Comparing event detection methods in single-channel analysis using simulated data

Candidate:

Mathieu Francis Dextraze

Supervisor:

Corrie daCosta

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the

Master's degree in Chemistry

Department of Chemistry and Biomolecular Sciences

Faculty of Science

University of Ottawa

ABSTRACT

With more states revealed, and more reliable rates inferred, mechanistic schemes for ion channels have increased in complexity over the history of single-channel studies. At the forefront of single-channel studies we are faced with a temporal barrier delimiting the briefest event which can be detected in single-channel data. Despite improvements in single-channel data analysis, the use of existing methods remains sub-optimal. As existing methods in single-channel data analysis are unquantified, optimal conditions for data analysis are unknown. Here we present a modular single-channel data simulator with two engines; a Hidden Markov Model (HMM) engine, and a sampling engine. The simulator is a tool which provides the necessary *a priori* information to be able to quantify and compare existing methods in order to optimize analytic conditions. We demonstrate the utility of our simulator by providing a preliminary comparison of two event detection methods in single-channel data analysis; Threshold Crossing and Segmental k-means with Hidden Markov Modelling (SKM-HMM).

ACKNOWLEDGEMENTS

I would like to thank Professor Corrie DaCosta for allowing me to work with him on this project. I am truly grateful for Dr. daCosta's support and supervision over the course of these last couple of years. He has done a marvelous job of providing me with resistance in my learning process. Thank you Corrie.

I would like to thank Melissa McNulty for sharing her knowledge in Molecular Biology. She is a real wizard when it comes to Molecular Biology. Because of her professional tips, my Polymerase Chain Reactions (PCR) have never failed. Thank you Melissa.

I would like to thank my peers Christian Tessier and Johnathon Emlaw. Christian Tessier has provided me with his real patch-clamp data for which he has worked countless hours to acquire. Both Johnathon and Christian have provided me with critical thoughts and have helped me to strengthen my research. Thank you Christian and Johnathon.

I would like to thank Kyle Briggs in the Department of Physics at the University of Ottawa for helping me to understand some aspects of the system properties of the Axopatch 200B. Thank you Kyle.

I would like to thank Remigijus Lape in the Department of Neuroscience, Physiology, and Pharmacology at the University College London for helping me by correspondence. He has helped with setting up the Time-Course Fitting software SCAN[33]. Thank you Remis.

I would like to thank Josee Rouleau and Anette Campeau in the Administration in the Department of Chemistry and Biomolecular Sciences at the University of Ottawa. They are amazing people. Always greeting me with a smile and providing great service. I truly appreciate the work that they do everyday for everyone in the department. Thank you Josee and Anette.

I would like to thank Dr. John Baenziger and Dr. Roberto Chica for being on my Thesis Advisory Committee and providing me with guidance.

I would like to thank Dr. Tom Woo and Dr. Vincent Tabard-Cossa for agreeing to be examiners of my thesis. Thank you both for taking the time to read through this document.

TABLE OF CONTENTS

TITLE PAGE.....1

ABSTRACT.....2

ACKNOWLEDGEMENTS.....3

TABLE OF CONTENTS.....4

1 INTRODUCTION.....10
1.1 Context.....10
1.1.1 Ion Channels.....10
1.1.2 Muscle-type acetylcholine receptor.....13
1.1.3 Patch-clamp.....15
1.1.4 Single-channel data.....20
1.1.5 Single-channel kinetics.....24
1.1.6 Single-channel data analysis.....30
1.2 Goal of the project.....33
1.2.1 The problem of brief events in single-channel kinetics.....33
1.2.2 A brief history of single-channel kinetics.....36
1.2.3 Our approach to address the problem of brief events in single-channel kinetics.....38
1.2.4 Designing a simulator.....39
1.2.5 Preliminary comparison of detection methods.....42

2 METHODS.....44
2.1 Simulator.....44
2.1.1 HMM engine.....44
2.1.1.1 Randomness.....44
2.1.1.2 Model files.....44
2.1.1.3 Probability matrix.....45
2.1.1.4 State probability distribution (SPD).....47
2.1.1.5 Conductance probability distribution (CPD).....49
2.1.1.6 State lifetime random sample set.....52
2.1.1.7 State transition random sample set.....53
2.1.2 Sampling engine.....54
2.1.2.1 Step-response matching.....54
2.1.2.2 Noise modeling.....57
2.1.2.3 Simulation of step-response.....58
2.1.2.4 Simulation of noise.....59
2.1.2.5 Application of filters.....60
2.1.2.6 Power Spectral density.....61
2.2 Preliminary comparison of event detection methods.....62
2.2.1 HMM settings.....62
2.2.2 Sampling settings.....62
2.2.3 TAC detection settings.....63
2.2.4 QUB detection settings.....64
2.2.5 Accuracy plots.....65
2.2.6 Missed and false event plots.....66
2.2.7 MIL settings.....67
2.2.8 Kinetic parameter plots.....67

3 RESULTS....68
3.1 Simulator....68
3.1.1 HMM engine....68
3.1.1.1 HMM simulation algorithm....68
3.1.1.2 Transition sequence of events....71
3.1.1.3 State distributions....73
3.1.1.4 Conductance distributions....77
3.1.2 Sampling engine....80
3.1.2.1 Matching measured step-response....80
3.1.2.2 Matched measured noise....82
3.1.2.3 Sampling interval....84
3.1.2.4 Sampling algorithm....87
3.2 Preliminary comparison of event detection methods....89
3.2.1 Overview of detection results....89
3.2.2 Missed and false events....93
3.2.3 Accuracy of detected events....96
3.2.4 Recovery of kinetic parameters....100

4 DISCUSSION....104
4.1 Simulator....104
4.1.1 HMM engine....104
4.1.2 Sampling engine....106
4.1.3 General discussion of the simulator....108
4.2 Preliminary comparison of event detection methods....110
4.2.1 Missed and false events....110
4.2.2 Resolution and accuracy of detected events....112
4.2.3 Recovered kinetic parameters....113
4.3 Future work....114
4.4 Summary....115

5 APPENDIX....116

6 REFERENCES....152

LIST OF FIGURES

Figure 1. Schematic of ion channel activity. ……..……12

Figure 2. Ribbon diagram of the muscle-type nAChR. ……..……14

Figure 3. Schematic of the cell-attached single-channel patch-clamp. ……17

Figure 4. Components of the patch-clamp step response. ……...……18

Figure 5. Power spectrum of real patch-clamp noise. ……...……19

Figure 6. Example rectangular pulses with simulated response. ……..……22

Figure 7. Filtered step-response. ……23

Figure 8. A kinetic scheme and its matrix notation. ……..……28

Figure 9. Example SPD and CPD. ……29

Figure 10. Depiction of event detection methods. ……32

Figure 11. Illustration of a missed and a false event. ……35

Figure 12. Historical evolution of the nAChR kinetic scheme. ……..……37

Figure 13. Schematic of the simulator. ……..……41

Figure 14. Visual depiction of the step-response fitting procedure. ……56

Figure 15. Example output of the HMM engine. ……..……...…70

Figure 16. Results of the state distribution test part 1. ……75

Figure 17. Results of the state distribution test part 2. ……76

Figure 18. Results from the conductance distribution test. ……..……79

Figure 19. Fit results over a measured step-response at 1 µs sampling interval. ……81

Figure 20. Autoregressive modeling of real noise. ……83

Figure 21. Simulated events at different sampling intervals. ……...……85

Figure 22. Results of the sampling interval test. ……...……86

Figure 23. Depiction of the sampling process. ……..……88

Figure 24. Events detected across a range of filter cut-off frequencies. ……90

Figure 25. Events detected across a range of ligand concentration. ……91

Figure 26. Dwell duration histograms for detected events. ……92

Figure 27. Dwell duration histograms for detected events. ……93

Figure 28. Missed and false events from Threshold Crossing and SKM-HMM. ……95

Figure 29. Accuracy plots of detected events from SKM-HMM and Threshold Crossing. ……98

Figure 30. Accuracy plot comparing SKM-HMM and Threshold Crossing for events in the range of 10^{-4} to 10^{-5} seconds. ……99

Figure 31. Kinetic parameter plots from detected events. ……102

Figure 32. Kinetic parameter plot from *a priori* exact event durations. ……103

Figure 33. Comparison of real and simulated nAChR single-channel data. ……149

Figure 34. Example of simulated single-channel data with two channels. ……150

Figure 35. Example of simulated single-channel data with subconductance levels. ……151

LIST OF TABLES

Table 1. Transition rate matrix (Q) of the SPM at 3 μM [ACh]. ……..……45

Table 2. Probability matrix (P) of the SPM at 3 μM [ACh]. ……...…46

Table 3. Baseline and open channel standard deviations for simulated noise across a range of digital Gaussian filter cut-off frequencies. ……64

Table 4. Numbers of true events used for accuracy plots across a range of digital Gaussian filter cut-off frequencies. ……...…96

Table 5. Convergence of MIL for SKM-HMM and Threshold-Crossing. ……...……100

Table 6. S/N ratio used for the x-axis of missed and false event plots. ……..……129

Table 7. Dead-times used for kinetic fitting in MIL from events detected by SKM-HMM. ……130

Table 8. Percent difference between expected and observed means of the SPD for the all-connected model (see 5.1 (2)). ……..……139

Table 9. Percent difference between expected and observed parameter λ of the SPD for the all-connected model (see 5.1 (2)). ……..……140

Table 10. Percent difference between expected and observed weight α of the SPD for the all-connected model (see 5.1 (2)). ……..……141

Table 11. Percent difference between expected and observed means of the SPD for the SPM model at 3 μM [ACh] (see 5.1 (1)). ……...……142

Table 12. Percent difference between expected and observed parameter λ of the SPD for the SPM model at 3 μM [ACh] (see 5.1 (1)). ……..…..……143

Table 13. Percent difference between expected and observed weight α of the SPD for the SPM model at 3 μM [ACh] (see 5.1 (1)). ……144

Table 14. Missed events for detections from SKM-HMM and Threshold Crossing (1/2). ……145

Table 15. Missed events for detections from SKM-HMM and Threshold Crossing (2/2). ……..……146

Table 16. False events for detections from SKM-HMM and Threshold Crossing (1/2). ……..……147

Table 17. False events for detections from SKM-HMM and Threshold Crossing (2/2). ……..…158

LIST OF TERMS AND SYMBOLS

nAChR: Nicotinic Acetylcholine Receptor.

τ: The RC time constant.

S/N: Signal to noise ratio. The mean open channel current divided by the standard deviation of the baseline current.

Rise time: the time it takes to go from 10 to 90% of the maximum amplitude.

SPD: State Probability Distribution

CPD: Conductance Probability Distribution

HMM: Hidden Markov Model

SKM-HMM: Segmental k-means with Hidden Markov Modeling

Event: An ion channel's residency in a particular state

Dwell: An ion channel's residency in a particular conductance level

AR model: Autoregressive model

SPM: Standard Prime Model

PSD: Power Spectral Density

FRET: Fluorescence Resonance Energy Transfer

ACh: Acetylcholine

1 INTRODUCTION

1.1 CONTEXT

1.1.1 Ion channels

Ion channels are transmembrane proteins composed of one or more subunits found on the surface of excitable cells[2]. At a molecular level, their function is like that of an aqueous channel. The hydrophobic nature of the cell membrane renders it relatively impermeable to ions. Ion channels thereby provide ions with a permeable pathway to diffuse across the cell membrane, down their electrochemical gradients.

Ion channels undergo conformational changes, and can exist in a variety of states that can be broadly divided into two classes: “open” or “closed” (Figure 1). When the channel’s pore forms an aqueous connection between the intra- and extracellular environments, the channel is in an open state. When the channel’s pore does not form an aqueous connection between the intra- and extracellular environments, the channel is said to be in a closed state. Ion channels may have multiple distinct open and closed states[1].

Ion channels reversibly transition between active open and inactive closed states[3], a process referred to as gating. The rates of transition between open and closed states are typically governed by external factors. In other words, external factors regulate the gating of ion channels. Ion channels are commonly classified by the external factor that regulates their gating. Known mechanisms of gating regulation involve factors such as changes in the transmembrane voltage, ambient temperature, or the presence of extracellular ligands[4,5].

Ion channels are diverse, and have important roles in multiple biological systems[8,9]. In the nervous system, voltage-gated ion channels are responsible for propagating the action potential[6]. Ligand-gated ion channels mediate chemo-electric coupling between cells, allowing them to communicate with one another via a chemical synapse[7]. Thus, ion channels are important molecular components to the nervous system.

Given that ion channels have essential and diverse roles in many physiological processes, it is not surprising that they comprise an important class of drug targets[10]. Research on ion channels is important because dysfunction of ion channels leads to a variety of diseases[11]. The list of ion channel

related diseases is large enough to warrant the coining of a term “channelopathy.” Single molecule kinetic studies of ion channels have helped to develop therapeutic drugs[22,23,24], as well as an understanding of their mechanism of action.

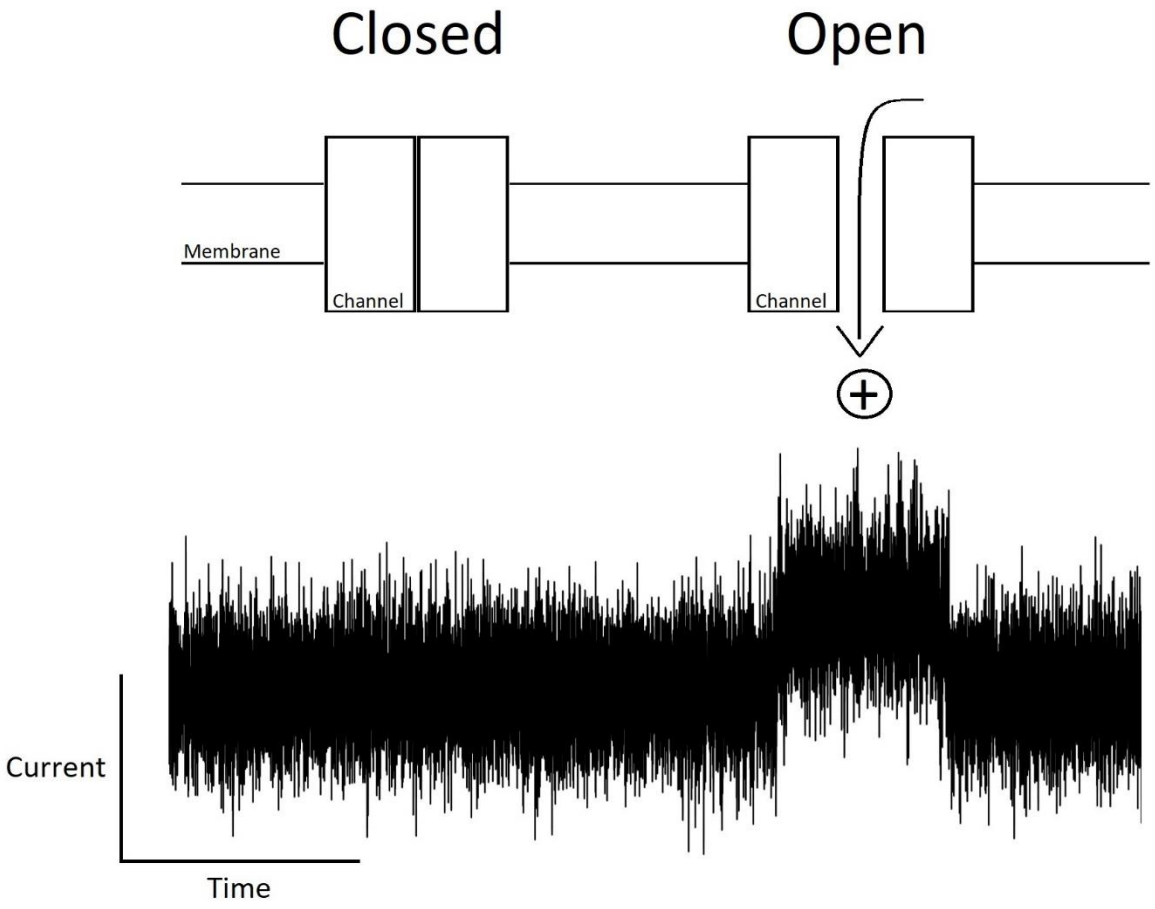

Figure 1. Schematic of ion channel activity. The top image shows a cross-section of the cell membrane with a closed and an open channel. The bottom image shows an example of real single-channel activity recorded from the patch-clamp.

1.1.2 Muscle-type acetylcholine receptor

For my research I have chosen to focus on the muscle-type nicotinic acetylcholine receptor (nAChR) as a model ion channel. The muscle-type nAChR is found at the neuromuscular junction (a chemical synapse between a motor neuron and a muscle fiber,) where it converts the binding of acetylcholine into the initial depolarization of muscle cells that leads to muscle contraction[13].

The muscle-type nAChR is a pentameric ligand-gated ion channel whose gating activity is regulated by the binding of the agonist acetylcholine. It is composed of five subunits arranged pseudo-symmetrically in a barrel shape containing a central pore[12] (Figure 2). In the embryonic form, the muscle-type nAChR is composed of two α1 subunits, and one of each of β1, γ, and δ. In the adult form, the γ subunit is replaced with the ε subunit[16]. The muscle-type nAChR has two acetylcholine binding sites; one found at the interface between the α1 and ε subunits, the other at the interface between the α1 and the δ interface[14,15].

The muscle-type nAChR is part of a broader class of nAChRs containing both muscle- and neuronal-type nAChRs. Diseases involving dysfunctional nAChRs include neurodegenerative diseases, psychiatric diseases, as well as diseases whose symptoms include muscle weakness, and epilepsy[17]. Involvement of nAChRs in these diseases has motivated research in the development of therapeutic agonists and antagonists targeting nAChRs[18,19,27]. Extensive single-channel research has been conducted on the muscle-type nAChR, and it is a well enough characterized model ion channel for studying methods in single-channel data analysis[25,26,27,29].

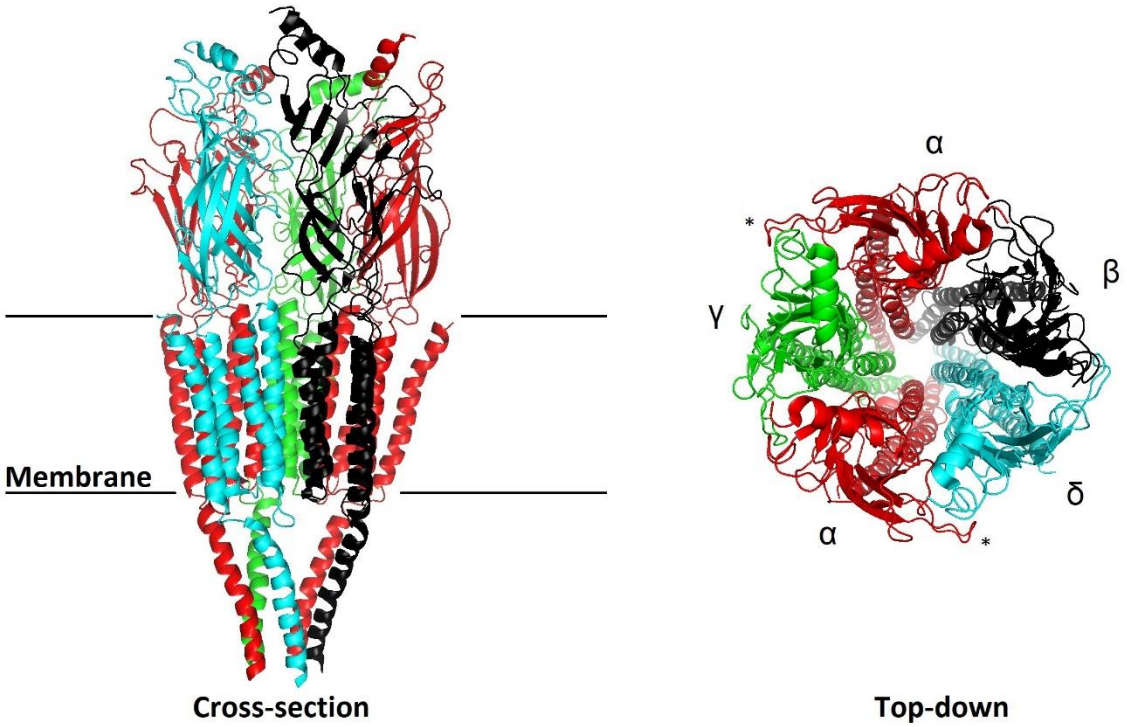

Figure 2. Ribbon diagram of the muscle-type nAChR. On the left, a cross-section of the nAChR in the cell-membrane. On the right, a top-down view into the pore of the channel. Sub-units are labeled as per their Greek letter notation. ACh binding sites are labeled as *. The structure is resolved by Cryo-electron microscopy at 4 Å resolution. (PDB: 2bg9, ref. 12)

1.1.3 Patch-clamp

The patch-clamp technique is a method that was developed by Erwin Neher and Bert Sakmann, for which they were awarded the 1991 Nobel Prize in Physiology and Medicine[32]. While patch-clamping was originally used to study ion channels, it has become a versatile technique used to study the electrophysiological properties of both biological, and synthetic membranes[30,31]. Additionally, patch-clamping is mainly used to study the properties of ion channels at the single molecule level.

The cell-attached patch-clamp technique for single-channel measurements involves the formation of a gigaohm (GΩ) seal between the opening of a glass micro-pipette and a small patch of cell membrane containing the ion channel of interest. Inside the glass micro-pipette is a solution containing ions (typically Cl^-, K^+, and Na^+,) and potential agonists, antagonists, or other small-molecules. In the modern patch-clamp set-up, the glass micro-pipette is connected to an integrated head-stage amplifier which detects pipette currents through an electrode (Figure 3).

The circuitry of the patch-clamp used to detect and record single-channel data that is summarized here is described in detail elsewhere[33]. Briefly, the modern patch-clamp employs a capacitor-feedback amplifier design to record single-channel currents. The capacitor-feedback amplifier is a low-noise current-to-voltage converter, which converts small currents from a patch into voltages which are sampled by a computer. Voltage values are then digitally converted back into current values, which are the values we see in a trace of single-channel data. Due to its circuitry, the patch-clamp has well characterized, and predictable system properties[35,36,37].

Predictable system properties are essential in order to simulate patch-clamp single-channel data. Particularly, it is important to know the step response; how the system responds to a step change in current (i.e. when the channel opens or closes). Additionally, as noise in a recorded signal is unavoidable, it is important to know the characteristics of the noise in order to accurately mimic real noise.

The step response of the patch-clamp has the form of an RC step response (a ubiquitous response for many systems.) The term RC refers to the resistance and capacitance of a resistor in series with a capacitor, which defines a constant that describes the response time of the step response. The functional form of the RC step response (Equation 1) is an exponential with a time

constant τ that has units of time. The complicated circuitry of the patch-clamp ensures that the time constant τ stays as low as possible (roughly 1.6 μs in the modern patch-clamp[33]).

$$i(t) = c(1 - \mathrm{e}^{-t/\tau}) \quad [1]$$

Where *i(t)* is the current as a function of time *t* with an exponential time constant τ and step size *c*.

The main source of noise in the patch-clamp, Johnson-Nyquist noise, is due to thermal agitation of the atoms in the conductors of the patch-clamp setup (these conductors include the electrode inside the pipette, as well as the wires of the internal circuitry of the head-stage)[40]. This Johnson-Nyquist noise passes through a differentiator amplifier in the head-stage which introduces a high-frequency bias to the noise. As a result of differentiation, the noise increases as a function of the square of the frequency (commonly referred to as f^2 noise. Figure 5)

The Axopatch™ 200B from Axon Instruments ® has a built-in analog 4-pole Bessel filter which has variable cutoff frequencies in the range of 1 to 100 kHz. The analog Bessel filter has an effect on the shape of the step response, and on the frequency characteristics of the noise. The Bessel filter will cause the step response to have a sigmoidal-like shape (Figure 4). The Bessel filter will also cause the amplitude of noise with frequencies above the cutoff frequency to decay at 80 dB/decade (Figure 5).

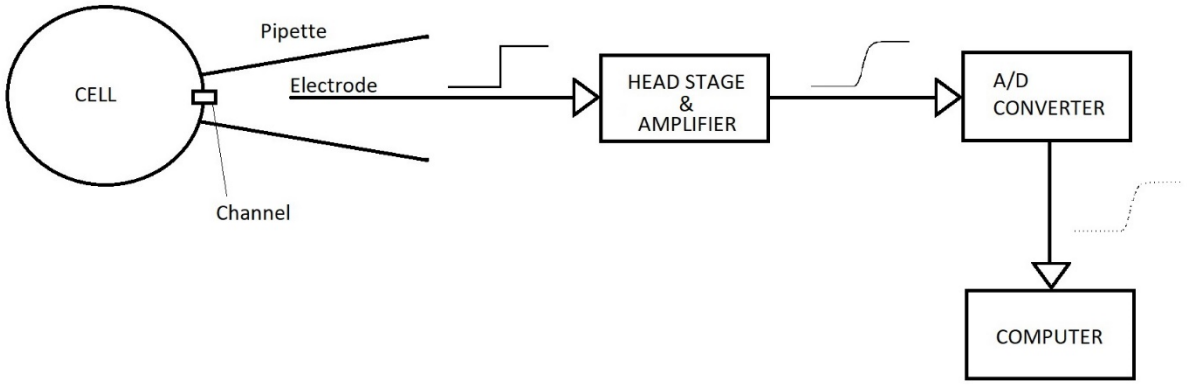

Figure 3. Schematic of the cell-attached single-channel patch-clamp. A pipette has formed a GΩ seal on a patch of cell containing a single channel. An electrode detects single-channel activity as step changes in current amplitude. The head stage responds to the step change in amplitude and produces a step response. The analog to digital converter samples the analog signal into discrete current values which are then stored on a computer.

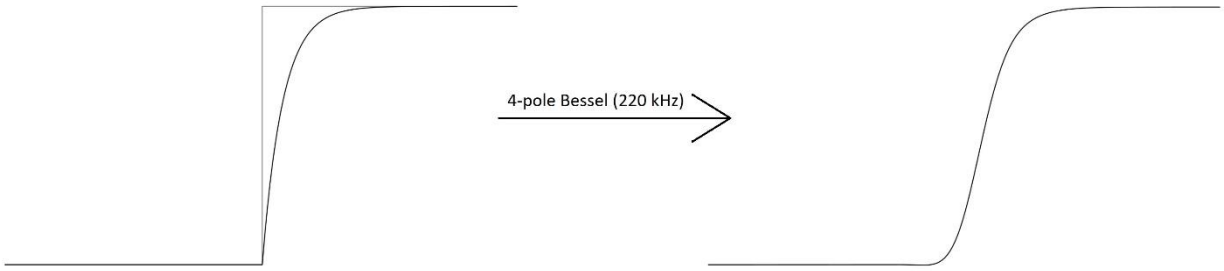

Figure 4. Components of the patch-clamp step response. On the left a square step with an RC component overlaid. On the right, the step-response has been filtered with a digital approximation to an analog 4-pole Bessel filter. The RC time constant is 1.69 µs, the Bessel cut-off frequency is 220 kHz.

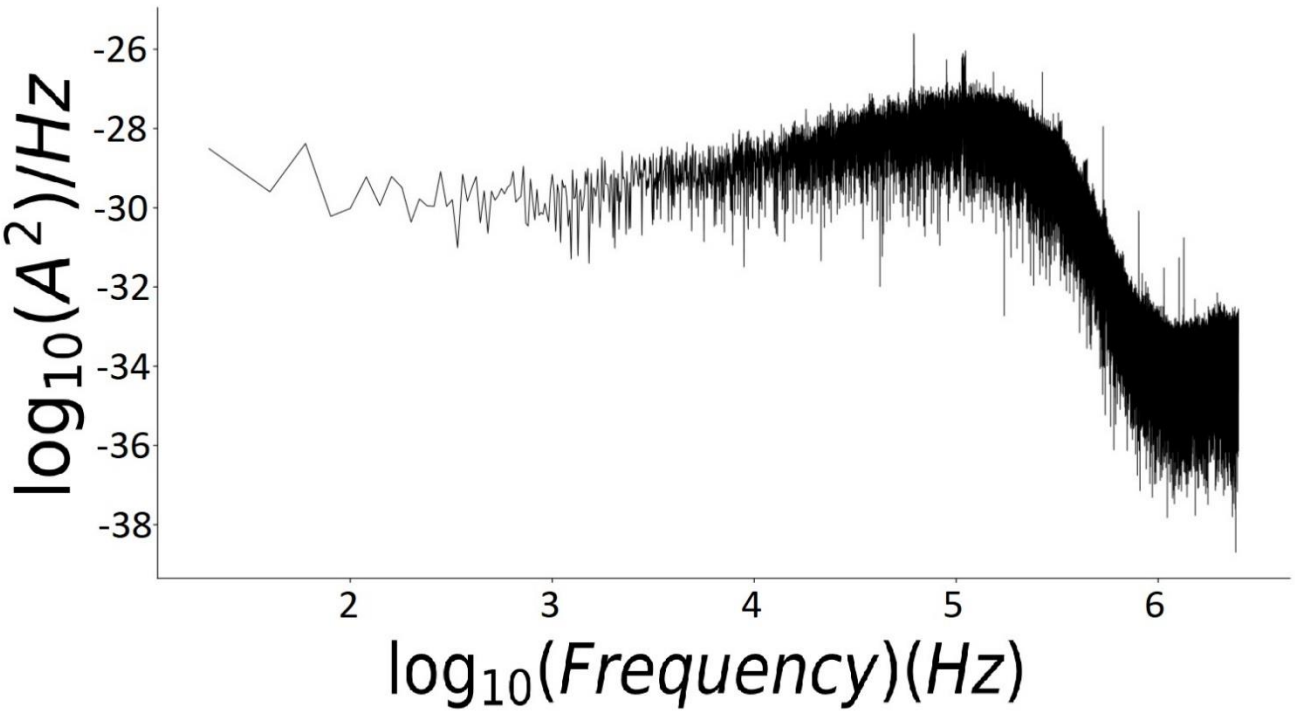

Figure 5. Power spectrum of real patch-clamp noise. An example of a power spectrum of real patch-clamp noise recorded with a sampling interval of 0.2 µs. The f^2 characteristic appears between 1 and 100 kHz. The 80 dB/decade attenuation effect of the analog Bessel filter appears between 100 and 1000 kHz. The x-axis display $\log_{10}$ frequency in Hz. The y-axis display $\log_{10}$ squared Amperes per Hz (scaled Decibels). Details about calculating the power spectrum can be found in section 2.1.2.6.

1.1.4 Single-channel data

Single-channel data typically display two conductance levels: an open channel level with current amplitudes in the 1-20 pA (10^{-12}) range, and a closed channel baseline level ideally near zero pA. Some channels appear to have subconductance states; a state of conductance where the channel is open, yet its current amplitude is below its maximum amplitude[42,43].

Due to the small currents that flow through ion channels, their signal is often swamped by background noise. A common practice to overcome this limitation is to increase the signal to noise ratio (S/N). There are many ways to define signal to noise ratio depending on the type of signal. One definition, which is appropriate for single-channel data where the mean open channel current is expected to be constant, is the ratio of the mean open channel current (μ) to the standard deviation (σ) of the baseline current (μ/σ).

The most common way to increase S/N is to filter the data, either through an analog or digital low-pass filter. Analog low-pass filters use physical circuitry to process a signal. Digital low-pass filters use computational operations to process a signal. Both analog and digital low-pass filters have the same effect of averaging the values of the signal by attenuating high-frequency fluctuations in the signal (Figure 6).

The physical transitions between open and closed states of a single channel are effectively modeled as instantaneous step changes in current amplitude. When the single channel opens, the rate of ion flow through the pore goes from zero to its maximum rate (~10^7 ions per second) which is determined by the membrane potential, and the resistance within the pore of the channel. The increase of the flow rate (which is the current,) between zero and its maximum value happens so quickly that it can be modeled as an instantaneous step change in current. The time scale for the conformational change of the channel is on the order of 10-100 ps, which is much faster than the interval in which we sample our data (~1 μs).

When a step change in current is detected by the head-stage amplifier, the system responds according to its step-response. The time it takes for the system to output a signal reaching full amplitude is related to the time constant τ. The time it takes for the signal to reach full amplitude is called the rise time (formally defined as the time it takes to go from 10 to 90% of the maximum amplitude.)

If several transitions between open and closed states occur in rapid succession, then the resulting input signal to the head-stage amplifier will be a succession of step-changes in current amplitude. If the output signal for one step does not reach full amplitude before the next step occurs, then the output signal will change directions to reflect the next step. The resulting output signal is attenuated, having reached a lower than maximum amplitude (Figure 6).

Since the output signal of the amplifier carries noise, signals that are attenuated by the rise-time of the step-response can be “buried” in the noise. This causes ambiguity when attempting to recover signals relating to ion channel activity.

Filtering the data not only improves signal to noise, but has the adverse effect of lengthening the rise-time of the step-response (Figure 7). This lengthening of the rise-time causes further attenuation of step signals. Thus, filtering to recover signals from noise should be done carefully. It is akin to the archaeologist who mindfully brushes away dirt, taking care not to damage the fossil underneath. Filtering should always be done to maximize the amount of accurate information one can reliably extract from the data. Choosing the right filter setting is not trivial, and only with simulated data, where openings and closings are known *a priori*, is it possible to optimize the filter setting.

Additionally, analog output signals are digitally sampled by a computer, yielding discrete current values separated by a fixed interval of time (a process known as digitization of an analog signal). This interval of time is the sampling interval, and is the inverse of the sampling rate. For example, if the sampling interval is 1 μs, then the sample rate is 1/1 μs or 1000 kHz.

To summarize, single-channel data are a series of discrete current values in the pA range. Single-channel data typically display two current levels. Transitions between open and closed states are recorded over a stretch of time. The random succession of transitions leads to a trace of data containing a variety of signals from channel activity; from fully resolved channel openings, to heavily attenuated peaks.

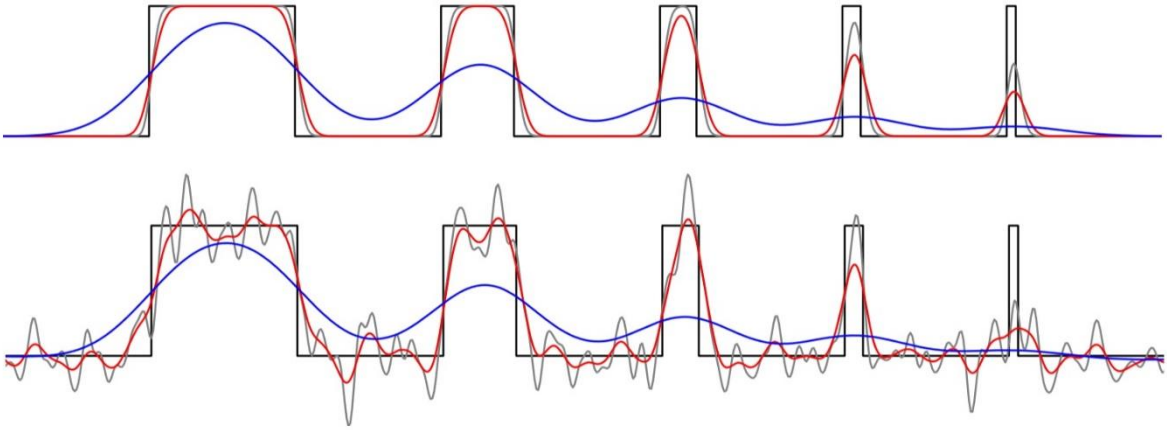

Figure 6. Example rectangular pulses with simulated response. The top trace shows a series of simulated rectangular pulses separated by 80 µs intervals (80, 40, 20, 10, and 5 µs from left to right.) The step-responses of the series are simulated with a 1 µs sampling interval, an RC constant of 1.64 µs, and a Bessel filter cut-off frequency of 200 kHz. The simulated data are digitally filtered with a Gaussian filter with cut-off frequencies of 50 kHz (gray trace), 25 kHz (red trace), and 5 kHz (blue trace). From left to right, shortening of the rectangular pulse corresponds to increasing attenuation of the step-response. The bottom trace is the same as the top trace, but with added noise.

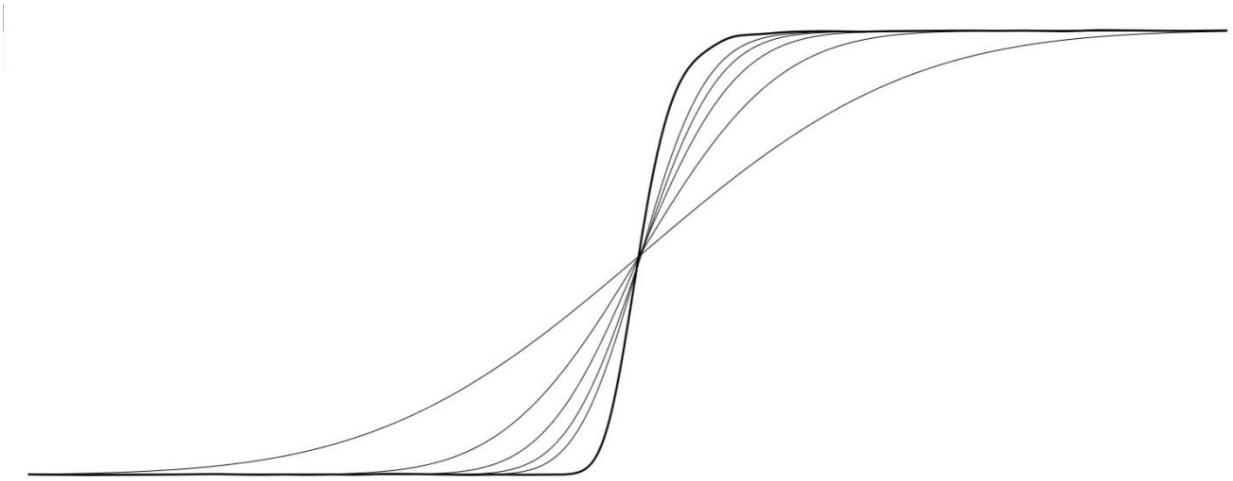

Figure 7. Filtered step-response. The thicker line shows a real measured step-response from the patch-clamp. The thinner lines show the step-response digitally Gaussian filtered with increasing cut-off frequencies. As the cut-off frequency increases, the step-response rise time lengthens. The x and y axes were omitted for visual purposes. The x-axis is time, and the y-axis represents current amplitude.

1.1.5 Single-channel kinetics

The study of single-channel kinetics aims to understand the ion channel's gating activity by modeling the activity as transitions between discrete states. Kinetic models for single channel activity are like most kinetic models for chemical reactions, in that they include states connected by reversible transitions (Figure 8).

Since we are dealing with the behavior of a single channel, the rate constant can be converted to a probability of undergoing a particular transition. The interpretation of the rate constant as a probability is discussed in detail elsewhere[61]. Single-channel transitions are random variables. That is, a single ion channel will remain in one state for a random period of time before making a transition to another state. Thus, the rate constant is interpreted in a probabilistic way.

However, a single rate constant does not by itself describe the kinetic behavior of a channel. When describing the kinetic behavior of the channel, all rate constants in the model are considered simultaneously. Put briefly, all of the rate constants for transitions going into a particular state describe how frequently this state is visited by the channel. Additionally, all of the rate constants for transitions leading away from one state describe how long the channel will stay in that state (referred to as the lifetime of the state) Lastly, the rate constants in and out of each state must be considered globally with the rate constants in and out of each other state to fully describe the steady-state kinetic behavior of the channel.

The lifetime of a state is described by an exponentially distributed probability density function (pdf) with a mean value equal to the reciprocal of the sum of the rates leaving that state. (Equation 2) The area under the pdf up to a certain time *t* is the probability that the lifetime of that state will be equal to or less than *t*.

$$f(t) = \lambda e^{-\lambda t} \qquad [2]$$

Where f(t) is the probability density of the state's lifetime *t* with mean 1/λ (λ = the sum of the rates leaving the state.)

For the activity of single-channels, if the channel is in steady-state, it can be modeled as a time-homogeneous discrete Markov process with the following properties:

1. The probability of making a transition does not change over time.
2. The probability of making a transition depends only on the current state of the channel.
3. Transitions between states are instantaneous and discrete.

As kinetic models for ion channels can contain many states with connectivity defined by their reversible transition rates, a convenient notation for these models is in matrix form (Figure 8). The matrix notation is also a common form for analyzing discrete Markov processes. Matrix notation has several useful properties:

1. Looking across a row in the matrix, you can identify all of the rates for leaving a state.
2. Looking down a column in the matrix, you can identify all of the rates for entering a state.
3. The diagonal elements are the negative of the sum of all of the rates in row.

The aforementioned features of matrix notation are not only visually appealing, but also make calculating probability densities easy with simple matrix algebra operations. Details for calculating probability densities are well-established and found elsewhere[62]. The transition rate matrix is denoted by Q, while the probability matrix is denoted by P.

There are two types of probability densities that are important for single-channel kinetics. For a given model, one can calculate the state probability distribution (SPD), and the conductance probability distribution (CPD) (Figure 9). These distributions are mixtures of exponentials (defined by the sum of weighted exponential terms, Equation 3)

$$f(t) = \sum_{i=1}^{N} a_i \lambda_i e^{-\lambda_i t} \quad [3]$$

Where *f(t)* is the probability density of the state's lifetime *t* with mean 1/λ (for a number of states N). Each term is weighted according to *a* so that the total area under the summed f(t) is equal to 1.

The SPD describes the lifetime of each individual state, regardless of its conductance level. It is the scaled sum of the individual pdf for each state in the model.

The CPD describes the lifetime of a state, or an aggregate of connected sates having the same conductance level. As opposed to the SPD, the CPD is calculated using sub-matrices derived from the transition rate matrix[62]. Moreover, since single-channel data contains only information about conductance, the CPD reflects the observed event duration histogram (where open or closed events are binned according to duration.)

These probability distributions, however, in their simple exponential form are not very informative. Performing a log transformation of the x-axis, combined with a square root transformation of the y-axis of these distributions makes them visually more informative, and at the same time facilitates fitting the distribution with its underlying individual exponential components[44] (Equation 4).

$$g(t) = \sum_{i=1}^{N} a_i exp(ln(\lambda_i) + ln(t) - exp(ln(\lambda_i) + ln(t))) \quad [4]$$

Where the symbols of Equation 4 are defined as in Equation 3, except *e* has been replaced by exp() for clarity of notation.

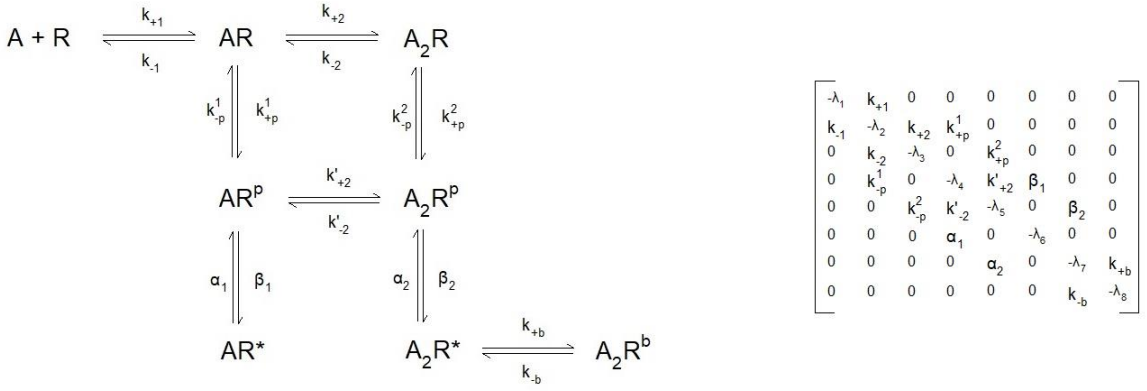

Figure 8. A kinetic scheme and its matrix notation. On the left, a kinetic scheme for the nAChR (SPM[1]). On the right, the matrix notation for the kinetic scheme on the left.

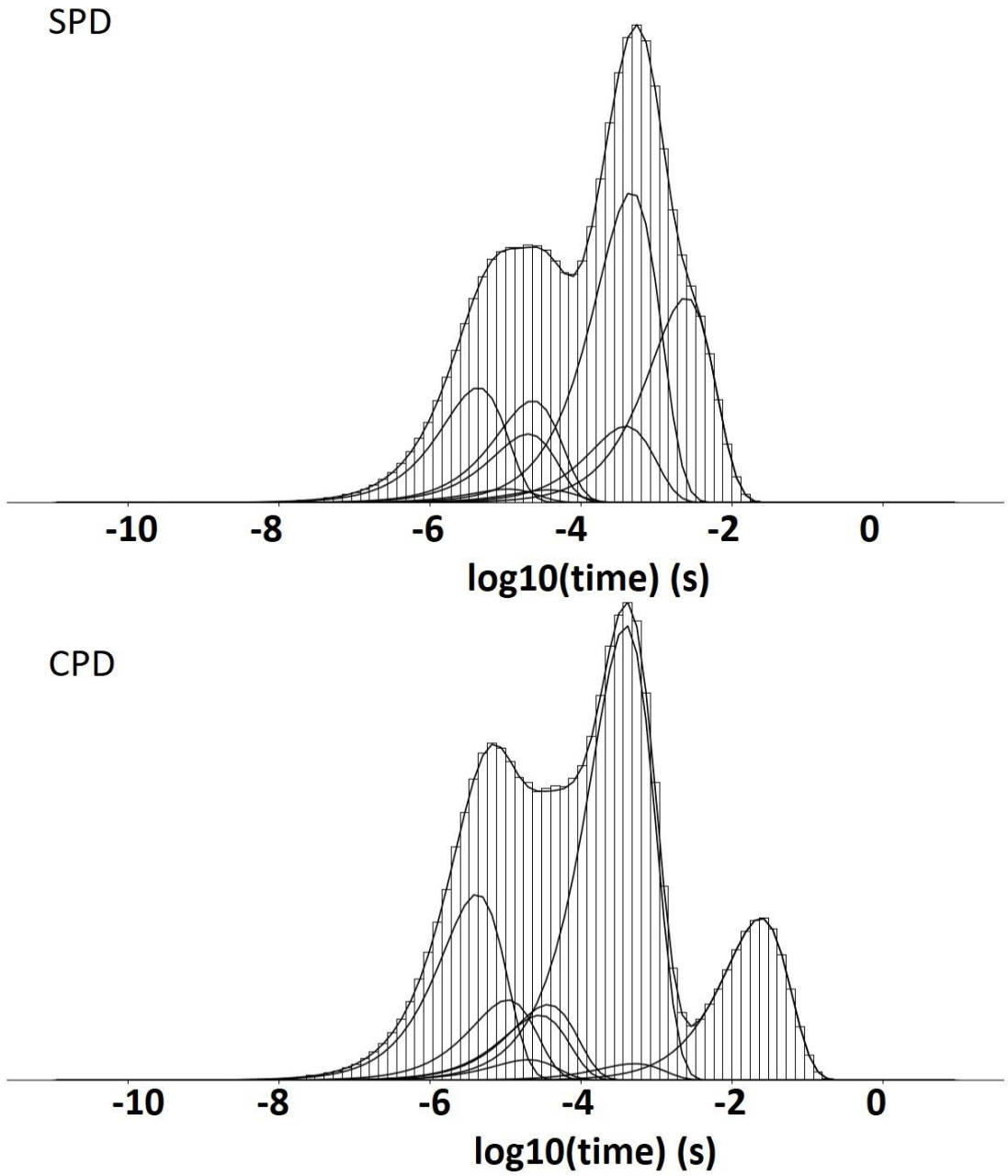

Figure 9. Example SPD and CPD. The top diagram displays the summed SPD (edge line), the individual components of the SPD, and a corresponding duration histogram of simulated events. The bottom diagram displays the summed CPD (edge line), the individual components of the CPD, and a corresponding duration histogram of simulated dwells. The x-axis shows $\log_{10}$ units of second. The y-axis has been omitted for visual purposes, but typically represents event count or proportion of events for a particular component. For details on calculation of SPD and CPD see sections 2.1.1.4 and 2.1.1.5 respectively. The SPD and CPD here correspond to the SPM[1] at 3 µM ACh.

1.1.6 Single-channel data analysis

Once raw data is acquired, there are many ways to analyze it in order to extract different kinds of information from the data. For my thesis we will consider an application of single-channel analysis to extract kinetic parameters (i.e. rate constants) from single-channel data.

The first step in analysis is to idealize the data. Idealization of the data involves recovering the exact durations and current amplitudes from the events of the ion channel's activity. An event has two defining characteristics, the duration of the event, and the amplitude of the event. In a data record, looking along the axis of time, an event is a stretch of time for which the current amplitude remains the same. For a single channel with one open channel amplitude, a step change in current to the open state must (1) be preceded by a closed state, and (2) eventually be followed by a step change in current back to the closed state. Thus, idealization of single channel data with a single open amplitude is the process of recovering the rectangular pulses of channel activity from noisy filtered data.

Idealization of single channel data is not a trivial process. As described in the previous section, single channel data is noisy, and contains many peaks with attenuated amplitudes. Additionally, even for signals having reached maximum amplitude, recovering the actual duration of the event can be difficult due to noise as well as filtering, and sampling effects.

Currently, idealization of the data is a built-in property of event detection methods. The aim of an event detection method is to detect the occurrence of an event in a noisy trace of data. It makes sense that the detection method has some inherent way to idealize the data. However, if the detection method has a known systematic error in idealization, then it is possible to do a post-detection correction of event durations[47,48,49].

Event detection methods exist for many disciplines requiring signal analysis. Here we consider three event detection methods which have been deemed appropriate for single-channel data: (1) Threshold crossing[50], (2) Segmental k-means with hidden Markov modeling[51], and (3) Time course fitting[33] (Figure 10).

Threshold crossing is conceptually the simplest of the three methods. A transition occurs when the current amplitude crosses a specified threshold of the maximum open channel amplitude. In single-channel data, a threshold of 50% is typically used, in order to ensure equal detection of both

openings and closings. An event is defined by two transitions, therefore the data must cross the 50% threshold twice for an open or closed event to be detected. Some algorithms for threshold crossing interpolate the threshold crossing point between sample points, and thus durations of events are not an increment of the sampling interval.

The segmental k-means with hidden Markov modeling (SKM-HMM) is considerably more complicated than threshold crossing. This method takes an iterative statistical approach to detecting events by assigning a conductance level to each sample point in the data. Until a maximum likelihood criterion is met, the method will iterate between two steps: (1) idealization of the data by maximizing the *a posteriori* probability of the state sequence, and (2) re-estimation of model parameters.

Time course fitting of the data considers the shape of step-response with respect to some degree of filtering. For example, if the data is filtered with a digital Gaussian filter, a time course fitting algorithm will attempt to fit possible transitions with the filtered step-response in order to recover the square step change in current when the transition occurred.

Once the data is detected and idealized, the next step is to fit the parameters of a kinetic model to the observed data. Kinetic parameters are extracted by fitting the observed sequence of events[46].

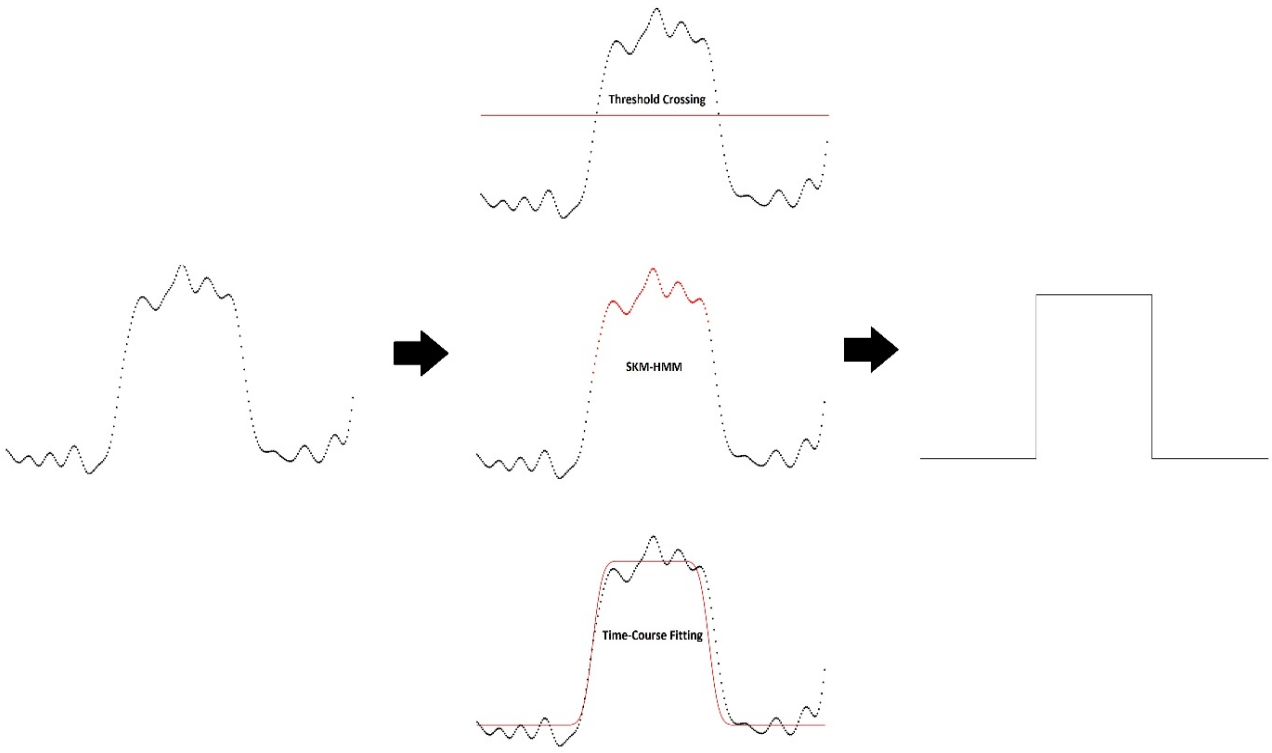

Figure 10. Depiction of event detection methods. On the left, a sampled noisy event. On the right, an ideal rectangular pulse. In the center, a depiction of three event detection methods. On the top, Threshold Crossing detects and idealizes the event using a 50 % amplitude threshold (red line). In the middle, SKM-HMM detects and idealizes the event by assigning to each sample point a conductance level (red and black dots.) On the bottom, Time-Course Fitting detects and idealizes the event by fitting a Gaussian-filtered square-pulse.

1.2 GOAL OF THE PROJECT

1.2.1 The problem of brief events in single-channel kinetics

Problems in single-channel kinetics ultimately stem from a limited temporal resolution of the data. This temporal resolution, aptly named the dead-time, is a limit on the briefest event which can be detected. Although the dead-time is a universal feature of single-channel experiments, its definition and causes are dependent on the physical set-up as well as any post-processing and analysis of the data.

The dead-time is not just a problem for brief events. The dead-time affects all events. For example, a brief opening that is missed by a detection algorithm will cause two closed events to appear as one longer closed event (Figure 11). Although the term dead-time brings our attention to the problem of brief events, an improvement in the dead-time will have a positive impact on the quality of the data as a whole.

Problems in single-channel kinetics in general are centered around misinformation (i.e. inaccurate, false, or missing information.) Misinformation can be introduced anywhere in the process from gathering data to extracting kinetic parameters from the data.

Ultimately, misinformation stems from the physical conditions of the patch-clamp set-up. The amplifier in the head-stage directly imposes a limit on the temporal resolution on the data because of the rise-time of the step response. Moreover, anything that contributes to a lower S/N will have a negative impact on the temporal resolution of the data (because the data will require more filtering). Factors which affect S/N include the conductance of the channel, the type of cell on which the channel is expressed, and the quality of the seal between the pipette and the cell patch.

In detection and idealization, events can be missed due to filtering, or falsely introduced by excess noise (Figure 11). Additionally, events can have inaccurately recovered duration. In Threshold detection, however, a systematic error in the accuracy of brief events can be corrected[50].

Missed, false, and inaccurate events undoubtedly lead to inaccuracy of the inferred kinetic parameter. Despite missing information, a lot of work has been done to develop methods to account for missed events in kinetic fitting[46,53,54,55]. However, little work has been done to address the effect of

inaccurate events on kinetic fitting. To address the effect of inaccurate events on kinetic fitting would require a realistic simulator that provides precise *a priori* information.

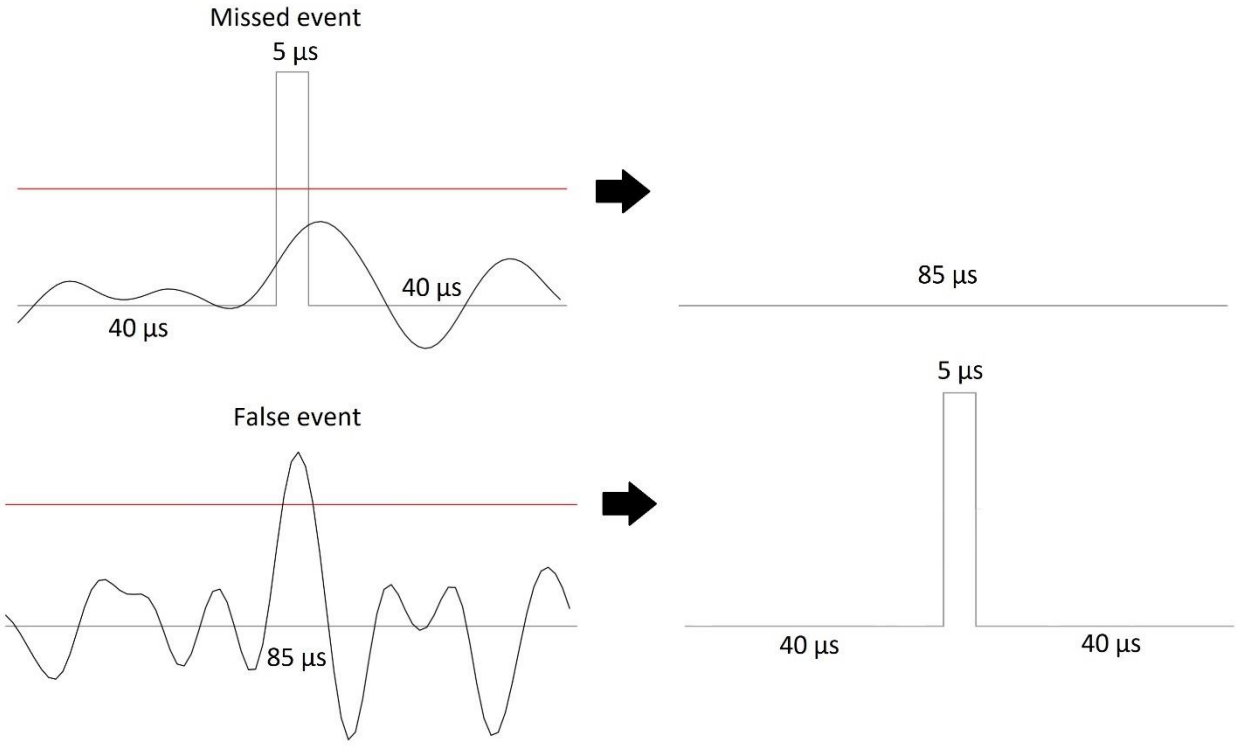

Figure 11. Illustration of a missed and a false event. On the top left, the underlying 5 µs event (gray rectangular pulse,) has been attenuated by the rise-time of the step-response. The noisy event (black trace) does not reach the 50 % amplitude threshold (red line), and hence is missed. The top right shows the result of missing an event; a recovered closed event with a longer duration of 85 µs. The bottom left shows that there is no underlying rectangular pulse. However, noise (black trace) crosses the 50 % amplitude threshold (red line) to produce a false event. The bottom right shows the result of detecting a false event; a falsely introduced event causes an 85 µs closing to appear as two 40 µs closings flanking a 5 µs opening.

1.2.2 A brief history of single-channel kinetics

The history of single-channel kinetics shows us the impact of methodological improvements on our ability to resolve complicated kinetic models (Figure 12). In this section we consider a brief history in the development of the muscle-type nAChR model. For brevity, we do not mention every model invoked throughout the history of the muscle-type nAChR model development. We mention only some of the models, highlighting important improvements in model resolution.

Before the existence of single-channel techniques, in 1957, J. del Castillo and B. Katz proposed one of the first kinetic models for the muscle-type nAChR, which did not include rate constants[28]. Following this, the proposal of an extended model by A. Karlin in 1967, which contains an extra binding step and an extra open state[59]. Measuring rates was attempted by D. Colquhoun and B. Sakmann in 1985, using single-channel data[58]. A final refinement of the rates for this extended model was made in 2004 by W. Lee and S. Sine using improved methods for fitting sequence of events[29,57].

More recently a new extension of the model was proposed by Mukhtasimova *et al.* in 2016 containing a primed state, between binding and gating[1]. With improved physical conditions, the conductance of the channel was increased, and the noise was decreased. This raw improvement in the S/N allowed for Mukhtasimova *et al*. to increase their digital cut-off frequency from 10 kHz to 25 kHz. Increasing the cut-off decreased the rise-time of the step response, and hence improved the dead-time (from 22 μs to 8 μs).

To summarize, methodological improvements have increased our ability to resolve more states, but also with more accurate rate constants. The resolution of states is important because it helps us better understand the function of the protein. For example, single-channel kinetics' unique ability to differentiate between binding and gating sheds light on the general issue of differentiating affinity (binding equilibrium constants) from efficacy (gating equilibrium constants)[52]. Moreover, resolving more states makes it possible for us to understand how mutations alter the different aspects of ion channel function.

Despite all that has been done, work remains to improve both the physical conditions of single-channel experiments, and the analysis of single-channel records. Single-channel data is precious as it contains a treasure trove of information. We need to make sure that we implement the best practices in order to reliably extract as much information as possible from the data.

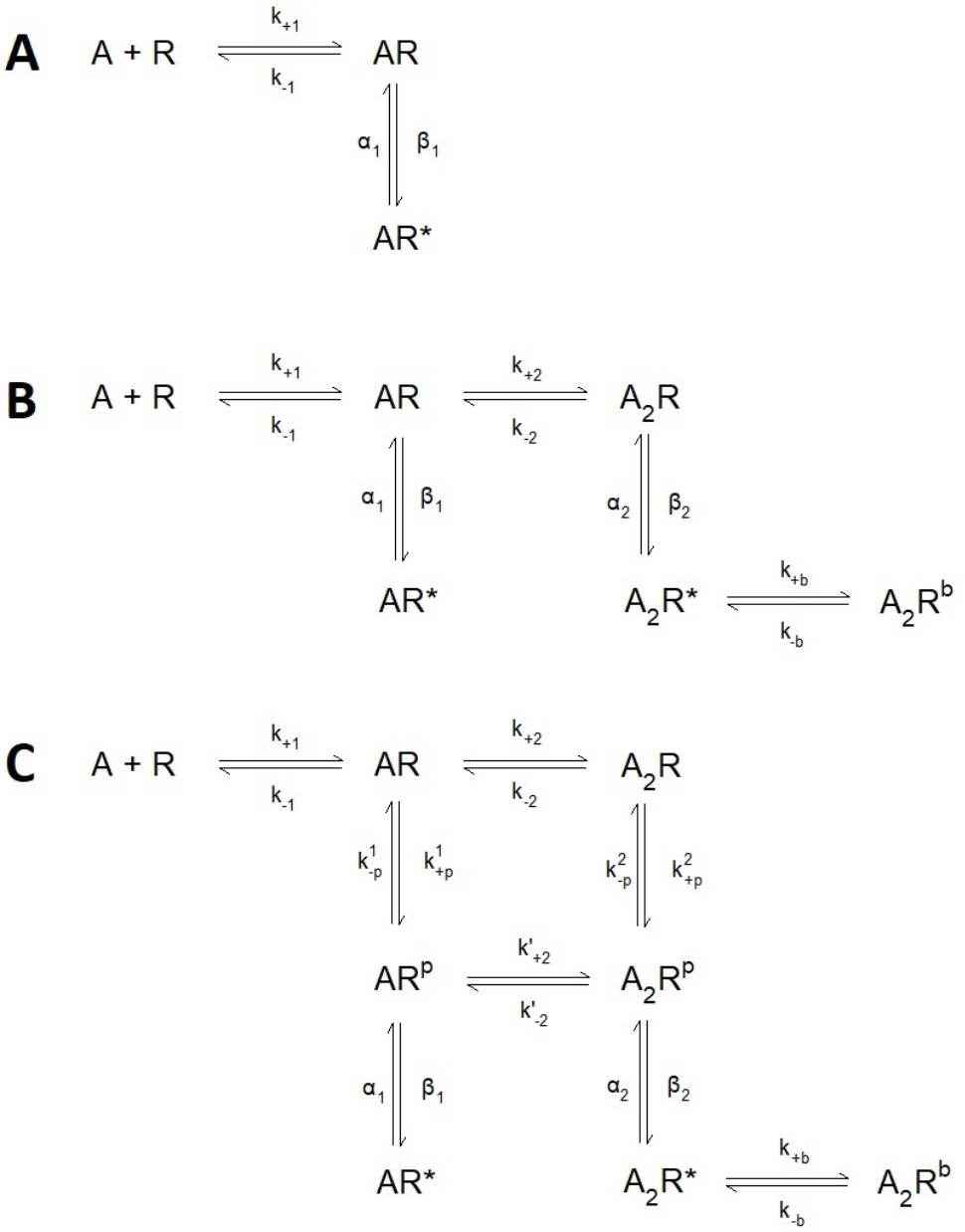

Figure 12. Historical evolution of the nAChr kinetic scheme. (A) The original del Castillo & Katz model from 1957[28]. (B) The extended del Castillo & Katz model with block from 2004[29]. (C) The standard prime model (SPM) from 2016 (based on the SPM from 2009)[1].

1.2.3 Our approach to address the problem of brief events in single-channel kinetics

There are essentially two sides to the problem of brief events in single-channel kinetics: the physical conditions of the experiment, and the analysis of the data. For the research presented here, our approach to addressing the problem is by improving methods of analysis. The short reason is that technological improvements are hard to come by. Perhaps a more motivating reason is the fact that improvements in single-channel data analysis alone have contributed to resolving more complicated kinetic models[50,53,54,55,57].

More rigorous practices in data analysis remain to be developed. Despite the variety of methods that exist for single-channel data analysis, the choice of which methods to use is not obvious. Quantitative comparisons of existing methods are lacking. Moreover, procedures involving a given set of methods remain sub-optimal, not properly tested or compared.

Given the lack of quantified comparisons and optimization in the field of single-channel data analysis, we find motivation to establish grounds for making such comparisons and optimizations of methods. The tacit assumption is that these improvements in methodology will lead to greater understanding of the underlying biology, through illuminating mechanisms of ion channel function.

1.2.4 Designing a simulator

Our approach to improve methods of single-channel data analysis has led us to the development of a modular, realistic single-channel data simulator using the programming language Python. The goal of the simulator is to provide grounds for method comparison and optimization by providing realistic data with corresponding *a priori* information.

Recent improvements in model resolution bring to the forefront of single-channel kinetics the importance of resolving and quantifying short-lived states in ion channels[1]. Being able to resolve and quantify short-lived states requires a greater resolution of brief events in single-channel data. Hence, it is important for the analytic method of choice to be able to perform well when faced with a data set containing the briefest of single-channel events. A rigorous quantitative characterization of methods in single-channel data analysis does not exist. The question of which methods perform best when faced with brief events, as in state-of-the-art mechanistic schemes of nAChR function is unanswered.

Our motivation for designing a simulator is that a realistic data set containing precise *a priori* information is necessary to quantify and compare methods in single-channel data analysis. Simulators do exist in current data analysis software, for example in the software TAC or QUB. However, existing simulators are either imprecise, having limited time resolution, or are simply missing *a priori* information all-together. QUB, for example, will simulate data where transitions always occur on a sample point, when realistically transitions can happen at any time between sample points. In fact, the probability that a transition occurs precisely on any sample point is vanishingly small. In TAC, data are simulated, but the *a priori* exact transition times are missing from the simulation. As a result, data simulated from TAC need to be analyzed in order to recover the instantaneous transitions, which introduces a detection-dependent bias to any data simulated by TAC.

Realism is an important factor in the design of our simulator. The more alike simulated data is to real data, the more likely comparison and quantification of methods are to be valid. Existing simulators, to some degree, lack important elements of realism. For example, QUB simulates data with white noise, yet it has been shown that noise from a patch-clamp is not white. The frequency characteristics of noise may seem like a trivial factor, but some analytic methods in single-channel analysis depend on the characteristics of the noise.[60] The traditional forward-backward Baum-Welch algorithm, which is used to find the parameters of an HMM, assumes that background noise is white,

and has been shown to perform poorly when noise is not white[60]. Thus, simulated data with white noise can introduce a favorable bias for some methods. In TAC, data are simulated with a Gaussian-filtered square pulse step-response. Realistically, however, the step-response of the patch-clamp also has an RC component, and a shape influenced by the analog Bessel filter.

To reduce the introduction of bias in simulated data, we designed a simulator with two separate engines: (1) An HMM engine, which generates an array of high time-resolution (resolution limited by computer memory) kinetic events from a kinetic model. (2) A sampling engine, which takes a sequence of events, and yields a data trace containing sampled data points, realistic patch-clamp noise, and a realistic patch-clamp step-response (Figure 13).

The design of the simulator is modular, allowing for a wide range of functionality for designing complicated, and realistic data sets. A few important features of modular functionality are listed below:

1. Step-response matching (for a variety of step-responses)
2. Noise matching (for a various kinds of noise)
3. Multiple channels (more than one ion channel in a patch)
4. Multiple models (different kinds of ion channels in one patch)
5. Sub-conductance levels (amenable to nanopore experiments, or ion channels with sub-conductance states)

It is important that the step-response and noise of the simulator be variable, since different set-ups can have different system properties.

As mentioned, the main applications of the simulator are for method comparison and optimization. However, the simulator can be used for hypothesis testing, testing newly developed methods, and validating new kinetic models.

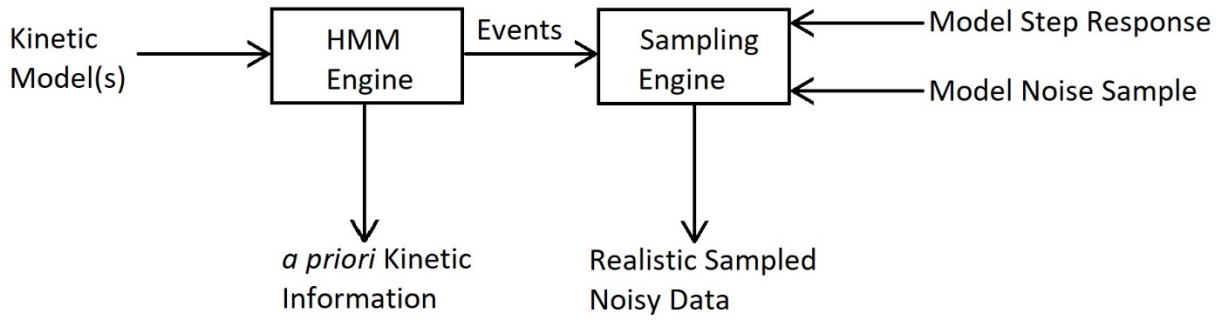

Figure 13. Schematic of the simulator. The boxes represent the engines of the simulator. The HMM engine takes as input one or more kinetic models, and outputs simulated events and a simulation file which contains the *a priori* information from the simulation. The Sampling engine takes as input events from the HMM engine (or from elsewhere,) a model step response, and a model sample of baseline noise. The Sampling engine then outputs realistic sampled noisy data.

1.2.5 Preliminary comparison of detection methods

To illustrate the utility of our simulator, we have designed a preliminary comparison of two widely used event detection methods in single-channel data analysis. Our comparison is between Threshold Crossing and SKM-HMM. Both methods have also been used extensively for detecting events that are subsequently used for kinetic fitting to mechanistic schemes. Originally, we had planned on comparing the three most widely used methods, which includes Time-Course Fitting. However, due to practical limitations we have left the comparison including Time-Course Fitting to future work.

For Threshold Crossing, we chose TAC as the software application for this method. TAC not only detects events via a specified threshold amplitude, but also idealizes the duration of events using a cubic-spline interpolation of values between data points. As a result, idealized events will have a time resolution which is not limited by the sampling interval.

For SKM-HMM, we chose the widely used QUB as it is the sole application for this method. As previously mentioned, this method uses an iterative statistical approach to assign an ideal conductance value to each sample point. Thus, idealized events will have a time-resolution equal to the sampling interval.

Threshold Crossing and SKM-HMM take two very different approaches to detecting and idealizing events.

To compare Threshold Crossing and SKM-HMM, we have designed and simulated a test data set. This data set includes the latest improved kinetic model for the muscle-type nAChR; the Standard Prime Model (SPM)[1]. Moreover, to ensure that a wide, realistic range of kinetic events are included, we have simulated with a concentration range spanning 3 to 300 µM of the agonist acetylcholine.

Additionally, we seek to determine which method performs best according to a range of S/N ratios. Thus, we have simulated with a range of digital Gaussian filter cut-off frequencies from the lowest extreme at 5 kHz to the highest extreme at 50 kHz.

To ensure that the sampled noisy data are as real as possible, we have simulated with a step-response, and with a stretch of baseline noise measured from the Axopatch^{TM} 200B. We have chosen

the sampling interval of 1 μs, as it is the current standard. A sampling interval of 1 μs is also convenient for computer storage of these large data sets.

To be considered for kinetic analysis, a typical single-channel record will contain roughly 10 000 events (5000 closings, 5000 openings.) For each concentration in our range, we have simulated 100 000 events (10 times more events than in real data.)

To assess the performance of each method, we have defined three characteristics: (1) missed events, (2) false events, and (3) accuracy of recovered duration. We can easily compare each method based on percentage of missed events and percentage of false events.

An additional, but indirect measure of performance is to test how each method performs when submitting their idealized events to kinetic fitting. Since we know the exact rate constants of the kinetic model that was used to simulate the data set, we can indirectly test the performance of the detection method by how well an established kinetic fitting algorithm can recover the true rate constants. The idea is that if the detection method misses too many events, introduces too many false events, or poorly recovers event durations, then these errors should be reflected in the parameters derived from kinetic fitting.

2 METHODS

2.1 SIMULATOR

2.1.1 HMM ENGINE

2.1.1.1 Randomness

To model randomness we use a standard pseudo-random number generator (PRNG) called the Mersenne Twister[64] which is implemented in the Python-based library Numpy[65]. Choosing is modeled as random sampling with replacement from pre-calculated sampled distributions.

2.1.1.2 Model files

Model files are text files which contain the transition rate matrix Q and the conductance levels of the states.

Example of the format of the model file:

<%s3uM_model%>
{States:0,0,0,0,0,1,1,0}
{TRM:-405,405,0,0,0,0,0,0}
{TRM:1410,-2137,402,325,0,0,0,0}
{TRM:0,25000,-43400,0,18400,0,0,0}
{TRM:0,34100,0,-49750,6450,9200,0,0}
{TRM:0,0,86400,17200,-228600,0,125000,0}
{TRM:0,0,0,27300,0,-27300,0,0}
{TRM:0,0,0,0,2100,0,-2550,450}
{TRM:0,0,0,0,0,0,93000,-93000}
<%%>

2.1.1.3 Probability matrix

The probability matrix P is derived from the transition rate matrix Q. The rows of the P matrix contain the probabilities of making transitions to other states. Diagonal elements are set to zero (the probability of staying in the same state is fixed to zero.) For each row in the matrix, the row elements are divided by the sum of the row elements. Dividing the row elements by the sum of the row elements converts transition rates into probability values between 0 and 1. Example below:

STATE #	1	2	3	4	5	6	7	8
1	-405	405	0	0	0	0	0	0
2	1410	-2137	0	0	402	325	0	0
3	0	0	-2550	0	0	0	450	2100
4	0	0	0	-27300	0	27300	0	0
5	0	25000	0	0	-43400	0	0	18400
6	0	34100	0	9200	0	-45450	0	2150
7	0	0	93000	0	0	0	-93000	0
8	0	0	125000	0	86400	17200	0	-228600

Table 1. Transition rate matrix (Q) of the SPM at 3 µM [ACh]. The elements of the matrix are rate constants in s^{-1}. The row elements indicate rate constants of transition from the indicated state # of the row to the indicated state # of the column (e.g. the rate of going from state 1 to state 2 is 405 s^{-1}.) The diagonal elements indicate the rate of staying in a particular state.

STATE #	1	2	3	4	5	6	7	8
1	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.6598	0.0000	0.0000	0.0000	0.1881	0.1521	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1765	0.8235
4	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
5	0.0000	0.5760	0.0000	0.0000	0.0000	0.0000	0.0000	0.4240
6	0.0000	0.7503	0.0000	0.2024	0.0000	0.0000	0.0000	0.0473
7	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.0000	0.0000	0.5468	0.0000	0.3780	0.0752	0.0000	0.0000

Table 2. Probability matrix (P) of the SPM at 3 µM [ACh]. The elements of the matrix are transition probabilities as values between 0 and 1. The row elements indicate the probability of transition from the indicated state # of the row to the indicated state # of the column (e.g. the probability to transition from state 1 to state 2 is 1.) The diagonal elements are set to zero, because the simulation algorithm removes the possibility of staying in the same state.

2.1.1.4 State probability distribution (SPD)

State probability distributions are calculated according to Equation 4 from section 1.5:

$$g(t) = \sum_{i=1}^{N} a_i \exp(\ln(\lambda_i) + \ln(t) - \exp(\ln(\lambda_i) + \ln(t))) \quad [4]$$

Where g(t) is the sum of log-transformed exponential decay terms. Each term describes the probability distribution of the lifetime of a state in the kinetic model. Each term has a mean equal to $1/\lambda_i$, where λ_i is the absolute value of the diagonal element in the Q matrix. Each term is weighted according to a_i so that the relative size of each term reflects the steady-state occupancies of each state. The values on the x-axis are $\log_{10}$(seconds) and represent the lifetime of a particular residence in that state. The values on the y-axis represent the proportion of time spent in a particular state. The values on the y-axis can be scaled such that individual [x, y] pairs represent [lifetime, probability] respectively. The importance of the y-axis is that each term is scaled such that they are proportional to the steady-state occupancy of states.

Steady-state occupancy of states are solved directly using the relationship:

$$p(\infty)Q = 0 \quad [5]$$

Where p(∞) is a row matrix whose elements are the steady-state occupancies (values between 0 and 1), and Q is the transition rate matrix. This method is explained in detail elsewhere[62]. Briefly, Equation 5 is derived from a relationship which follows from the Law of Mass Action. The rate of change of the channel's occupancy of states at steady-state is equal to zero. To solve Equation 5 for p(∞), an matrix S is formed by appending a column of ones to the Q matrix. Thus for a Q matrix whose size is *k* x *k,* S will be a *k* x (*k*+1) matrix. A *k* x 1 row vector **u** is defined containing only ones. The solution of p(∞) is given by:

$$p(\infty) = u(SS^T)^{-1} \quad [6]$$

Where S^T is the transpose of S, and $(SS^T)^{-1}$ is the inverse of the matrix multiplication of S by S^T.

The weights a_i are calculated as follows for a *k* x *k* matrix Q:

1. Diagonal elements of Q are set to zero.
2. Q^p is defined as p(∞)Q.
3. A *k* x 1 row vector **a** is defined such that an element *k* is the sum of the column elements in the k^{th} column of Q^p.
4. Each value of **a** is divided by the sum of the elements in **a** to yield weights $\mathbf{a_k}$ whose values are between 0 and 1.

2.1.1.5 Conductance probability distribution (CPD)

The following is a summary of what can be found in greater detail elsewhere.[62]

Conductance probability distributions are described by two distributions (open and closed distributions):

Open:

$$f_o(t) = \Phi_o \exp(Q_{oo}t)(-Q_{oo})u_o \quad [7]$$

$$\Phi_o = {p_c(\infty)Q_{co}}\Big/{p_c(\infty)Q_{co}u_o} \quad [8]$$

Closed:

$$f_c(t) = \Phi_c \exp(Q_{cc}t)(-Q_{cc})u_c \quad [9]$$

$$\Phi_c = {p_o(\infty)Q_{oc}}\Big/{p_o(\infty)Q_{oc}u_c} \quad [10]$$

Where the terms are defined as follows:

1. Φ_o, Φ_c are weighting terms for the open and closed distributions respectively.

2. Q_{oo} is the sub-matrix of Q whose rows and columns include only the open states.
3. Q_{cc} is the sub-matrix of Q whose rows and columns include only the closed states.
4. Q_{co} is the sub-matrix of Q whose rows are the closed states, and whose columns are the open states
5. Q_{oc} is the sub-matrix of Q whose rows are the open states, and whose columns are the closed states.
6. u_o, u_c are unit row vectors with dimensions *k* x 1 where k is the number of open or closed states.
7. $P_o(\infty)$, $p_c(\infty)$ are the steady-state occupancies of the open and closed states respectively (taken directly from $p(\infty)$).

These distributions can also be represented by a log-tranformed exponential decay as in equation 4:

$$g(t) = \sum_{i=1}^{N} a_i \exp(\ln(\lambda_i) + \ln(t) - \exp(\ln(\lambda_i) + \ln(t))) \quad [4]$$

Where λ_i are the eigenvalues of the matrices Q_{oo} or Q_{cc} for the open and closed distributions respectively. The weights a_i are defined as:

Open:

$$a_i = \frac{-1}{\lambda_i} \Phi_o A_i Q_{oo} u_o \quad [11]$$

Closed:

$$a_i = \frac{-1}{\lambda_i} \Phi_c A_i Q_{cc} u_c \quad [12]$$

Where A_i are the spectral matrices of Q_{oo} or Q_{cc} respectively. Spectral matrices are derived as described elsewhere[62].

2.1.1.6 State lifetime random sample set

In accordance with our random sampling with replacement model, randomly sampled distributions of state lifetimes are generated for each state in the kinetic model. The distributions for any state are randomly sampled from the individual component of the SPD corresponding to that state. Randomly sampled distributions are generated as follows:

1. Each component of g(t) from Equation 4 is evaluated in a specified range and incrementation (in this work it has been evaluated from -25 to 1 log-units of seconds, with an incrementation to yield an x-axis of 10 000 000 points).
2. The values of each evaluated g(t) component are then divided by the sum of the values in that respective component to yield probability values between 0 and 1.
3. From each scaled and incremented component in g(t), an array of random samples of lifetimes are generated (The sample size is an adjustable variable. In this work we generated 10 000 000 random samples per state).

These randomly sampled distributions are then used as the lifetime marble bags in the simulation algorithm. Step 2 implies that each component is no longer scaled in accordance with steady-state occupancies. In step 2, each component is now scaled equally. The only difference between the components is their mean lifetime, otherwise they have the same shape, and the same area. When events are simulated, the state components in the duration histogram are naturally scaled back to steady-state occupancy by the stochastic process of making transitions within the matrix P.

2.1.1.7 State transition random sample set

In accordance with our random sampling with replacement model, randomly sampled distributions of state transitions are generated for each state in the kinetic model. Randomly sampled state transition distributions are sampled directly from the probabilities of the probability matrix P as follows:

1. For each state in the kinetic model there is a corresponding row in P containing the probabilities of transitioning to another state.
2. A sample size is chosen for the randomly sampled distribution (in this work the sample size is 10 000 000).
3. For each state, 10 000 000 transitions are randomly sampled from the possible choices in the corresponding row of P.

These randomly sampled distributions are then used as the transition marble bags in the simulation algorithm.

2.1.2 SAMPLING ENGINE

2.1.2.1 Step-response matching

Step-responses are matched using a custom least-squares algorithm. The algorithm is a custom designed parameter-space searching algorithm (Figure 14). The reason why we use a custom designed algorithm is to make it possible to fit the parameters of a filter (or a cascade of filters) that has been applied to a signal. The algorithm can find three kinds of parameters: (1) the transition time of the square step, (2) one parameter of the step function (for example, the slope of a linear step response, or the time constant of an exponential step response), (3) the cut-off frequencies of any number of low-pass filters implemented in the simulator (see 2.1.2.5 for details about filters).

After the user specifies starting search boundaries for the parameter values, the fitting algorithm works as follows:

1. A parameter space is created whose dimensions are the number of parameters and whose boundaries are the specified search boundaries (e.g. a cubic three-dimensional space is created when fitting three parameters). Each possible combination of parameters is represented by the parameter space.
2. A step response is simulated for each possible combination of parameters.
3. The root mean squared error (RMSE) comparing the real step response to the simulated step response for each possible combination of parameters is calculated. The value of the RMSE is set in the location of the parameter space corresponding to the combination of parameters that were used to simulate the step response.
4. The algorithm checks the parameter space to see where the smallest RMSE is. If the smallest RMSE is found at any boundary of the parameter space, then the algorithm proceeds to step 5. Otherwise, the algorithm checks to see if the current smallest RMSE is within a percent difference cut-off of the previous smallest RMSE from an earlier iteration of the algorithm. If the current RMSE is within the percent-difference cut-off, then the algorithm terminates. Otherwise, the algorithm proceeds to step 6.

5. The algorithm re-adjusts the parameter search boundaries. First it zooms-out of the current space, then it shifts over in order to center the space around the smallest RMSE from step 4. Then the algorithm proceeds to step 1.
6. The algorithm re-adjusts the parameter search boundaries. First it shifts the parameter space to center the smallest RMSE from step 4, then it zooms-in in order to refine the values of the parameter space. Then the algorithm proceeds to step 1.

The algorithm ends on step 4 when the smallest RMSE of the current iteration is within a percent difference cut-off of the previous iteration's smallest RMSE. The values at the coordinates of the parameter space with the smallest RMSE are the resulting fit parameters.

The algorithm searches for a well in the parameter space where there exists a minimum value of the RMSE at the bottom of the well. Step 5 occurs when the parameter space does not have enough information to determine if the smallest RMSE is in a well (i.e. the smallest RMSE of the current space is hitting a boundary of the space.) Step 5 has the effect of zooming out the values of the parameter space with the intention to push the boundaries back so that the smallest RMSE is no longer hitting a boundary. Step 5 also has the effect of shifting the space over in order to make the smallest RMSE nearest to the center of the space as possible. This shifting has the purpose of aligning the smallest RMSE with the center to facilitate the zooming in effect of step 6. Step 6 occurs when the algorithm has enough information to determine if the smallest RMSE is in a well (i.e. the smallest RMSE of the current space is near the center of the space and is the smallest value in the space.)

The coordinates of the smallest RMSE in the space yield the parameters of the fit.

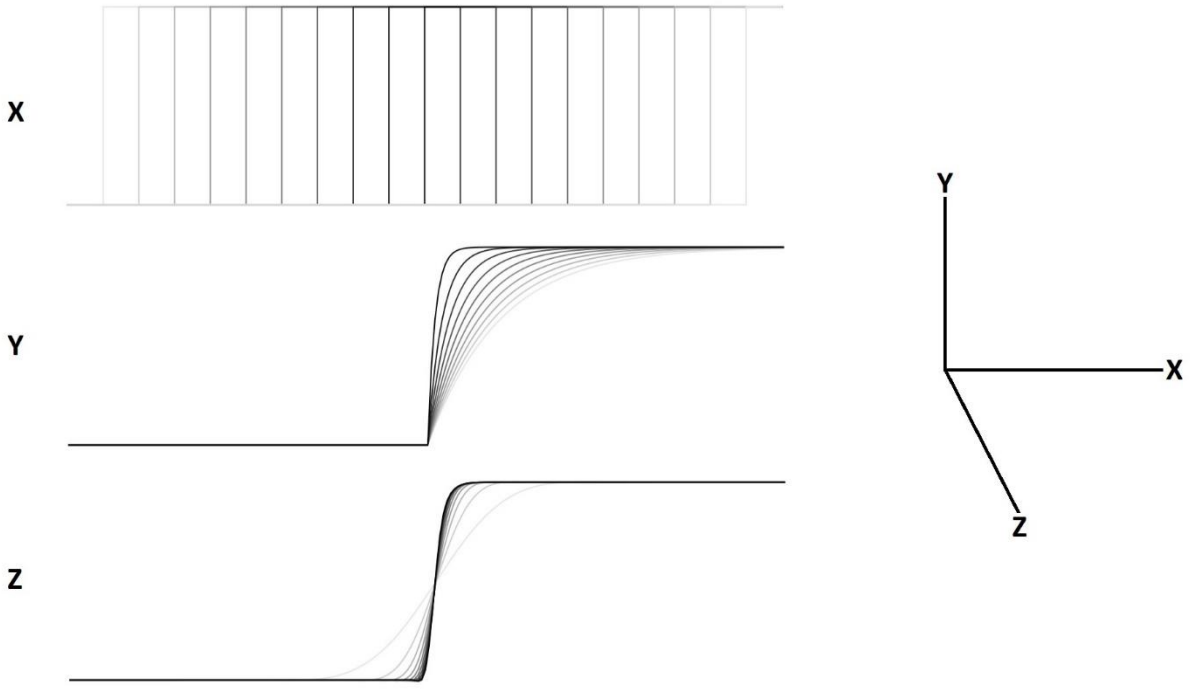

Figure 14. Visual depiction of the step-response fitting procedure. The example here involves fitting a step-response on three parameters: (X) the x-axis location of the square step, (Y) the RC time constant of the RC component, and (Z) the Bessel filter cut-off frequency. For this example, a 3-Dimensional parameter space is created with the axes (X, Y, and Z) containing a range of parameter values to be fit.

2.1.2.2 Noise modeling

Noise is modeled with an autoregressive (AR) model. The coefficients and variance of the model are found as the solution of the Yule-Walker equations. The solution to the Yule-Walker equations is implemented in the Python program Spectrum using the Levinson-Durbin recursion algorithm[66]. A sample of noise is loaded into the simulator application. If the length of the signal is less than 1 000 samples, then 25 % of the total number of samples is used as the number of coefficients to be fit. If the signal is greater than 1 000 samples, then 250 coefficients are fit.

2.1.2.3 Simulation of step-response

The simulation of one step response happens as a cascade of successive operations on an ideal square step. First, the exact transition time of the step must be converted into units of the sampling interval. Then an x-axis is created representing the stretch of time within which the step occurred in units of the sampling interval. The step-response function is then evaluated along the x-axis. Lastly, a digital approximation to an analog Bessel filter is applied to the step-response.

The maximum value for the size of the step-response is one, and the minimum value is zero. Any sub-conductance levels are values between zero and one.

The evaluation of an indefinite succession of steps is, in accordance with the superposition principle for linear systems, the sum of the individual steps. Each step yields an evaluation of the step-response function. Evaluated step-responses are summed together. Then as a final step a digital approximation to an analog Bessel filter is applied to the entire signal.

2.1.2.4 Simulation of noise

A stretch of noise the length of the simulated signal is generated using random Gaussian distributed values with zero mean, and variance determined by the AR model. The AR model is then applied to the random noise using an infinite impulse response (IIR) filter using the coefficients of the AR model as the denominator coefficients in the system function. Then the noise is scaled such that its standard deviation is equal to the standard deviation of the model noise. Any state dependent noise is also scaled appropriately for any stretch of time in which the system resides in that state. The Bessel filter is not applied here (like it is for the step-response,) because the AR model includes the frequency response of the analog Bessel filter. Thus, as a by-product of modeling the noise of an already Bessel-filtering system, we are capable of generating Bessel-filtered noise.

2.1.2.5 Application of filters

The list of filters available in the application are in addition the digital Gaussian filter, digital approximations of these analog filters:

1. Butterworth
2. Chebyshev Type I
3. Chebyshev Type II
4. Elliptic
5. Bessel

These filters are implemented in SciPy. The digital approximation filters are applied using a forward-backward bilinear method. The Gaussian filter is applied using a convolution method.

2.1.2.6 Power spectral density

The power spectral density of the noise are estimated using a Python implementation of the Fast Fourier Transform (FFT) method described and used by MATLAB[67].

1. The FFT of the noise is taken. Only the positive side of the spectrum is kept (the positive frequencies.)
2. The absolute values of the FFT are squared.
3. These values are scaled by the factor: $1/_{Fs \cdot N}$

   (Fs is the sampling frequency in Hz, N is the length of the signal in sample points.)
4. The values are then doubled (multiplied by 2). This is to preserve the total power, since half the spectrum was thrown out.
5. The final values on the y-axis are the $\log_{10}$ of the values from step 4.
6. The values on the x-axis are the $\log_{10}$ values of the frequency spectrum from 0 Hz up to half the sampling frequency.

The units of the y-axis are $\log_{10}(A^2)/Hz$. The units of the x-axis are in $\log_{10}(Hz)$. Taking the integral over the entire spectrum yields the total power of the noise, otherwise known as the variance of the noise (σ^2). Taking the integral over a small region of the power spectrum (e.g. between 100 and 1000 Hz,) gives you the contribution to the variance of the signal from frequency components having frequencies in the given range.

2.2 PRELIMINARY COMPARISON OF DETECTION METHODS

2.2.1 HMM settings

The SPM[1] was used to simulate events with the ACh concentration range below:

3, 6, 10, 18, 30, 60, 100, 180, 300 µM [ACh]

For each concentration of ACh, 6 500 000 events were simulated, the first 100 000 rectangular pulses were taken as the dwells to be sampled by the sampling engine.

2.2.2 Sampling settings

The sampling interval is 1 µs. The step-response includes an RC component with a time constant τ = 1.65 µs. The step-response also includes a 4-pole Bessel filter with a cut-off frequency of 194 Hz. (For details about the fitting of step-response parameters see 2.2.1 and 3.1.2.1).

The AR coefficients of the noise model are summarized in the appendix. The maximum open channel amplitude is 12.27 pA. The open channel standard deviation for the noise is 6.67 pA, the closed channel baseline standard deviation is 6.31 pA. The maximum amplitude and standard deviations for the noise were estimated using real single-channel data of the wild-type muscle-type nAChR (For details about the modeling of noise see 2.1.2.2, 2.1.2.4 and 3.1.2.2).

The simulated data were digitally filtered with a Gaussian filter with the following cut-off frequencies:

5, 10, 15, 20, 25, 30, 35, 40, 45, 50 kHz

2.2.3 TAC detection settings

TAC's own digital Gaussian filter was turned off. Thus, no filtering or modification of the data was performed on TAC's behalf. The specification of the rise-time was the smallest possible value for the chosen sampling interval at 0.001 ms. The maximum open amplitude was fixed at 12.27 pA with a half-amplitude threshold of 6.13 pA. As there are no secondary conductance levels, nor any baseline drift, the events were detected using TAC's fully automated detection option. The resulting list of transitions were then converted to dwells.

2.2.4 QUB detection settings

No modification or filtering was performed on the data on QUB's behalf. The model used for detection is the del Castillo & Katz extended model with block.[29] The open channel amplitude was fixed at 12.27 pA. The closed channel amplitude was fixed at 0.00 pA. The open channel and closed channel standard deviations for the noise depend on the digital Gaussian filter frequency, and are summarized in Table 3.

Cut-Off (kHz)	Baseline (A)	Open (A)
5	1.94E-13	2.05E-13
10	3.65E-13	3.86E-13
15	5.55E-13	5.87E-13
20	7.59E-13	8.02E-13
25	9.73E-13	1.02E-12
30	1.16E-12	1.23E-12
35	1.39E-12	1.47E-12
40	1.64E-12	1.73E-12
45	1.87E-12	1.97E-12
50	2.10E-12	2.22E-12

Table 3. Baseline and open channel standard deviations for simulated noise across a range of digital Gaussian filter cut-off frequencies.

2.2.5 Accuracy plots

Accuracy plots compare the duration of detected dwells to the duration of simulated dwells. The detected dwells that are used to make accuracy plots are the subset of detected dwells that correspond to actual simulated dwells. That is, the dwells used to make accuracy plots are not false events. Additionally, these detected dwells do not include those who contain underlying missed events. Both the simulated and the detected dwells are put on a $\log_{10}$ scale. The duration of simulated dwells are plotted on the x-axis, while the duration of corresponding detected dwells are plotted on the y-axis. Deviation of data points from the central line y = x indicate a deviation of the duration of the detected well from the actual duration of the simulated dwell.

2.2.6 Missed and false event plots

Missed and false events for each concentration and each filter setting were summed. The summed counts were then converted to a percentage. Missed event counts are converted to a percentage by dividing the count by the total number of simulated dwells 100 000. False event counts are converted to a percentage by dividing the count by the total number of detected dwells (varies depending on the condition). Thus, missed events are represented as a percent of the total number of possible dwells that could be detected (so that we can say: out of *x* simulated events, *y* % were not detected.) Moreover, false events are represented as a percent of the total number of events that were detected (so that we can say: out of *x* detected events, y % are falsely detected.) In the accuracy plots, if the percent of false events is greater than 100 %, the bar simply displays 100 %. The x-axis are the different S/N ratios (μ/σ) of the different filter settings in the range that we simulated with. The S/N ratios were calculated by dividing the maximum open channel amplitude of 12.27 pA by the standard deviations of each filtered baseline (see table 3.) The y-axis is on a percentage scale. The dots represent the percent of missed events, while the bars represent the percent of false events.

2.2.7 MIL settings

TAC detected dwells were corrected using a dead-time correction based on the cut-off frequency of the digital Gaussian filter. TAC dead-times are dependent on the cut-off frequency of the digital Gaussian filter. Since the dead-time for events detected by SKM-HMM is not well defined, we tried a range of dead-times (see table 7 in the appendix). For any given filter setting, the detected events for the entire concentration range were used as a global ensemble for model fitting. The model used for fitting is the SPM^1 with initial parameters being those that were used to simulate the data.

2.2.8 Kinetic parameter plots

The kinetic parameter plots compare the actual rate constants used for simulation with the fitted rate constants from MIL. The x-axis represent the range of values for the actual rate constants used for simulation. The y-axis represent the range of values for the rate constants recovered from MIL. Dots are [x, y] pairs corresponding to [actual, recovered] rate constants respectively. Deviation of the dots from the central line y = x show deviation of the recovered rate constant from the actual rate constant.

3 RESULTS

3.1 SIMULATOR

3.1.1 HMM Engine

3.1.1.1 HMM simulation algorithm

To begin a simulation, the user first loads into the application one or more kinetic models from a single model file (or multiple individual files.)

Then the user chooses simulator options which include:

1. **Models**: Which models to use.
2. **Channels**: How many channels are to be simulated for each model.
3. **Mode**: Either ‘time’ for a user-specified total simulation time, or ‘events’ for a user-specified total number of events.
4. **Append_flanks**: Whether or not to append flanking events of fixed duration at the beginning and end of the simulation.
5. **Initial_state**: Either a randomly chosen initial state according to stead-state occupancies, or a manually chosen initial state.

When the options are chosen, the simulation will run until the Mode criteria is met. The first step before the algorithm begins to run is to calculate the SPD for each model. Then the simulator converts the transition rate matrix Q into the probability matrix P. Then the simulator generates the random sample distributions for state lifetimes and state transitions for each model. These distributions act like marble bags in what is essentially a marble-picker algorithm modeled as random sampling with replacement. The core of the simulation algorithm iterates over these four steps (given an initial state):

1. Chose and log the duration of the event according to the current state's random sample lifetime distribution.
2. Log the conductance level of the current state.
3. Log the matrix index of the current state.
4. Chose the next state according to the current state's random sample transition distribution.

The resulting output of the simulation (assuming we simulated with one model and one channel for simplicity,) is a 32-bit or 64-bit data array with three columns: time, cstate (conductance level), and mstate (model state). The time column has the event durations in seconds. The cstate column has the conductance levels of the events. The mstate column has the state of the event indexed by its position in the transition rate matrix (with 0 being the first position).

The initial output contains more information than we would be able to observe in real data. Namely, the state of the channel is known for every event. This information is hidden from us in real data, hence the Hidden Markov Model. Given that it is possible to transition from a closed state to another closed state (likewise for any other conductance level,) the simulated data contains successive events in which the conductance level remains the same. However, since events in real data are defined as rectangular current pulses, events detected by a detection algorithm will always alternate between different conductance levels. One may choose to save (write to disk) the initial output of the simulation containing the hidden information for any purpose. Additionally, the simulator provides the user with the option to convert the initial output into a data array of ideal rectangular pulses, where each successive event alternates in conductance level (Figure 15). This is achieved by summing together all successive event durations of like conductance level. Both the initial output and the modified rectangular pulse output are compatible with the sampling engine, and they will both yield the same output of realistic noisy data with the same rectangular pulses. However, it is preferable to convert the initial output to rectangular pulses, since this is economical for computer memory, and will speed up the sampling algorithm considerably.

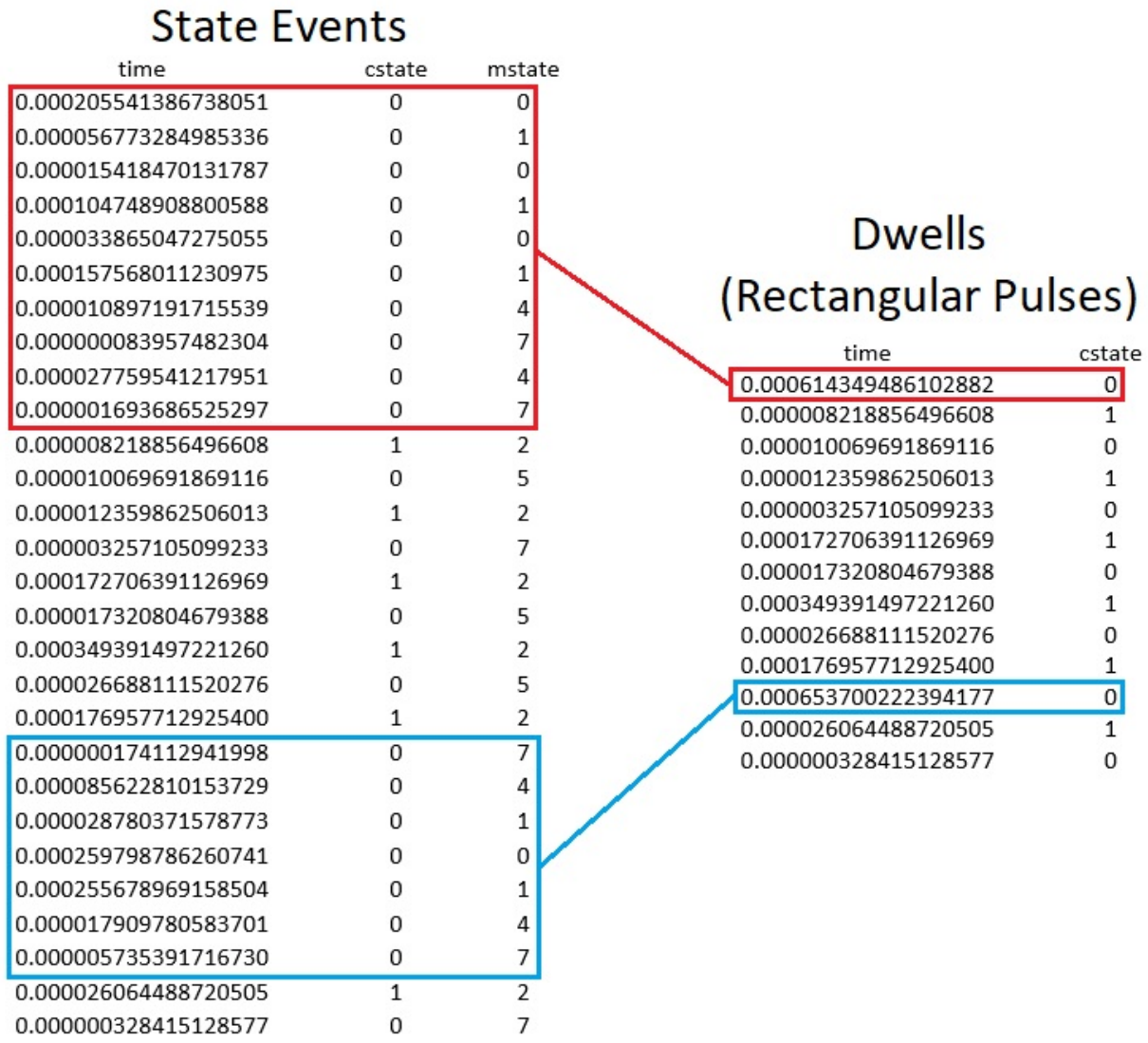

Figure 15. Example output of the HMM engine. On the left, the direct output of the HMM engine displaying three columns of information: the duration of the event (time,) the conductance level (cstate,) and the state indexed by its position in the kinetic matrix (mstate). On the right, a modified output of the HMM engine where successive states of like conductance level have been pooled together. The red and blue cases show the effect of pooling, where the durations of successive states of like conductance have been summed together.

3.1.1.2 Transition sequence of events

We want to know if the HMM simulation algorithm behaves according to the first property of our defined Markov process (see 1.1.5). That is, we want to know if the simulator generates transitions in accordance with the transition rate matrix Q that defines the set of allowable transitions. To validate the HMM simulation algorithm, we have designed a test. The output of the test tells us if the algorithm has generated any false positive transitions (seeing a transition where we shouldn't,) or false negative transitions (not seeing a transition where we should).

The test takes as an input the initial output of the simulation containing the *a priori* state information (see 3.1.1.1). The test also requires the transition rate matrix for comparison. When the test starts, a matrix of the same dimensions as the transition rate matrix is made, except all of its entries are zero (lets call this matrix C). The test then scans the input. For each transition, the test applies a plus one count in C where that transition occurred. For example, for an 8-state model states are indexed according to their position in the matrix from 0 to 7. A transition from state 2 to state 5 would be counted up in C in the location of the second row and the fifth column. At the end of the test, the matrix C tells us how many of each kind of transition was seen during the simulation. If no transitions of a particular kind were seen then a zero can be seen in the corresponding location in C.

The matrix C is then compared to Q. The comparison will return a message with the location and type of discrepancy between the two matrices. A false positive is detected if in Q there is a zero where in C there is a positive integer. A false negative is detected if in C there is a zero where in Q there is a positive integer.

We ran this test on three different kinetic models: (1) The SPM at 3 µM ACh, (2) A modified SPM where one of the transition rate constants is 1 s^{-1}, and (3) and 8-state model where all the states are connected with the same rate constant of 100 s^{-1}.

For (1) and (2) we tested across a range of amount of transitions listed below:

[10, 100, 1000, 10 000, 100 000, 1 000 000, 10 000 000, 100 000 000 transitions]

For (3) we tested for 1 000 000 transitions.

We want to see for the given models roughly how many transitions are needed in order not to see any false negatives. For (1), the algorithm consistently simulates without any false negatives with 1000 events or more. For (2), the algorithm consistently simulates without any false negatives with 10 000 events or more. For (3), 1 000 000 events is sufficient to not see any false negatives. No false positives for any model were detected.

The results of the test can be found in the appendix (see 5.5.) Based on these results, we can conclude that the simulator does not generate transitions which are not allowed by the matrix Q.

3.1.1.3 State distributions

We want to know if the simulator generates state events of durations that follow the shape, mean, and relative size of the individual components of the SPD. To do this we designed a test that involves fitting the observed distribution of state event durations to the individual components of the SPD. The test passes if the parameters from the fit are below 1 % different of the expected parameter. There are three parameters with which we compare the simulated event duration histogram and the exact expected parameter used in the simulation: (1) The mean $1/\lambda_i$, (2) the parameter λ_i, and (3) the weight a_i. The fits are performed using a non-linear least squares fit implemented in Scipy (namely, scipy.optimize.curve_fit). The function used for the fit is Equation 4.

The observed mean $1/\lambda_i$ is calculated as an arithmetic mean. The expected mean $1/\lambda_i$ is calculated using the known parameter λ_i.

The observed parameter λ_i is obtained for each state by fitting the observed duration histogram to equation 4. The observed duration histogram is log-binned (natural log) in 5 000 bins on a range from -25 to 0 log-units in seconds. The plots, however, display the event durations with an x-axis on a log base 10 scale in 100 bins for visual purposes.

The observed weight a_i is obtained by dividing the number of observed transitions into a state by the total number of transitions. The expected weight a_i are calculated as described elsewhere (see 2.1.1.4).

We tested two different models: (1) the SPM model at 3 µM ACh, and (2) a model with all states connected and all rate constants equal to 100 s^{-1} (see 5.1 (1) and (2) for the models.) The tests are run in triplicate. The mean of the observed parameter for the three trials is compared to the expected parameter using the formula for percent difference below:

$$percent\ difference = \frac{(observed\ value - expected\ value)}{expected\ value} * 100 \quad [13]$$

For (1), we tested first with 1 000 000 events. Two parameters came back with more than 1 % difference between the observed and expected parameter. The mean $1/\lambda_6$ for state 6 had a percent difference of -1.01 %, and the parameter λ_3 for state 3 had a percent difference of -1.78 %. Then we tested with 10 000 000 events. No parameter came back with more than 1 % difference between the observed and expected parameter (Figure 17).

For (2), we tested with 1 000 000 events. No parameter came back with more than 1 % difference between the observed and expected parameter (Figure 16).

The results are summarized in tables 8 to 13 in section 5.6 in the appendix. Based on these results, the simulator appropriately models an HMM with state probability densities that approximate equation 4.

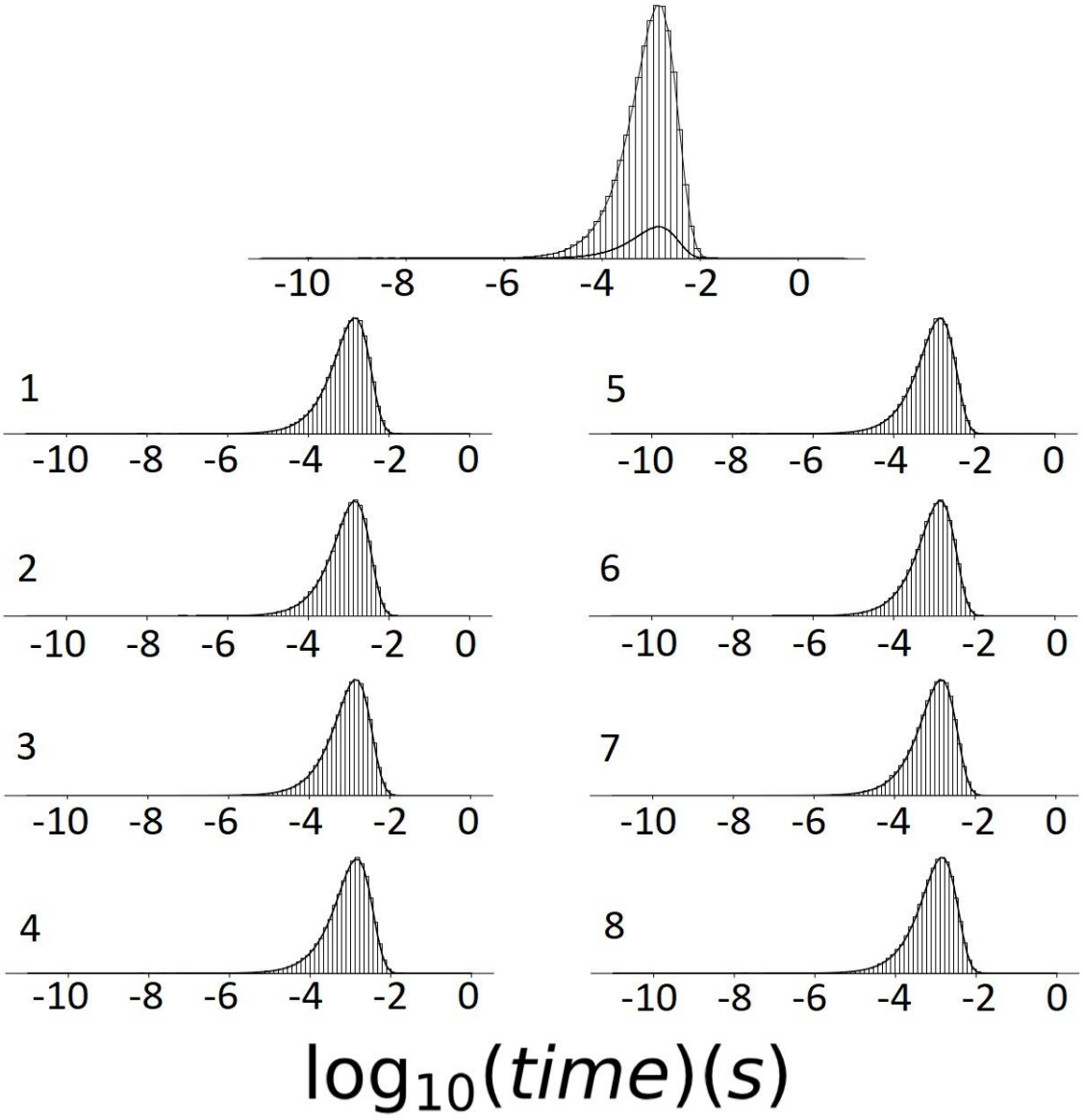

Figure 16. Results of the state distribution test part 1. Results for one trial (trial 1) of the test for an 8-state model with all states connected with rate constants of 100 s^{-1} (see 5.1 (2) for model.) The top diagram shows a duration histogram with all simulated events, as well as the fit SPD with its individual components. The diagrams labeled 1-8 show duration histograms corresponding to the simulated events of the individual components, as well as the fit of the individual component. The components were fit according to the SPD (see section 2.1.1.4 for details). The x-axis shows event duration in $\log_{10}$ seconds. The y-axis is omitted for visual purposes, but represents proportion of events in a particular state (see 2.1.1.4 for details about the y-axis).

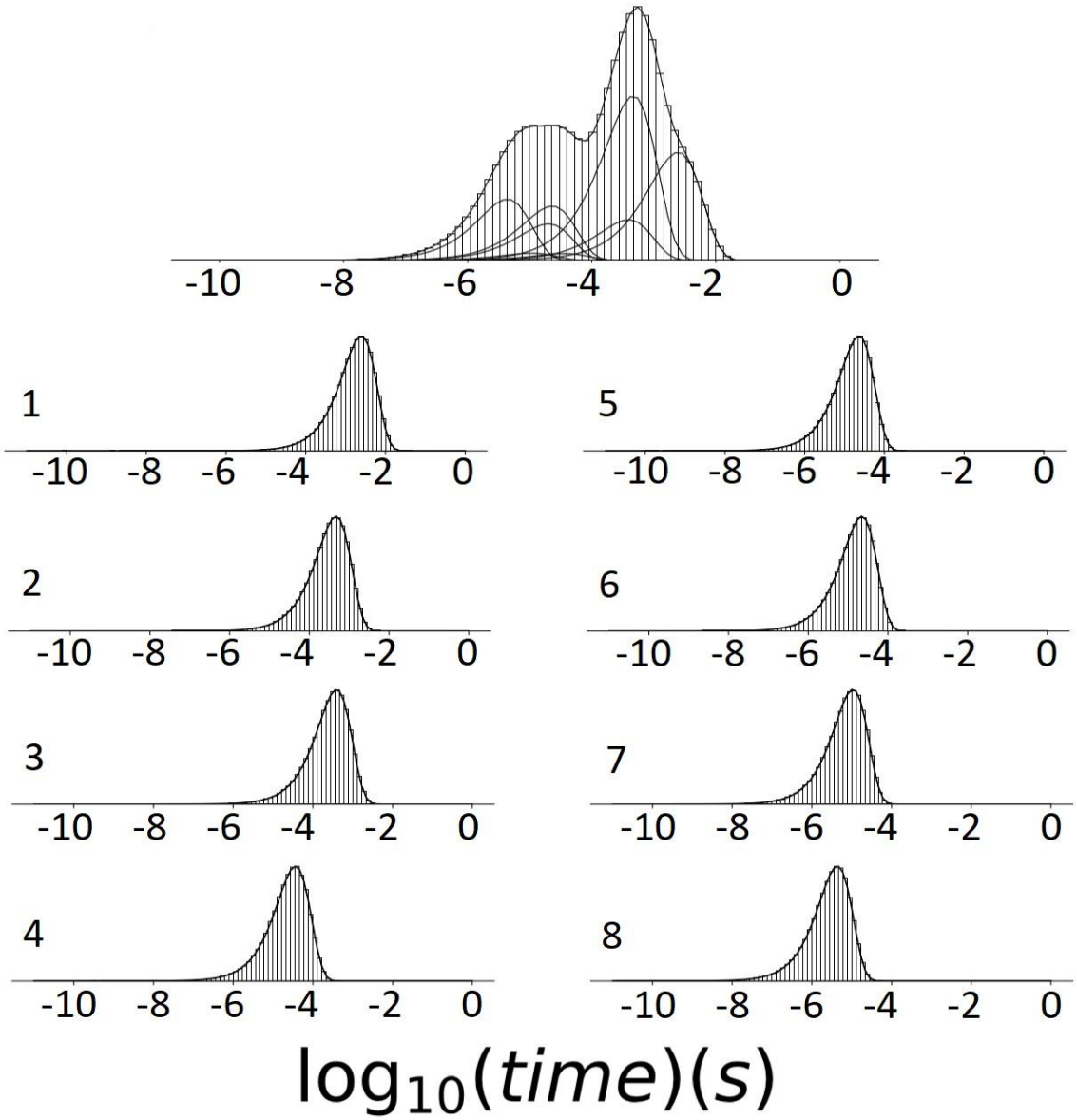

Figure 17. Results of the state distribution test part 2. Results for one trial (trial 1) of the test for the SPM[1] at 3 µM ACh. The top diagram shows a duration histogram with all simulated events, as well as the fit SPD with its individual components. The diagrams labeled 1-8 show duration histograms corresponding to the simulated events of the individual components, as well as the fit of the individual component. Diagrams are labeled according to their state in table 1. The components were fit according to the SPD (see section 2.1.1.4 for details). The x-axis shows event duration in $\log_{10}$ seconds. The y-axis is omitted for visual purposes, but represents proportion of events in a particular state (see 2.1.1.4 for details about the y-axis).

3.1.1.4 Conductance distributions

We want to know if simulated data from the HMM engine follows the shape of the CPD after turning the data into rectangular pulses (see 3.1.1.1). To know this we designed a test comparing the duration histogram of the simulated rectangular pulses to the CPD by a measurement of the root-mean-square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(E_n - O_n)^2}{N}} \quad [14]$$

Where E_n is the expected value, and O_n is the observed value for N samples.

We tested a triplicate set of 1 000 000 dwells simulated from the SPM at 3 µM, 30 µM, and 300 µM [ACh] (Figure 18). The events of the observed duration histogram are natural log-binned in 5 000 bins in a range from -25 to 0 log-units of seconds. The expected CPD was calculated on the same range, and with the same incrementation as the observed duration histogram. The x-axis values of the CPD are the center value of the corresponding bin in the observed duration histogram. The CPD was scaled by dividing the y-axis values by the sum of the y-axis values. The observed duration histogram was scaled by dividing the event count in each bin by the total event count. Scaling both distributions ensures that their y-axes are on the same scale. The RMSE is thus a measure of the difference in shape between the two distributions. The expected values E_n are the scaled values of the y-axis of the CPD. The observed values O_n are the scaled bin counts of the observed duration histogram. Both E_n and O_n are values between 0 and 1.

- For the 3 µM model, the mean RMSE of the triplicate set is 1.42e-05 +/- 6.80e-08.
- For the 30 µM model, the mean RMSE of the triplicate set is 1.43e-05 +/- 8.51e-08.
- For the 300 µM model, the mean RMSE of the triplicate set is 1.45e-05 +/- 1.91e-07.

Given that the values of E_n and O_n are between 0 and 1, an RMSE in the order of 10^{-5} is a strong indication that there is little difference between the expected shape of the CPD and the observed shape of the CPD. Given enough sampling of states from the SPD, we are confident that the observed CPD approximate the expected CPD.

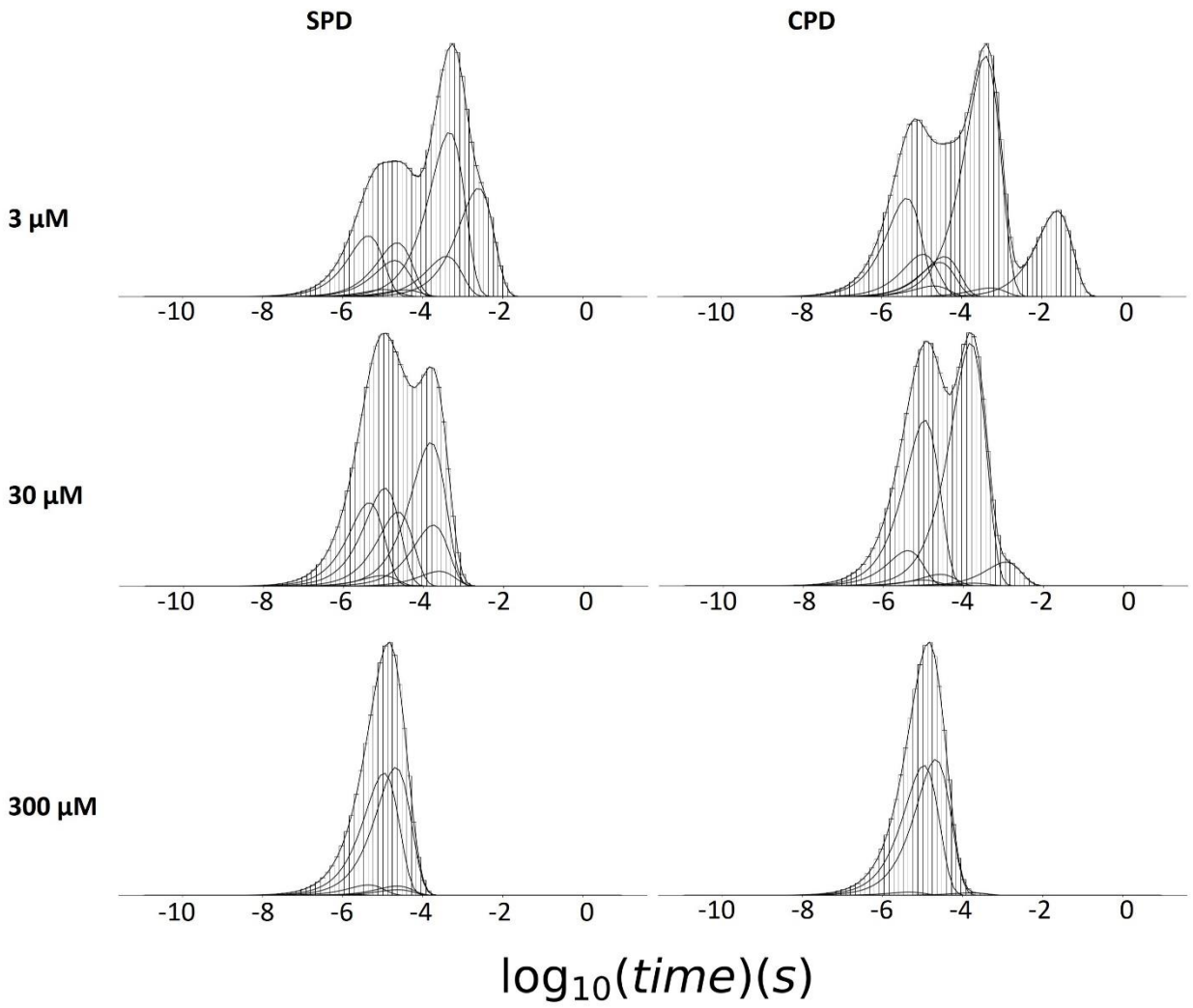

Figure 18. Results from the conductance distribution test. Events simulated from the SPM at three different concentrations (3, 30, and 300 µM [ACh]) were converted to rectangular pulses (dwells). The SPDs and event duration histograms are plotted on the left column, while the CPDs (reflecting the conversion to rectangular pulses) with corresponding dwell duration histograms are plotted on the right column. Duration histograms show events and dwells log-binned into 100 bins on an x-axis from -11 to 1 log-units of seconds. For details about the test, and the calculation of CPDs see sections 3.1.1.4 and 2.1.1.5.

3.1.2 Sampling Engine

3.1.2.1 Matching measured step-responses

To match the step-response of the patch-clamp we used a measured step-response containing an RC component and an analog Bessel filter component. The step-response was originally sampled with an interval of 0.2 μs, but was down-sampled to a 1 μs interval for the purpose of simulating a data set with a sampling interval of 1 μs. The step-response parameters were fit according to our custom algorithm. The parameters of the fit were the Bessel filter's cut-off frequency in Hz, and the time constant τ of the RC component in seconds. The fit returned 194Hz for the Bessel cut-off frequency, and 1.65 μs for the time constant τ, with a final minimum RMSE value of 0.00152.

The time constant τ from the fit (1.65 μs) came back relatively close to the expected value of the real patch-clamp[33] which is roughly 1.6 μs.

The minimum value of RMSE of 0.00152 is low considering the step-response values are scaled to be in the range between 0 and 1. The average difference between the real values and the simulated values is 0.00152. This means that the sample points of the simulated step response differ from the sample points of the real response on average by a number which is 0.152 % of the maximum open amplitude value (Figure 19).

Figure 19. Fit results over a measured step-response at 1 µs sampling interval. The black dots represent the data points from the measured step-response. The transparent red dots represent an overlaid simulated step-response with the results from the fit. For details about the fit and parameters see section 3.1.2.1.

3.1.2.2 Matched measured noise

To fit the AR model coefficients to the patch-clamp noise we used a stretch of real patch-clamp baseline noise containing 69 000 sample points (Figure 20). The maximum open amplitude, baseline standard deviation, and open channel standard deviation were matched to a real recording of a WT muscle-type nAChR with 10 µM [ACh]. The baseline mean current and standard deviation are 3.22 +/- 6.31 pA. The open channel mean current and standard deviation are 15.49 +/- 6.67 pA. The maximum open amplitude is obtained by subtracting the mean open current from the baseline current:

$$O_{MAX} = O - C$$

$$O_{MAX} = (15.49 - 3.22)\ pA$$

$$O_{MAX} = 12.27\ pA$$

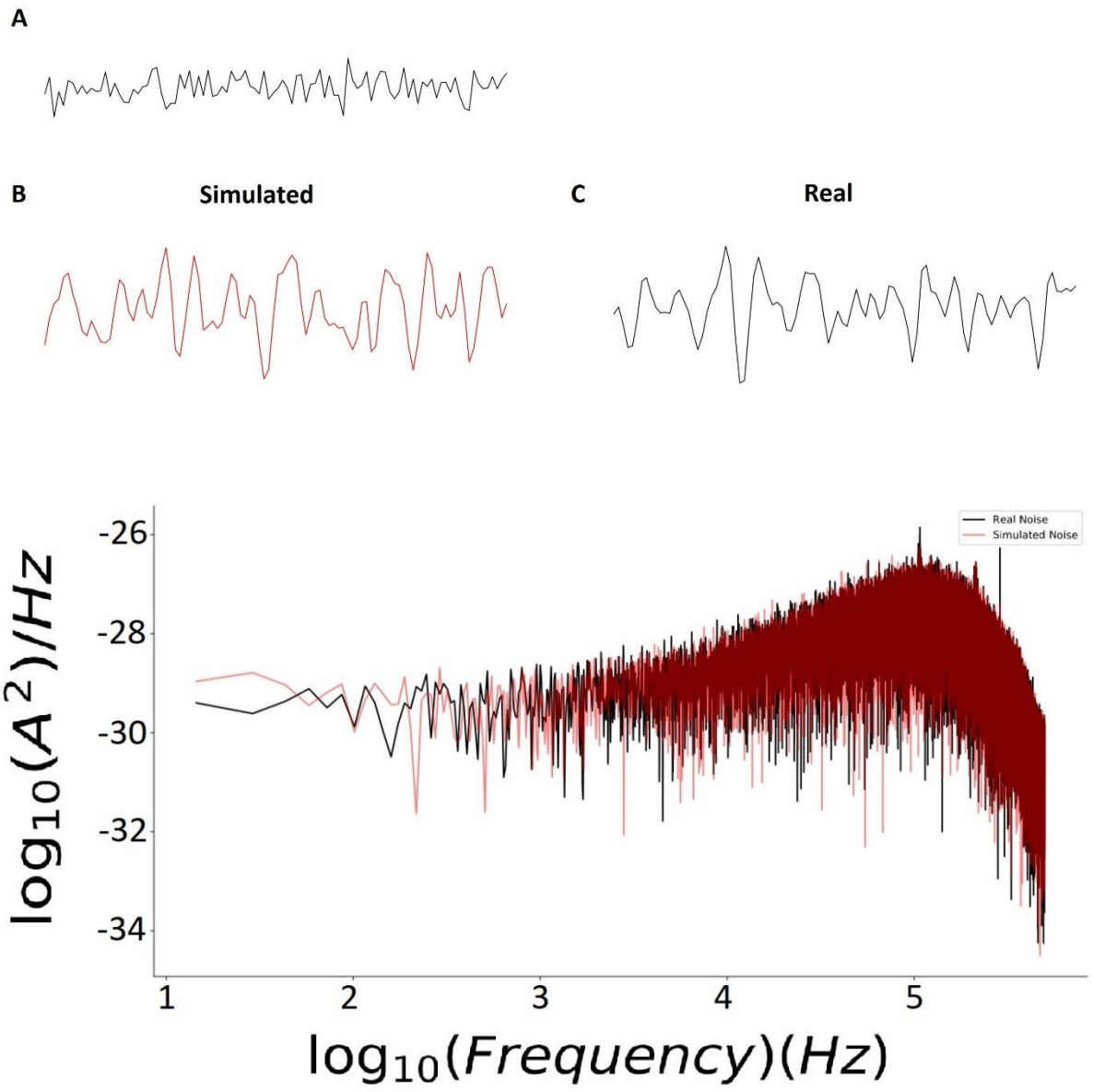

Figure 20. Autoregressive modeling of real noise. (A) Simulated Gaussian-distributed white noise. (B) The same noise as in (A), but with an applied AR model, and scaled to an appropriate standard deviation. (C) a segment of the real patch-clamp noise used to fit the coefficients of the AR model. The power spectrum shows an overlay of the modeled noise spectrum over the real noise spectrum. This example noise was modeled in the same way that was used for the simulated data set for the preliminary comparison of detection methods. For details about the AR modeling see sections 2.1.2.2, 2.1.2.4 and 3.1.2.2. For details about the power spectrum see section 2.1.2.6.

3.1.2.3 Sampling interval

The sampling interval for the sampling engine can be any interval of time. The simulator converts the units of the time constant of the step response into units of the sampling interval. This allows for the normalization of the x-axis such that it is the same no matter the sampling interval. The result is that the step response function, which is evaluated on the x-axis gives the same result with any sampling interval (Figure 21). Conversion of the time constant in units of the sampling interval is also useful to simplify the code of the sampling algorithm. By normalizing everything in units of the sampling interval, the math of the sampling algorithm is the same regardless of the sampling interval. For example, the exact transition time between sample points is always going to be a value between 0 and 1 in units of the sampling interval.

We conducted a small test to make sure that there is no difference between traces sampled at different sampling intervals. The test involves taking the difference between sample points for the same set of dwells sampled at different sampling intervals. The set of dwells tested here are the same as those displayed in Figure 6. We sampled these dwells with three different sampling intervals: 0.2, 1, and 2 µs. For clarity, lets refer to each sampled trace as *x*, *y,* and *z* (for 0.2, 1, and 2 µs intervals respectively). We took the linear difference (for overlapping sample points) between these traces as such: (*x – z)*, (*y – z*), and (*x – y).* The superposition of the traces, and their linear differences are plotted in figure 22. The results show that there is zero difference between the amplitude values for any of the difference pairs. In other words, the amplitude value at analogous times is the same irrespective of the sampling interval.

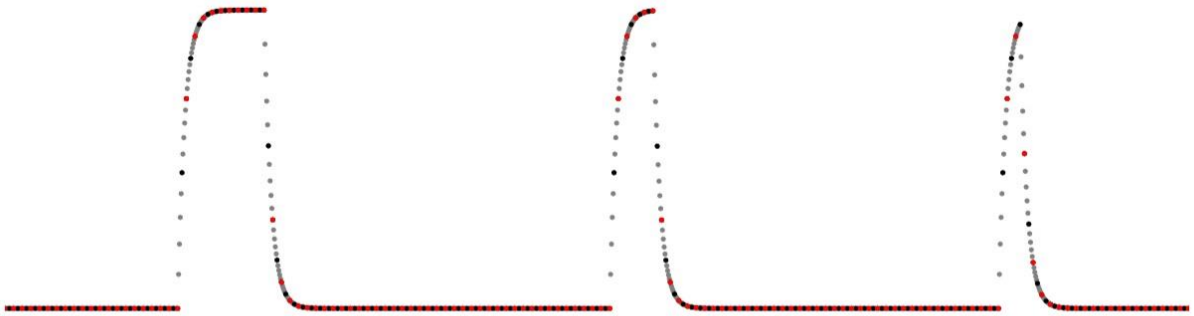

Figure 21. Simulated events at different sampling intervals. Three simulated rectangular pulses separated by 80 µs intervals (from left to right: 20, 10, 5 µs). The gray dots are sampled at 0.2 µs. The black dots are sampled at 1 µs. The red dots are sampled at 2 µs.

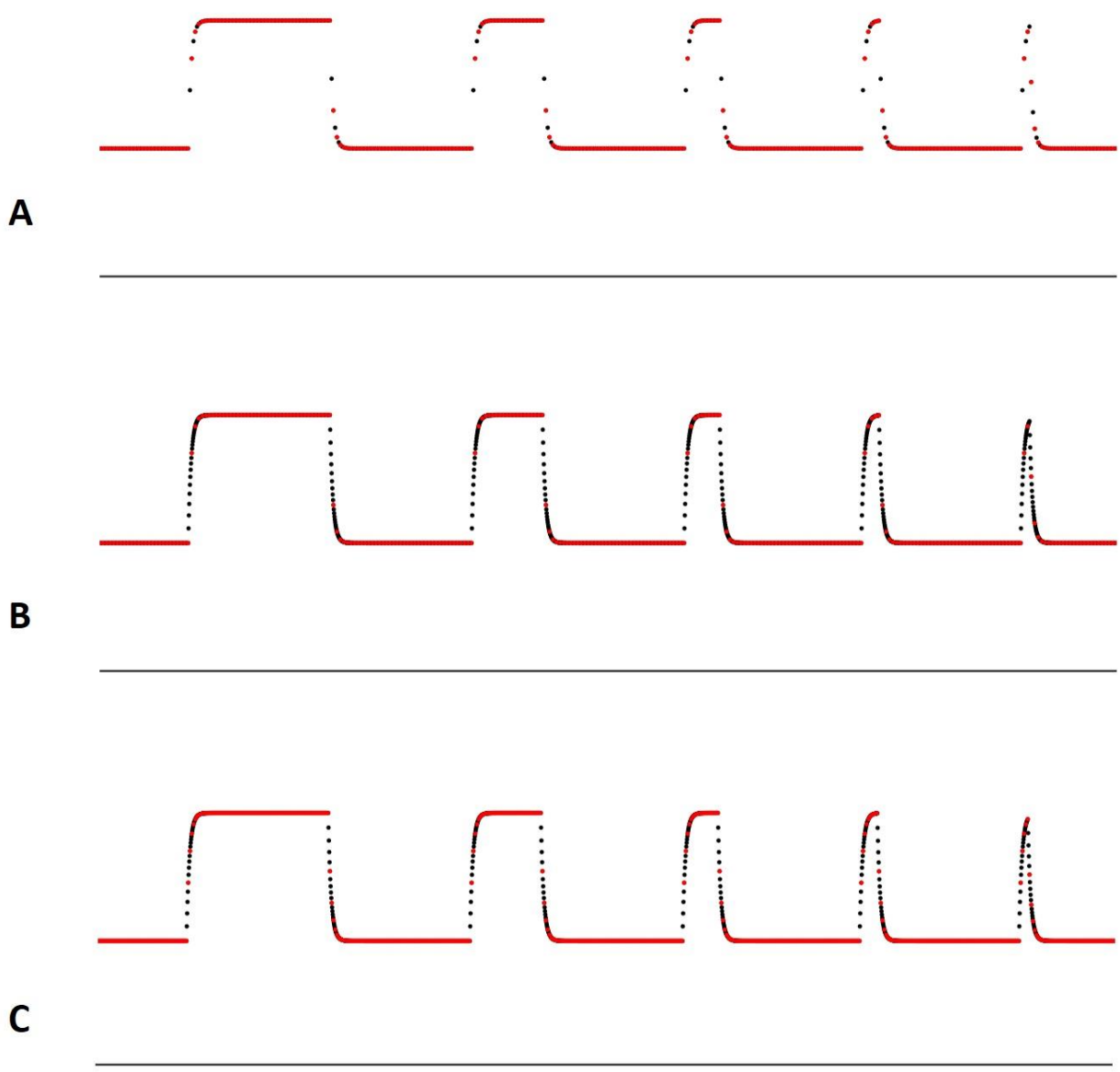

Figure 22. Results of the sampling interval test. Each of A, B, and C show two traces. The top trace is an overlay of sampled events (see Figure 6 for event details.) The bottom trace shows the difference between the overlapping points of the top trace (overlaid events). (A) The difference between 1 (black) and 2 (red) µs sampling intervals. (B) The difference between 0.2 (black) and 2 (red) µs sampling intervals. (C) The difference between 0.2 (black) and 1 (red) µs sampling intervals.

3.1.2.4 Sampling algorithm

The sampling algorithm is modeled as a linear time-invariant (LTI) system. Previously simulated ideal square pulses (from the HMM engine,) are transformed into sampled sequence of step-responses. Noise is generated and modeled separately. Then the sampled step-response signal is summed with the generated noise (Figure 23). It is optional, but recommended for analysis, to write to disk the signal without noise, the signal with noise, as well as the noise alone.

The user chooses sampling options which include:

1. **Sample_rate:** the sample rate in Hz (inverse of the sampling interval).
2. **Step_response:** the step-response function.
3. **Step_params:** the parameters of the step-response which includes the RC time constant and the filter type, filter order, and cut-off frequency in Hz.
4. **Noise_model_params:** the parameters of the AR noise model; the variance and the coefficients.
5. **State_stds:** the standard deviations of baseline and state-dependent noise in Amperes.
6. **Max_amp:** the value of the maximum open amplitude in Amperes.

Once the sampling options are chosen, the user loads into the application a sequence of dwells. The dwells are then sampled as described above in the order presented in Figure 23.

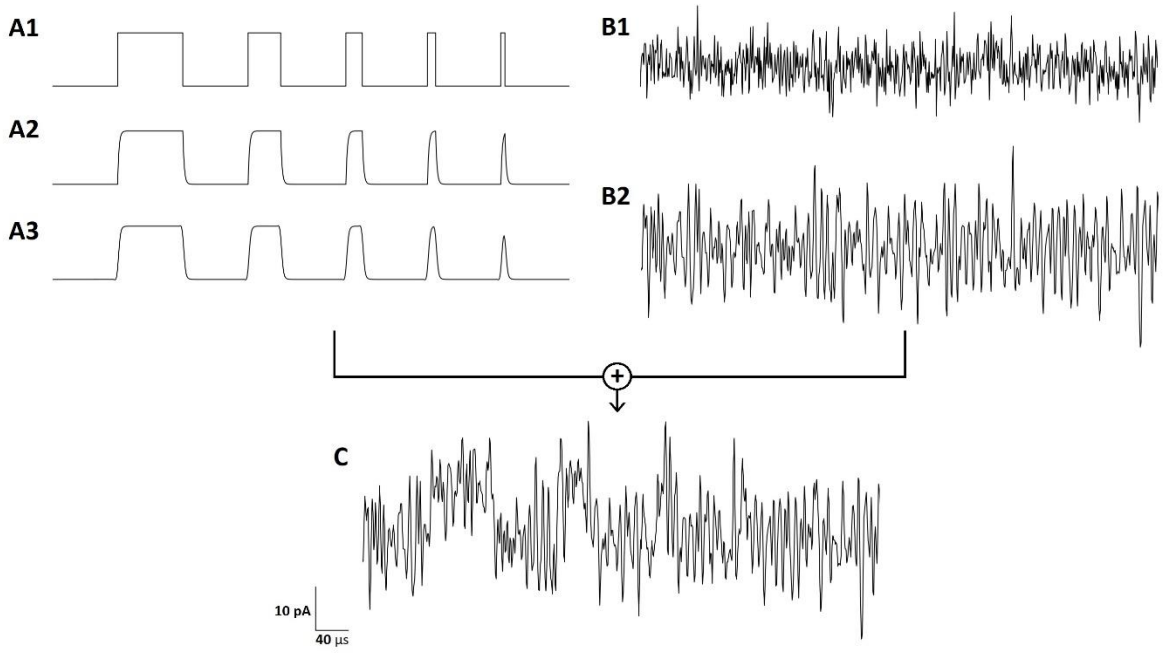

Figure 23. Depiction of the sampling process. (A1) Events are represented as ideal rectangular pulses (A2) The step response function is applied, turning the ideal pulses from A1 into sampled step-responses (sample points not shown for visual purposes.) (A3) The Bessel filter is applied to the sampled signal from A2. (B1) Random Gaussian-distributed white noise is generated with variance determined by the AR model. (B2) The AR model is applied to the noise from B1, the noise is then appropriately scaled. (C) A3 and B2 are summed to give a sampled noisy trace with underlying events.

3.2 Preliminary Comparison of Event Detection Methods

3.2.1 Overview of detection results

Figures 24 and 25 show some examples of idealized dwells detected by Threshold Crossing and by SKM-HMM. In general (across the conditions that we tested,) both methods detect fully resolved openings with accurate duration. There seems to be a range of filter frequencies over which both methods perform reasonably well. Figure 24 shows a segment of a 30 μM [ACh] trace filtered at three cut-off frequencies within the range where the methods perform well (15-35 kHz.) At filter cut-off frequencies bellow 15 kHz both methods start missing an appreciable number of events. At filter cut-off frequencies above 35 kHz both methods start introducing false events. Figure 25 shows segments from three traces at different concentrations of ACh (10, 30, and 100 μM [ACh],) all filtered with a digital Gaussian cut-off frequency of 25 kHz. As the concentration of ligand increases, the kinetics get faster, and both methods seem to miss the same brief events that show up more often in the higher end of the concentration range (due to rapid open channel blocking and unblocking by agonist). Figures 26 and 27 show some example dwell duration histograms of the events that were detected by Threshold Crossing and SKM-HMM. Figure 26 shows the dwell duration histograms for events detected from the entire trace at 30 μM [ACh] across the same range of filter cut-offs as in Figure 24. What can be observed across the duration histograms of Figure 26 is that both methods show an overestimation of longer events. This overestimation is shown by the discrepancy between the real duration histogram and the detected duration histograms for event durations in the range between 10^{-3} and 10^{-2} seconds. Another observation is that both methods not only miss the briefest events, but also miss a sizable amount of the longer events in the duration range between 10^{-4} and 10^{-3} seconds. Increasing the filter cut-off frequency allows both methods to detect more events. In every case, Threshold Crossing misses fewer events than does SKM-HMM (this is apparent when we observe the missed event plots in Figure 28). Figure 27 shows duration histograms for events detected from three traces at different concentrations of ACh filtered with a 25 kHz digital Gaussian filter (in parallel with the conditions for Figure 25). At a 25 kHz cut-off, across the range of ligand concentration, both methods show a similar drop-off in detected events of duration between the range of 10^{-5} and 10^{-6} seconds. The drop-off for Threshold crossing tends to be more gradual than the drop-off for SKM-HMM.

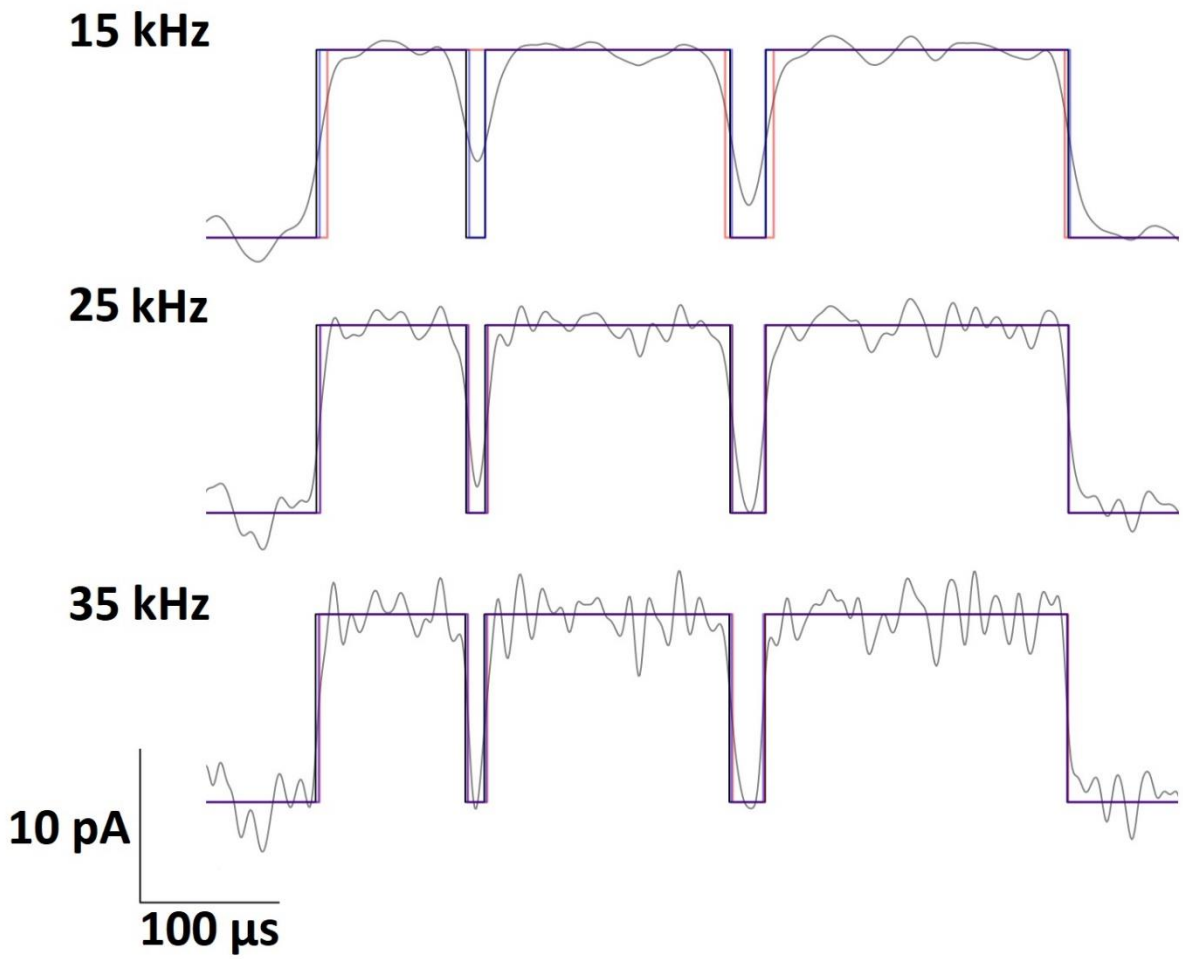

Figure 24. Events detected across a range of filter cut-off frequencies. A segment of a simulated data from the SPM at 30 µM [ACh] filtered with a digital Gaussian filter at cut-off frequencies of 15, 25, and 35 kHz. The original trace is in gray. The simulated ideal dwells are in black. Detections from Threshold Crossing are in transparent blue. Detections from SKM-HMM are in transparent red.

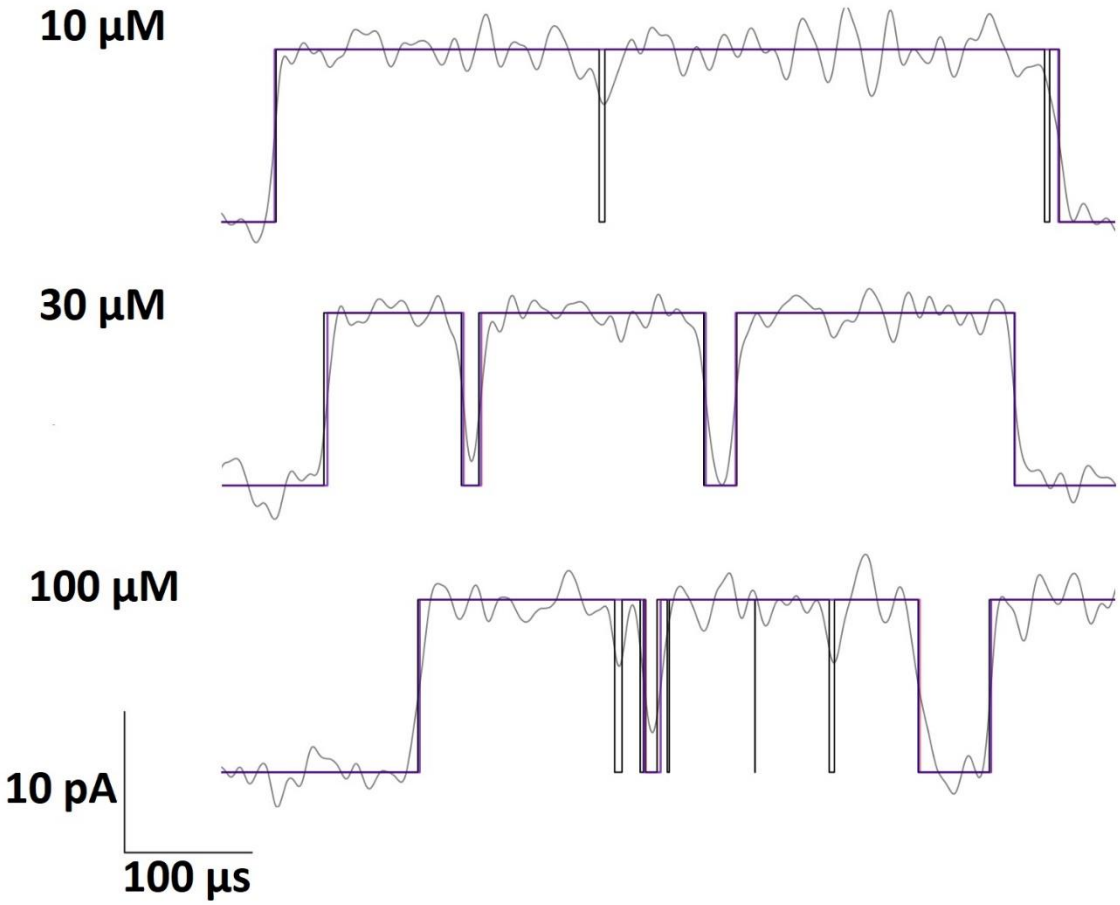

Figure 25. Events detected across a range of ligand concentration. A segment of a simulated data from the SPM at 10, 30, and 100 µM [ACh] filtered with a digital Gaussian filter at cut-off frequency of 25 kHz. The original trace is in gray. The simulated ideal dwells are in black. Detections from Threshold Crossing are in transparent blue. Detections from SKM-HMM are in transparent red.

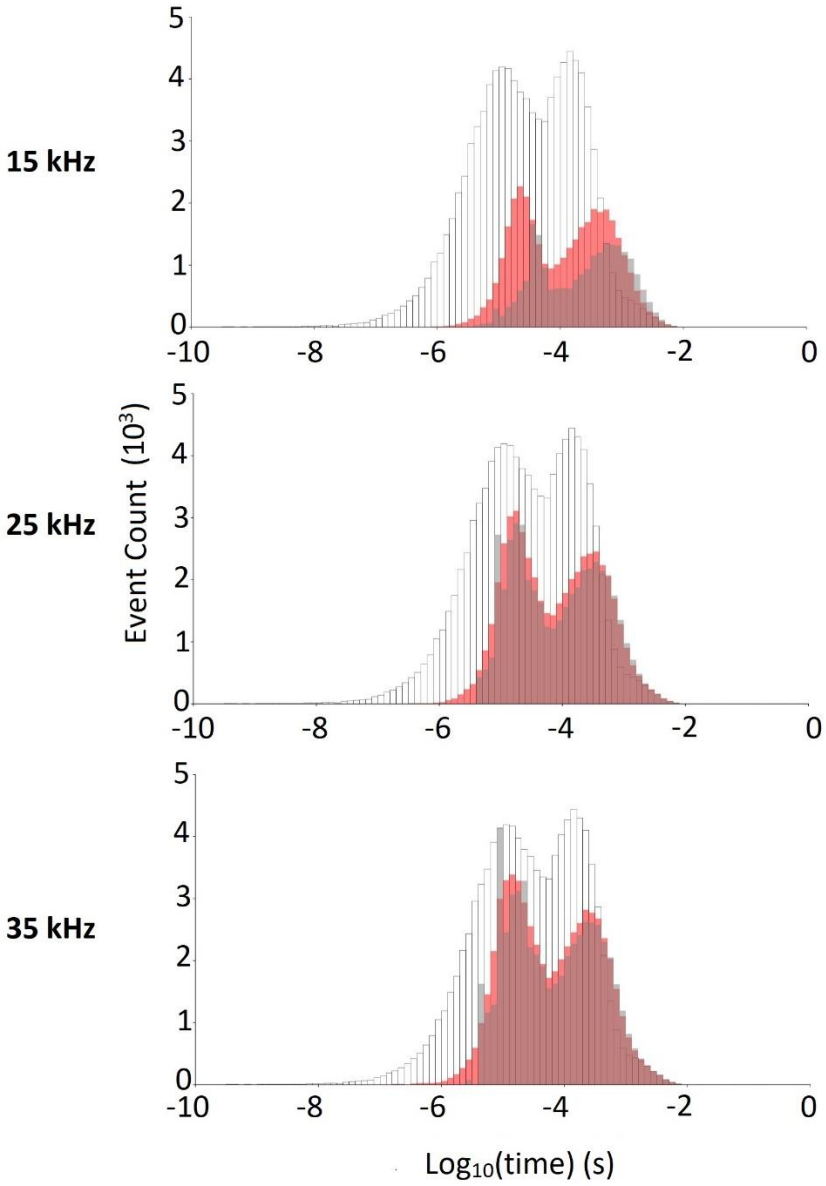

Figure 26. Dwell duration histograms for detected events. Dwell duration histograms comparing events detected from simulated events of the SPM at 30 µM [ACh] at three digital Gaussian filter cut-off frequencies (15, 25, 35 kHz). The white bins are the real dwell durations. The transparent red bins are dwell durations from Threshold Crossing detected events. The transparent gray bins are dwell durations from SKM-HMM detected events. The y-axis shows event counts. The x-axis display $\log_{10}$ values of time in seconds. Dwells were log-binned in 100 bins on an x-axis range from -11 to 1 log-units.

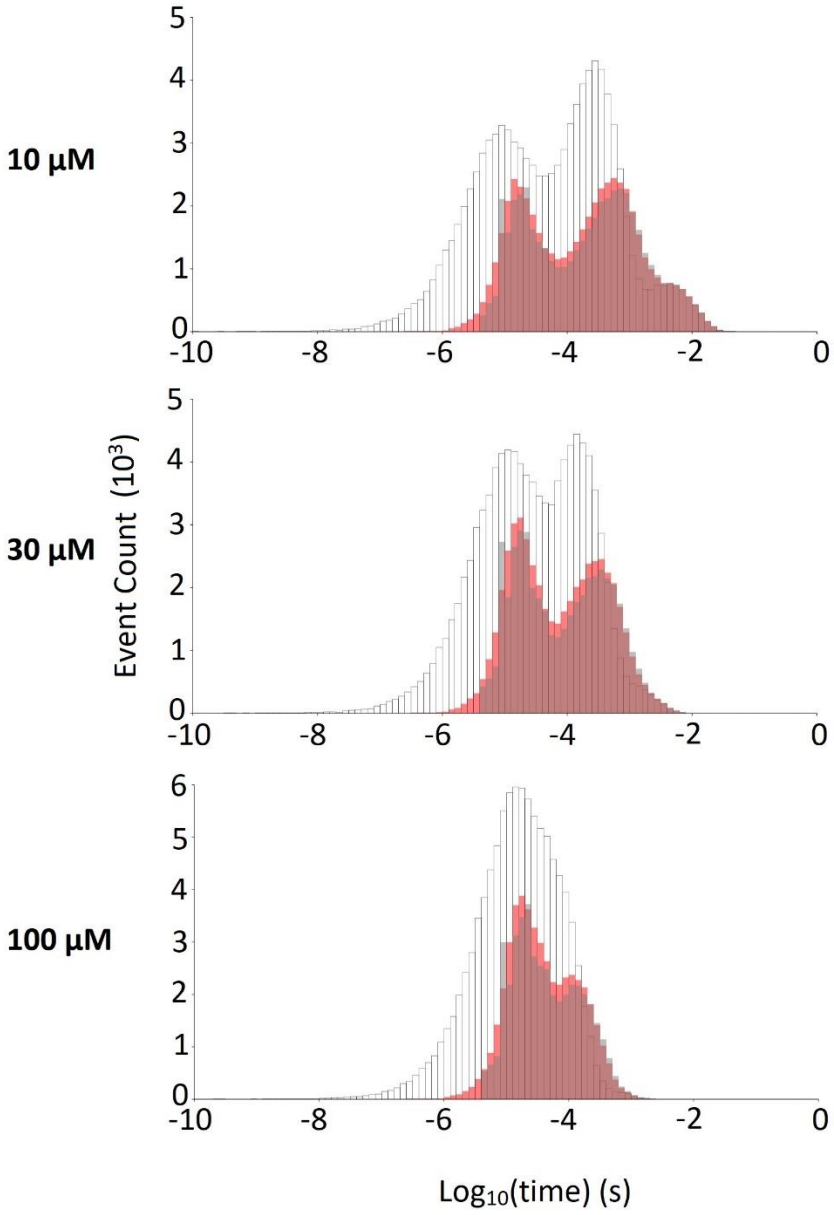

Figure 27. Dwell duration histograms for detected events. Dwell duration histograms comparing events detected from simulated events of the SPM at 10, 30, and 100 µM [ACh] with digital Gaussian filter cut-off frequency of 25 kHz. The white bins are the real dwell durations. The transparent red bins are dwell durations from Threshold Crossing detected events. The transparent gray bins are dwell durations from SKM-HMM detected events. The y-axis shows event counts. The x-axis display $\log_{10}$ values of time in seconds. Dwells were log-binned in 100 bins on an x-axis range from -11 to 1 log-units.

3.2.2 Missed and false events

Figure 28 summarizes the detection results visually in terms of percent of missed events, and percent of false events (conveniently plotted on the same axis.) These plots were made for the entire concentration range, but the trend across the range is clear by looking at select concentrations as in Figure 28. The exact numbers used to make the missed and false event plots are summarized in tables 14 to 17 in section 5.7 in the Appendix. The plots are made on an x-axis which depicts a range of S/N ratios. Within the simulated model, the S/N ratio x-axis generalizes the implication of the results to any single-channel regardless of its particular conductance level, or regardless of the size of the noise envelope. If we take a look at just one of the missed event plots (for any concentration) in isolation, we can see a trend across S/N ratios. As the S/N increases (due to increased filtering by lower cut-off frequencies,) the percent of missed events increases until it reaches what seems like a plateau. The distribution of percent of missed events across the range of S/N ratios looks like a rectangular hyperbola, however we were unable to fit the distribution to any particular function. Additionally, the percent of false events is highest at the lowest S/N ratio, and drops off rapidly to zero by a S/N of 10 (which corresponds roughly to a 35 kHz Gaussian cut-off).

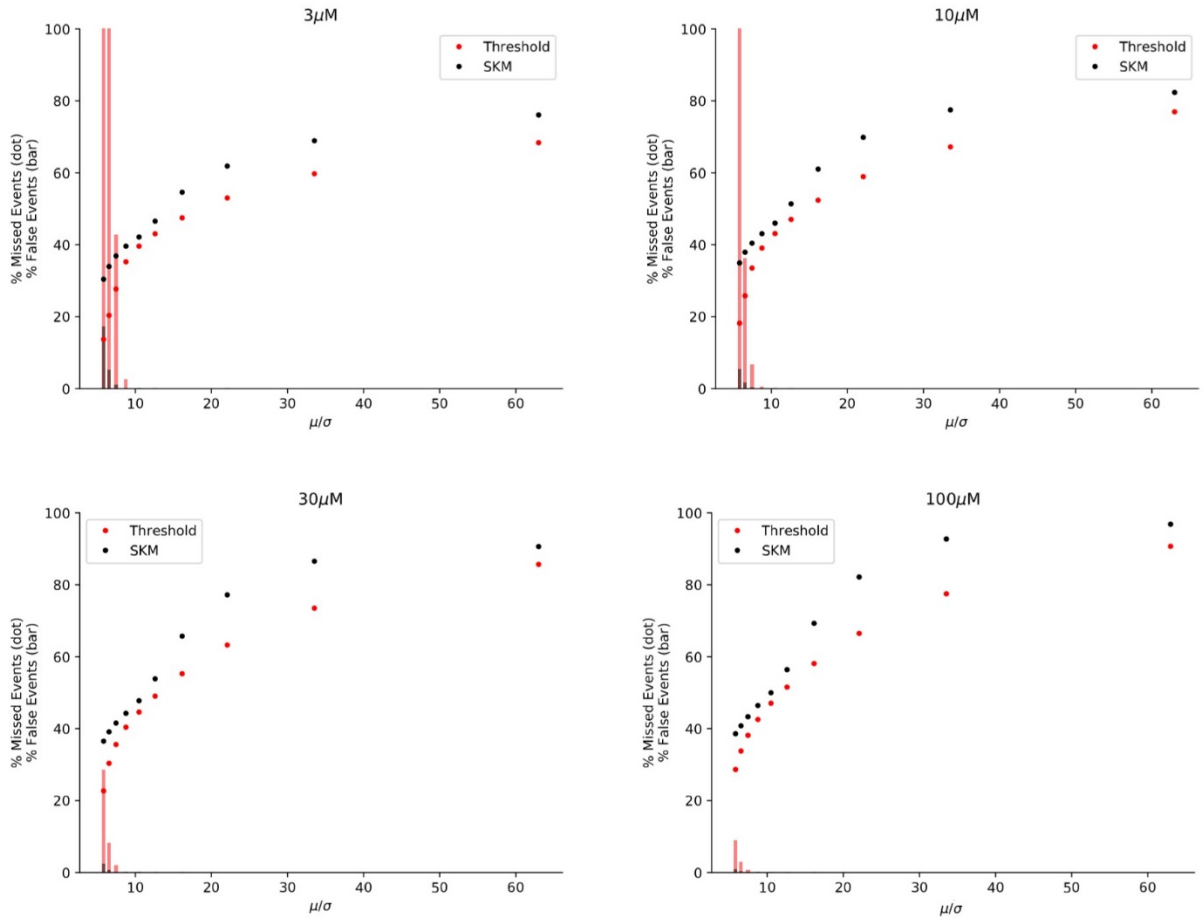

Figure 28. Missed and false events from Threshold Crossing and SKM-HMM. Plots depicting the % missed events and % false events resulting from detections from Threshold Crossing and SKM-HMM across a range of ligand concentration. The concentration of ACh is labelled at the top of each plot. The x-axis is the S/N ratio resulting from different digital Gaussian filter cut-off frequencies. The y-axis is on a percent scale. Dots represent percent missed events; bars represent percent false events. For details about the plots see section 2.2.6.

3.2.3 Accuracy of detected events

The accuracy of detected events can be summarized by accuracy plots as in Figure 29. The detected events plotted in the accuracy plots are true events. That is, the detected events of the accuracy plots correspond exactly with an event that was simulated. Additionally, there are no missed events underlying the detected events in the accuracy plots (for details see section 2.2.5). It is important to bear in mind this correspondence between detected and actual event when interpreting the accuracy plots. Out of all true detected events, those which had underlying missed events somewhere within the detected event were not considered for the accuracy plot. With this in mind, if we turn our attention to the accuracy plot for either SKM-HMM or Threshold Crossing at a cut-off frequency of 50 kHz, we can see a diffuse pattern of data points. This pattern of data points means that many true events that were detected by either method had poorly recovered durations. Thus, filtering less may increase the number of detected events, but at the cost of their accuracy. There seems to be a similar trend for both methods as the filter cut-off frequency gets smaller. Filtering more (lowering the cut-off frequency,) tends to reduce the spread across the accuracy plot. This could mean that events become more accurate as the filtering increases. Another interpretation is that the events which cause the detection algorithm to yield inaccurate durations get filtered out as the degree of filtering increases.

It is important to highlight that the number of events in the accuracy plots for SKM-HMM differ from the number of events for Threshold-Crossing. These numbers are summarized in the table below:

Cut-Off (kHz)	SKM-HMM	Threshold Crossing
50	424930	453419
25	281971	316683
5	24057	66552

Table 4. Numbers of true events used for accuracy plots across a range of digital Gaussian filter cut-off frequencies.

In every case, Threshold-Crossing has detected more true events than has SKM-HMM. This is an interesting observation considering both algorithms were submitted to the same data set. This means that Threshold Crossing has recovered more events that do not contain underlying missed events than SKM-HMM.

Another interesting observation, the effect of the resolution of the detection algorithm is reflected in the accuracy plot. Figure 30 shows a zoomed-in version of the 25 kHz cut-off accuracy plot from Figure 21. Recall that SKM-HMM assigns conductance states to sample points, thus the briefest event duration that SKM-HMM can recover is the sampling interval. The effect of SKM-HMM's resolution of event duration recovery is that many events will appear to have the same duration. Events which have the same duration are plotted on a horizontal line in the accuracy plot. The result of the limited resolution manifests itself in the accuracy plot as discrete increments of recovered event duration with gaps in between where there are no events (something that I will refer to as incrementation). In contrast to this, Threshold-Crossing's resolution with a 1 μs sampling interval is eight decimal places in units of seconds due to the cubic-spline interpolation of values between sample points. The result of Threshold-Crossing's resolution is the absence of incrementation in the accuracy plot in the observed range between 10^{-5} and 10^{-4} seconds. Incrementation does happen for Threshold-Crossing, but it is only apparent in the 10^{-8} to the 10^{-7} range, but events in this range are well beyond the dead time.

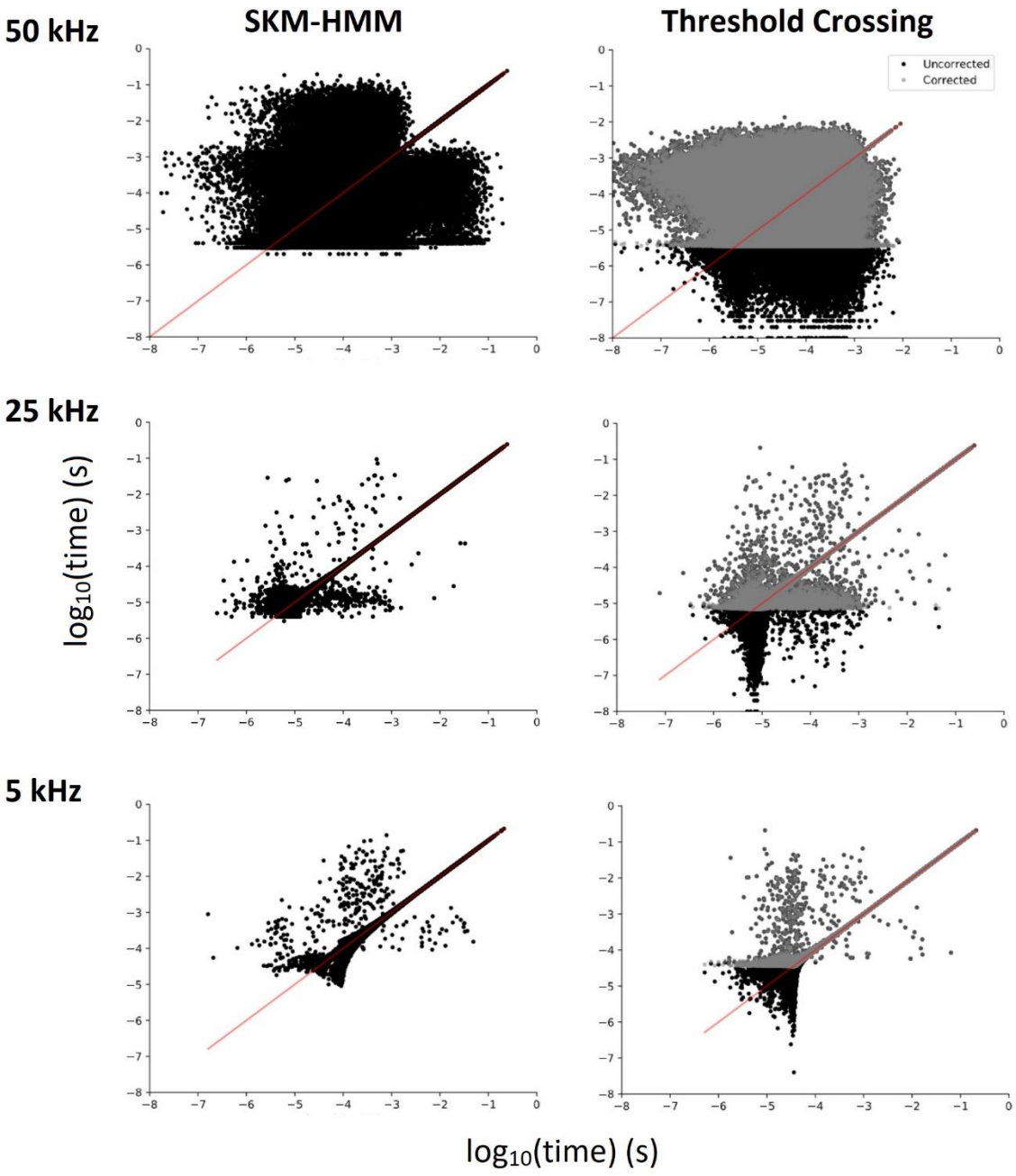

Figure 29. Accuracy plots of detected events from SKM-HMM and Threshold Crossing. The left column shows the accuracy plots for detected events from SKM-HMM. The right column shows the accuracy plots for detected events from Threshold Crossing. Each row shows the accuracy plots for a different digital Gaussian filter cut-off frequency (50, 25, 5 kHz respectively.) The plots are on a $\log_{10}$-$\log_{10}$ time scale in seconds. The central line is the measure of accuracy. For the Threshold Crossing plots, the gray dots represent corrected durations. For details about accuracy plots see section 2.2.5

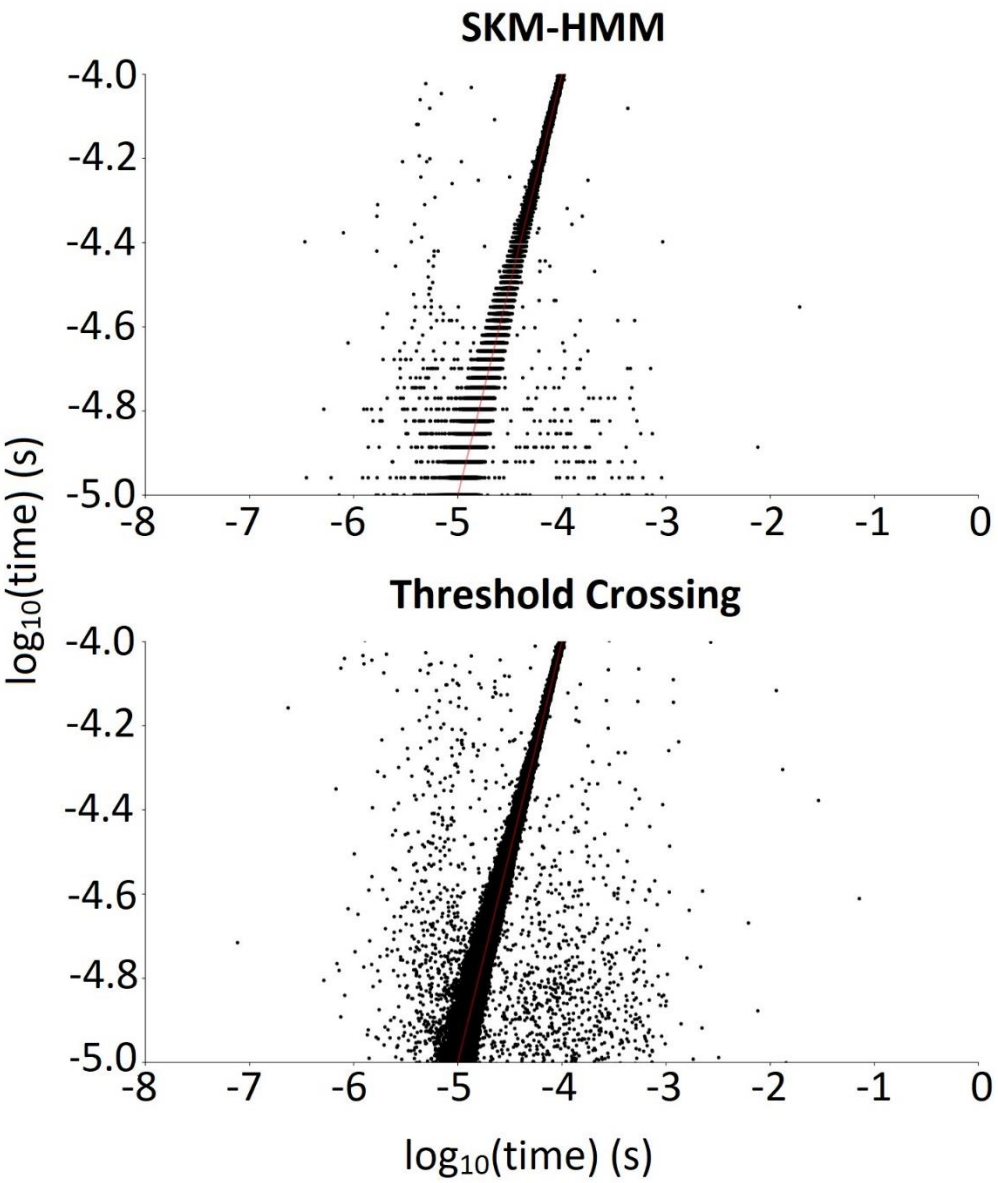

Figure 30. Accuracy plot comparing SKM-HMM and Threshold Crossing for events in the range of 10^{-4} to 10^{-5} seconds. The top plot displays the accuracy of events for SKM-HMM. The bottom plot displays accuracy of events for Threshold Crossing. Both plots display results for data filtered with a 25 kHz cut-off frequency for a digital Gaussian filter. The plots are on a $\log_{10}$-$\log_{10}$ time scale in seconds. The central line is the measure of accuracy. For the Threshold Crossing plots, the gray dots represent corrected durations. For details about accuracy plots see section 2.2.5

3.2.4 Recovery of kinetic parameters

For each filter cut-off frequency in the range that we simulated we attempted to recover kinetic parameters by fitting the observed sequence of events using MIL[46]. Global fitting was attempted using detected dwells from each concentration in the concentration range for the SPM[1]. It was not possible to recover kinetic parameters for every cut-off frequency, since the fitting algorithm would not converge on its likelihood criteria. Below is a table summarizing whether MIL recovered kinetic parameters for a given cut-off frequency:

Cut-Off (kHz)	SKM-HMM	Threshold-Crossing
5	NO	NO
10	NO	NO
15	NO	YES
20	NO	YES
25	NO	YES
30	YES	YES
35	YES	YES
40	YES	NO
45	YES	NO
50	YES	NO

Table 5. Convergence of MIL for SKM-HMM and Threshold-Crossing. For a range of digital Gaussian cut-off frequencies, the table displays if MIL was able to converge on its ML criteria.

Figure 31 shows plots depicting the accuracy of recovered kinetic parameters for either method at cut-offs of 30 and 35 kHz. The most apparent observation is that the accuracy of recovered kinetic parameters declines as the value of the rate constant gets larger for events detected by SKM-HMM. On the other hand, the accuracy for recovered kinetic parameters remains rather constant across the range of parameters for events detected by Threshold-Crossing. An interesting observation on the side of SKM-HMM is that recovered rate constants diverge from the central line in roughly the same timescale that we observed incrementation in the accuracy plots. Thus, it is plausible that the resolution of the detection algorithm plays an important role in the recovery of kinetic parameters. More generally, the better the resolution of the recovered event, the better MIL is capable of recovering faster rate constants (higher rate constant resolution.) This is important because faster rate constants contribute largely to the lifetime of brief events. If we assume that the resolution of the event duration is correlated to the resolution of the rate constant, then we should see divergence from the central line for rate constants in the range of 10^7 to 10^8 s^{-1} for Threshold Crossing (corresponding to the incrementation seen in the 10^{-7} to 10^{-8} s range in the accuracy plot).

Figure 32 shows the accuracy of recovered kinetic parameters using the *a priori* simulated dwell durations for the entire concentration range. As expected, the parameters are recovered with negligible discrepancy from the actual parameter.

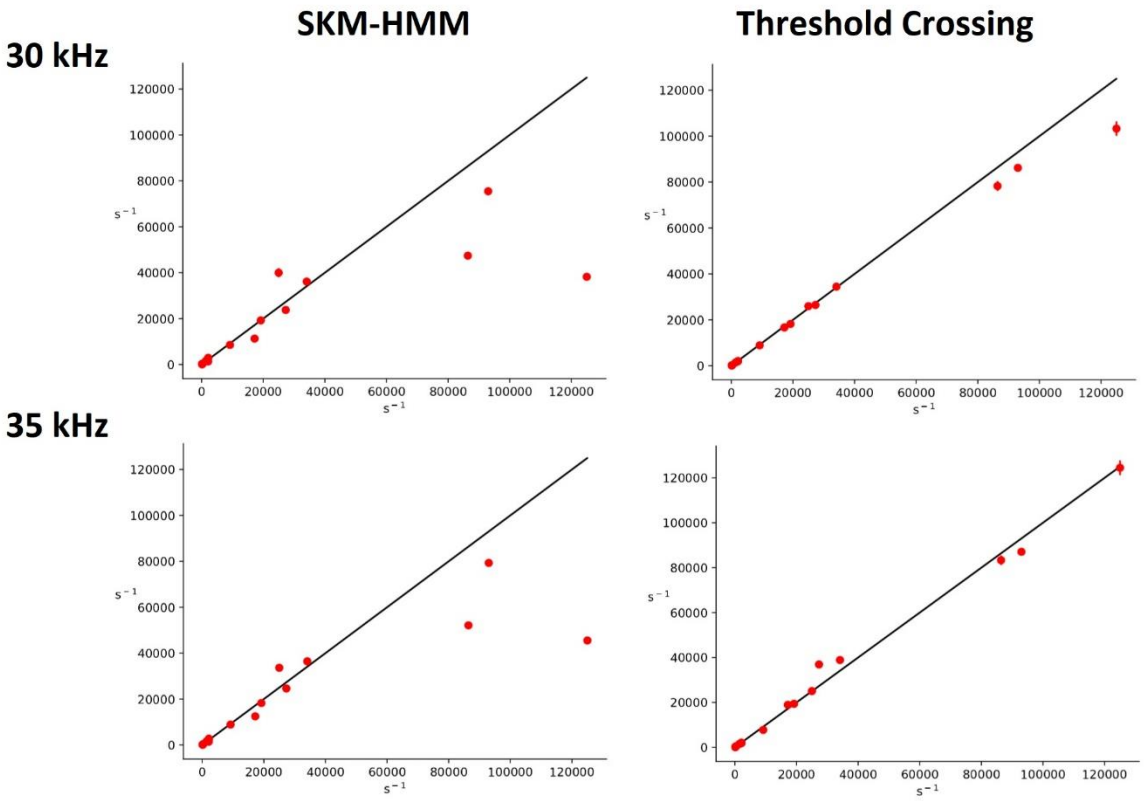

Figure 31. Kinetic parameter plots from detected events. Events detected by SKM-HMM or Threshold Crossing were submitted to fitting kinetic parameters in MIL. These plots show the results of the kinetic fits for events detected at two digital Gaussian cut-off frequencies (30 and 35 kHz.) Both axes are on a linear scale and represent rate constant values in units of s^{-1}. The central line is the measure of accuracy. The dots are [x, y] coordinates representing [actual, recovered] kinetic parameter. For details about the plots see section 2.2.8.

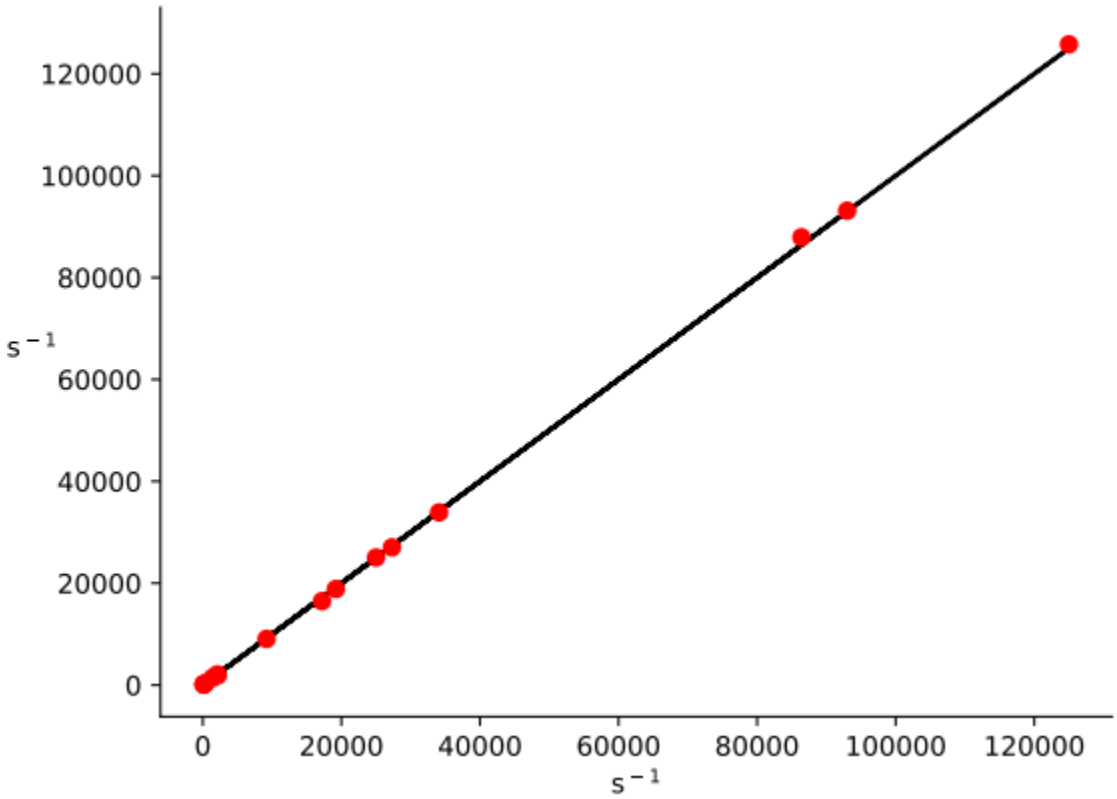

Figure 32. Kinetic parameter plot from *a priori* exact event durations. Simulated dwells used for the test data set of the preliminary comparison were submitted to kinetic fitting by MIL. Both axes are on a linear scale and represent rate constant values in units of s^{-1}. The central line is the measure of accuracy. The dots are [x, y] coordinates representing [actual, recovered] kinetic parameter. For details about the plots see section 2.2.8

4 DISCUSSION

4.1 Simulator

4.1.1 HMM engine

The HMM engine simulates large numbers of events in a very short amount of time. On a laptop with 8 GB of RAM, the simulator can generate 10^9 events within a few minutes. The speed of the simulator is due to pre-calculation of lifetime and transition distributions (i.e. the marble bags). The speed of the simulator is convenient for generating large data sets, which are useful for statistical analysis.

We have demonstrated that the HMM engine can simulate events whose duration histogram approximates the expected SPD within 1 % difference. This has been demonstrated for a model whose states are all connected, and for a physiologically relevant 8-state model. However, the HMM engine can simulate events that will approximate the SPD for any model.

We have demonstrated that the HMM engine does not violate the allowable transitions of the transition rate matrix. The HMM engine will simulate events by making stochastic transitions in accordance with the transition rate matrix.

We have demonstrated that the HMM engine will produce events whose conversion into rectangular pulses (dwells) leads to the approximation of the CPD. Thus, by simulating state events with lifetimes defined by the SPD, the simulator is capable of producing dwells with lifetimes defined by the CPD.

It is gratifying to see that the HMM engine can recover the steady-state occupancy of states simply by making transitions within the transition rate matrix. That is, the HMM engine's algorithm does not explicitly take into account the steady-state occupancies. It is completely blind to the steady-state occupancies. The marble-bag distribution of lifetimes for any state differs only by the mean lifetime, they are otherwise the same in shape and size. It is only through making transitions within the matrix that the distribution for a particular state gets naturally scaled according to its steady-state occupancy. The natural scaling back to the steady-state occupancy happens by the fact that the size of the duration histogram for a state is determined by how many times that state is visited (and not by how much time the channel remains in that state). How many times a state is visited is implicit in the

transition rate matrix. As any column in the matrix defines all rates leading into a state, integrating the values in the column over a finite period of time yields the amount of transitions into that state seen in that finite period of time. In accordance with the recovery of steady-state occupancy for states, the conversion of the state duration histogram into the dwell duration histogram recovers the steady-state occupancy of dwells (thereby approximating the CPD.)

Thus, not only is the math of the system valid, but the stochastic process is also valid. By following a few simple rules about making transitions within a matrix (defined in 1.5) the engine is capable of simulating data which approximate the probability densities that defines the lifetimes of the states in the model.

4.1.2 Sampling engine

We have demonstrated that the sampling engine is capable of matching the step-response of the patch-clamp, which includes an RC component, and a Bessel filter component. The fitting algorithm is relatively quick to fit the parameters of the patch-clamp step-response. The RMSE of 0.00152 is relatively low considering the range of values in the step-response (between 0 and 1). Additionally, the overlay of the fit onto the measured step-response from Figure 19 gives us confidence in our fitting algorithm. If the starting boundaries are too far from the smallest RMSE, then the algorithm will take a while before it converges (or it may not converge at all). The results of the fit empirically yield the lowest RMSE value resulting from a combination of parameters. The resulting fit parameters (such as RC time constant and Bessel cut-off frequency,) may not be the real parameters of the physical system. The algorithm simply yields the combination of parameters which minimizes the difference between the simulated step-response and the real step-response. Though we have not demonstrated it explicitly in this research, our algorithm finds a much better match to the real step-response than a simple least-squares fit to a sigmoidal function. Additionally, fitting with both the RC component and the Bessel filter yields a much better fit than either just the RC component or the Bessel filter alone.

The Bessel filter cut-off is in the 100-200 kHz range. The Axopatch 200B analog Bessel filter is not exactly 100 kHz so it is possible that the real cut-off of the system is closer to 200 kHz. Moreover, there could be some discrepancy between the analog frequency response, and the digital approximation of the frequency response used by the implemented filter. However, the actual number of the cut-off frequency used by the digital filter does not matter so long as it gives us a simulated step-response which has an empirically minimized error with the real step-response.

We plan on testing the fitting algorithm on measured step-responses with noise in the future. If we are capable of accurately fitting the parameters of a step-response given noise, then this fitting algorithm could be a precursor to a time-course fitting algorithm. However, this requires more testing with simulations and noise. For now, we are confident in our algorithm's ability to recover step-response parameters for a step-response that does not contain appreciable noise (which can be acquired from the Axopatch 200B through averaging several thousand synchronized individual step responses).

We have demonstrated that the sampling engine is capable of matching real patch-clamp noise by modeling the noise as a random AR process. The PSD of figure 20 gives us confidence that our simulated noise is appropriately modeled as it contains the frequency characteristics of real noise. The AR model is versatile because it allows us to model noise from any system (including nanopore and single-molecule FRET). The AR modeling of noise is quite easy using Spectrum[66] with the well-established Levinson-Durbin recursion algorithm.

Converting the RC time constant of the step-response to units of the sampling interval, as well as putting the x-axis in units of the sampling interval grants the algorithm compatibility with any sampling interval. Moreover, the units of the sampling interval allow the algorithm to simulate step-responses beginning at any point in time with simple calculations. Any transition that occurs between a sample point simply needs to have the values of its x-axis bumped up by a value between 0 and 1. The calculation is the same for any sampling interval, as the x-axis and the RC constant is normalized to the sampling interval.

The speed of simulating successive step-responses depends on the size of the RAM of the system. If the x-space for every transition in a given sequence of events can be represented in RAM all at once, then the algorithm is very fast (it will sample 10 seconds of data at 1 µs sampling interval in a few seconds of real time.) However, if the x-space for every transition cannot be represented in RAM all at once, then the algorithm will automatically parse the x-space into smaller blocks so that the computer can process these blocks one by one. If parsing occurs, then the speed of the algorithm is dependent only on the number of blocks (~ a few seconds per block.) Parsing will most likely occur for data with slow kinetics, where there is a lot of time between transitions. The more time between transitions, the more sample points need to be represented in memory.

4.1.3 General discussion of the simulator

As mentioned in section 1.2.4 the simulator has a modular design, which allows any user to simulate data sets with a variety of conditions. The variability in step-response matching and noise modeling makes the simulator applicable to virtually any system. Moreover, the ability for the simulator to simulate single-molecules with subconductance states makes the simulator applicable to single-molecule fields like nanopore or single-molecule FRET experiments.

The realism of the simulator grants validity to quantification and comparison of methods in single-molecule data analysis. Thus, the simulator could be used to quantify and compare existing methods (as was done for this present research.) Additionally, one could use the simulator to aid with the development of new analytic methods in single-molecule analysis by providing test data sets with suitable testing conditions and *a priori* information. Moreover, because of the stochastic HMM engine, the simulator can be used to test hypotheses about single-molecule kinetics, but also test the validity of newly proposed kinetic models with any number of states.

The high resolution of the simulator makes it suitable to address the problem of brief events in single-channel kinetics. The simulator simulates kinetic events with a variable time resolution of 32-bits or 64-bits (user defined). Moreover, since the lifetime distributions are sampled on a $\log_{10}$ scale the resolution of event durations is maintained across a wide range of decades. Additionally, the sampling algorithm is capable of simulating patch-clamp step-responses for step changes in current that occur at any point in time. Transitions rarely, if at all, occur exactly on a sample point. By adjusting the x-axis on which the step-response is evaluated, we are capable of generating a sampled step-response whose transitions occurred anywhere between two sample points. Thus, the resolution time of the transition is preserved in sampling the data as it is in simulating the event from a lifetime distribution.

What remains for the simulator is the development of a user-friendly graphical interface such that anyone can use the simulator regardless of their level of knowledge of computers.

The design of the simulator is variable not only on the surface, but in the structure of the underlying code. The simulator was written in Python with the intention to be modular not only in usage, but also in expansion (i.e. extensible). Any savvy programmer can extend the application by

following the structure of the program's code. Thus, if there are additional functions desired, they can easily be implemented without disrupting the core of the simulator.

4.2 Preliminary comparison of event detection methods

4.2.1 Missed and false events

The general trend for missed events for either detection method is that filtering more causes the method to miss a greater percentage of events. As we filter less, both detection algorithms miss fewer events. There is a tradeoff however, since filtering less allows the noise envelope to cause false events to be detected. Threshold Crossing is more sensitive to the detection of false events than is SKM-HMM. All it takes for a false event to be detected by Threshold Crossing is for the noise to cross the threshold. Thus, any brief spike caused by high-frequency noise can be detected as a brief event by Threshold-Crossing. On the other hand, SKM-HMM seems to be resilient to the introduction of false events. It could be that because SKM-HMM takes explicit account of the noise by fitting amplitude histograms, it is better capable of discerning real from false events. We initially thought that because SKM-HMM did not introduce many false events we could potentially gain more information by detecting with SKM-HMM with a 45 or a 50 kHz Gaussian filter. Unfortunately for SKM-HMM, the accuracy and resolution of the events play an important role in fitting kinetic parameters with fast rate constants. SKM-HMM may provide us with more information (10-20 % more events with a 45 kHz Gaussian,) but the accuracy and resolution of their recovered durations is not good enough to fit rate constants which define the lifetime of brief events.

Looking across the concentration range, as the concentration of ACh increases, the percent of false events declines appreciably. False events occur because the noise envelope is so large that some noise peaks are detected as events. When a channel transitions between states rapidly, stretches of baseline or open channel currents are short. Thus, long stretches of baseline, or open channel currents on which there exists substantial noise become less and less over an increasing ligand concentration range, which results in fewer opportunities for the detection algorithm to detect a false event. As the concentration range increases the percent of missed events increases regardless of the S/N ratio. This result is explained as increasing the concentration of ACh increases the proportion of brief events over long events.

Comparison of the two detection methods across the full range of S/N ratios, reveals that Threshold-crossing has a lower percentage of missed events (evident by observing the plots from Figure 28.) This is the case for the entire tested concentration range.

There appears to be a window of opportunity for improvement for Threshold Crossing by optimizing the S/N ratio. The most state-of-the-art report today used a 25 kHz Gaussian filter, and reported a deadtime of 8 μs[1]. Using a 35 kHz Gaussian filter would yield a deadtime of roughly 5 μs. For example, at 25 kHz, the percent of missed events for the SPM at 30 μM [ACh] is 49.039 %, yet at 35 kHz, the percent of missed events is 40.405 %. This is a nearly 10 % increase in number of events which could be detected at 35 kHz. The percent of false events introduced at 25 kHz for the SPM at 30 μM [ACh] is 0.000 %, while at 35 kHz it is only 0.248 %. We believe that a small introduction of false events is negligible when we compare this to the number of real events being detected. Moreover, the kinetic fit (i.e. resulting rate constants) for events detected at 35 kHz by Threshold Crossing are superior to those detected at 25 kHz, as the derived rates hardly diverge from the central line in the plot in Figure 31.

4.2.2 Resolution and accuracy of detected events

The general trend for either detection method is that they detect more inaccurate events the less we filter the data. When filtered less, the noise can introduce considerable error in recovering the true duration of an event. Filtering more nullifies the effect that the noise has on the accuracy of event duration recovery. However, the downside of filtering more is that the briefest events disappear altogether.

The incrementation of event durations observed in the accuracy plot for SKM-HMM could explain the deviation from the central line for faster rate constants in the kinetic parameter plot. However, this must be explored in more detail to confirm this hypothesis. If the resolution of the event plays an important role on our ability to fit fast rate constants, then it may suggest that methods which provide a greater resolution of event duration are superior. If there is a clear predictable correlation between the resolution of the event duration and the resolution of the fit rate constant, then we may be able to use this relationship to our advantage to further characterize existing event detection methods. Event detection methods may have an inherent limitation on the fastest rate constant that can be reliably fit from recovered events. This motivates us to explore the resolution of Time-Course Fitting, which could be on-par or superior to Threshold-Crossing.

4.2.3 Recovered kinetic parameters

The negligible deviation of recovered versus actual kinetic parameter displayed in Figure 32 supports the conclusion that our dwells were properly simulated according to our model.

The recovered kinetic parameters from events detected by SKM-HMM showed the same deviation from the central line for faster rate constants regardless of the Gaussian filter frequency. This is consistent with the idea that the resolution of the event duration has an effect on the resolution of the rate constant that can be fit. SKM-HMM's event duration resolution is dependent on the sampling interval, and is independent of filter cut-off frequency. As the event duration resolution is independent of the filter frequency, so is the resolution on the rate constant that can be fit by events detected from SKM-HMM.

The recovered kinetic parameters from events detected by Threshold-Crossing deviate from the central line much less when compared to events detected by SKM-HMM. In general, the faster parameters tend to be underestimated for both methods. However, the underestimation is appreciably less for Threshold-Crossing.

It is difficult to explain the deviation from the central line with the information that we have drawn from our analysis. The most parsimonious explanation is that this deviation is related to the resolution of detected event durations. However, we cannot say for certain that resolution either by itself, or in conjunction with some other factors ultimately causes underestimation of kinetic parameters. We would need to perform more analysis in order to find the root cause of the deviation.

The propensity of a detection method to recover kinetic parameters seems to be a good measure of performance for the detection method (at least for the purpose of studying ion channel mechanisms with established kinetic parameters and models.) In the end, we are looking for more reliable fits of the data to kinetic models of increasing complexity. Albeit, recovering kinetic parameters as a measure of performance does not by any means inform us on the reason why the method is performing the way it is. However, it does provide us with a simple measure on the limitation of the detection method. The recovered kinetic parameter plots essentially tell us the range of kinetic parameter values that can be reliably fit from events detected by a particular method.

4.3 Future work

The modularity of the HMM engine can be improved by including the possibility to simulate events from distributions other than the SPD (for example, Gaussian distributed events). The addition of other possible distributions would make the simulator applicable to a wider variety of single-molecule processes.

The custom step-response fitting algorithm can be further improved. The algorithm fails to converge on a unique solution if the starting boundaries are not appropriately defined. Additionally, the algorithm can be refined to improve its efficiency. In addition, currently the fitting algorithm does not perform well when there is noise in the signal. If the algorithm could reliably and efficiently fit individual step-responses with noise, it could represent a pre-cursor to a new time-course fitting algorithm.

The simulator is lacking a graphical user interface. We plan on designing and implementing a user-friendly graphical interface which would facilitate the use of the simulator without any knowledge of the Python programming language. Ultimately, our intention is to make the simulator publicly available, and accessible to anyone doing single-molecule experiments.

The preliminary comparison of SKM-HMM and Threshold Crossing will be extended to include Time-Course fitting. We also plan on expanding our test conditions to include a range of sampling intervals.

Our initial quantification of Threshold Crossing has revealed the possibility of optimizing the S/N (by simply adjusting the digital Gaussian cut-off frequency) to improve the dead-time in our analysis of real data. The quantification suggests that it may be possible to improve the dead-time by filtering at ~35 kHz rather than at 25 kHz. This observation provides further motivation to refine our analysis to determine the optimal filter setting for data analysis, thereby maximizing the biological insight from our precious single-molecule data.

4.4 Summary

We have designed and implemented a modular single-channel data simulator containing an HMM engine and a sampling engine.

The simulator is capable of simulating single-channel gating activity from a defined kinetic model. The distribution of simulated events approximates the expected SPD, and the distribution of simulated dwells approximates the expected CPD.

The simulator can take any sequence of events or dwells to produce a sampled realistic noisy trace of data. The simulator has the capability of matching any real patch-clamp step response and noise. Additionally, the simulator can simulate with any number of subconductance levels (Figure 35). The simulator can also simulate a single trace of data containing sequences of dwells from an indefinite number of individual ion channels (Figure 34).

We have used our simulator to compare two event detection methods: SKM-HMM and Threshold Crossing. We have designed and simulated a test data set comprising a number of testable conditions. The test data set contains simulated WT muscle-type nAChR activity for a wide agonist concentration range (3-300 μM [ACh], Figure 33), which provides our comparison with a variety of kinetic events from the briefest to the longest events seen in real state-of-the-art data. The test data is sampled at a 1 μs sampling interval, with realistic patch-clamp noise, and a range of digital Gaussian filter cut-off frequencies (5-50 kHz). Moreover, each element of the test data set contains 100 000 events which is 10 times more than in typical real data.

Our quantification of SKM-HMM and Threshold Crossing reveals that both methods miss roughly 50 % of the events that were simulated, but that the proportion of missed events depends ultimately on the degree of filtering. For any given filter setting, Threshold Crossing misses fewer events than does SKM-HMM. Threshold Crossing however introduces more false events than does SKM-HMM. The accuracy of events is comparable for SKM-HMM and Threshold Crossing. The time resolution of detected events may affect the ability for MIL to recover fast kinetic parameters. Events detected by Threshold Crossing yield better accuracy for faster kinetic parameters than events detected by SKM-HMM.

5 APPENDIX

5.1 Models used for simulation

(1) SPM from 3 to 300 µM [ACh]:

<%s3uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-405,405,0,0,0,0,0,0}

{TRM:1410,-2137,402,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-49750,6450,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-2550,450}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s6uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-810,810,0,0,0,0,0,0}

{TRM:1410,-2539,804,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-56200,12900,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-3000,900}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s10uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-1350,1350,0,0,0,0,0,0}

{TRM:1410,-3075,1340,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-64800,21500,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-3600,1500}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s18uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-2430,2430,0,0,0,0,0,0}

{TRM:1410,-4147,2412,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-82000,38700,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-4800,2700}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s30uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-4050,4050,0,0,0,0,0,0}

{TRM:1410,-5755,4020,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-107800,64500,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-6600,4500}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s60uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-8100,8100,0,0,0,0,0,0}

{TRM:1410,-9775,8040,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-172300,129000,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-11100,9000}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s100uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-13500,13500,0,0,0,0,0,0}

{TRM:1410,-15135,13400,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-258300,215000,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-17100,15000}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s180uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-24300,24300,0,0,0,0,0,0}

{TRM:1410,-25855,24120,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-430300,387000,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-29100,27000}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

<%s300uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-40500,40500,0,0,0,0,0,0}

{TRM:1410,-41935,40200,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-688300,645000,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-47100,45000}

{TRM:0,0,0,0,0,0,93000,-93000}

<%%>

(2) All-connected with rate constants at 100 s^{-1}:

<%sCuM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-700,100,100,100,100,100,100,100}

{TRM:100,-700,100,100,100,100,100,100}

{TRM:100,100,-700,100,100,100,100,100}

{TRM:100,100,100,-700,100,100,100,100}

{TRM:100,100,100,100,-700,100,100,100}

{TRM:100,100,100,100,100,-700,100,100}

{TRM:100,100,100,100,100,100,-700,100}

{TRM:100,100,100,100,100,100,100,-700}

<%%>

(3) SPM at 3 µM [ACh] with one rate fixed at 1 s^{-1}:

<%s3uM_model%>

{States:0,0,0,0,0,1,1,0}

{TRM:-405,405,0,0,0,0,0,0}

{TRM:1410,-2137,402,325,0,0,0,0}

{TRM:0,25000,-43400,0,18400,0,0,0}

{TRM:0,34100,0,-49750,6450,9200,0,0}

{TRM:0,0,86400,17200,-228600,0,125000,0}

{TRM:0,0,0,27300,0,-27300,0,0}

{TRM:0,0,0,0,2100,0,-2550,450}

{TRM:0,0,0,0,0,0,1,-1}

<%%>

(4) del Castillo & Katz from 3 to 300 µM [ACh] used for SKM-HMM detection:

<%dck3uM_model%>

{States:0,0,0,1,1,0}

{TRM:-813,813,0,0,0,0}

{TRM:7190,-7833,576,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-1728,78}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck6uM_model%>

{States:0,0,0,1,1,0}

{TRM:-1626,1626,0,0,0,0}

{TRM:7190,-8409,1152,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-1806,156}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck10uM_model%>

{States:0,0,0,1,1,0}

{TRM:-2710,2710,0,0,0,0}

{TRM:7190,-9177,1920,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-1910,260}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck18uM_model%>

{States:0,0,0,1,1,0}

{TRM:-4878,4878,0,0,0,0}

{TRM:7190,-10713,3456,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-2118,468}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck30uM_model%>

{States:0,0,0,1,1,0}

{TRM:-8130,8130,0,0,0,0}

{TRM:7190,-13017,5760,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-2430,780}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck60uM_model%>

{States:0,0,0,1,1,0}

{TRM:-16260,16260,0,0,0,0}

{TRM:7190,-18777,11520,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-3210,1560}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck100uM_model%>

{States:0,0,0,1,1,0}

{TRM:-27100,27100,0,0,0,0}

{TRM:7190,-26457,19200,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-4250,2600}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck180uM_model%>

{States:0,0,0,1,1,0}

{TRM:-48780,48780,0,0,0,0}

{TRM:7190,-41817,34560,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-6330,4680}

{TRM:0,0,0,0,113000,-113000}

<%%>

<%dck300uM_model%>

{States:0,0,0,1,1,0}

{TRM:-81300,81300,0,0,0,0}

{TRM:7190,-64857,57600,67,0,0}

{TRM:0,19300,-63000,0,43700,0}

{TRM:0,2190,0,-2190,0,0}

{TRM:0,0,1650,0,-9450,7800}

{TRM:0,0,0,0,113000,-113000}

<%%>

5.2 AR coefficients (co) and variance (va) of noise modeled for the preliminary comparison of methods:

Coefficients are displayed in a Numpy array for convenience.

va = 1.4986026891559968e-23

co = np.array([-1.48132761e+00, 1.72791064e+00, -1.37279044e+00, 1.23429416e+00,
       -8.89468428e-01, 7.95441402e-01, -4.73049208e-01, 3.83375349e-01,
       -1.85070180e-01, 1.79476580e-01, -3.15431131e-02, 7.81488497e-02,
        1.30551403e-02, 2.74467626e-02, 6.46777345e-02, 1.35519039e-02,
        3.57745148e-02, 7.86124970e-03, 3.94414226e-02, 2.89789357e-02,
        2.27229991e-02, 2.87597479e-02, 8.66094827e-03, 3.70411492e-02,
        1.91174313e-02, 3.71343933e-02, -1.05172765e-02, 2.74910642e-02,
        2.34774043e-02, 1.19914969e-02, 2.26193159e-02, 8.23905015e-03,
        2.06153798e-02, 2.20217103e-02, 1.06045002e-02, 2.51958730e-02,
       -8.36331085e-03, 2.21554039e-02, 1.67107475e-02, 1.63234819e-02,
        3.87241240e-03, 1.74422102e-02, 1.74247251e-02, 1.21054283e-02,
        1.29179671e-02, 2.62326406e-03, 1.19609705e-02, 1.52103910e-02,
        6.42337274e-03, 1.82369280e-02, 6.40390993e-03, 5.56390466e-03,
        2.62403335e-02, -7.94255767e-04, 9.73989167e-03, -9.10209233e-04,
        1.80449804e-02, 8.81960920e-03, -5.51288674e-03, 2.66141588e-02,
        2.49452355e-03, 9.98460541e-03, 4.33289011e-03, 2.06501408e-02,
       -1.88314573e-02, 2.72759917e-02, -9.80525585e-03, 2.61657646e-02,
       -1.25205382e-02, 2.18934778e-02, 4.61175857e-03, 7.67350006e-03,
        4.02230172e-03, 3.06054349e-03, 1.22180778e-02, -1.74466605e-02,
        2.79332688e-02, -6.69606744e-04, -6.62865075e-03, 2.18901118e-02,

-6.95401763e-03, 1.54046985e-02, -1.15764424e-02, 1.61909867e-02,

-2.06300335e-03, 5.86845332e-03, 4.92822320e-03, 9.33694061e-03,

 1.36842263e-03, 2.81005245e-03, 6.44692159e-03, 2.50512893e-03,

 2.87416400e-03, 9.99261979e-04, -4.07003561e-03, 2.06949265e-02,

-2.48869151e-02, 3.97864197e-02, -3.12114106e-02, 3.40627427e-02,

-2.71552085e-02, 3.47920728e-02, -2.47669508e-02, 2.28379912e-02,

-9.02542698e-03, 1.15742012e-02, 4.29620971e-04, 3.33645917e-03,

 8.99461702e-03, -3.64718418e-03, 5.41663485e-03, -2.36475882e-03,

 1.49727941e-02, -1.22941717e-02, 1.09033448e-02, 8.36759202e-03,

 1.96622346e-04, -6.75148670e-04, 3.15213688e-03, 8.57385466e-03,

-8.79829398e-03, 5.39476501e-03, 3.57362439e-03, -2.83432003e-03,

 2.40427147e-04, 1.50549769e-02, -6.68343066e-03, 4.41790750e-03,

 5.22319069e-03, -4.18782120e-03, 2.16426151e-02, -2.75999998e-02,

 3.17741458e-02, -1.52746613e-02, 1.30679184e-02, -8.97510738e-03,

 1.96606126e-02, -1.11446681e-02, 5.01920750e-03, 1.30877571e-02,

-1.36742443e-02, 1.52919954e-02, -1.13905338e-02, 2.92658680e-02,

-2.55377388e-02, 2.51694958e-02, -8.79452216e-03, 1.70047510e-02,

-2.03042650e-02, 2.69417520e-02, -1.51199119e-02, 2.47176936e-02,

-3.66292425e-02, 4.85155116e-02, -3.99104983e-02, 4.12004857e-02,

-3.83242187e-02, 3.77060374e-02, -2.61896716e-02, 2.14219194e-02,

-1.25413507e-02, 1.36222692e-02, -9.88579102e-03, 1.81929318e-02,

-7.48342685e-03, 1.50096989e-02, -1.18075217e-02, 1.88025506e-02,

-1.03547852e-02, 1.16338754e-02, -1.09362109e-02, 2.09453211e-02,

-2.02823307e-02, 1.67148460e-02, -4.11768919e-03, 9.94783416e-03,

-1.63078831e-02, 2.75518432e-02, -1.95947241e-02, 1.67222726e-02,

-1.43709652e-02, 2.28429396e-02, -1.82303721e-02, 2.20649821e-02,

-1.81937003e-02, 2.90117521e-02, -2.68000746e-02, 2.31249888e-02,

-4.13425638e-03, 1.85164108e-03, -5.69605168e-04, 2.25536432e-03,

 7.19286447e-03, -9.64124721e-03, 5.35927252e-03, 7.17011556e-03,

-7.37466756e-03, 1.26744539e-02, -1.71285946e-02, 2.38250612e-02,

-2.14949886e-02, 1.51241944e-02, -1.07137533e-03, -1.28679941e-03,

-1.07331236e-03, 9.17586922e-03, -1.98503617e-03, 4.18548935e-03,

-1.19302745e-03, 5.11918059e-03, 2.44608226e-03, 3.33707426e-03,

-4.76896176e-04, 6.12678574e-03, -3.06990202e-03, 3.38994339e-03,

 6.28377709e-03, -5.58561818e-03, 9.51674757e-04, 3.21158338e-03,

-3.64296771e-03, 1.56634934e-04, -1.90771395e-03, 2.67010483e-03,

-7.42284014e-03, 7.35071660e-03, -1.04037423e-02, 9.87432974e-03,

-8.76913191e-03, 3.71154712e-03, -5.35979342e-03, 1.12378024e-02,

-1.86031082e-02, 1.83665729e-02, -1.33350565e-02, 4.71832548e-03,

-3.75520515e-03, 1.05009348e-02, -1.02712777e-02, 5.43993158e-03,

-5.37208709e-04, -1.55951548e-04, -4.23563862e-03, 6.55247451e-03,

-5.59894565e-03, 4.03351832e-03, -5.47388396e-03, 5.08533845e-03,

-5.30429857e-03, -2.78477431e-04])

5.3 S/N ratio used for the x-axis of missed and false event plots

Cut-Off (kHz)	Baseline (A)	S/N
5	1.95E-13	63.00
10	3.66E-13	33.53
15	5.56E-13	22.08
20	7.60E-13	16.15
25	9.74E-13	12.60
30	1.17E-12	10.49
35	1.40E-12	8.77
40	1.64E-12	7.47
45	1.87E-12	6.55
50	2.10E-12	5.83

Table 6. S/N ratio used for the x-axis of missed and false event plots.

5.4 Dead-times used for kinetic fitting in MIL from events detected by SKM-HMM

Cut-Off (kHz)	3	5	7	10	12	15	20
5	NO	NO	NO	NO	NO	NO	NO
10	NO	NO	NO	NO	NO	NO	NO
15	NO	NO	NO	NO	NO	NO	NO
20	NO	NO	NO	NO	NO	NO	NO
25	NO	NO	NO	NO	NO	NO	NO
30	NO	NO	YES	YES	NO	NO	NO
35	NO	YES	YES	YES	NO	NO	NO
40	NO	YES	YES	YES	YES	NO	NO
45	YES	YES	YES	YES	NO	NO	NO
50	YES	YES	YES	YES	NO	NO	NO

Table 7. Dead-times used for kinetic fitting in MIL from events detected by SKM-HMM. A yes indicates that the MIL algorithm was able to converge, a no indicates that the algorithm was not able to converge. The dead-times are in units of μs (3, 5, 7, 10, 12, 15 and 20 μs).

5.5 Results from the transition sequence test

(1) SPM at 3 μM [ACh]:

10 events:

[[-4. 4. 0. 0. 0. 0. 0. 0.]

[3. -4. 0. 0. 1. FN 0. 0.]

[0. 0. -0. 0. 0. 0. FN FN]

[0. 0. 0. -0. 0. FN 0. 0.]

[0. 1. 0. 0. -1. 0. 0. FN]

[0. FN 0. FN 0. -0. 0. FN]

[0. 0. FN 0. 0. 0. -0. 0.]

[0. 0. FN 0. FN FN 0. -0.]]

100 events:

[[-17. 17. 0. 0. 0. 0. 0. 0.]

[17. -23. 0. 0. 3. 3. 0. 0.]

[0. 0. -17. 0. 0. 0. 4. 13.]

[0. 0. 0. -2. 0. 2. 0. 0.]

[0. 3. 0. 0. -9. 0. 0. 6.]

[0. 2. 0. 2. 0. -6. 0. 2.]

[0. 0. 4. 0. 0. 0. -4. 0.]

[0. 0. 14. 0. 6. 1. 0. -21.]]

1000 events:

[[-230. 230. 0. 0. 0. 0. 0. 0.]

[229. -362. 0. 0. 79. 54. 0. 0.]

[0. 0. -76. 0. 0. 0. 14. 62.]

[0. 0. 0. -11. 0. 11. 0. 0.]

[0. 77. 0. 0. -123. 0. 0. 46.]

[0. 56. 0. 11. 0. -71. 0. 4.]

[0. 0. 14. 0. 0. 0. -14. 0.]

[0. 0. 62. 0. 44. 6. 0. -112.]]

10000 events:

[[-230. 230. 0. 0. 0. 0. 0. 0.]

[229. -362. 0. 0. 79. 54. 0. 0.]

[0. 0. -76. 0. 0. 0. 14. 62.]

[0. 0. 0. -11. 0. 11. 0. 0.]

[0. 77. 0. 0. -123. 0. 0. 46.]

[0. 56. 0. 11. 0. -71. 0. 4.]

[0. 0. 14. 0. 0. 0. -14. 0.]

[0. 0. 62. 0. 44. 6. 0. -112.]]

100000 events:

[[-23493. 23493. 0. 0. 0. 0. 0. 0.]

[23492. -35739. 0. 0. 6762. 5485. 0. 0.]

[0. 0. -7576. 0. 0. 0. 1336. 6240.]

[0. 0. 0. -1628. 0. 1628. 0. 0.]

[0. 6315. 0. 0. -11009. 0. 0. 4694.]

[0. 5932. 0. 1628. 0. -7922. 0. 362.]

[0. 0. 1336. 0. 0. 0. -1336. 0.]

[0. 0. 6240. 0. 4247. 809. 0. -11296.]]

1000000 events:

[[-236880. 236880. 0. 0. 0. 0. 0. 0.]

[236879. -360077. 0. 0. 67869. 55329. 0. 0.]

[0. 0. -73700. 0. 0. 0. 12737. 60963.]

[0. 0. 0. -16040. 0. 16040. 0. 0.]

[0. 63343. 0. 0. -109729. 0. 0. 46386.]

[0. 59855. 0. 16040. 0. -79691. 0. 3796.]

[0. 0. 12737. 0. 0. 0. -12737. 0.]

[0. 0. 60963. 0. 41860. 8322. 0. -111145.]]

10000000 events:

[[-2374737. 2374737. 0. 0. 0. 0. 0. 0.]

[2374736. -3599300. 0. 0. 677116. 547448. 0. 0.]

[0. 0. -737706. 0. 0. 0. 130859. 606847.]

[0. 0. 0. -160419. 0. 160419. 0. 0.]

[0. 630694. 0. 0. -1096154. 0. 0. 465460.]

[0. 593870. 0. 160419. 0. -791403. 0. 37114.]

[0. 0. 130859. 0. 0. 0. -130859. 0.]

[0. 0. 606847. 0. 419038. 83536. 0. -1109421.]]

100000000 events:

[[-23731856. 23731856. 0. 0. 0. 0. 0. 0.]

[23731856. -35989530. 0. 0. 6776273. 5481397. 0. 0.]

[0. 0. -7384635. 0. 0. 0. 1300003. 6084632.]

[0. 0. 0. -1604574. 0. 1604574. 0. 0.]

[0. 6317513. 0. 0. -10967122. 0. 0. 4649609.]

[0. 5940157. 0. 1604574. 0. -7916386. 0. 371655.]

[0. 0. 1300003. 0. 0. 0 -1300003. 0.]

[0. 0. 6084632. 0. 4190849. 830415. 0. -11105896.]]

(2) SPM at 3 µM [ACh] with one rate fixed at 1 s^{-1}:

10 events:

[[-1. 0. 0. 1. 0. 0. 0. 0.]

[0. -0. FN 0. 0. 0. 0. 0.]

[0. 1. -1. 0. 0. FN FN 0.]

[FN 0. 0. -3. 0. 0. 0. 3.]

[0. 0. 0. 0. -0. 0. FN 0.]

[0. 0. 1. 0. 0. -1. 0. FN]

[0. 0. FN 0. FN 0. -0. FN]

[0. 0. 0. 2. 0. 1. FN -3.]]

100 events:

[[-2.   0.   0.   2.   0.   0.   0.   0.]

[0. -32. 32.   0.   0.   0.   0.   0.]

[0. 32. -41.   0.   0.   3.   6.   0.]

[1.   0.   0. -5.   0.   0.   0.   4.]

[0.   0.   0.   0. -1.   0.   1.   0.]

[0.   0.   3.   0.   0. -5.   0.   2.]

[0.   0.   6.   0.   1.   0. -7.   FN]

[0.   0.   0.   3.   0.   2.   1. -6.]]

1000 events:

[[-10.   0.   0.   10.   0.   0.   0.   0.]

[0. -264. 264.   0.   0.   0.   0.   0.]

[0. 265. -388.   0.   0.   59.   64.   0.]

[9.   0.   0. -55.   0.   0.   0.   46.]

[0.   0.   0.   0. -19.   0.   19.   0.]

[0.   0. 54.   0.   0. -91.   0. 37.]

[0.   0. 70.   0. 19.   0. -89.   FN]

[0.   0.   0. 45.   0. 32.   6. -83.]]

10000 events:

[[-140.   0.   0.   140.   0.   0.   0.   0.]

[0. -2321. 2321.   0.   0.   0.   0.   0.]

[0. 2321. -3522.   0.   0.   693.   508.   0.]

[139.   0.   0. -801.   0.   0.   0.   662.]

[0.   0.   0.   0. -157.   0.   157.   0.]

[0.   0.   645.   0.   0. -1131.   0.   486.]

[0.   0.   556.   0.   157.   0. -746.   33.]

[0.   0.   0.   662.   0.   438.   81. -1181.]]

100000 events:

[[-1252.   0.   0.   1252.   0.   0.   0.   0.]

[0. -24013. 24013.   0.   0.   0.   0.   0.]

[0. 24013. -36376.   0.   0.   6772.   5591.   0.]

[1251.   0.   0. -7077.   0.   0.   0.   5826.]

[0.   0.   0.   0. -1606.   0.   1606.   0.]

[0.   0.   6313.   0.   0. -10898.   0.   4585.]

[0. 0. 6050. 0. 1606. 0. -8011. 355.]

[0. 0. 0. 5825. 0. 4127. 814. -10766.]]

1000000 events:

[[-12800. 0. 0. 12800. 0. 0. 0. 0.]

[0. -238846. 238846. 0. 0. 0. 0. 0.]

[0. 238847. -361429. 0. 0. 67780. 54802. 0.]

[12799. 0. 0. -72648. 0. 0. 0. 59849.]

[0. 0. 0. 0. -15878. 0. 15878. 0.]

[0. 0. 63103. 0. 0. -109427. 0. 46324.]

[0. 0. 59480. 0. 15878. 0. -79078. 3720.]

[0. 0. 0. 59848. 0. 41647. 8398. -109893.]]

10000000 events:

[[-130278. 0. 0. 130278. 0. 0. 0. 0.]

[0. -2380844. 2380844. 0. 0. 0. 0. 0.]

[0. 2380844. -3604237. 0. 0. 675796. 547597. 0.]

[130277. 0. 0. -735000. 0. 0. 0. 604723.]

[0. 0. 0. 0. -159866. 0. 159866. 0.]

[0. 0. 630524. 0. 0. -1093860. 0. 463336.]

[0. 0. 592869. 0. 159866. 0. -790295. 37560.]

[0. 0. 0. 604722. 0. 418065. 82832. -1105619.]]

100000000 events:

[[-1293858. 0. 0. 1293858. 0. 0. 0. 0.]

[0. -23798364. 23798364. 0. 0. 0. 0. 0.]

[0. 23798364. -36054160. 0. 0. 6772092. 5483701. 0.]

[1293857. 0. 0. -7320886. 0. 0. 0. 6027029.]

[0. 0. 0. 0. -1607064. 0. 1607064. 0.]

[0. 0. 6314616. 0. 0. -10957402. 0. 4642786.]

[0. 0. 5941178. 0. 1607064. 0. -7923347. 375105.]

[0. 0. 0. 6027028. 0. 4185310. 832582. -11044920.]]

(3) All-connected with rate constants at 100 s^{-1}:

1000000 events:

[[-124438., 17824., 17785., 17594., 17788., 17814., 17860., 17773.]

[18063., -125101., 17977., 18030., 17675., 17659., 17864., 17833.]

[17653., 17828., -124620., 17953., 17777., 17842., 17801., 17766.]

[17689., 17848., 17858., -125412., 17818., 18152., 18022., 18025.]

[17426., 17961., 17721., 17965., -124818., 18018., 17917., 17810.]

[17819., 17752., 17869., 17917., 17810., -125045., 18018., 17860.]

[17800., 17772., 17819., 17894., 18010., 17891., -125342., 18156.]

[17988., 18116., 17591., 18059., 17940., 17669., 17860., -125223.]]

5.6 Results for the state distribution test

(1) All-connected 1 000 000 events

Means			
Expected	Observed	σ	Difference (%)
0.00142857	0.00142964	3.61E-06	0.0747
0.00142857	0.00142543	1.87E-06	-0.2197
0.00142857	0.00142414	9.97E-07	-0.3105
0.00142857	0.00143090	3.39E-06	0.1633
0.00142857	0.00142792	5.68E-06	-0.0455
0.00142857	0.00142596	2.49E-06	-0.1830
0.00142857	0.00142912	4.71E-06	0.0387
0.00142857	0.00143061	7.76E-06	0.1428

Table 8. Percent difference between expected and observed means of the SPD for the all-connected model (see 12.1 (2)).

Parameter fit (λ)			
Expected	Observed	σ	Difference (%)
700	699.33	2.75	-0.0955
700	701.08	1.48	0.1546
700	702.02	1.22	0.2886
700	698.32	0.74	-0.2400
700	701.07	4.09	0.1535
700	700.50	2.58	0.0712
700	701.46	2.76	0.2079
700	698.11	4.17	-0.2703

Table 9. Percent difference between expected and observed parameter λ of the SPD for the all-connected model (see 12.1 (2)).

Weights			
Expected	Observed	σ	Difference (%)
0.125	0.12517	0.0002447	0.1325
0.125	0.12461	0.0006144	-0.3083
0.125	0.12527	0.0000475	0.2123
0.125	0.12490	0.0003091	-0.0771
0.125	0.12480	0.0002464	-0.1629
0.125	0.12532	0.0003238	0.2587
0.125	0.12510	0.0002419	0.0787
0.125	0.12483	0.0002703	-0.1339

Table 10. Percent difference between expected and observed weight α of the SPD for the all-connected model (see 12.1 (2)).

(2) SPM at 3 µM [ACh] 10 000 000 events

Means			
Expected	Observed	σ	Difference (%)
0.00246914	0.00246834	6.02E-07	-0.0320
0.00046795	0.00046819	4.36E-07	0.0519
0.00039216	0.00039268	4.76E-07	0.1330
0.00003663	0.00003664	5.23E-08	0.0340
0.00002304	0.00002303	3.92E-08	-0.0655
0.00002200	0.00002198	1.01E-08	-0.0822
0.00001075	0.00001074	1.39E-08	-0.1072
0.00000437	0.00000438	2.00E-09	0.1278

Table 11. Percent difference between expected and observed means of the SPD for the SPM model at 3 µM [ACh] (see 12.1 (1)).

Parameter fit (λ)			
Expected	Observed	σ	Difference (%)
405	405.26	0.077	0.0651
2137	2136.16	2.100	-0.0391
2550	2545.61	3.281	-0.1722
27300	27237.56	55.230	-0.2287
43400	43419.54	83.100	0.0450
45450	45494.17	64.577	0.0972
93000	93164.76	317.389	0.1772
228600	228421.60	299.082	-0.0780

Table 12. Percent difference between expected and observed parameter λ of the SPD for the SPM model at 3 µM [ACh] (see 12.1 (1)).

Weights			
Expected	Observed	σ	Difference (%)
0.23763955	0.23729850	0.0002661	-0.1435
0.36016718	0.35986503	0.0003764	-0.0839
0.07358807	0.07376647	0.0002131	0.2424
0.01601787	0.01603070	0.0000441	0.0801
0.10964061	0.10983947	0.0001967	0.1814
0.07913177	0.07909557	0.0001363	-0.0458
0.01298613	0.01300320	0.0000804	0.1315
0.11082882	0.11110107	0.0003709	0.2456

Table 13. Percent difference between expected and observed weight α of the SPD for the SPM model at 3 µM [ACh] (see 12.1 (1)).

5.7 Missed and false events from the preliminary comparison of SKM-HMM and Threshold Crossing.

(1) Missed events:

	Cut-Off (kHz)									
[ACh] (µM)	5		10		15		20		25	
3	68381	76089	59741	68919	53021	61879	47487	54599	43049	46555
6	72767	78867	63882	73221	56508	66245	50313	58237	45447	49354
10	76960	82348	67208	77485	58936	69862	52382	61030	47043	51370
18	81821	86956	71166	82494	61642	74262	54318	64033	48526	53196
30	85678	90622	73489	86534	63251	77192	55278	65711	49039	53855
60	89450	90646	76636	90776	65510	80912	57145	68210	57145	68210
100	90696	96841	77498	92723	66492	82162	58101	69284	51552	56402
180	90833	98169	78214	93984	67974	83502	59926	70992	53497	58272
300	90137	98903	79198	94584	69983	84864	62430	73000	56420	60608

Table 14. Missed events for detections from SKM-HMM and Threshold Crossing (1/2). Across the entire concentration range (left column) for digital Gaussian cut-off frequencies ranging from 5 to 25 kHz. The columns in blue are numbers of missed events from Threshold Crossing. The columns in pink are numbers of missed events from SKM-HMM. The number of missed events is out of 100 000 possible events.

	Cut-Off (kHz)									
[ACh] (µM)	30		35		40		45		50	
3	39608	42156	35262	39607	27695	36897	20360	33961	13702	30414
6	41709	44515	37700	41624	31357	39070	23412	36612	16082	33419
10	43115	46011	39079	43074	33517	40447	25806	37932	18194	34951
18	44309	47479	39959	44145	34889	41288	28131	38706	20360	35890
30	44604	47769	40405	44271	35594	41552	30371	39118	22701	36517
60	45867	49123	41323	45356	36860	42322	31915	39892	25881	37486
100	47068	49982	42544	46425	38151	43317	33785	40792	28651	38591
180	48893	51749	44543	48122	40303	45053	36170	42669	31608	40486
300	51927	54510	47590	50993	43459	47989	39716	45495	35677	43385

Table 15. Missed events for detections from SKM-HMM and Threshold Crossing (2/2). Across the entire concentration range (left column) for digital Gaussian cut-off frequencies ranging from 30 to 50 kHz. The columns in blue are numbers of missed events from Threshold Crossing. The columns in pink are numbers of missed events from SKM-HMM. The number of missed events is out of 100 000 possible events.

(2) False events:

	Cut-Off (kHz)									
[ACh] (µM)	5		10		15		20		25	
3	0	0	0	0	2	0	0	0	2	0
6	0	0	0	0	0	0	1	0	1	0
10	0	0	0	0	0	0	0	0	1	0
18	0	0	0	0	0	0	1	0	2	0
30	0	0	0	0	0	0	1	0	6	0
60	0	0	0	0	0	0	1	0	1	0
100	0	0	0	0	0	0	1	0	2	0
180	0	0	0	0	0	0	2	0	7	0
300	0	0	0	0	0	0	1	0	3	1

Table 16. False events for detections from SKM-HMM and Threshold Crossing (1/2). Exactly like the layout of the missed event tables (Tables 14 and 15,) but counting the number of false events introduced by either Threshold Crossing (blue) or SKM-HMM (pink).

	Cut-Off (kHz)									
[ACh] (µM)	30		35		40		45		50	
3	31	3	1695	44	30949	740	189164	3524	645122	12014
6	17	3	562	26	9413	288	60012	1448	209652	4603
10	21	1	318	22	4480	214	26922	1069	94198	3532
18	19	1	222	23	2174	165	11917	735	42281	2191
30	9	2	148	10	1333	102	5767	496	22149	1515
60	17	3	79	8	702	60	3371	280	10859	846
100	11	0	82	7	495	45	2037	218	6387	531
180	13	3	69	14	345	39	1312	161	3774	394
300	16	1	83	12	296	39	953	111	2396	335

Table 17. False events for detections from SKM-HMM and Threshold Crossing (2/2). Exactly like the layout of the missed event tables (Tables 14 and 15,) but counting the number of false events introduced by either Threshold Crossing (blue) or SKM-HMM (pink).

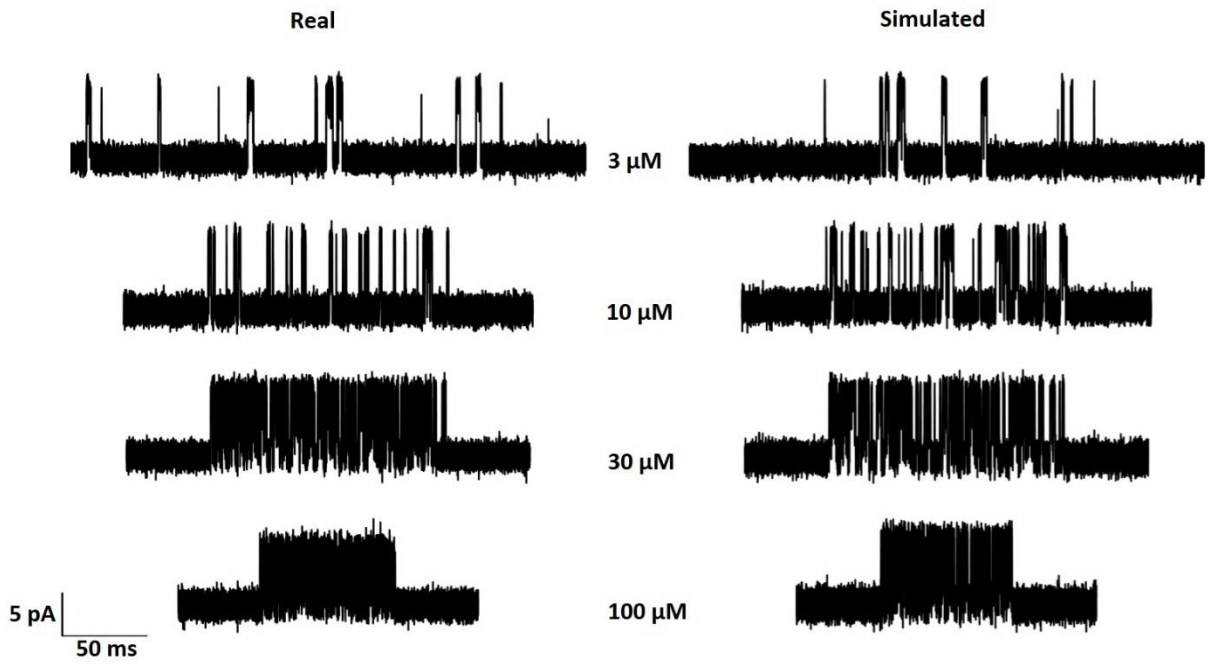

Figure 33. Comparison of real and simulated nAChR single-channel data. The left column displays real data, the right column displays simulated data. Each row shows a different ligand concentration (3, 10, 30 and 100 µM [ACh]).

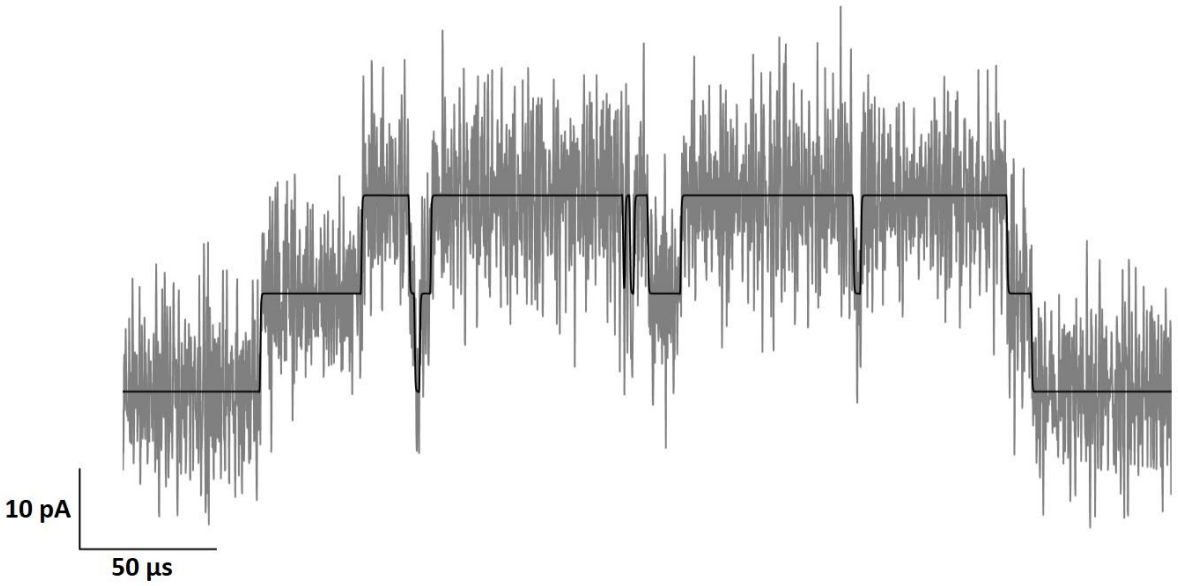

Figure 34. Example of simulated single-channel data with two channels.

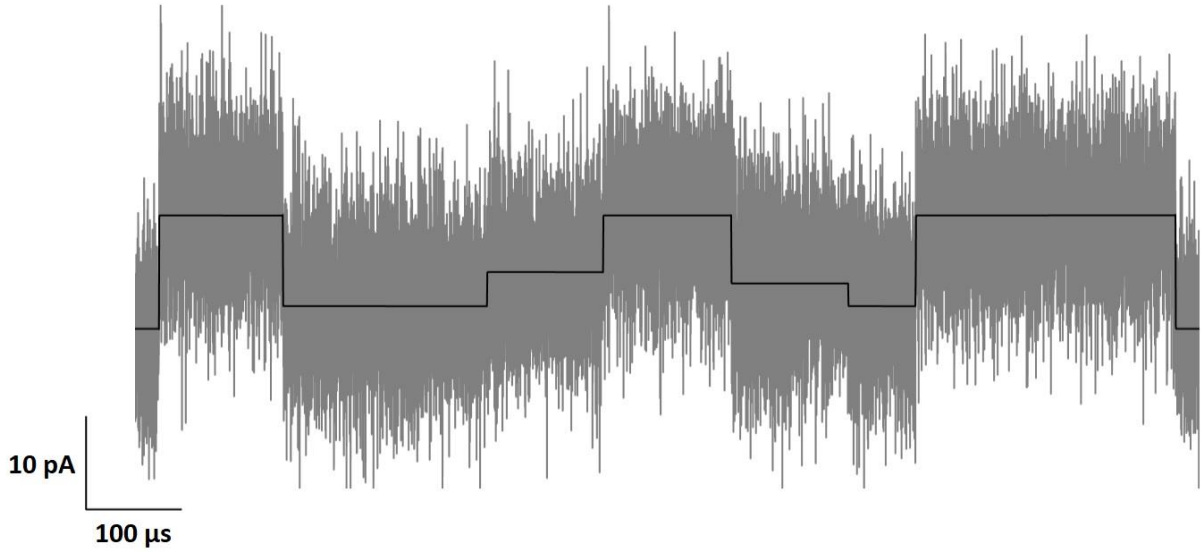

Figure 35. Example of simulated single-channel data with subconductance levels.

6 REFERENCES

1. Mukhtasimova N, daCosta CJ, Sine SM. 2016. Improved resolution of single channel dwell time reveals mechanisms of binding, priming and gating in muscle AChR. *J Gen Physiol.* 148(1):43-63.

2. Hille B. 1978. Ionic channels in excitable membranes. Current problems and biophysical approaches. *Biophys J.* 22(2): 283-294.

3. Jackson MB. 1984. Spontaneous openings of the acetylcholine receptor channel. *Proc. Natl. Acad. Sci. USA.* 81:3901-3904.

4. Duclohier H. 2009. Structure-function studies on the voltage-gated sodium channel. *Biochimica et Biophysica Acta.* 1788(11):2374-2379.

5. Cascio M. 2004. Structure and function of the glycine receptor and related nicotinicoid receptors. *JBC.* 279(19):19383-19386.

6. Barnett MW, Larkman PM. The action potential. *Practical Neurology.* 2007. 7:192-197.

7. Foster M, Sherrington CS. 1897. Textbook of Physiology 7th ed. London: Macmillan. 3:929.

8. Thorneloe KS, Nelson MT. 2005. Ion channels in smooth muscle: regulators of intracellular calcium and contractility. *Can J Physiol Pharmacol.* 83(3):215-242.

9. Stojilkovic SS, Tabak J, Bertram R. 2010. Ion channels and signaling in the pituitary gland. *Endocr Rev.* 31(6):845-915.

10. Kaczorowski GJ et al. 2008. Ion channels as drug targets: the next GPCRs. *J Gen Physiol.* 131(5):399-405.

11. Hübner CA, Jentsch TJ. 2002. Ion channel diseases. *Human Molecular Genetics.* 11(20):2435-2445.

12. Unwin N. 2005. Refined structure of the nicotinic acetylcholine receptor at 4 Å resolution. *J Mol Biol.* 346:967-989.

13. Unwin N. 2013. Nicotinic acetylcholine receptor and the structural basis of neuromuscular transmission: insights from Torpedo postsynaptic membranes. *Q Rev Biophys.* 46(4):283-322.

14. Brejc K et al. 2001. Crystal structure of an Ach-binding protein reveals the ligand-binding domain of nicotinic receptors. *Nature.* 411(6835):269-276.

15. Blount P, Merlie JP. 1989. Molecular basis of the two nonequivalent ligand binding sites of the muscle nicotinic acetylcholine receptor. *Neuron.* 3(3):349-357.

16. Mishina M et al. 1986. Molecular distinction between fetal and adult forms of muscle acetylcholine receptor. *Nature.* 321(6068):406-411.

17. Lindstrom J. 1997. Nicotinic acetylcholine receptors in health and disease. *Mol Neurobiol.* 15(2):193-222.

18. Decker MW, Meyer MD, Sullivan JP. 2001. The therapeutic potential of nicotinic acetylcholine receptor agonists for pain control. *Expert Opin Investig Drugs.* 10(10):1819-1830.

19. Verma S et at. 2018. Muscarinic and nicotinic acetylcholine receptor agonists: current scenario in Alzheimer's disease therapy. *J Pharm Pharmacol.* 70(8):985-993.

20. Tonge PJ. 2018. Drug-target kinetics in drug discovery. *ACS Chem Neurosci.* 9(1):29-39.

21. Samuele A et al. 2013. The power of enzyme kinetics in the drug development process. *Curr Pharm Biotechnol.* 14(5):551-60.

22. Moczydlowski E. 1992. Analysis of drug action at single-channel level. *Methods Enzymol.* 207:791-806.

23. Chopra, DA et al. 2017. A single-channel mechanism for pharmacological potentiation of GluN1/GluN2A NMDA receptors. *Scientific Reports.* 7(6933).

24. Corradi J, Bouzat C. 2016. Understanding the bases of function and modulation of α7 nicotinic receptors: implications for drug discovery. *Mol Pharmacol.* 90(3):288-299.

25. Mathie A et al. 1991. Conductance and kinetic properties of single nicotinic acetylcholine receptor channels in rat sympathetic neurones. *J Physiol.* 439:717-750.

26. daCosta CJ, Sine SM. 2013. Stoichiometry for drug potentiation of a pentameric ion channel. *PNAS.* 110(16):6595-6600.

27. Wu ZS et al. 2015. Ion channels gated by acetylcholine and serotonin: structures, biology and drug discovery. *Acta Pharmacol Sin.* 36(8):895-907.

28. del Castillo J, Katz B. 1957. Interaction at end-plate receptors between different choline derivatives. *Proc R Soc Lond B Biol Sci.* 146(924):369-381.

29. Lee WY, Sine SM. 2004. Invariant aspartic acid in muscle nicotinic receptor contributes selectively to the kinetics of agonist binding. *J Gen Physiol.* 124(5):555-67.

30. Sakmann B, Neher E. 1984. Patch clamp techniques for studying ionic channels in excitable membranes. *Annu. Rev. Physiol.* 46():455-472.

31. Kim J et al. 2013. A patch-clamp ASCI for nanopore-based DNA analysis. *IEEE Trans Biomed Circuits Syst.* 7(3):285-295.

32. Sakmann B, Neher E. 1976. Single-channel currents recorded from membrane of denervated frog musle fibres. *Nature.* 260(5554):799-802.

33. Sigworth FJ. 1995. Electronic design of the patch clamp. Edited by Sakmann B and Neher E, Single-channel recording, 2nd edition. *Plenum Press, New York.* 95-127.

34. Finkel AS. 1991. Progress in instrumentation technology for recording from single channels and small cells, in: Cellular and Molecular Neurobiology: A Practical Approach. *Oxford University Press.*3-25.

35. Levis RA, Rae JL. 1992. Constructing a patch clamp setup. *Methods Enzymol.* 207:18-66.

36. Levis RA, Rae JL. 1993. The use of quartz patch pipettes for low noise single channel recording. *Biophys. J.* 65:1666-1677

37. Wang HL et al. 2000. Fundamental gating mechanism of nicotinic receptor channel revealed by mutation causing a congenital myasthenic syndrome. *J Gen Physiol.* 116(3):449-462.

38. Raillon C et al. 2012. Fast and automatic processing of multi-level events in nanorpore translocation experiments. *Nanoscale.* 4:4916-4924.

39. Tabard-Cossa V et al. 2007. Noise analysis and reduction in solid-state nanopores. *Nanotechnology. 18(305505).*

40. Nyquist H. 1928. Thermal agitation of electic charge in conductance. *Phys Rev.* 32:110-113.

41. Smeets RM et al. 2008. Noise in solid-state nanopores. *Proc Natl Acad Sci USA.* 105(2):417-412.

42. Ahern GP et al. 1997. Subconductance states in single-channel activity of skeletal muscle ryanodine receptors after removal of FKBP12. *Biophys J.* 72(1):146-162.

43. Premkumar LS, Qin F, Auerbach A. 1997. Subconductance states of a mutant NMDA receptor channel. *J. Gen. Physiol.* 109:181-189.

44. Sigworth FJ, Sine SM. 1987. Data transformations for improved display and fitting of single-channel dwell time histograms. *Biophys J.* 52(6):1047-1054.

45. Sivilotti L, Colquhoun D. 2016. In praise of single channel kinetics. *J Gen Physiol.* 148(2):79-88.

46. Qin F, Auerbach A, Sachs F. 1996. Estimating sing-channel kinetic parameters from idealized patch-clamp data containing missed events. *Biophys J.* 70:264-280.

47. McManus OB, Blatz AL, Magleby KL. 1987. Sampling, log binning, fitting and plotting durations of open and shut intervals from single channels and the effects of noise. *Pflugers Arch.* 410(4-5):530-553.

48. Sine SM, Steinbach JH. 1986. Activation of acetylcholine receptors on clonal mammalian BC3H-1 cells by low concentrations of agonist. *J. Physiol.* 373(1):129-162.

49. Magleby KL. 1992. Ion channels. Preventing artifacts and reducing errors in single-channel analysis. *Methods Enzymol.* 207:763-791.

50. Sachs F, Neil J, Barkakati N. 1982. The automated analysis of data from single ionic channels. *Pflugers Arch.* 395:331-340.

51. Qin F. 2004. Restoration of single-channel currents using the segmental k-means method based on hidden Markov modeling. *Biophys J.* 86(3):1488-1501.

52. Colquhoun D. 1998. Binding, gating affinity and efficacy: the interpretation of structure-activity relationships for agonists and of the effects of mutating receptors. *Br J Pharmacol.* 125(5):924-47.

53. Hawkes AG, Jalali A, Colquhoun D. 1990. The distribution of the apparent open times and shut times in a single channel record when brief events cannot be detected. *Philos. Trans. R. Soc. Lond. A.* 332():511-538.

54. Hawkes AG, Jalali A, Colquhoun D. 1992. Asymptotic distributions of apparent open times and shut times in a single channel record allowing for the omission of brief events. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 337():383-404.

55. Colquhoun D, Hawkes AG, Srodzinki K. 1996. Joint distributions of apparent open and shut times of single-ion channels and maximum likelihood fitting of mechanisms. *Philos. Trans. R. Soc. Lond. A.* 354():2555-2590.

56. Colquhoun D, Hatton CJ, Hawkes AG. The quality of maximum likelihood estimates of ion channel rate constants. *J Physiol.* 547(3):699-728.

57. Ohno K et al. 1996. Congenital myasthenic syndrome caused by decreased agonist binding affinity due to a mutation in the acetylcholine receptor epsilon subunit. *Neuron.* 17(1):157-170.

58. Colquhoun D, Sakmann B. 1985. Fast events in single-channel currents activated by acetylcholine and its analogues at the frog muscle end-plate. *J. Physiol.* 369:501-557.

59. Karlin A. 1967. On the application of “a plausible model” of allosteric proteins to the receptor for acetylcholine. *J Theor Biol.* 16(2):306-320.

60. Venkataramanan L et al. 1998. Identification of Hidden Markov Models for Ion Channel Currents-Part I: Colored Background Noise. *IEEE Trans. On Sig. Proc.* 46(7):1901-1914.

61. Colquhoun D, Hawkes AG. 1995. The principles of the stochastic interpretation of ion-channel mechanisms. Edited by Sakmann B and Neher E, Single-channel recording, 2nd edition. *Plenum Press, New York.* 397-482.

62. Colquhoun D, Hawkes AG. 1995. A Q-Matrix Cookbook: How to write only one program to calculate the single-channel and macroscopic predictions for any kinetic mechanism. Edited by Sakmann B and Neher E, Single-channel recording, 2nd edition. *Plenum Press, New York.* 589-633.

63. Colquhoun D, Hawkes AG. 1995. Fitting and statistical analysis of single-channel records. Edited by Sakmann B and Neher E, Single-channel recording, 2nd edition. *Plenum Press, New York.* 483-587.

64. Matsumoto M, Nishimura T. 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation.* 8(1):3-30.

65. Jones E, Oliphant E, Peterson P, et al. 2001. SciPy: Open Source Scientific Tools for Python. http://www.scipy.org/

66. Cokelae et al. 2017. ‘Spectrum’: spectral analysis in python. *Journal of Spen Source Software.* 2(18): 384.

67. MATLAB and Statistics Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States. https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html