# Modeling ion channel dynamics via exact stochastic techniques

Sara Baldinelli[1], Letizia De Pietri[1], Roan Spadazzi[1]

[1] *University of Trento, Trento 38123, Italy*

## Introduction

Membrane potential is the difference in electric charge across the membrane of a cell and is a key driver for excitable cells like neurons **[1]**. Changes in neuronal membrane potential are partially caused by the random opening and closing of voltage-gated ion channels (mainly Potassium and Calcium). This biophysical process is relevant to model as it allows to investigate the electrical dynamics at the basis of neuronal functioning, signaling and disease **[2]**. These phenomena can be effectively simulated by using hybrid frameworks that integrate continuous differentiable components, like membrane voltage, together with discrete components, such as the number of open/close ion channels. Such systems are usually modeled through piecewise-deterministic Markov processes (PDMP), where continuous changes are represented by Ordinary Differential Equations (ODEs), and discrete variations by stochastic mechanisms together with rate functions and transition measures.
The paper, on the other hand, presents a different approach: two pathwise stochastic representations with stochastic counting processes implying exact simulation algorithms **[3]**. This matters as, previously, in the neuroscience field, only approximate simulation methods have been used. When comparing the approximate method (PDMP) against the presented stochastic strategy, we will observe that the trajectories describing the behavior of the system diverge even though they are driven by identical noise sources. Using the method described by the paper such approximations could be avoided.
We will see that we can speed up the computations by employing an alternative, more recent, approach: the Rejection-based Stochastic Simulation Algorithm (RSSA) **[4]**. We prove that neuronal dynamics are well captured by this exact strategy and we suggest further developments for the description of the system.

## 1. Model Description & Implementation

This paper presents two different algorithms to model this system: a Random Time Change (RTC) representation and a "Gillespie" method **[5, 6]**, both with their corresponding numerical simulation strategies.

### 1.1 RTC representation

This representation simulates the opening and closing of ion channels as a Poisson process **[7]**. The switching of the ion channel states (open/close) can be modeled as events occurring at random intervals that are drawn from an exponential distribution.

$$\textbf{(1)} \quad X(t) = X_0 + \sum_k Y_k \left( \int_0^t \lambda_k(V(s), X(s))\, ds \right) \zeta_k \qquad \textbf{(2)} \quad C\frac{dV}{dt} = I_{\text{app}}(t) - I_V(V(t)) - \left( \sum_{i=1}^{d} g_i^o X_i(t) \right) (V(t) - V_X)$$

The first equation represents what value the component X assumes at time t. It sums the value of the initial state ($X_0$) with the contribution given by each reaction k (the counting process). This is calculated by integrating over the propensity function of reaction k (of jumping from one state to another, $\lambda_k(s, X(s))$) multiplied by the independent unit rate Poisson processes ($Y_k$) and the reaction vector ($\zeta_k$).
The second one shows the derivative of the voltage (V) in time following the Kirchhoff's current conservation law: it is composed of the applied current ($I_{\text{app}}(t)$), the deterministic voltage-dependent currents ($I_V(V(t))$) and the total conductance associated with the channel represented by vector X all divided by the capacitance (C).

### 1.2 Gillespie representation

This other representation directly models the opening and closing of ion channels in an iterative way. The waiting time between events is sampled from an exponential distribution and the transition propensities depend on the voltage and the current state.

**(3)** $X(t) = X(0) + \sum_{k=1}^{M} \zeta_k \int_0^t 1\left\{\xi_{R_0(s-)} \in [q_{k-1}(s-), q_k(s-)]\right\} dR_0(s)$ **(4)** $R_0(t) = Y\left(\int_0^t \lambda_0\big(V(s), X(s)\big)\, ds\right)$

The first equation represents the value of component X at time t. It is derived by summing up the initial value ($X_0$) to the contribution of each reaction k. The way it differs from the previous representation is the way the additional integral (in formula **3**) is characterized: the reaction will fire if the uniform random variable ($\xi_{R_0(s-)}$) falls within the range ($[q_{k-1}(s-), q_k(s-)]$) for reaction k. The second equation directly depicts the "holding time" ($R_0$) in each state, or how long it would last before the next reaction occurs. The derivative of the voltage is modeled as previously described in formula **2**.

### 1.3 Algorithm comparison

**Algorithm 1** Random Time Change

1: $X$ = Initial number of molecules of each species
2: $V$ = Initial voltage
3: $t = 0$
4: Set $t_{Max}$
5: For each k, $\tau_k = 0$, $T_k = 0$
6: **for** $k = 1$ to $M$ **do**
7: $\quad r_k = \sim U(0,1)$
8: $\quad \tau_k = ln(1/r_k)$
9: **end for**
10: **while** $t < t_{Max}$ **do**
11: $\quad$ Integrate forward in time until $\int_t^{t+\Delta} \lambda_k(V(s), X(s))ds = \tau_k - T_k$ holds with index $\mu$
12: $\quad$ **for** $k = 1$ to $M$ **do**
13: $\quad\quad T_k = T_k + \int_t^{t+\Delta} \lambda_k(V(s)X(s))ds$
14: $\quad$ **end for**
15: $\quad t = t + \Delta$
16: $\quad X \leftarrow X + \zeta_\mu$
17: $\quad$ Given $r = \sim U(0,1)$ set $\tau_\mu = \tau_\mu + ln(1/r)$
18: **end while**
19: **return** $X$

**Algorithm 2** Gillespie

1: $X$ = Initial number of molecules of each species
2: $V$ = Initial voltage
3: $t = 0$
4: Set $t_{Max}$
5: **while** $t < t_{Max}$ **do**
6: $\quad r = \sim U(0,1)$
7: $\quad$ Integrate forward in time until $\int_t^{t+\Delta} \lambda_0(V(s), X(s))ds = ln(1/r)$ holds
8: $\quad \xi = \sim U(0,1)$
9: $\quad$ Find k in $(1, ..., M)$ for which $\xi \in [q_{k-1}((t+\Delta)-), q_k((t+\Delta)-)]$
10: $\quad t = t + \Delta$
11: $\quad X \leftarrow X + \zeta_k$
12: **end while**
13: **return** $X$

The two algorithms share the initialization step of $X$, $V$, $t$ and $t_{Max}$. The first difference is the amount of random numbers generated between the two methods: the first algorithm uses, prior to the while loop, one random number per reaction ($r_k$, lines 6-9) and, inside it, another random number for each iteration ($r$, line 17). On the other side, the second algorithm generates two random variables per iteration ($r$ and $\xi$, lines 6 and 8). Assuming the same number of iterations, the second algorithm uses twice as many random numbers as the first one; since this system models few reactions (opening/closing of ion channels), we can consider negligible the quantity of random numbers produced in the for loop (lines 6-9) of algorithm 1. Another difference between the two is the way they advance in time ($\Delta$). The RTC relies on reaction-specific propensities ($\lambda_k$) with each reaction having its own stochastic timing, going forward until the first reaction fires (line 11) with index $\mu$. Having $\Delta$, the cumulative integral of the propensity for species k is updated ($T_k$, line 13) and so is the total waiting time for reaction $\mu$ ($\tau_\mu$, line 17).
The second algorithm, on the other hand, relies on the total propensity ($\lambda_0$). In this case, we do not have reaction-specific timings, but rather the total reaction time until the next event (line 7). We find the reaction k for which the randomly generated number $\xi$ falls within the $q$[1] interval up to time $t + \Delta$.

### 1.4 Applying the Morris-Lecar system

In order to model both the continuous voltage and the discrete number of open/closed ion channels the Morris-Lecar system **[8]** is considered. It models oscillations of barnacle muscle fibers, and the deterministic equations constituting it can be repurposed, together with a stochastic component, to study the evolution of this system. The random variables constituting this model will therefore be the voltage $V$ (-∞, +∞) and the number of open ion channels (0,1,2,...,$N_{tot}$), specifically Calcium ($M$) and Potassium ($N$). Lastly, the total number of ion channels is constant ($M_{tot}$, $N_{tot}$). The evolution of $V$, $M$ and $N$ is strictly linked.

[1] q is calculated by dividing the cumulative propensity up to reaction k by the total propensity. This gives the cumulative probability of one of the first k events occurring. The random number has to fall within the interval constituted by $q_{k-1}$ and $q_k$.

- $v_K = -84mV$
- $g_K = 8\mu S$
- $C = 20\mu F$
- $I_{app} = 100nA$
- $\phi_n = 0.04$
- $v_L = -60mV$
- $g_L = 2\mu S$
- $v_a = -1.2mV$
- $v_c = 2mV$
- $v_{Ca} = 120mV$
- $g_{Ca} = 4.4\mu S$
- $v_b = 18mV$
- $v_d = 30mV$
- $\phi_m = 0.4$

The $v_K$, $v_Ca$ and $v_L$ parameters represent the reversal potential of Potassium (K), Calcium (Ca) and leak (L); $g$ parameters are the maximum conductance of the ions and leak current; $C$ is the capacitance; $I_{app}$ the applied current (constant); $v_a$, $v_c$, are half activation points for Ca and K dynamics and $v_b$, $v_d$ the slopes; $\phi_n$ and $\phi_m$ scaling factors.

**(5)** $\frac{dV}{dt} = F(V(t),N(t),M(t)) = \frac{1}{C}\left(I_{\text{app}} - g_L(V(t)-v_L) - g_{\text{Ca}}\frac{M(t)}{M_{\text{tot}}}(V(t)-v_{\text{Ca}}) - g_K\frac{N(t)}{N_{\text{tot}}}(V(t)-v_K)\right)$

**(6)** $M(t) = M(0) - Y_{\text{close}}\left(\int_0^t \beta_m(V(s))M(s)\,ds\right) + Y_{\text{open}}\left(\int_0^t \alpha_m(V(s))\big(M_{\text{tot}} - M(s)\big)\,ds\right)$

**(7)** $N(t) = N(0) - Y_{close}\left(\int_0^t \beta_n(V(s))N(s)ds\right) + Y_{open}\left(\int_0^t \alpha_n(V(s))(N_{tot} - N(s))ds\right)$

Equation **5** is the applied version of equation **2**, representing the Kirchhoff's current conservation law with two ion channels (potassium and calcium) and leak, defining how the membrane potential ($V$) changes over time. While $V$ evolves continuously in time, the number of open Calcium and Potassium channels $M$ and $N$ changes only of one unit in a discrete way. Consequently, equations **6** and **7**, are explicit versions of equation **1**. Gates close and open at rates proportional respectively to the rate constants ($\beta$ and $\alpha$) multiplied by the available channels. Logically, the closing propensity depends on the amount of open channels (e.g $M(s)$, for the Calcium) and the opening on the amount of closed channels (e.g. $M_{tot}$-$M(s)$).

## 2. Paper Results

Figure **1** depicts the trajectory generated by the RTC algorithm with the full Morris-Lecar model considering two ion channels and leak. Given a constant external current $I_{app}$, we observe a sequence of noise dependent spikes. We can see that rising voltage corresponds to the opening of both ion channels and vice-versa. The chosen constants $I_{app}$ = $100nA$, $V$ = $-50mV$, $M_{tot} = N_{tot} = 40$, put the system in a periodic behavior as a stable limit cycle: independently on the initial conditions, it will converge towards the oscillatory behavior typical of neurons and excitable entities (Figure **6** in Supplementary Material) **[9]**. The panel on the right displays this cyclic trend, also showing that the voltage evolves continuously while the number of open ion channels discretely.

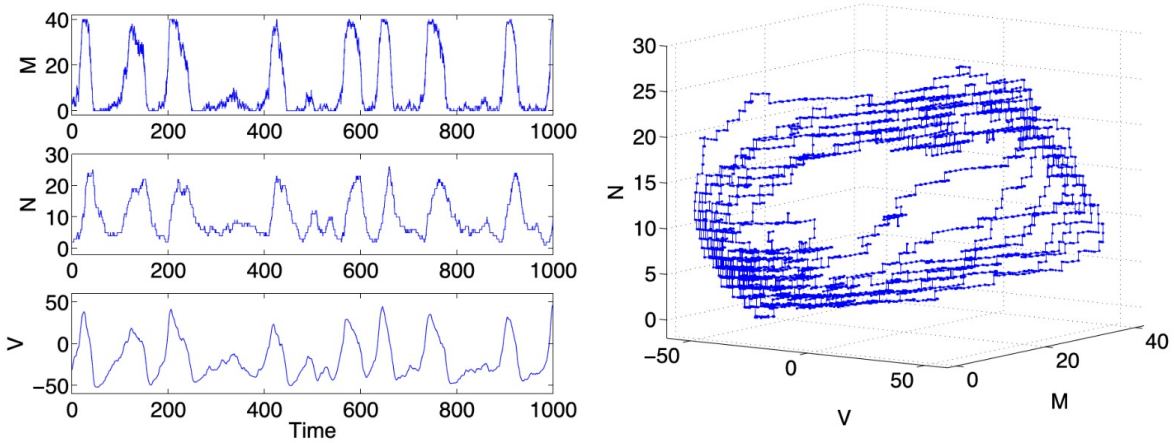


**Figure 1.** The figure shows the trajectory generated by the RTC algorithm by setting $M_{tot} = N_{tot} = 40$ channels, an applied current $I_{app} = 100nA$ and a voltage $V$ = $-50mV$. The first 2 panels on the left show, respectively, the number of open calcium channels ($M$) and the number of open potassium channels ($N$), over time. While, the bottom left panel, depicts the voltage ($V$), as a function of time. In the right panel, the trajectory is plotted in the (V,M,N) phase space.

This simulation was compared against an approximate method based on piecewise constant propensities. Indeed, it is "piecewise" because propensities are considered fixed between channel transition events, meaning the method is part of the

$\tau$-leaping algorithm family **[10]**. The current state is updated based on the propensity following the most recent jump rather than considering a time-dependent continuously changing voltage. This approximation leads to delayed dynamics visible in Figure **2**. Starting from the same initial conditions, the trajectories seemingly overlap at the beginning but, due to the error accumulation, they gradually diverge in time.

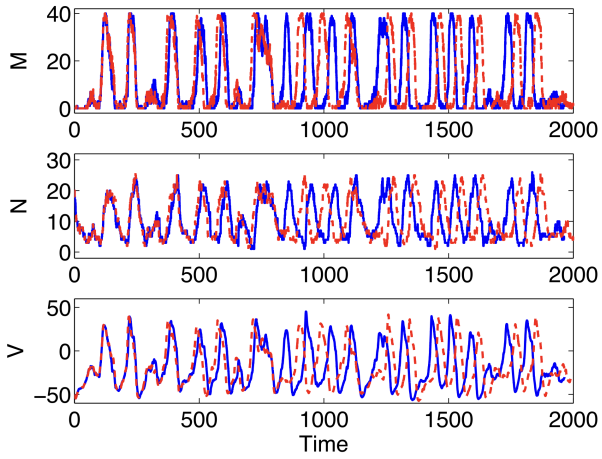


**Figure 2.** The figure represents the comparison between the exact algorithm (blue solid line) and the approximate piecewise constant method (red dashed line). It can be noticed that the two trajectories, starting from the same initial conditions, diverge as time increases.

To furtherly compare the two approaches, the number of open ion channels across different voltage values is shown in the histogram of Figure **3** at different amounts of $M_{tot}$ and $N_{tot}$. It can be seen that for lower values of $M_{tot}$ and $N_{tot}$ the difference in the histograms is clear. On the contrary, as these numbers increase, they appear more similar. This behavior could result from the fact that a larger system size implies higher propensities (note the presence of the number of ion channels in equations **6** and **7**). Knowing that $\tau$-leaping approximations work by assuming such high values, it is reasonable that the histograms are alike. On the contrary, a low number of ion channels leads to a system with a high stochasticity that cannot be fully captured by an approximate method.

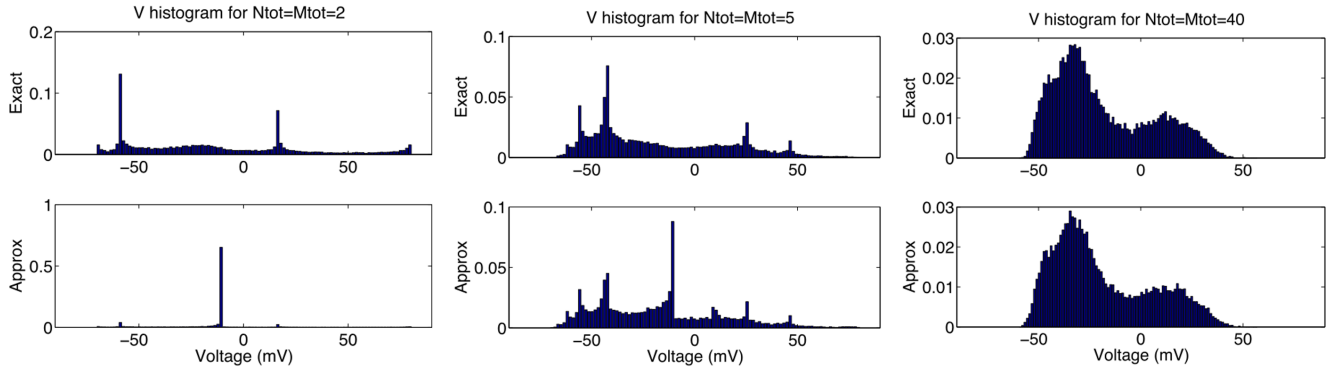


**Figure 3.** In the figure, three panels are shown, each depicting histograms of the number of open channels at different voltage values for the exact RTC and approximate piecewise constant algorithms. The values of $N_{tot}$ and $M_{tot}$ are 2, 5, and 40, respectively, for each panel. The similarity between histograms increases as these numbers grow.

## 3. RSSA Implementation

We know that to speed up the computations of the simulation a working strategy would be decreasing the number of time the propensity is directly computed. One way to achieve this is by applying RSSA, an exact stochastic method that exploits propensity upper/lower bounds together with an acceptance-rejection based sampling technique **[11]**. The reasoning behind this algorithm is that the propensity bounds will usually remain unchanged throughout many steps of the simulation; this gives the simulation the possibility to avoid recomputing the propensities. The fluctuation interval is calculated by choosing a $\delta$ fluctuation rate and applying it to the current population $X_i(t)$ as follows: $[\underline{X_i}, \overline{X_i}] = [(1-\delta_i)X_i(t), (1+\delta_i)X_i(t)]$. Three random numbers $r_1$, $r_2$ and $r_3$ are generated for each reaction selection iteration; once the propensity bounds are computed and the reaction to be fired is selected (using $r_1$) we have to check whether this event is real (to move forward in time applying the reaction) or not (to move only in time). In order to do this we take a scaled $r_2$ and check where it is located with respect to the bounds: if it is less than the lower bound, we have a fast acceptance event where the reaction is fired without recomputing

propensities; if it is between the lower bound and the real propensity, we have a slow acceptance step, where we still accept the firing of the reaction but only after recomputing its propensity; finally, if it is greater than the actual propensity we have a rejection step where we move only in time. To simulate moving in time, while keeping into account the number of rejections prior to the firing of a real event, $r_3$ is employed. Finally, the firing time is computed and the state of the system is updated. If the new state does not belong to the previously calculated $[\underline{X_i}, \overline{X_i}]$ interval, we recompute the bounds.

We adapted the Morris-Lecar system and the previously described ion channel dynamics to the RSSA formalism. Beside the inputs already defined for the previous algorithm, the ones needed to run the simulation are: the initial conditions (the number of ion channels *Mtot* and *Ntot* and the voltage *initialV*), the state change matrix $v$ for all reactions, the fluctuation rate $\delta$ and the vector $c$ containing the reaction rates. This reasoning is displayed in the following MATLAB **[12]** code:

```
initialState = [Mtot, ceil(Ntot/2), 0, ceil(Ntot/2)]; % [closed Ca,closed K,open Ca,open K]
% all Ca channels (40) and half K channels (20) start closed
initialV = -50; % initial voltage
vMinus = [1, 0, 0, 0;  % Ca opening (1 closed Ca -> 1 open Ca)
          0, 0, 1, 0;  % Ca closing (1 open Ca -> 1 closed Ca)
          0, 1, 0, 0;  % K opening (1 closed K -> 1 open K)
          0, 0, 0, 1]; % K closing (1 open K -> 1 closed K)
vPlus =  [0, 0, 1, 0;
          1, 0, 0, 0;
          0, 0, 0, 1;
          0, 1, 0, 0];
v = vPlus - vMinus;    % state change matrix
c = [alpha_m(initialV), beta_m(initialV),  % Calcium reaction rates
     alpha_n(initialV), beta_n(initialV)]; % Potassium reaction rates
delta = 0.1; % fluctuation rate
```

As previously mentioned, ion channel dynamics are modeled discretely while the voltage continuously evolves between state transitions. In order to simulate this, the value of the voltage is numerically updated by solving its differential equation (the one of Equation **5**) between the current simulation time and the next one resulting from firing a reaction. Reaction rates ($c$) are updated accordingly to the newly computed voltage (*currentV*).

```
[~, uOut] = ode23(@dudtfunc, [currentTime, currentTime+tau], % time interval
                  [currentV, currentState(3), currentState(4), Mtot, Ntot]);
                 %[voltage, open Ca, open K, total Ca, total K]
currentV = uOut(end, 1); % new voltage
c = [alpha_m(currentV), beta_m(currentV), alpha_n(currentV), beta_n(currentV)];
    % rates updated with the new voltage
currentTime = currentTime + tau;         % time update
currentState = currentState + v(mu,:);   % state update
```

In Figure **4**, we reproduced the plots of Figure **1**, by applying the RSSA approach. In the left panel, the dynamics of the Calcium and Potassium channels together with the Voltage are displayed, while on the right the V,M,N phase space is shown.

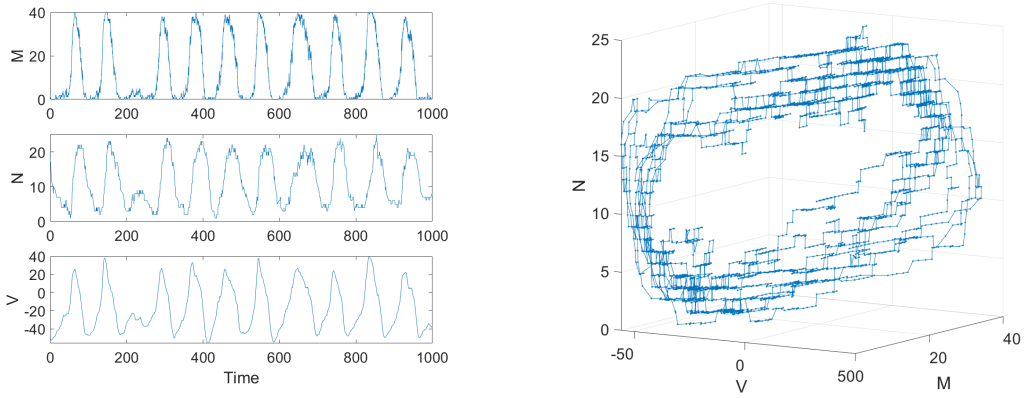


**Figure 4.** The figure shows the trajectory generated by the RSSA algorithm by setting $M_{tot} = N_{tot} = 40$ channels, an applied current $I_{app} = 100nA$ and voltage $V = -50mV$. On the left we see the number of open calcium channels ($M$), the number of open potassium channels ($N$) and the voltage ($V$), as a function of time. In the right panel, the trajectory is plotted in the (V,M,N) phase space.

By comparing the two results, we observe similar periodic behaviors especially resulting from a stable limit cycle visible in the 3D plot. Moreover, the difference in the trajectories is caused by the stochasticity of the simulations, but we can see that the trend is the same.
To furtherly assess this, we similarly created histograms depicting on the Y axis the proportion of open ion channels against the respective voltage value on the X axis (Figure 5). Specifically, we do 100 runs with $t_{Max} = 2000$, $\delta = 0.1$, $M_{tot} = N_{tot} = 2, 5, 40$.

```
voltage = Dynamics(:, 5);                          % voltage values
open_channels = Dynamics(:, 3) + Dynamics(:, 4);   % number of open channels
[bin_counts, ~, bin_indices] = histcounts(voltage, bin_edges);  % bin voltage values
for i = 1:num_bins
    bin_totals_RSSA(i) = bin_totals_RSSA(i) + sum(open_channels(bin_indices == i));
    % counts open channels for each bin
end
total_open_channels_RSSA = total_open_channels_RSSA + sum(open_channels);
% total number of open channels
bin_proportions_RSSA = bin_totals_RSSA / total_open_channels_RSSA;
% proportions of open channels for each bin
```

First, voltage and open channels data are extracted; then, the `histcounts()` function is used to split the voltage values into predefined bins, in our case 100 of equal length. `bin_counts` is a vector that contains the count of voltage elements that fall into each of the bins, `bin_edges` contains the edges of each bin and `bin_indices` indicates which bin each voltage value belongs to. Subsequently, a loop is employed to count the number of open channels for each bin and the total number of open channels is accumulated in the variable `total_open_channels_RSSA`. This last variable is used to calculate the proportion of open channels for each bin (`bin_proportions_RSSA`). The same procedure is applied to the RTC simulation.
In Figure 5, we can observe a high similarity between RSSA (on the top) and RTC (below). Moreover, the similarity increases as the total number of ion channels does too. Still, even with the very high stochasticity given by the low number of channels (two), the histograms are alike, showing comparable peaks and proving the consistency of RSSA as an exact stochastic method.

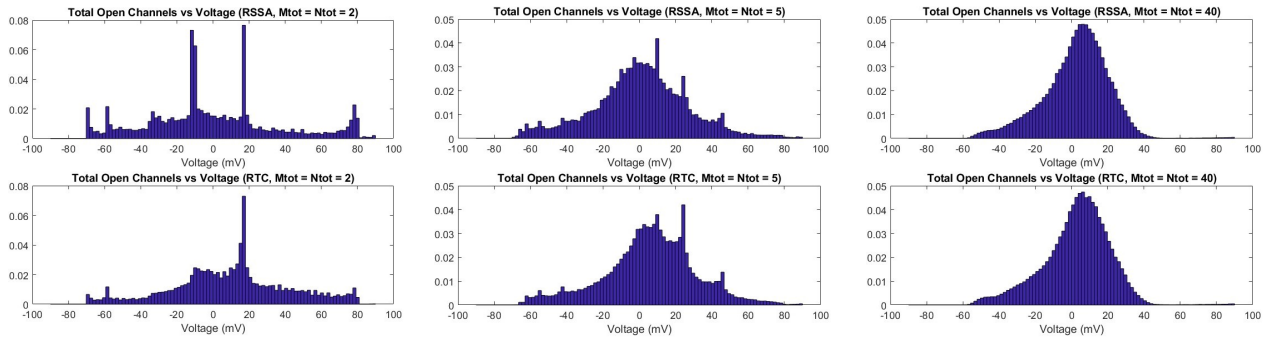


**Figure 5.** Three panels are shown, each depicting histograms of the number of open channels at different voltage values for the RSSA and RTC algorithms. The values of $N_{tot}$ and $M_{tot}$ are 2, 5, and 40, respectively, for each panel. The number of iterations was set to 100 and $T_{Max}$ to 2000. The similarity between histograms is noticeable in all the cases.

## 4. Discussion

The authors claim that generally, in neuroscience, approximated algorithms are widely used. This paper gives a different perspective, as it shows how such algorithms may have pitfalls with low numbers of modeled ion channels. Indeed, these algorithms base themselves on assumptions that may not hold; for example, the chemical Langevin approach assumes that the ion channel state increments are normally distributed in a specified time interval. This is not the case when considering low amounts of total ion channels. Similarly, the approximate piecewise constant method previously explored, does not account for the variations in membrane potential between state changes, leading to divergence when compared against an exact stochastic method (Figure 2). Approximate algorithms would be more effective with either smaller state change time intervals or bigger number of modeled ion channels.
On the other hand, one of the main drawbacks of the RTC algorithms is the computational time it requires. This would be a problem if the system complexity was increased by, for example, adding more states and transitions [13]. A way to speed up the running time of the simulation is by switching to an RSSA algorithm. We proved the consistency between the two exact stochastic approaches by both analyzing the behavior of the states across time and by comparing the distribution of the number of open ion channels against different voltages (Figure 5). The author's histograms of Figure 3 were obtained by running a

singular simulation with $t_{Max}$ = 200.000. Figure **7 in Supplementary Material** displays the results of this simulation setting applied to the comparison between RSSA and RTC algorithms. Differently from the comparison between RTC and approximate piecewise constant algorithms in Figure **3**, our histograms show a high similarity at all $M_{tot}$ and $N_{tot}$ values. While this method generates many data points useful to plot in a density based histogram, it might happen that the system enters in an "abnormal" state, especially influenced by high stochasticity (given by a low number of ion channels). To try to avoid the predominance of this phenomena, we decided to run multiple simulations (100) with a lower $T_{Max}$ of 2.000 (**5**) that generate an almost equal amount of sample points, but they tackle different behaviors that are all included in a single histogram.
Additionally, since the bottleneck of exact stochastic algorithms tends to be the high number of propensity calculations, RSSA manages to improve the computational efficiency by updating them only when necessary. This leads to an improvement of an order of magnitude in running time (for $t_{Max} = 4000$, $\delta = 0.1 - 0.2$, $dT = 1e - 5$). Still, a different bottleneck is present: the usage of the `ode23()` function to update continuously the voltage of the system. This problem could be solved by machine learning approaches: for example, it is possible to train a neural network for the prediction of the voltage based on the channel states and initial conditions, avoiding the tedious integral computation.
From a biological perspective, a further improvement of this model would be adding layers of complexity in terms of possible states the system can be in and more transition types between them (like the Hodgkin-Huxley model **[14]** does). While it would better describe the biological phenomena, it might be necessary to combine it with complexity reduction methods that would ignore, with negligible loss of accuracy, the Poisson processes not directly affecting the conductance of the channels. Note that this approach, called stochastic shielding **[15]**, is proposed by the authors in combination with the RTC approach.
From a theoretical standpoint, it would be possible to model voltage as a stochastic electron flux rather than a continuous deterministic one. This would be achieved by considering either individual ions crossing the membrane or electrons being transferred across it. While it would be accurate from a physical and quantum modeling point of view, the huge amount of molecules that would be handled would make this higher complexity not worth, as the effect would be very negligible.
Finally, it would be possible to transfer these considerations and models, made for oscillations observed in barnacle muscle fibers, to simulate human neuronal dynamics. This would require a re-parametrization procedure specific for the neuron type considered, a new model specification (such as the previously mentioned Hodgkin-Huxley model) and appropriately calibrated transition rate values. An effective strategy would be collecting experimental data points for a specific human neuron, and fitting a model to it. For example, it is possible to tune the parameters by optimizing a fitness function with a genetic algorithm. This heuristic approach is suitable because it allows to calibrate both continuous and discrete parameters.

## 5. Conclusion

By proposing an alternative to approximate approaches, the authors show that is it possible to efficiently and accurately simulate ion channel dynamics. They apply exact stochastic simulations (specifically RTC and Gillespie), bringing well known methods to the field of computational neuroscience. These models can be considered "hybrid" in the sense that they are capable of representing both continuous voltage changes and discrete ion channel dynamics. Taking advantage of newly developed algorithms, we suggest an RSSA approach that further improves the computational efficiency while maintaining the exactness of the simulations. The time gained makes other improvements possible such as increasing the complexity of the modeled system. The remaining bottleneck, caused by a continuous voltage calculation, could be tackled by employing machine learning techniques. Lastly, this knowledge could be transposed to human neuronal dynamics by appropriately calibrating new parameters to the human setting.

## Acknowledgments

We express our appreciation to Professor Luca Marchetti for his support during the teaching of the "Network Modeling and Simulation" course.

## Abbreviations

**PDMP:** piecewise-deterministic Markov processes; **ODE:** Ordinary Differential Equation; **RTC:** Random Time Change; **RSSA:** Rejection-based Stochastic Simulation Algorithm.

## References


[1] Carlo Laing and Gabriel J Lord. *Stochastic methods in neuroscience*. OUP Oxford, 2009.

[2] John A White, Jay T Rubinstein, and Alan R Kay. Channel noise in neurons. *Trends in neurosciences*, 23(3):131–137, 2000.

[3] David F Anderson, Bard Ermentrout, and Peter J Thomas. Stochastic representations of ion channel kinetics and exact stochastic simulation of neuronal dynamics. *Journal of computational neuroscience*, 38:67–82, 2015.

[4] Thanh Vo Hong, Roberto Zunino, et al. Rssa: a rejection-based stochastic simulation algorithm. 2013.

[5] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[6] Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58(1):35–55, 2007.

[7] David F Anderson and Thomas G Kurtz. Continuous time markov chain models for chemical reaction networks. In *Design and analysis of biomolecular circuits: engineering approaches to systems and synthetic biology*, pages 3–42. Springer, 2011.

[8] Harold Lecar. Morris-lecar model. *Scholarpedia*, 2(10):1333, 2007.

[9] Bard Ermentrout and David Hillel Terman. *Foundations of mathematical neuroscience*. Citeseer, 2010.

[10] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 07 2001.

[11] Luca Marchetti, Corrado Priami, Vo Hong Thanh, et al. *Simulation algorithms for computational systems biology*, volume 1. Springer, 2017.

[12] The MathWorks Inc. Matlab version: 9.13.0 (r2022b), 2022.

[13] Lorin S Milescu, Tadashi Yamanishi, Krzysztof Ptak, and Jeffrey C Smith. Kinetic properties and functional dynamics of sodium channels during repetitive spiking in a slow pacemaker neuron. *Journal of Neuroscience*, 30(36):12113–12127, 2010.

[14] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[15] Nicolaus T Schmandt and Roberto F Galán. Stochastic-shielding approximation of markov chains and its application to efficiently simulate random ion-channel gating. *Physical review letters*, 109(11):118101, 2012.

## Supplementary Material

Supplementary material, together with the Matlab codes utilized in our analysis can be found in the GitHub repository at the following link: `https://github.com/Sara-Baldinelli/NM-project`

$$\alpha_m = \phi_m \cosh\left(0.5\frac{v-v_a}{v_b}\right)\left(0.5(1+tanh(\frac{v-v_a}{v_b}))\right)$$

$$\beta_m = \phi_m \cosh\left(0.5\frac{v-v_a}{v_b}\right)\left(1-0.5(1+tanh(\frac{v-v_a}{v_b}))\right)$$

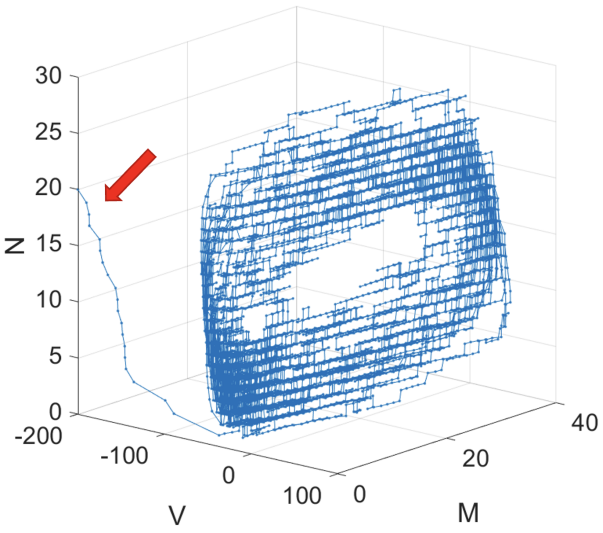


**Figure 6.** 3D phase space (Voltage V, Calcium channels M, Potassium channels N) showing how, starting from different initial conditions ($V = -200mV$, red arrow), the trajectory ends up in the stable limit cycle.

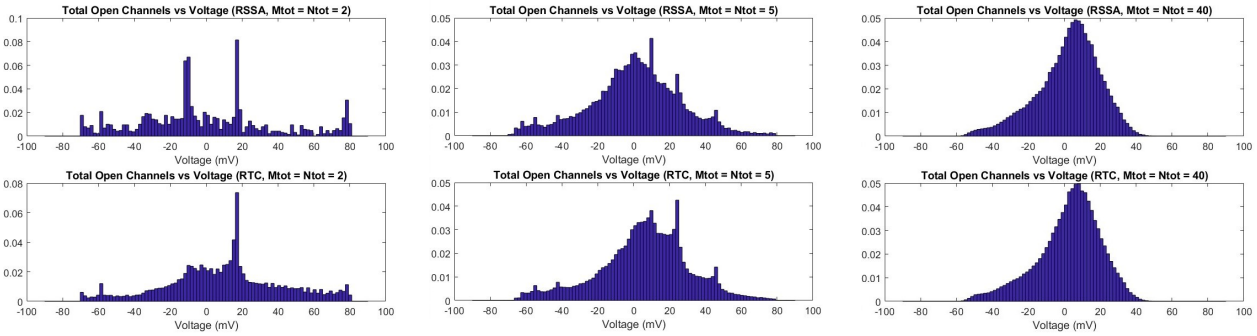


**Figure 7.** Three panels are shown, each depicting histograms of the number of open channels at different voltage values for the RSSA and RTC algorithms. The values of $N_{tot}$ and $M_{tot}$ are 2, 5, and 40, respectively, for each panel. The number of iterations was set to 1 and $T_{Max}$ to 200000. The similarity between histograms is noticeable in all the cases.