Journal of Scientific Computing (2020) 84:10
https://doi.org/10.1007/s10915-020-01261-6

A Combined Offline–Online Algorithm for Hodgkin–Huxley Neural Networks

Zhong-qi Kyle Tian[1] · Jennifer Crodelle[2] · Douglas Zhou[1]

Received: 23 December 2019 / Revised: 29 April 2020 / Accepted: 10 June 2020

Abstract
Spiking neural networks are widely applied to simulate cortical dynamics in the brain and are regarded as the next generation of machine learning. The classical Hodgkin–Huxley (HH) neuron is the foundation of all spiking neural models. In numerical simulation, however, the stiffness of the nonlinear HH equations during an action potential (a spike) period prohibits the use of large time steps for numerical integration. Outside of this stiff period, the HH equations can be efficiently simulated with a relatively large time step. In this work, we present an efficient and accurate offline–online combined method that stops evolving the HH equations during an action potential period, uses a pre-computed (offline) high-resolution data set to determine the voltage value during the spike, and restarts the time evolution of the HH equations after the stiff period using reset values interpolated from the offline data set. Our method allows for time steps an order of magnitude larger than those used in the standard Runge–Kutta (RK) method, while accurately capturing dynamical properties of HH neurons. In addition, this offline–online method robustly achieves a maximum of a tenfold decrease in computation time as compared to RK methods, a result that is independent of network size.

Keywords Fast algorithm · Offline–online method · Hodgkin–Huxley · Numerical simulation

1 Introduction

The Hodgkin–Huxley (HH) model [7,16,18], originally proposed to describe the detailed generation of action potentials in the squid's giant axon, is widely used to simulate neural network dynamics in the neuroscience field [7,16,18]. The HH system characterizes the evolution of both a neuron's voltage and its gating variables related to the opening and closing of ion channels. In practice, numerical simulation of the HH equations generally

✉ Douglas Zhou
zdz@sjtu.edu.cn

[1] School of Mathematical Sciences, MOE-LSC, Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai, China

[2] Courant Institute of Mathematical Sciences, New York University, New York, NY, USA

Published online: 24 June 2020

involves the evolution of the ordinary differential equations using standard methods, e.g., Runge–Kutta (RK) schemes. However, due to the severe nonlinearities of the system, the HH equations become very stiff during an action potential period, forcing the use of a small time step to avoid instability. Often, computational neuroscientists need to simulate many realizations of a large network of neurons to study network properties or simulate the system for a long time to obtain steady-state network dynamics [20,51]. These simulations can take an impractically large amount of time when evolving the system using a standard numerical scheme with a small time step. Therefore, it is valuable to design fast algorithms that allow a large time step to numerically evolve the HH system of equations.

One approach that can support the use of a large time step to evolve the HH neural network was proposed in Ref. [3], based on the exponential time differencing (ETD) method [5,27]. In the previous work, the HH equations are linearly approximated in each time step, and then analytically solved. The proposed ETD method in Ref. [3] is proven to be unconditionally stable for HH equations, but it will be inaccurate when using a large time step. Another approach that supports a large time step is based on the library method [42]. In this reference, a library is built using stationary information in the regimes of stable periodic firing. However, neurons driven by stochastic spike input are generally not in the stable periodic state. As a result, the library method fails to capture certain important characteristics, e.g., the transient dynamics, Hopf bifurcation point of an HH neuron, and accurate shapes of action potentials when the neuron is not in a stable periodic state.

In this work, we propose a combined offline–online (COO) method that can overcome all the above issues related to the previous works. Note that the underlying dynamics that make the HH equations stiff stem from a large transmembrane current resulting from the opening and closing of sodium and potassium ion channels. When a neuron fires an action potential, sodium and potassium ion channels open, causing a transmembrane current with a maximum value over 250 μA cm^{-2}. This stiff period lasts on average about 3.5 ms, the duration of an action potential. Then, the HH equations enter a relatively non-stiff state (small slope) where the transmembrane current is around 5 μA cm^{-2}. During this time, a large time step in the numerical scheme can be used to accurately resolve the voltage until the next action potential occurs and the transmembrane current jumps up again. Based on this observation, our COO method simply skips over the duration of an action potential when the transmembrane current is large. To be specific, when an HH neuron's membrane potential reaches a firing threshold, one stops evolving its HH equations and restarts after the stiff period with reset values interpolated from a pre-computed (offline) high-resolution data set. Therefore, one can skip the stiff period and use a large time step to evolve the HH equations to raise efficiency.

We demonstrate through numerical simulations that our COO algorithm can well capture statistical properties of HH neural networks such as the average firing rate, firing patterns, and chaotic attractors [1,13,15]. We further give a quantitative efficiency estimation of the COO method compared with the standard RK method, e.g., the second-order RK (RK2) method. Importantly, we show that the COO method can robustly use a much larger time step and raise the computational efficiency one order of magnitude over the RK2 method. This high efficiency does not rely on the dynamical regime, network size, network connectivity structure, or neuron types such as excitatory and inhibitory. Finally, we provide a detailed error analysis of the COO method which may give insight about how to set up the offline data set and improve its performance for certain situations.

Note that the offline data set in this work was constructed using a prototypical set of HH neuron parameters given in Ref. [8]. For other HH neuron parameters, the action potentials may have different shapes, and our offline data construction procedure can be easily general-

ized to these cases. In addition, the offline data construction procedure can also be extended to other HH-type neurons with different ion channels, such as fast-spiking, intrinsically-bursting, and low-threshold spiking neurons [32]. We emphasize that the high efficiency and well-captured statistical properties of the COO method with corresponding offline data set for HH neural network simulations can still be guaranteed.

The outline of the paper is as follows. In Sect. 2, we introduce the HH neuron equations, including the parameter set used in the construction of the offline data set. In Sect. 3, we describe the standard RK method used to evolve HH networks. In Sect. 4, we present the COO method, including details on construction and application of the data set. In Sect. 5, we compare the numerical results obtained by the standard RK method with the COO method and provide a detailed error analysis of the COO method. Several issues related to the COO method, e.g., the generalization of the method, are discussed in Sect. 6.

2 The Model

The dynamics of the ith neuron of an HH network with N excitatory neurons is governed by the following set of equations

$$C\frac{dV_i}{dt} = -(V_i - V_{Na})G_{Na}m_i^3 h_i - (V_i - V_K)G_K n_i^4 - (V_i - V_L)G_L + I_i^{\text{input}}, \quad (1)$$

$$\frac{dz_i}{dt} = (1 - z_i)\alpha_z(V_i) - z_i\beta_z(V_i), \quad \text{for } z = m, h, n, \quad (2)$$

where C is the cell membrane capacitance; V_i is the membrane potential; m_i, h_i, and n_i are gating variables; V_{Na}, V_K, and V_L are the reversal potentials for the sodium, potassium, and leak currents, respectively; and G_{Na}, G_K, and G_L are the corresponding maximum conductances. The rate variables α and β are defined in Ref. [8] as

$$\alpha_m(V) = (0.1V + 4)/(1 - \exp(-0.1V - 4)), \qquad \beta_m(V) = 4\exp(-(V + 65)/18),$$
$$\alpha_h(V) = 0.07\exp(-(V + 65)/20), \qquad \beta_h(V) = 1/(1 + \exp(-3.5 - 0.1V)),$$
$$\alpha_n(V) = (0.01V + 0.55)/(1 - \exp(-0.1V - 5.5)), \quad \beta_n(V) = 0.125\exp(-(V + 65)/80).$$

The input current is given by $I_i^{\text{input}} = -G_i(t)(V_i - V_G)$ where the conductance is defined as

$$G_i(t) = f\sum_l H(\sigma_d, \sigma_r, t - s_{il}) + \sum_j S_{ij}\sum_l H(\sigma_d, \sigma_r, t - \tau_{jl}), \quad (3)$$

with V_G as the reversal potential and s_{il} as the spike time of the feedforward Poisson input with strength f and rate ν. The spike-induced conductance change is defined by an *alpha* function [14,40]

$$H(\sigma_d, \sigma_r, t) = \frac{\sigma_d\sigma_r}{\sigma_d - \sigma_r}\left[\exp\left(-\frac{t}{\sigma_d}\right) - \exp\left(-\frac{t}{\sigma_r}\right)\right]\Theta(t), \quad (4)$$

where σ_d and σ_r are the slow decay and fast rise time scale, respectively, and $\Theta(\cdot)$ is the Heaviside function. S_{ij} is the coupling strength from the jth neuron to the ith neuron and τ_{jl} is the lth spike time of the jth neuron. We take the parameters as in Ref. [8] that $C = 1\ \mu\text{F cm}^{-2}$, $V_{Na} = 50$ mV, $V_K = -77$ mV, $V_L = -54.387$ mV, $G_{Na} = 120\ \text{mS cm}^{-2}$, $G_K = 36\ \text{mS cm}^{-2}$, $G_L = 0.3\ \text{mS cm}^{-2}$ and $V_G = 0$ mV. Time constants are set as $\sigma_r = 0.5$ ms and $\sigma_d = 3.0$ ms.

When the voltage V_i reaches the firing threshold, V^{th}, we say the ith neuron fires a spike at this time. Instantaneously, all of its postsynaptic neurons receive this spike and the affected change of conductance follows Eq. (4). For the sake of simplicity, we here consider an all-to-all connected network of excitatory neurons with $S_{ij} = S/N$. However, the conclusions shown in this work hold for more complicated situations, e.g., networks of both excitatory and inhibitory neurons with more realistic connectivity, such as, a network in which the coupling strength follows the typical log-normal distribution [19,41].

3 Runge–Kutta Method

We first introduce the standard RK2 method with a fixed time step Δt [45] for the numerical evolution of the HH neural network. For the ease of illustration, we define the vector

$$X_i(t) = [V_i(t), m_i(t), h_i(t), n_i(t), G_i(t)] \tag{5}$$

to represent all dynamic variables of the ith neuron in the above HH network model.

Consider the evolution of the HH network from t_0 to $t_0 + \Delta t$. Since we do not know if any neuron spiked within the time step until the end of the time step, we evolve the network without considering the input of those synaptic spikes and recalibrate their influence at the end of the time step. The feedforward input in the time step are treated in the same way. If any neuron spiked within this time step, say neuron i, the spike time is estimated by a linear interpolation

$$t_{\text{spike}} = \frac{V^{\text{th}} - V_i(t_0)}{V_i(t_0 + \Delta t) - V_i(t_0)} \Delta t + t_0 \tag{6}$$

to maintain an overall second-order numerical accuracy [14,24,37]. The conductance of each neuron before recalibration is evolved by

$$\tilde{G}_i(t_0 + \Delta t) = f \sum_{s_{il} \le t_0} H(\sigma_d, \sigma_r, t_0 + \Delta t - s_{il}) + \sum_j \frac{S}{N} \sum_{\tau_{jl} \le t_0} H(\sigma_d, \sigma_r, t_0 + \Delta t - \tau_{jl}), \tag{7}$$

and should be recalibrated as

$$\begin{aligned} G_i(t_0 + \Delta t) = \tilde{G}_i(t_0 + \Delta t) &+ f \sum_{t_0 < s_{il} \le t_0 + \Delta t} H(\sigma_d, \sigma_r, t_0 + \Delta t - s_{il}) \\ &+ \sum_j \frac{S}{N} \sum_{t_0 < \tau_{jl} \le t_0 + \Delta t} H(\sigma_d, \sigma_r, t_0 + \Delta t - \tau_{jl}), \end{aligned} \tag{8}$$

for $i = 1, 2, ..., N$. The procedure of the RK2 method is given in Algorithm 1.

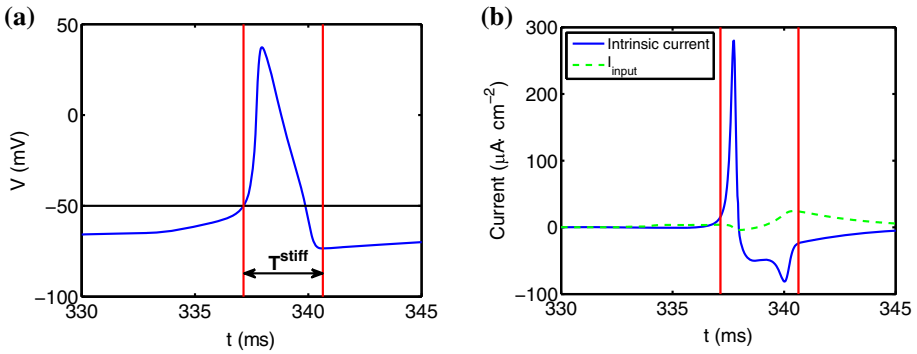

Fig. 1 Typical firing event of an HH neuron in an all-to-all connected network of 128 excitatory neurons using the RK2 method with $\Delta t = 0.03125$ ms. **a** The trajectory of voltage V. The black horizontal line indicates the firing threshold V^{th} and the red vertical lines indicate the stiff period. **b** The trajectory of the intrinsic current (blue solid curve) and input current I^{input} (green dashed curve). The intrinsic current is the sum of ionic and leak currents. The coupling strength is $S = 1.2$ mS cm^{-2} (Color figure online)

Algorithm 1: RK2 algorithm

Input: an initial time t_0, time step Δt, feedforward input times $\{s_{il}\}$
Output: $\{X_i(t_0 + \Delta t)\}$ and $\{\tau_{il}\}$ (if any fired)

```
for i = 1 to N do
    Advance the HH equations for the ith neuron without considering any spike input
    using RK2 scheme.
    if V_i(t_0) < V^th and V_i(t_0 + Δt) ≥ V^th then
        The ith neuron spiked in [t_0, t_0 + Δt].
        Estimate the spike time by Eq. (6).
    end
end
Update the conductance of each neuron by Eq. (8).
```

4 Combined Offline–Online Method

When a neuron fires an action potential, the HH equations are stiff for about 3.5 ms (the duration of the action potential), denoted by T^{stiff} and shown in Fig. 1a. This stiff period requires a sufficiently small time step to accurately resolve the dynamics; however, outside of this stiff period, the HH equations are relatively non-stiff, as shown in Fig. 1, and the dynamics can be resolved using a large time step. The idea of the COO method is to pre-compute a high-resolution (offline) data set of V, m, h, n during the stiff period such that we can stop evolving the dynamic variables V, m, h, n during the stiff period T^{stiff}, and restart after the stiff period with the values interpolated from this offline data set. Thus we can avoid evolving the HH equations during the stiff period and use a large time step to evolve the HH equations outside of the stiff period.

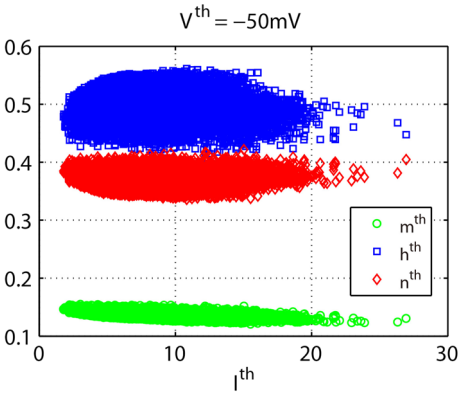

Fig. 2 The ranges of values $m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ in terms of I^{th}

4.1 Building Offline Data Set

We now describe, in detail, the method by which we build the offline data set. When a neuron's membrane potential reaches the firing threshold V^{th} and fires an action potential, we calculate and record the values $I^{\text{input}}, m, h, n$ at that spike time and denote them by $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, respectively. If we know the exact trajectory of I^{input} for the subsequent stiff period, T^{stiff}, we can use a sufficiently small time step to evolve Eqs. (1, 2) for T^{stiff} ms with initial values $V^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ to obtain high-resolution trajectories of V, m, h, n. The values at the end of the stiff period (after 3.5 ms) are called the reset values of these variables and are denoted by $V^{\text{re}}, m^{\text{re}}, h^{\text{re}}, n^{\text{re}}$.

However, it is impossible to obtain the exact trajectory of I^{input} without knowing the future feedforward and synaptic spike times. As shown in Fig. 1b, I^{input} varies during the stiff period usually in the range of $[-15, 20]\ \mu\text{A cm}^{-2}$, while the intrinsic current, the sum of ionic and leak currents, is about $20\ \mu\text{A cm}^{-2}$ at the spike time, and quickly rises to the peak value about $300\ \mu\text{A cm}^{-2}$, then quickly drops to the negative value less than $-50\,\mu\text{A cm}^{-2}$ for the remainder of the stiff period. Clearly the intrinsic current is dominant in the stiff period and so the variation of the I^{input} can be almost ignored in comparison with the intrinsic current, thus we can regard it as constant during the stiff period to build the offline data set. We emphasize that this is the only assumption made in the COO method. Then, for a suite of $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, we can obtain the corresponding suite of $V^{\text{re}}, m^{\text{re}}, h^{\text{re}}, n^{\text{re}}$ by evolving Eqs. (1, 2) for T^{stiff} ms with initial values $V^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ and constant input $I^{\text{input}} = I^{\text{th}}$.

To build the suite of threshold values for all variables, we choose N_I, N_m, N_h, N_n different values of $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, respectively, equally distributed in their ranges. For each combination of $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, we evolve the Eqs. (1, 2) for a time interval of T^{stiff} ms using the RK2 method with a sufficiently small time step, e.g., $\Delta t = 1 \times 10^{-6}$ ms. During this stiff period, as mentioned before, we take a constant input current; specifically, we assume that $I^{\text{input}} = I^{\text{th}}$.

In this work, we take the ranges $[0, 50]\ \mu\text{A cm}^{-2}$, $[0, 0.3]$, $[0.2, 0.6]$, and $[0.3, 0.6]$ for $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, respectively, with sampling intervals of $\Delta I = 2.5\ \mu\text{A cm}^{-2}$, $\Delta m = \Delta h = \Delta n = 0.02$. This corresponds to sample numbers $N_I = 21$, $N_m = 16$, $N_h = 21$, $N_n = 16$. The full suite of values for $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ is shown in Fig. 2. This offline data set is built with a total size of $8N_I N_m N_h N_n$ possible combinations of values for the neuron variables and occupies only 7 megabytes of data in binary form, a small amount of space for today's

computers. Note that this data set only records the threshold and reset values, but not the whole trajectory of the action potential. However, we point out that the whole trajectory of an action potential can still be well recovered if the voltage traces computed from the suite of above dynamic variables are recorded. When including voltage traces in the second data set, its size becomes much larger: $N_I N_m N_h N_n (4 + T^{\text{stiff}}/\delta t)$, where δt is the time interval to record the voltage traces and can be much larger than the sufficiently small time step Δt in evolving, e.g., $\delta t = 0.02$ ms while $\Delta t = 1 \times 10^{-6}$ ms in this work.

One important obstacle in building the offline data set is to choose a proper firing threshold value V^{th}. The firing threshold should be low enough such that the HH equations can be numerically evolved using a large time step, but high enough that a neuron will definitely spike after its membrane potential reaches the firing threshold. In this work, we take $V^{\text{th}} = -50$ mV. Correspondingly, we take the length of stiff period $T^{\text{stiff}} = 3.5$ ms according to the dynamics of HH system in Eqs. (1, 2), which is long enough to cover all stiff parts of the resulting action potential from normal firing events. Note that, with changes in the neuron parameters such as the leak conductance and reversal potential, these values might also need to be changed to account for different dynamics in the neuron's behavior.

4.2 Using the Offline Data Set

Next, we illustrate how to implement the COO method. Once a neuron's membrane potential exceeds the firing threshold V^{th} for the time step from t_0 to $t_0 + \Delta t$, we compute the values of $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ at the threshold using linear interpolation:

$$z^{\mathbf{th}} = \frac{z(t_0 + \Delta t) - z(t_0)}{\Delta t}(t_{\text{spike}} - t_0) + z(t_0), \quad \text{for } z = I, m, h, n, \tag{9}$$

where the spike time, t_{spike}, is estimated by linear interpolation as shown in Eq. (6). Then, we stop evolving V, m, h, n for the following T^{stiff} ms and calculate the reset values of these variables $V^{\text{re}}, m^{\text{re}}, h^{\text{re}}, n^{\text{re}}$ at $t_{\text{spike}} + T^{\text{stiff}}$ ms using the threshold values $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$. Note that since it is unlikely that the computed threshold values from Eq. (9) are exactly those that were used to build the data set, we can perform a linear interpolation from the pre-computed high-resolution data set. For example, suppose that I^{th} falls between two data points I_0^{th} and I_1^{th} in the data set, but $m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$ are exactly the sample points when building the data set. Then, the reset values z^{re} can be approximated by a linear combination of the reset values that result from the data points in data set on either side of the calculated threshold value, weighted by the difference between the calculated threshold value and each data set value:

$$\begin{aligned} z^{\text{re}} = &\left(\frac{I^{\text{th}} - I_1^{\text{th}}}{I_0^{\text{th}} - I_1^{\text{th}}}\right) z^{\text{re}}(I_0^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}) \\ &+ \left(\frac{I^{\text{th}} - I_0^{\text{th}}}{I_1^{\text{th}} - I_0^{\text{th}}}\right) z^{\text{re}}(I_1^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}), \quad \text{for } z = V, m, h, n. \end{aligned}$$

By extension, when m^{th} falls between the data points m_0^{th} and m_1^{th}, h^{th} between h_0^{th} and h_1^{th}, and n^{th} between n_0^{th} and n_1^{th}, we can approximate the reset values in the same way:

$$z^{\text{re}} = \sum_{i,j,k,l=0,1} \left(\frac{I^{\text{th}} - I_{1-i}^{\text{th}}}{I_i^{\text{th}} - I_{1-i}^{\text{th}}} \frac{m^{\text{th}} - m_{1-j}^{\text{th}}}{m_j^{\text{th}} - m_{1-j}^{\text{th}}} \frac{h^{\text{th}} - h_{1-k}^{\text{th}}}{h_k^{\text{th}} - h_{1-k}^{\text{th}}} \frac{n^{\text{th}} - n_{1-l}^{\text{th}}}{n_l^{\text{th}} - n_{1-l}^{\text{th}}}\right) z^{\text{re}}(I_i^{\text{th}}, m_j^{\text{th}}, h_k^{\text{th}}, n_l^{\text{th}}) \tag{10}$$

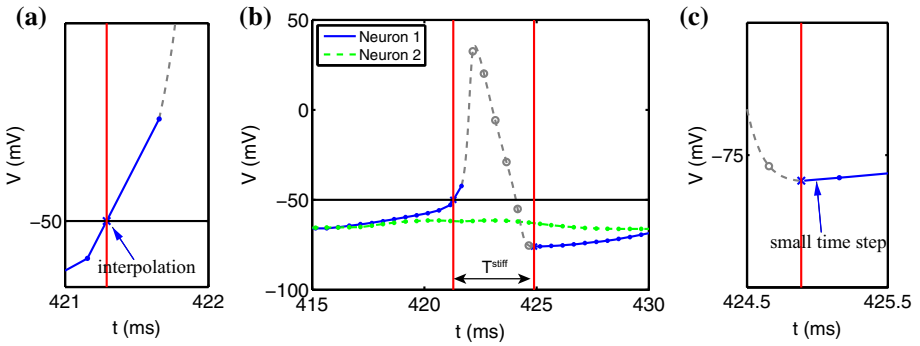

Fig. 3 Illustration of the COO method for two sample neurons, say neuron 1 and neuron 2. **a** Estimating the spike time of neuron 1 through linear interpolation. **b** After neuron 1 fires an action potential, we record the values $I^{\text{th}}, m^{\text{th}}, h^{\text{th}}, n^{\text{th}}$, stop evolving its V, m, h, n for the following T^{stiff} ms, and restart with the values $V^{\text{re}}, m^{\text{re}}, h^{\text{re}}, n^{\text{re}}$ interpolated from the data set. The blue (for neuron 1) and green (for neuron 2) dots indicate the time nodes where we use the standard RK2 scheme while the gray circles indicate the time nodes that we only evolve the conductance G. The gray curve represents the shape of the action potential that is recovered using the second data set of voltage traces. **c** The end of the stiff period may be in the middle of the original time step and we evolve it with a suitably small time step to obtain the values of all dynamic variables at the end of the RK2 time step (Color figure online)

for $z = V, m, h, n$. Figure 3a demonstrates how the COO method uses interpolation to find the spike time, while Fig. 3b shows how one skips the evolution of HH equations during the stiff period. The shape of the action potential as shown in gray in Fig. 3b is missing if one is not interested in the spike shape. Otherwise, it can be recovered through linear interpolation of values from the data set with recorded voltage traces as discussed previously.

We evolve the conductance G analytically during the stiff period by Eq. (3). Note that, when we begin to evolve the dynamic variables for this neuron at the end of the stiff period, it may be the case that we are in the middle of a time step. In this case, we use a suitably small time step, as illustrated in Fig. 3c, to ensure that the conductance and neuron variables are back on the same time-stepping scheme. Then, we can return to a large time step using the standard RK2 scheme to evolve all of the dynamic variables. The procedure of the COO algorithm is given in Algorithm 2.

Algorithm 2: Combined offline–online algorithm

Input: an initial time t_0, time step Δt, feedforward input times $\{s_{il}\}$
Output: $\{X_i(t_0 + \Delta t)\}$ and $\{\tau_{il}\}$ (if any fired)

```
for i = 1 to N do
    Advance the HH equations for the ith neuron without considering any spike input
    using RK2 scheme.
    if V_i(t_0) < V^th, V_i(t_0 + Δt) ≥ V^th then
        The ith neuron spiked in [t_0, t_0 + Δt].
        Estimate the spike time by Eq. (6).
        Calculate the threshold values by Eq. (9).
        Obtain the reset values by Eq. (10).
        Stop evolving V_i, m_i, h_i, n_i during the stiff period.
        Evolve the ith neuron using a suitably small time step as illustrated in Fig. 3c.
    end
end
Update the conductance of each neuron by Eq. (8).
```

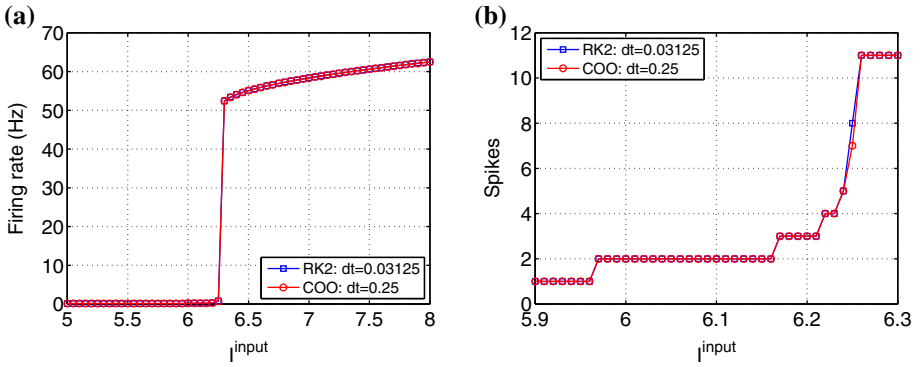

Fig. 4 **a** The firing rate of an individual HH neuron as a function of constant input I^{input} (μA cm^{-2}) for a total run time of 10 s which is sufficiently long for the neuron to converge to the stationary state. **b** The number of spikes during the transient period (200 ms) with initial voltage $V = -65$ mV. The blue squares and red circles in **a** and **b** indicate the results using RK2 method with $\Delta t = 2^{-5} = 0.03125$ ms and the COO method with a much larger time step $\Delta t = 0.25$ ms, respectively (Color figure online)

5 Numerical Results

5.1 Transient Dynamics and Hopf Bifurcation Point of an Individual HH Neuron

In this section, we first demonstrate the performance of the COO method for an individual HH neuron. As described above, we build the data set using dynamic information that covers almost all possible combinations of threshold values observed in general spiking events. Thus, the COO method is expected to capture the transient dynamics and the Hopf bifurcation point of an individual HH neuron with a relatively small error. A single neuron driven by constant input can fire regularly and periodically [11,21] only when the input current is greater than a critical value, $I^{\text{input}} \approx 6.2$ μA cm^{-2} (type II behavior). The neuron has a sudden jump near this critical value from no firing to regular firing at a nonzero firing rate because of a subcritical Hopf bifurcation [9,21,36], as demonstrated in Fig. 4a. Below the critical value, some spikes may appear before the neuron converges to a stationary quiescent state. The number of spikes during this transient period depends on how close the constant input is to the critical value as shown in Fig. 4b. Our offline data set built with dynamic information is sufficiently accurate such that our COO method can well capture the transient dynamics and the Hopf bifurcation point even with a time step that is 8 times greater than that used in the RK2 method. Note that we use a maximum time step of about $\Delta t = 0.03125$ ms for the RK2 method under the constraint of the dynamic stability of the numerical scheme as shown later in Table 1. We should point out that when the constant input is sufficiently large (over 155 μA cm^{-2}), the neuron will converge to a stationary quiescent state. However, such large input is often beyond the physiological range and is not discussed in this work.

Since our offline data set and the second data set of voltage traces are very accurate, the COO method can also achieve accurate trajectories of membrane potentials, especially the shapes of action potentials. To show this, we drive an individual HH neuron by a Poisson spike train and evolve it for a long time (over 10 s). As shown in Fig. 5, the COO method with a large time step can achieve accurate trajectories of the membrane potential compared with the RK2 method using a sufficiently small time step $\Delta t = 1 \times 10^{-6}$ ms.

Table 1 Simulation of the all-to-all connected excitatory network with $S = 0.3$ mS cm^{-2} for a total run time $T = 10$ s

	RK2			COO		
Δt (ms)	CPU (s)	Largest λ	Relative error (%)	CPU (s)	Largest λ	Relative error (%)
2^{-7}	91.61	-0.050	0	86.95	-0.050	0
2^{-6}	45.92	-0.050	0	44.11	-0.050	0
2^{-5}	22.77	-0.050	0	21.54	-0.049	0
2^{-4}	***	***	***	10.96	-0.049	0
2^{-3}	***	***	***	5.49	-0.049	0.083
2^{-2}	***	***	***	2.83	-0.048	0.33
0.314	***	***	***	2.16	-0.048	0.50

The relative error is in the mean firing rate between each method and the RK2 method using a sufficiently small time step $\Delta t = 1 \times 10^{-6}$ ms. Asterisks for the RK2 method indicate overflow errors

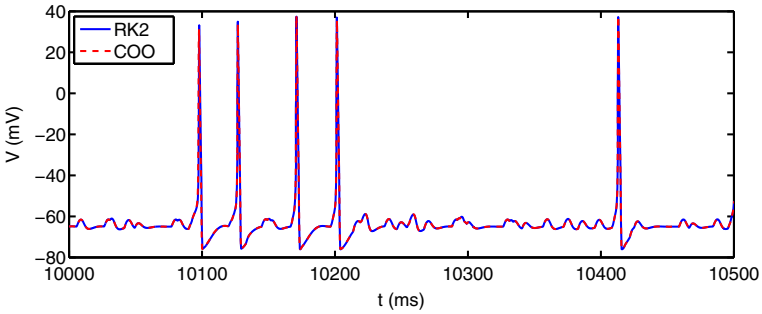

Fig. 5 A comparison of the trajectory of V obtained from the RK2 method with $\Delta t = 1 \times 10^{-6}$ ms (blue solid curve) and the COO method with linear interpolation and $\Delta t = 0.25$ ms (red dashed curve) (Color figure online)

5.2 Lyapunov Exponent

We now demonstrate the performance of the COO method for an all-to-all connected network of $N = 128$ excitatory neurons. For simplicity, we fix the feedforward input parameters at $f = 0.1$ mS cm^{-2}, $\nu = 100$ Hz, and keep the coupling strength S as the only remaining parameter that can be varied. The concepts described in this section can be easily extended to other types of HH networks and other dynamical regimes.

We first study the chaotic dynamical property of the HH network by computing the largest Lyapunov exponent, one of the most important tools used to characterize dynamical stability [28]. Given a sufficiently small initial perturbation in the state of each neuron (V, m, h, or n), the Lyapunov exponents can measure the average rate of divergence or convergence of the reference (unperturbed) and the perturbed trajectories of the HH network [29,30,44]. A positive largest Lyapunov exponent signifies a chaotic system, while a negative one signifies a non-chaotic system.

To calculate the largest Lyapunov exponent, we denote $\mathbf{X} = [X_1, X_2, ..., X_N]$ to represent all of the variables of all of the neurons in the HH model, where X_i indicates the state (V_i, m_i, h_i, n_i, and conductance G_i) of the ith neuron, as previously described in Eq. (5) for one neuron. Let the reference and perturbed trajectories be denoted by $\mathbf{X}(t)$ and $\tilde{\mathbf{X}}(t)$, respectively. Then, the largest Lyapunov exponent is defined by

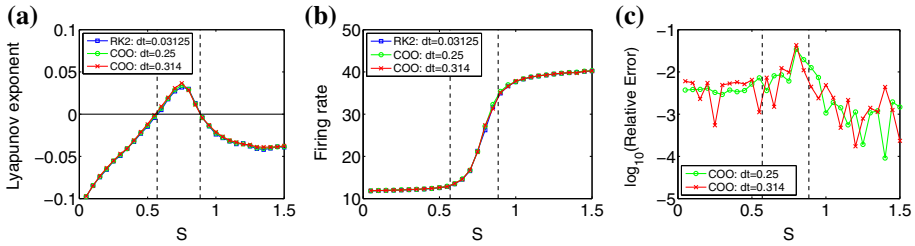

Fig. 6 **a** The largest Lyapunov exponent of the HH network versus the coupling strength S. **b** Mean firing rate of the network versus the coupling strength S. **c** The relative error in the mean firing rate between the COO method and the RK2 method versus the coupling strength S. The blue squares, green circles, and red crosses represent the results using the RK2 method with $\Delta t = 2^{-5} = 0.03125$ ms, the COO method with a time step of $\Delta t = 0.25$ ms and with the maximum time step of $\Delta t = 0.314$ ms, respectively. The vertical dashed lines indicate the values of S in which the network is in a chaotic regime. The total run time is 60 s to obtain convergent statistical properties of the HH network (Color figure online)

$$\lambda = \lim_{t\to\infty}\lim_{\epsilon\to 0}\frac{1}{t}\ln\frac{||\tilde{\mathbf{X}}(t)-\mathbf{X}(t)||}{||\epsilon||} \tag{11}$$

where ϵ is the initial separation between the two trajectories. However, we cannot use Eq. (11) to compute the largest Lyapunov exponent directly, because the ratio $||\tilde{\mathbf{X}}(t)-\mathbf{X}(t)||/||\epsilon||$ is unbounded as $t\to\infty$, arising from the exponential decay or growth rates of the principal axes of the ellipsoid [29], and it will make the numerical computation ill-conditioned. The standard algorithm to compute the largest Lyapunov exponent for continuous systems can be found in Refs. [30,47,50]. Since, for the COO method, the values of V, m, h, n are not numerically evolved during the stiff period, the standard algorithms to compute the largest Lyapunov exponent will not work. To circumvent this difficulty, we compute the largest Lyapunov exponent using the algorithm in Ref. [49] proposed for discontinuous systems.

We compute the largest Lyapunov exponent as a function of the coupling strength, S, ranging from 0 to 1.5 mS cm^{-2} using the RK2 and COO methods. The total run time T is 60 s, which is sufficiently long to yield convergence of the largest Lyapunov exponent. As shown in Fig. 6a, the COO method with large time steps can obtain accurate estimations for the largest Lyapunov exponent compared with the RK2 method with a small time step. As shown in Fig. 6b, we compute the mean firing rate, denoted by R, and show that the COO method, with two different time steps, can accurately reproduce the average firing rate as observed in the network using the RK2 method with a small time step. In Fig. 6c, we compute the relative error in the mean firing rate between the RK2 method and the COO method, defined as

$$E_{\mathrm{R}} = |R_{\mathrm{COO}} - R_{\mathrm{RK2}}|/R_{\mathrm{RK2}}, \tag{12}$$

where the subscript "COO" and "RK2" represent the COO and RK2 methods, respectively. The COO method using a large time step of $\Delta t = 0.25$ ms and a maximum time step of $\Delta t = 0.314$ ms can both achieve at least 2 digits of accuracy for the non-chaotic regimes.

The largest Lyapunov exponent determines that the system is chaotic for $0.55 \lesssim S \lesssim 0.9$ mS cm^{-2}, as shown by the vertical dashed lines in all three plots of Fig. 6, and that the system is non-chaotic for $0 \le S \lesssim 0.55$ and $0.9 \lesssim S \le 1.5$ mS cm^{-2}. Thus, there are three typical dynamical regimes that occur in the HH model network: a low-firing, asynchronous regime as shown in Fig. 7a, a medium-firing, chaotic regime as shown in Fig. 7b, and a high-firing, synchronous regime as shown in Fig. 7c. We choose three coupling strengths within each

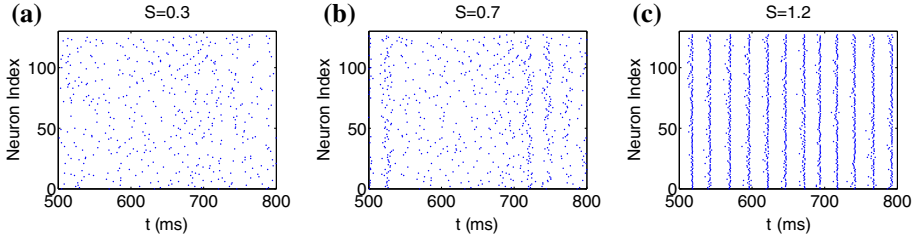

Fig. 7 Raster plots of firing events in three typical dynamical regimes with coupling strength **a** S = 0.3, **b** S = 0.7, **c** S = 1.2 mS cm^{-2}

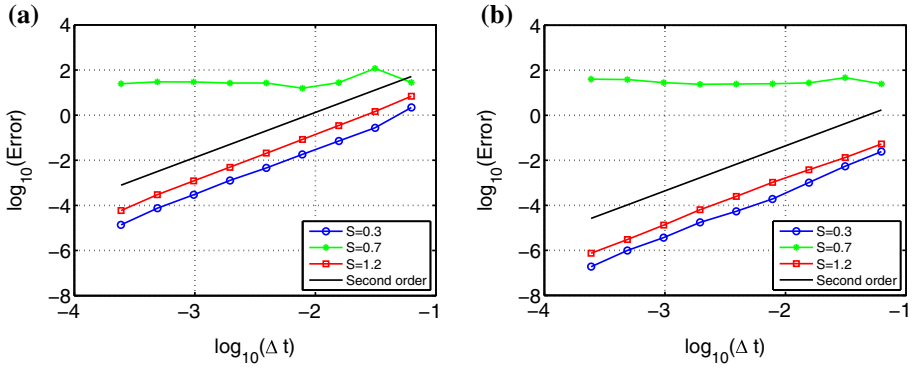

Fig. 8 Convergence tests for the **a** RK2 method and **b** COO method. The convergence tests are performed by evolving the HH network for a total run time of $T = 2000$ ms. We show the results for the coupling strength $S = 0.3$ (blue), 0.7 (green), and 1.2 (red) mS cm^{-2}, respectively. The black line indicates a scaling with exponent 2 (Color figure online)

regime, $S = 0.3$, 0.7, and 1.2 mS cm^{-2}, to represent these three typical dynamical regimes, respectively, and discuss the issue of numerical convergence in the following simulations.

5.3 Convergence Tests

In this section, we verify the second-order numerical accuracy of the RK2 and the COO methods by performing convergence tests. We use a sufficiently small time step of $\Delta t = 1 \times 10^{-6}$ ms to evolve the HH neural network by the RK2 (COO) method to obtain a high-precision solution $\mathbf{X}^{\text{high}}$ at the end time of $T = 2000$ ms for the RK2 (COO) method. To calculate the convergence and the order of numerical accuracy, we also compute the solution $\mathbf{X}^{(\Delta t)}$ using a variety of time steps $\Delta t = 2^{-4}, 2^{-5}, ..., 2^{-12}$ ms by the RK2 and COO methods. The numerical error is measured in the L^2-norm as follows

$$E = ||\mathbf{X}^{(\Delta t)} - \mathbf{X}^{\text{high}}||. \tag{13}$$

As shown in Fig. 8, both the RK2 and the COO methods can achieve second-order numerical accuracy for the non-chaotic regimes $S = 0.3$ and 1.2 mS cm^{-2}. As expected, for the chaotic regime in which $S = 0.7$ mS cm^{-2}, a convergent numerical solution can not be achieved.

5.4 Computational Efficiency

We next demonstrate the computational efficiency of the COO method. Table 1 shows some timing results obtained on Visual Studio using an Intel i7 2.6 GHz processor. The RK2 method can achieve high accuracy for a numerically stable time step, though the allowed maximum time step size is relatively small. The COO method can achieve accurate largest Lyapunov exponents and mean firing rates using various values of time steps compared with the RK2 method using a very small time step. Thus, the efficiency of the COO method relies on the number of evolved time steps given that it can obtain accurate statistical properties of HH neurons.

A straight forward way to measure the efficiency is to compare the real simulation time that it takes for a computer to evolve the network to a common final run time for each method. We denote this efficiency by κ_{time} and define it as the ratio of time for the RK2 method to the time for the COO method as follows

$$\kappa_{\text{time}} = \hat{T}_{\text{RK2}}/\hat{T}_{\text{COO}}, \tag{14}$$

where $\hat{T}_{\text{RK2}}$ and $\hat{T}_{\text{COO}}$ represent the real simulation time that the RK2 and COO methods take to evolve the network, respectively.

To analytically estimate the efficiency of the COO method, we compute efficiency using another measure, the number of calls to the numerical scheme. Since both the RK2 and COO methods are based on the standard RK2 scheme, we can compare the number of times the standard RK2 scheme is called by each method. For the RK2 method, the call number per neuron is $T/\Delta t_{\text{RK2}}$, where Δt_{RK2} is the time step used in the RK2 method and T is the total run time. For the COO method, when a neuron fires a spike, we stop evolving its V, m, h, n variables for the following stiff period, T^{stiff}, and then, once we're out of the stiff period, we take a small time step to evolve the system back to the original time-stepping scheme (recall Fig. 3c). To approximate the call number for the COO method we take the total number of calls to the RK2 scheme for each neuron, $T/\Delta t_{\text{COO}}$, where Δt_{COO} is the time step used in the COO method, and subtract the average time that each neuron spends in the stiff period (since there are no calls to the RK2 scheme during this time), $T \cdot R \cdot T^{\text{stiff}}$, where R is the average firing rate of the network. Therefore, the approximate call number per neuron for the COO method is $(T - T \cdot R \cdot T^{\text{stiff}})/\Delta t_{\text{COO}} + T \cdot R$, where $T \cdot R$ represents the call to the RK2 scheme using a small time step when the neuron leaves the stiff period (recall Fig. 3c). Then the efficiency of the COO method estimated by call number, denoted by κ_{num}, is defined as the ratio of the call number of the RK2 method to the call number of the COO method as follows

$$\kappa_{\text{num}} \approx \frac{T/\Delta t_{\text{RK2}}}{(T - T \cdot R \cdot T^{\text{stiff}})/\Delta t_{\text{COO}} + T \cdot R} = \frac{\Delta t_{\text{COO}}}{(1 - T^{\text{stiff}} \cdot R + \Delta t_{\text{COO}} \cdot R)\Delta t_{\text{RK2}}}. \tag{15}$$

Note that the linear interpolation used in the COO method may take a certain small amount of time while its contribution to the time cost is not included in κ_{num}. Thus, κ_{num} is a bit overestimated.

In Fig. 9a, we show the efficiency of the COO method as measured by the ratio of the physical simulation time it takes to run the simulation between the RK2 method and the COO method, κ_{time}, and the ratio of the number of times the RK2 scheme is called in the code between the RK2 method and the COO method, κ_{num}. We also use the analytical expression in Eq. (15) to calculate an approximation to the call number and show that indeed this equation provides a good estimation of efficiency as measured by the actual call number in the code.

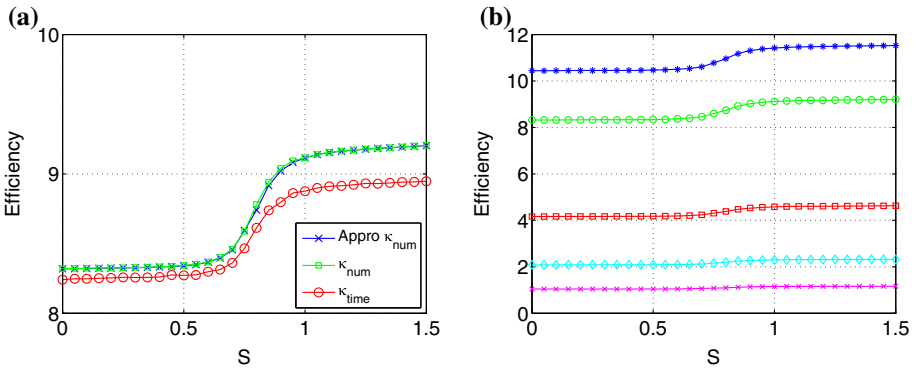

Fig. 9 **a** Efficiency of the COO method measured by time cost, κ_{time}, and call number of standard RK2 scheme, κ_{num} for the all-to-all connected excitatory network. The blue crosses, green squares, and red circles indicate efficiency measured by Eq. (15), the actual call number in the code, and the actual time cost in the code, respectively. The time steps are $\Delta t = 2^{-5} = 0.03125$ ms for the RK2 method and $\Delta t = 0.25$ ms for our method. **b** Efficiency of our COO method measured by the call number with time steps $\Delta t = 0.314, 0.25, 0.125, 0.0625, 0.03125$ ms from top to bottom. Total run time is 50 s (Color figure online)

Notice that the efficiency as measured by the call number is slightly higher than the efficiency as measured by time cost. This is because when a neuron fires a spike, we call the offline data set and evolve the conductance G during the stiff period, which takes certain extra time, but is not included in the efficiency measured by the call number. When the mean firing rate is high, and the call to the data set increases, this extra time is no longer negligible. Even so, these two kinds of efficiency still show good agreement and the COO method is much more efficient than the standard RK2 method as evidenced by the value of the ratio being much larger than one.

Figure 9b shows the efficiency of the COO method as measured by Eq. (15) for several different time steps. The COO method can stably achieve high efficiency with a maximum over 10 times of speedup for a maximum time step of $\Delta t = 0.314$ ms. Note that this high efficiency is independent of the size of network, structural connectivity, dynamical regimes or networks with both excitatory and inhibitory neurons, since Eq. (15) mainly relies on the ratio of time steps between the RK2 and the COO methods.

5.5 Network Firing Patterns

In this section, we show that the COO method can accurately capture different firing patterns elicited by networks of neurons. The information that is communicated by a neural network is realized through the firing patterns of the network [17,26,43]. These firing patterns are essential in understanding the function of the brain [12,23,38,39] and can be used to infer the structural connectivity of neural networks [31,33,39,48,51]. To verify that the COO method can accurately capture the distribution of firing patterns, we randomly choose 10 neurons from the all-to-all connected network of $N = 128$ excitatory neurons and record their spike times. Then, we transform those spike times into a binary time series with a time bin of 10 ms, where the binary data is 1 if there is a spike during the time bin and 0 otherwise. In this way, the 10 neurons can comprise a 10-dimensional binary vector with a total of $2^{10} = 1024$ different kinds of combinations of firing patterns. Then we evolve the network for a sufficiently long run time and compute the probability of each pattern. This probability

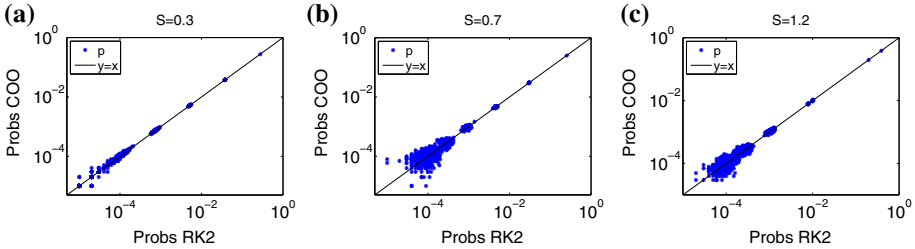

Fig. 10 Comparison of the distribution of the firing patterns generated by the network simulated using the RK2 and COO methods with coupling strength **a** 0.3, **b** 0.7, and **c** 1.2 mS cm^{-2}. Time steps are the same as in Fig. 4. Total run time is 1000 s to obtain convergent probability distributions (Color figure online)

is approximated by counting the number of observations of each pattern over the total number of patterns observed in the simulation.

Figure 10 compares the probability of obtaining each pattern using the standard RK2 method with a very small time step with the probability of obtaining each pattern using the COO method with a large time step. The star in Fig. 10 indicates the probability of finding the same pattern in the networks generated using each method. If the star lies on the diagonal line, then the COO method can capture the exact the same probability of the corresponding pattern as the RK2 method.

To quantitatively characterize the closeness of the above two probability distributions, we do chi-square two sample tests for the comparison of the distribution of the firing patterns in the networks simulated using the RK2 and COO methods. We assume that the null hypothesis is that the two groups of samples come from a common distribution, then we compute the test statistics for networks with coupling strengths $S = 0.3$, 0.7, and 1.2 mS cm^{-2} and show that all of these networks have a p-value greater than 0.999. Therefore, the distributions from the RK2 and COO methods cannot be distinguished in the statistical sense, i.e., the COO method with a large time step yields a network with almost the same firing patterns as the network using the RK2 method with a very small time step. We emphasize that even in the chaotic regime, our COO method with a large time step can still capture accurate mean firing rates and the distribution of firing patterns as shown in Figs. 6b and 10b, respectively.

5.6 Error of Combined Offline–Online Method

In this section, we demonstrate the error of our COO method. There are three kinds of error that arise in the COO method: error from the numerical approximation of the solution to the ordinary differential equations, error from the interpolation to use the data set, and error from the assumption that we keep I^{input} constant throughout the stiff period. The first error is simply $O(\Delta t^2)$ since the COO method is based on the RK2 scheme. The other two kinds of error come from using the data set. For simplicity, we consider the error in the voltage for one single neuron to illustrate the magnitude of all variables

$$\Delta V = |V_{\text{COO}} - V_{\text{RK2}}|, \tag{16}$$

where $|\cdot|$ takes the absolute value, the subscript "RK2" indicates the high-resolution solution computed by the RK2 method with a sufficiently small time step $\Delta t = 1 \times 10^{-6}$ ms and "COO" indicates the solution from the COO method.

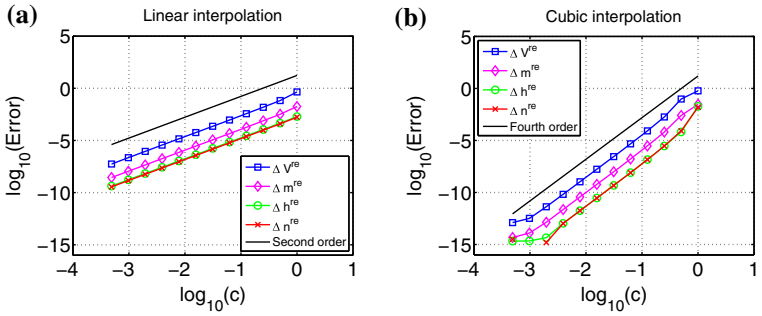

Fig. 11 The error of the reset values for a single HH neuron driven by constant input. The error resulting from **a** linear interpolation and **b** cubic interpolation in the COO method. The blue squares, magenta diamonds, green circles, red crosses indicate the error of the reset values V^{re}, m^{re}, h^{re}, n^{re}, respectively. The black line in **a** and **b** indicates a second-order and fourth-order accuracy, respectively. The time step used in these simulations is $\Delta t = 1 \times 10^{-6}$ ms (Color figure online)

The error that results from interpolation to find the reset values can be well described by the error of the reset value ΔV^{re}, comparing the reset value generated by the data set and the corresponding value computed using the RK2 method. To fairly compare the result from the RK2 method and COO method, we use a constant input I^{input} to drive a single HH neuron in both cases. Then, the error from linear interpolation is $\Delta V^{\text{re}} = O(\Delta I^2 + \Delta m^2 + \Delta h^2 + \Delta n^2)$. If we first take the sample interval $\Delta I = 5\ \mu\text{A cm}^{-2}$, $\Delta m = \Delta h = \Delta n = 0.04$ and keep halving them as $c\Delta I, c\Delta m, c\Delta h, c\Delta n$ for $c = 2^0, 2^{-1}, ..., 2^{-11}$, then it is straight forward that $\Delta z^{\text{re}} = O(c^2)$ for $z = V, m, h$, and n as shown in Fig. 11a. In the same way, a cubic interpolation yields $\Delta z^{\text{re}} = O(c^4)$ for $z = V, m, h$, and n as shown in Fig. 11b.

Lastly, we consider the error due to the assumption of constant input during the stiff period. To do this, we drive a single HH neuron by a Poisson spike train and compare the difference in the voltage, ΔV. The highly precise voltage trace is computed using the RK2 method with a sufficiently small time step $\Delta t = 1 \times 10^{-6}$ ms. To illustrate the error due to the assumption of constant input, we also evolve this neuron using the RK2 method with $\Delta t = 1 \times 10^{-6}$ ms; But, when the neuron fires an action potential, we set its input as constant $I^{\text{input}} = I^{\text{th}}$ for the following T^{stiff} ms. As shown in Fig. 12, once the neuron fires an action potential, the error due to the assumption of constant input is significantly greater than the error from both the time step and the order of interpolation to use the data set.

Therefore, the major source of error comes from the assumption of constant input during the stiff period. It is difficult to reduce this error since the online feedforward and synaptic spike input is impossible to know in advance. Hence, we use relatively large sample intervals to reduce the size of the offline data set, choose a linear interpolation to use the data set, and use a second-order numerical time-stepping scheme to save computational cost. We also find that the error will not accumulate with the call number of the data set, as shown in Fig. 12, by the quick decay in error after each spike (call to data set).

6 Conclusion

In conclusion, we have developed a combined offline–online method that accurately and efficiently evolves a network of HH equations and reproduces the firing patterns that are typical of three different dynamical regimes. Through use of a pre-computed (offline) high-

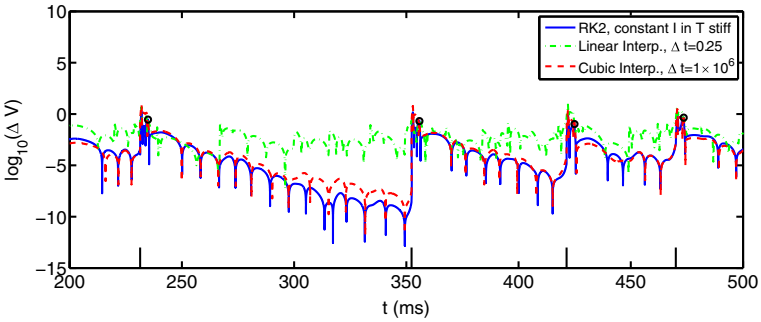

Fig. 12 The error in voltage, ΔV, for a single HH neuron driven by a Poisson spike train. The highly precise solution is obtained using the RK2 method with $\Delta t = 1 \times 10^{-6}$ ms. The blue solid line is the error due to the constant input assumption during the stiff period using the RK2 method with $\Delta t = 1 \times 10^{-6}$ ms. The green dash-dotted line is the error due to linear interpolation to use the COO method with $\Delta t = 0.25$ ms, while the red dashed line is the error due to cubic interpolation to use the COO method with $\Delta t = 1 \times 10^{-6}$ ms. The short black lines on the horizontal axis indicate the spike times of the neuron and the black circles indicate the error in voltage at the end of stiff period (spike time plus T^{stiff} ms), both of which are obtained from the highly precise solution (Color figure online)

resolution data set, we save computational cost by skipping the numerical evolution of the HH equations during the stiff period (time during which an action potential occurs) and using a large time step to evolve the HH equations outside of the stiff period. We point out that our COO method can be viewed as a numerical reduction of the HH neuron to an integrate-and-fire (I&F) neuron [25,34,40]. However, our COO method still keeps the original HH dynamics such as evolution of gating variables and detailed action-potential shapes which can be well recovered if needed, while these important properties are generally lost in the I&F neuron. Our method can capture accurate transient dynamics and the Hopf bifurcation point for an individual HH neuron, as well as chaotic attractors, mean firing rates, and firing patterns of HH neural networks. Besides, our COO method can also capture accurate action-potential shapes by using a data set with pre-computed voltage traces, if one has interest in recovering action-potential shapes. In addition, our COO method can be easily extended to networks with different properties, e.g., large-scaled or excitatory and inhibitory networks, while still maintaining high accuracy and efficiency.

COO method with data set of small sizes We emphasize that the main source of error in the COO method stems from the assumption of constant input current during the stiff period in building the offline data set. Since it is impossible to have prior knowledge of the future feedforward and synaptic spike times before evolving, it is difficult to refine this assumption to reduce the error of data set. Therefore, it is suitable to use relatively large sample intervals to build a small data set, use a linear interpolation to use the data set, and evolve with a second-order numerical scheme. As shown in Fig. 13, the COO method with a coarse data set built using very few sample numbers $N_I = 3$, $N_m = 3$, $N_h = 3$, $N_n = 3$ can accurately capture the statistical properties of HH networks, such as the largest Lyapunov exponents and mean firing rates for both the chaotic and non-chaotic dynamical regimes. This is because the trajectory of an HH neuron is highly contracted at the time when the neuron is out of the spike period and the reset values have very small fluctuations as shown in Fig. 14. The COO method still has relatively high accuracy when using the coarse data set to perform interpolations for dynamical variables. However, the recovery of the action potential shape

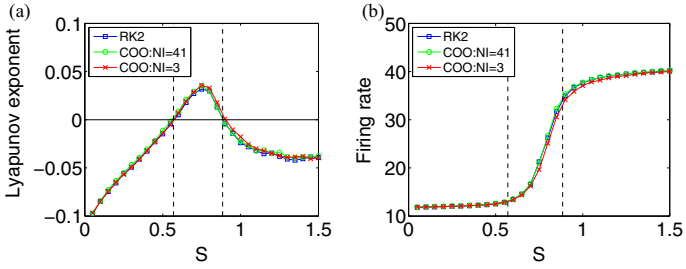

Fig. 13 Numerical performance of the COO method using an offline data set of different sizes. **a** The largest Lyapunov exponent of the HH network versus the coupling strength S. **b** Mean firing rate of the network versus the coupling strength S. The blue squares, green circles, and red crosses represent the results using the RK2 method with $\Delta t = 2^{-5} = 0.03125$ ms, the COO method using offline data set built with sample numbers $N_I = 41$, $N_m = 31$, $N_h = 41$, $N_n = 31$ ($\Delta I = 1.25\ \mu\text{A cm}^{-2}$, $\Delta m = \Delta h = \Delta n = 0.01$) and that with sample numbers $N_I = 3$, $N_m = 3$, $N_h = 3$, $N_n = 3$ ($\Delta I = 20\ \mu\text{A cm}^{-2}$, $\Delta m = \Delta h = \Delta n = 0.16$), respectively. The vertical dashed lines indicate the values of S for which the network is in a chaotic regime. The time step for the COO method is $\Delta t = 0.25$ ms and the total run time is 60 s (Color figure online)

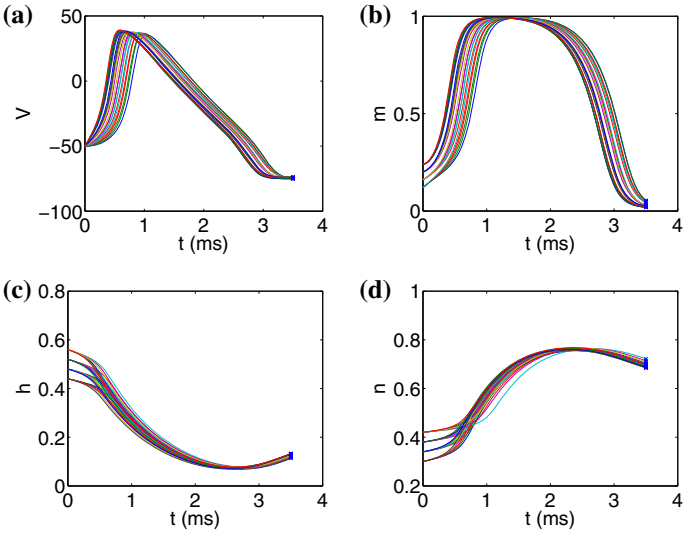

Fig. 14 Typical trajectories of **a** V, **b** m, **c** h, and **d** n during the spike period

during the spike period requires a dense data set since one can clearly observe that the voltage traces shown in Fig. 14a are scattered.

The interval of the stiff period in the COO method When building the offline data set, it is important to choose a proper time interval for the stiff period (duration of action potential). The stiff period should cover all stiff parts of the dynamical variables and be determined by the dynamics of the HH model. As shown in Fig. 14a, the choice of 3.5 ms as the stiff period is appropriate. However, if the parameters in the HH model are changed, the interval of the stiff period should also be changed to account for different dynamics of the HH model.

COO method for gap-junctionally coupled networks We should point out that accurate action-potential shapes are important in some situations, e.g., for the electrically coupled neurons. Such coupling exists among local inhibitory neurons and occurs through gap junctions

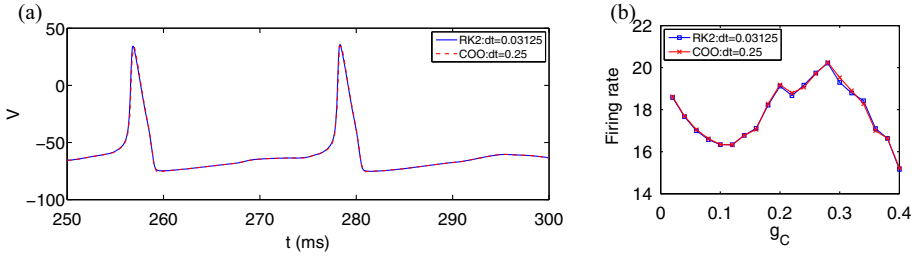

Fig. 15 Consider an all-to-all connected HH network of 80 excitatory and 20 inhibitory neurons with electrical couplings. **a** A comparison of the voltage trajectory of a randomly chosen inhibitory neuron. **b** Mean firing rate of the network versus the electrical coupling strength g_C. The value range of g_C is chosen from experimental data [10,46] and numerical simulation works [6,22]. The blue curve indicates the results obtained from the RK2 method with $\Delta t = 0.03125$ ms, while the red curve indicates the results obtained from the COO method with data set built using sample numbers $N_I = 21$, $N_m = 16$, $N_h = 21$, $N_n = 16$ ($\Delta I = 2.5$ μA cm^{-2}, $\Delta m = \Delta h = \Delta n = 0.02$) and $\Delta t = 0.25$ ms. The parameters are set as $V_G^E = 0$ mV, $V_G^I = -80$ mV, $\sigma_r^E = 0.5$ ms, $\sigma_d^E = 3.0$ ms, $\sigma_r^I = 0.5$ ms, $\sigma_d^I = 7.0$ ms [8] and $f = 0.1$ mS cm^{-2}, $\nu = 150$ Hz, $S = 0.5$ mS cm^{-2}. In **a**, $g_C = 0.4$ mS cm^{-2} (Color figure online)

[2,4,35], where voltage traces (action-potential shapes) play an essential role in generating synchronous activity among neurons [6,22,46]. To clarify the high efficiency and accuracy of our COO method for the electrically coupled networks, we consider an all-to-all connected HH network of 80 excitatory and 20 inhibitory neurons with electrical couplings among inhibitory neurons. The input current I_i^{input} is given by $I_i^{\text{input}} = I_i^E + I_i^I + I_i^{EC}$ with

$$I_i^E = -G_i^E(t)(V_i - V_G^E), \quad I_i^I = -G_i^I(t)(V_i - V_G^I), \quad I_i^{EC} = -\frac{g_C}{M}\sum_{j=1}^{M}(V_i - V_j), \tag{17}$$

where G_i^E and G_i^I are excitatory and inhibitory conductances, respectively, V_G^E and V_G^I are the corresponding reversal potentials, I_i^{EC} is the electrical current, g_C is the strength of electrical coupling, and M is the total number of inhibitory neurons. The conductances are defined as

$$G_i^E(t) = f\sum_l H(\sigma_d^E, \sigma_r^E, t - s_{il}) + \sum_j S_{ij}^E \sum_l H(\sigma_d^E, \sigma_r^E, t - \tau_{jl}), \tag{18}$$

$$G_i^I(t) = \sum_j S_{ij}^I \sum_l H(\sigma_d^I, \sigma_r^I, t - \tau_{jl}), \tag{19}$$

where H is the *alpha* function given in Eq. (4). For simplicity, we take $S_{ij}^Q = S/N, Q = E, I$.

In the COO method, when an inhibitory HH neuron fires an action potential, the action-potential shape is recovered using the data set of pre-computed voltage traces as shown in Fig. 15a, and the recovered voltage trace is used to compute its postsynaptic neuron's electrical current I^{EC}. When taking a relatively large value of electrical coupling $g_C = 0.4$ mS cm^{-2} [6,22], the electrical current of the HH neuron is in the range of $[-40.1, 31.8]$ μA cm^{-2}. This electrical current is much smaller than the intrinsic current during the stiff period as shown in Fig. 1b. Therefore, it will not further introduce large stiffness in the computation of HH equations. Consequently, our COO method still allows using a large time step to evolve the HH equations. For instance, the COO method with a large time step $\Delta t = 0.25$ ms can

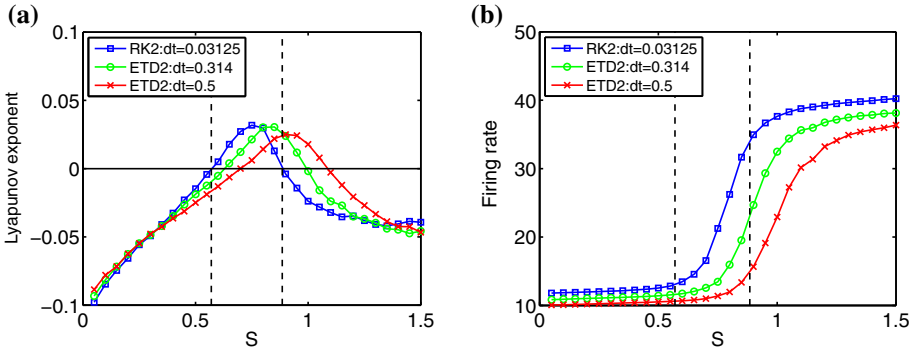

Fig. 16 Numerical performance of the ETD2 method. **a** The largest Lyapunov exponent of the HH network versus the coupling strength S. **b** Mean firing rate of the network versus the coupling strength S. The blue squares, green circles, and red crosses represent the results using the RK2 method with $\Delta t = 2^{-5} = 0.03125$ ms, the ETD2 method with time step $\Delta t = 0.314$ ms and $\Delta t = 0.5$ ms, respectively. The vertical dashed lines indicate the values of S for which the network is in a chaotic regime. The HH network and parameters are the same as those used in Fig. 6 (Color figure online)

achieve accurate results for both the shape of action potential and the mean firing rate as shown in Fig. 15.

Comparison with library-based method We next discuss about the difference between our method and the library-based method [42]. The previous work, i.e., the library-based method, also allows one to evolve HH dynamics with a large time step, by establishing a library data set in the spike period. The choice of reset voltage value V^{re} when the neuron is just out of the spike period is determined by the voltage trace when the neuron is in the periodic firing regime by receiving constant current input I^{th}. Meanwhile, note that the gating variables m, h, and n do not explicitly depend on one another. Therefore, based on the voltage trace in the periodic firing regime, one can evolve the dynamics of gating variables separately to obtain their reset values, i.e., m^{re}, h^{re}, n^{re}.

However, there is an implicit dependence among the gating variables m, h, and n and the voltage trace obtained from the periodic firing regime may not be sufficiently accurate to well capture the voltage dynamics when the neuron is driven by a general current input. In contrast, our offline data set is built by taking into account the above factors. As a consequence, the library-based method cannot capture the transient dynamics, Hopf bifurcation point, and accurate shape of action potentials [42], while our method can.

Comparison with ETD method Another approach to efficiently evolve the HH neural network is the ETD method [3,5]. For instance, a second-order ETD (ETD2) method is proposed in Ref. [3] by first linearly approximating the HH equations and then analytically solving the approximated HH equations. This method is proven to be unconditionally stable for the HH system [3]. We demonstrate that the ETD2 method will be inaccurate when using a large time step. As shown in Fig. 16, the ETD2 method can indeed use much larger time steps, e.g., $\Delta t = 0.314$ ms or $\Delta t = 0.5$ ms. However, the obtained results are not consistent with those from the RK2 method using a very small time step ($\Delta t = 0.03125$ ms). In contrast, by using the large time step $\Delta t = 0.314$ ms, the numerical results of our COO method agree well with the results of the RK2 method using a very small time step ($\Delta t = 0.03125$ ms) as shown in Fig. 6.

Comparison with implicit method We point out that standard implicit methods, e.g., the implicit Euler method, may not be effective for the simulation of HH neural networks. The

implicit method requires solving the entire nonlinear HH system using iterative methods such as fixed point iteration or Newton's method. It requires extra computation to achieve convergence of iteration in each time step. Moreover, as illustrated in Ref. [3], the time step allowed in the implicit Euler method for HH network is on the order of 0.01 ms which is much smaller than the time step allowed in the explicit RK2 method. Therefore, an implicit method is inefficient for the simulation of HH neural networks.

Finally, we emphasize the versatility of the COO method. We point out that the data set can be easily rebuilt for use with a variety of HH models, including those with different parameter values or other HH-type neurons (e.g., neurons with bursting and adaptation behavior) [32]. We have provided a thorough and comprehensive manual for constructing and using the data set that can be reproduced in each of these cases. The COO method can still retain many properties of these HH-type neurons and can use large time steps to achieve high computational efficiency. Moreover, the COO method can be easily implemented in parallel, where each neuron can be independently advanced and recalibrated, as illustrated in Algorithm 2. The COO method is easy to use, and it is computationally efficient and accurate, making it an excellent option for simulating large networks of HH neurons.

Acknowledgements This work was supported by National Key R&D Program of China (2019YFA0709503), NSFC-11671259, NSFC-11722107, SJTU-UM Collaborative Research Program, and the Student Innovation Center at Shanghai Jiao Tong University (D.Z.); the NSF Mathematical Sciences PostDoctoral Research Fellowship (MSPRF) DMS-1703761 (J.C.). We dedicate this paper to our late professor David Cai.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aihara, K.: Chaotic oscillations and bifurcations in squid giant axons. In: Chaos, pp. 257–269 (1986)
2. Beierlein, M., Gibson, J.R., Connors, B.W.: A network of electrically coupled interneurons drives synchronized inhibition in neocortex. Nat. Neurosci. **3**(9), 904–910 (2000)
3. Börgers, C., Nectow, A.R.: Exponential time differencing for Hodgkin–Huxley-like ODEs. SIAM J. Sci. Comput. **35**(3), B623–B643 (2013)
4. Connors, B.W., Long, M.A.: Electrical synapses in the mammalian brain. Annu. Rev. Neurosci. **27**, 393–418 (2004)
5. Cox, S.M., Matthews, P.C.: Exponential time differencing for stiff systems. J. Comput. Phys. **176**(2), 430–455 (2002)
6. Crodelle, J., Zhou, D., Kovacic, G., Cai, D.: A role for electrotonic coupling between cortical pyramidal cells. Front. Comput. Neurosci. **13**, 33 (2019)
7. Dayan, P., Abbott, L.: Theoretical neuroscience: computational and mathematical modeling of neural systems. J. Cogn. Neurosci. **15**(1), 154–155 (2003)
8. Dayan, P., Abbott, L.F.: Theoretical Neuroscience, vol. 806. MIT Press, Cambridge (2001)
9. Ding, L., Hou, C.: Stabilizing control of hopf bifurcation in the Hodgkin–Huxley model via washout filter with linear control term. Nonlinear Dyn. **60**(1–2), 131–139 (2010)
10. Galarreta, M., Hestrin, S.: A network of fast-spiking cells in the neocortex connected by electrical synapses. Nature **402**(6757), 72–75 (1999)
11. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge (2002)
12. Gu, H., Pan, B.: A four-dimensional neuronal model to describe the complex nonlinear dynamics observed in the firing patterns of a sciatic nerve chronic constriction injury model. Nonlinear Dyn. **81**(4), 2107–2126 (2015)
13. Guckenheimer, J., Oliva, R.A.: Chaos in the Hodgkin–Huxley model. SIAM J. Appl. Dyn. Syst. **1**(1), 105–114 (2002)

14. Hansel, D., Mato, G., Meunier, C., Neltner, L.: On numerical simulations of integrate-and-fire neural networks. Neural Comput. **10**(2), 467–483 (1998)
15. Hansel, D., Sompolinsky, H.: Chaos and synchrony in a model of a hypercolumn in visual cortex. J. Comput. Neurosci. **3**(1), 7–34 (1996)
16. Hassard, B.: Bifurcation of periodic solutions of the Hodgkin–Huxley model for the squid giant axon. J. Theor. Biol. **71**(3), 401–420 (1978)
17. Hertz, J., Prügel-Bennett, A.: Learning short synfire chains by self-organization. Netw. Comput. Neural Syst. **7**(2), 357–363 (1996)
18. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500 (1952)
19. Ikegaya, Y., Sasaki, T., Ishikawa, D., Honma, N., Tao, K., Takahashi, N., Minamisawa, G., Ujita, S., Matsuki, N.: Interpyramid spike transmission stabilizes the sparseness of recurrent network activity. Cerebral Cortex **23**(2), 293–304 (2012)
20. Ito, S., Hansen, M.E., Heiland, R., Lumsdaine, A., Litke, A.M., Beggs, J.M.: Extending transfer entropy improves identification of effective connectivity in a spiking cortical network model. PLoS ONE **6**(11), e27431 (2011)
21. Koch, C., Segev, I.: Methods in Neuronal Modeling: From Ions to Networks. MIT Press, Cambridge (1998)
22. Kopell, N., Ermentrout, B.: Chemical and electrical synapses perform complementary roles in the synchronization of interneuronal networks. Proc. Natl. Acad. Sci. **101**(43), 15482–15487 (2004)
23. Mainen, Z.F., Sejnowski, T.J.: Influence of dendritic structure on firing pattern in model neocortical neurons. Nature **382**(6589), 363 (1996)
24. Mattia, M., Del Giudice, P.: Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. Neural Comput. **12**(10), 2305–2329 (2000)
25. McLaughlin, D., Shapley, R., Shelley, M., Wielaard, D.J.: A neuronal network model of macaque primary visual cortex (v1): orientation selectivity and dynamics in the input layer $4c\alpha$. Proc. Natl. Acad. Sci. **97**(14), 8087–8092 (2000)
26. Monteforte, M., Wolf, F.: Dynamic flux tubes form reservoirs of stability in neuronal circuits. Phys. Rev. X **2**(4), 041007 (2012)
27. Nie, Q., Wan, F.Y., Zhang, Y.T., Liu, X.F.: Compact integration factor methods in high spatial dimensions. J. Comput. Phys. **227**(10), 5238–5255 (2008)
28. Oseledec, V.I.: A multiplicative ergodic theorem. Lyapunov characteristic numbers for dynamical systems. Trans. Moscow Math. Soc. **19**(2), 197–231 (1968)
29. Ott, E.: Chaos in Dynamical Systems. Cambridge University Press, Cambridge (2002)
30. Parker, T.S., Chua, L.: Practical Numerical Algorithms for Chaotic Systems. Springer, Berlin (2012)
31. Perkel, D.H., Gerstein, G.L., Moore, G.P.: Neuronal spike trains and stochastic point processes: II. Simultaneous spike trains. Biophys. J. **7**(4), 419–440 (1967)
32. Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., Destexhe, A.: Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. Biol. Cybern. **99**(4), 427–441 (2008)
33. Quinn, C.J., Coleman, T.P., Kiyavash, N., Hatsopoulos, N.G.: Estimating the directed information to infer causal relationships in ensemble neural spike train recordings. J. Comput. Neurosci. **30**(1), 17–44 (2011)
34. Rangan, A.V., Cai, D.: Fast numerical methods for simulating large-scale integrate-and-fire neuronal networks. J. Comput. Neurosci. **22**(1), 81–100 (2007)
35. Revel, J., Karnovsky, M.: Hexagonal array of subunits in intercellular junctions of the mouse heart and liver. J. Cell Biol. **33**(3), C7 (1967)
36. Rinzel, J., Ermentrout, G.B.: Analysis of neural excitability and oscillations. Methods Neuronal Model. **2**, 251–292 (1998)
37. Shelley, M.J., Tao, L.: Efficient and accurate time-stepping schemes for integrate-and-fire neuronal networks. J. Comput. Neurosci. **11**(2), 111–119 (2001)
38. Shinomoto, S., Kim, H., Shimokawa, T., Matsuno, N., Funahashi, S., Shima, K., Fujita, I., Tamura, H., Doi, T., Kawano, K., et al.: Relating neuronal firing patterns to functional differentiation of cerebral cortex. PLoS Comput. Biol. **5**(7), e1000433 (2009)
39. Shlens, J., Field, G.D., Gauthier, J.L., Grivich, M.I., Petrusca, D., Sher, A., Litke, A.M., Chichilnisky, E.: The structure of multi-neuron firing patterns in primate retina. J. Neurosci. **26**(32), 8254–8266 (2006)
40. Somers, D.C., Nelson, S.B., Sur, M.: An emergent model of orientation selectivity in cat visual cortical simple cells. J. Neurosci. **15**(8), 5448–5465 (1995)
41. Song, S., Sjöström, P.J., Reigl, M., Nelson, S., Chklovskii, D.B.: Highly nonrandom features of synaptic connectivity in local cortical circuits. PLoS Biol. **3**(3), e68 (2005)

42. Sun, Y., Zhou, D., Rangan, A.V., Cai, D.: Library-based numerical reduction of the Hodgkin–Huxley neuron for network simulation. J. Comput. Neurosci. **27**(3), 369–390 (2009)
43. Sussillo, D., Abbott, L.F.: Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–557 (2009)
44. Thompson, J.M.T., Stewart, H.B.: Nonlinear Dynamics and Chaos. Wiley, New York (2002)
45. Tian, Z.Q.K., Zhou, D.: Exponential time differencing algorithm for pulse-coupled Hodgkin–Huxley neuronal networks. arXiv preprint arXiv:1910.08724 (2019)
46. Wang, Y., Barakat, A., Zhou, H.: Electrotonic coupling between pyramidal neurons in the neocortex. PLoS ONE **5**(4), e10253 (2010)
47. Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A.: Determining Lyapunov exponents from a time series. Phys. D: Nonlinear Phenom. **16**(3), 285–317 (1985)
48. Xu, Z.Q.J., Bi, G., Zhou, D., Cai, D.: A dynamical state underlying the second order maximum entropy principle in neuronal networks. Commun. Math. Sci. **15**(3), 665–692 (2017)
49. Zhou, D., Rangan, A.V., Sun, Y., Cai, D.: Network-induced chaos in integrate-and-fire neuronal ensembles. Phys. Rev. E **80**(3), 031918 (2009)
50. Zhou, D., Sun, Y., Rangan, A.V., Cai, D.: Spectrum of Lyapunov exponents of non-smooth dynamical systems of integrate-and-fire type. J. Comput. Neurosci. **28**(2), 229–245 (2010)
51. Zhou, D., Xiao, Y., Zhang, Y., Xu, Z., Cai, D.: Granger causality network reconstruction of conductance-based integrate-and-fire neuronal systems. PLoS ONE **9**(2), e87636 (2014)