Università degli Studi di Padova

Dipartimento di Fisica e Astronomia Galileo Galilei

Matlab Simulation of the Two Open State Model

Francesca Dodici

Abstract

Ion channels are macromolecular pores in cell membranes. They represent the most fundamental excitable elements in the cell membrane as through their opening and closing, they allow a regulated flux of ions, shaping the signals and responses of the nervous system. The opening and closing of a single channel is a stochastic event and as such it should be studied with the tools of probability theory. Single channels can also present more than one open or closed states, with different transition probabilities between each other. Moreover, there are channels showing the presence of substates of reduced conductance with respect to the fully open state. In this article, the simulation of the current signal from a single channel with two open states will be carried out. The same simulation will be repeated considering different conductances for the two open states. Finally, the combined signal of up to 1000 channels will be simulated, in order to study the relationship between current noise and number of channels.

Dipartimento di Fisica e Astronomia "Galileo Galilei", Università degli Studi di Padova, Padua, Italy

e-mail: francesca.dodici@studenti.unipd.it

Contents

1 Introduction 1

2 Methods 2

2.1 Dual state model 2

Theoretical model • Simulation

2.2 Two-open state model 3

Theoretical model • Simulation

2.3 Three-state model 4

Theoretical model • Simulation

2.4 Global signal 5

Simulation

3 Results and Discussion 5

3.1 Dual state model 5

3.2 Two-open state model 5

3.3 Three-state model 6

3.4 Global signal 10

4 Conclusion 10

References 10

Supplementary Material 11

1. Introduction

Physiologists have long known the central role played by ions in the excitability of nerve and muscle. More specifically, excitation and electrical signaling in the nervous system involve the movement of ions through ion channels. These channels are found in the membranes of all cells, both prokaryotic and eukaryotic. Their known functions include establishing a resting membrane potential, shaping electrical signals, gating the flow of messenger Ca^{2+} ions, controlling cell volume, and regulating the net flow of ions and fluids. An ion channel may be considered as an excitable molecule, as it is specifically responsive to certain stimuli. These can be a membrane potential change, a neurotransmitter or other chemical stimulus, a mechanical deformation, and so on. Independently on the underlying mechanism, the channel's response, called gating, can be schematized as a simple opening or closing of the pore. The open pore has the important property of selective permeability, allowing some restricted class of small ions to flow passively down their electrochemical activity gradients at a rate that is very high when considered from a molecular viewpoint. This high throughput rate is a diagnostic feature that allows to distinguish ion channel mechanisms from those of other ion transport devices such as the Na^+ - K^+ pump [1].

Single-channel kinetics has proven a powerful tool to reveal information about the gating mechanisms that control the opening and closing of ion channels. Because of the intrinsic randomness of molecular motion, the transitions between the different states are random events as well and thus, to describe the dynamic of the system, one must work in terms of probabilities. Discrete state Markov (DSM) models, named after Russian mathematician Andrey Markov, have proven highly useful to describe single-channel gating. These models assume that channels gate by moving among different conformational and/or agonist bound states of the channel protein. Open and closed states of the channel represent two different conformational states [2]. In a DSM model, the following two assumptions are employed:

1. The transitions between states are Markov Processes and as such one can find a time interval dt small enough that the probability of having more than one transition in it is negligible.

2. The probability that a transition will occur is independent on how long the channel has been open and equal simply to the product of the transition rate with the interval dt.

For these reasons, Markov models are often referred to as memoryless processes.

2. Methods

2.1 Dual state model

The simplest case that can be taken into account is the one where the channel considered can only switch between open (O) and closed (C):

$$C \underset{\beta}{\overset{\alpha}{\rightleftarrows}} O$$

where α and β are the forward and backward transition rates.

This simple dual state model was initially implemented in early analyses of the channel properties at the neuromuscolar junction to describe the time course of a postsynaptic current where N channels, each one carrying a unitary current I_o, open at once due to neurotrasmitter released into the synaptic cleft. Since unbound neurotransmitters are removed very rapidly there is no chance of rebinding a receptor and making the channel open a second time [1].

2.1.1 Theoretical model

To model this system, let's start by defining $p_o(t)$ as the probability density function of the event "open channel", such that $p_o(t) \cdot dt$ is the probability that the channel will be open in the time interval $[t, t+dt]$. Thus, the probability that the channel remains open for at least a time t is given by the distribution function:

$$P_o(t) = \int_t^{\infty} p_o(s) ds$$

and the probability to remain open for at least a time $t+dt$ can be computed as the product of $P_o(t)$ and the probability that the channel does not close in the time interval dt.

On the basis of the assumptions considered in a DSM model, the probability that a closure event will not occur during dt is $1-\beta dt$ and consequently:

$$P_o(t+dt) = P_o(t) \cdot (1-\beta dt) \quad (1)$$

Rearranging eq. (1) the following differential equation for the open-time distribution function is found:

$$\frac{dP_o}{dt} = -\beta P_o \quad (2)$$

By imposing the starting condition $P_o(0) = 1$ [1] the solution can be immediately found to be:

$$P_o(t) = e^{-\beta t} \quad (3)$$

By repeating the same procedure with the closed-time distribution function one can also find:

$$P_c(t) = e^{-\alpha t} \quad (4)$$

The probability density functions for the open and closed states are then obtained as minus the time derivative of $P_o(t)$ and $P_c(t)$ respectively.

2.1.2 Simulation

To simulate the single channel current and the statistics of the time duration of the two states, the Matlab [3] code DualStateModel.m is employed[2]. The algorithm takes as input:

- the time step `dt` (in ms);
- the total simulation time `T` (also in ms);
- the probabilities of opening and closing in dt `alpha` and `beta`;
- the average value of the current `Io` when the channel is open (in pA).

A temporary variable `curr_state` is used to keep track of the channel state at each iteration: if `curr_state` $=0$ the channel is closed, if it's $=1$ it's open. The initial state is set to C. Then, at each iteration, the algorithm extracts a random variable between 0 and 1 and

- if `curr_state` $=0$ (C) it checks whether the random number is smaller than the opening probability $po = \alpha \cdot dt$. If the condition is verified `curr_state` is set to 1 (O);
- if `curr_state` $=1$ (O) it checks whether the random number is smaller than the closing probability $pc = \beta \cdot dt$. If the condition is verified `curr_state` is set to 0 (C).

A vector `state` is used to store the full list of zeroes and ones corresponding to the sequence of states at each iteration. This vector is then used to generate a graph displaying the temporal evolution of the channel current, adding some random noise to reproduce the real behaviour of a patch clamp recording.

By setting a longer simulation time (at least 100 000 ms) it is possible to study the statistics of the open and closed states

[1] Which is trivially verified since the channel will surely be open for at least 0 s

[2] See **Supplementary material** for the code

(i.e. the histogram of the time duration). To do this, an algorithm is implemented to generate two arrays `open_array` and `closed_array` containing the duration of every opening and closing event simulated. This is done starting from `state` and calculating the length of each string of consecutive ones (O) or zeroes (C) in it. The histograms of these time duration simulate the probability density functions $p_c(t)$ and $p_o(t)$ which, recalling the definition of $P_c(t)$ ans $P_o(t)$ in *Paragraph 2.1.1* are:

$$p_c(t) = -\frac{dP_c}{dt} = \alpha e^{-\alpha t} \tag{5}$$

$$p_o(t) = -\frac{dP_o}{dt} = \beta e^{-\beta t} \tag{6}$$

2.2 Two-open state model

Sometimes in single-channel experiments, the succession of O and C states is not as evenly distributed as in the previous model, but there might be clusters of closing events followed by a long time in the open state. To model this behaviour one can assume the existence of two open states:

$$C \underset{\beta}{\overset{\alpha}{\rightleftarrows}} O_1 \underset{\delta}{\overset{\gamma}{\rightleftarrows}} O_2$$

where the transition rates α and β between closed and opened are fast while γ and δ between the two open states are slow. Similarly some channels might follow a two-closed state model, with fast opening events followed by long periods in the closed state. A type of channel that can be described by this model is a ligand-gated channel. Since the behaviour of the two systems is exactly symmetrical, this paper will only focus on the two-open state version.

2.2.1 Theoretical model

Following the same framework used in *Paragraph 2.1.1* to find eq.(2) one can compute an analogous set of differential equations for the two open time distribution functions $P_{o1}(t)$ and $P_{o2}(t)$. These two represent the probabilities that the channel will remain open (either in O1 or O2) for at least a time t once found in the state O1 or O2 respectively. The closed-time distribution $P_c(t)$ can also be found, with results similar to those of the previous chapter.

$$\frac{dP_c}{dt} = -\alpha P_c \tag{7}$$

$$\frac{dP_{o1}}{dt} = \delta P_{o2} - (\beta+\gamma) P_{o1} \tag{8}$$

$$\frac{dP_{o2}}{dt} = -\delta P_{o2} + \gamma P_{o1} \tag{9}$$

The solution to eq.(7) is the same as in the two state model, namely:

$$P_c(t) = e^{-\alpha t} \tag{10}$$

whereas for eqs.(8) and (9) one can perform the first derivative of eq. (8):

$$\frac{d^2 P_{o1}}{dt^2} = \delta \frac{dP_{o2}}{dt} - (\beta+\gamma)\frac{dP_{o1}}{dt} \tag{11}$$

and then substituting $\frac{dP_{o2}}{dt}$ derived from eq.(9) and P_{o2} from eq.(8) results in the following second-order differential equation for P_{o1}:

$$\frac{d^2 P_{o1}}{dt^2} + (\beta+\delta+\gamma)\frac{dP_{o1}}{dt} + \beta\delta P_{o1} = 0 \tag{12}$$

which is known to have solution:

$$P_{o1} = a_1 e^{\lambda_1 t} + a_2 e^{\lambda_2 t} \tag{13}$$

where a_1 and a_2 are constants depending on the initial conditions and λ_1 and λ_2 are the two solutions of the characteristic polynomial associated to the differential equation (12):

$$\lambda^2 + (\beta+\delta+\gamma)\lambda + \beta\delta = 0$$
$$\Longrightarrow \lambda_{1,2} = \frac{-(\beta+\delta+\gamma) \pm \sqrt{(\beta+\delta+\gamma)^2 - 4\beta\delta}}{2}$$

Repeating the same steps for P_{o2} results in a differential equation identical to eq.(12) and thus the solution for the second open state will also be:

$$P_{o2} = b_1 e^{\lambda_1 t} + b_2 e^{\lambda_2 t} \tag{14}$$

with same $\lambda_{1,2}$ as in eq.(13) but different coefficients b_1 and b_2. The starting conditions result from the following considerations:

1. Both the open and the closed state have a 100% probability to last at least 0 s.
2. The second open state O2 can only be accessed from the first one O1, but not directly from C.
3. Since this is a Markov process, only one transition can occur in the time step dt

$$\Longrightarrow P_c(0) = 1,\ \ P_{o1}(0) = 1,\ \ P_{o2}(0) = 0 \tag{15}$$

Imposing these starting conditions, the coefficients in eqs.(13) and (14) turn out to be:

$$a_1 = -\frac{\lambda_2+\beta+\gamma}{\lambda_1-\lambda_2} \qquad b_1 = \frac{\gamma}{\lambda_1-\lambda_2}$$
$$a_2 = \frac{\lambda_1+\beta+\gamma}{\lambda_1-\lambda_2} \qquad b_2 = -\frac{\gamma}{\lambda_1-\lambda_2}$$

The probability $P_o(t)$ of having the channel still open after a time t is given by the sum of the probabilities of being in either O1 or O2:

$$P_o = P_{o1} + P_{o2} \tag{16}$$

2.2.2 Simulation

Another Matlab code TwoOpenStateModel.m [3] is utilised to simulate the single channel current and the statistics of the time duration of the three states. The algorithm takes as input:

[3] See **Supplementary material** for the code

- the time step `dt` (in ms);
- the total simulation time `T` (also in ms);
- `alpha` probability of having a $C \to O1$ transition in dt;
- `beta` probability of having a $O1 \to C$ transition in dt;
- `gamma` probability of having a $O1 \to O2$ transition in dt;
- `delta` probability of having a $O2 \to O1$ transition in dt;
- `Io` average value of the current when the channel is open (in pA).

Similarly to what is done in DualStateModel.m, a temporary variable `curr_state` is used to keep track of the channel state at each iteration. In this case however, since there is an additional open state, the two are differentiated by setting `curr_state` to 1 for O1 and to 2 for O2. The initial state is set to C. The same algorithm described in *Paragraph* 2.1.2, properly modified to take into consideration the additional open state, is used also in this routine. Two vectors `stateO1` and `stateO2` are used to store a list of zeroes and ones corresponding to the sequence of closed and open states (O1 for `stateO1` and O2 for `stateO2`) at each iteration.

Since the second open state is considered only to take into consideration the presence of different transition rates but the current is the same as in the first one, the two vectors `stateO1` and `stateO2` are summed into `stateO` to generate a graph displaying the temporal evolution of the channel current. Some random noise is added to reproduce the real behaviour of a patch clamp recording.

Again, setting a longer simulation time (at least 100 000 ms) it is possible to study the statistics of the two open and closed state (i.e. the histogram of the time duration). For the closed state, the same procedure as in *Paragraph* 2.1.2 is used. On the other hand, for the open states the following routine is implemented:

1. A 1D array `state_durat` of length $N = T/dt$ is created. This array contains 0 at the timesteps where the state is C, 1 when it's O1 and 3 when it's O2.
2. Using the Matlab function `diff()` on `state_durat` converts it to an array with information on the changes of state between consecutive timesteps. When the state remains the same there is a 0, a $+1$ corresponds to a $C \to O1$ transition, a -1 to a $O1 \to C$ transition, a $+2$ to $O1 \to O2$ and finally a -2 to $O2 \to O1$.
3. Remembering the definition of P_{o1} in *Paragraph* 2.2.1, to generate the histogram corresponding to this pdf one has to compute the duration of each open state starting in O1. This is equal to the number of steps between a $+1$ and the first consecutive -1 in `state_durat`. These duration are stored in `O1_durat`.
4. Recalling also the definition of P_{o2} in *Paragraph* 2.2.1, to generate the histogram of this pdf one has to compute the duration of each open state starting in O2. This is equal to the number of steps between a $+2$ and the first consecutive -1 in `state_durat`. These duration are stored in `O2_durat`.

Having computed these duration one can then proceed in the same fashion as in *Paragraph* 2.1.2 to generate the histograms simulating the probability distributions p_c, p_{o1}, p_{o2} and p_o.

2.3 Three-state model

Another interesting situation is that of a channel whose second open state has a smaller conductance with respect to the fully open state. This case is more similar to a three-state model in the form:

$$C \underset{\beta}{\overset{\alpha}{\rightleftarrows}} S \underset{\delta}{\overset{\gamma}{\rightleftarrows}} O$$

where C is the closed state, O the fully-open one and S the substate of smaller conductance.

Both GABA- and glycine-activated channels (GlyRs) have multiple levels of conductance. A possible cause for sublevels would be if the separate activated subunits of a channel could induce open/close-like conformational changes of the channel before all subunits are in an activated state. [1].

2.3.1 Theoretical model

This situation too can be considered a Markov process with transition probabilities independent on the duration of state prior to the transition itself, finding:

$$\begin{cases} \dfrac{dP_c}{dt} = -\alpha P_c + \beta P_s \\ \dfrac{dP_s}{dt} = \alpha P_c - (\beta+\gamma) P_s + \delta P_o \\ \dfrac{dP_o}{dt} = \gamma P_s - \delta P_o \end{cases} \tag{17}$$

This system can also be re-cast in matrix form as:

$$\begin{pmatrix} \dfrac{dP_c}{dt} \\ \dfrac{dP_s}{dt} \\ \dfrac{dP_o}{dt} \end{pmatrix} = \begin{pmatrix} -\alpha & \beta & 0 \\ \alpha & -\beta-\gamma & \delta \\ 0 & \gamma & -\delta \end{pmatrix} \begin{pmatrix} P_c \\ P_s \\ P_o \end{pmatrix}$$

$$\implies \frac{d}{dt}\mathbf{P} = \mathbf{Q} \cdot \mathbf{P} \tag{18}$$

where $\mathbf{P}$ is the vector of the distribution functions and $\mathbf{Q}$ is the 3×3 matrix whose off-diagonal elements Q_{ij} $(i \neq j)$[4]

[4] The subscript i, j correspond to the 3 states C "closed", S "open-substate" and O "fully open". To simplify the notation, in the following discussion we will take $c = 1$, $s = 2$ and $o = 3$

represent the transition rates from state i to state j. On the other hand the diagonal elements Q_{jj} are the sum of the transition rates from state j to all other states. The solution to eq.(18) is known to be:

$$P_i = \sum_{j=1}^{3} k_{ij} e^{\lambda_j t} \tag{19}$$

where the λ_j are the 3 eigenvalues of matrix $\mathbf{Q}$ and the coefficients k_{ij} depend on the starting conditions

$$\Longrightarrow P_c(0) = 1,\ \ P_s(0) = 1,\ \ P_o(0) = 1 \tag{20}$$

which are trivially verified due to the fact that all 3 states can last at least 0 s.

As the number of states considered increases it is more efficient to implement this matrix formalism to find the solutions.

2.3.2 Simulation

In this case, the Matlab code ThreeStateModel.m is utilised to simulate the single channel current and the statistics of the time duration of the three states. The simulation of the channel state works exactly as in TwoOpenStateModel.m. In this case a vector `stateO` is filled with the sequence of zeroes and ones where the latter correspond to the timesteps when the channel is fully open. `stateS` does the same but when the channel is in the substate of smaller conductance. The two vectors are then summed but, before that, `stateS` is multiplied by s, the fraction of current in the substate with respect to the one in the fully open state. In this way, the different conductance is taken into account. Random noise is added also in this simulation to reproduce the real behaviour of a patch clamp recording.

With a longer simulation time (at least 100 000 ms) it is possible to study the statistics of the three states exactly as in the previous simulations, with the difference that here S is considered as a different state with respect to O and studied independently.

Another difference is that in this code, the λ roots are found by solving the eigenvalue problem associated to matrix Q (see eq. (18)). This is done using the Matlab function `eig()`. Having found the solutions, the histograms of the three states can then be fitted with a linear combination of the three exponential factors $e^{\lambda_i t}$. To perform the fit the `lsqcurvefit()` method is used.

2.4 Global signal

As a final step one can simulate the current signal resulting when considering an increasing number of channel. Although the single-channel signal is a noisy switch-like signal, it can be seen that with an increasing number of channels this discrete switch-like behaviour is lost, becoming more similar to the signal recorded by a patch-clamp recording in the whole cell-configuration.

2.4.1 Simulation

To simulate the current output when an increasing number `C` of channels is considered, the Matlab routine ManyChannels.m is created. The algorithm repeats the simulation of the single channel behaviour used in TwoOpenStateModel.m `C` times and then sums the outputs. In this way, the temporal evolution of the total current can be derived.

3. Results and Discussion

3.1 Dual state model

The results of the DualStateModel.m simulation are displayed in *Figure 1*. For this simulation the input parameters are set to:

`dt` = 0.1 ms

`T` = 1000 ms

`alpha` = 0.02 ms^{-1}

`beta` = 0.04 ms^{-1}

`Io` = 50 pA

The temporal evolution of the current displayed in *Figure 1a* shows a random switching between the two states, with slightly longer duration of the closed state as it can be seen more clearly from the histogram in *Figure 1b*. The counts for the lower values of current corresponding to C are about twice the ones for O. This is to be expected due to the fact that the transition rate β for $O \rightarrow C$ was set to be 2α. Thus, the characteristic time of the C states $\tau_c = \alpha^{-1} = 50$ ms is double the one of the O ones $\tau_o = \beta^{-1} = 25$ ms and the channel spends on average more time in the C state.

Then the simulation time is increased to `T` = 100 000 ms to simulate the duration of the channel states. In *Figures 1d* and *1c* it can be seen that the theoretical pdfs indeed fit perfectly the simulated data.

3.2 Two-open state model

The results of the TwoOpenStateModel.m simulation are displayed in *Figure 2*. For this simulation the input parameters are set to:

`dt` = 0.1 ms

`T` = 1000 ms

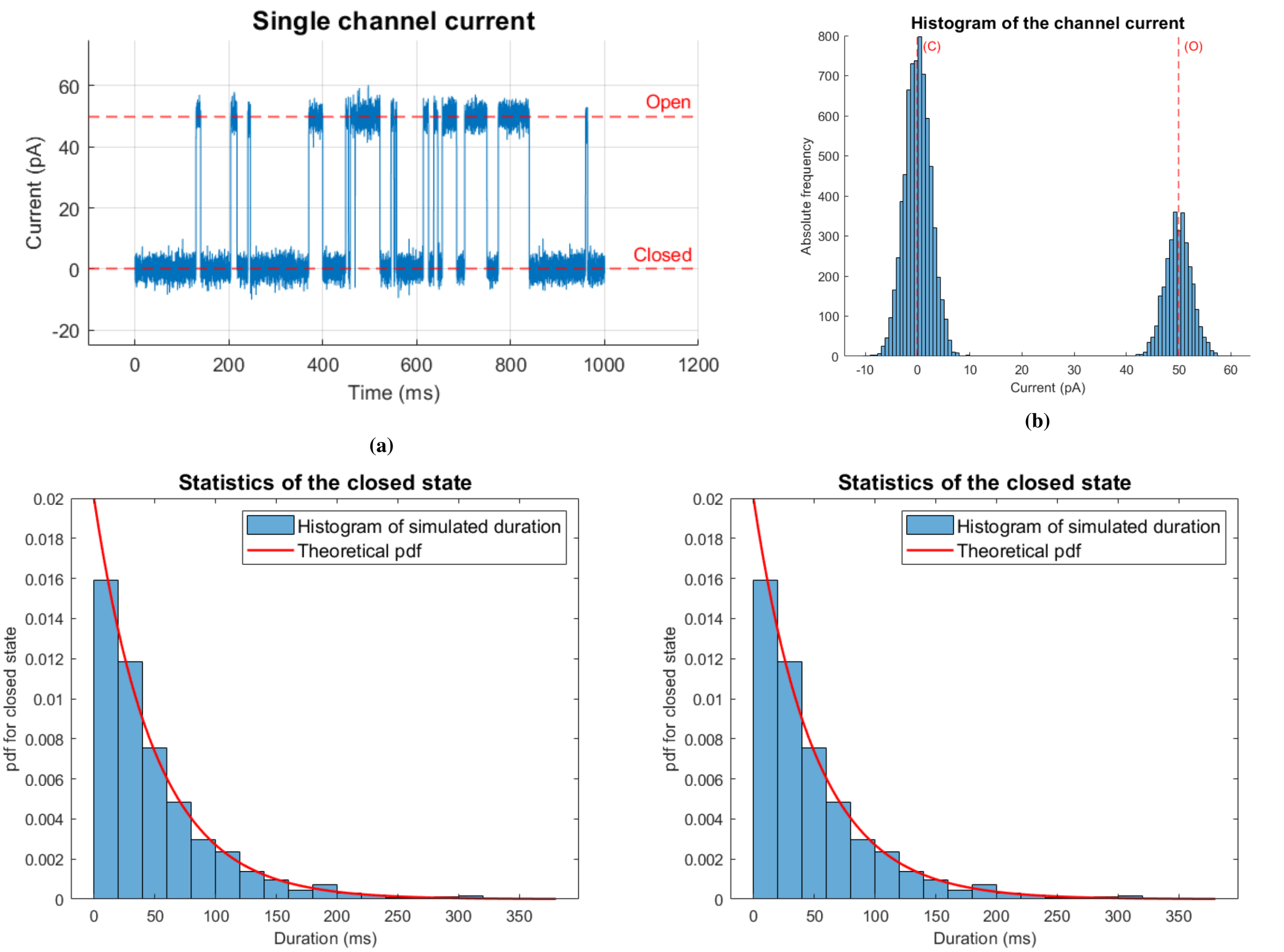
`alpha` = 0.02 ms^{-1}

`beta` = 0.02 ms^{-1}

`gamma` = 0.004 ms^{-1}

`delta` = 0.004 ms^{-1}

`Io` = 50 pA



(a) (b) (c) (d)

Figure 1. Dual state model with $\alpha = 0.02$ **and** $\beta = 0.04$. (1a) Temporal evolution of the simulated single channel current; (1b) Histogram of the channel current; (1c) Simulated duration of the closed states and corresponding probability density function p_c; (1d) Simulated duration of the open states and corresponding probability density function p_o

As required by the model the transition rates between O1 and O2 (γ and δ) were set to be about an order of magnitude smaller than the ones between O1 and C (α and β). The temporal evolution of the current displayed in *Figure 2a* shows fast switching between O and C alternated by very long time intervals spent by the channel in the open state. This is due to the presence of the second open state with slower transition rate and it also reflects in the higher counts for the open state in the histogram in *Figure 2b*.

Then the simulation time is increased to `T` = 100 000 ms to simulate the duration of the channel states. In *Figures 2c* and *2d* are shown the histogram of the simulated opening events starting with O1 and O2 respectively. The agreement with the theoretical distribution is good. In particular, the one for P_{o2} can be further improved by simulating more events. The agreement with the theory's predictions is good also for the opening and closing probability density functions, as shown in *Figures 2e* and *2f*.

3.3 Three-state model

The results of the ThreeStateModel.m simulation are displayed in *Figure 3*. For this simulation the input parameters are set to:

`dt` = 0.1 ms

`T` = 1000 ms

`alpha` = 0.02 ms^{-1}

`beta` = 0.04 ms^{-1}

`gamma` = 0.03 ms^{-1}

`delta` = 0.02 ms^{-1}

`s` = 0.5

`Io` = 50 pA

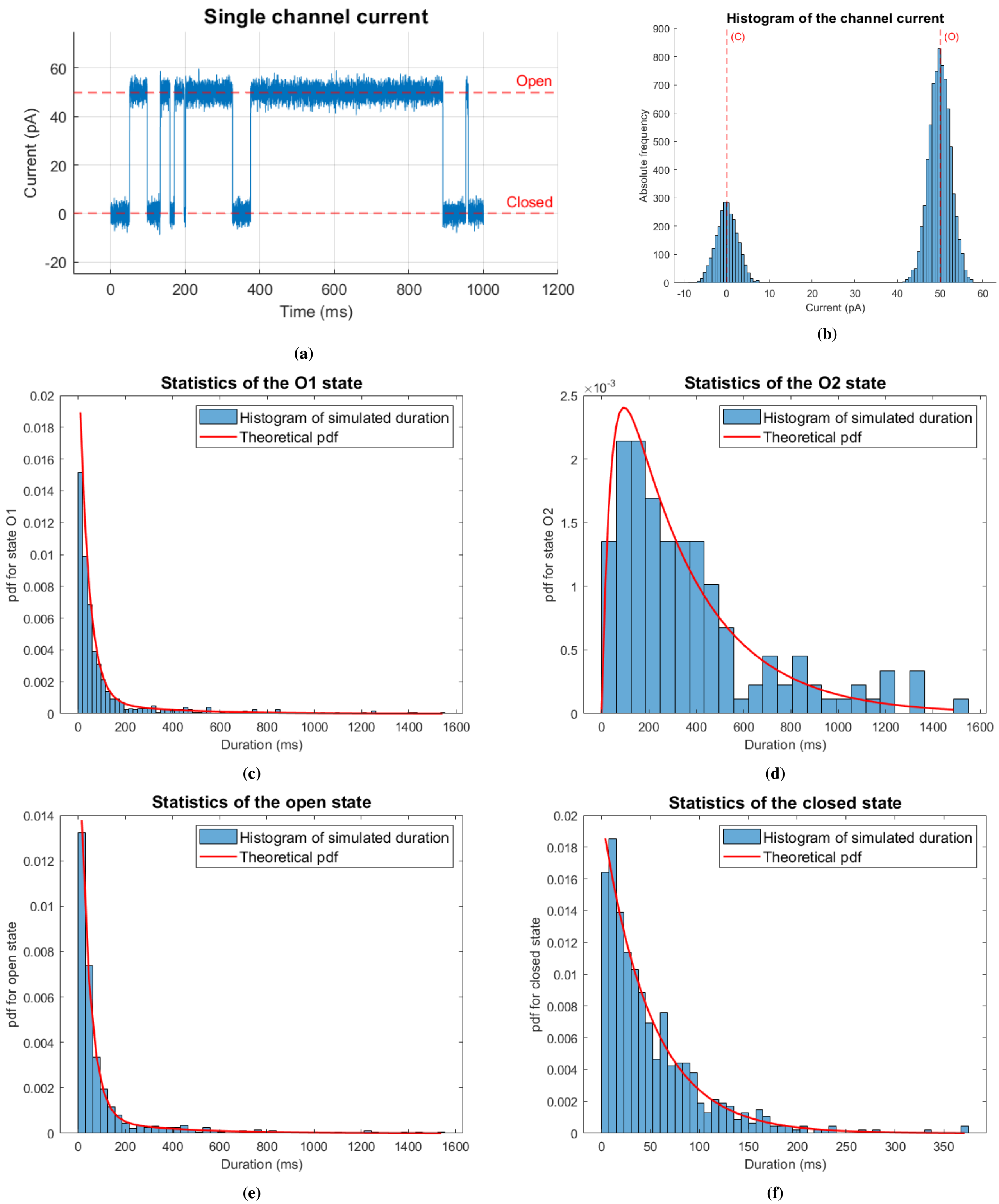


Figure 2. Two open state model with $\alpha = 0.02$ and $\beta = 0.02$, $\gamma = 0.004$ and $\delta = 0.004$. (2a) Temporal evolution of the simulated single channel current; (2b) Histogram of the channel current; (2c) Simulated duration of the open states starting in O1 and corresponding theoretical probability density function; (2d) Simulated duration of the open states starting in O2 and corresponding theoretical probability density function; (2e) Simulated duration of the open states and corresponding probability density function p_o; (2f) Simulated duration of the closed states and corresponding probability density function p_c.

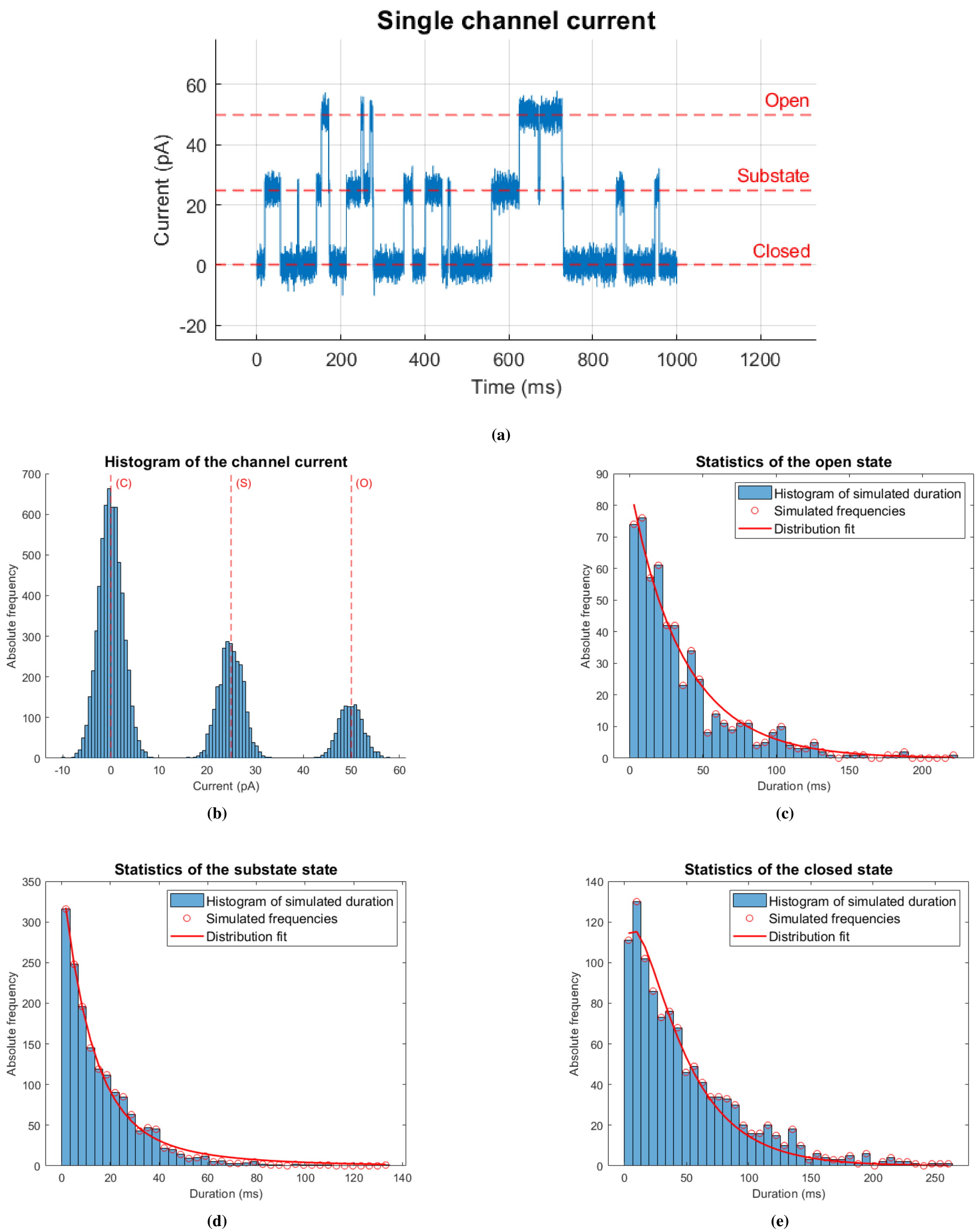


Figure 3. Three state model with $\alpha = 0.02$, $\beta = 0.04$, $\gamma = 0.03$ and $\delta = 0.02$. (3a) Temporal evolution of the simulated single channel current; (3b) Histogram of the channel current; (3c) Simulated duration of the open states and corresponding fit; (3d) Simulated duration of the substate with smaller conductance and corresponding fit; (3e) Simulated duration of the closed states and corresponding fit.

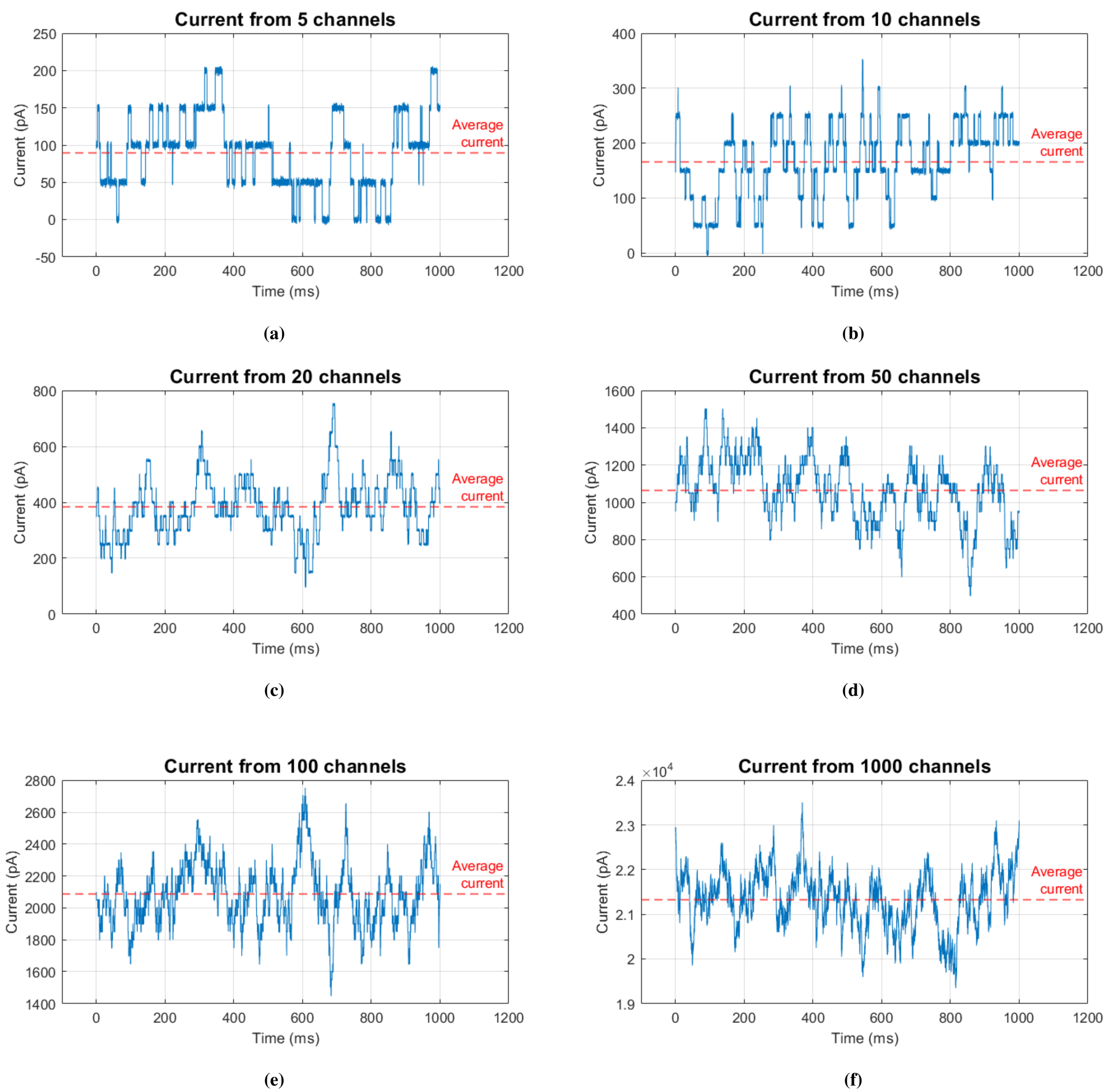


Figure 4. Total current from many channels following the two open state model with $\alpha = 0.02$, $\beta = 0.04$, $\gamma = 0.002$ and $\delta = 0.004$. (4a) Simulation of 5 channels; (4b) Simulation of 10 channels; (4c) Simulation of 20 channels; (4d) Simulation of 50 channels; (4e) Simulation of 100 channels; (4f) Simulation of 1000 channels;

Again, the temporal evolution of the current displayed in *Figure 3a* shows a random switching between the states. Since the conductance of the substate is set to be half the one of the fully open state the current in S is as well, on average, half the one in O. The number of counts displayed in the histogram in *Figure 3b* are consistent with the transition rates set. In particular, since `beta` is set to be the biggest transition rate, the channel spends, on average, more time in the closed state.

Then the simulation time is then increased to `T` = 100 000 ms to simulate the duration of the channel states. The solution of the eigenvalue problem results in:

$$\lambda_1 = -0.0837$$
$$\lambda_2 = 0$$
$$\lambda_3 = -0.0263$$

In *Figures 1d*, *3d* and *1c* the simulated duration are shown. The histograms are fitted with a linear combination of the three exponential factors:

$$y = ae^{\lambda_1 t} + be^{\lambda_2 t} + ce^{\lambda_3 t} \tag{21}$$

It can be seen that the fitting procedure leads to good results in all three cases. The fit coefficients are shown in *Table 1*.

State	a	b	c
C	−47.23	−47.23	202.9
S	150.5	150.5	58.72
O	2.039	2.039	82.97

Table 1. Fit parameters for the ThreeStateModel duration as linear combination of exponentials.

3.4 Global signal

The results of the ManyChannels.m simulation are displayed in *Figure 4*. For this simulation the input parameters are set to:

`dt` = 0.1 ms

`T` = 1000 ms

`alpha` = 0.02 ms^{-1}

`beta` = 0.04 ms^{-1}

`gamma` = 0.002 ms^{-1}

`delta` = 0.004 ms^{-1}

`Io` = 50 pA

The number of channels simulated `C` is set to different values. The temporal evolution of the total current displayed in *Figure 4a* corresponds to the simulation of 5 channels. It can be seen that the discrete nature of the current due to the switching between states with a constant average value of the current is still clearly visible. This digital-like appearence becomes less and less present as the number of channel increases to 10 (*Figure 4b*), then 20 (*Figure 4c*), 50 (*Figure 4d*), 100 (*Figure 4e*) and finally 1000 (*Figure 4f*). The discrete nature of the noise due to the switching of the single channels is progressively reduced until it basically becomes a random noise around an average value of current.

4. Conclusion

In the light of the results obtained, it can be concluded that by setting the proper values of the transition rates, the Matlab routines presented in this article can be used to simulate the behaviour of a real single channel of a cell. This, only provided that the original working hypothesis of the state changes being Markov Process and the transition probabilities being independent on how long the channel has been opened are verified.

References

[1] Bertil Hille et al. *Ion channels of excitable membranes*. Sinauer, Sunderland, MA, 2001.

[2] Yanyan Geng and Karl L. Magleby. Single-channel kinetics of bk (slo1) channels. *Frontiers in Physiology*, 5, 2015.

[3] Timothy A Davis. *MATLAB primer*. CRC Press, 2011.

Supplementary Material

The Matlab codes utilized for the simulations are here reported.

DualStateModel.m

```
function DualStateModel(dt,T,alpha,beta,Io)

%Syntax: DualStateModel(dt,T,alpha,beta,Io)
%
%dt    is the simulation time step (in ms) supposing that no more than a
%      single closure or opening event can occur in this time period(Markov event);
%T     is the total simulation time (in ms);
%alpha is the opening probability in dt (i.e. Po = alpha*dt);
%beta  is the closing probability in dt (i.e. Pc = beta*dt);
%Io    is the current of the open state (in pA).
%
%Example:   DualStateModel(0.1,1000,0.02,0.04,50); to look at a single event
%           DualStateModel(0.1,100000,0.02,0.04,50); to look at the statistics

N = ceil(T/dt);      %Number of steps
state = zeros(1,N); %Array of channel state at all time-steps
curr_state = 0;      %Initial state is set as closed
t = dt;              %State time
T_array = dt*(1:N); %Array of absolute time
Po = alpha*dt;       %Opening probability
Pc = beta*dt;        %Closure probability

%Compute channel state at all time steps
for i =1:N-1
    if curr_state == 0
        if rand<=Po
            curr_state = 1;
            state(i+1) = 1;
            t = dt;
        else
            curr_state = 0;
            t = t+dt;
        end
    else
        if rand <= Pc
            curr_state = 0;
            t = t+dt;
        else
            curr_state = 1;
            state(i+1) = 1;
            t = t+dt;
        end
    end
end

%Plot the temporal evolution of single channel current
figure;
real_current = Io*(state + (randn(1,N))*0.05); %add normally-distributed random noise
horO = yline(Io,'--r','Open');
horO.LineWidth = 1;
hold on;
horC = yline(0,'--r','Closed');
horC.LineWidth = 1;
hold on;
plot(T_array,real_current);
xlabel('Time (ms)')
ylabel ('Current (pA)')
title ('Single channel current', 'FontSize',14)
axis([-T/10 T+T/5 -Io/2 Io+Io/2]);
pbaspect([2 1 1])
grid on

%Plot statistics
figure;
```

```
vertO = xline(Io,'--r','(O)');
vertO.LineWidth = 1;
vertO.LabelOrientation = 'horizontal';
hold on;
vertC = xline(0,'--r','(C)');
vertC.LineWidth = 1;
vertC.LabelOrientation = 'horizontal';
hold on;
histogram(real_current, 100);
xlabel('Current (pA)')
ylabel ('Absolute frequency')
title ('Histogram of the channel current', 'FontSize',14)

%Compute open and close state duration
logi = logical(state);

open_array = diff(logi);
open_array = find(open_array);
open_array = diff(open_array);
open_array(rem(find(open_array),2)==0)=[];

clogi = [1 state ~state(end)];
clogi = logical(clogi);

closed_array = diff(clogi);
closed_array = find(closed_array);
closed_array = diff(closed_array);
closed_array(rem(find(closed_array),2)==0)=[];

%Close time histogram
figure;
hC = histogram(closed_array*dt);
t_c = 0: hC.BinEdges(end)/100 : hC.BinEdges(end);
histogram(closed_array*dt, "Normalization", 'pdf');
xlabel('Duration (ms)')
ylabel('pdf for closed state')
title('Statistics of the closed state', 'FontSize',14)
hold on;
plot(t_c, alpha*exp(-alpha*t_c),'r', 'LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf', 'FontSize',12)

%Open time histogram
figure;
hO = histogram(open_array*dt);
t_o = 0: hO.BinEdges(end)/100 : hO.BinEdges(end);
histogram(open_array*dt, "Normalization", 'pdf');
xlabel('Duration (ms)')
ylabel('pdf for open state')
title('Statistics of the open state', 'FontSize',14)
hold on;
plot(t_o, beta*exp(-beta*t_o),'r','LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf','FontSize',12)

end
```

TwoOpenStateModel.m

```matlab
function TwoOpenStateModel(dt,T,alpha,beta,gamma,delta,Io)

%Syntax: TwoOpenStateModel(dt,T,alpha,beta,gamma,delta,Io)
%
%dt    is the simulation time step (in ms) supposing that no more than a
%      single closure or opening event can occur in this time period (Markov event);
%T     is the total simulation time (in ms);
%alpha is the transition rate C->O1 in dt;
%beta  is the transition rate O1->C in dt;
%gamma is the transition rate O1->O2 in dt;
%delta is the transition rate O2->O1 in dt;
%Io    is the current of the open state (in pA).
%
%Example:   TwoOpenStateModel(0.1,1000,0.02,0.02,0.004,0.004,50); to look at a single event
%           TwoOpenStateModel(0.1,100000,0.02,0.02,0.004,0.004,50); to look at the
%           statistics

N = ceil(T/dt);          %Number of steps
stateO1 = zeros(1,N);    %Array of state O1 (1=channel is in O1, 0= channel either C or O2)
stateO2 = zeros(1,N);    %Array of state O2 (1=channel is in O2, 0= channel either C or O1)
curr_state = 0;          %state at current step (0=C, 1=O1, 2=O2)
t = dt;                  %State time
T_array = dt*(1:N);      %array of absolute time
Pc1 = alpha*dt;          %C -> O1 probability
P1c = beta*dt;           %O1-> C  probability
P12 = gamma*dt;          %O1-> O2 probability
P21 = delta*dt;          %O2-> O1 probability
r = rand(1,N);           %array of random numbers between 0 and 1

%Compute channel state throughout simulation time
for i =1:N-1
    if curr_state == 0      %if state at current iteration is closed
        if r(i)<=Pc1
            curr_state = 1;
            stateO1(i+1) = 1;
            t = dt;
        else
            curr_state = 0;
            t = t+dt;
        end
    elseif curr_state == 1  %if state at current iteration is O1
        if r(i)<=P1c
            curr_state = 0;
            t = dt;
        elseif (r(i)>P1c) && (r(i)<=(P1c+P12))
            curr_state = 2;
            stateO2(i+1) = 1;
            t = dt;
        else
            curr_state = 1;
            stateO1(i+1)= 1;
            t = t+dt;
        end
    else                    %if state at current iteration is O2
        if r(i)<=P21
            curr_state = 1;
            stateO1(i+1)= 1;
            t = dt;
        else
            curr_state = 2;
            stateO2(i+1)= 1;
            t = t+dt;
        end

    end
end

stateO = stateO1 + stateO2; %stateO is 0 when closed and 1 when open (O1 or O2)
```

```matlab


%Plot the temporal evolution of single channel current
figure;
real_current = Io*(stateO + (randn(1,N))*0.05); %randn to add normally-distributed random noise
horO = yline(Io,'--r','Open');
horO.LineWidth = 1;
hold on;
horC = yline(0,'--r','Closed');
horC.LineWidth = 1;
hold on;
plot(T_array,real_current);
xlabel('Time (ms)')
ylabel ('Current (pA)')
title ('Single channel current', 'FontSize',14)
axis([-T/10 T+T/5 -Io/2 Io+Io/2]);
pbaspect([2 1 1])
grid on

%Plot statistics
figure;
vertO = xline(Io,'--r','(O)');
vertO.LineWidth = 1;
vertO.LabelOrientation = 'horizontal';
hold on;
vertC = xline(0,'--r','(C)');
vertC.LineWidth = 1;
vertC.LabelOrientation = 'horizontal';
hold on;
histogram(real_current, 100);
xlabel('Current (pA)')
ylabel ('Absolute frequency')
title ('Histogram of the channel current', 'FontSize',14)


%Compute duration of the three states
%OPEN TOT
logi = logical(stateO);

open_array = diff(logi);
open_array = find(open_array);
open_array = diff(open_array);
open_array(rem(find(open_array),2)==0)=[];

%CLOSED
clogi = [1 stateO ~stateO(end)]; %to count also first closure event
clogi = logical(clogi);

closed_array = diff(clogi);  %array with 0 btwn steps where state doesn't change, 1 when 0->1, -1 when 1->0
closed_array = find(closed_array);  %array with positions of nonzero elements (where state changes)
closed_array = diff(closed_array);
closed_array(rem(find(closed_array),2)==0)=[]; %at odd positions we have durations of closed states

%OPEN
state_durat = stateO1 + 3*stateO2;  %now C = 0, O1 = 1, O2 = 3
state_durat = diff(state_durat);    %array where 0: no transition,
                                    %             1: trans C -> O1,  -1: trans O1 -> C,
                                    %             2: trans O1 -> O2, -2: trans O2 -> O1
%OPEN1
O1_durat = find(state_durat==1|state_durat==-1);  %an open state starting with O1 goes from 1 to -1
O1_durat = diff(O1_durat);
O1_durat (rem(find(O1_durat),2)==0)=[]; %at odd positions we have durations of open states

%OPEN2
O2_durat = [];
for i = 1:N-1
    if state_durat(i)==2
        for j = i:N-1
            if state_durat(j)== -1
```

```
                a = j-i;
                O2_durat =[O2_durat a];
                i = j+1;
                break
            end
        end
    end
end

%Define coefficients for solution of differential equation
b = beta + delta + gamma;
D = sqrt(b^2 - 4*beta*delta);
lambda1 = (-b+D)/2;
lambda2 = -(b+D)/2;

a1 = -(lambda2 + beta + gamma)/(lambda1 - lambda2);
a2 = (lambda1 + beta + gamma)/(lambda1 - lambda2);
b1 = gamma/(lambda1 - lambda2);
b2 = -gamma/(lambda1 - lambda2);

%Closed time histogram
figure;
[f_C, edgC] = histcounts(closed_array*dt,50);
t_c = edgC(1:end-1) + diff(edgC)/ 2;
histogram(closed_array*dt,50,'Normalization','pdf');
xlabel('Duration (ms)')
ylabel('Absolute frequency')
ylabel('pdf for closed state')
title('Statistics of the closed state', 'FontSize',14)
hold on;
plot(t_c, alpha*exp(-alpha*t_c),'r','LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf','FontSize',12)

%O1 time histogram
figure;
[f_O1, edgO1] = histcounts(O1_durat*dt,80,'Normalization','pdf');
t_o1 = edgO1(1:end-1) + diff(edgO1)/ 2;
histogram(O1_durat*dt, 80, 'Normalization','pdf');
xlabel('Duration (ms)')
ylabel('pdf for state O1')
title('Statistics of the O1 state', 'FontSize',14)
hold on;
PO1 = a1*exp(lambda1*t_o1) + a2*exp(lambda2*t_o1);
Area = trapz(t_o1,PO1);
PO1_normalized = PO1./Area;
plot(t_o1, PO1_normalized,'r','LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf', 'FontSize',12)

%O2 time histogram
figure;
[f_O2, edgO2] = histcounts(O2_durat*dt,25,'Normalization','pdf');
t_o2 = edgO2(1:end-1) + diff(edgO2)/ 2;
histogram(O2_durat*dt,25,'Normalization','pdf');
xlabel('Duration (ms)')
ylabel('pdf for state O2')
title('Statistics of the O2 state', 'FontSize',14)
hold on;
t = linspace(0,edgO2(end-1),100);
PO2 = b1*exp(lambda1*t) + b2*exp(lambda2*t);
Area = trapz(t,PO2);
PO2_normalized = PO2./Area;
plot(t, PO2_normalized,'r','LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf', 'FontSize', 12)

%Total open time histogram
figure;
[f_O, edgO] = histcounts(open_array*dt,50);
t_o = edgO(1:end-1) + diff(edgO)/ 2;
histogram(open_array*dt,50,'Normalization','pdf');
xlabel('Duration (ms)')
```

```
ylabel('pdf for open state')
title('Statistics of the open state', 'FontSize',14)
hold on;
PO = -(lambda1*(a1*exp(lambda1*t_o) + b1*exp(lambda1*t_o)) + lambda2*(a2*exp(lambda2*t_o) + b2*exp(
    lambda2*t_o)));
plot(t_o, PO,'r','LineWidth',1.5);
legend('Histogram of simulated duration','Theoretical pdf', 'FontSize',12)

end
```

ThreeStateModel.m

```
function ThreeStateModel(dt,T,alpha,beta,delta,gamma,s,Io)

%Syntax: ThreeStateModel(dt,T,alpha,beta,delta,gamma,s,Io)
%
%dt    is the simulation time step (in ms) supposing that no more than a
%      single closure or opening event can occur in this time period (Markov event);
%T     is the total simulation time (in ms);
%alpha is the transition rate C->S in dt;
%beta  is the transition rate S->C in dt;
%gamma is the transition rate S->O in dt;
%delta is the transition rate O->S in dt;
%Io    is the current of the fully open state (in pA);
%s     fraction of total current Io in the substate.
%
%Example:   ThreeStateModel(0.1,1000,0.02,0.04,0.03,0.02,0.5,50); to look at a single event
%           ThreeStateModel(0.1,100000,0.02,0.04,0.03,0.02,0.5,50); to look at the
%           statistics

N = ceil(T/dt);          %Number of steps
stateS = zeros(1,N);     %Array of state S (1=channel is in S, 0= channel either C or O)
stateO = zeros(1,N);     %Array of state O (1=channel is in O, 0= channel either C or S)
curr_state = 0;          %Initial state is set as closed (0=C, 1=S, 2=O)
t = dt;                  %State time
T_array = dt*(1:N);      %array of absolute time
Pcs = alpha*dt;          %C->S probability
Psc = beta*dt;           %S->C probability
Pso = gamma*dt;          %S->O probability
Pos = delta*dt;          %O->S probability
r = rand(1,N);           %Array of random numbers between 0 and 1

%Compute channel state at all time steps
for i =1:N-1
    if curr_state == 0
        if r(i)<=Pcs
            curr_state = 1;
            stateS(i+1) = 1;
            t = dt;
        else
            curr_state = 0;
            t = t+dt;
        end
    elseif curr_state == 1
        if r(i)<=Psc
            curr_state = 0;
            t = dt;
        elseif (r(i)>Psc) && (r(i)<=(Psc+Pso))
            curr_state = 2;
            stateO(i+1) = 1;
            t = dt;
        else
            curr_state = 1;
            stateS(i+1) = 1;
            t = t+dt;
        end
    else
        if r(i)<=Pos
            curr_state = 1;
            stateS(i+1) = 1;
            t = dt;
        else
            curr_state = 2;
            stateO(i+1) = 1;
            t = t+dt;
        end

    end
end

```

```
state = stateO + s*stateS; %state is 0 when closed, s when S and 1 when open

%Plot the temporal evolution of single channel current
figure;
real_current = Io*(state+(randn(1,N))*0.05); %add normally-distributed random noise
horO = yline(Io,'--r','Open');
horO.LineWidth = 1;
horO = yline(s*Io,'--r','Substate');
horO.LineWidth = 1;
hold on;
horC = yline(0,'--r','Closed');
horC.LineWidth = 1;
hold on;
plot(T_array,real_current);
xlabel('Time (ms)')
ylabel ('Current (pA)')
title ('Single channel current', 'FontSize',14)
axis([-T/10 T+T/3 -Io/2 Io+Io/2]);
pbaspect([2 1 1])
grid on

%Plot statistics
figure;
vertO = xline(Io,'--r','(O)');
vertO.LineWidth = 1;
vertO.LabelOrientation = 'horizontal';
hold on;
vertS = xline(Io*s,'--r','(S)');
vertS.LineWidth = 1;
vertS.LabelOrientation = 'horizontal';
hold on;
vertC = xline(0,'--r','(C)');
vertC.LineWidth = 1;
vertC.LabelOrientation = 'horizontal';
hold on;
histogram(real_current, 100);
xlabel('Current (pA)')
ylabel ('Absolute frequency')
title ('Histogram of the channel current', 'FontSize',14)


%Compute duration of the three states
%Duration of fully open state
logiO = logical(stateO);

O_array = diff(logiO);
O_array = find(O_array);
O_array = diff(O_array);
O_array (rem(find(O_array),2)==0)=[];

%Duration of open substate
logiS = logical(stateS);

S_array = diff(logiS);
S_array = find(S_array);
S_array = diff(S_array);
S_array (rem(find(S_array),2)==0)=[];

%Duration of closed state
clogi = [1 state ~state(end)];
clogi = logical(clogi);

C_array = diff(clogi);
C_array = find(C_array);
C_array = diff(C_array);
C_array(rem(find(C_array),2)==0)=[];

%Eigenvalue problem
Q = [-alpha beta 0; alpha -beta-gamma delta; 0 gamma -delta];
l = eig(Q);
```

```
disp(l);

%Closed time histogram
figure;
hC = histogram(C_array*dt,40);
[f_C, edgC] = histcounts(C_array*dt,40);
tc = edgC(1:end-1) + diff(edgC)/ 2;
histogram(C_array*dt,40);
xlabel('Duration (ms)')
ylabel('Absolute frequency')
title('Statistics of the closed state', 'FontSize',14)
hold on;
% Simulated points
tc = tc.';
yc = f_C;
yc = yc.';
plot( tc , yc, 'ro')
hold on;
% Fit with exponential
Fc = @(xc,xdata)xc(1)*exp(l(1)*xdata) + xc(2)*exp(l(1)*xdata) + xc(3)*exp(l(3)*xdata);
x0 = [1 1 1]; %initial conditions
[xc,resnorm,~,~,output] = lsqcurvefit(Fc,x0,tc,yc)

plot(tc,Fc(xc,tc), 'color', 'r','LineWidth',1.5)
legend('Histogram of simulated duration', 'Simulated frequencies' ,'Distribution fit', 'FontSize',12)

%Open substate time histogram
figure;
hS = histogram(S_array*dt,40);
[f_S, edgS] = histcounts(S_array*dt,40);
ts = edgS(1:end-1) + diff(edgS)/ 2;
histogram(S_array*dt,40);
xlabel('Duration (ms)')
ylabel('Absolute frequency')
title('Statistics of the substate state', 'FontSize',14)
hold on;
% Simulated points
ts = ts.';
yo = f_S;
yo = yo.';
plot( ts , yo, 'ro')
hold on;
% Fit with exponential
Fs = @(xs,xdata)xs(1)*exp(l(1)*xdata) + xs(2)*exp(l(1)*xdata) + xs(3)*exp(l(3)*xdata);
x0 = [1 1 1]; %initial conditions
[xs,resnorm,~,exitflag,output] = lsqcurvefit(Fs,x0,ts,yo)

plot(ts,Fs(xs,ts), 'color', 'r','LineWidth',1.5)
legend('Histogram of simulated duration', 'Simulated frequencies' ,'Distribution fit', 'FontSize',12)

%Open time histogram
figure;
hO = histogram(O_array*dt,40);
[f_O, edgO] = histcounts(O_array*dt,40);
to = edgO(1:end-1) + diff(edgO)/ 2;
histogram(O_array*dt,40);
xlabel('Duration (ms)')
ylabel('Absolute frequency')
title('Statistics of the open state', 'FontSize',14)
hold on;
% Simulated points
to = to.';
yo = f_O;
yo = yo.';
plot( to , yo, 'ro')
hold on;
% Fit with exponential
Fo = @(xo,xdata)xo(1)*exp(l(1)*xdata) + xo(2)*exp(l(1)*xdata) + xo(3)*exp(l(3)*xdata);
x0 = [1 1 1]; %initial conditions
[xo,resnorm,~,exitflag,output] = lsqcurvefit(Fo,x0,to,yo) %run solver
```

```
plot(to,Fs(xo,to), 'color', 'r','LineWidth',1.5)
legend('Histogram of simulated duration', 'Simulated frequencies' ,'Distribution fit', 'FontSize',12)

end
```

ManyChannels.m

```
function ManyChannels(dt,T,alpha,beta,gamma,delta,C,Io)

%Syntax: ManyChannels(dt,T,alpha,beta,gamma,delta,C,Io)
%
%dt    is the simulation time step (in ms) supposing that no more than a
%      single closure or opening event can occur in this time period (Markov event);
%T     is the total simulation time (in ms);
%alpha is the transition rate C->O1 in dt;
%beta  is the transition rate O1->C in dt;
%gamma is the transition rate O1->O2 in dt;
%delta is the transition rate O2->O1 in dt;
%C     is the number of channel simulated
%Io    is the current of the open state (in pA).
%
%Example:   ManyChannels(0.1,1000,0.02,0.04,0.002,0.004,5,50); to look at 5 channels

N = ceil(T/dt);          %Number of steps
states = zeros(C,N);     %Array of states (1=channel is in O1, 0= channel either C or O2)
curr_state = 0;          %state at current step (0=C, 1=O1, 2=O2)
t = dt;                  %State time
T_array = dt*(1:N);      %Array of absolute time
Pc1 = alpha*dt;          %C -> O1 probability
P1c = beta*dt;           %O1-> C  probability
P12 = gamma*dt;          %O1-> O2 probability
P21 = delta*dt;          %O2-> O1 probability

%Compute channel state at all time steps
for rep = 1:C-1
    r = rand(1,N);           %array of random numbers between 0 and 1
    for i =1:N-1
        if curr_state == 0
            if r(i)<=Pc1
                curr_state = 1;
                states(rep,i+1) = 1;
                t = dt;
            else
                curr_state = 0;
                t = t+dt;
            end
        elseif curr_state == 1
            if r(i)<=P1c
                curr_state = 0;
                t = dt;
            elseif (r(i)>P1c) && (r(i)<=(P1c+P12))
                curr_state = 2;
                states(rep,i+1) = 1;
                t = dt;
            else
                curr_state = 1;
                states(rep,i+1) = 1;
                t = t+dt;
            end
        else
            if r(i)<=P21
                curr_state = 1;
                states(rep,i+1) = 1;
                t = dt;
            else
                curr_state = 2;
                states(rep,i+1) = 1;
                t = t+dt;
            end
        end
    end
end

current = sum(states);
real_current = Io*current + Io*(randn(1,N))*0.05; %add normally-distributed random noise
```

```
real_current = real_current(2:end); %remove 0 at beginning
T_array = T_array(2:end);
Iavg = mean(real_current);

%Plot the temporal evolution of the total current
figure;
plot(T_array,real_current);
horO = yline(Iavg,'--r',['Average'; 'current']);
horO.LineWidth = 1;
hold on;
xlabel('Time (ms)')
ylabel ('Current (pA)')
title (['Current from ', num2str(C) ,' channels'], 'FontSize',14)
xlim([-T/10 T+T/5]);
pbaspect([2 1 1])
grid on


end
```