interest recently. The evolution in time of the population density is determined by a partial differential equation (PDE). Most of the algorithms proposed to solve this PDE have used finite difference schemes. Here, I use the method of characteristics to reduce the PDE to a set of ordinary differential equations, which are easy to solve. The method is applied to leaky-integrate-and-fire neurons and produces an algorithm that is efficient and yields a stable and manifestly nonnegative density. Contrary to algorithms based directly on finite difference schemes, this algorithm is insensitive to large density gradients, which may occur during evolution of the density.

1 Introduction

In the past decade, knowledge on brain function and organization has increased dramatically. This is partly due to the application of established techniques like in vivo electrode recoding of neural activity and partly due to the emergence of noninvasive brain imaging techniques like PET and fMRI. In particular, with the latter technique, the identification of large-scale cortical networks becomes possible. Moreover, with the advent of a new generation of 3T machines and the development of new analysis techniques, the time resolution appears to be much better than previously believed possible. Events that take place at a subsecond timescale can be located with a spatial precision in the millimeter range.

The advance of experimental techniques calls for a similar increase in sophistication of modeling techniques. Thus far, the modeling of higher cognitive functions has relied a great deal on artificial neural networks (ANNs). ANNs, however, are unsuitable for modeling transient phenomena and also their biological plausibility is not always clear (e.g., Maass, 1997; de Kamps & van der Velde, 2001). In order to model large-scale cortical networks in a way that allows confrontation with experimental data, techniques are required that reflect the neural substrate of such networks and are modest in their use of computational resources. In physics, similar conflicting

requirements have led to the emergence of statistical physics. The techniques of statistical physics can be applied to large-scale cortical networks, as has been demonstrated in, for instance, population density techniques or the spike-response model (e.g., Stein, 1965; Johannesma, 1966; Wilbur & Rinzel, 1982; Kuramoto, 1991; Abbott & van Vreeswijk, 1993; Gerstner & van Hemmen, 1992; Gerstner, 1995). These techniques, which have a long-standing history in neuroscience, consider the response of a large population of neurons rather than that of individual neurons; thus, networks of populations, which correspond to millions of neurons, may be simulated with reasonable efficiency.

A novel approach to the application of population density techniques to model large populations of sparsely connected neurons was introduced by Knight, Manin, and Sirovich (1996) and subsequently developed by many others.

In this approach, the dynamics of individual neurons, which are described by a state vector $\vec{v}$, determines the evolution of a density function $\rho(\vec{v}, t)$. As $\rho(\vec{v}, t)d\vec{v}$ gives the probability that a neuron in the population is in state $\vec{v}$, the density function characterizes the behavior of the whole population. In its simplest form, the state vector is one-dimensional and can be applied to leaky-integrate-and-fire (LIF) neurons (Omurtag, Knight, & Sirovich, 2000; Knight, 2000; Nykamp & Tranchina, 2000). An example of an application to a higher-dimensional system can be found in Casti et al. (2002), where a two-dimensional model of integrate-and-fire-or-burst (IFB) neurons is considered. Synaptic dynamics can be included as a multidimensional system, but this system can also be reduced to a one-dimensional system, which is computationally much more efficient (Haskell, Nykamp, & Tranchina, 2001).

The evolution equation in this approach is a partial differential integral (PDE) equation, which describes the evolution of $\rho(\vec{v}, t)$ under the influence of neuronal dynamics and a synaptic input. The synaptic input is assumed to be a spike train, which is distributed according to an inhomogeneous Poisson distribution with rate $\sigma(t)$. In a full version of the equation, the synaptic efficacies may be a stochastic variable, which is distributed according to a given probability function.

From the density $\rho(\vec{v}, t)$ the population's firing rate readily follows. Hence, a solution of the PDE allows the determination of a population's firing rate in response to a time-dependent input. An efficient solution of the PDE therefore allows for an efficient simulation of transient phenomena in cortical networks, as was demonstrated convincingly by Nykamp and Tranchina (2000) in a model of orientation tuning. As analytic solutions are available only for a limited number of cases, such as stationary input (Sirovich, Omurtag, & Knight, 2000) or a two-compartment model (Sirovich, 2003), efficient numerical solutions are essential in applications of this technique.

So far, most approaches to solve this PDE numerically have been based on finite difference schemes (Omurtag et al., 2000; Nykamp & Tranchina, 2000;

Casti et al., 2002). A problem with finite difference schemes is that large-density gradients may occur naturally during the evolution of the density over time (e.g., Omurtag et al., 2000; Nykamp & Tranchina, 2000; Casti et al., 2002). Indeed, as Nykamp and Tranchina (2000) and Casti et al. (2002) show, considerable effort is required to ensure stability of these schemes. Here I present an alternative approach, which is not based on finite difference schemes. It is based on the observation that in general, it is possible to carry out a coordinate transformation in v-space that transforms away the effects of neuronal dynamics. The coordinate transformation can be calculated by finding the characteristics of the PDE. For some important cases, like the LIF neuron but also the IFB neuron (Casti et al., 2002), an analytic expression for this solution can be found. In the resulting coordinate system, v'-space, the PDE reduces to a set of ODEs, which are in fact the Master equations of a Poisson jump process (Gardiner, 1997). The process in v'-space is universal in the sense that it has a form independent of the neuronal model in v-space. It describes the transfer of probability between points in v'-space. These master equations can be solved easily by numerical or analytical means.

These ideas are applied to the LIF neurons. It turns out that although the master equations are readily solvable, their time dependence in v'-space complicates the solution somewhat. Moreover, the boundary conditions must be handled with care. This is caused by the fact that the reset and threshold potentials, which are constant in v-space, are moving in v'-space. If the numerical problems related to these issues are handled carefully, a stable algorithm results, which compares well to finite difference schemes in terms of efficiency. The main virtue of this approach, however, seems to be its absolute insensitivity to gradient densities. The Herculean efforts by Casti et al. (2002) to produce a stable finite difference scheme, able to handle the large density gradients that occur during the evolution of an IFB population, show that this is an important issue.

Finally, I will discuss the relation between this approach and some of the many ideas that are discussed by Knight (2000), which are somewhat related.

2 The Population Density Equation

For completeness sake, in this section, I will briefly recapitulate the population density formalism. For a derivation of the equations and a discussion of the ideas behind them, I recommend Omurtag et al. (2000). In general a point neuron model is described by a set of ODEs,

$$\frac{d\vec{v}}{dt} = \vec{F}(\vec{v}), \tag{2.1}$$

which describes the evolution of the neuron's state variables in the absence of external input. Examples of point neuron models that are described by equation 2.1 are the LIF neuron, the IFB neuron (Casti et al., 2002), and the Hodgkin-Huxley model. In every point neuron model, there is at least a

membrane potential V. The state vector $\vec{v}$ will sometimes be represented by the n-tuple $(v_0, \ldots, v_{N-1})$. By convention, the first component of $\vec{v}$, v_0, will be v, the rescaled potential,

$$v = \frac{V - V_r}{V_\theta - V_r}, \tag{2.2}$$

with V the membrane potential in mV, V_r the reversal potential in mV, and V_θ the threshold potential in mV. v is restricted to the interval $(-\infty, 1)$. An important example of a neuron model that is described by equation 2.1 is the LIF neuron, described by

$$\frac{dv}{dt} = -\gamma v. \tag{2.3}$$

Here γ is the inverse of the membrane time constant in s^{-1}. If v is pushed across $v = 1$, the neuron spikes and is reset to a rescaled potential v_{reset}. The population density equation is given by

$$\frac{\partial \rho(\vec{v}, t)}{\partial t} = -\frac{\partial}{\partial \vec{v}} \cdot \vec{J}(\vec{v}, t). \tag{2.4}$$

Here $\rho(\vec{v}, t)$ is the population density. From equation 2.4 follows

$$\oint_{\partial D} \hat{n} \cdot \vec{J} = 0, \tag{2.5}$$

where D is the neuron's phase space and $\hat{n}$ is a boundary normal vector. Outside D, it is natural to choose

$$\rho(\vec{v}, t) = 0, \ v \notin D, \tag{2.6}$$

so that no inward flux due to neuronal dynamics is possible. Moreover, it is appropriate to choose the no-flux boundary conditions,

$$\hat{n} \cdot \vec{J} = 0, \tag{2.7}$$

everywhere at ∂D, except at $v = 1$.

$J(\vec{v}, t)$ has three components:

$$\vec{J} = \vec{J}^{stream} + \vec{J}^{input} + \vec{J}^{reset}. \tag{2.8}$$

Here $\vec{J}^{stream}$ is given by

$$\vec{J}^{stream}(v, t) = \rho(v, t)\vec{F}(\vec{v}). \tag{2.9}$$

$\vec{J}^{input}$ is defined by

$$\vec{J}^{input}(\vec{v}, t) = \sigma(t) \int_{-\infty}^{1} \vec{e}_0 \tau(v_0 - w_0) \rho(w_0, v_1, \ldots, v_{N-1})\, dw_0, \tag{2.10}$$

where $\vec{e}_0$ is the unit vector in the v direction and v_0, w_0 the zeroth component of vectors $\vec{v}$ and $\vec{w}$, respectively. Here, the case is considered where the population described by ρ receives external input from an input population with rate $\sigma(t)$. If the probability that an input spike causes a membrane depolarization of magnitude h is given by $p(h)$, then $\tau(v - w)$ is given by

$$\tau(v - w) = H(v - w) + \int_{v-w}^{\infty} p(h)\, dh. \tag{2.11}$$

Here $H(x)$ is the Heaviside step function:

$$H(x) = \begin{cases} x < 0 & H(x) = 0 \\ x \geq 0 & H(x) = 1 \end{cases}. \tag{2.12}$$

In this article, I will consider only a gaussian distribution of membrane depolarizations of magnitude $p(h)$

$$p(h) = \frac{1}{\sqrt{2\pi}\alpha} e^{\frac{-(h-\bar{h})^2}{2\alpha^2}} \tag{2.13}$$

or the nonstochastic limit $\alpha \to 0$, in which case

$$p(h) = \delta(h - \bar{h}), \tag{2.14}$$

where $\delta(x)$ is the Dirac distribution. I will refer to $(\bar{h}, \alpha)$ as the input parameters.

The flux at the threshold potential corresponds to the fraction of neurons that fire per unit time, that is, the firing rate of the population, which is given by

$$\begin{aligned} r(t) = &\int dv_1 \ldots dv_{N-1} \vec{e}_0 \cdot \vec{F}(1, v_1, \ldots, v_{N-1}) \rho(1, v_1, \ldots, v_{N-1}) \\ &+ \sigma(t) \int dv_1 \ldots dv_{N-1} \int_{-\infty}^{1} \vec{e}_0 \tau(1 - v_0) \rho(v_0, v_1, \ldots, v_{N-1})\, dv_0. \end{aligned} \tag{2.15}$$

Neurons that have spiked will be reset to membrane potential v_{0_r}. Therefore, the third current component $\vec{J}^{reset}$ is given by

$$\vec{J}^{reset}(\vec{v}, t) = \vec{e}_0 r(t) H(v - v_{0_r}). \tag{2.16}$$

The action of the divergence operator $\frac{\partial}{\partial \vec{v}}$ on the Heaviside function yields a delta function source at the return point v_{0_r}. This component ensures probability conservation. Probability density, which has left the system at the threshold potential, will be reintroduced at the reset potential.

3 Idea for an Algorithm

3.1 Transforming Away Neuronal Dynamics. The idea of transforming away neuronal dynamics is illustrated most clearly when $p(h) = \delta(h - \bar{h})$. Equation 2.4 then reduces to:

$$\frac{\partial \rho(\vec{v}, t)}{\partial t} + \frac{\partial}{\partial \vec{v}} \cdot (\vec{F}(\vec{v})\rho(\vec{v}, t)) = -\sigma(t)[\rho(\vec{v}) - \rho(\vec{v_{\bar{h}}})], \tag{3.1}$$

with

$$\vec{v} = (v, v_1, \ldots, v_{N-1}) \tag{3.2}$$

$$\vec{v_{\bar{h}}} = (v - \bar{h}, v_1, \ldots, v_{N-1}). \tag{3.3}$$

The characteristics $\vec{v}'(t)$ are a family of one-parameter curves in $\vec{v}$-space, determined by the ODE system (John, 1981),

$$\frac{d\vec{v}}{dt} = \vec{F}(\vec{v}), \tag{3.4}$$

and the initial value $\vec{v}(0)$. For initial conditions $\vec{v}(0)$, the solution of this system can be written formally:

$$\vec{v'}(t) = \vec{v}(t, \vec{v}(0)). \tag{3.5}$$

Using equation 3.1, one finds that on a characteristic line, the total time derivative of ρ is given by

$$\begin{aligned} \frac{d\rho(\vec{v'}, t)}{dt} &= \frac{\partial \rho(\vec{v'}, t)}{\partial t} + \frac{\partial \rho(\vec{v'}, t)}{\partial \vec{v'}} \cdot \frac{d\vec{v'}}{dt} \\ &= -\rho(\vec{v'}, t)\frac{\partial}{\partial \vec{v}} \cdot \vec{F}(\vec{v'}) - \sigma(t)\{\rho(\vec{v'}, t) - \rho(\vec{v'_{\bar{h}}}, t)\}, \end{aligned} \tag{3.6}$$

which after a further transformation,

$$\rho'(\vec{v'}, t) = e^{\int^t \frac{\partial \cdot \vec{F}(\vec{v'})}{\partial \vec{v'}} dt'} \rho(\vec{v'}, t), \tag{3.7}$$

reduces to

$$\frac{d\rho'(\vec{v'}, t)}{dt} = -\sigma(t)\{\rho'(\vec{v'}, t) - \rho'(\vec{v'_{\bar{h}}}, t)\}. \tag{3.8}$$

Hence, by the solution of the system 3.4 of ODEs, a coordinate transformation can be found that transforms equation 3.1 into an ODE:

$$\begin{cases} v & \rightarrow & \vec{v}'(t) & = & \vec{v}(t, \vec{v}(0)) \\ \rho(\vec{v}, t) & \rightarrow & \rho'(\vec{v}', t) & = & e^{\int^t \frac{\partial \cdot \vec{F}(\vec{v}')}{\partial \vec{v}'} dt'} \rho(\vec{v}', t). \end{cases} \tag{3.9}$$

An application of this formalism to the LIF neuron is straightforward: in the LIF model, $\vec{v}$ has a single component, v, $-\infty < v < 1$, as do all currents. For the neuronal dynamics, in this case restricted to leakage, we have

$$F(v) = -\gamma v, \tag{3.10}$$

which leads to

$$J^{stream}(v, t) = -\gamma v \rho(v, t). \tag{3.11}$$

Here, γ is the inverse of the membrane time constant in s^{-1}. The input current simplifies to

$$J^{input}(v, t) = \sigma(t) \int_{-\infty}^{1} \tau(v - w) \rho(w)\, dw, \tag{3.12}$$

where $\tau(v - w)$ is given by equation 2.11, while the reset current 2.16 now reads:

$$J^{reset}(v, t) = J^{input}(1, t) H(v - v_{reset}). \tag{3.13}$$

Using equations 2.6 and 3.11, one finds

$$J^{stream}(1, t) = 0, \tag{3.14}$$

or

$$\rho(1, t) = 0. \tag{3.15}$$

This leads to the following expression for the firing rate,

$$r(t) = \sigma(t) \int_{-\infty}^{1} \tau(1 - v) \rho(v)\, dv, \tag{3.16}$$

which for equation 2.13 reads

$$r(t) = \frac{1}{2} \sigma(t) \int_{-\infty}^{1} erfc\left(\frac{1 - v - \bar{h}}{\sqrt{2}\alpha} \right) \rho(v)\, dv. \tag{3.17}$$

And since I considered the limit $\lim \alpha \to 0$, that is, $p(h) = \delta(h - \bar{h})$, this becomes

$$r(t) = \sigma(t) \int_{1-\bar{h}}^{1} \rho(v)\, dv. \tag{3.18}$$

Equation 3.1 now reduces to

$$\frac{\partial \rho}{\partial t} - \gamma \frac{\partial}{\partial v}(\rho v) = -\sigma(t)[\rho(v, t) - \rho(v - \bar{h}, t)]. \tag{3.19}$$

It now remains to find the coordinate transformation to v'-space. The ODE that defines the characteristics is

$$dt = -\frac{dv}{\gamma v}. \tag{3.20}$$

Solving yields the following coordinate transformation:

$$\begin{aligned} v &\to v' = v(0)e^{-\gamma t} \\ \rho(v, t) &\to \rho'(v', t) = e^{-\gamma t}\rho(ve^{-\gamma t}, t). \end{aligned} \tag{3.21}$$

Indeed, equation 3.19 changes under this transformation equation into

$$\frac{d\rho'(v', t)}{dt} = \sigma(t)\{\rho'(v' - \bar{h}e^{\gamma t}) - \rho(v')\}, \tag{3.22}$$

which in the absence of input ($\sigma(t) = 0$) implies a constant density profile in v'-space and in this special case directly translates into the solution in v-space:

$$\rho(v, t) = e^{\gamma t}\rho(ve^{\gamma t}, 0). \tag{3.23}$$

3.2 The Master Equations. Equation 3.8 and its LIF version, 3.22, are equations which describe transfer of probability from one point in v' space to another. When $\gamma = 0$ and $\bar{h} = 1/2 - \epsilon$, with small but arbitrary ϵ ($\epsilon = 0$ would violate boundary condition 3.14), for instance, equation 3.1 and boundary condition 3.13 are completely equivalent to the following set of ODEs:

$$\begin{aligned} \frac{d\rho_0}{dt} &= \sigma(t)\,[\rho_2(t) - \rho_0(t)] \\ \frac{d\rho_1}{dt} &= \sigma(t)\,[\rho_0(t) - \rho_1(t)] \\ \frac{d\rho_2}{dt} &= \sigma(t)\,[\rho_1(t) - \rho_2(t)]\,. \end{aligned} \tag{3.24}$$

Here, $\rho_0 = \rho(0, t)$, $\rho_1 = \rho(1/2 - \epsilon, t)$, and $\rho_2 = \rho(1 - 2\epsilon, t)$, respectively. This equivalence follows from the simple fact that in the absence of neuronal dynamics, only points that correspond to a potential that is an integer multiple of $\bar{h}$ can have nonzero density, so that in general, a system of rank $[1/\bar{h}] + 1$ results, where $[x]$ is the entier of x. Note that all explicit references to the potential v have disappeared from the system above.

The emergence of these equations, and in general that of equation 3.8, is hardly surprising. For constant σ, they would be the master equations of a Poisson jump process (Gardiner, 1997), and for time-dependent $\sigma(t)$ they are the inhomogeneous version of these equations, subject to boundary condition 3.13. This process also can be interpreted as a "one-sided" random walk with a transition probability $W(n + 1 \mid n, t) = \sigma$; otherwise, $W(n \mid m, t) = 0$. Here n, an integer number, is the position reached at time t. In a coordinate system where neuronal dynamics is absent, the jump process is all that is left, so one would expect equations of the form 3.8.

In matrix notation, these equations can be given by

$$\frac{d\vec{\rho}}{dt} = \mathcal{A} \cdot \vec{\rho} \tag{3.25}$$

with

$$\mathcal{A} = \sigma(t) \begin{pmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}. \tag{3.26}$$

This equation system can be solved analytically or numerically by very simple means. If a numerical algorithm is used, the resulting density is manifestly nonnegative if a sufficiently small time step is chosen. For $p(h) = \delta(h - \bar{h})$, the set of transport equations can be found with the discretization procedure, described below, with $N = [1/\bar{h}] + 1$ points. As argued above, when $\gamma = 0$, the resulting set of ODEs is exactly equivalent to equation 3.19. An analytical treatment for the zero-leak case, in the limit $\alpha \to 0$, is given by Sirovich (2003). For finite α, the correspondence between the ODEs and the original PDE becomes approximate, but by taking N large enough, the resulting set of ODEs can approximate the original PDE to any required precision. Here, too, however, the set of ODEs is solvable by simple analytic or numerical means, which results in a density that is manifestly nonnegative. This is in stark contrast to the oscillations and negative densities that are observed when naive finite different schemes are used to solve equation 3.19 for the case where $\gamma > 0$ (Nykamp & Tranchina, 2000).

3.3 The Basic Idea. The coordinate transformation 3.21 effectively dissociates the two distinct physical processes that are considered in the population density equation: neuronal dynamics and potential jumps caused

by synaptic input. Each process is characterized by its own timescale: the neuronal dynamics is characterized by γ^{-1} for the LIF neuron, whereas the synaptic input is characterized by the inverse of the input rate, $\sigma^{-1}(t)$. The latter is considerably shorter for realistic cases, and if a numerical method is chosen to describe this process, the integration step is typically chosen even an order of magnitude shorter. This means that during several integration steps, v'-space, and the input parameters expressed in v' coordinates, may be considered to be constant. This implies that over such a period, the simple system of ODEs, described above, has to be solved. Over a longer period, Δt, but still with $\Delta t \ll \gamma^{-1}$, v'-space does grow, but slowly and linearly. The idea is now to add extra bins to the representation of the density occasionally. Each time an extra bin is added, the matrix $\mathcal{A}$ has to be recalculated, using new effective input parameters, and the rank of the system increases by one. It turns out one can do this with reasonable efficiency. Before Δt reaches γ^{-1}, one must rebin the solution to the original number of bins, so as to avoid exponential growth of v'-space. This is the basic idea of the algorithm that will be presented below.

4 Implementation Issues

4.1 Discretization in v-Space. As leakage will be transformed away, I will consider only $\gamma = 0$ and adapt the discretization scheme from Omurtag et al. (2000) for J^{input} and J^{reset}. To discretize, first one has to choose a (rescaled) potential $v_{\min} \leq 0$, such that $\rho(v, t) \approx 0$, for all t, $v \leq v_{\min}$. The idea is that one can always find a $v_{\min}$ such that $\rho(v, t) \approx 0$ for $v < v_{\min}$. For $\gamma > 0$, such a $v_{\min}$ always exists. For $\gamma = 0$, such a $v_{\min}$ exists only if $\bar{h} > 0$. The case $\gamma = 0, \bar{h} \leq 0$ is rather pathological, and I will not consider it here.

If v-space is discretized into N points, one may represent $\rho(v, t)$ by an array of N bins, where the bin number i is related to potential v by

$$v_i = i\Delta v + v_{\min}, \tag{4.1}$$

with $\Delta v = (1 - v_{\min})/(N - 1)$. For $i = 1, \ldots, N - 1$ equation 1.4 corresponds to:

$$\frac{d\rho_i}{dt} = -\frac{(J_i - J_{i-1})}{\Delta v}. \tag{4.2}$$

Here,

$$J_i(t) = J_i^{(input)}(t) + J_i^{(reset)}(t), \tag{4.3}$$

with $J_i^{(input)}(t) \equiv J^{input}(v_i, t)$ and $J_i^{(reset)}(t) \equiv J^{reset}(v_i, t)$. For $i = 0$, we have

$$\frac{d\rho_0}{dt} = -\frac{1}{\Delta v} J_0. \tag{4.4}$$

Discretizing equation 2.10, using equations 2.11 and 2.13, yields

$$J_i^{(input)} = \sigma(t)\Sigma_j Q_{ij}\rho_j, \tag{4.5}$$

where

$$Q_{ij} = \frac{1}{2} erfc\left(\Delta v \frac{[i-j-h-0.5]}{\sqrt{2}\alpha}\right) \equiv Q(i-j), \tag{4.6}$$

with $h \equiv \bar{h}/\Delta v$. The subtraction of $0.5\Delta v$ serves to make $\lim_{\alpha\to 0} Q_{ij} = \frac{1}{2}H(i-j-h)$. In the limit $\alpha \to 0$, this leads to transport of probability from one bin to exactly one bin if N is a multiple of $\bar{h}$.

Now define $r = (v_{reset} - v_{\min})/\Delta v$, the reset bin. For $J^{reset}(v,t)$ one finds, using equation 3.13,

$$J_i^{(reset)}(t) = H(i-r)J_{N-1}^{(input)}(t), \tag{4.7}$$

where $J(1,t) = J^{input}(1,t)$ was used, which is due to equation 3.14 and the fact that $v_{reset} < 1$.

Application of equation 4.2 and use of equations 4.5 and 4.7 directly lead to the following ODE:

$$\frac{d\vec{\rho}}{dt} = \mathcal{A}\cdot\vec{\rho}. \tag{4.8}$$

Here, the matrix $\mathcal{A}$ is given by

$$\mathcal{A} = \mathcal{A}^{(input)} + \mathcal{A}^{(reset)}. \tag{4.9}$$

The components of $\mathcal{A}^{(input)}$ for $i = 0, j = 0, \ldots, N-1$ are given by

$$\mathcal{A}_{ij}^{(input)} = \sigma(t)Q(i-j), \tag{4.10}$$

and for $i = 1, \ldots, N-1$; $j = 0, \ldots, N-1$ by

$$\mathcal{A}_{ij}^{(input)} = \sigma(t)(Q((i-j) - Q(i-1-j)), \tag{4.11}$$

and those of $\mathcal{A}^{(reset)}$ by

$$\mathcal{A}_{ij}^{(reset)} = \sigma(t)Q(N-1-j)\delta_{i,r} \tag{4.12}$$

4.2 Implementation of the Algorithm. To implement the ideas formulated above, the density $\rho(v, t)$ at $t = 0$ is represented in an array of N bins. These bins are fixed in v'-space and consequently move in v space as time progresses. The content of bin i is denoted by $C(i)$. The following relation between potential and bin number,

$$v_i = e^{-\gamma t}(i\Delta v + v_{\min}), \tag{4.13}$$

is used instead of equation 4.1, and $e^{\gamma t}C(i)$ rather than $C(i)$ represents $\rho(v_i, t)$. This then effectively implements the transformation of equation 3.21, and the array of bins represents ρ in v'-space rather than v-space. In the absence of input, the contents of the bin remain constant, and these constant contents of the array of bins represent the decaying density profile of equation 3.23.

As noted above, if the size of the array is kept constant, an interval that is shrinking in v-space is represented. To prevent this, at least for $v > 0$, an extra point is added to the array after a time $\Delta t = 1/\gamma \ln\{(N_{orig} + 1)/N_{orig}\}$ with

$$C(N_{orig} + 1) = 0, \tag{4.14}$$

where $N_{orig} = N$ at time $t = 0$. Boundary condition 3.14 is implemented by 4.14. One may repeat this process, and as long as $t < 1/\gamma$, the array size will grow approximately linearly. At some time $t = t_{reset}$, one must rebin the density profile, so that it fits in an array of N_{orig} bins again. One may consider adding bins to the front of the array as well, so as to cover the entire interval $[v_{\min}, 1]$ at all times, or alternatively, one may accept the shrinkage of the negative side of the interval from $v_{\min}$ to $v_{\min}e^{-\gamma t_{rebin}}$. For situations where the population density profile extends only to small negative values in v-space, this is acceptable and leads to a simpler implementation.

If there is input, the contents of the array will change and must be calculated. As long as $t < 1/\gamma \ln\{(N + 1)/N\} \equiv t_{N+1}$, the input parameters $\bar{h}$ and α can be considered constant. Equation 4.8 can then be used directly to calculate $\frac{d\vec{\rho}}{dt}$ and to evolve $\rho_i(t)$ from $t = 0$ to $t = t_{N+1}$. It is natural to take a time step that divides t_{N+1} into an entire number: $h = t_{N+1}/k$ for some integer k. Symbolically, one may write:

$$\vec{\rho}(t + h) = \vec{\rho}(t) + h\frac{d\vec{\rho}(t)}{dt}. \tag{4.15}$$

Obviously, such a step may be implemented by a Runge-Kutta method, which was used here, or by a Euler method, which consists of a literal implementation of equation 4.15. At time $t = t_{N+1}$, a bin is added to the density array, and one has to recalculate matrices $\mathcal{A}^{(reset)}$ and $\mathcal{A}^{(input)}$. These matrices now will have rank $N + 1$ instead of N and have to be calculated

from new values for the input parameters:

$$\begin{cases} \bar{h}' = e^{\gamma t_{N+1}} \bar{h} \\ \alpha' = e^{\gamma t_{N+1}} \alpha. \end{cases} \tag{4.16}$$

The entire process now can be repeated until $t = t_{rebin}$.

For rebinning, cubic polynomial interpolation is used. During this interpolation, the contents of the reset bin, which typically contains much more density than the bins around it, is replaced by the average of its neighbors. After rebinning, an amount of probability is added to the reset bin, which, together with the probability already present, adds to one.

4.3 Efficiency. The algorithm as presented above is essentially the one presented by Omurtag et al. (2000), with two important differences. First, there is no leakage term matrix $\mathcal{A}^{(leak)}$ present in this algorithm. Second, the matrices $\mathcal{A}^{(input)}$ and $\mathcal{A}^{(reset)}$ must be recalculated regularly, whereas for constant input parameters, they must be calculated only once in Omurtag et al. (2000). Since the algorithm presented here is equivalent to the one presented in Omurtag et al. (2000) without leakage, it is at least as fast, provided that the computing time to calculate the matrices $\mathcal{A}$ can be reduced to a minimum.

One important observation is that the function Q_{ij} is dependent only on the difference $i - j$, as shown in equation 4.5, and by equation 4.11, this is also true for $\mathcal{A}_{ij}^{(input)}$. All rows of $\mathcal{A}^{(input)}$ can therefore be represented in one single vector $\vec{\mathcal{A}}$, whose indices are in the range $[-N + 1, N - 1]$, where N is the rank of the matrix $\mathcal{A}^{(input)}$. One can now obtain every row of $\mathcal{A}^{(input)}$ by using a pointer to the right element of $\vec{\mathcal{A}}$.

A second observation is that for constant input parameters, only a limited number of matrices $A^{(input)}$ must be calculated. If N slowly increases from N_{orig} to N_{rebin}, one must calculate $N_{rebin} - N_{orig}$ different matrices, but they will be used over and over again. They can easily be computed before the evolution of the density starts. If the input parameters are not constant, the matrices $\mathcal{A}$ must be recalculated during evolution, but this would also be true for the scheme used by Omurtag et al. (2000) and indeed for every other scheme.

The accuracy of the algorithm is determined by the number of grid points at the start of the evolution. On average, during evolution, the algorithm uses more grid points, which does not, however, lead to improved accuracy, because this is lost once rebinning takes place. This and the rebinning itself are forms of overhead, do not occur in the original scheme, but do not have a large impact on the total computing time. I conclude that the algorithm presented here from a computational point of view is almost identical to the scheme used by Omurtag et al. (2000) for $\gamma = 0$ and because no leakage

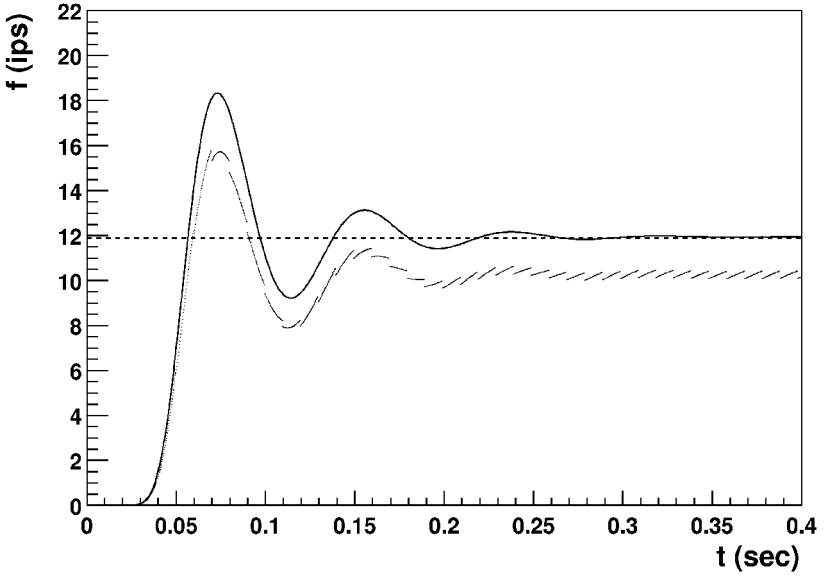

Figure 1: Result of the algorithm for $\bar{h} = 0.03$, $\alpha = 10^{-2}$. The solid curve gives the firing rate as calculated from equation 3.17. It is in excellent agreement with earlier results from Omurtag et al. (2000, figure 4) and asymptotes to the theoretical equilibrium value of $11.82\,\mathrm{s}^{-1}$. The jagged curve results from a naive application of equation 4.5.

has to be included, it is at least as efficient. Since that scheme is by far the simplest of all finite difference schemes (see, e.g., Nykamp & Tranchina, 2000), the algorithm presented here compares well to these schemes in terms of efficiency.

5 Results

The algorithm was applied to a grid of $N = 200$ points with input parameters $\bar{h} = 0.03, \alpha = 10^{-2}$. The inverse membrane time constant, γ, was taken to be $20s^{-1}$. Evolution was carried out over 0.4 s. From the evolving density profile, the firing rate can be calculated, although some care must be exercised. The solid curve in Figure 1 follows from equation 3.17, where the integral was evaluated numerically. Cubic spline interpolation was used to evaluate $\rho(v)$ between grid points. The equilibrium firing rate ($\lim_{t\to\infty} r(t)$) can be calculated analytically and is $11.82\ \mathrm{s}^{-1}$ for the chosen input parameters (Sirovich et al., 2000). The solid curve indeed asymptotes to this value. Moreover, the result agrees well with an earlier calculation by Omurtag et al. (2000, Fig. 4).

It is instructive to use equation 4.5 for $i = N - 1$ to calculate the firing rate,

$$r(t) = \sigma(t)\Sigma_j Q(N - 1 - j)\rho_j, \tag{5.1}$$

which is in line with the method used by Omurtag et al. (2000). The result is given by the lighter jagged curve in Figure 1, which underestimates the firing rate systematically and also shows rebinning artifacts. To realize where this discrepancy comes from, it is necessary to understand that the rate is determined by only a few density points in the highest bins and that only a slight misrepresentation of the density profile there will result in a serious deviation in the calculated rate. For instance, at the start of the evolution, for $N = 200, \bar{h} = 0.03, \alpha = 10^{-3}$, only the density in the six highest bins contributes to discretized rate integral. As the evolution progresses, the cut-off, which is centered at $v = 1 - \bar{h} = 0.97$, will move and be somewhere in the middle of a bin. The result is that part of the density in that bin should have been added to the rate contribution. For equation 5.1, where Q for small α may be considered as a step function, the contribution will be added only if the cut-off encompasses the whole bin, however. This is a consequence of the representation of the density in a frame of reference, which is time dependent (in contrast to Omurtag et al., 2000).

The algorithm is very stable, and the numerical evolution of the differential equation is absolutely insensitive to the density gradient. The density profile in Figure 2, for instance, was obtained for the same parameters that were used above, except for α, which was chosen as $\alpha = 10^{-8}$. This leads to a very steep density profile near $v = 0$, which the algorithm handled without a problem.

The only moment where the density gradient may cause problems is at the time of rebinning. Isolated bins that contain density, such as, for instance, the reset bin, must be handled separately. They are best removed or replaced by the average density in their neighborhood. The resulting density profile is usually smooth enough to interpolate, which I have done using cubic polynomial interpolation. The removed density may then be replaced in its bins.

6 Discussion

In this article, I have used the method of characteristics to reduce the population equation to a set of ODEs. This method has previously been used to write down a formal solution of a population equation (e.g., Gerstner & van Hemmen, 1992; Gerstner, 2001) and also in an analytical solution of the two-compartment model (Sirovich, 2003). Here, however, it is used as a basis for a numerical solution in response to an arbitrary time-dependent input. In the case of LIF neurons, where a simple analytic expression for the characteristics of the population density equation can be found, the method results in a simple and elegant algorithm to find the population density. This

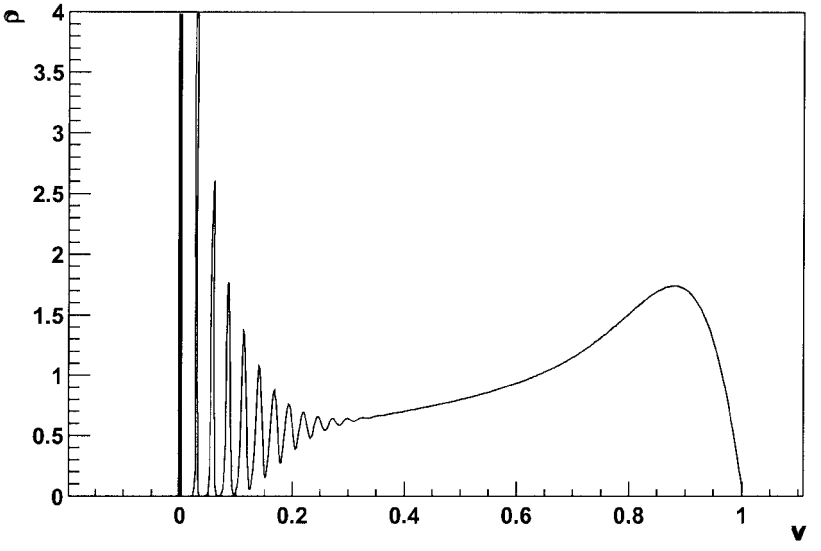

Figure 2: The equilibrium density profile of the density ρ. Here $\alpha = 10^{-8}$ was used, which effectively means an infinitely sharp distribution for the input synapses of the population. This leads to a very steep density profile near $v = 0$, which is no problem for the algorithm described here.

algorithm relies on very simple and stable numerical techniques, which result in a population density that is manifestly nonnegative. It is insensitive to the density gradient and is easy to implement.

Finite difference schemes, on the other hand, are sensitive to density gradients. Steep density gradients may occur quite naturally during the evolution of the density, for instance, at the reset potential but also elsewhere (see Figure 2). As Nykamp and Tranchina (2000) and in particular Casti et al. (2002) demonstrate, quite sophisticated algorithms are necessary to ensure a stable solution. The algorithm presented here is at least as efficient as any of the finite difference schemes in terms of computing time. This is particularly true for input parameters, which are time-dependent. Both kinds of methods have in common that they evolve the density in time.

A very direct way to calculate solutions of the population density equation was proposed by Knight (2000): v-space is discretized, and the population density equation reduces to a set of ODEs, much like equation 4.8:

$$\frac{d\vec{\rho}}{dt} = \mathcal{Q}\vec{\rho}. \tag{6.1}$$

A solution can be given directly in terms of the eigenvalues of $\mathcal{Q}$. Moreover, Knight (2000) argues that only a few eigenvalues dominate the solution, so that a very efficient way to solve the population density equation is

obtained. There is a relation between this method and the method proposed in this article: in the absence of neuronal dynamics ($\gamma = 0$, for LIF neurons), they are equivalent. Indeed the matrix in equation 3.26 is the simplest of all operators Q. The most important conceptual difference between the method of Knight (2000) and the one presented in this article is that in the former, no transformation to v'-space is made, before discretization.

The method proposed by Knight (2000) is certainly the most efficient for a large number of cases, such as, for instance, the jump response (Knight, Omurtag, & Sirovich, 2000). There are other cases for which a numerical solution like the one discussed in this article may be more appropriate. One situation where this might be the case is if the input parameters are time dependent. Such time dependence might, for instance, come from neural processes like adaptation. Or part of the population's input may be considered as an effective input, due to a large number of afferent neuron populations. Sometimes such an input is well described by a gaussian white noise (Amit & Tsodyks, 1991), whose varying mean and variance reflect transient changes of a network's activity. Such a contribution might be effectively described by a single population with time-dependent input parameters. For time-dependent input parameters, the diagonalization of $\mathcal{Q}$ must be repeated regularly, and it may be that this becomes too time-consuming in some cases.

To summarize, time-stepping methods to solve the population density equation probably have their place. The method presented in this article is particularly efficient for time-dependent input parameters. If such a method is appropriate, stability becomes an issue. Transformation to v'-space could be instrumental in obtaining a stable method, as the algorithm developed in this article shows.

Acknowledgments

I thank Frank van der Velde for a careful reading of the manuscript for this article. I also thank an anonymous referee for numerous suggestions to improve the text.

References

Abbott, L. F., & van Vreeswijk, C. (1993). Asynchronous states in networks of pulse-coupled oscillators. *Physical Review E, 48*, 1483–1490.

Amit, D. J., & Tsodyks, M. V. (1991). Quantitative study of attractor neural network retrieving at low spike rates: I. substrate-spikes, rates, and neuronal gain. *Network, 2*, 259–273.

Casti, A., Omurtag, A., Sornborger, A., Kaplan, E., Knight, B., Victor, J., & Sirovich, L. (2002). A population study of integrate-and-fire-or-burst neurons. *Neural Computation, 14*, 947–986.

Gardiner, C. W. (1997). *Handbook of stochastic methods for physics, chemistry, and the natural sciences*. New York: Springer-Verlag.

Gerstner, W. (1995). Time structure of the activity in neural network models. *Physical Review E, 51*, 738–758.

Gerstner, W. (2001). Coding properties of spiking neurons: Reverse and cross-correlations. *Neural Networks, 14*, 599–610.

Gerstner, W., & van Hemmen, J. L. (1992). Associative memory in a network of "spiking" neurons. *Network: Computation in Neural Systems, 3*, 139–164.

Haskell, E., Nykamp, D. Q., & Tranchina, D. (2001). Population density methods for large-scale modelling of neuronal networks with realistic synaptic kinetics: Cutting the dimension down to size. *Network: Computation in Neural Systems, 12*, 141–174.

Johannesma, P. I. M. (1966). *Stochastic neural activity: A theoretical investigation*. Unpublished doctoral dissertation, University of Nijmegen.

John, F. (1981). *Partial differential equations*. New York: Springer-Verlag.

de Kamps, M., & van der Velde, F. (2001). From artificial neural networks to spiking populations of neurons and back again. *Neural Networks, 14*, 941–953.

Knight, B. W. (2000). Dynamics of encoding in neuron populations: Some general mathematical features. *Neural Computation, 12*, 473–518.

Knight, B. W., Manin, D., & Sirovich, L. (1996). Dynamical models of interacting neuron populations in visual cortex. In *Symposium on Robotics and Cybernetics: Computational Engineering in Systems Applications*. E. C. Gerf, France: Cite Scientifique.

Knight, B. W., Omurtag, A., & Sirovich, L. (2000). The approach of a neuron population firing rate to a new equilibrium: An exact theoretical result. *Neural Computation, 12*, 1045–1055.

Kuramoto, T. (1991). Collective synchronization of pulse-coupled oscillators and excitable units. *Physica D, 50*, 15–30.

Maass, W. (1997). Fast sigmoidal networks via spiking neurons. *Neural Computation, 9*, 279–304.

Nykamp, D. Q., & Tranchina, D. (2000). A population density approach that facilitates large-scale modeling of neural networks: Analysis and an application to orientation tuning. *Journal of Computational Neuroscience, 8*, 19–50.

Omurtag, A., Knight, B. W., & Sirovich, L. (2000). On the simulation of large populations of neurons. *Journal of Computational Neuroscience, 8*, 51–63.

Sirovich, L. (2003). Dynamics of neuronal populations: Eigenfunction theory, some solvable cases *Network: Computation in Neural Systems, 14*, 249–272.

Sirovich, L., Omurtag, A., & Knight, B. W. (2000). Dynamics of neural populations: The equilibrium solutions. *SIAM Journal of Applied Mathematics, 60*, 2009–2028.

Stein, R. B. (1965). A theoretical analysis of neuronal variability. *Biophysics Journal, 5*, 173–194.

Wilbur, W. J., & Rinzel, J. (1982). An analysis of Stein's model for stochastic neuronal excitation. *Biological Cybernetics, 45*, 107–114.

Received August 13, 2002; accepted February 18, 2003.