Measuring Synchronization in Simple Neural Oscillators

Shivansh Dave

August 10, 2020

To be submitted as a term paper for BIOL 419 (Spring, 2020)

Contents

Summary

In this study, I have constructed simple neural oscillators using a pair of altered Morris-Lecar model neurons by varying different parameters such as system noise levels, synaptic connection type and neuron type (i.e., bursting/regular) to quantify and compare degree of synchrony between the two firing coupled neurons. For measuring synchronization I use two different methods (i.e., cross-correlation of raw voltage traces, and correlation using gaussian smoothed spike trains) to contrast their effectiveness for quantifying synchrony across different type of networks.

1 Introduction

Synchronization of neural firing is known to play a role in various neural phenomena. It plays an important role in memory formation. For example, tighter coupling in firing of neurons in medial temporal lobe in the brain is correlated with increased memory performance in humans and animals. In the same study it was found that neural synchronization is linked to a potential cellular mechanism for memory storage and timing-based learning tasks (Jutras and Buffalo 2010). In addition to this, synchronization is also associated with epilepsy, which affects about 3 million adults and 0.5 million children in the US and 50 million people worldwide (CDC 2018; WHO 2020). Mormann et al. (2003) found that decrease in neural synchronization precedes epileptic seizures and neural desynchronization is an immanent part of seizure initiating mechanism in humans. (Mormann et al. 2003)

Neural synchronization being an interesting topic to study, it is important to be able to understand how one can achieve synchronized firing using a simple system and to be able to quantify the synchrony in a simple neural oscillator. For an oscillator, the synchronization can be of broadly two types, namely, in-phase synchronization and anti-phase synchronization (Pikovsky and Rosenblum 2007), and different kinds of oscillator may give rise to different synchronization properties (González-Miranda 2014). For example, in certain conditions inhibitory synapse in an oscillator gives rise to synchronous firing as opposed to excitation (Van Vreeswijk, Abbott, and Bard Ermentrout 1994). In an oscillator, noise level may cause desynchronization and cause the weaker coupling between two neurons. Computational tools can allow us to construct a simple oscillator and let us study the effect of various factors such as system noise levels etc, on the degree of synchronization achieved by the oscillator.

To make neurons for oscillators, I use Morris-Lecar (ML) model, which is a reduced two-dimensional non-linear model for simulating neurons in contrast to the 4-dimensional Hodgkin–Huxley model (Lecar 2007). I use Ca^{2+} and K^{+} ion channels to simulate internal neuronal dynamics, as explained originally in (Morris and Lecar 1981). Now, if I connect two of such neurons to each other, would they give rise to sustained synchronous firing? and how does the degree of synchrony vary when I couple those neurons differently? This study is mainly focused on this question.

For simulating a simple oscillator, I use a half-center oscillator (HCO) made through synaptically connecting two ML neurons reciprocally with each other. Since different types of oscillators give rise to different oscillation patterns, I use many different HCOs (See section 2.4), made by varying the type of synapses and neurons. To get a bursting type HCO, I altered the ML neurons to remove their dependence on external stimulation current for initiating spikes. These bursting ML neurons have slow internal feedback current, responsible for their bursting properties (Mainen and Sejnowski 1996).

How synchronized is an oscillator for a given noise level in the neurons? To be able to study effects of noise (which can represent physical phenomenon such as ambient temperature, etc.), I stochastically simulate operation of both the ion channels and compare the synchronization of stochastically simulated HCO to the deterministic simulations. The smaller the count of ion channels present in a neuron, the larger the noise levels will be for that system. Thus, I vary system size (i.e., Ω) for each ion channel for quantifying their effects on synchronization.

2 Methods

I used the Morris–Lecar (ML) model as a base for making my neurons for deterministic simulations (See Section 2.1.1). I altered the model parameters to get the ion channels (i.e., K+ and Ca2+) to operate stochastically (see 2.1.2) by modeling the Chemical Langevin Equations for a first order single variable chemical reaction (see 2.1.3). The altered ion channel properties and a slow internal feedback current are used for making the ML neurons to operate in the "bursting mode" (see 2.1.4).

Chemical synapses of the type excitatory as well as inhibitory are achieved by introducing the synapses on ML neurons (see 2.3). Two ML neurons of any type (i.e., deterministic/stochastic and plain/bursting) are reciprocally connected to each other using chemical synapses (i.e., excitatory or inhibitory) to get the functioning Half center oscillator (HCO), (see 2.4). The HCO shows a varied degree of synchronizations based on the type of neurons and the connections. I employ and compare both, the gaussian smoothed spike-time method (see 2.5.2) and the Pearson's linear correlation of raw voltage traces, to compare the degree of synchronization between the participating ML neurons in various types of HCOs.

2.1 ML Neurons

I am simulating an array of altered Morris-Lecar neurons (i.e., ML neurons) in parallel to find the next membrane voltage and the ion channel states using the present conditions using a Matlab script I wrote for this purpose (see B.1.1). Any of these neurons can receive a chemical synapse and project a chemical synapse onto any other neuron connected in the network. The Matlab simulation script (with a "demo" network) is used for initializing and simulating the network for a given duration (see Appendix B.1.1).

This section contains the details on the mathematical and computational tools used for basic operational realization of ML neurons. Subsequent sections will cover any further modifications such as introducing synapses (see 2.3) and enabling slow internal feedback current for bursting mode (see 2.1.4).

2.1.1 Deterministic Neuron

Firstly, to achieve deterministic simulation of ML neurons I used equations (1)-(6), and used some constant parameters (7), which I referred from (Anderson, Ermentrout, and Thomas 2015; Lecar 2007).

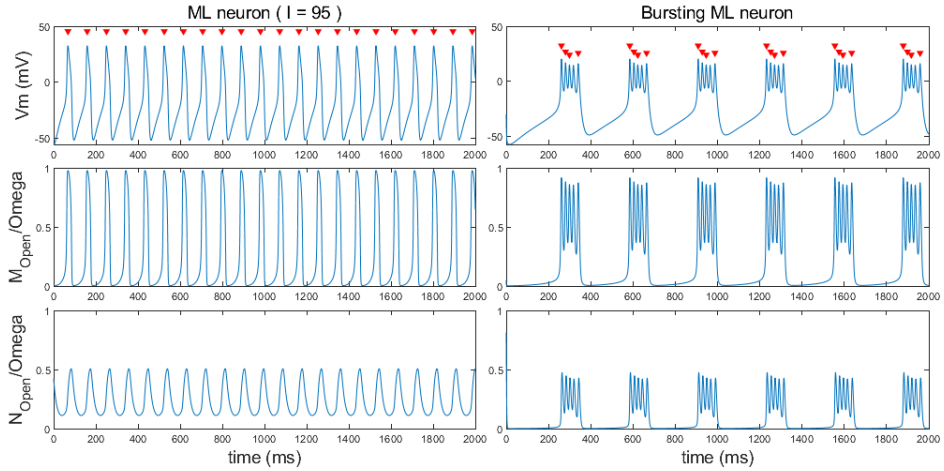


Figure 1: (A) Deterministic ML Neuron membrane voltage and activation of K^+ and Ca^{2+} ion channels, (left) in plain/regular ML neurons; (right) ML neurons with bursting mode.

$$\frac{dv}{dt} = f(v,n) = \frac{1}{C}(I_{\text{app}} - g_{\text{Ca}} m_\infty(v)(v - v_{\text{Ca}}) - g_{\text{K}} n(v - v_{\text{K}}) - g_{\text{L}}(v - v_{\text{L}})) \tag{1}$$

$$\begin{aligned}\frac{dn}{dt} = g(v,n) &= \alpha(v)(1-n) - \beta(v)n \\ &= (n_\infty(v) - n)/\tau(v)\end{aligned} \tag{2}$$

Equation (1) shows the deterministic membrane voltage for the ML model neurons. In which, $n; (n \in [0,1])$ represents a fraction of K^+ ion channels open at the given time t. I_{app} is the current value, measured in $\mu A/cm^2$, which can represent either the current-clamp current (i.e, I_{stim}) in case of regular or plain ML neurons (left in Figure1a) or it can represent the slow internal feedback current (i.e., $I_{internal}$) for the bursting mode ML neurons (see 2.1.4; right plots in Figure1a). Eq.(2) represents the dynamical opening of K^+ ion channels which depends on α and β, which are the *per capita* transition rates of ion channels to open or close dynamically (3). ALternatively, the dynamic operation of n can be computed by the steady-state value of n (i.e., the fraction of K^+ ion channels open in steady-state) and temporal dynamics of the recovery process of channel operation, τ, as shown in (2). The values of α, β, n_∞ and τ depend on a present value of membrane voltage, v, and can be modeled using the voltage constants (i.e., v_c, v_d). For convenience, a membrane voltage dependent variable, ξ, is defined (3). The voltage constants v_a and v_b are used to model membrane voltage dependent dynamic operation of Ca^{2+} ion channels.

$$\begin{aligned}&\phi = 0.04\ s^{-1},\ v_a = -1.2\ mV,\ v_b = 18\ mV,\ v_c = 2\ mV,\ v_d = 30\ mV, \\ &\xi = \frac{v - v_c}{v_d},\ \alpha(v) = \frac{\phi \cosh(\xi/2)}{1 + e^{2\xi}},\ \beta(v) = \frac{\phi \cosh(\xi/2)}{1 + e^{-2\xi}},\end{aligned} \tag{3}$$

$$n_\infty(v) = \frac{\alpha(v)}{\alpha(v) + \beta(v)} \ = \ \frac{1}{2}(1 + \tanh \xi) \tag{4}$$

$$m_\infty = \frac{1}{2}\left(1 + \tanh(\frac{v - v_a}{vb})\right) \tag{5}$$

$$\tau(v) = \frac{1}{\alpha(v) + \beta(v)} \ = \ \frac{1}{\phi \cosh(\xi/2)} \tag{6}$$

Other fixed constants used for deterministic simulations are listed below (7). Ion channel maximum conductance $g_{\text{K}}, g_{\text{Ca}}$ *and* g_{L} are for K^+, Ca^{2+} and leak channels respectively, which are measured in $m\mho/cm^2$. Reversal potentials for these ion channels (i.e., $v_{\text{K}},\ v_{\text{Ca}},\ v_{\text{L}}$) and other voltage constants (i.e., $v_{a,b,c,d}$) are measured in mV. ϕ, measured in s^{-1} is a rate constant for the recovery process of K^+ ion channels. v_a and v_c are the voltages at which the ion channel activation function m_∞ and n_∞, respectively, becomes 0.5; and v_b and v_d, respectively, are the slope values for those membrane voltage dependent functions. C is the membrane capacitance measured in $\mu F/cm^2$. (Morris and Lecar 1981)

$$\begin{aligned}&v_{\text{Ca}} = 120\ mV,\ v_{\text{K}} = -84\ mV,\ v_{\text{L}} = -60\ mV, \\ &g_{\text{Ca}} = 4.4\ m\mho/cm^2,\ g_K = 8\ m\mho/cm^2,\ g_{\text{L}} = 2\ m\mho/cm^2, \\ &C = 20\ \mu F/cm^2,\ I_{\text{app}} = (variable;\ measured\ in\ \mu A/cm^2)\end{aligned} \tag{7}$$

Using the equations (1)-(7), by applying $I_{\text{app}} = I_{stim} = 95$, I see the membrane voltage and ion channels as shown in the left in Figure 1a. Red-markers indicate the identified spikes in the membrane voltage traces (see 2.5.1). The signal in the second is normalized Ca^{2+} ion channel activation computed using m_∞ and the signal in the third row represents the normalized activation of K^+ ion channels (i.e., $n, n \in [0,1]$), computed using equation (2). Bursting mode ML neurons show fast and slow dynamics, that is the fast spikes for action potential and the slow oscillation of *plateau potential*, more details on the simulation parameters are given in section 2.1.4.

2.1.2 Stochastic ML Neuron

Stochastic activation of both, the K^+ and Ca^{2+}, ion-channels are used for simulation of ML neurons and to compute membrane voltage (Figure 1b). For the simulation I used the following equations (8)-(20), and used constant parameters (21) for the ML neurons, as described in (Anderson, Ermentrout, and Thomas 2015).

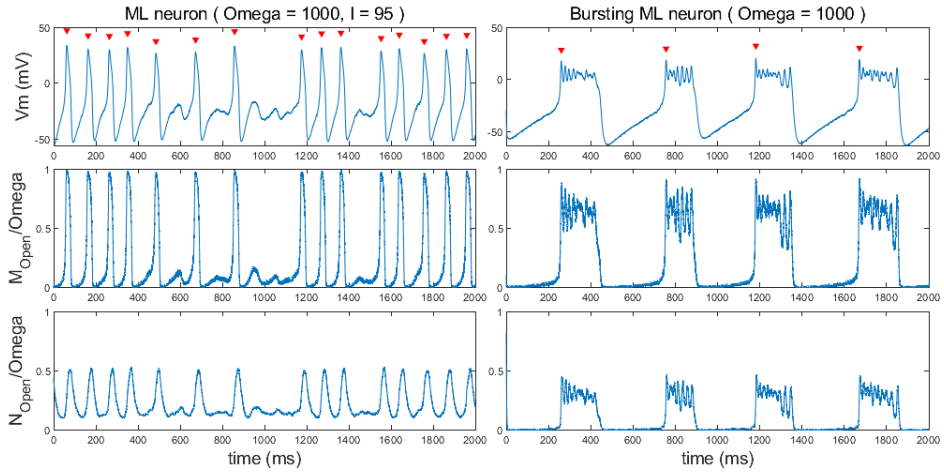


Figure 1: (B) Stochastic ML Neuron membrane voltage and Stochastic activation of K^+ and Ca^{2+} ion channels, (left) in plain/regular ML neurons; (right) ML neurons with bursting mode.

Equation (8) shows the stochastic membrane voltage for the ML model neurons with n and m; $(n, m \in [0, 1])$ represents the fraction of K^+ and Ca^{2+} ion channels open at the given time t. Both of which depend on the instantaneous membrane voltage of the ML neuron Eq. (9)-(10). Stochastic activation of K^+ and Ca^{2+} ion channels depends on the *per capita* transition rates α_n, β_n and α_m, β_m, respectively [(12), (13), (17), (18)]. Same as the deterministic equations, the I_{app} $(\mu A/cm^2)$ represents either the current-clamp current (i.e, I_{stim}) in case of regular or plain ML neurons (left, in Figure 1b) or the slow internal feedback current (i.e., $I_{internal}$) for the bursting mode ML neurons (see 2.1.4; right plots, in Figure 1b).

$$\frac{dv}{dt} = F(v,n,m) = \frac{1}{C}(I_{\text{app}} - g_{\text{Ca}} m(v - v_{\text{Ca}}) - g_{\text{K}} n(v - v_{\text{K}}) - g_{\text{L}}(v - v_{\text{L}})) \tag{8}$$

$$\begin{aligned}\frac{dn}{dt} = G(v,n,m) &= \alpha_n(v)(1-n) - \beta_n(v)n \\ &= (n_\infty(v) - n)/\tau_n(v)\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{dm}{dt} = H(v,n,m) &= \alpha_m(v)(1-m) - \beta_m(v)m \\ &= (m_\infty(v) - m)/\tau_m(v)\end{aligned} \tag{10}$$

Cell voltage dependent functions of $\alpha_m, \beta_m, \tau_m$ and m_∞ are modeled using the voltage constants (i.e., v_a, v_b (mV)) and recovery process rate constant ϕ_m (s^{-1}) for stochastic operation of Ca^{2+} ion channels. The activation threshold (i.e., $n = 0.5$) and the slope of the voltage dependent activation function, n_∞ is set by V_a and v_b, respectively. For convenience, a membrane voltage dependent variable, ξ_m, is defined using v_a, v_b [Equations (11)-(15)].

$$\phi_m = 2\ s^{-1},\ v_a = -1.2\ mV,\ v_b = 18\ mV,$$

$$\xi_m = \frac{v - v_a}{v_b}, \tag{11}$$

$$\alpha_m(v) = \frac{\phi_m \cosh(\xi_m/2)}{1 + e^{2\xi_m}}, \tag{12}$$

$$\beta_m(v) = \frac{\phi_m \cosh(\xi_m/2)}{1 + e^{-2\xi_m}}, \tag{13}$$

$$\begin{aligned} m_\infty(v) &= \frac{\alpha_m(v)}{\alpha_m(v) + \beta_m(v)} \\ &= \frac{1}{2}(1 + \tanh \xi_m) \end{aligned} \tag{14}$$

$$\begin{aligned} \tau_m(v) &= \frac{1}{\alpha_m(v) + \beta_m(v)} \\ &= \frac{1}{\phi \cosh(\xi_m/2)} \end{aligned} \tag{15}$$

Same as the Ca^{2+} channels, the K^+ channels are stochastically simulated using voltage dependent functions such as $\alpha_n, \beta_n, \tau_n$ and n_∞. These functions are modeled using the voltage constants (i.e., v_c, v_d (mV)) and rate constant ϕ_n (s^{-1}) and for convenience, a membrane voltage dependent variable, ξ_n, is defined using v_c, v_d [Equations (16)-(20)]. v_c sets the activation threshold (i.e., $n = 0.5$) and v_d sets the slope of the voltage dependent activation function, n_∞.

$$\phi_n = 0.04\ s^{-1},\ v_c = 2\ mV,\ v_d = 30\ mV,$$

$$\xi_n = \frac{v - v_c}{v_d}, \tag{16}$$

$$\alpha_n(v) = \frac{\phi_n \cosh(\xi_n/2)}{1 + e^{2\xi_n}}, \tag{17}$$

$$\beta_n(v) = \frac{\phi_n \cosh(\xi_n/2)}{1 + e^{-2\xi_n}}, \tag{18}$$

$$\begin{aligned} n_\infty(v) &= \frac{\alpha_n(v)}{\alpha_n(v) + \beta_n(v)} \\ &= \frac{1}{2}(1 + \tanh \xi_n) \end{aligned} \tag{19}$$

$$\begin{aligned} \tau_n(v) &= \frac{1}{\alpha_n(v) + \beta_n(v)} \\ &= \frac{1}{\phi \cosh(\xi_n/2)} \end{aligned} \tag{20}$$

The stochastic ML neuron constants (21) such as reversal potentials, conductance and capacitance are set exactly the same as deterministic parameters (7). (Morris and Lecar 1981; Anderson, Ermentrout, and Thomas 2015)

$$\begin{aligned} &v_{\mathrm{Ca}} = 120\ mV,\ v_{\mathrm{K}} = -84\ mV,\ v_{\mathrm{L}} = -60\ mV, \\ &g_{\mathrm{Ca}} = 4.4\ m\mho/cm^2,\ g_K = 8\ m\mho/cm^2,\ g_{\mathrm{L}} = 2\ m\mho/cm^2, \\ &C = 20\ \mu F/cm^2,\ I_{\mathrm{app}} = (variable;\ measured\ in\ \mu A/cm^2) \end{aligned} \tag{21}$$

Figure 1b is an example simulation of stochastically activating Ca^{2+} and K^+ ion channels (displayed in the second and third rows, respectively) using the equations (8)-(21). For that purpose system size (i.e., $\Omega_{N,M}$)

for both ion channels is set to 1000 and for the plain neurons (on the left) are made to spontaneously spike action potentials by applying $I_{\text{app}} = I_{stim} = 95$ with red-markers to indicate the identified spikes in the membrane voltage traces (see 2.5.1). Stochastic ML neurons in bursting mode are explained in section 2.1.4. The following section, 2.1.3 shows the derived model using Langevin equations for simulating these neurons.

2.1.3 Setting up Chemical Langevin Equation

In the study both the ion channels in ML neurons, K^+ and Ca^{2+} channels are operating stochastically for the stochastic simulations. I use Chemical Langevin Equations (CLE) to simulate ion channels stochastically. This method is an approximation method and uses Stochastic Differential Equations (SDE) for stochastic simulation, which is faster than other approximation methods like τ-leaping and the exact methods like Gillespie's exact method (Higham 2008). To speed up computation CLE uses diffusion approximation in addition to time approximations used in τ-leaping method, and with faster reaction rates, to get sufficiently large number of reactions happening (for each species), the errors incur would be small by switching to CLE as compared to other exact methods (Wilkinson 2019).

As shown in (22), for each ion channels (for each different ions involved in the simulation, i.e., K^+, Ca^{2+} for ML neurons) I can compare it to a first order reaction chemical reaction showing the rates of switching the states between "open" and "close". (Note : Here N_{open}, N_{close} shows the states for K^+ ion channels, and similarly one can get another sets of equations for Ca^{2+} ion channels represented by M_{open}, M_{close}.). Similar method is present for Hodgkin–Huxley model in (Ermentrout and Terman 2010).

$$\mathrm{N_{close}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} \mathrm{N_{open}} \tag{22}$$

From (22), two sub-reactions can be formulated along with a stoichiometry matrix for each sub-reaction. (23) and (24) shows the chemical hazard function for channel opening and closing reactions.

$$\begin{aligned} &\textit{Reaction } (1) : \mathrm{N_{close}} \xrightarrow{\alpha} \mathrm{N_{open}}, \\ &h_1(N) = \alpha N_{close} \ ; \quad \zeta_1 = \begin{pmatrix} -1 \\ +1 \end{pmatrix} \end{aligned} \tag{23}$$

$$\begin{aligned} &\textit{Reaction } (2) : \mathrm{N_{open}} \xrightarrow{\beta} \mathrm{N_{close}}, \\ &h_2(N) = \beta N_{open} \ ; \quad \zeta_2 = \begin{pmatrix} +1 \\ -1 \end{pmatrix} \end{aligned} \tag{24}$$

Since in my model, mass conservation is applicable as no immigration/birth/death are not happening, for simplifying the computation, I can convert 2D sets of equations to 1D equations. Total count for K^+ ion channels in the simulation is denoted by Ω_N (similarly Ω_M for Ca^{2+} channels). $N(t)$ represents the count of channels in N_{open} state at a given time, t, so the count of channels in N_{close} state would be $\Omega_N - N(t)$ at that time. I then updated the reaction hazard function for 1D system @ref((eq:1d).

$$\begin{aligned} \Omega_N = N_{open} + N_{close} \ ; \quad N(t) = N_{open}(t) \\ \therefore \ N_{close}(t) = \Omega_N - N_{open}(t) \\ h_{1(N)} = \alpha(\Omega_N - N) \\ h_{2(N)} = \beta N \end{aligned} \tag{25}$$

Chemical Langevin Equations can be set up as described (Higham 2008; Wilkinson 2019) for ion channel simulation. For each time step of τ, the state of each ion channels counts can be updated using the following equation :

$$
\begin{aligned}
N(t+\tau) &= N(t) + \tau(h_{1(N)} - h_{2(N)}) \\
&\quad + \sqrt{\tau\ h_{1(N)}}\ \xi_1(t) \\
&\quad - \sqrt{\tau\ h_{2(N)}}\ \xi_2(t)\ ;\ \ \xi_{1,2}(t) \sim \mathcal{N}(0,1) \qquad (26)\\
\implies N(t+\tau) &= N(t) + \tau(h_{1(N)} - h_{2(N)}) \\
&\quad + \sqrt{\tau}\sqrt{h_{1(N)} + h_{2(N)}}\ \xi(t)\ ;\ \ \xi(t) \sim \mathcal{N}(0,1)
\end{aligned}
$$

I use (27)-(29) for simulating simple ML neurons (Figure 1b) to update each ion channel stochastically using CLE.

$$\frac{dV}{dt} = F(V(t), N(t), M(t)) = \frac{1}{C}(I_{\text{app}} - g_{\text{Ca}}\frac{M(t)}{\Omega_M}\,(V(t) - v_{\text{Ca}}) - g_{\text{K}}\frac{N(t)}{\Omega_N}\,(V(t) - v_{\text{K}}) - g_{\text{L}}(V - v_{\text{L}})) \quad (27)$$

$$
\begin{aligned}
N(t+\tau) &= N(t) + \tau(\alpha(\Omega_N - N(t)) - \beta N(t)) \\
&\quad + \sqrt{\tau}\sqrt{\alpha(\Omega_N - N(t)) + \beta N(t)}\ \xi_N(t) \qquad (28)\\
M(t+\tau) &= M(t) + \tau(\alpha(\Omega_M - M(t)) - \beta M(t)) \\
&\quad + \sqrt{\tau}\sqrt{\alpha(\Omega_M - M(t)) + \beta M(t)}\ \xi_M(t)\ ;\quad \xi_{N,M}(t) \sim \mathcal{N}(0,1) \qquad (29)
\end{aligned}
$$

Values of other parameters and constants are as described earlier in 2.1.2. Also, see 2.3 for the equations with adding chemical synapse used for simulation of HCOs.

2.1.4 Bursting mode in ML Neurons

ML Neurons explained in previous sections were plain/regular neurons (i.e., non-bursting), where input current is the current clamp current (i.e., $I_{\text{app}} = I_{stim}$). But, I also used ML neurons in bursting mode (right side in Figure 1a & 1b). In bursting mode, the Current clamp current is set to zero (i.e., $I_{stim} = 0$) and a slow feedback current internal current, computed as (30), is responsible for *plateau potential* and reaching the action potential firing threshold is responsible for fast dynamics of spikes (Izhikevich 2006). For bursting mode, I used the following parameters, which I found suitable to elicit fast and slow dynamics present in this mode.

$$
\begin{aligned}
I_{\text{app}} &= I_{internal} \\
\varepsilon = 0.01;\ v_o &= -26; \\
\frac{dI_{internal}}{dt} &= \varepsilon * (v_o - V(t)) \qquad (30)
\end{aligned}
$$

For simulating ion channels most of the ML neuron parameters and constants are used as described in the previous section (For deterministic simulation (7); stochastic simulation (21)). The changes in the parameters are shown as below.

For achieving bursting mode in deterministic simulation (right side, Figure 1a), the changes made are shown in (31).

$$\phi = 0.23,\ v_c = 12,\ v_d = 17.4 \qquad (31)$$

Similarly, for bursting mode in stochastic simulation (right side, Figure 1b), the changes in parameters are shown in (32).

$$\phi_m = 2,\ \phi_n = 0.23,\ v_c = 12,\ v_d = 17.4 \tag{32}$$

Detailed description on designing different types of neurons is given in (Rinzel and Ermentrout 1998).

2.2 Effect of Current and Noise

Stimulus current would change the output of regular ML neurons and change in system size would affect the noise level in both the types of ML neurons, regular and bursting mode.

2.2.1 Varying current and system sizeIn plain ML neurons

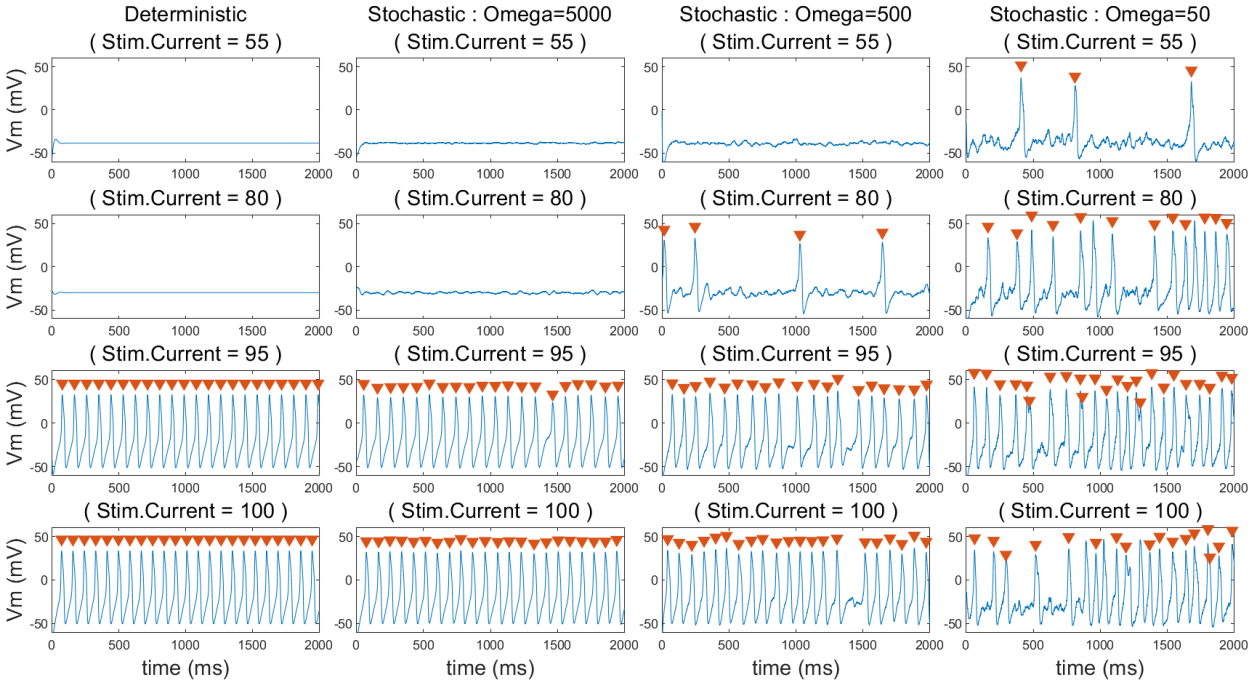


Figure 2: (A) Plain (non-bursting) ML neurons for different system size and stim. current.

Figure 2a shows regular ML neurons changing behavior with varied current (along the column) and the noise (along the row). Threshold amount of stimulus current for deterministic neurons to start firing is about 89 $\mu A/cm^2$. For stochastic neurons, increasing noise (i.e., reducing Ω) allows more spontaneous spikes for sub-threshold stimulus currents.

2.2.2 In Bursting mode

Figure 2b shows Bursting mode ML neurons under various noise levels. Changing noise levels affects the rhythm and bursting dynamics of ML neurons. Increasing noise increases bursting frequency in such neurons.

To summarize, higher system size in stochastic simulation causes lower noise, and the resulting patterns look more comparable to deterministic simulations. Also, for regular neurons, increasing current causes frequent firing of spikes. Effects of noise are drastic in lower current simulations.

2.3 Adding Synapse

To be able to create a network out of ML neurons, I altered the ML model neurons, once again, to have a chemical synapse of both the inhibitory and excitatory types. Each neuron in the array of network, can be

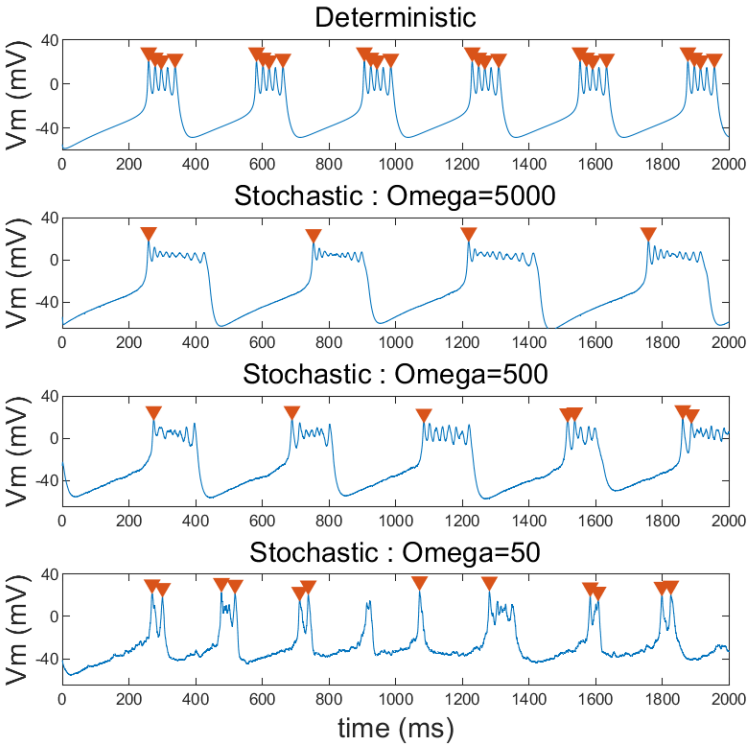


Figure 2: (B) ML neurons in bursting mode with different system size. No external stimulus current is provided here.

configured separately to receive either no-synapse, inhibitory or excitatory synapses from a desired neuron from the network (Appendix B.1.2).

2.3.1 Activation function

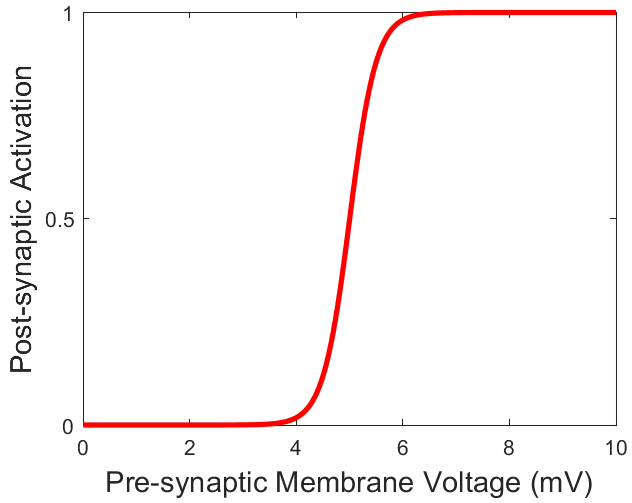


Figure 3: (A) Synaptic Activation Function

Figure 3a shows the activation function with a synaptic threshold of +5 mV and the function slope of 0.5. I used (33) to define the sigmoidal activation function, which is inspired from (Yu and Thomas, n.d.).

By changing Omega, your system size. Currently it looks like you have Omega=100 for the

potassium channels, by default (line 89 of your code). If you make Omega equal to 1e4 then the noise should be reduced by 90%, so to speak. I tried this in your code, and the cells stopped firing. This means that with current=60, a cell is in the "excitable" regime, where it will only fire if forced (by noise or by the other cell).
I tried setting the current to 100 and then the cells both fired even when omega was 1e4.

$$\begin{aligned} syn_{Threshold} = 5,\ syn_{Slope} = 0.5, \\ S_{syn}(v) = \frac{1}{2}(1 + \tanh((v - syn_{Threshold})/syn_{Slope})); \end{aligned} \tag{33}$$

2.3.2 Synaptic current

Synaptic current (measured in $\mu A/cm^2$) is computed using the following formula for Inhibitory and excitatory synapses in (34).

$$\begin{aligned} v_{\text{syn}}(exci.) = 100\ mV,\ v_{\text{syn}}(inhi.) = -100\ mV,\ g_{\text{syn}} = 1\ m\mho/cm^2 \\ I_{\text{syn}}(v) = -\ g_{\text{syn}} S_{syn}(v)(v - v_{\text{syn}})\ \ \mu A/cm^2 \end{aligned} \tag{34}$$

Synaptic current is added in in I_{app} along with other possible currents such as current clamp current, I_{stim}, or the internal feedback current $I_{internal}$. Thus, for regular ML neurons with synapse will become as following,

$$\begin{aligned} (For\ plain\ ML\ neuron\ with\ Synapse)\ \ I_{\text{app}} = I_{stim} + I_{\text{syn}}(v) \\ (For\ bursting\ ML\ neuron\ with\ Synapse)\ \ I_{\text{app}} = I_{internal} + I_{\text{syn}}(v) \end{aligned} \tag{35}$$

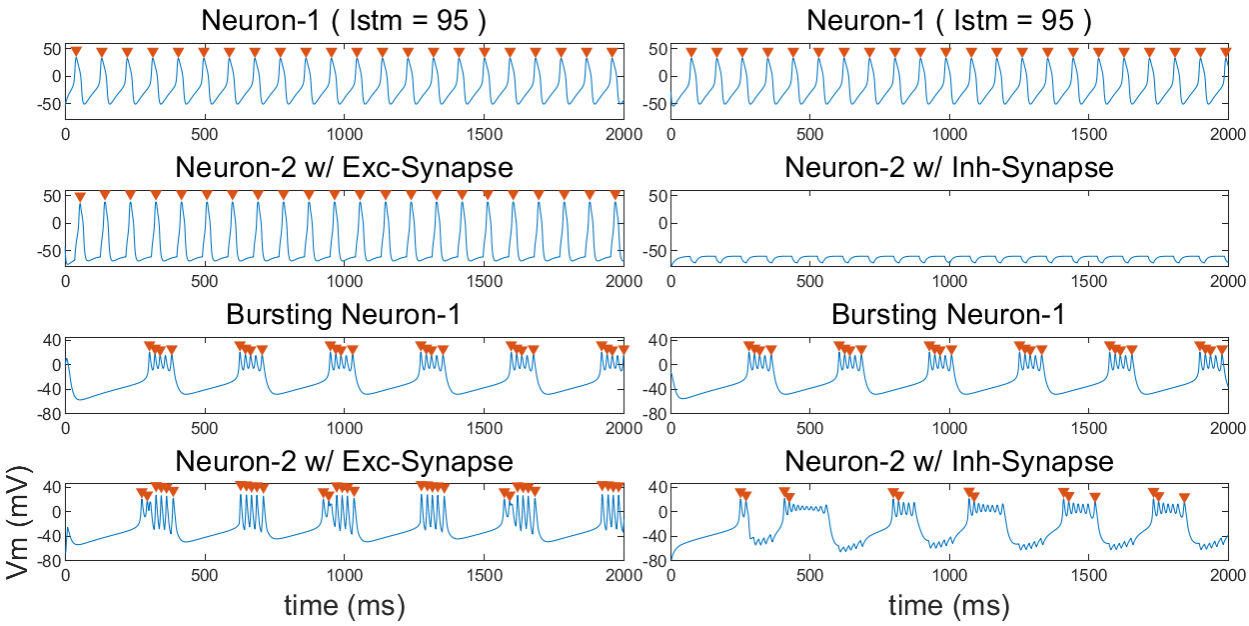


Figure 3: (B) Synaptic operations in deterministic ML neurons

2.3.2.1 Synapses in deterministic simulation Figure 3b shows the effect of a synapse in deterministic ML neurons in regular and bursting mode with the synapse is formed on Neuron-2. Regular type Neuron-1 are having current clamp, $I_{stim} = 95$ to elicit spontaneous spikes in Neuron-1. Neuron-2 is left unconnected for both the cases to see the effect of presynaptic activity.

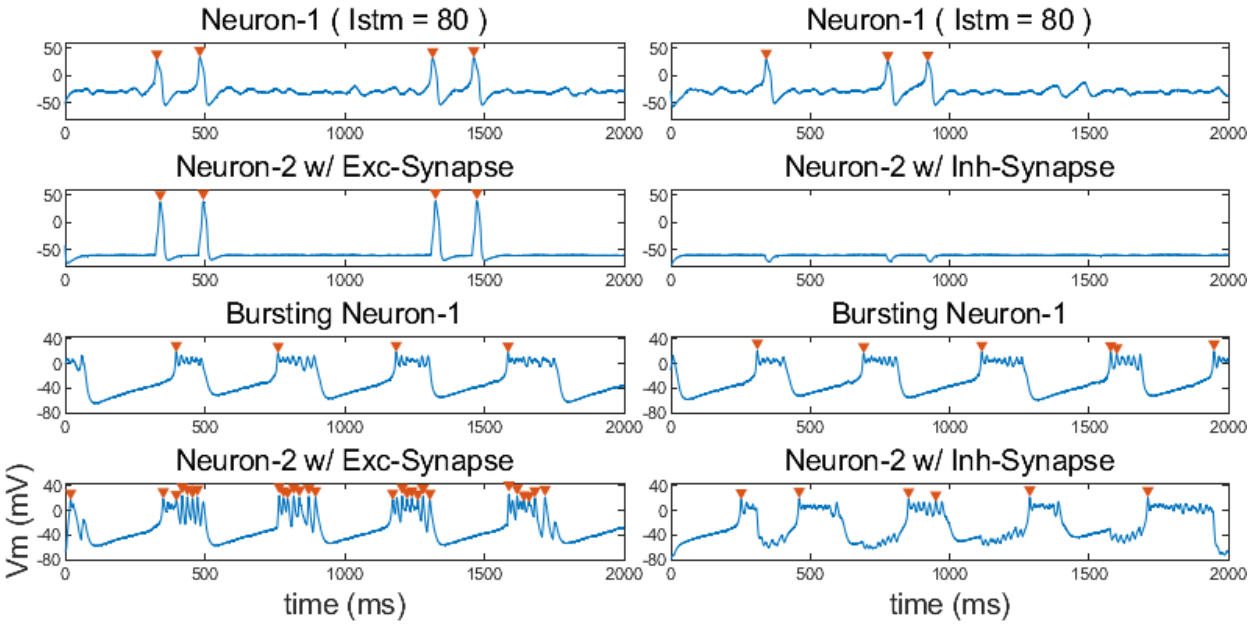


Figure 3: (C) Synaptic operations in deterministic and stochastic ML neurons

2.3.2.2 Synapses in stochastic simulation Figure 3c shows synapse in stochastic ML neurons with $\Omega_{M,N} = 500$ for all neurons. Neuron-1 (regular type) are firing action potentials due to the current clamp stimulus current $I_{stim} = 80$ and the bursting neurons (at the bottom of the image), show the firing due to internal feedback current. Each of the "driving" neuron is labeled as Neuron-1, which is projecting a synapse onto Neuron-2 via either excitatory (left-column) or inhibitory (right-column) chemical synapse. Figure 3c displays stochastic neurons.

2.4 Half Center Oscillator

Half center oscillator is formed through a reciprocal connection of two ML neurons and each of them are supplied with super-threshold stimulus current in case of plain ML neuron types. So combining the type of synapse type of ML neuron, I get the following types of HCOs :

1. Excitatory HCO with ML neurons, i.e, e-HCO(1)
2. Inhibitory HCO with ML neurons, i.e, i-HCO(1)
3. Excitatory HCO with bursting ML neurons, i.e, e-HCO(2)
4. Inhibitory HCO with bursting ML neurons, i.e, i-HCO(2)

Each of these HCO types can either be simulated using deterministic or stochastic methods. In case of stochastic simulation, one can vary the noise levels to see the effect on it's synchronization.

2.4.1 Plain ML neurons HCO

Figure 4a shows HCO with plain ML neurons with inhibitory and excitatory synapses. Stimulus current is enabled for each of these neurons, $I_{stim} = 95$. Effect of noise level is seen through varying system size, $(\Omega_{M,N})$.

2.4.2 Bursting mode ML neurons HCO

Similarly, HCO with bursting ML neurons is shown in Figure 4b where stimulus current is set to zero and each neuron is having internal feedback current.

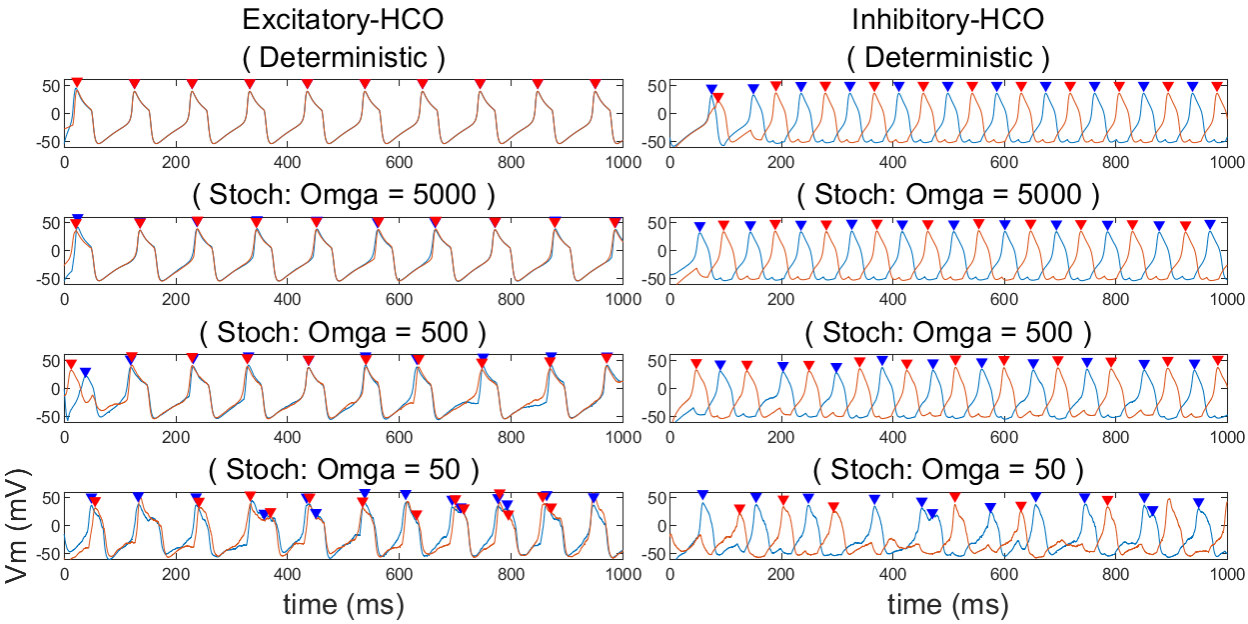


Figure 4: (A) HCO with non-bursting ML neurons

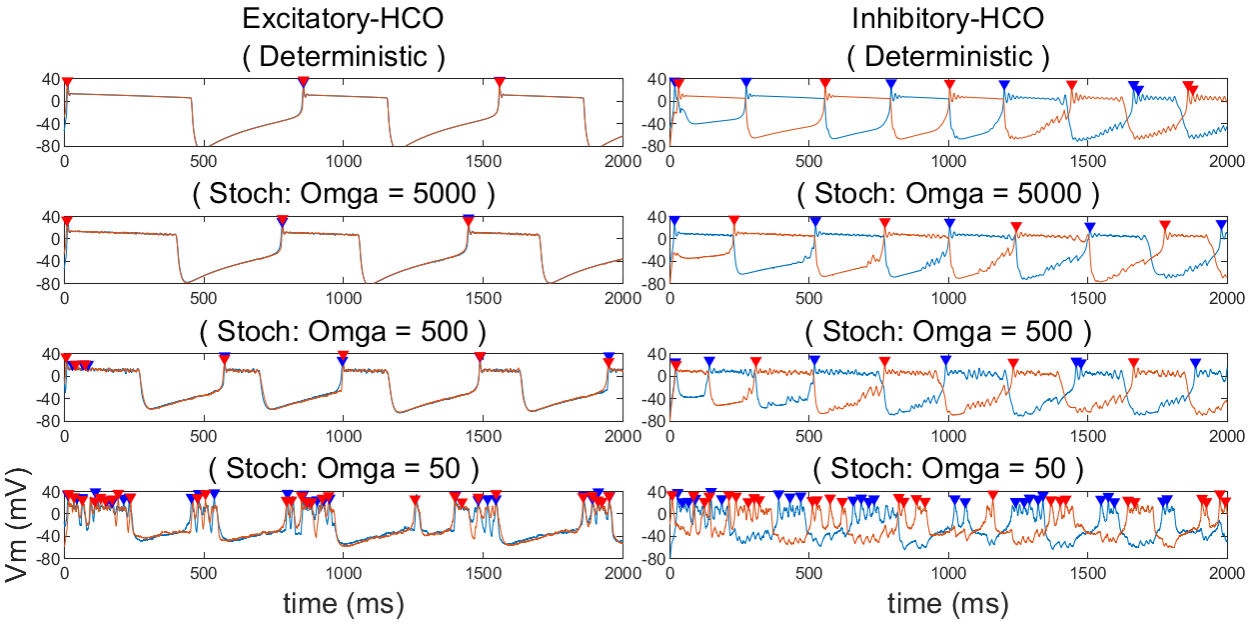


Figure 4: (B) HCO with bursting ML neurons

It appears that both excitatory and inhibitory HCOs show synchronization where the e-HCO show in-phase oscillations and i-HCO show out-phase oscillations. In general, lower noise level shows tight coupling between the oscillators in either case. (See section 4 for discussion).

2.5 Measuring Synchrony

Once I get the voltage traces of both the neurons, I measure the amount of synchronization for each HCO type, and later to compare the synchronization among all types. For measuring synchrony I use two methods.

1. Cross-correlation of two raw voltage traces.
2. Gaussian smoothed spike-time based binless correlation.

The first method does not make any assumptions and it compares raw voltage traces to each other to each other. Pearson's linear correlation coefficient is fond for the normalized voltage traces of neuron-1 and neuron-2 for a given HCO. However, this method can be influenced by the noise levels in the system, which can be an issue for lower system size simulations. So, in addition to this method, I am also using gaussian smoothed spike-train for finding correlation, which incurs less error than simple *peristimulus time histogram* (PSTH) as this method uses gaussian kernel and eliminates the noise generated from binning of the spike times, as previously described in detail (Kruskal et al. 2007; Victor and Purpura 1997).

2.5.1 Detect spikes

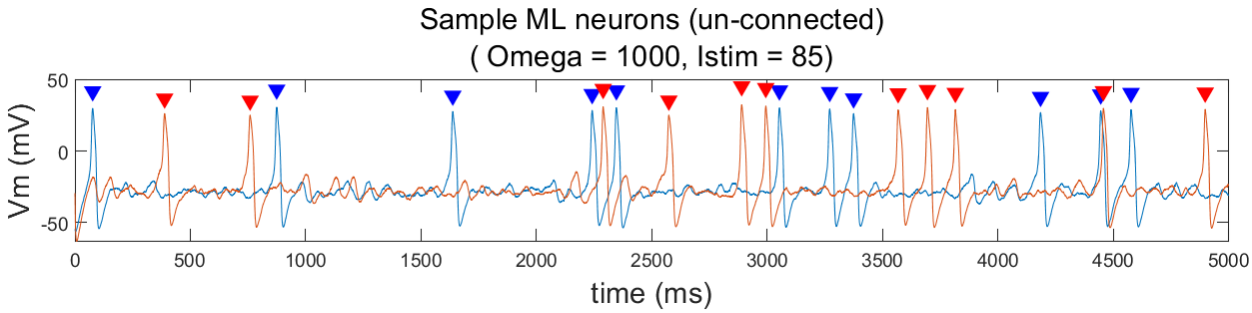


Figure 5: Detecting spikes in ML neurons using amplitude threshold of 15mV.

I am finding and saving spike-times along with raw voltage traces for each simulation because one of the methods to quantify synchronization requires them. Figure 5 shows threshold based spikes detection result for a sample ML neuron with the voltage threshold set to 15 mV. I find local maxima to identify peaks in a signal. I use a refractory period of 15ms to avoid detecting multiple spikes for a single action potential, which would have been an issue when an unfiltered neuron has multiple local-maxima. Two sample neurons here are not connected to each other via any synapse.

2.5.2 Gaussian smoothing

Previously Figure 5 shows identified spike-times (filled markers) and raw membrane voltage traces for two ML neurons. Here, in the Figure 6 the gaussian-smoothed spike-train with a gaussian-kernel of different σ (S.D.) are shown for the same neurons 1 and 2. Gaussian window size calculated from the Bin-Width (W) using $\sigma = W/\sqrt{12}$ (Kruskal et al. 2007). Kernels are shown in the figure on the right pane.

2.5.3 Single Run : Effect of bin-width on cross-correlation

As previously shown, varying σ would change the gaussian-smoothed spike trains, and subsequently it will change the correlation. Here I am comparing correlation coefficient found for various bin-width (thus, varied σ) and comparing it across a wider range of noise levels (i.e., $\Omega_{M,N} = \{5e4, 5e3, 5e2, 5e1, 5e0\}$) for each HCO types.

Figure 7 shows correlation of raw traces (i.e., without identifying spike-times) in upper left corner; and also shows the effect of S.D. of the gaussian window (i.e., $\sigma = W/\sqrt{12}$) used for getting the binless correlation between the spike trains. The raw membrane voltage traces for each of the neurons in all the HCOs are shown in Appendix, Figure-9. It is apparent that raw traces and smaller bin-width (W=50) yields more similar correlation than the higher bin-widths (W=250,500) and increasing the noise levels generally reduces

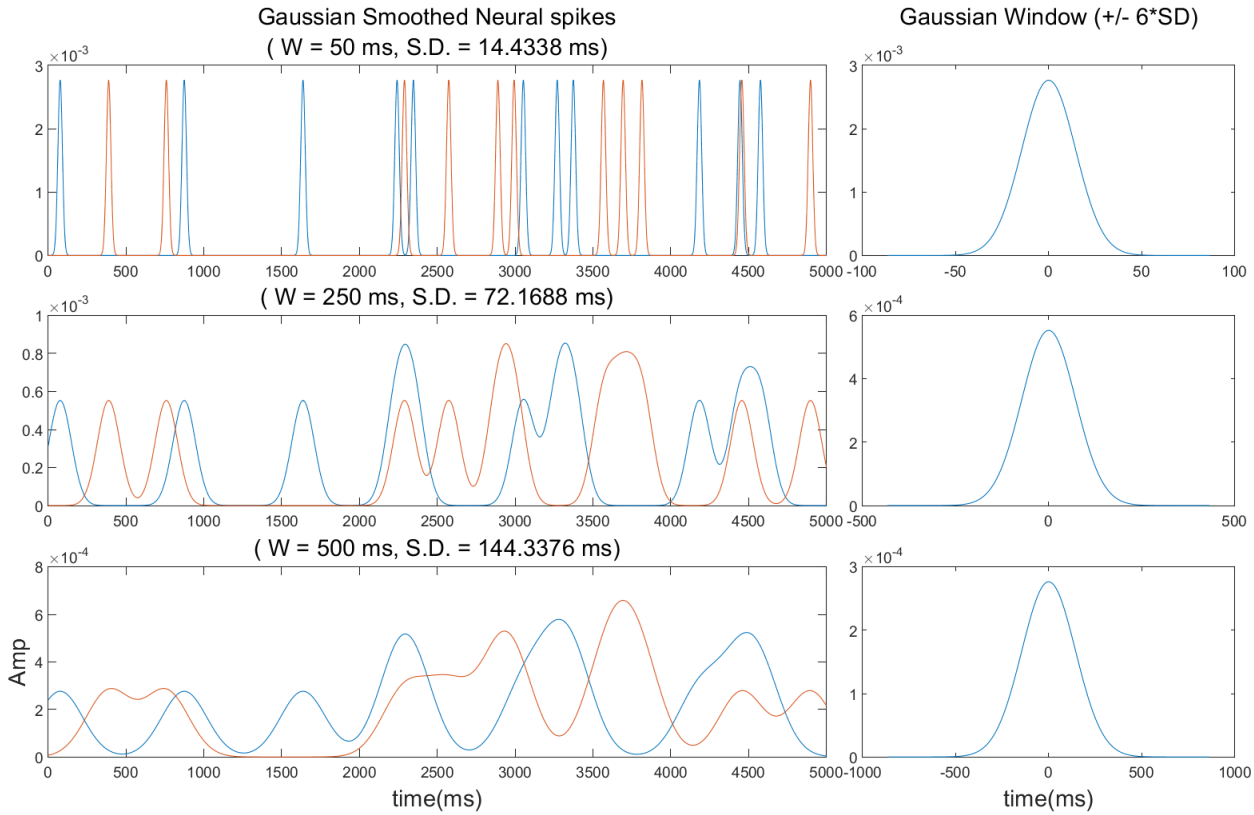


Figure 6: Gaussian smoothing of spike trains with Gaussian kernel of different σ.

the coupling of the oscillators. Wider bin-width shows higher value of correlation as compared to smaller widths. (See section 4 for discussion).

3 Results

While finding such correlation I noticed that some values were fluctuating a lot across the trials, thus, I was getting slightly varying plots of Figure 7. So, for comparing the effect of noise (i.e., system size) on various kinds of HCOs, I plotted the correlations of multiple iterations (n=25) on a semi-log scale plot. The result is shown in Figure 8.

(Note : Correlation values in the scatterplot in Figure 8 contain a small jitter in X-axis direction, to better visualize the points and variabilities without affecting their Y-axis values.)

Figure 8 quantifies the synchronization between two neuron cells simulated using deterministic and stochastic methods while varying noise levels. The correlation values are compared across different methods of finding correlation (i.e., raw traces without gaussian smoothing and with gaussian smoothing using various kernel sizes). Smaller kernel (W=50) gives less variable results across multiple iterations. Positive values show in-phase synchronization and negative values here suggest out-of phase synchronization. The absolute value suggests the strength of the coupling between neurons in the HCO. Following the raw traces and binless correlation (W=50), it is apparent that reducing noise levels (i.e, higher $log_{10}(\Omega)$ values) give stronger correlation of the oscillators and deterministic simulation gives similar results with small noise levels, as one would expect. More discussion on variabilities due to bin-width and counter-intuitive trends seen in bursting type inhibitory-HCO (in purple) is done in the following section (See Section 4).

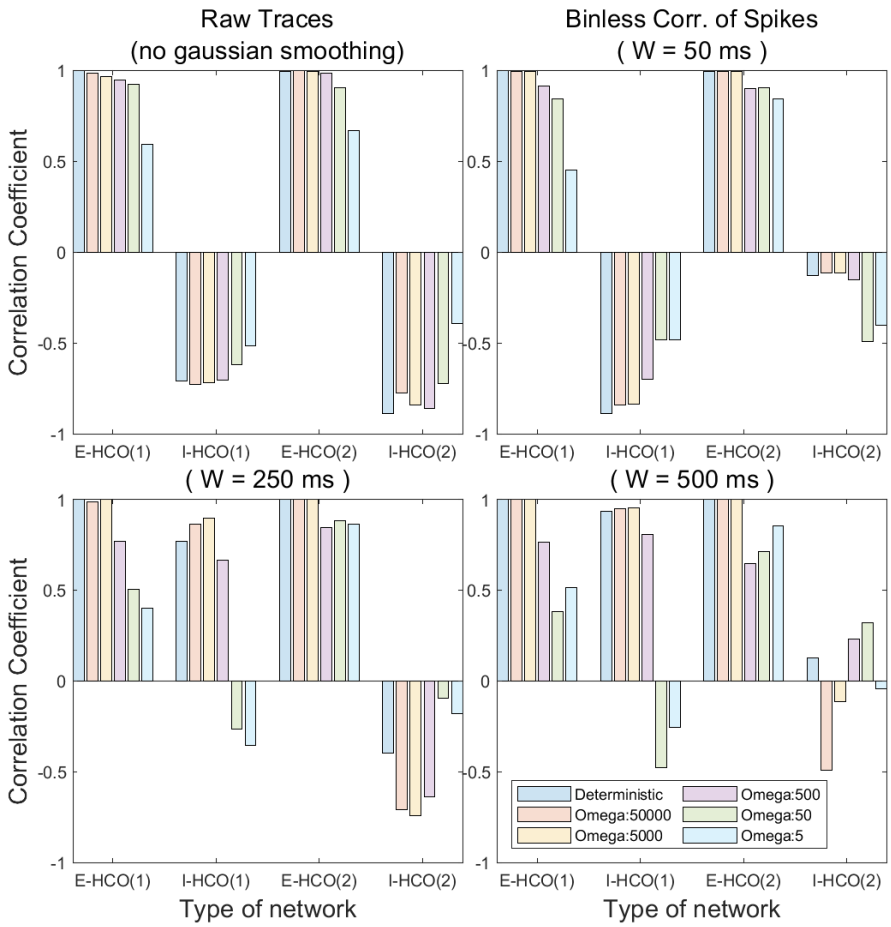


Figure 7: Single iteration values of spike train correlation for various kinds of HCOs and for different system sizes.

4 Discussion and Conclusions

1. **Higher binwidth in binless correlation leads to more variabilities and inflated correlation values.**

Figure 8 shows correlation comparison for different HCOs computed using various methods in 25 iterations. Firstly, variations of the values across 25 iterations are more when the gaussian kernel used is wider (e.g., W=250, 500, etc.) and also with this method, more values show positive correlation as compared to negative correlation. This may be the case because wider kernels would overlap more for neighboring spikes and show an overall positive (i.e., in phase) correlation as opposed to anti-phased correlation. Higher variabilities indicate that gaussian smoothing using a larger gaussian kernel would make correlation coefficient more sensitive for the relative spike position. For example, if there are more than one spike present close-by, this method would give very high signal for them (see in Figure 6), and ultimately this becomes an issue as the gaussian smoothed signal would vary quite a lot based on randomness in spike position.

2. **Binless correlation (with small width kernel) is more efficient in predicting correlation for non-bursting HCOs and excitatory-HCOs.**

While comparing raw traces and 50ms window results, one can see that gaussian smoothing works better for non-bursting type HCOs (in blue, yellow). That is because raw correlation depends on the actual shape of the spikes (see Figure 4a) and binless correlation only uses identified spike-times. So, with higher noise in simulation (i.e., low omega), a spike in one neuron may fail to cause a spike in another neuron, yet the shape

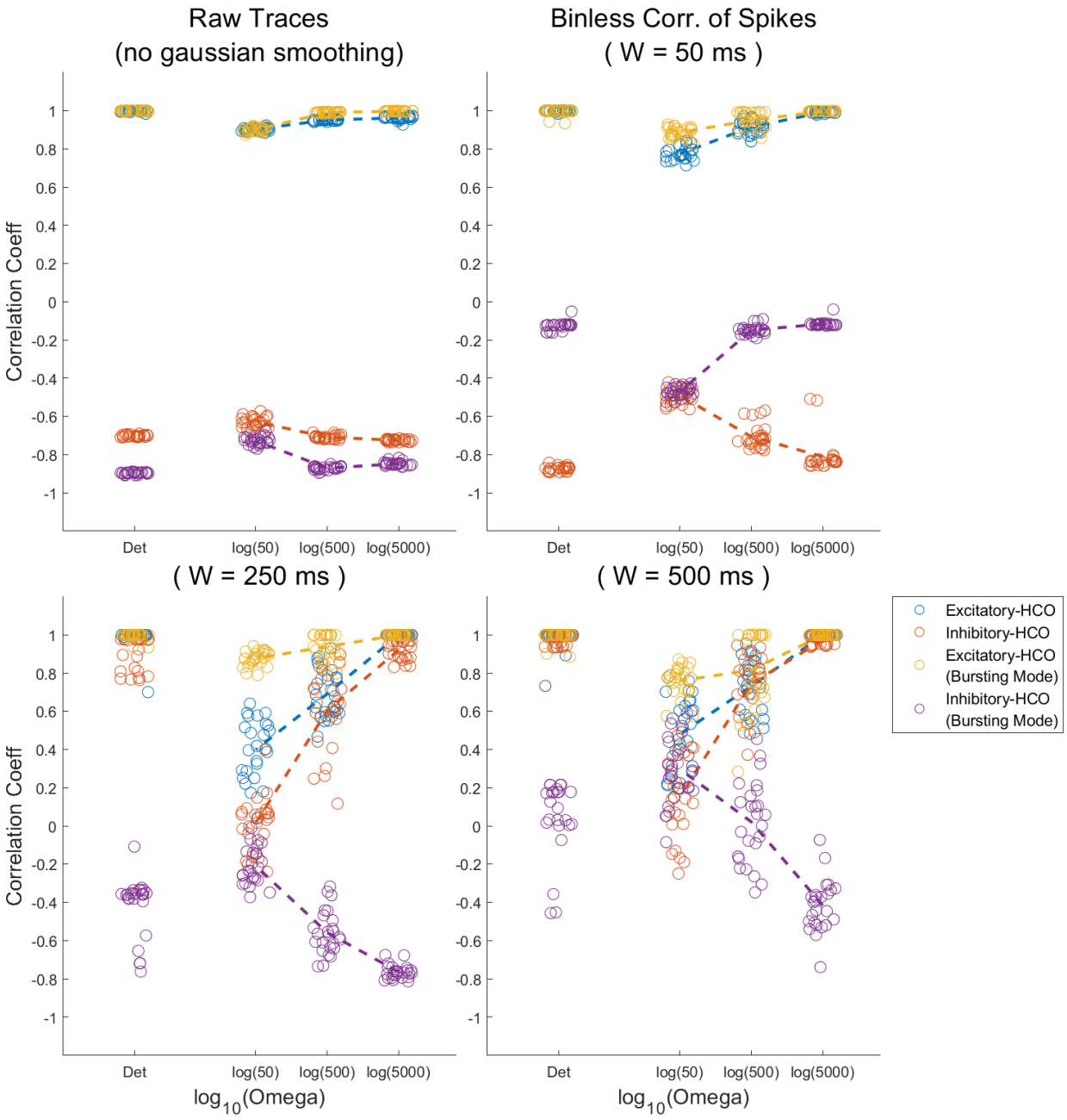


Figure 8: Comparing binless correlations in different types of HCOs for multiple runs (n=25).

of the raw trace would change which may lead the raw-signal method to count higher higher correlation than spike-time based method. Also due to noise levels, if the peaks are not aligned but the overall spike shapes are matching, the same thing may happen. Thus, here binless correlation with identified spike-times is more useful for predicting correlation given certain noise levels.

3. **For bursting type inhibitory-HCO binless correlation with "wider-width" kernels works the best. Binless correlation with small-width kernel is in-efficient and raw-traces method completely disregard spike-times in favor of slow *plateau potential*.**

See bursting inhibitory-HCO (in Purple) in Figure 8, for which the trend looks very different between

raw-traces and 50ms kernel methods. This is mainly because the raw-traces method would largely influenced by the *plateau potential* due to slow internal bursting current, whereas in the binless correlation method the only identified spike-times would be used due to the pattern our bursting-ML neurons use, the fast current feedback (i.e., a spike) would vary a lot based on the coupling type and noise level. Thus, the binless method shows it more like un-synchronized (i.e., values close to zero) for inhibitory HCOs and raw-traces method show it as anti-phased synchronized, the result largely driven by the slow *plateau potentials*.

However, a wider-width kernel would disregard the exact spike-times but still works better for inhibitory-bursting-HCOs (see W=250, Figure 8). This is because the *plateau potentials* itself oscillates (i.e., turns on-and-off) and the spikes are segregated in chunks (see 4b), so the gaussian smoothed traces with wider kernels would also oscillate in anti-phase. So, one can see why the wider-kernel is more efficient in predicting correlation for bursting type inhibitory-HCOs.

4. **Simple oscillator made using two ML neurons shows synchronized oscillations.**

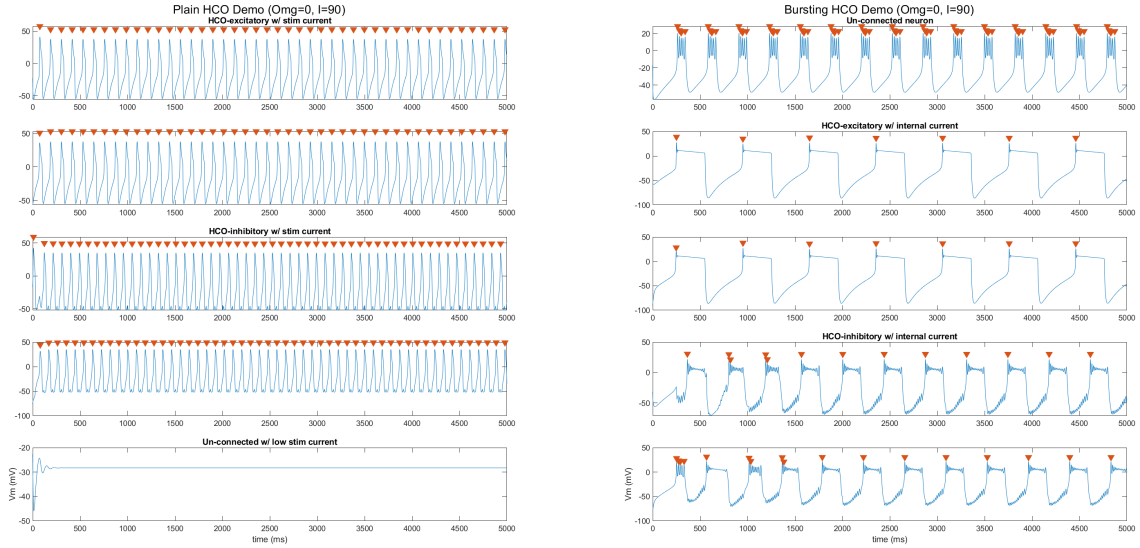
At the very least, the simple altered ML neurons show synchronous firing when coupled through any type of synapse, creating a simple oscillator. Excitatory-HCOs show stronger correlations as compared to inhibitory-HCOs. For a sustained firing, and in-turn, achieve synchrony, ML neurons either need slow internal feedback current or constant external stimulus current.

5. **Excitatory-HCOs show in-phase synchronization and inhibitory-HCOs show ant-phase synchronization. Reducing Noise improves the degree of synchrony.**

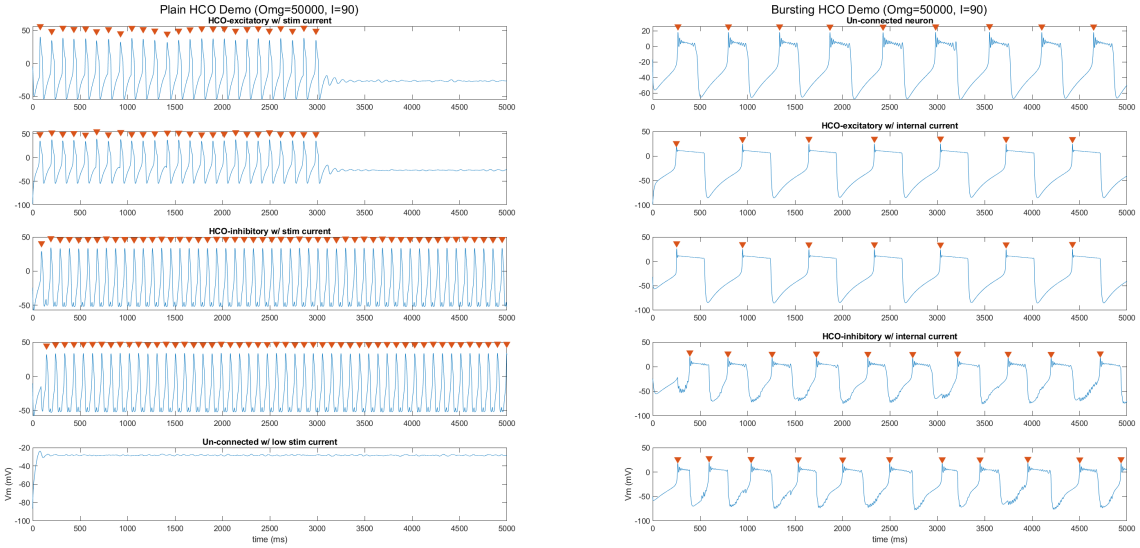
As explained earlier in discussion points 2 and 3, synchronization for non-bursting-HCOs and excitatory-bursting-HCOs should be compared using binless correlation method with 50ms kernel (upper right, Figure 8)) and for bursting type inhibitory-HCO, I would use 250ms kernel (lower left, Figure 8)). Comparing the correlation coefficients, qualitatively, across various HCOs it appears that excitatory-HCOs (bursting and non bursting alike) show in-phase correlation, with bursting type HCO being higher tolerant to system noise levels. Similarly, inhibitory-HCOs show anti-phase synchronization.

A Appendix A : Single-Run Raw Plots (Ω and HCO types)

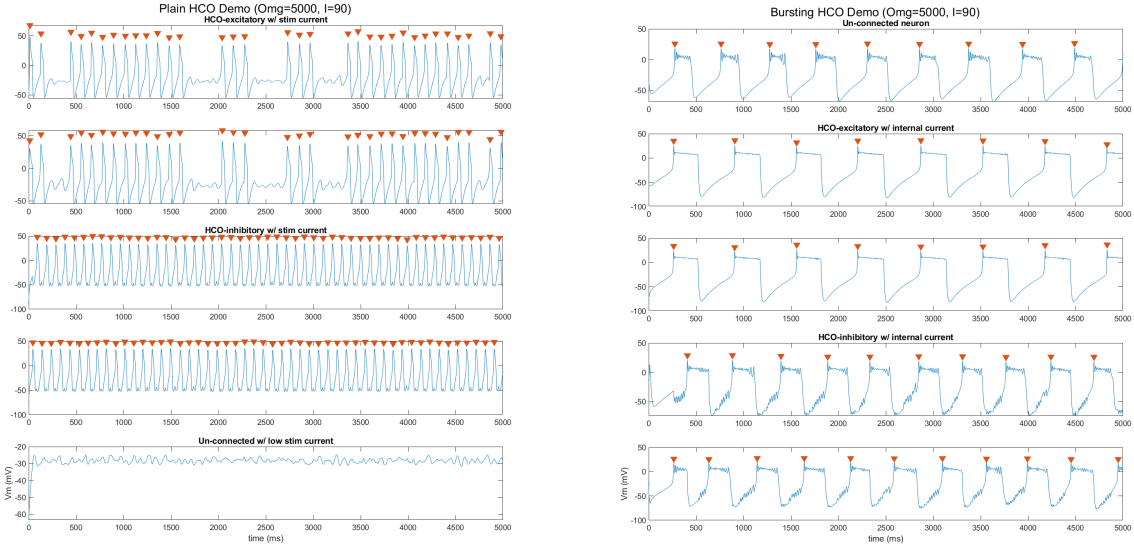
These plots are the raw voltage traces for each neurons in all HCOs, which are used to compute single run correlations in Figure 7.



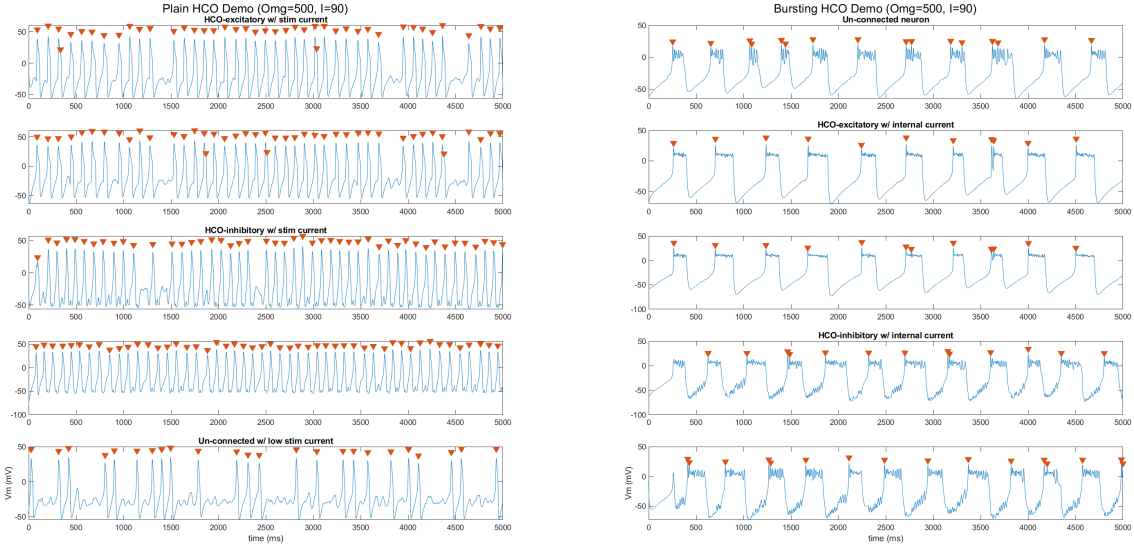
(a) Deterministic HCO



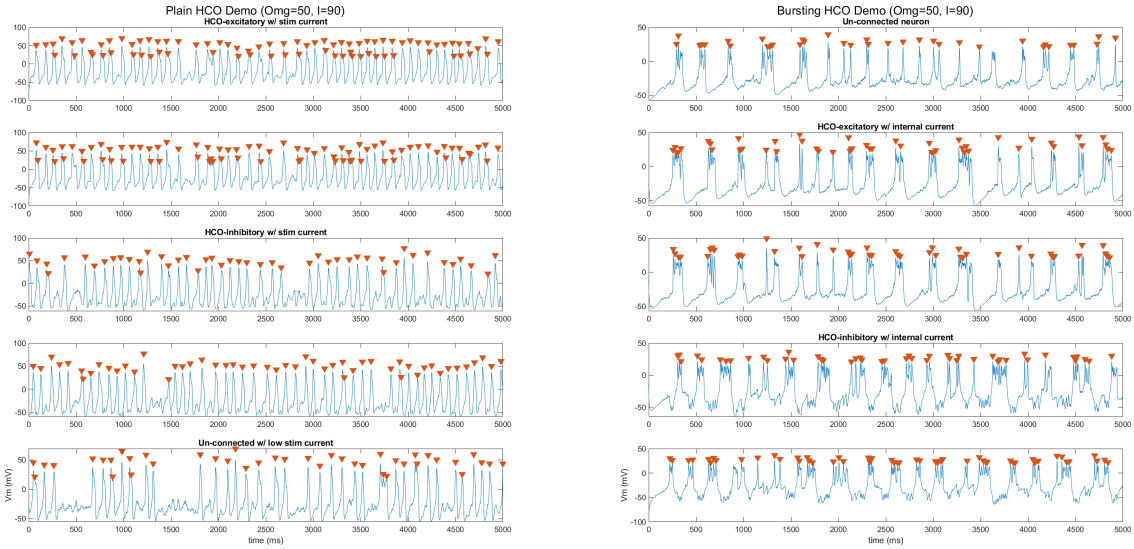
(b) Stochastic HCO with Omega=50,000.



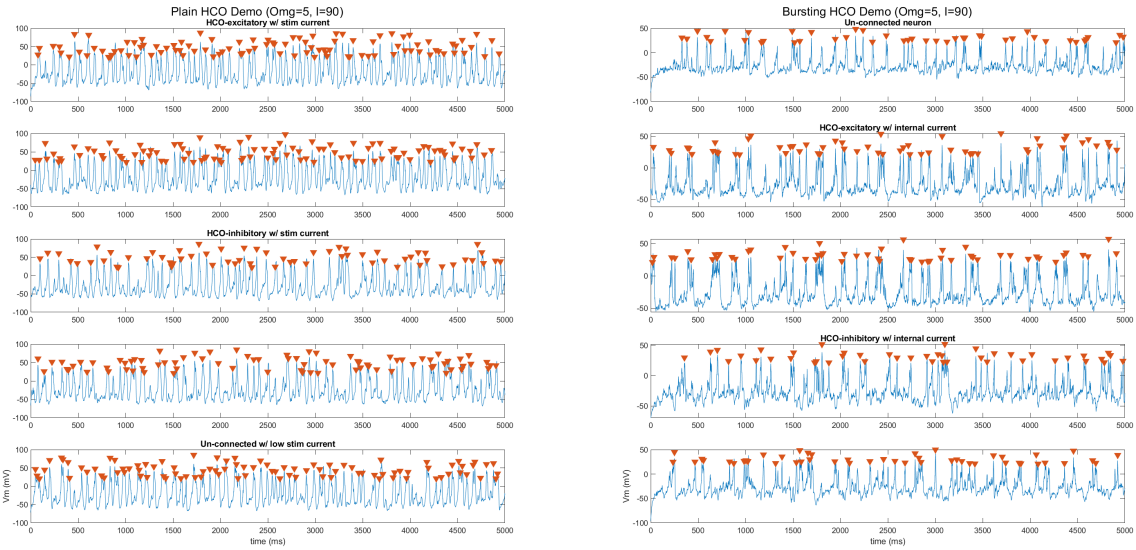
(c) Stochastic HCO with Omega=5000.



(d) Stochastic HCO with Omega=500.



(e) Stochastic HCO with Omega=50.



(f) Stochastic HCO with Omega=5.

Figure 9: Single-Run Raw Plots

B Appendix B : Codes

B.1 Matlab code for simulation

B.1.1 ML_neurons : Parallel simulation of an array of neurons connected in a network

```
function [Vm,nK,nCa,Iint] = ML_neurons(det,dt,Istim,Iint,omega,nK,nCa,Vm,
    PreSynVm,synapse,burst)
%% Each passing var (except dt) can be an array

% ---- Setting up neuron cellular parameters ----
% Model parameters for each neurons (from (Anderson et. al., 2015))
vCa = 120;                  % Rev.Pot for Calcium channels
gCa = 4.4;                  % Calcium conductance
vK  = -84;                  % Rev.Pot for Potassium channels
gK  =   8;                  % Potassium conductance
vL  = -60;                  % Rev.Pot for leak channels
gL  =   2;                  % Leak channels conductance
Cm  =  20;                  % Membrane Conductance

% Defining synapse  (from Z.Yu, PJT, 2020)
vSyn = 100*synapse;           % Make Rev.pot. neg for inhibitory synapse
gSyn = 1*abs(synapse);      % Make conductance zero for no-synapse
synThr = 5; synSlope = 0.5;          % sigmoid activation fxn parameters
Syn = @(v) 0.5*(1+tanh((v-synThr)/synSlope));
% Syn = @(v) 1./(1 + exp(-(synSlope/2).*(v-synThr)));

% Bursting
% Slow current feedback for bursting
epsi = .01; v0 = -26;
dIint = @(v) epsi*(v0-v).*burst;

% K+ ion channel parameters
% vc = 12; vd = 17.4; phi_n = 0.23; % Adapted for bursting (from mlsqr)
% vc = 2; vd = 30; phi_n = 0.04; % For simple (non-bursting) neurons
vc = 2 + burst*10; vd = 30 - burst*12.6; phi_n = 0.04 + burst*0.19;
xi_n    = @(v) (v-vc)./vd;                    % scaled argument for n-gate
    input
ninf    = @(v) 0.5*(1+tanh(xi_n(v)));        % n-gate activation function
tau_n   = @(v) 1./(phi_n.*cosh(xi_n(v)/2));  % n-gate activation t-const
alpha_n = @(v) ninf(v)./tau_n(v);            % per capita opening rate
beta_n  = @(v) (1-ninf(v))./tau_n(v);        % per capita closing rate

% Ca2+ ion channel parameters
va = -1.2; vb = 18; phi_m = 2;
xi_m    = @(v) (v-va)/vb;                    % scaled argument for m-gate input
minf    = @(v) 0.5*(1+tanh(xi_m(v)));        % m-gate activation function
tau_m   = @(v) 1./(phi_m*cosh(xi_m(v)/2));   % m-gate time constant
alpha_m = @(v) minf(v)./tau_m(v);            % per capita opening rate
beta_m  = @(v) (1-minf(v))./tau_m(v);        % per capita closing rate

% ---- Setting up Chemical Langevin Equation ----
tau = dt;
% stochasting K+ channels
omega_K = omega;
```

```
No = nK.*omega_K;
dWk = randn(size(No)); % N(0,1)
h1k = @(v,X) alpha_n(v).*(omega_K-X);
h2k = @(v,X) beta_n(v).*(X);
N_next = @(v,X) X + tau.*(h1k(v,X) - h2k(v,X)) ...
    + sqrt(tau.*(h1k(v,X) + h2k(v,X))).*dWk;

% stochasting Ca2+ channels
omega_Ca = omega;
Mo = nCa.*omega_Ca;
dWca = randn(size(Mo)); % N(0,1)
h1ca = @(v,X) alpha_m(v).*(omega_Ca-X);
h2ca = @(v,X) beta_m(v).*(X);
M_next = @(v, X) X + tau.*(h1ca(v,X) - h2ca(v,X)) ...
    + sqrt(tau.*(h1ca(v,X) + h2ca(v,X))).*dWca;

% Enforce valid limits
lim = @(n,Low,Max)(n>Max).*Max + (n<Low).*Low + (n<=Max & n>=Low).*n;

% Simulation using CLE or Deterministic eqns

% Update ion channels counts and fraction for each neuron
if det==1                          % ( FOR DETERMINISTIC SIM )
    nK = nK + dt*(ninf(Vm)-nK)./tau_n(Vm);
    nCa = minf(Vm);
else                               % ( FOR STOCHASTIC SIM )
    No = N_next(Vm,No);
    Mo = M_next(Vm,Mo);
    No = lim(No,zeros(size(No)),omega_K);
    Mo = lim(Mo,zeros(size(Mo)),omega_Ca);
    nK  = No./omega_K;
    nCa = Mo./omega_Ca;
end
Iint = Iint + dt*dIint(Vm);

% Update membrane voltage for each neuron
dV = (dt/Cm).*( Istim + Iint ...
    - gL*(Vm-vL) ...
    - gK*nK.*(Vm-vK) ...
    - gCa*nCa.*(Vm-vCa) ...
    - gSyn.*Syn(PreSynVm).*(Vm-vSyn) );
Vm = Vm + dV;

end
```

B.1.2 ML_network : Initializing and simulating a network of ML neurons

(Note : "Demo=1" option in this function gives a demo of all HCO types.)

```
function [V,t,spikes,X] = ML_network(demo, det, n, seed)
%%    (for Demo) : [V,t,spikes] = ML_network(1,0);  % Stochastic demo
%     (for Demo) : [V,t,spikes] = ML_network(1,1);  % Deterministic demo
%     (from fxn) : [V,t,spikes] = ML_network(2,det,n);
%     (other..)  : [V,t,spikes] = ML_network(0);  % (Uses network @ line -27)
%
```

```
% Actual ML neuron is the function "ML_neurons" (at the very end; line# ~150)

% Read Network Design
if demo == 1
    % Demo: 1,2 : HCO-excitatory w/ stim current
    %       3,4 : HCO-inhibitory w/ stim current
    %       5   : Un-connected w/ low stim current
    %       6   : Busrting neuron (w/ NO stim-current) w/ NO synapse
    %       8,7 : HCO-excitatory w/ Busrting neuron
    %      10,9 : HCO-inhibitory w/ Busrting neuron
    % Neuron #  1  2  3  4  5  6  7  8  9  10
    net     = [ 2, 1, 4, 3, 0, 0, 8, 7,10, 9]; % Neuron connection
    synapse = [ 1, 1,-1,-1, 0, 0, 1, 1,-1,-1]; % Synapse : inhi/exci/off
    burst   = [ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]; % Burst mode : on/off
    if det==1
        Istim=[90,90,90,90,88, 0, 0, 0, 0, 0]; % Current-clamp (nA)
    else
        Istim=[85,85,85,85,80, 0, 0, 0, 0, 0]; % Current-clamp (nA)
    end
    dt = 0.1;  tmax = 3e3; system_size = 1e3; seed = 10;

elseif demo == 0
    % Network Design
    net     = []; % Neuron connection
    synapse = []; % Synapse : inhi/exci/off
    Istim   = []; % Current-clamp (nA)
    burst   = []; % Burst mode : on/off
    system_size = []; % noise (array or single value)
    tmax = 3e3;
    dt = 0.1;
else
    % read data from 'n' variable
    net=n.net; synapse=n.synapse; Istim=n.Istim; burst=n.burst;
    system_size=n.system_size; tmax=n.tmax; dt=n.dt; demo=n.demo;
end

% Simulation
% Initialization
if exist('seed','var'), rng(seed); end
sz = size(net);
omega = system_size.*ones(sz);
Vm   = -100*rand(sz); % Start with random voltage
nK   = rand(sz); % Start with random fraction
nCa  = rand(sz); % Start with random fraction
Iint = 0*ones(sz);
t = (0:dt:tmax)';
synapse(net==0) = 0; % turn-off synapse for unconnected neurons
net(net==0) = find(net==0); % (note-1)
V = nan(length(t),sz(2));
M = nan(length(t),sz(2));
N = nan(length(t),sz(2));

% Find membrane voltage at each time-step
for i=1:length(t)
```

```
61      [Vm,nK,nCa,Iint] = ML_neurons(det, dt, Istim, Iint, omega, ...
62          nK, nCa, Vm, Vm(net), synapse, burst);
63      V(i,:) = Vm; M(i,:) = nCa; N(i,:) = nK;
64  end
65  % Save Ion-channels state
66  X.M = M; X.N = N; X.Omega = system_size; X.t = t;
67
68  % Identify Spikes
69  spikeThr = 15;  % mV; Neural spike detection threshold
70  refTh = 15/dt; % ms->Steps; Refractory period for spikes
71  spikes=cell(sz(2),3);
72
73  for i=1:sz(2)
74      % Find Peaks
75      xval=V(:,i);
76      len_x = length(xval); val=[]; index=[]; xi=2;  % start at second data
            point
77      while xi < len_x-1
78          if xval(xi) > xval(xi-1)
79              if xval(xi) > xval(xi+1)                    % definite max
80                  val =[val xval(xi)];
81                  index = [ index xi];
82              elseif xval(xi)==xval(xi+1) && xval(xi)==xval(xi+2) % 'long' flat
                    spot
83                  xi = xi + 2;                        % skip 2 points
84              elseif xval(xi)==xval(xi+1)                 % 'short' flat spot
85                  xi = xi + 1;                        % skip one point
86              end
87          end
88          xi = xi + 1;
89      end
90      % Filter Peaks
91      index(val<=spikeThr) = [];          % Apply Vm threshold
92      val(val<=spikeThr) = [];
93      ind=2;
94      while ind <= length(index)          % Apply refractory period
95          if abs(index(ind)-index(ind-1)) < refTh
96              index(ind)=[];
97              val(ind)=[];
98          else
99              ind=ind+1;
100         end
101     end
102     % Save Spikes
103     spikes{i,1} = index;                % Save Spike index
104     spikes{i,2} = val;                  % Save Spike amplitude (mV)
105     spikes{i,3} = t(index);             % Save Spike time (ms)
106 end
107
108 % Plot traces
109 if demo == 1
110         % plot Demo network
111     figure
112     suptitle("Plain HCO Demo (Omg="+system_size+", I="+Istim(1)+")")
```

```
113         for i = 1:5
114             subplot(5,1,i)
115             plot(t,V(:,i))
116             hold on
117             scatter(spikes{i,3}, 1.4*spikes{i,2}, 55, 'v', 'filled')
118             if i==1, title('HCO-excitatory w/ stim current');
119             elseif i==3, title('HCO-inhibitory w/ stim current');
120             elseif i==5, title('Un-connected w/ low stim current');
121             end
122         end
123         ylabel('Vm (mV)')
124         xlabel('time (ms)')
125 
126         figure
127         suptitle("Bursting HCO Demo (Omg="+system_size+", I="+Istim(1)+")")
128         for i = 1:5
129             subplot(5,1,i)
130             plot(t,V(:,i+5))
131             hold on
132             scatter(spikes{i+5,3}, 1.4*spikes{i+5,2}, 55, 'v', 'filled')
133             if i==2, title('HCO-excitatory w/ internal current');
134             elseif i==4, title('HCO-inhibitory w/ internal current');
135             elseif i==1, title('Un-connected neuron');
136             end
137         end
138         ylabel('Vm (mV)')
139         xlabel('time (ms)')
140 
141     elseif demo == 0
142         % all neurons in one plot
143         figure
144         suptitle('Vm (mV) vs  time for each neuron')
145         for i = 1:sz(2)
146             subplot(sz(2),1,i)
147             plot(t,V(:,i))
148             hold on
149             scatter(spikes{i,3}, 1.4*spikes{i,2}, 55, 'v', 'filled')
150         end
151         xlabel('time (ms)')
152     end
153 
154     % NOTES:
155     %
156     % (1) un-connected neurons get a dummy connection to Neuron1, for valid
            indexing.
157     % I have made sure that this dummy connection does not affect in anyway,
158     % because the synapse is forcefully "turned-off" in the previous line.
159     end
```

B.1.3 BinlessCorrelation : Plotting gaussian smoothed spike-trains and computing cross-correlation between two traces.

```
1 function [C,gs] = GetBinlessCorr(spk_ind_1,spk_ind_2,W,t,dt,plt,ttl)
2 %%
```

```
3
4  gs.t=t;
5  gs.W=W;
6
7  % Get spike train in bool values
8  sz = size(gs.t);
9  if isempty(spk_ind_1) || isempty(spk_ind_2)
10     C=0; return;
11 end
12 spike_train_1 = false(sz);
13 spike_train_1(spk_ind_1) = true;
14 spike_train_2 = false(sz);
15 spike_train_2(spk_ind_2) = true;
16
17 % Defining the gaussian kernel
18 gs.sd = gs.W/sqrt(12)/dt;   % SAMPLES; SD of Gaussian Curve
19 t_window = -gs.sd*6:1:gs.sd*6; % 6*sd window
20 gs.win = 1/(sqrt(2*pi)*gs.sd) * exp(-(t_window.^2)/(2*gs.sd^2));
21 gs.tw = t_window*dt;
22 gs.sd = gs.sd*dt;
23
24 % Convolute it with a gaussian kernel
25 gs.V1 = conv(spike_train_1,gs.win,'same');
26 gs.V2 = conv(spike_train_2,gs.win,'same');
27
28 % Find correlation coeff for gaussian smoothed signal
29 meanV1 = mean(gs.V1); meanV2 = mean(gs.V2);
30 Cov12 = (gs.V1 - meanV1).*(gs.V2 - meanV2);
31 Var1 = (gs.V1 - meanV1).^2;
32 Var2 = (gs.V2 - meanV2).^2;
33 C = sum(Cov12) / ( sqrt(sum(Var1))*sqrt(sum(Var2)) );
34
35 if exist('plt','var')
36     if plt==0 || plt==2
37         figure;
38         plot(gs.tw,gs.win);
39         title("Gaussian Window (+/- 6*SD) W="+gs.W+"ms");
40         xlabel('time(ms)'); ylabel('Amp');
41     end
42     if plt==1 || plt==2
43         figure;
44         plot(gs.t,gs.V1); hold on; plot(gs.t,gs.V2);
45         xlabel('time(ms)'); ylabel('Amp');
46         title("Gaussian Smoothed Neural spikes  W="+gs.W+"ms")
47         legend({'Neuron-1','Neuron-2'})
48     end
49     if plt==5
50         figure;
51         plot(gs.t,gs.V1); hold on; plot(gs.t,gs.V2);
52         xlabel('time(ms)'); ylabel('Amp');
53         title(ttl)
54         legend({'Neuron-1','Neuron-2'})
55     end
56
```

```
end

end
```

B.2 Other things : LaTeX, Figures, Codes

- All Matlab simulation script, Matlab scripts for all figures, tex/md files for generating this report and figures files are available here :
  - <Google Drive>
  - <TODO: Add Github>

References

Anderson, David F., Bard Ermentrout, and Peter J. Thomas. 2015. "Stochastic Representations of Ion Channel Kinetics and Exact Stochastic Simulation of Neuronal Dynamics." *Journal of Computational Neuroscience* 38 (1): 67–82. https://doi.org/10.1007/s10827-014-0528-2.

CDC. 2018. "Epilepsy." https://www.cdc.gov/epilepsy/about/fast-facts.htm.

Ermentrout, G. Bard, and David H. Terman. 2010. *Mathematical Foundations of Neuroscience*. Vol. 35. Interdisciplinary Applied Mathematics. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-87708-2.

González-Miranda, J. M. 2014. "Pacemaker Dynamics in the Full MorrisLecar Model." *Communications in Nonlinear Science and Numerical Simulation* 19 (9): 3229–41. https://doi.org/10.1016/j.cnsns.2014.02.020.

Higham, Desmond J. 2008. "Modeling and Simulating Chemical Reactions." *SIAM Review* 50 (2): 347–68. https://doi.org/10.1137/060666457.

Izhikevich, Eugene M. 2006. "Bursting." *Scholarpedia* 1 (3): 1300. https://doi.org/10.4249/scholarpedia.1300.

Jutras, Michael J, and Elizabeth A Buffalo. 2010. "Synchronous Neural Activity and Memory Formation." *Current Opinion in Neurobiology*, Cognitive neuroscience, 20 (2): 150–55. https://doi.org/10.1016/j.conb.2010.02.006.

Kruskal, Peter B., Jessica J. Stanis, Bruce L. McNaughton, and Peter J. Thomas. 2007. "A Binless Correlation Measure Reduces the Variability of Memory Reactivation Estimates." *Statistics in Medicine* 26 (21): 3997–4008. https://doi.org/10.1002/sim.2946.

Lecar, Harold. 2007. "Morris-Lecar Model." *Scholarpedia* 2 (10): 1333. https://doi.org/10.4249/scholarpedia.1333.

Mainen, Zachary F., and Terrence J. Sejnowski. 1996. "Influence of Dendritic Structure on Firing Pattern in Model Neocortical Neurons." *Nature* 382 (6589): 363–66. https://doi.org/10.1038/382363a0.

Mormann, Florian, Thomas Kreuz, Ralph G Andrzejak, Peter David, Klaus Lehnertz, and Christian E Elger. 2003. "Epileptic Seizures Are Preceded by a Decrease in Synchronization." *Epilepsy Research* 53 (3): 173–85. https://doi.org/10.1016/S0920-1211(03)00002-0.

Morris, C., and H. Lecar. 1981. "Voltage Oscillations in the Barnacle Giant Muscle Fiber." *Biophysical Journal* 35 (1): 193–213. https://doi.org/10.1016/S0006-3495(81)84782-0.

Pikovsky, Arkady, and Michael Rosenblum. 2007. "Synchronization." *Scholarpedia* 2 (12): 1459. https://doi.org/10.4249/scholarpedia.1459.

Rinzel, John, and G. Bard Ermentrout. 1998. *Chapter-7 : Methods in Neuronal Modeling: From Ions to Networks*. Edited by Christof Koch and Idan Segev. 2nd ed. Computational Neuroscience. Cambridge, Mass: MIT Press.

Van Vreeswijk, Carl, L. F. Abbott, and G. Bard Ermentrout. 1994. "When Inhibition Not Excitation Synchronizes Neural Firing." *Journal of Computational Neuroscience* 1 (4): 313–21. https://doi.org/10.1007/BF00961879.

Victor, Jonathan D., and Keith P. Purpura. 1997. "Metric-Space Analysis of Spike Trains: Theory, Algorithms and Application." *Network: Computation in Neural Systems* 8 (2): 127–64. https://doi.org/10.1088/0954-898X_8_2_003.

WHO. 2020. "Epilepsy." https://www.who.int/westernpacific/health-topics/epilepsy.

Wilkinson, Darren James. 2019. *Stochastic Modelling for Systems Biology*. Third edition. Chapman & Hall/CRC Mathematical and Computational Biology. Boca Raton: CRC Press, Taylor and Francis Group.

Yu, Zhuojun, and Peter J. Thomas. n.d. "Dynamical Consequences of Sensory Feedback in a Half-Center Oscillator Coupled to a Simple Motor System." Unpublished.