A novel method to rapidly fit conductance-based models to individual neurons

Felix Benjamin Kern

Submitted for the degree of Doctor of Philosophy

University of Sussex

September 2019

Declaration

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Felix Benjamin Kern

Acknowledgements

I wish to thank my supervisors, George Kemenes and Thomas Nowotny, for sending me on this journey, for their guidance along the way, and for their patience, trust, and support. I am grateful to Naoki Kogo for inspiring the StdpC extension with his ambitious project, and for patiently and tirelessly testing the software and reporting back with many bug reports. I also want to thank Tony Lewis for his generosity in allowing me to use his laboratory and resources for experiments with *Xenopus* oocytes, and his students, Ruolin Ma and Tiffany Morcom, for doing most of the legwork involved in preparing the oocytes for use in my experiments. I am indebted to Rafael Levi, with whose collaboration I was able to make an important addition to my work with closed-loop model fitting.

Finally, I owe a debt of thanks to my colleagues in the lab, to the Sussexsport team and the many friends I have made while sweating my stress away, to my family at home in Switzerland who have long suffered my absence, and to my amazing friends both near and far. You have all contributed to this work by helping to keep me sane and happy.

Since this thesis has passed examination, I can now also thank my examiners, Chris Buckley and Volker Steuber, who have offered insightful comments, asked important questions, and perhaps most importantly, shown enthusiasm for the work presented herein.

UNIVERSITY OF SUSSEX

FELIX BENJAMIN KERN

DOCTOR OF PHILOSOPHY IN COMPUTATIONAL NEUROSCIENCE

A novel method to rapidly fit conductance-based models to individual neurons

SUMMARY

In this thesis, I present a new method of model optimisation that allows the calibration of conductance-based models of neuronal membrane potential to data from just a single neuron, and achieves good correspondence with the reference data in mere minutes. These properties are desirable because they allow investigations of individual variability among neurons of a given type, of homoeostatic processes and non-synaptic plasticity events, as well as of the contribution of particular neuronal properties to the dynamics of small circuits.

In the first chapter, the thesis introduces in detail the working principle of the method, which can be summed up as **m**odel **o**ptimisation using **st**imuli to **i**solate **p**arameter **s**ubsets ("MOSTIPS"), and represents a major part of the work and novelty of this project. The second chapter focusses on the construction of accurate models of two mammalian potassium channels which, being ectopically expressed in *Xenopus laevis* oocytes, served as a validation tool for the new method. In the third chapter, I evaluate the new method, presenting results from fitting models to data from synthetic sources as well as the above-mentioned oocytes. Finally, the fourth chapter contains a number of related results from closed-loop electrophysiology approaches, including extensions to the dynamic clamp protocol for both single neurons and hybrid circuits composed of live and simulated neurons, as well as preliminary results from a closed-loop model fitting approach closely related to the main work presented above.

The thesis concludes that the newly developed approaches to model fitting constitute valuable additions to existing methods. The MOSTIPS method achieves tightly constrained parametrisations using both less data and less processing

time than classical methods, while the related closed-loop fitting approach produces results that closely follow ongoing changes in evoked activity patterns in real time. Conversely, some issues have been left unanswered, including the contribution of the stimulus generation and selection algorithm, the success of which I have been unable to establish, as well as whether the methods developed herein can reliably identify relevant properties of individual cells. Nevertheless, both the particular methods and the general approach of using prior estimates of the model and its parameter values to propose stimulus patterns represent major advances in the field of neuron model optimisation.

Table of Contents

List of abbreviations....xi
List of figures....xii
List of tables....xiv
1 Introduction....1
1.1 Hodgkin-Huxley models....1
1.2 Existing model optimisation methods....2
1.3 Neurons are not like peas in a pod....4
1.4 Individual variability matters....6
2 MOSTIPS: An algorithm for model optimisation using stimuli that isolate parameter subsets....10
2.1 Introduction....10
2.1.1 Model optimisation is a difficult problem....10
2.1.2 The key idea: actively isolate parameters....11
2.2 Finding stimuli that isolate parameter subsets....12
2.2.1 How to stimulate, and what to measure?....12
2.2.2 Approaches to quantifying parameter isolation....14
2.2.2.1 Estimating sensitivity....15
2.2.2.2 Judging parameter isolation from sensitivity....17
2.2.2.2.1 The bubble algorithm....17
2.2.2.2.2 The cluster algorithm....18
2.2.3 Stimulus generation....20
2.2.3.1 Searching for a search algorithm....21
2.2.3.2 Finding MAP-Elites....22
2.2.4 Robustness screening....25
2.2.4.1 Varying the target parameter only....26
2.2.4.2 Paired parameter space sampling....27
2.2.4.3 Unpaired parameter space sampling....28

2.2.5 Selection strategy....29
2.3 Model optimisation....31
2.3.1 General considerations....31
2.3.2 Model optimisation algorithms....32
2.3.2.1 Genetic algorithm....32
2.3.2.2 Differential evolution....34
2.4 Implementation details....36
2.4.1 General overview....36
2.4.2 Parallel computation....37
2.5 Conclusion....40
3 Potassium channel models....42
3.1 Introduction....42
3.2 Methods....43
3.2.1 Oocyte preparation....43
3.2.2 Electrophysiology....44
3.2.3 Stimulus protocols....44
3.2.4 Software....46
3.3 Building single-component models from literature....46
3.3.1 Kv2.1....47
3.3.2 Kv1.4....50
3.4 Parameter estimation and fitting procedures....52
3.4.1 Classical estimation of passive parameters....52
3.4.2 Model-free active parameter estimation....56
3.4.3 Time constant fitting with the tail current protocol....58
3.4.4 Model-constrained full-record fitting....59
3.5 Fixing model mismatch with a second component....60
3.5.1 Single-component fits are qualitatively unsatisfactory....60
3.5.2 Two-component Kv2.1x....61
3.5.3 Kv2.1x full-record fit....66

3.5.4 Two-component Kv1.4x....67
3.6 Conclusion....71
4 Empirical proof of concept....74
4.1 Introduction....74
4.2 Methods....75
4.2.1 Models....75
4.2.1.1 Kv2.1....76
4.2.1.2 Kv2.1x and Kv2.1k....77
4.2.1.3 Kv1.4x....77
4.2.1.4 Kboth....77
4.2.1.5 HH (Squid axon)....78
4.2.1.6 B1 (*Lymnaea stagnalis* buccal motor neuron)....79
4.2.2 Simulations....81
4.2.3 Stimulus generation....82
4.2.4 Stimulus screening and selection....85
4.2.5 Oocyte stimuli and recordings....87
4.2.6 Model optimisation....88
4.3 Fitting against synthetic data....89
4.3.1 Convergence within populations....90
4.3.1.1 Introduction....90
4.3.1.2 Parameter space distance....92
4.3.1.3 Results....93
4.3.2 Distance to reference parameters....97
4.3.2.1 Introduction....97
4.3.2.2 Results....99
4.3.3 Conclusions....105
4.4 Fitting against oocytes expressing ion channels....105
4.4.1 Random observation controls....106
4.4.2 Convergence within populations....107

4.4.3 Distance to reference parameters....112
4.4.4 Cross-validation....117
4.4.5 Conclusions....119
5 Closed-loop approaches....122
5.1 Introduction....122
5.2 Closed-loop model fitting....122
5.2.1 Motivation....122
5.2.1.1 An alternative approach....123
5.2.2 Implementation....124
5.2.2.1 Cost function....124
5.2.2.2 Stimulus selection....126
5.2.2.3 Model optimisation....126
5.2.3 Application....127
5.2.3.1 Methods....127
5.2.3.2 Model....128
5.2.3.3 Results....130
5.2.4 Conclusion....134
5.3 Extensions to StdpC....135
5.3.1 Existing features....136
5.3.2 Hybrid networks with complete neuron models....137
5.3.2.1 Motivation....137
5.3.2.2 Implementation....138
5.3.2.3 Application....141
5.3.3 Synaptic delay....142
5.3.4 Synaptic background noise....144
5.3.4.1 Motivation....144
5.3.4.2 Implementation....144
5.3.4.3 Application....145
5.3.5 Synaptic stochasticity....147

5.3.6 Software-defined voltage clamp....148
5.3.6.1 Motivation....148
5.3.6.2 Implementation....149
5.3.6.3 Application....150
5.3.7 Additional tools....151
5.3.8 Interface improvements....153
5.3.9 Outlook....157
6 Discussion....159
6.1 Delineating the method's applicability....160
6.2 K^+ channels have two components in oocytes....161
6.3 MOSTIPS is competitive in fitting predictive models....162
6.4 MOSTIPS does not solve model specificity....163
6.5 Closed-loop fitting as a way forward....164
6.6 More tools for the field to use....165
6.7 Future work....165
References....167
Appendix....A-1

List of abbreviations

CPG	Central Pattern Generator
CPU	Central Processing Unit
cRNA	Complementary Ribonucleic Acid
CUDA	Compute Unified Device Architecture
DE	Differential Evolution
GA	Genetic Algorithm
GeNN	GPU-enhanced Neural Networks
GPU	Graphics Processing Unit
HH	Hodgkin-Huxley squid axon model
MAP-Elites	Multidimensional Archive of Phenotypic Elites
mGA	Genetic Algorithm (without crossover)
MOSTIPS	Model Optimisation using Stimuli to Isolate Parameter Subsets
NS	Novelty Search
opt	Model Optimisation
PCA	Principal Components Analysis
PD	Parameter Space Distance
PID	Proportional-Integral-Differential
PSC	Post-Synaptic Conductance
RK4	4^{th}-order Runge-Kutta
RKF45	$4^{th}/5^{th}$-order Runge-Kutta-Fehlberg
RMS	Root Mean Square
RMSE	Root Mean Squared Error
RTDO	Real-Time Dynamic Observer
sg	Stimulus Generation
std	Standard Deviation
StdpC	Spike-Timing Dependent Plasticity Clamp
VC	Voltage Clamp
xGA	Genetic Algorithm (with crossover)

List of figures

Figure 2.1: A demonstration of the bubble and cluster algorithms.....19
Figure 2.2: Example error profiles.....27
Figure 3.1: Stimulus protocol command voltages.....45
Figure 3.2: Kv2.1 activation and inactivation steady state curves.....48
Figure 3.3: Kv2.1 time course data and models.....49
Figure 3.4: Kv1.4 activation and inactivation.....51
Figure 3.5: Kv1.4 time course data and model.....52
Figure 3.6: Representative example traces of a Kv2.1-expressing oocyte.....53
Figure 3.7: I-V plot derived from Kv2.1 activation protocol data.....54
Figure 3.8: Capacitance protocol step data.....54
Figure 3.9: Tail current fits.....55
Figure 3.10: Single-component Kv2.1 kinetics fail to accurately fit activation and slow inactivation processes.....61
Figure 3.11: Kv2.1 and Kv2.1x least-squares fits.....62
Figure 3.12: Kv2.1x fast and slow components.....62
Figure 3.13: Kv2.1x steady-state activation and inactivation values.....63
Figure 3.14: Kv2.1x inactivation time constants.....64
Figure 3.15: Fitted time constants for both Kv2.1x components.....65
Figure 3.16: Kv2.1x full-record kinetic fits.....67
Figure 3.17: Kv1.4 single-component and two-component fits.....68
Figure 3.18: Kv1.4x fast and slow components.....68
Figure 3.19: Kv1.4x fast and slow steady state inactivation.....69
Figure 3.20: Kv1.4x fast and slow inactivation time constants.....69
Figure 3.21: Kv1.4x fast and slow steady-state activation.....70
Figure 3.22: Kv1.4x fast and slow activation time constants.....71
Figure 4.1: Example MAP-Elites archive selection.....86
Figure 4.2: Example convergence for the B1 model.....91

Figure 4.3: Convergence within populations of fits to noise-free reference traces generated by originally parametrised models.....94
Figure 4.4: Convergence within populations of fits to noise-free reference traces generated by randomly parametrised models.....95
Figure 4.5: Convergence within populations of fits to noisy reference traces generated by randomly parametrised models.....97
Figure 4.6: Example distance to reference for the B1 model.....98
Figure 4.7: Distance to reference for fits against synthetic data.....100
Figure 4.8: Example convergence for the Kv1.4x model.....108
Figure 4.9: Convergence in the Kv2.1k model.....109
Figure 4.10: Convergence within populations of fits against oocyte data.....110
Figure 4.11: Example distance to reference for the Kv1.4x model.....112
Figure 4.12: Distance to reference in the Kv2.1k model.....114
Figure 4.13: Distance to reference for fits against oocyte data.....115
Figure 4.14: Convergence across fits to the same data.....116
Figure 4.15: Cross-validation error of oocyte MOSTIPS fits compared to reference.....118
Figure 5.1: Closed-loop fitting progress in three B1 cells.....132
Figure 5.2: PCA of the candidate parameter sets in closed-loop fitting.....133
Figure 5.3: A hybrid circuit built from live pyramidal neurons and model inhibitory neurons.....142
Figure 5.4: Effects of modelled synaptic background noise.....146
Figure 5.5: StdpC interface comparison.....154
Figure 5.6: StdpC graph interface and performance monitor.....155

List of tables

Table 2.1: Mutation operators....23
Table 2.2: MAP-Elites outcome dimensions....24
Table 3.1: Fitting cost comparison between single-component and two-component models....62
Table 3.2: Base value, range, and MOSTIPS metaparameters for the Kv2.1x model....66
Table 4.1: Model-specific hyperparameters for stimulus generation, robustness screening, and model optimisation....76
Table 4.2: Base value, range, and MOSTIPS metaparameters for the Kv2.1 model. 76
Table 4.3: Base value, ranges, and MOSTIPS metaparameters for the Kv1.4x model....77
Table 4.4: Base value, ranges, and MOSTIPS metaparameters for the Kboth model combining Kv1.4x and Kv2.1x....78
Table 4.5: Base value, range, and MOSTIPS metaparameters for the HH model. 79
Table 4.6: Base value, range, and MOSTIPS metaparameters for the B1 model..81
Table 4.7: Kv2.1 distance to reference by parameter....102
Table 4.8: HH distance to reference by parameter....103
Table 4.9: B1 distance to reference by parameter....104
Table 4.10: Cross-validation error of oocyte reference and MOSTIPS parameter sets....118
Table 5.1: Adjusted B1 model parameters' central values....129

1 Introduction

One of the most well-known and iconic phenomena of neuroscience, the action potential, is typically taught and described in terms of a mathematical model of a membrane containing voltage-gated sodium and potassium channels. This description, known as the Hodgkin-Huxley model, is one of the cornerstones of computational neuroscience, and forms part of a larger class of conductance-based neuron models. Though originally devised to describe the membrane potential of the squid giant axon (Hodgkin & Huxley, 1952a, 1952b, 1952c, 1952d; Hodgkin *et al.*, 1952), the general form of the model has since been applied much more widely (Rinzel, 1990; Catterall *et al.*, 2012). While both extensions, e.g. to multi-compartment models (Fitzhugh, 1962; Migliore & Shepherd, 2002), and generalisations, e.g. to Markov chain models (Armstrong, 1969; Hille, 1991; Strassberg & DeFelice, 1993), have been made, these will not be considered in this thesis.

1.1 Hodgkin-Huxley models

Under the Hodgkin-Huxley formalism, the membrane potential $V(t)$ is typically modelled as

$$C\frac{dV}{dt} = I_{inj} - g_l(V - E_l) - \sum_i I_i \qquad \text{(Equation 1.1)}$$

where C is the membrane capacitance, I_{inj} is any current injected through the experimenter's electrode, g_l and E_l are the conductance and reversal potential of a passive "leak" current, and finally, I_i are active ionic conductances, carried across a membrane by voltage-gated ion channels. In general, ion channels are modelled as active conductances with a maximum conductance $\bar{g}_i$ and reversal potential E_i. Active, here, means that the actual conductance value varies with time and membrane potential, based on the opening and closing of individual ion channels. That is, $\bar{g}_i$ is the conductance value in the (hypothetical) case that all

channel are opened. The mechanism for channel opening and closing, known as gating, is modelled with p activation gates and q inactivation gates, all acting independently. The voltage-dependent open state of these gates is described by the gating variables $m(t)$ and $h(t)$, respectively, according to equations 1.2.

$$\begin{aligned} I_i(t) &= m^p h^q \bar{g}_i (V - E_i) \\ \frac{dm}{dt} &= \frac{m_\infty(V) - m}{\tau_m(V)} \\ \frac{dh}{dt} &= \frac{h_\infty(V) - h}{\tau_h(V)} \end{aligned} \qquad \text{(Equation 1.2)}$$

The gating variables usually depend on voltage in some exponential fashion, using e.g. some form of Boltzmann function for the steady-state $x_\infty(V)$ and a related function for the time course $\tau_x(V)$. The choice of these functions depends on the particular current or ion channel species modelled and, once identified, is typically considered invariant in the context of experiments on a given system. Once the model structure is in place, there are many parameters that need to be adjusted in order for a model to reproduce, and eventually predict, the behaviour of a particular system. The process of tuning parameters is known as model optimisation. Due to the challenging nature of the problem – particularly when large numbers of parameters are involved – model optimisation has been tackled in a number of different ways, which I will briefly review in the following.

1.2 Existing model optimisation methods

The archetypal voltage clamp model optimisation procedure (Hodgkin & Huxley, 1952d; Willms *et al.*, 1999; Lee *et al.*, 2006) uses a combination of pharmacological agents such as tetrodotoxin or 4-aminopyridine to selectively block ion channels, ion replacement, replacing e.g. sodium with lithium ions, to suppress currents carried by the replaced ions, and channel-specific stimulus protocols to inactivate certain channels. By cleverly combining these procedures, experimenters can isolate and measure individual current types. These measurements are then combined to parametrise a composite model that includes all currents.

Usually, the final product is constructed from data pooled across a number of neurons, because gathering all required data from a single cell is impossible (with irreversible channel blockers) or at the very least not practically feasible due to the amount of time and disruptive manipulation required.

While model optimisation based on separation of currents may work for systems with a very limited number of distinct conductances, and indeed to fit the kinetics of individual currents, most systems are not neatly separable. Early approaches to the optimisation of more complex models entailed tuning parameters by hand to achieve a particular set of activity patterns (Traub *et al.*, 1991; De Schutter & Bower, 1994). Given the potentially large influence of parameter changes and the intractable complexity of parameter interactions, however, the validity of such hand-tuned models beyond the activity patterns considered for tuning is questionable.

With access to more computational power, some groups attempted to exhaustively search through parameter space, trying to find models that reproduce generic activity patterns seen in reference neurons (Bhalla & Bower, 1993; Prinz *et al.*, 2003; Prinz, Bucher, *et al.*, 2004; Taylor *et al.*, 2006). While this approach may increase the likelihood of finding a model that fits the reference data well, it can also turn up many false positives. With a rich set of conductances, neuron models are effectively degenerate, meaning that they can produce indistinguishable activity patterns from many different parameter combinations (Prinz, Bucher, *et al.*, 2004; Marder & Goaillard, 2006). Thus, a database approach gives no indication of how accurate a given solution is with respect to the actual biophysics it purports to model.

Other approaches are focused on observing specific instances of neuronal activity recorded in the absence of chemical or electrical manipulation. Among the more computationally expensive, but highly effective approaches for this are those that harness statistical methods, either straightforwardly fitting against membrane potential traces (Huys & Paninski, 2009; Toth *et al.*, 2011; Kostuk *et*

al., 2012; Meliza *et al.*, 2014), or transposing the problem into a dynamical systems perspective and fitting against voltage trajectories in state space (Vavoulis *et al.*, 2012). Using similar kinds of reference data, but different fitting methods, some approaches apply more explicit search methods such as evolutionary algorithms or simulated annealing to locate optimal parameter combinations (Vanier & Bower, 1999; Van Geit *et al.*, 2008; Svensson *et al.*, 2012).

In order to achieve models that faithfully reproduce the full range of neuronal activity, however, passive recordings of membrane potential are often not good enough, as the neuron may never enter states that provide adequate information about some parameters. Perturbing the system, either with random current clamp or voltage clamp injections (Hobbs & Hooper, 2008; Tomaiuolo *et al.*, 2012; Brookings *et al.*, 2014) or with specially selected stimuli (Nowotny *et al.*, 2008; Druckmann *et al.*, 2011), promises to improve the amount of information contained in a given reference trace.

1.3 Neurons are not like peas in a pod

A fundamental assumption underlying most of the model optimisation approaches outlined above is that the neurons used during optimisation are homogeneous, to the extent that any differences in current dynamics or conductances are negligible. That is, the implicit and often untested assumption is made that all neurons of a given type – i.e. the particular system that a model is supposed to describe – show identical or near-identical behaviour, and that this behaviour arises from a similarly identical or near-identical biological "implementation" in terms of the currents present in these neurons. Indeed, the formulation of the conductance-based models suggests a direct mapping linking the mathematical description of currents and the activity of specific ion channel types in the neuronal membrane.

However, a careful exploration of how model parameters relate to the biophysical reality they supposedly describe (Foster *et al.*, 1993; Marder & Goaillard,

2006) casts doubt on this assumption. The invertebrate neurons investigated in the Marder lab are morphologically and functionally identified, meaning that, from one animal to another, they appear in the same approximate location, connect to the same pre- and post-synaptic partners, exhibit the same activity pattern and fulfil the same circuit role. By all relevant indicators, therefore, one would expect these neurons to also be roughly identical in terms of their ion channels and membrane currents. Surprisingly, however, the relative magnitudes of different conductances varies between cells (Golowasch *et al.*, 2002; Schulz *et al.*, 2006), and the behaviour of neurons depends on the relative magnitude of conductances in complex, non-linear fashion (Goldman *et al.*, 2001). Therefore, model optimisation approaches that use average data from multiple neurons are not only unrepresentative of any individual cell, but may in fact exhibit behaviour that differs strongly from that of its reference cells.

In vertebrate neuroscience, there is a notion of cell types, identified by morphology, function, genetic identity, and activity patterns (Lein *et al.*, 2007; Kim *et al.*, 2017; Erö *et al.*, 2018; Murakami *et al.*, 2018). However, here, too, there is variability within these types and subtypes (Pospischil *et al.*, 2008; Migliore *et al.*, 2018), with a growing recognition that there is no sharp boundary distinguishing clusters (of e.g. activity or gene expression patterns) that we might identify as types (Phillips *et al.*, 2018).

In addition to inter-individual variability within a given neuron type, neuronal properties also change over time in response to or as a consequence of changing morphology, environment, and electrical activity (Turrigiano *et al.*, 1994; Desai *et al.*, 1999; MacLean *et al.*, 2003; Soofi *et al.*, 2014). While some of these changes are likely to be homoeostatic, regulating e.g. excitability towards a set point of desired activity, other changes may be unpredictable or even chaotic in nature (Gal *et al.*, 2010). Finally, neurons in both invertebrate systems (Antonov *et al.*, 2001; Kemenes *et al.*, 2006; Nikitin *et al.*, 2008, 2013) and vertebrate systems (de Jonge *et al.*, 1990; Moyer *et al.*, 1996; Saar *et al.*, 1998) are known to exhibit

non-synaptic plasticity (Zhang & Linden, 2003; Mozzachiodi & Byrne, 2010), that is, they alter their behaviour in functionally relevant ways to support various forms of memory.

Taken together, these phenomena show that, at least in some situations, models that abstract beyond the individual cell and attempt to describe a neuron type are insufficient and will lead to false predictions of the neurons' behaviour. To make matters worse, because biophysically realistic models contain a substantial number of distinct conductances, there are potentially many different parameter combinations, or regions of parameter space, that give rise to comparable activity patterns (Prinz *et al.*, 2003; Prinz, Bucher, *et al.*, 2004).

1.4 Individual variability matters

In many contexts, of course, it is irrelevant whether a model's internal structure matches that of the modelled system; the important feature is often just the input-output relationship, or the overall dynamics of the system in the average case. As long as such features are captured faithfully and serve the modeller's goals adequately, there is no need for the model to accurately reproduce underlying mechanisms or their individual differences. Such is the case, for example, in network models that use conductance-based neuron models that reflect average response properties of entire populations of neurons.

However, there are lines of inquiry and methodologies, focused particularly on single neurons or small circuits, that make explicit reference to underlying mechanisms, and therefore require accurate models thereof. This is particularly important in investigations using dynamic clamp (Robinson & Kawai, 1993; Sharp *et al.*, 1993; Prinz, Abbott, *et al.*, 2004; Economo *et al.*, 2010). Briefly, dynamic clamp is a method of electrically inserting virtual conductances into a membrane by measuring the membrane potential, simulating the desired conductance in real time, and continuously injecting a corresponding current through an electrode. This can be used e.g. to virtually modulate a particular ionic conductance that is

expected or known to be present in the target membrane (Ma & Koester, 1996; Hughes *et al.*, 1998; Zhang *et al.*, 2003). Investigations of this kind would benefit greatly from having a cell-specific model of the manipulated conductance.

Dynamic clamp can also be used to construct hybrid circuits composed of a combination of real and modelled neurons (Wilders *et al.*, 1996; Le Masson *et al.*, 2002; Debay *et al.*, 2004; Oprisan *et al.*, 2004; Sorensen *et al.*, 2004; Sieling *et al.*, 2009). This is a particularly useful tool to investigate e.g. the dynamics of central pattern generators (CPG), i.e. tightly coupled neuronal circuits that generate stereotyped rhythmic activity (Marder & Calabrese, 1996). It allows us to substitute a real neuron with its model and systematically probe model (or synapse) parameters for their relevance to circuit-level phenomena. However, if the individual characteristics of the substituted neuron have functional relevance in this context, any such investigation may have to take these potentially idiosyncratic characteristics into account. For example, the activity pattern of the pyloric CPG in crustaceans has been shown to exhibit robust dynamic invariants (Elices *et al.*, 2019) in the intact circuit, which have so far resisted any attempts at replication in hybrid circuits (personal communication with I. Elices).

Thus, for both single-neuron and circuit investigations where individual variability is known or suspected to play a role, it would be very advantageous to be able to optimise models to individual neurons. Ideally, a tool to do this would provide a fully optimised, biophysically accurate model of a single neuron within experimentally relevant time scales (i.e., minutes), with minimal disruption of the reference neuron, such that the experimenter can use the optimised model on either the reference cell, or on the circuit to which the reference cell belongs or belonged.

Almost all of the model optimisation methods reviewed above fall short of this goal. Most methods require either large amounts of data (often from multiple cells), large amounts of time, or both. A promising approach is taken by Tomaiuolo et al. (2012), who fit models to voltage traces under white noise current

injection, achieving results both within the desired time frame and optimised to a single neuron's activity. However, much like the database approach (Prinz *et al.*, 2003; Prinz, Bucher, *et al.*, 2004), this method is susceptible to false positives due to model degeneracy.

Milescu et al. (2008) propose a single-neuron fitting method based on recording voltage traces in an intact cell, then pharmacologically blocking a conductance and reinserting it via dynamic clamp, adjusting its parameters to recover the previously recorded activity pattern. While they have done this only for a single sodium conductance, the method could in principle be iterated to achieve a complete neuron-specific and biophysically accurate model. However, applying channel blockers constitutes a significant and potentially irreversible disruption to the system, such that this method is likewise unsuitable, particularly in the context of hybrid circuits.

Finally, Reyes-Sanchez et al. (2018) use a set of closed-loop algorithms to achieve and maintain desired activity patterns in hybrid circuits, gradually adjusting parameters in a feedback-driven manner. The resulting model neurons are calibrated not to match the replaced real neuron, but to act as a robust substitute in the circuit. While useful for investigations of circuit-level phenomena, the lack of a direct reference to the replaced neuron means that any model that provides the expected input-output relationship is acceptable to the algorithm, including models with simplified internal dynamics. Thus, this method also does not provide a clear mapping between the reference neuron and the properties of the model.

In this thesis, I present two closely related methods that attempt to address this gap in our toolbox. The first method, elucidated in detail in chapter 2, focuses on biophysical specificity, aiming to optimise models to a point where the model parameters reflect an underlying reality in the reference neuron. A thorough proof of concept of this method is presented in chapter 4, based on the model system developed in chapter 3.

The second method is an extension of the first, sacrificing parameter specificity to an extent in order to optimise models faster and less intrusively. Developed with a view towards application in hybrid circuits, it is presented in chapter 5 alongside an account of my efforts in improving and extending a set of tools for dynamic clamp.

2 MOSTIPS: An algorithm for model optimisation using stimuli that isolate parameter subsets

2.1 Introduction

2.1.1 Model optimisation is a difficult problem

Model optimisation is the process of finding parameter values that accurately reflect properties of the modelled system or that, at a minimum, permit the model to accurately predict the system's behaviour. In order to optimise the model of a particular neuron's membrane currents, we need to fine-tune values for the membrane capacitance, the leak conductance and reversal potential, and the maximum conductance and reversal potential of any putatively present ion channel species. Additionally, it may also be necessary to fine-tune parameters governing the voltage dependence and kinetics of channel gating. Finally, the size and configuration of compartments in a multi-compartment model may also need to be optimised.

This presents significant challenges: Firstly, as noted in the general introduction, conductance-based models are degenerate, allowing different parameter combinations to exhibit the same surface-level behaviour or, more precisely, to replicate a given data set with high fidelity without necessarily reflecting e.g. the true current densities. Secondly, parameter space – the set of all possible parameter combinations – grows with the power of the number of parameters considered for optimisation, a problem aptly named the curse of dimensionality. As a consequence, a naïve optimisation algorithm must either spend enormous computational resources to fully explore parameter space, or sample the space so sparsely that finding a globally optimal parameter set becomes highly unlikely. This problem is compounded by the presence of non-linear interactions between

parameters, which may mislead optimisation approaches that rely on gradients, and which are in plentiful supply in conductance-based neuron models.

Therefore, an accurate model optimisation algorithm must solve the degeneracy problem and not be misled by parameter interactions, and a fast algorithm must circumvent the curse of dimensionality. Ideally, we would like to be able to optimise each parameter independently of the rest; if that were possible, we could guarantee that parameter values reflect an underlying reality, and we would be freed from sampling the entire parameter space, sampling instead just along each target parameter's axis while holding the remaining parameters constant. Unfortunately, interactions between parameters make this approach seemingly impossible.

2.1.2 The key idea: actively isolate parameters

Observing a conductance-based neuron model over time, however, we notice that the influence of certain subsystems over the behaviour of the whole system varies. For example, while fast sodium currents dominate the rise of a neuronal action potential, slower potassium currents become influential during repolarisation. Under voltage clamp, this is even more pronounced, with classical step protocols routinely maximising the contribution of target currents while minimising that of others, e.g. by holding the membrane potential at an intermediate level to inactivate sodium channels before probing the properties of non-inactivating potassium currents. In other words, it is possible to at least partially decouple some parameters from others – those governing distinct currents – by observing the system in particular states in which those parameters are highly influential.

This observation underlies the key idea behind the model optimisation method presented in this chapter: It should be possible to drive the system to states that highlight the contribution of each model parameter, thus approaching the ideal situation outlined above. Each parameter could then be optimised to fit the system's behaviour in states that are most prominently governed by that parameter.

In order to implement this optimisation strategy, we need two things: Firstly, we need to find stimulus patterns and associated observation windows that force the system into a state dominated, or at least strongly influenced, by each parameter. While this could be done by hand for simple models with few parameters, my objective was to design a largely automated, model-agnostic method, which is detailed in section 2.2. Secondly, parameter isolation is unlikely to be perfect, that is, non-target parameters will almost certainly have some level of influence over the system's observed behaviour even with optimal stimuli and observation windows. Thus, we need an optimisation algorithm that can both make good use of such parameter isolation as is achievable, and deal with a number of different cost functions – the parameter-specific stimuli and observations – which should nonetheless all point towards a single, globally optimal solution. My approach to this problem is described in section 2.3. In section 2.4, I outline the software implementation of this set of algorithms, which I have used to generate the results in chapter 4, picking out some of the salient details that make it possible to simulate the large numbers of neuron models required by the algorithms. Finally, in section 2.5, I conclude the chapter with a discussion of the difficulties involved in testing and verifying the method.

2.2 Finding stimuli that isolate parameter subsets

2.2.1 How to stimulate, and what to measure?

In search of observations that can reliably isolate the influences of individual model parameters, it is useful to consider first the kind of data that is most suitable. Targeted stimulation can obviously provide more information about any parameter than passive observation of the membrane potential, but should we use current clamp or voltage clamp? The advantage of current clamp – injecting current to observe membrane potential – is its experimental ease and its relatively low impact on the cell. However, active processes such as channel gating both influence and are influenced by membrane potential, which precludes pre-

cise control over the system state. Furthermore, due to the system's highly nonlinear nature, minor parameter changes can precipitate large changes in the observed voltage trace, which makes optimisation very difficult. In voltage clamp, on the other hand, we dictate one of the system's main control parameters, the neuron's membrane potential, while measuring the current required for this control. By breaking the feedback loop between channel gating and the voltage that causes it, we achieve much tighter control over the system state, which at least to some extent allows us to separate the system into its ionic currents, as demonstrated by (Hodgkin *et al.*, 1952; Hodgkin & Huxley, 1952d, 1952a, 1952b, 1952c) and countless subsequent investigations of membrane currents using this method. Since isolating a current is equivalent to isolating a subset of a model's parameters, and therefore close to single parameter isolation, I have spent the majority of my efforts on working with voltage clamp data.

Voltage clamp does come with added complexity, however: Unlike the open-loop current clamp paradigm, where the measured membrane potential does not influence the injected current, voltage clamp requires closed-loop control. Usually, this is performed in hardware by a feedback amplifier and associated circuitry, which needs to be manually tuned to achieve good control over a cell's membrane potential. As a consequence, the achieved membrane potential, and thus by way of the feedback loop the injected current, depend on experimental parameters such as the feedback gain A and the access resistance R_a of the injecting electrode and its headstage. To not falsely attribute these influences to the neuron model, voltage clamp simulation in this chapter and in chapter 3 always includes a coarse model of the basic voltage clamp circuit, as follows. Given a command voltage V_{cmd} and a membrane voltage V_m, the clamping amplifier's output voltage is $V_o = A(V_{cmd} - V_m)$, which is dropped over the injecting electrode and the membrane, i.e., $V_o = V_m + R_a I_{inj}$ (Halliwell *et al.*, 1994). Substituting V_o and rearranging, we can express the injected current as a function of clamp and control parameters, i.e., $I_{inj} = \frac{A}{R_a}(V_{cmd} - V_m) - \frac{1}{R_a}V_m$. During fitting

to experimental data, the true values for A and R_a are used, while simulations in the preliminary stages detailed below are run with approximate values from experience.

The voltage clamp stimuli used throughout this work are combinations of linear voltage segments, i.e. steps with constant voltage and ramps with constant increase or decrease in voltage. This is inspired by common voltage clamp protocols using similar techniques, sharpening the notion that the investigated quantities are currents; besides, using linear segments reduces both the complexity of simulation and the dimensionality of stimulus space, which keeps the task of finding stimuli tractable without requiring an unreasonable amount of computational resources. In principle, however, the algorithms presented in this chapter are agnostic to both the form of the stimuli and of the model to which they are applied.

2.2.2 Approaches to quantifying parameter isolation

Having decided on the kind of data to use, we next need a method of judging the goodness of parameter isolation, with an eye towards using this judgement as a cost function when algorithmically generating stimuli. Since we can not expect to perfectly isolate each parameter, this measure should reflect a parameter's relative influence, or in other words, the system's sensitivity to changes in this parameter. Further, since the system state evolves during stimulation, we must expect this sensitivity to change over time, too, so as we are looking for highly sensitive regions, we should capture instantaneous, i.e. time-dependent, sensitivity. This suggests a form of sensitivity analysis evaluated against specific stimuli and sampled over time; from the resulting parameter-specific sensitivity traces, we can then extract observation windows during which the system's response, i.e. the membrane or clamp current, is dominated by a target parameter.

2.2.2.1 Estimating sensitivity

Mathematically speaking, we can describe the clamp current I_S evoked by a given stimulus S as a function of time t and the set of k model parameters $\mathbf{p} = [p_1, \ldots, p_k]$. Sensitivity to changes in a given parameter p_i corresponds to the partial derivative $J_i(S;t) = \left|\frac{\partial}{\partial p_i} I_S(t, \mathbf{p})\right|$, so we can write sensitivity to parameter changes as the Jacobian $\mathbf{J}(S;t) = [J_1(S;t), \ldots, J_k(S;t)]$, assuming that our models are differentiable. To provide an intuition, evaluating $J(S;t)$ over time for a given parameter set $\mathbf{p}$ would produce a set of traces – one for each parameter – rising and falling with the sensitivity of the clamp current to that parameter. In the general case of arbitrary input stimuli and model structures, however, the Jacobian is not analytically accessible. Therefore, we must approximate $J(S;t)$. To do this, I make two simplifying assumptions: Firstly, I assume that each partial derivative $J_i(S;t)$ is approximately linear in its target parameter p_i, which is indeed the case for many, though by no means all, parameters under voltage clamp (see e.g. the left-hand side plots in Figure 2.2 on page 27). Secondly, I assume that each $J_i(S;t)$ is relatively unaffected by the value of non-target parameters. While less defensible, this assumption may hold within certain limits; more importantly, it allows us to express our estimate of $J_i(S;t)$ as a real value, rather than as a function of $\mathbf{p}$.

In practice, the method I use to approximate $\mathbf{J}(S;t)$ is predicated on simulating a model in voltage clamp using several parameter sets and analysing the differences in the resulting clamp current. The naïve approach, which we might call radial detuning, would be to start from a well-chosen parameter set $\mathbf{p_0}$, populated e.g. during model identification by hand-tuning or using parameter values from relevant literature, and to approximate $\frac{\partial}{\partial p_i} I_S(t, \mathbf{p_0})$ by shifting each parameter p_i in turn by a small amount σ_i. For normalisation purposes, each σ_i can be tuned to produce an average J_i of similar magnitude across many stimuli. While this method generates a good approximation for the local $\mathbf{J}(S;t,\mathbf{p_0})$ and is computa-

tionally cheap, requiring only $k+1$ simulations, it makes no attempt to capture any remaining nonlinearities or parameter interactions and may therefore not generalise well even in a tightly constrained parameter space.

In recognition of this, I use a method referred to as spiral detuning, which follows a more generic elementary effects strategy (Morris, 1991; Saltelli & Annoni, 2010). To alleviate the problem of local specificity, I choose $m+1$ starting points with $\mathbf{p_0}$ being the hand-chosen parameter set, and $\mathbf{p_1}, \ldots, \mathbf{p_m}$ sampled uniformly from within the permissible parameter space. Further, rather than stepping radially from each starting point, I detune each successive parameter without stepping back to the origin, which results in a spiralling trajectory through parameter space; this both reduces the dependence on the starting point, and facilitates implementation by allowing me to detune each parameter more than once where this is computationally advantageous. As above, however, only parameter sets that differ in exactly one parameter are compared, each such pairwise comparison yielding one elementary effect $J_i^j(S;t) = \frac{1}{\bar{\Delta}_i}\left|\frac{\partial I_S(t,\mathbf{p_j})}{\partial p_i}\right|$. Because each J_i^j only partly depends on the detuning step size σ_i, normalisation is performed post hoc at the level of elementary effects, using $\bar{\Delta}_i$, the mean elementary effect size across many stimuli, rather than normalising σ_i. Finally, I average across the elementary effects of each parameter to approximate global sensitivity $\mathbf{J}(S;t) = \left[\bar{J}_1(S;t), \ldots, \bar{J}_k(S;t)\right]$.

Although no additional attempt is made to analyse or harness the nonlinearities and parameter interactions that this method could reveal, the extended sampling across parameter space should make the estimate more robust. In practice, I use only a small number of starting points during the search for stimuli in order to keep the computational cost bearable. This trade-off is not unreasonable, as a more comprehensive sampling of parameter space is performed in a later step, as described in section 2.2.4.

2.2.2.2 Judging parameter isolation from sensitivity

As a reminder, the purpose of measuring parameter sensitivity is to find stimuli, or regions of interest within stimuli, during which a given model is highly sensitive to a target parameter and relatively insensitive to non-target parameters. In other words, we need an algorithm that finds, for a given stimulus S and from among the evaluated time points $\mathbb{T}_S$, a set of observation windows $obs \subset \mathbb{T}_S$ that fulfils the criterion of high relative sensitivity to a target parameter. Additionally, we need to score these observations with a parameter-specific fitness function $f_i(S, obs)$ in order to maximise parameter isolation, i.e. relative sensitivity. I propose two approaches to achieve this, each taking a different perspective on the problem.

2.2.2.2.1 The bubble algorithm

The intuition for the "bubble" algorithm is to consider an overlay of the sensitivity traces for each parameter; a region of interest is a period during which the target parameter's trace rises above the rest, forming a "bubble" of high relative sensitivity, as illustrated in the blue shaded area in Figure 2.1. Although we may hope to find stimuli for each parameter that produce "absolute" bubbles, meaning regions where $J_i(S;t) > J_j(S;t)\ \forall j \in \{1,\dots,k\}, j \neq i$, this is by no means guaranteed. In contrast, because the components of $\mathbf{J}(S;t)$ are normalised across many stimuli, as detailed in section 2.2.2.1, there must for each parameter be stimuli and observations where $J_i(S;t) > \bar{J}(S;t)$, the mean sensitivity across all parameters. Such "relative" bubbles, therefore, can reliably be found and optimised for. To do so, I define a measure for instantaneous parameter isolation, $\phi_i(S;t) = \frac{J_i(S;t)}{\bar{J}(S;t)}$. A valid bubble, then, is an interval $obs = [t_{start}, t_{end}]$ such that $\phi_i(S;t) > 1\ \forall t \in obs$, and its score or fitness is the mean isolation value within the bubble, discretised to $f_i(S, obs) = \frac{1}{\|obs\|} \sum_{t \in obs} \phi_i(S;t)$. By using the mean magnitude of ϕ rather than, say, its integral, f prefers short observation win-

dows with strong parameter isolation, i.e. with $J_i \gg \bar{J}$, to long windows where J_i only barely exceeds $\bar{J}$ and the target parameter p_i is not well separated from the remaining parameters.

While the bubble approach reliably identifies regions of high relative parameter influence, it makes no attempt to control or even consider other parameter influences except in terms of their mean value. Thus, not only may the target parameter be far from the most influential one during the observation, but in addition, the sensitivity profile may change drastically within a bubble, which makes it difficult to credit the error between model and observation to individual parameters. To address these challenges, we need a more holistic approach that takes all parameter influences equally into account.

2.2.2.2.2 The cluster algorithm

The inspiration for the “cluster” algorithm comes from the realisation that perfect, or even good, single-parameter isolation is unlikely to be attainable in the general case. As a consequence, during model optimisation proper (see section 2.3), it makes sense to consider the relative influence of all parameters on a given observation, fitting each according to its weight. A better approach to finding good observations, then, would be to find observations during which relative parameter sensitivity changes little, so that credit for the current error can be assigned with greater certainty. Here, I am no longer looking for contiguous observation windows, but clusters of observations with a similar profile of relative sensitivity, hence the algorithm’s name.

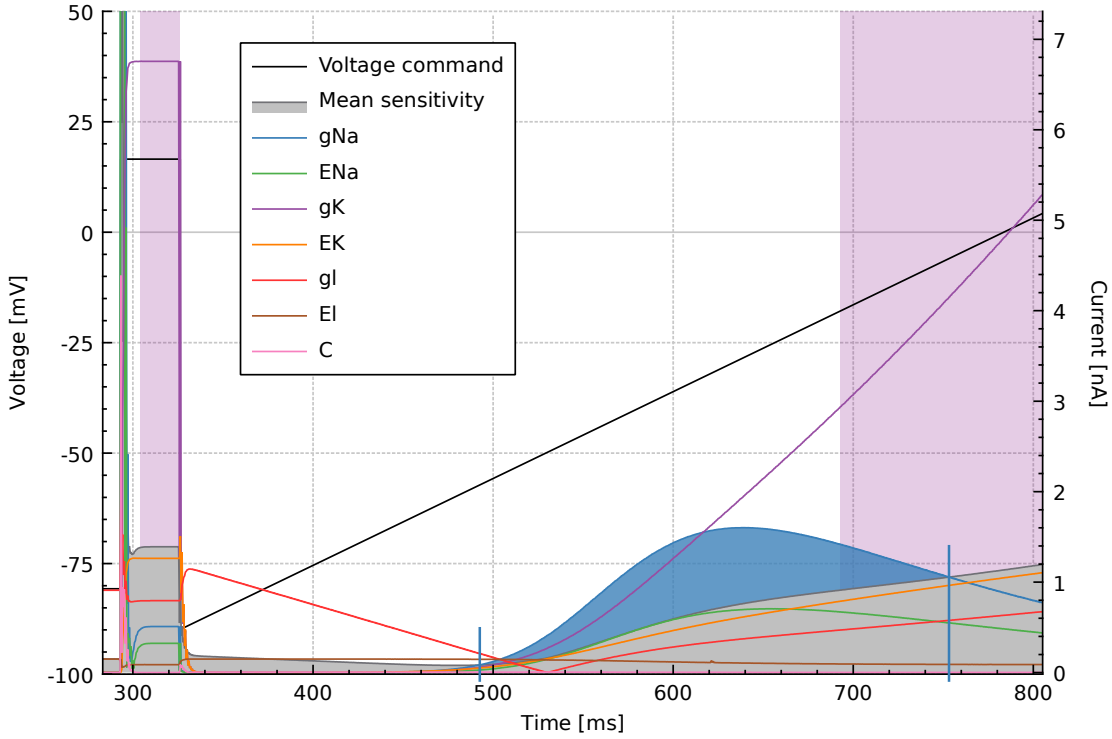


Figure 2.1: A demonstration of the bubble and cluster algorithms, using the Hodgkin-Huxley squid axon model and an opportune stimulus. Coloured traces represent each parameter's sensitivity $J_\bullet(S;t)$, calculated using radial detuning with normalised $\sigma_\bullet$, while the grey trace and shaded area represents the mean sensitivity $\bar{J}(S;t)$.

*Shaded in **blue** and marked by vertical blue bars is a **bubble** for the Na^+ maximum conductance gNa. Note how drastically the relative sensitivities change within the chosen period, and how J_{gNa} is far outweighed by J_{gK} in the latter part. Thus, while the current error of a deviating parameter set might indicate an error in gNa, it may also reflect an inaccurate gK, yet the bubble algorithm ignores this contribution.*

*Shaded in **purple** are two areas that the **cluster** algorithm considers similar (with a similarity threshold of 0.98, i.e., all $\hat{\mathbf{J}}$ in the cluster are within 11.5° of the cluster head). Although absolute sensitivity differs within that cluster, the relative sensitivity across parameters clearly remains comparable.*

What does it mean for relative sensitivity to be similar? Remember that $\mathbf{J}(S;t)$ is a vector representing the response current's sensitivity to each parameter in $\mathbf{p}$. If we consider the unit vectors $\hat{\mathbf{J}} = \frac{\mathbf{J}}{\|\mathbf{J}\|}$, then $\hat{\mathbf{J}}_1 \cdot \hat{\mathbf{J}}_2 = \cos\measuredangle(\hat{\mathbf{J}}_1, \hat{\mathbf{J}}_2)$ provides a similarity measure ranging from 0 (orthogonal) to 1 (identical). Setting a threshold θ,

we can define a cluster of similar observations $C = \{t_1, \ldots, t_n\}$ such that $\hat{\mathbf{J}}(S; t_i) \cdot \hat{\mathbf{J}}(S; t_j) > \theta \ \forall i, j \in \{1, \ldots, n\}$. To delineate useful clusters of this kind, we may want to look for high target parameter sensitivity directly; alternatively – since we're already resigned to only partial isolation – we may want to simply maximise the amount of useful data by finding the longest few clusters in a given stimulation, score these according to target parameter sensitivity, and hope for decent isolation to emerge over the course of the search algorithm. I use this latter approach because, as an added benefit, applying this twofold objective of long duration and high sensitivity may help to ease the search algorithm's tendency to overfit.

To find large clusters, I compute the similarity $D_{ij} = \hat{\mathbf{J}}(S; t_i) \cdot \hat{\mathbf{J}}(S; t_j)$ for all pairs (t_i, t_j), and count $N(i)$, the number of time points t_j for which $D_{ij} > \theta$. I then find the time point t_a at which $\hat{\mathbf{J}}(S; t_a)$ is similar to the largest number of neighbours, i.e. $a = \arg\max N(i)$, and extract these neighbours as a cluster $C_a = \{t_j : D_{aj} > \theta\}$. While not strictly a cluster by the definition above – similarity to the cluster head t_a does not entail similarity among all members – it is a close enough approximation as to be useful for our purposes. To iterate the clustering process, I select the next cluster head $b = \arg\max N(i)$ from the not yet clustered time points and extract its associated cluster C_b, and so on until a maximum number of clusters is reached, or an extracted cluster is too small to be useful. Finally, using $\mathbf{J}_C = \sum_{t \in C} \mathbf{J}(S; t)$, I compute each cluster's average parameter sensitivity $\hat{\mathbf{J}}_C = \frac{\mathbf{J}_C}{\|\mathbf{J}_C\|}$, using its components as the fitness value for each parameter, i.e. $f_i(S, C) = \hat{J}_{C,i}$.

2.2.3 Stimulus generation

Having described the methods I use to evaluate stimuli and observation windows, we now turn to the algorithm that generates stimuli which maximise these measures. The stimuli considered, as mentioned in section 2.2.1, are composed of sequences of command voltage steps or ramps. Although this is a relatively

simple set, it is still too large to search exhaustively, so we need an efficient search algorithm. The following section relates how I arrived at my chosen solution, which long pre-dates my conception of the cluster algorithm; any reference to fitness, therefore, refers to variations of the bubble algorithm.

2.2.3.1 Searching for a search algorithm

To identify promising stimuli, I first turned to a genetic algorithm (GA; Holland, 1992; Reeves & Rowe, 2002; De Jong, 2006), which starts from a population of randomised stimuli, scores these by a fitness function, selects the more successful ones for mutation (e.g. random changes to stimulus properties, or exchange of properties between two stimuli, see Table 2.1) to generate a new population, iterating this process until a halting condition is reached. The results I achieved with this approach, however, were brittle: Trying several variations of the bubble fitness function produced either extremely short observation windows, or extremely long and uninformative ones; stimuli rarely converged across runs and were useless for model optimisation even under perfect conditions. This suggested that the fitness function might be highly deceptive, presenting no clear path towards a global optimum and instead guiding solutions towards many different local optima.

Looking for alternatives that might overcome such a rugged fitness function, I considered novelty search (NS; Lehman & Stanley, 2011), which – rather than rewarding high fitness outright – looks for solutions that differ in some aspects of their outcomes unrelated to fitness, building an archive of novel solutions. Faced then with a large and varied archive of ostensibly high-quality solutions, I used a GA to filter and optimise for fitness alone. My understanding was that even with a very rugged and deceptive fitness function, at least some of the novel solutions should be near the global fitness maximum, which would then be found. As with the pure GA approach, however, the combined NS+GA approach

seemed to drive stimuli into a corner of high fitness, where they generalised very poorly to actual model optimisation.

Clearly, I could not rely on just a fitness function, no matter how much sense it made on paper. I judged that adding additional desiderata to the fitness function, such as long but meaningful observations or high clamp current levels, would have diluted the simplicity of the function, potentially making it more brittle still, and I wanted to avoid multi-objective optimisation for fear of simply multiplying the problem of inadequate objective functions. Instead, I realised that I needed to select for additional desirable features not during the search, but after it – and using a better, more comprehensive metric (described in section 2.2.4) than the univariate fitness function that had proven so unreliable. In other words, I needed not a search algorithm that spits out a single best solution, but an illuminating algorithm that produces a diverse set of high-quality solutions.

2.2.3.2 Finding MAP-Elites

To be able to select from a range of useful outcome variables, I turned to the MAP-Elites algorithm (Mouret & Clune, 2015). This algorithm expands on the idea of novelty, replacing the population of candidate solutions used in GA and NS with an archive which is explicitly structured by selected outcome measures. The archive consists of a hypercube defined by these outcome dimensions and is partitioned into grid cells by a choice of bins along each dimension. Each such cell can be occupied by a single candidate, namely the one with the highest fitness value that falls within the corresponding bins along each outcome axis. A populated archive thus represents a set of solutions that differ from each other in the secondary outcome measures, but are all more “fit” than any other candidate evaluated with the same secondary outcomes.

The algorithm is initialised with a number of randomly generated stimuli, whose performance is calculated both in terms of the fitness function and of the chosen outcome measures. Candidates are then binned along each of the outcome di-

mensions and thereby assigned a cell in the archive; if that cell is already occupied, the candidate with the lower fitness value is discarded. Then, once all initial candidates are processed, new candidates are drawn uniformly from the archive, mutated as in a GA, and evaluated in turn.

Operator	P	Details
Crossover	2	With a second parent chosen uniformly from all available candidates. Crossover between stimuli of duration d is performed by drawing a random time t_c such that $0 < t_c < d$, and generating a new stimulus with all steps and ramps from one parent in $[0, t_c]$, and those from the other in $[t_c, d]$.
Voltage change	2	The command voltage after one step or at the end of one ramp, chosen uniformly from the stimulus's steps and ramps, is increased or decreased by a randomly drawn voltage $V \sim N(0, 10)$ mV.
Time change	2	As above, but for time, with $t \sim N(0, 5)$ ms.
Number of steps	1	A randomly drawn step or ramp is removed from or newly generated and added to the stimulus.
Step swap	0.5	Two distinct steps or ramps are drawn uniformly from within the stimulus, exchanging their voltage values as well as the flag designating them a step or a ramp.
Step type	1	One randomly drawn step is turned into a ramp, or vice versa.

Table 2.1: Mutation operators applied in order to generate slight changes in stimuli. In my MAP-Elites implementation, novel stimuli are generated from existing ones using n separate mutation operations, with $n = \max(1, k),\ k \sim N(2, 2)$ drawn separately for each new stimulus. Operators are chosen at random with relative probabilities as indicated in the second column.

To evolve stimuli with this algorithm, I have used various outcome dimensions; those used for the results presented in chapter 3 are summarised in Table 2.2. Further, the archive is initialised with a set number of bins along each outcome dimension, usually between 32 and 128. After a number of epochs, the number of bins is doubled and their size halved to increase the archive's resolution as it begins to saturate.

Outcome dimension	Notes
Duration	Total duration of the evaluated observation window
Onset time	Bubble onset time. This is not used for cluster-type fitness, because it is less meaningful when multiple observation windows are possible.
Voltage deviation	Mean deviation between command and holding voltage over the entire stimulus
Current	Mean clamp current within the evaluated observation window, estimated from the simulation of all starting point models. The maximum is user-defined, as it depends on the model used.
Number of clusters	Total number of valid clusters in the stimulus as a measure of its complexity. Hard-coded to 32 integer bins.
Target parameter	Index of the parameter for which fitness is evaluated. This dimension is unaffected by changes in archive resolution.

Table 2.2: MAP-Elites outcome dimensions

The MAP-Elites algorithm has several important advantages over the GA-based approaches. Because it eschews the notion of a population, it produces the best stimuli in each niche of outcomes, rather than discarding them in favour of candidates with higher fitness and different outcomes, and the search is unbiased, sampling the entire space of outcomes evenly. As a result, it gives a clear view of the fitness landscape, or how (achieved) fitness is affected by the chosen outcome variables. This gives me as an experimenter the option to make conscious decisions to trade off between several desirable stimulus properties, rather than putting me at the fitness function's mercy. Finally, by using the target parameter as an outcome measure, a single run of the algorithm can produce candidate stimuli for all parameters, which allows it to harness synergies between the fitness functions for different parameters, and to escape local optima more easily by exposing the evaluated stimuli to several different effective fitness functions.

Conversely, the solutions that come out of a MAP-Elites run are overwhelmingly numerous. I have typically produced archives with between 10^4 and 10^7 stimuli,

in stark contrast to the single “best” solution to come out of a GA. Thus, the MAP-Elites algorithm cannot stand on its own; it must be accompanied by a subsequent stimulus selection process. In the next section, we will explore some approaches to this, and end with a description of the strategy I have come up with to whittle the many candidates down to an individual solution to use in subsequent experiments.

2.2.4 Robustness screening

A potential shortfall of the measures to quantify parameter isolation described in section 2.2.2 is that they may be overly specific to the parameter sets used in their calculation, and may not generalise well across parameter space. Particularly parameters such as current densities may differ significantly between and across cells, experiments, and model assumptions. Since stimulus generation as laid out above is a computationally costly process, however, it isn’t feasible to adjust to such differences during model optimisation. Therefore, it is important that we select stimuli that generalise well, i.e. that provide good information even in the face of changed parameters. In other words, the data gathered with a given stimulus and observation window should always point our model optimisation algorithm in the right direction, regardless of where in parameter space the true model and the candidates lie. To an extent, this can be achieved during stimulus generation by choosing to evaluate parameter sensitivity at various points in parameter space, as I do with spiral detuning and multiple starting points. However, because the model generation algorithm has to evaluate a very large number of candidate stimuli, there is simply no room for a thorough sampling of parameter space. Additionally, as noted above, there is a need for an alternative evaluation method that can help filter a completed MAP-Elites archive and support the choice of a single best stimulus.

2.2.4.1 Varying the target parameter only

As a first approximation to judging robustness, one might attempt to map the error landscape of a given stimulus/observation pair. A naïve approach to this would be to focus only on the target parameter of a stimulus, exploring a range of its values and relating them to the reported error (i.e. the mean squared deviation of the clamp current between reference data and a candidate model over the observed period). The examples in Figure 2.2 show the result of this process for a number of stimuli evolved against two parameters of a model of a neuron in the crustacean stomatogastric ganglion (Liu *et al.*, 1998; Golowasch *et al.*, 2002), using radial detuning, bubble evaluation, and a MAP-Elites generation approach. How could this data be useful? One might argue that a steeper relationship between parameter value and error could be better for model optimisation, if the optimisation algorithm made use of the magnitude of the error gradient – which, however, neither of those I use do (see section 2.3). Alternatively, one might select for smooth relationships, such that the error increases monotonically with increasing distance to the correct model; however, for many parameters (particularly maximum conductances), a monotonic relationship is almost tautologically true, and so also fails to constitute a good selection criterion. More importantly, varying only one parameter at a time and assuming correct values for the rest ignores the reality of model optimisation, where we must expect all parameters to be mostly wrong most of the time. Besides, in trying to assess robustness, we cannot assume a single true model, but should instead expect any parameter set to be a possible end point of optimisation.

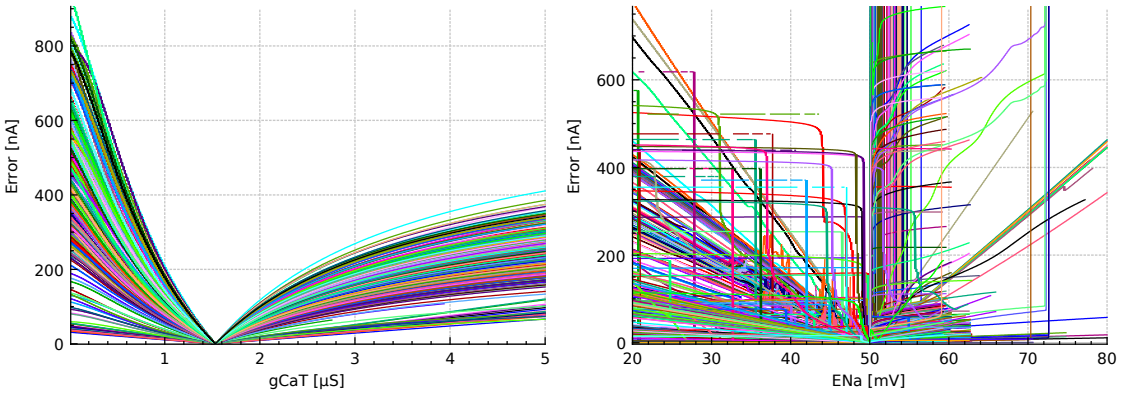


Figure 2.2: Example error profiles. Starting from a crustacean stomatogastric ganglion model (Golowasch et al., 2002) with 11 free parameters, including equilibrium potentials, current densities, and capacitance, stimuli of 300 ms duration were generated using radial detuning, bubble evaluation function, and MAP-Elites with three outcome dimensions (duration, onset time, and voltage deviation). 548 and 505 stimuli (gCaT and ENa, respectively) were chosen from the archive by selecting for bubble durations up to 12 ms and highest fitness along the voltage deviation axis. The target parameter was then varied systematically, and the mean observed current error against the default parameter set $|I_{default} - I_{shifted}|$ reported. The two plots are representative of the typical outcome for current densities (left) and equilibrium potentials (right). Many of the stimuli targeting ENa are clearly very brittle and could be filtered out by this method, whereas the gCaT profiles are qualitatively so similar that there is no obvious sensible selection criterion.

A better approach, therefore, would be to sample uncorrelated points in parameter space and use each parameter set as a reference against which others are judged. I again propose two approaches, one narrowly focused on variation in a target parameter, developed against bubble stimuli, the other considering parameter space in its full dimensionality as is appropriate for cluster stimuli. In both approaches, I start from a large number of models sampled uniformly from parameter space, i.e. from within the range of allowable parameter values.

2.2.4.2 Paired parameter space sampling

In the narrow, parameter-specific version, a uniformly sampled model population is paired with a detuned version of itself, in which the target parameter is shifted

by small amount in every model. Each stimulus under consideration is simulated using all models. Each tuned model is then used as a reference against which all other tuned/detuned pairs are evaluated. The quantities of interest are the errors $\varepsilon = \sum_{t \in obs} |I_{probe}(t) - I_{reference}(t)|$ of the tuned and the detuned model of a pair. Intuitively, this error should increase with increasing distance between probe and reference, so that the error gradient points towards the reference model; this is directly analogous to a monotonic fall of the error towards the target value in Figure 2.2, but now with disparate values for non-target parameters. We can describe the proportion of probe/reference pairs that satisfy this intuition as the accuracy of a stimulus. This provides a better measure than the naïve approach described above, in that it probes non-target parameter variation, and correctly punishes stimuli that produce deceptive observations.

2.2.4.3 Unpaired parameter space sampling

However, for cluster-based observations, this approach is inadequate: Since here, we know the relative importance of each parameter, it makes little sense to consider only the distance along one parameter-space axis and ignore other, perhaps more influential parameter differences. Indeed, given a stimulus/observation pair's sensitivity vector $\mathbf{J}$, we should expect the error with respect to a reference model to grow roughly in proportion to the components of $\mathbf{J}$ as we move away from it in parameter space. To test this, I define a sensitivity-weighted distance D_{ij} between two models $\mathbf{p_i}$ and $\mathbf{p_j}$ as $D_{ij} = \|(\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{J}\|$. Again using a population of M models sampled uniformly from parameter space, now without detuning, I compute the pairwise sample correlation coefficient ρ between the weighted distance D_{ij} and the corresponding error ε_{ij} for all pairs $\{i, j\}$ with $i, j \in \{1 \dots M\}, i < j$. A strong positive correlation indicates a relatively smooth error landscape, where sensitivity-weighted distance to the reference model is predictive of the observed error. This is a useful metric for model optimisation, as it means that, on average, a model with a lower error with respect to data is

also closer – as far as we can tell, given the sensitivity of the stimulus/observation pair – to the true model in parameter space.

This method effectively subsumes the paired approach in section 2.2.4.2, in that the sensitivity vector $\mathbf{J}_{sens}$ can be replaced with a target-only weighting $\mathbf{J}_{to}$, i.e. considering only distance along the target parameter axis, which corresponds to the target-only detuning of the paired approach. Conversely, we can leave D_{ij} an unweighted Euclidean distance, ignoring parameter sensitivity entirely and using an all-ones $\mathbf{J}_{uw}$. In either case, the expectation is that if the model optimisation algorithm uses a search function similar to the weighting chosen here – that is, if it searches through parameter space in steps scaled by $\mathbf{J}_\bullet$ –, then a high ρ means that, on average, the error drops with a step towards the true model, and grows with a step away from it.

2.2.5 Selection strategy

The final part of stimulus generation, having synthesised a large archive of candidate stimuli and outlined a method of screening them for robustness, is to select a single appropriate stimulus for each parameter to use in model optimisation. The limiting factor here is the computational cost of robustness screening, where each stimulus is evaluated against a large number of models. Thus, we need to pre-select promising stimuli from the MAP-Elites archive. I have employed the following pre-selection strategies:

- Dimensionality reduction: In archives with more than two outcome dimension, only the two most experimentally relevant features are retained. For each bin in the remaining two-dimensional grid, the stimulus with the highest fitness value is chosen, and all other candidates discarded.
- Outcome limits: For the remaining outcome dimensions, appropriate limits are set, e.g. a lower bound to the duration of observation or to the observed current. Such limits ensure that non-viable stimuli are excluded by

adding experimenter knowledge such as the expected magnitude of instrumentation noise or the duration of current injection artefacts.

- Tolerant Pareto front: As a final pre-selection method, a modified Pareto optimality criterion is used, maximising over fitness and each of the remaining outcome dimensions. Briefly, a solution x with fitness x_1 and outcome dimensions x_2, x_3 is Pareto-optimal if there is no other solution y such that $y_i \geq x_i \ \forall i$ and $y_i > x_i$ for at least one i. Since solutions in the archive are binned along the outcome dimensions, a pure Pareto optimality criterion would select a very small set of solutions[1] and might therefore discard useful (i.e., potentially robust), but slightly suboptimal solutions. To counteract this, I define a tolerance value $\vartheta_i > 0$ for each dimension, including fitness, and declare a solution x to be tolerantly-optimal if there is no solution y such that $y_i \geq x_i + \vartheta_i \ \forall i$ and $y_i > x_i + \vartheta_i$ for at least one i.

The set of all pre-selected stimulus/observation pairs $\mathbf{S}$ is then evaluated for robustness, yielding for each candidate S weighted, unweighted and target-only $\rho(S)$. Based on their fitness value $f(S)$, candidates are then assigned a score $\zeta(S) = f(S)\frac{\rho(S)-\rho_{min}}{\rho_{max}-\rho_{min}}$, where $\rho_{max} = \max \rho(S_i \in \mathbf{S})$ and $\rho_{min} = \lfloor \rho(S_i \in \mathbf{S}) \rfloor$, i.e. either -1 or 0, depending on the presence of negative correlations (unless otherwise noted, see section 4.2.5). Finally, the stimulus with the highest score is selected for use in model optimisation. A complete set of stimuli thus contains one stimulus for each fitted parameter, evolved to maximise that parameter's influence, screened for robustness, and selected for a combination of high relative parameter influence and high robustness to different parametrisations.

1 Although the theoretical maximum number of Pareto-optimal solutions is equal to the product of the outcome bin counts, in practice, the Pareto front is often a narrow sliver at the edge of "outcome space", and thus effectively limited to something close to the smallest bin count.

2.3 Model optimisation

Having outlined how I arrive at a small set of parameter-specific stimuli, we now turn to the algorithms I use to optimise models to data gathered with these stimulus sets. As a reminder, the point of using these stimuli is to reduce the dimensionality of the search space, with each stimulus rendering some or most of the model parameters irrelevant to the observed outcome, while highlighting the influence of others. In other words, we can think of each stimulus/observation pair as a lens that allows us to look at a low-dimensional version of parameter space and identify regions of high correspondence between the data and our models.

2.3.1 General considerations

To make good use of this transformation, our fitting algorithm should incorporate information about which parameters are being targeted, e.g. by limiting its search to sensitive parameters. Thus, rather than using all stimulus/observation pairs at once, attempting to fit all parameters to all data in an unconstrained manner, we break optimisation down into constrained, partial fits to individual observations with known parameter sensitivity profiles, iterating over each stimulus/observation pair in turn to home in on an optimal parameter set.

Therefore, our fitting algorithm must also be able to deal with the multiple fitness criteria presented by the stimulus/observation pairs. This is a non-trivial point: assuming a noise-free environment and a perfect match between the model structure and the target neuron, all stimulus/observation pairs would agree on a single globally optimal parameter set. However, since we cannot assume a perfect match, it is conceivable that global optima differ between observations. Since, given the first consideration above, each partial fit can only make use of one fitness criterion, a deterministic fitting algorithm could easily get stuck in a loop, jumping from one optimum to the next. These optima could even be global in the context of their respective fitness function, but may be nothing of the sort when compared to all data. Thus, we need an exploratory algorithm

that moves less purposefully, retains imperfect fits, and thereby arrives at solutions that are satisfactory to all fitness criteria.

2.3.2 Model optimisation algorithms

Again, I propose two (albeit closely related) algorithms that match the above requirements, a genetic algorithm (GA) and a self-adapting differential evolution algorithm (DE). Both use a set of candidate models, termed their population, which is generated randomly in the first epoch and evolved using a combination of selection and mutation from one epoch to the next. In each epoch, the population is evaluated by simulating it with one stimulus, yielding an error $\varepsilon = \sum_{t \in obs} (I_{model}(t) - I_{reference}(t))^2$ for each model. The stimulus/observation pair for the next epoch is then chosen, so that the algorithms can incorporate the sensitivity vector $\mathbf{J}(S)$ into their offspring generation schemes. The stimulus is chosen semi-randomly, in that the probability of choosing a given stimulus S is proportional to the number of epochs since S was last used; this is to prevent the model population from following a stereotyped trajectory through the various fitness landscapes that might appear in sequential scheduling, while reducing the likelihood that a stimulus is not used for many epochs. Next, using ε, evaluated "parent" models are selected and propagated to generate the "offspring" population used in the next epoch. The key difference between the two algorithms lies in their population dynamics and mutation schemes, i.e., how they select and generate offspring models from the parent population, as described below.

2.3.2.1 Genetic algorithm

In the GA, the parent population is sorted by error. The n_R lowest-performing candidate models are re-initialised with randomly generated parameter values, giving the algorithm an opportunity to escape local optima. Conversely, the n_E highest-performing models (the "elite") are retained in the offspring population without change so as not to lose high-quality candidates to deleterious muta-

tions. The remaining $N - n_R - n_E$ offspring models are generated by mutating parent models. Algorithmically, the process goes something like this:

sort population by decreasing error
randomly reinitialise models {1, 2, … n_R}
for i from n_R + 1 to N – n_E:
    pick parents a, b from models {i, i+1, … N}
    generate offspring model i by mutation from parents a, b
retain models {N – n_E + 1, …, N} unchanged

Note that the i-th model of the new population is, in the general case, offspring not of the i-th model of the parent population, but rather of randomly chosen models with a performance equal or better than the i-th parent. In this way, reproduction is somewhere between a completely elitist GA, where only the best candidates reproduce, and a purely fitness-based GA, where the probability of reproduction is proportional to a candidate's fitness. Here, the probability of reproducing is an approximately exponential function of a candidate's rank.

From the chosen parent models, an offspring is generated in two steps. Firstly, with a "crossover" probability Cr, parameter values are picked at random (p=0.5) from either parent; otherwise, parameter values are copied from only one of the two parents. Secondly, mutations are applied, meaning small random changes to the parameter values, governed by the sensitivity vector $\mathbf{J}(S_{e+1})$ of the following epoch's stimulus/observation pair. In a given epoch e, parameters $\mathbf{p}$ are updated as either $p_i(e+1) = p_i(e) + R_i,\ R_i \sim N(0, s_i)$ for additive parameters, or $p_i(e+1) = R_i p_i(e), R_i > 0,\ R_i \sim N(1, s_i)$ for multiplicative parameters. Parameters are defined as additive or multiplicative as part of the model definition, following the intuition that some parameters (e.g. reversal potential) have an additive effect on model behaviour, while others (e.g. current densities) have a multiplicative effect. The standard deviation s_i of the random factors R_i depend on a model-defined step size σ_i, a global step size modifier $\hat{\sigma}$ decaying with a half-life of λ epochs, and the sensitivity $J_i(S_{e+1})$, i.e. $s_i = J_i \sigma_i \hat{\sigma} 2^{-e/\lambda}$. While the decaying step size helps to ensure that the searched parameter space volume contracts as

the models become more refined, the critical inclusion of $\mathbf{J}(S_{e+1})$ focuses the search on those parameters that are influential on the next evaluation's fitness criterion, while perturbing comparatively irrelevant parameters only little.

2.3.2.2 Differential evolution

As an alternative to the GA, which requires a large number of metaparameters (such as Cr, $\boldsymbol{\sigma}$, λ and others), I also use a differential evolution (DE) algorithm (Storn & Price, 1997; Das & Suganthan, 2011; Buhry *et al.*, 2012), modified to self-adapt its strategy and metaparameters as detailed by (Qin *et al.*, 2009). In DE, models compete not against the entire population, but only against their parent, replacing the parent if they perform better, and being discarded in favour of a new offspring model otherwise. Offspring generation in DE is performed by a variety of crossover-like techniques, rather than random mutation. Models are treated as vectors in parameter space; to generate a new model, a donor vector $\boldsymbol{v}$ is created by combining a target vector with the scaled difference between one or more pairs of (other) vectors. The choice of target vectors depends on the particular strategy used and is at least in part random. I follow the authors of the self-adapting DE (Qin *et al.*, 2009) in using the mutation strategies rand/1, rand/2, rand-to-best/2 and current-to-rand/1. Let $\boldsymbol{v}_i$ be the donor vector for a parent model $\boldsymbol{p}_i$, let $r1 \ldots r5$ be mutually exclusive model indices $\neq i$, let $F \sim N(0.5, 0.3)$ and $K \in [0, 1]$ be scale factors drawn randomly for each donor vector, and let $\odot$ denote element-wise multiplication. The strategies are defined as follows:

- rand/1: $\boldsymbol{v}_i = \boldsymbol{p}_{r1} + F \cdot (\boldsymbol{p}_{r2} - \boldsymbol{p}_{r3}) \odot \mathbf{J}$
- rand/2: $\boldsymbol{v}_i = \boldsymbol{p}_{r1} + F \cdot (\boldsymbol{p}_{r2} - \boldsymbol{p}_{r3} + \boldsymbol{p}_{r4} - \boldsymbol{p}_{r5}) \odot \mathbf{J}$
- rand-to-best/2: $\boldsymbol{v}_i = \boldsymbol{p}_i + F \cdot (\boldsymbol{p}_{best} - \boldsymbol{p}_i + \boldsymbol{p}_{r1} - \boldsymbol{p}_{r2} + \boldsymbol{p}_{r3} - \boldsymbol{p}_{r4}) \odot \mathbf{J}$
- current-to-rand/1: $\boldsymbol{v}_i = \boldsymbol{p}_i + (K \cdot (\boldsymbol{p}_{r1} - \boldsymbol{p}_i) + F \cdot (\boldsymbol{p}_{r2} - \boldsymbol{p}_{r3})) \odot \mathbf{J}$

Note the inclusion of $\mathbf{J}(S_{e+1})$ to again scale the mutation to the sensitivity of the next stimulus. The donor vector $\boldsymbol{v}_i$ is then combined with the parent vector $\boldsymbol{p}_i$ with element-wise binary crossover to yield the offspring $\boldsymbol{x}_i$, i.e.,

$$\mathrm{x}_{i,j} = \begin{cases} v_{i,j}, & \text{if } \mathrm{rand}[0,1] < Cr_i(e) \\ p_{i,j}, & \text{otherwise.} \end{cases}$$

Since the fitness function differs from one epoch to the next, both parent and offspring are then evaluated under the new stimulus/observation pair, and the better-performing of the two is deemed successful and retained as parent for the next epoch.

Each new candidate is generated according to one of the four mutation strategies, the choice of which is random with an initial probability $p_k = \frac{1}{4},\ k \in \{1,2,3,4\}$ in the first epoch. In subsequent epochs, the number of successful (ns_k) and unsuccessful (nf_k) offspring is recorded for each strategy k. Unlike (Qin *et al.*, 2009), who keep a finite-size memory of ns_k and nf_k, I simply calculate decaying running averages $\widetilde{ns_k}$ and $\widetilde{nf_k}$ with a decay period λ such that $(\lambda+1)\widetilde{ns_k}(e+1) = \lambda\widetilde{ns_k}(e) + ns_k(e)$ and equivalently for $\widetilde{nf_k}$. This provides a similar functionality as a finite-size memory, but is easier and more efficient to implement. The probability p_k of choosing strategy k is then proportional to the success rates, i.e., $p_k = \frac{s_k}{\sum_k s_k}$, where $s_k = \frac{\widetilde{ns_k}}{\widetilde{ns_k} + \widetilde{nf_k}}$, so that the algorithm self-adapts to choose more successful strategies more often. The crossover rate $Cr_i(e) \sim N(\widetilde{Cr}, 0.1)$ is governed in analogous manner by a decaying, running average $\widetilde{Cr}$ of the mean crossover rate of successful offspring.

As a consequence of both self-adaptation and a purely crossover-defined mutation operator, this algorithm has only two free metaparameters, the population size and the self-adaptation decay constant λ. Particularly with the long fitting runs I present in chapter 3, I do not expect the algorithm's performance to be very sensitive to the choice of λ. Population size, on the other hand, can be very

influential on how fast and how robustly the algorithm converges (Gämperle *et al.*, 2002), although I have found no indication that large populations hinder convergence. With the level of high-performance parallel computation I use (see section 2.4 below), I have invariably used populations orders of magnitude larger than the dimensionality of parameter space and am confident that population size is therefore a negligible metaparameter.

2.4 Implementation details

2.4.1 General overview

Since part of my intention with this project has been to provide the neuroscience community with a readily applicable tool for the task of model optimisation, I developed all the algorithms described in this chapter into a unified software application called RTDO (real-time dynamic observer, so named for historic reasons). The software is written in C++ and uses the Qt framework to provide a graphical user interface. It allows access to a large number of settings for each of the algorithmic steps detailed above, provides an interface to run these steps and perform related analysis procedures, supplies data visualisations with the QcustomPlot library after and, in the case of model optimisation proper, while the algorithms are running, and logs and saves (almost) every computation for later reference and use. Some of the figures in this thesis are produced by the software.

RTDO is built for and used with the GNU/Linux operating system (CentOS 7, 64-bit) and run on a Dell Poweredge T630 with 32 GB of RAM and two Intel Xeon E5-2623 v3 processors with 8 cores each running at 3 GHz. Most of the model simulations, however, are offloaded onto an NVIDIA Tesla K40c graphics processing unit (GPU) as described below. The software environment includes gcc 4.8.5 and NVIDIA CUDA 9.1. Finally, for access to live electrophysiology data, I use a National Instruments PCIe-6251 data acquisition card driven by the open-source comedi driver.

The software requires a neuron model to be specified according to a custom XML format that describes the equations governing the model, its state variables, any fixed parameters, as well as, of course, the parameters to be optimised, their type, initial values and permissible value range. This model specification is supplied to RTDO at startup and compiled from within the software into a dynamically loadable module that contains all the model-specific computational routines, which is then loaded into RTDO and can be reused in subsequent runs. A direct consequence of this process is that any model that can be described in terms of the XML format can be optimised in the fashion described in this thesis; adding a new model is not a matter of programming, but simply of casting the model into the right format.

The model compilation step makes heavy use of GeNN (Yavuz *et al.*, 2016), from which version 2.2.2 is integrated into the RTDO code. Briefly, GeNN (GPU-enhanced Neuronal Networks) is a framework that turns a user-defined, C++-based model description into C++/CUDA code to simulate a user-defined neuronal network on a GPU. In RTDO, its primary use is to provide the compilation machinery and some of the details of GPU setup and use. The code produced jointly by RTDO and GeNN allows parallel simulation of a large number of models on the GPU, which is key to running any of the evolutionary algorithms outlined above at a reasonable speed.

2.4.2 Parallel computation

In GeNN, the particulars of a model's numerical integration are left up to the user, with the intent that the user provide the code for one integration interval, between which GeNN deals with communicating synaptic potentials or spikes between models. Each interval corresponds to one launch of a so-called kernel, i.e. the function that runs on the GPU and updates the internal state of all models. At the start of each kernel launch, and thus at each interval, variables need to be loaded into kernel memory, which causes a short delay. In the typical GeNN

use case, this is a minor issue, as only a model's state variables need to be loaded, while model parameters are baked into the simulation code as hard-coded values. In RTDO use, however, all fitted parameters are variables in GeNN terminology and thus need to be loaded at kernel launch. With even modestly sized models, this slows integration down to a crawl unless very large intervals are chosen. However, since parameters are never changed during a stimulus, most of these memory accesses are to effectively constant values. Furthermore, there is only limited (and no network-like) need for communication between models. Therefore, instead of supplying GeNN with code for a single interval, RTDO wraps integration in a loop over the entire stimulus duration and provides models with additional information about what to do, including a description of the stimulus/observation pair. Thus, it takes only a single kernel call and memory access to simulate the entire stimulus.

For integration proper, RTDO includes a choice of forward Euler integration, a fixed-step 4^{th}-order Runge-Kutta scheme (RK4), as well as an adaptive step size $4^{th}/5^{th}$ order Runge-Kutta-Fehlberg scheme (RKF45) (Fehlberg, 1970). Largely useless for the short intervals in the typical GeNN use case, step size adaptation is very efficient for the long unobserved stimulus sections in RTDO. Conversely, during observed sections, samples (e.g. current values) need to be generated at fixed intervals, so RKF45 has a smaller advantage over the fixed-step schemes here.

In NVIDIA CUDA, code is executed in parallel in so-called warps, or groups of 32 threads, that perform the same instruction on separate data. When threads within a warp diverge, e.g. during branching statements, the entire warp is forced to process each branch in turn, reducing the benefit of parallelism. Because step size adaptation causes the integration code to be called a varying number of times, it almost invariably causes such divergence; some threads will arrive at the end point of integration in fewer iterations than others and remain idle until the remainder of the warp has caught up with them. In RTDO, although each thread

can have its own stimulus, most applications – including all those described in this chapter – call for only one stimulus/observation pair in each warp. As a consequence, step sizes are unlikely to differ very widely, particularly in voltage clamp, and thread divergence has only limited impact on RKF45 performance. A possible exception to this is current clamp integration (see section 5.2), where easily integrated quiescent models and more computationally demanding spiking models may well coexist within the same warp, slowing execution to the slowest thread in the warp. However, I have not observed a significant overall slow-down over several hours of current clamp simulation; besides, the worst-case RKF45 performance is little slower than a comparable RK4 execution with the same minimum step size.

In addition to numerical integration, several other tasks are executed on the GPU, including most of the algorithms described in this chapter, with the notable exception of the reproduction and mutation operators during stimulus generation and model optimisation. Since writing high-performance CUDA code is a difficult process and an art in its own right, I will not dwell on the details here, but refer the interested reader to the src/cuda/ directory of the RTDO code, and outline only some general principles that I have followed during development. Firstly, wherever possible, I have used existing libraries for common tasks. For example, I use the Thrust library (Bell & Hoberock, 2012) to compute the correlation values for robustness analysis (section 2.2.4). Secondly, I have put considerable thought into organising the algorithms' implementations such that memory locality is respected in order to allow high-throughput coalesced memory access. Where appropriate, I make use of CUDA's "constant" memory area to store data that is used across all of a kernel's threads, such as the reference data during model optimisation. Thirdly, to exchange data between threads, I use so-called "warp shuffle" functions, which allow threads within a warp to access each other's registers as if they were their own, without requiring explicit synchronisation. Particularly for the cluster algorithm (section 2.2.2.2.2), in which, for each

stimulus, after generating the time series of elementary effects for each of the k parameters, the elementary effect vectors $\mathbf{J}(t) \in \mathbb{R}^k$ must be compared all-to-all within the time series, using a highly efficient inter-thread communication scheme is critical to keeping total running time within acceptable limits. Finally, to keep the code readable and maintainable, I have separated the algorithm implementations from model-specific code as far as possible, such that only a small fraction of the CUDA code, particularly model-specific data structure definitions and low-level functionality such as common operations over k-dimensional vectors of parameters or parameter-related values, is code-generated through GeNN, while most of the higher-level functionality is written in standard C++/ CUDA.

2.5 Conclusion

In this chapter, I have introduced the basic idea behind the model optimisation approach developed in this thesis, pointing out the possibility of fitting subsets of model parameters to data that are relatively invariant to other parameter influences, and thereby avoiding, to some extent, both the curse of dimensionality and the problem of model degeneracy. I have argued that classical voltage clamp protocols often do something similar, albeit at a different level of abstraction, isolating and probing currents rather than model parameters. Noting the advantages of voltage clamp, I have motivated its use in this project, mitigating the higher complexity this entails by including a model of the clamping amplifier in all simulations. I then detailed the algorithms I have developed to perform model optimisation starting from a given model structure. Broadly, the process involves a computationally heavy, but data-independent search for good stimuli (section 2.2), and a computationally much less intensive, and thus faster, search for parameter sets that fit data gathered using these stimuli (section 2.3). While the latter is a relatively straightforward task with few complicating considerations and many examples of at least broadly similar work in the literature, the former is

both complex and ambitious, constituting the major push towards the Unknown presented in this thesis. Partly for this reason, I have developed not just a single approach, but tried several alternative techniques with the same aim, hoping that one might prove successful. In some cases (stimulus generation proper, section 2.2.3, and robustness screening, section 2.2.4), this takes the form of an evolution from simple towards more complex and a priori more promising algorithms, while in others (quantifying parameter isolation, section 2.2.2, the final selection strategy, section 2.2.5, as well as model optimisation itself, section 2.3), I offer two or three techniques that, before experimental verification, appear comparable.

Due to the complexity of both the problem and the proposed solutions, assessing what works is frustratingly difficult. There is no particularly good way to test any of the steps detailed in this chapter on its own: We cannot independently verify the quality of a parameter isolation metric or of a set of stimuli without an adequate fitting algorithm, we cannot judge the quality of a fitting algorithm independently of the stimuli that are used to collect its input data, and we cannot judge thousands of stimuli for their robustness to the noise and parameter variability in real data without gathering enormous amounts of such data. Worse yet, because the primary goal of the project – optimising models to individual neurons, rather than to a potentially disparate set of neurons – has not been achieved at the level of detail aimed for here, independent verification is in fact impossible. Nevertheless, this thesis does not, of course, stand with a description of a novel method alone. The next chapter lays the groundwork for an intermediate step, or rather, a proof of concept using data from a simple biophysical system that can be accurately described without resorting to pharmacological intervention, namely, *Xenopus* oocytes. With the models from the next chapter in hand, we will then return to the MOSTIPS method in chapter 4, where I present my attempts at showing that the concept is sound and that the algorithms work as intended, and examine the results of my experiments in silico and in vitro.

3 Potassium channel models

3.1 Introduction

To show that the MOSTIPS method works, not just in a theoretical or simulated setting, but with real data, I needed a system that is suitably similar to a real neuron, but provides a high level of access to other parameter estimation procedures against which could serve as benchmarks for the new method. For this purpose, I turned to *Xenopus* oocytes, which are essentially – for the purposes of the experiments herein – passive, spherical membrane compartments that can be augmented with any given ion channel protein by simply inserting the appropriate RNA into the cell. Due particularly to their simple geometry, their large size, and their lack of uncontrollable active properties, oocytes provide an excellent opportunity to measure, or precisely estimate, the properties of any expressed channels. By expressing only one or two channel constructs, we can thus engineer a system that can serve as a benchmark for the MOSTIPS method.

In this chapter, I introduce the models of the potassium channels which I have used in my oocyte work. While these models were initially based on existing literature and would not normally require a chapter of their own, over the course of early experiments, I found that they unsatisfactorily described the currents I recorded. I therefore set about improving them, adjusting them to the data I had collected. Here, I document both the original models and how I had built them, and the data that demonstrated a model mismatch, as well as my attempt at improving the models to a more adequate standard.

All model improvements were done without direct reference to the MOSTIPS method, using instead the data that I had gathered by classical methods. Since the same data and analysis methods were also used to provide validation values, or the “ground truth”, against which I qualified the success of the new method, much of this chapter also serves in part as a methods section for the next

chapter, where I present MOSTIPS fitting results and relate them back to the "classical" parameter estimates detailed herein.

3.2 Methods

3.2.1 Oocyte preparation

All experimental work involving *Xenopus laevis* oocytes was conducted in Dr. Anthony Lewis's Ion Channel Laboratory at the University of Portsmouth. While I personally conducted some of the RNA injections, all other oocyte preparation work was done by staff of the Xenopus Resource Centre (provision of ovaries), and Dr. Ruolin Ma, then a PhD candidate in the lab, who prepared the oocytes from defolliculation to incubation, as well as preparing the cRNA at appropriate concentrations. The term cRNA (complementary RNA) refers to RNA strands with the same base sequence as its corresponding original messenger RNA, derived by copying the latter to a complementary DNA sequence template, then copying that back to RNA for injection.

Briefly, ovaries were harvested from anaesthetised *X. laevis* females. Oocytes were coarsely defolliculated with forceps, washed in OR-2 solution (in mM: NaCl 82.5, KCl 2.5, $MgCl_2$ 1, HEPES 10, calibrated to pH 7.4 with NaOH), and stirred for 1-2 hours in OR-2 with 1.6 mg/ml collagenase to complete defolliculation. The oocytes were then washed thoroughly in 1 M $CaCl_2$ and placed in ND96 (in mM: NaCl 96, KCl 2, $MgCl_2$ 1, $CaCl_2$ 1.8, HEPES 10, 1% Streptomycin, 0.1% Gentamycin, calibrated to pH 7.6 with NaOH) for recovery and incubation. Healthy stage V and VI oocytes were manually selected, injected with between 0.05 and 2 ng of the appropriate cRNA, and left to incubate for 1-2 days. Oocytes were prepared for one of three expression profiles by injecting cRNA of either Kv1.4, or Kv2.1, or both.

3.2.2 Electrophysiology

For data collection, two-electrode voltage clamp was performed at room temperature using an OC-725C amplifier (Warner Instruments), Digidata 1440A and pClamp 10 (Molecular Devices). Oocytes were transferred to a recording chamber perfused with a bath solution (in mM: NaCl 96, KCl 4, $MgCl_2$ 1, $CaCl_2$ 0.3, HEPES 10, calibrated to pH 7.6 with Tris base). Borosilicate glass micropipettes were pulled to tip resistances around 0.5 MΩ and filled with 3 M KCl. After checking the current electrode's resistance, I impaled the oocytes, activated voltage clamp at -80 mV with low initial gain, increasing the gain to 2000 over 1-2 seconds. Oocytes with large leak currents or no perceptible potassium current were discarded. After holding the oocytes at -80 mV for 1-3 minutes to allow the cells to recover from impalement, recording was initiated from within the pClamp software. Data were saved in the software's default ABF format. For further processing and analysis, the recordings were converted to the plain-text ATF file format and transferred to my lab at Sussex University using a portable hard disk drive.

3.2.3 Stimulus protocols

The following stimulus protocols were chained together, with a holding period of 10 s between each stimulation and a recording frequency of 40 kHz: First, to probe leak conductance and current activation, a series of 1-second steps from -80 mV to [-120, -110, ..., +60] mV; second, to probe deactivation, a brief (Kv1.4 only: 10 ms; Kv2.1 and joint expression: 100 ms) step to +60 mV followed by 1-second steps to [-120, -115, ..., -40] mV; and finally, to probe capacitance, a series of 5 square pulses of 50 ms duration from -80 mV to -90 mV and back. All three protocols are shown in Figure 3.1. Following these generic protocols, algorithmically generated stimulus protocols corresponding to the channels expressed in the oocyte were recorded at 10 kHz. No leak subtraction or filtering was performed for any of the protocols, as estimation of the leak conductance

and capacitance formed part of the analysis I intended to do. In a few cases, recording had to be stopped before all data was collected, as the cell or voltage clamp had degraded beyond the point of usefulness.

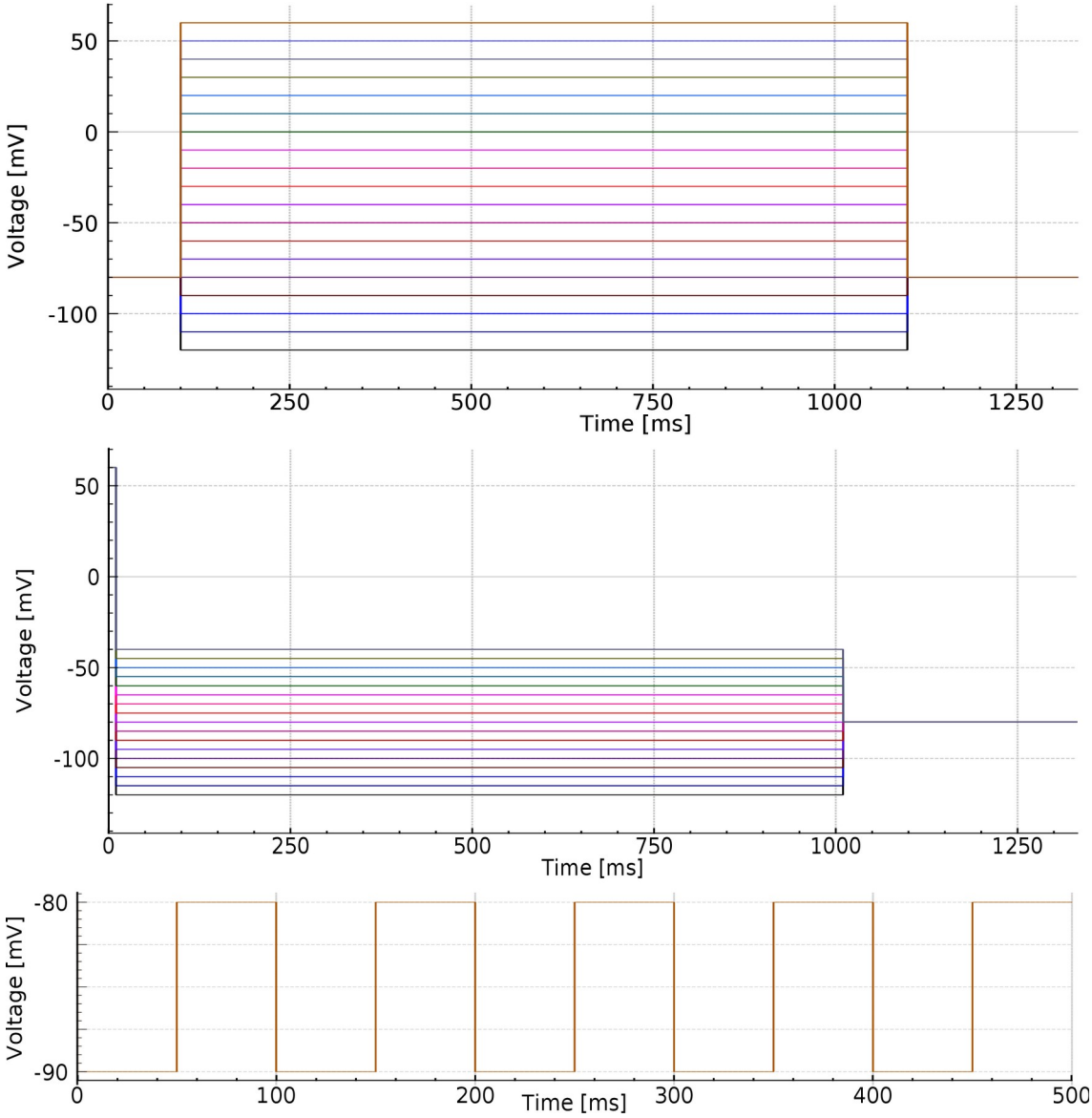


Figure 3.1: Stimulus protocol command voltages, top to bottom: Activation, tail currents, capacitance. Holding voltage for all three protocols was -80 mV. The prepulse to +60 mV in the tail current protocol (middle) was 10 ms for Kv1.4 (as shown here) and 100 ms for Kv2.1 and joint expression, with the remainder of the stimulus pushed back by 90 ms in those cases. In the capacitance protocol (bottom), due to a technical oversight, the recording of the final step at 450 ms was cut short, such that only 9 complete steps could be used for analysis.

3.2.4 Software

All parameter estimation and fitting procedures described in this chapter were implemented in an evolving combination of Jupyter notebooks (Kluyver *et al.*, 2016) and raw python scripts. I used the former for developing the fitting and estimation methods, running sanity checks on the data, the estimation processes, and the outcomes, as well as for visualisation, including most of the figures shown below. Raw python scripts were used to implement repeatedly used procedures such as loading recordings as well as some of the fitting and estimation methods once they were fully formalised and tested. I relied on a number of python libraries, including StimFit 0.15.5 (Guzman *et al.*, 2014) to read the ATF records, NumPy and SciPy (Oliphant, 2007) for data manipulation, Matplotlib (Hunter, 2007) for plotting, and the multiprocessing library to speed up full-record fitting. Least-squares fitting was performed using the scipy.optimize.least_squares function, using either the default Trust Region Reflective or the Levenberg-Marquardt method.

3.3 Building single-component models from literature

The original plan was to turn oocytes into a facsimile of simple neurons by jointly expressing sodium and potassium channels in approximately balanced quantities. However, since the Ion Channel lab is specialised in potassium channels, no sodium channels were available there, and procuring them from other sources turned out to be more difficult than anticipated. Therefore, I focussed my attention solely on two potassium channels, Kv1.4 and Kv2.1. These were chosen based on availability of their cRNA in Dr. Anthony Lewis's lab, as well as their comparatively good documentation in literature.

Both neuroscientists and cardiologists are interested in ion channels as key regulators of the behaviour of their respective cells of interest. Both Kv2.1 and Kv1.4 are present in both heart and brain tissue and are therefore studied in both fields. As a result, there is a considerable number of papers characterising these chan-

nels in circumstances similar to the ones I was preparing for, i.e., expressing cRNA derived from rat brain or cardiac tissue in *Xenopus* oocytes.

I hoped to harness the characterisations of these channels in literature to quickly build models of these channels that I could then use for classical and novel parameter estimation methods. However, the manner in which the channels are characterised generally does not translate directly into a model. For example, while we can assume potassium channels to have four activation gates, activation currents are usually fitted with a single exponential, such that translating reported values into a Hodgkin-Huxley format requires approximating such exponentials with fourth-order activation kinetics. In addition, not all data are reported numerically, particularly in older papers; where they were only available as figures, I used WebPlotDigitizer (Rohatgi, 2019) to extract the data I needed. In the following, I describe the process of building these models from the available data in literature.

3.3.1 Kv2.1

Kv2.1 is a delayed rectifier potassium current found in mammalian cardiac (Dixon & McKinnon, 1994; Yang *et al.*, 1994) and neural (Frech *et al.*, 1989; Murakoshi & Trimmer, 1999) tissue, also referred to as drk1, shab and KCNB1 (Ranjan & Khanna, 2019). Because Kv2.1 shows a slow, but pronounced inactivation (Frech *et al.*, 1989), I modelled the conductance with the usual four activation gates and an additional slow inactivation gate (Armstrong, 1969; Hille, 1991), mirroring other modellers' choices (Günay *et al.*, 2008):

$$I_K = \bar{g} n^4 h (V - E_K) \qquad \text{(Equation 3.1)}$$

$$\frac{dn}{dt} = \frac{n_\infty - n}{\tau_n} \qquad \text{(Equation 3.2)}$$

$$\frac{dh}{dt} = \frac{h_\infty - h}{\tau_h} \qquad \text{(Equation 3.3)}$$

First-order fits for steady-state activation were readily available from (VanDongen *et al.*, 1990; Kerschensteiner & Stocker, 1999) using rat cRNA in oocytes, and from (Shi *et al.*, 1994) in COS-1 cells, which are mammalian non-neuronal fibroblasts (Gluzman, 1981). Taken together, these data paint a fairly consistent picture, see Figure 3.2, and were used to hand-fit the fourth-order Boltzmann function in Equation 3.4.

Inactivation data were taken from the same sources, as well as from (Li *et al.*, 2015) and (Klemic *et al.*, 1998). Here, since inactivation is a first-order process, no translation to a different order is required, which made hand-fitting somewhat easier; the resulting steady-state function is shown in Figure 3.2 and Equation 3.5.

$$n_\infty = \frac{1}{1 + \exp(\frac{-24-V}{10.5})} \qquad \text{(Equation 3.4}^2\text{)}$$

$$h_\infty = \frac{1}{1 + \exp(\frac{-26-V}{-6})} \qquad \text{(Equation 3.5)}$$

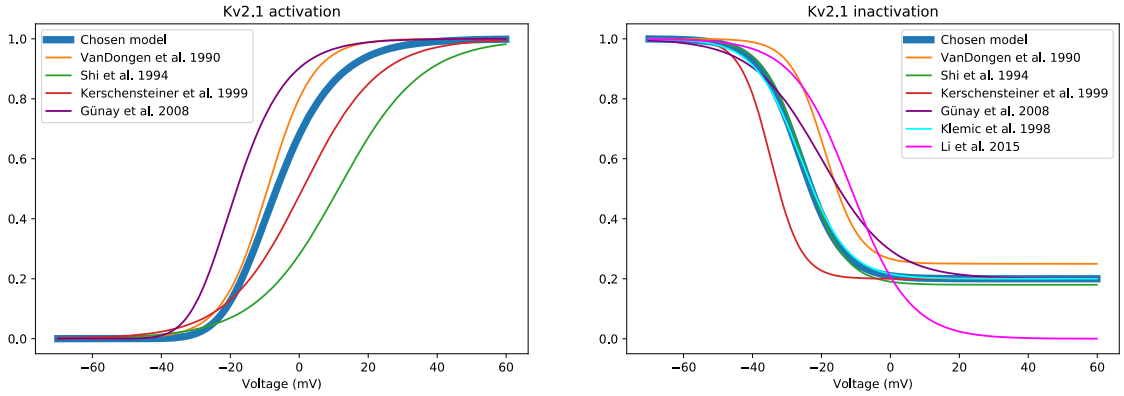


Figure 3.2: Kv2.1 activation and inactivation steady state curves, from various sources, together with the hand-fitted chosen model curves. The activation curves for VanDongen, Shi and Kerschensteiner are first-order Boltzmann curves, while the Günay and chosen model are fourth-order curves.

2 Note that steady-state variables $x_\infty \in [0, 1]$ are dimensionless. Here, and in subsequent voltage-dependent steady-state equations unless otherwise noted, V is the dimensionless value of the membrane potential expressed in mV.

Time course data were more difficult to find, with reliable activation data from oocyte expression only in (Kerschensteiner & Stocker, 1999), and much faster kinetics reported in COS-1 (Li *et al.*, 2015) and modelled in (Günay *et al.*, 2008), both of which are of doubtful use, as the kinetics are dependent on secondary subunits and other cofactors that are likely present in mammalian cells, even non-neuronal, but not in *Xenopus* oocytes. I therefore extracted oocyte-derived first-order time course data from Figure 1 in (Kerschensteiner & Stocker, 1999), fitted the resulting exponentials with fourth-order approximations over 200 ms, then hand-fitted Equation 3.6 to these data points, with the resulting curve shown in Figure 3.3. None of the other papers used for steady-state activation reported time course data, other than time-to-peak or figures of traces, neither of which seemed a reliable way of improving my estimate.

$$\tau_n = 5 + \frac{110}{\exp(\frac{V+36}{18}) + \exp(\frac{V+36}{-17})} \qquad \text{(Equation 3.6}^3\text{)}$$

$$\tau_h = 400 + \frac{3000}{1 + \exp(\frac{V+80}{-15})} \qquad \text{(Equation 3.7)}$$

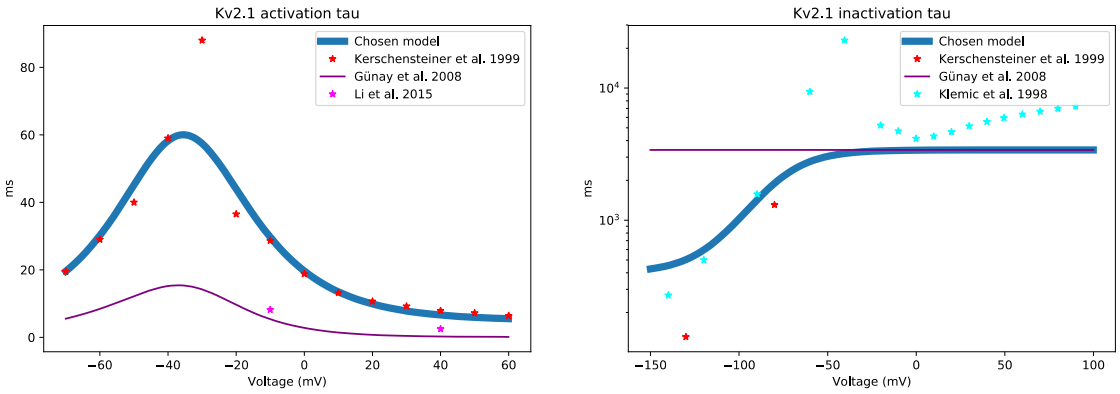


Figure 3.3: Kv2.1 time course data and models

For the inactivation time course, (Klemic *et al.*, 1998) provided a decent starting point, though here, too, I had to extract graphed data (from figure 3C, ibid.). Potassium channel inactivation may arise from several distinct mechanisms (Klemic

3 Note that here, and in subsequent equations unless otherwise noted, time constants τ_x are dimensionless values implicitly expressed in ms. As in steady-state equations, V is the dimensionless value of the membrane potential in mV.

et al., 1998; Kurata & Fedida, 2006), including closed-state inactivation that cannot be modelled in the Hodgkin-Huxley formalism, and is perhaps for this reason not modelled consistently (e.g. in classical Hodgkin-Huxley fashion analogous to sodium inactivation, in piecewise fashion (Destexhe & Huguenard, 2000), or even without any voltage dependence (Günay *et al.*, 2008)). My approach falls somewhere in between, trying to achieve a pronounced voltage dependence for recovery from inactivation at strongly negative voltages, while remaining approximately stable for higher voltages. Though Equation 3.7 does not capture the reported kinetics particularly well, I judged this model to be adequate for my purposes.

3.3.2 Kv1.4

Kv1.4 is an A-type potassium current found, like Kv2.1, in both cardiac muscle (Comer *et al.*, 1994; Rasmusson *et al.*, 1995; Wickenden *et al.*, 1999) and in the central nervous system (Stühmer *et al.*, 1989; Stephens *et al.*, 1996) and is also referred to as RCK4 and KCNA4 (Ranjan *et al.*, 2019). I constructed my initial model of this channel analogously to the Kv2.1 model, i.e. according to equations 3.1 to 3.3.

Luckily, a fair number of studies expressing Kv1.4 in oocytes is available. In particular, I relied on (Stühmer *et al.*, 1989; Nunoki *et al.*, 1994; McIntosh *et al.*, 1997; Hashimoto *et al.*, 2000; Hagiwara *et al.*, 2003) for both activation and inactivation steady-state data, with (Rettig *et al.*, 1994) adding further data for inactivation. Correcting again for fourth-order kinetics, I hand-fitted Boltzmann curves to the various data as shown in Figure 3.4, resulting in the steady-state functions in Equations 3.8 and 3.9.

$$n_\infty = \frac{1}{1+\exp(\frac{-55-V}{20})} \qquad \text{(Equation 3.8)}$$

$$h_\infty = 0.05 + \frac{0.95}{1+\exp(\frac{-48-V}{-5})} \qquad \text{(Equation 3.9)}$$

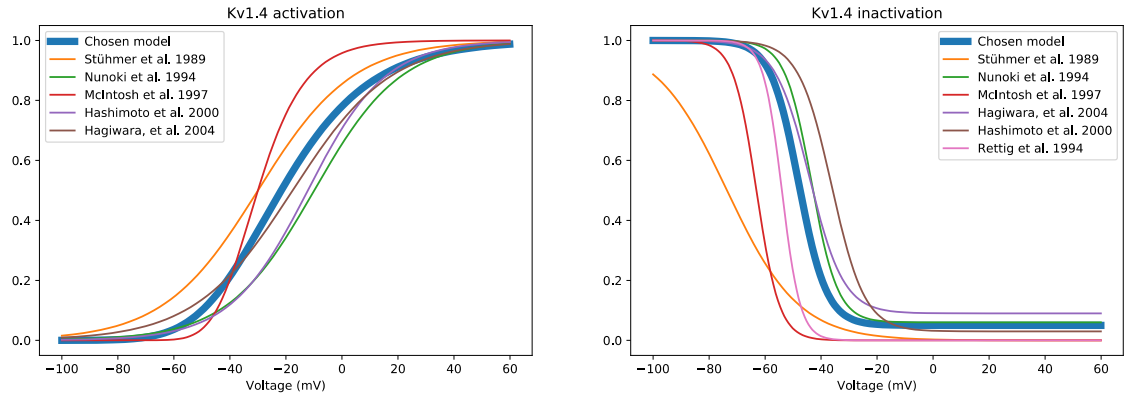


Figure 3.4: Kv1.4 activation and inactivation

Time course data were again a little more difficult, particularly for activation, which is faster and more strongly affected by inactivation than that of Kv2.1. Finding no useful data for activation, I lifted deactivation data from figure 7C in (McIntosh *et al.*, 1997), which was generated with a brief pulse to +60 mV to activate the conductance, then stepping down to various levels and observing the tail currents, resulting in single exponential decay time constants. To these exponentials, I fitted a Hodgkin-Huxley conductance equation of the form $g = (n_\infty - (n_\infty - 1)\exp(-t/\tau_4))^4 h_\infty$, using for n_∞ the fraction of the maximum current as reported in figure 1D (ibid.) and estimating h_∞ from figure 4A (ibid.). The resulting data points – if we can still call them that after so much uncertain processing – are shown in Figure 3.5 together with my attempt at fitting something sensible to it. I should point out that the points below -60 mV are almost entirely fictitious, being based on guesses of h_∞ rather than actual data, and should therefore be taken with a large grain of salt. The resulting model is shown in Equation 3.10.

$$\tau_n = 2 + \frac{102}{\exp(\frac{-45-V}{-6}) + \exp(\frac{-45-V}{40})} \qquad \text{(Equation 3.10)}$$

$$\tau_h = 35 + \frac{10^4}{1 + \exp(\frac{-72-V}{-7})} \qquad \text{(Equation 3.11)}$$

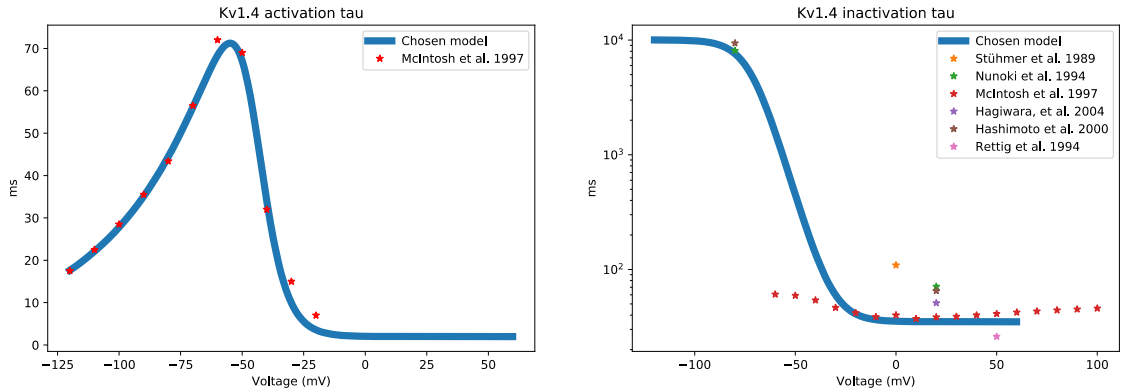


Figure 3.5: Kv1.4 time course data and model

Inactivation is somewhat better documented due to its prominence in the channel's currents, though good data are still lacking, again due to the uncertain interplay between different types of inactivation as well as recovery from it. Regardless, I gave my best attempt at matching all data available to me, resulting in Equation 3.11.

3.4 Parameter estimation and fitting procedures

3.4.1 Classical estimation of passive parameters

Having constructed these models to the best of my knowledge, I then proceeded to collect and analyse data from oocytes expressing these channels. The potassium channels expressed beautifully in the oocytes, giving rise to large currents with the expected features, as seen in a typical example of a Kv2.1 current in Figure 3.6. In the following, I describe how I estimated the "easy" parameters of interest – the leak conductance and equilibrium potential, capacitance, and potassium equilibrium potential – using this recording for illustration. These procedures were the same for oocytes expressing either one or both of the potassium channels, the only difference being the timing of the tail current steps for oocytes expressing Kv1.4 only, as noted in section 3.2.3.

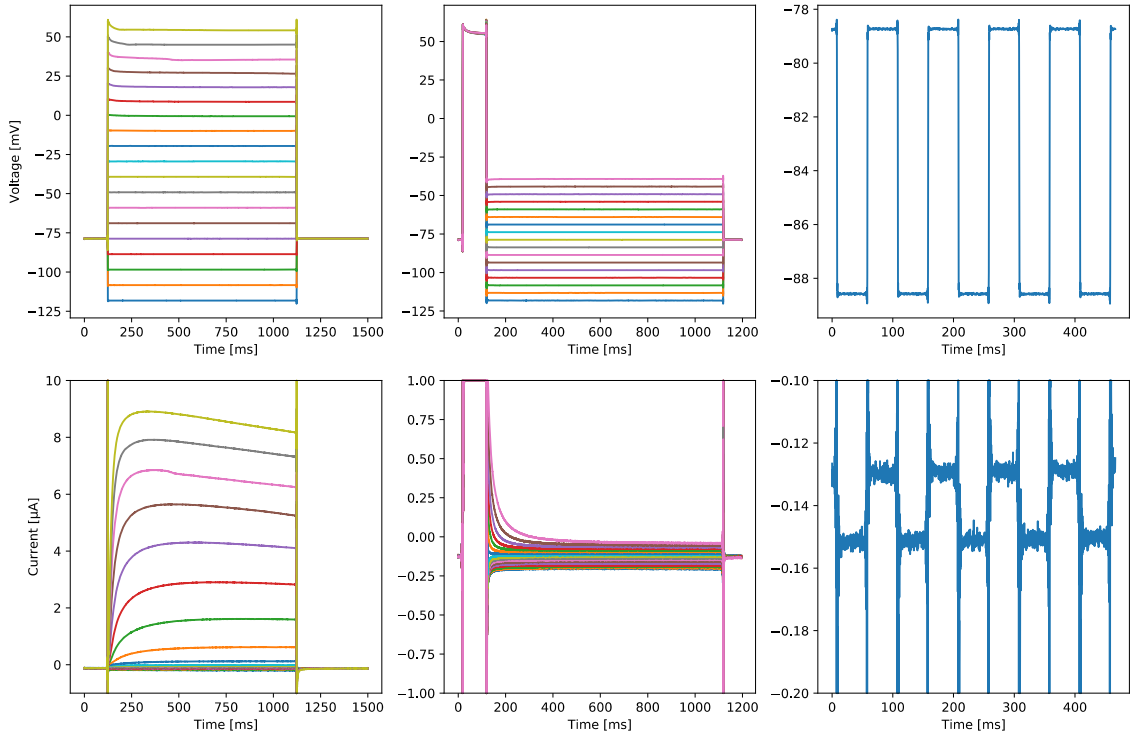


Figure 3.6: Representative example traces of a Kv2.1-expressing oocyte under each of the three voltage clamp protocols. Traces are IIR-filtered, downsampled to 4 kHz and clipped to the plotted range for display only. Top: achieved voltage, bottom: corresponding clamp current. Protocols from left to right: Activation, tail currents, capacitance.

The first analysis step was to estimate the leak conductance. For this, using the activation protocol, I computed the median voltage and current during steps to potentials below -80 mV, where the potassium conductances can safely be assumed absent, and fit a straight line to the resulting points in the current-voltage plane, yielding the equilibrium potential and conductance of the leak current, as shown in Figure 3.7.

Next, I estimated the capacitance using data from the capacitance protocol. Each of the 10 mV steps (in both directions, a total of 9 steps), the first two of which are shown in Figure 3.8, were analysed separately. The achieved voltage step size ΔV was estimated using the median voltage in each step. To account for leak, the median current of the final 12.5 ms of the step was subtracted as a baseline to yield I_{step}. Finally, the capacitance was estimated according to

$C_{step} = \frac{1}{\Delta V} \int I_{step}\, dt$, and the mean C_{step} across all 9 steps recorded. I note that, although the traces in Figure 3.6 look noisy, the integral is computed as a sum over all samples, roughly half of which are negative once the baseline is subtracted, such that noise should not greatly affect the estimate.

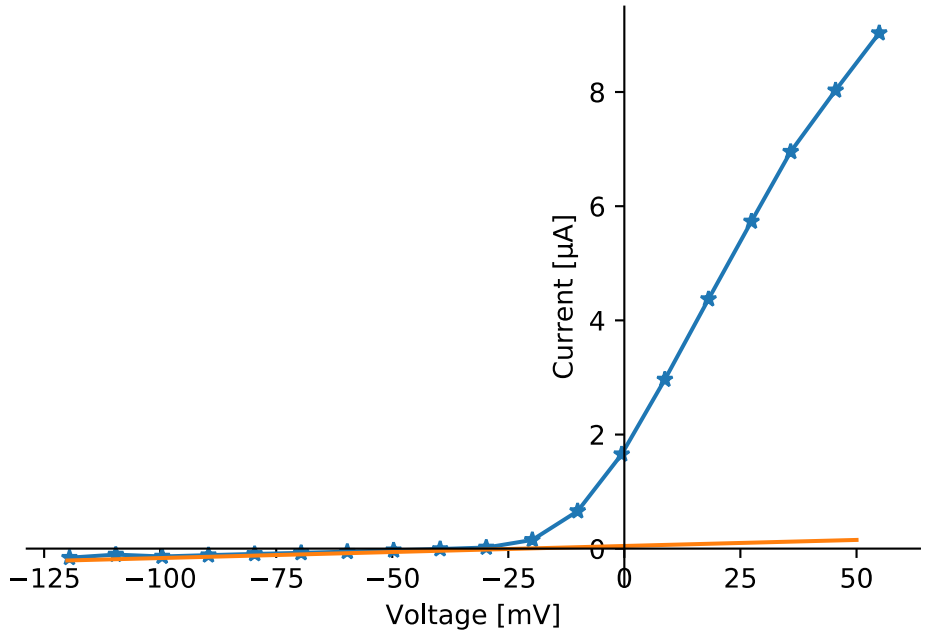


Figure 3.7: I-V plot derived from the activation protocol data in the previous figure. Blue, current maxima during the step; orange, leak current fit (El: -22.4 mV, gl: 2.1 µS)

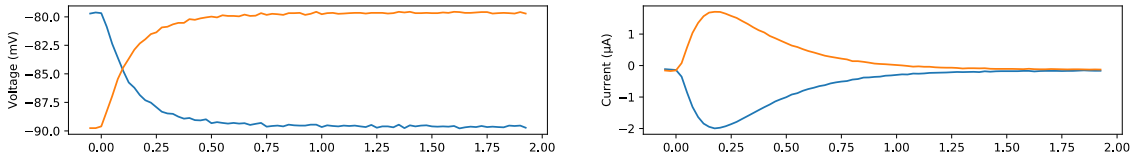


Figure 3.8: Two capacitance protocol steps, showing the first 2 ms of data from the beginning of the step. Left, voltage; right, corresponding current traces. Though the majority of the capacitive current flows within the period shown, the entire 50 ms were used during estimation to avoid bias. The capacitance in this example was estimated at 90.7 nF.

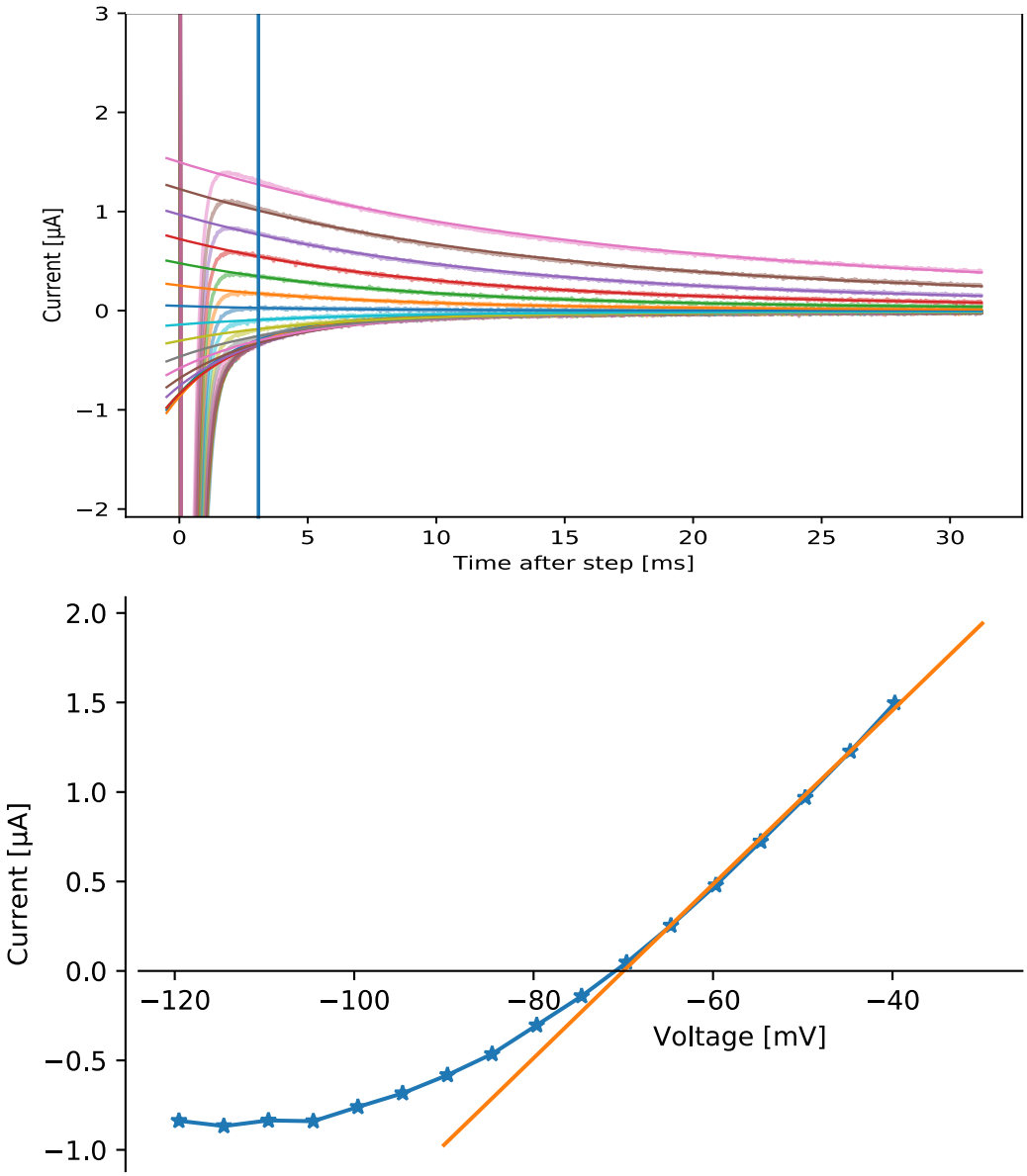


Figure 3.9: Tail current fits. Top: Tail currents and fitted exponentials. The blue vertical line shows the cutoff point designed to protect the fit against the capacitive artefact. Bottom: I-V plot showing the exponentials' values at t=0 (blue) and the straight-line fit to the positive values used to estimate EK, here -70.0 mV.

Finally, I used the tail current protocol to estimate the potassium equilibrium potential, since the I-V plot (Figure 3.7) proved inadequate for this. I cut the tail currents at a safe distance from the capacitive artefact, subtracted the leak current that was expected to be evoked by the median achieved voltage, and fit the remaining current trace with a sum of two exponential decays. The initial ra-

tionale for using a double exponential was a mere matter of achieving a good fit; only later did it become apparent that there are indeed two decaying components with distinct time constants, see section 3.5. The resulting fit (see Figure 3.9) was extrapolated back to the beginning of the step to estimate the current that would have flown, at the median voltage achieved during the fitted portion of the tail currents, with the channels in the largely open state promoted by the prepulse. Since up to step initiation, all traces derive from the same stimulus pattern (i.e., a prepulse to +60 mV), one should expect the resulting tail current-voltage relationship to be linear. There was, however, very consistent evidence of something that looks like rectification, the source of which is not entirely clear to me. It seems plausible that the tail currents are more difficult to fit at lower voltages, since the artefact at the step is greater; perhaps there is also an effect of faster deactivation that is hidden within the artefact. In any case, I could thus not trust the negative currents for a straight-line fit, and so restricted myself to the positive tail currents, fitting a straight line to find the reversal potential EK.

3.4.2 Model-free active parameter estimation

The classical way of finding the maximum potassium conductance would be to use the peak currents of the activation protocol (Figure 3.7) together with the EK estimate to derive a straight-line fit that describes the current-voltage relationship of fully opened channels. However, this would be unreliable, since both Kv1.4 and Kv2.1 are inactivating; the peak currents, therefore, are already reduced below the maximum conductance due to inactivation (Willms *et al.*, 1999; Lee *et al.*, 2006). More importantly still, in oocytes with joint expression of both channels, peak currents alone could not suffice to estimate two maximum conductances.

Thus, instead of estimating $\bar{g}$ from peak activation current, I decided to use the full activation protocol traces, fitting a Hodgkin-Huxley simulation to the data with a least-squares optimisation routine. Furthermore, rather than fully trusting

the literature-derived kinetics and fitting only the maximum conductance, I opted for a more exploratory approach that would allow me to verify or adjust the model kinetics as necessary.

Firstly, I posit a typical model structure of the form

$$I_K = \sum_{c \in components} g_c(V - E_K)$$
$$g_c = \bar{g}_c n_c^4 h_c \qquad \text{(Equation 3.12)}$$

with the activation and inactivation gating variables n_c and h_c following the differential equation

$$\frac{dx}{dt} = \frac{x_\infty - x}{\tau_x} \qquad \text{(Equation 3.13)}$$

Initially, I had no reason to assume more than one component for each channel, but was challenged in that assumption as shown in section 3.5, hence the more general formulation here.

Then, in order to avoid specifying any particular kinetic model (i.e., voltage-dependent functions $x_\infty(V)$ and $\tau_x(V)$), I assume $V = const.$ throughout the linear segments of each stimulus, approximating the voltage during a given step by the median measured membrane potential. This allows me to solve the differential equations for the gating variables explicitly, i.e.,

$$x(t) = x_\infty - (x_\infty - x_0)e^{-t/\tau_x}. \qquad \text{(Equation 3.14)}$$

Since the membrane is held at -80 mV before each protocol sweep, I can assume complete deactivation and deinactivation, i.e., $n_0 = 0$ and $h_0 = 1$, such that g_c can be formulated as

$$g_c(t) = a(1 - e^{-t/\tau_n})^4(h_\infty - (h_\infty - 1)e^{-t/\tau_h}) \qquad \text{(Equation 3.15)}$$

where $a = \bar{g} n_\infty^4$.

To fit this equation to the activation protocol, I prepared the most informative traces, that is, those with clearly detectable currents (command voltage >= -60 mV for Kv1.4, and >= -20 mV for Kv2.1), by subtracting the leak current as estimated above. Then, I least-squares fit Equation 3.15 to each trace separately, yielding trace-specific a, h_∞, τ_n and τ_h. Finally, I assumed a maximum achievable conductance $\bar{g} = \max_{traces}(a)$ and extracted trace-specific $n_\infty = \sqrt[4]{a/\bar{g}}$.

3.4.3 Time constant fitting with the tail current protocol

While the above procedure is sufficient to produce good estimates for the steady-state variables x_∞, which we can safely assume to follow a sigmoidal relationship to membrane potential, the time constants τ_x follow a more complex relationship and are therefore insufficiently constrained at lower voltages. To rectify this situation, I turned to the tail current protocol. Since this protocol has a prepulse before the current of interest, estimating the initial gating variables x_0 is a little trickier. Luckily, the activation protocol fits provide good estimates for the x_∞ and τ_x at the prepulse voltage. Using these estimates together with the median prepulse voltage, I calculated the gating state at the end of the prepulse with Equation 3.15.

Then, using this state as the new x_0, I dropped V to the median test pulse voltage and again fit the current equation to leak-subtracted data, with one important difference: Rather than fitting all kinetic variables, I drew the steady-state and inactivation values from modelled functions as described in more detail below. This left only the activation time constants to fit, which were thereby well constrained.

Finally, combining both activation and tail current protocol fits across all recordings, I decided on voltage-dependent functions for each x_∞ and τ_x by a combination of least-squares fitting and hand-tuning.

3.4.4 Model-constrained full-record fitting

With trustworthy models in hand, I could then turn to finding maximum conductances and kinetic parameters that best fit the recordings from a given oocyte. Notice that, in the preceding section, I fit the values for x_∞ and τ_x for each trace in isolation; now, and for the purposes of validating the MOSTIPS method, I am looking to fit the parameters that govern the kinetic equations $x_\infty(V)$ and $\tau_x(V)$ to an entire record.

To do this, I estimated the passive parameters as described in section 3.4.1, and preprocessed the traces from both activation and tail current protocols by subtracting the expected leak current and setting cutoff points to exclude the capacitive current artefact. Then, I provided a routine to integrate the complete current model (Kv2.1x, Kv1.4x, or both together, as appropriate, see section 3.5) with a forward Euler method at the records' sampling rate of 40 kHz. Initial conditions were assumed as $n_0 = 0$, $h_0 = 1$ as above. From the start of each trace, the expected current was then calculated based on the measured voltage in each sample. Finally, the current residuals in the observed regions (i.e., test pulses after the artefact cutoff) were collated across all traces in both protocols, and fed to a least-squares optimisation routine to fit either just the maximum conductances, or, in the case of the oocytes expressing only Kv2.1, the complete kinetics parameter set, too.

Although some results are shown in section 3.5.3, the main purpose of the full-record fit was, of course, to provide reference parameter values for comparison with the MOSTIPS method, as we will explore in the next chapter.

3.5 Fixing model mismatch with a second component

3.5.1 Single-component fits are qualitatively unsatisfactory

As shown in section 3.3 above, I had considerable difficulty finding good data on the channel kinetics despite the preponderance of relevant literature. As I will demonstrate below, however, the problem went deeper than just a lack of adequate kinetics. Instead, I was forced to concede that the models were not as good as I had hoped. I therefore decided to adjust the models to fit the data I had in hand.

Using the model-free fitting method outlined in section 3.4.2, I fit the single-component models to the activation protocol traces. For Kv2.1 currents, I tried several least-squares methods, and many initial guesses, but invariably ended up with one of two solutions, illustrated in Figure 3.10: The fitting algorithms either exploited the wide boundary conditions I gave them, using uncharacteristically fast inactivation to fit the shape of the current onset while ignoring slow inactivation, or conversely, fitting the slow inactivation correctly at the expense of a very inaccurate current onset.

For Kv1.4 currents, the problem was slightly more subtle, because the fitting algorithms could not ignore the much more pronounced inactivation. However, the fits looked qualitatively mismatched in the same way as the Kv2.1 fits that appropriately accounted for inactivation, namely with too sharp an activation shape, as shown in Figure 3.17. And so, since a single component with n^4 activation could not possibly accurately fit the current onsets I observed in either of the two channels, I decided to add a second component to each of the channel models.

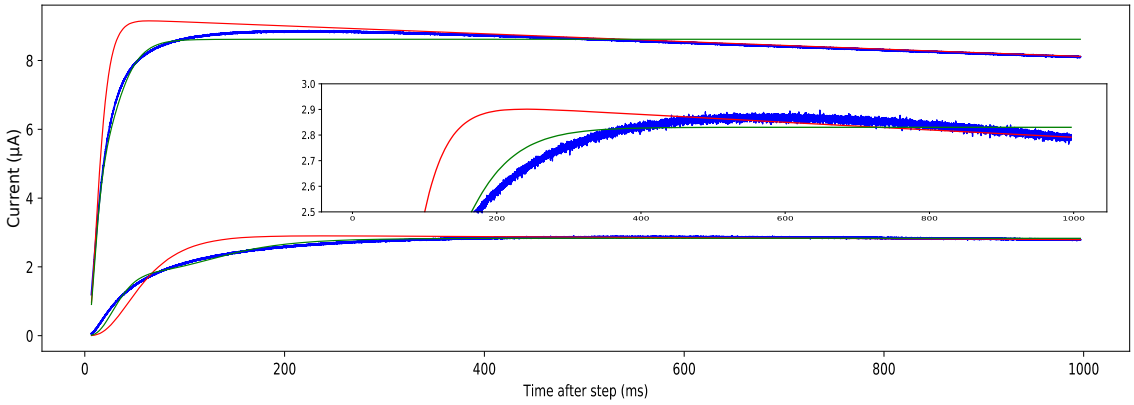


Figure 3.10: Single-component Kv2.1 kinetics fail to accurately fit activation and slow inactivation processes. Blue, data (median voltage 54.9 and 8.7 mV for the upper and lower trace, respectively). Green, least-squares fits that match the shape of activation, using fast inactivation kinetics. Red, fits that match observed slow inactivation, but not activation shape, hand-tweaked slightly to better match inactivation and thereby illustrate the problem. Inset, magnified view of the lower traces. The recording shown here is the same one used for illustration in section 3.4.1.

3.5.2 Two-component Kv2.1x

For Kv2.1, I added a non-inactivating component, since reasoned that the slow inactivation would not allow a distinction between two separate inactivating components. Thus, the extended model for Kv2.1, hereafter referred to as Kv2.1x, takes the form

$$I_K = (\bar{g}_f n^4 h + \bar{g}_s s^4)(V - E_K) \qquad \text{(Equation 3.16)}$$

or, in explicitly solvable and fittable terms,

$$I_K = a(1 - e^{-t/\tau_f})^4(h_\infty - (h_\infty - 1)e^{-t/\tau_h}) + b(1 - e^{-t/\tau_s})^4(V - E_K)$$

(Equation 3.17)

Again, I fit this equation, now with six parameters, to each trace separately, resulting in a much better qualitative fit as shown in Figure 3.11. With a little coaxing – i.e., guiding the fit by setting suggestive initial conditions – I achieved a very consistent separation into a fast and a slow component. By consistent, I mean two things: Firstly, that all recordings separated well, and that the fit improved

dramatically from one component to two components, both qualitatively and quantitatively, see Table 3.1. Secondly, separation was consistent in that within recordings, the relative size of the components was largely, though not entirely, stable between traces at different membrane potential levels, as shown in Figure 3.12.

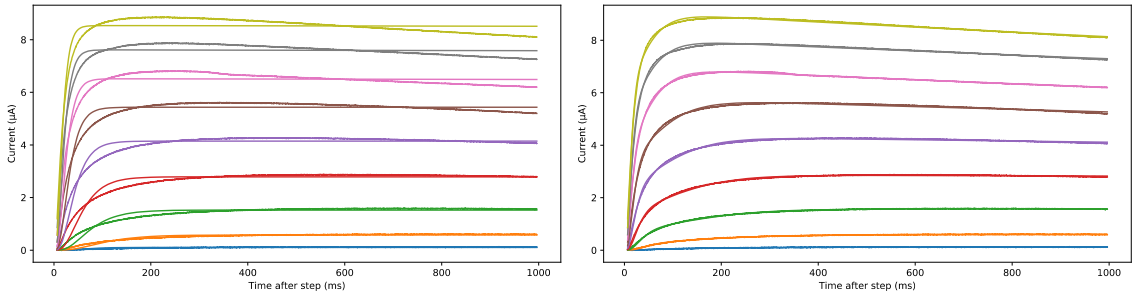


Figure 3.11: Kv2.1 least-squares fits using the single-component model (left) and the two-component model (Kv2.1x, right). Freed from trying to use inactivation to fit the current onset, the two-component model achieves a very tight fit throughout. Yes, there are two curves of each colour there!

	n	RMSE mean	RMSE std	t	p
Kv2.1	15	0.281	0.173	5.57	$6.96 * 10^{-5}$
Kv2.1x		0.090	0.050		
Kv1.4	6	0.131	0.046	5.62	$2.47 * 10^{-3}$
Kv1.4x		0.032	0.017		

Table 3.1: Fitting cost comparison between single-component and two-component models, significance-tested with a paired-samples t test. RMSE: Root mean squared current residual, in μA.

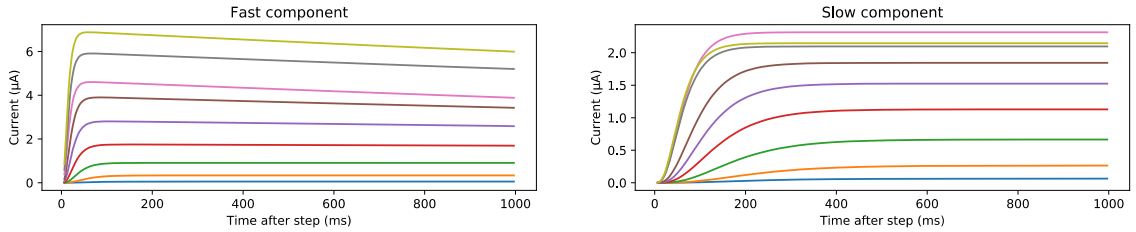


Figure 3.12: Kv2.1x fast and slow components that make up the two-component fit in Figure 3.11. Consistent separation was achieved by seeding the least squares fits with appropriate initial values for each trace. Though not quite perfect, both components look like they could be independent currents.

As it turns out, the failure of the single-component model to fit the data is by no means a coincidence. Upon expression in oocytes, Kv2.1 has indeed been shown to form channel complexes with endogenous MinK related proteins (MiRP), slowing the current kinetics of a subset of channels – presumably those interacting with MiRP – in an expression level dependent fashion (Gordon *et al.*, 2006). In other words, adding a second component, although initially motivated by a mismatch between data and model, is in fact based in biophysical reality.

Having fitted each recording to a two-component process, I could now ask what characteristics this channel (or rather, these channel complexes) exhibits. The steady-state variables behaved in accordance with expectations and are plotted in Figure 3.13 together with sigmoid curves of the form $x_\infty(V) = [1 + \exp((x_{mid} - V)/x_{slope})]^{-1}$ that best fit the data, the parameters for which are listed in Table 3.2.

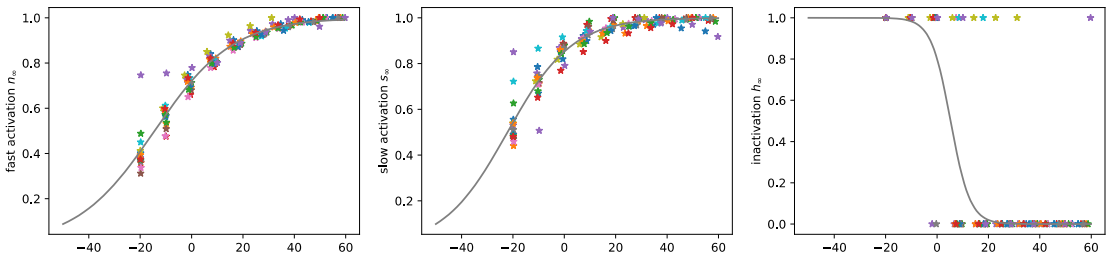


Figure 3.13: Kv2.1x steady-state activation and inactivation values plotted against the achieved voltage (asterisks, each colour represents a separate recording, n=15), and fitted sigmoids (grey lines). While the activation variables show a very clear sigmoidal relationship, the inactivation variable looks more like a binary switch. This may be due to a lack of good information, since the protocol used is not particularly suited to probing inactivation.

Next, I tackled the inactivation time constant τ_h, for which I had no useful data beyond the activation protocol, since h_∞ appeared to be close to 1 at the tail current voltages. Even in the activation protocol, a good part of the fitted curves fit better with no inactivation, as the high values for h_∞ in Figure 3.13 indicate. I therefore excluded all fits with $h_\infty > 0.5$, as well as those with $\tau_h > 10$ s, and instead included some of the literature values already used in section 3.3.1. The

resulting picture suggested an asymmetric relationship to voltage, so I fit a curve of the form

$$\begin{aligned} \tau_h(V) &= \tau_{h,min} + \frac{\tau_{h,lmax}}{L+1} + \frac{\tau_{h,rmax}}{L+R} \\ L &= \exp((\tau_{h,mid} - V)/\tau_{h,lslope}) \\ R &= \exp((\tau_{h,mid} - V)/\tau_{h,rslope}) \end{aligned} \qquad \text{(Equation 3.18)}$$

to the few data points available by hand, with the result shown in Figure 3.14 and Table 3.2.

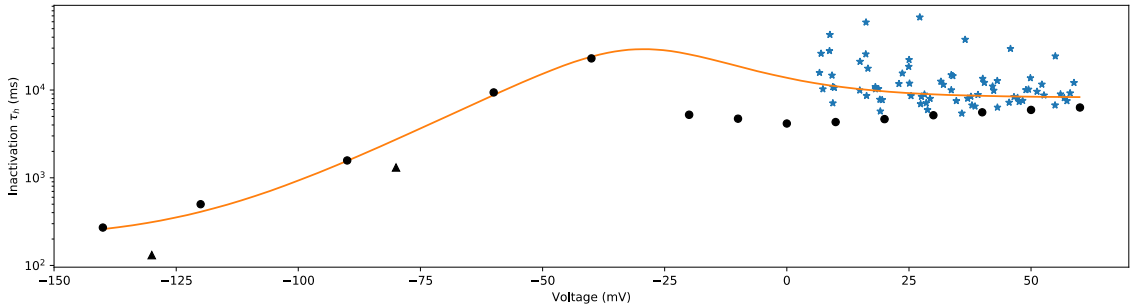


Figure 3.14: Kv2.1x inactivation time constants from activation protocol fitting (blue asterisks, n=15 recordings), Klemic et al., (1998, circles) and Kerschensteiner and Stocker (1999, triangles). The orange curve was fitted by hand.

Next, since the tail current protocol fitting method described in section 3.4.3 requires an estimate for high-voltage kinetics, I hand-fitted functions of the form

$$\tau_x(V) = \tau_{x,min} + \frac{\tau_{x,max}}{1 + \exp((\tau_{x,mid} - V)/\tau_{x,slope})} \qquad \text{(Equation 3.19)}$$

to the high-voltage τ_n and τ_s coming out of the activation protocol fits. I then fit the tail currents and adjusted the time constant functions to those fits, again by hand. Results are shown in Figure 3.15, with parameters for the approximations listed in Table 3.2. In retrospect, I am struck by the realisation that I could have done a much better job of fitting particularly the τ_n; at the time, I did not think to combine the data from both protocols, much less use a more principled fitting procedure than iterated hand-tuning. In my defence, I did try to least-squares fit the curves – but only to one side of the data at a time, which of course failed to produce results that fit the other side. I am, however, forced to present the fit in

its present, embarrassing state, because by the time I realised my error, I had already used this flawed model to generate MOSTIPS stimuli and collect data and did not have the time to return to this point.

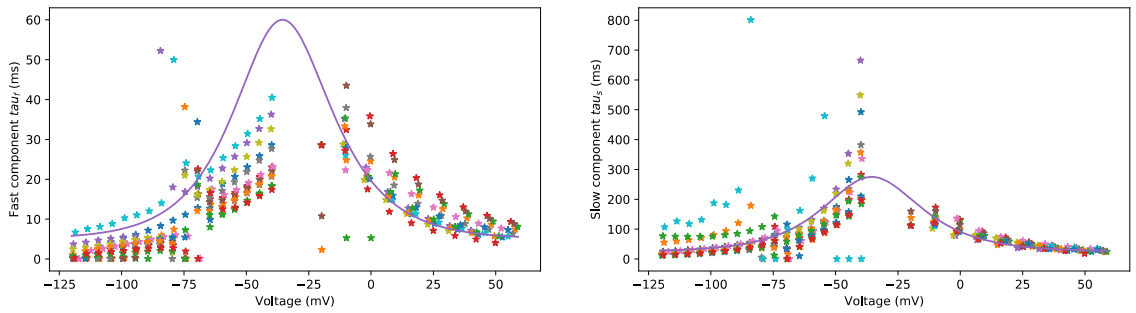


Figure 3.15: Fitted time constants for both Kv2.1x components. Each plot shows fitted values from both the activation protocol (>-25 mV) and the tail current protocol. Values belonging to the same cell are drawn in the same colour. The solid lines represent my approximations to the data. One recording was excluded due to low expression levels, which resulted in effectively undetectable tail currents, therefore here, n=14.

	unit	value	Range (sg)	Range (opt)	sigma	type
gK_fast	µS	60	[1, 200]	[1, 500]	0.05	*
gK_slow	µS	20	[1, 100]	[1, 500]	0.05	*
EK	mV	-70	[-90, -40]	[-130, -30]	0.1	+
gl	µS	2	[0.1, 30]	[0.01, 100]	0.01	*
El	mV	-10	[-30, 20]	[-60, 40]	0.1	+
C	nF	150	[50, 250]	[50, 500]	0.1	*
nK_mid	mV	-13.7	[-30, 0]		0.1	+
nK_slope	mV	14.8	[4, 40]		0.1	+
taunK_min	ms	5	[0, 20]		0.1	+
taunK_max	ms	110	[60, 160]		0.1	+
taunK_mid	mV	36	[20, 50]		0.1	+
taunK_slope1	mV	18	[5, 50]		0.1	+
taunK_slope2	mV	-17	[-50, -5]		0.1	+
hK_mid	mV	5.8	[-10, 20]		0.1	+
hK_slope	mV	-3.9	[-20, -1]		0.1	+
tauhK_lmin	ms	200	[0, 500]		1	+

	unit	value	Range (sg)	Range (opt)	sigma	type
tauhK_rmin	s	8	[4, 12]		0.1	+
tauhK_max	s	50	[10, 100]		0.1	+
tauhK_mid	mV	-30	[-50, -10]		0.1	+
tauhK_lslope	mV	-16	[-40, -4]		0.1	+
tauhK_rslope	mV	15	[4, 40]		0.1	+
sK_mid	mV	-19.7	[-40, 0]		0.1	+
sK_slope	mV	11.1	[4, 40]		0.1	+
tausK_min	ms	25	[0, 100]		0.1	+
tausK_max	ms	500	[250, 750]		1	+
tausK_mid	mV	36	[20, 50]		0.1	+
tausK_slope1	mV	18	[5, 50]		0.1	+
tausK_slope2	mV	-17	[-50, -5]		0.1	+

Table 3.2: Base value, range, and MOSTIPS metaparameters for the Kv2.1x model. Value ranges for the non-kinetic parameters are separated into those used during stimulus generation (sg) and the wider limits applied during optimisation (opt); for the kinetic parameters, the same limits were used for both tasks. The kinetic parameters (from nK_mid onwards) were only fitted in "Kv2.1k" fits, and were otherwise held fixed at the listed value.

3.5.3 Kv2.1x full-record fit

Using the full-record fitting method described in section 3.4.4, I then proceeded to estimate the full kinetic parameter set described above to each cell's activation and tail current recordings. The resulting kinetic curves are shown in Figure 3.16. Clearly, the steady-state curves were reasonably well chosen, as the agreement between the default model (strong blue lines) and the fits show. The wide agreement between records furthermore suggests that these parameters are also adequately constrained with the available data. This is less clear for the time constants, which are much more scattered, particularly for inactivation, which mirrors the difficulty I have had in fitting its kinetics from first principles. An additional protocol to better reveal inactivation would certainly have helped in generating a good model of the channel. The time course of fast activation, τ_n, strik-

ingly confirms the mismatch between the data and the hand-chosen parameters seen in Figure 3.15.

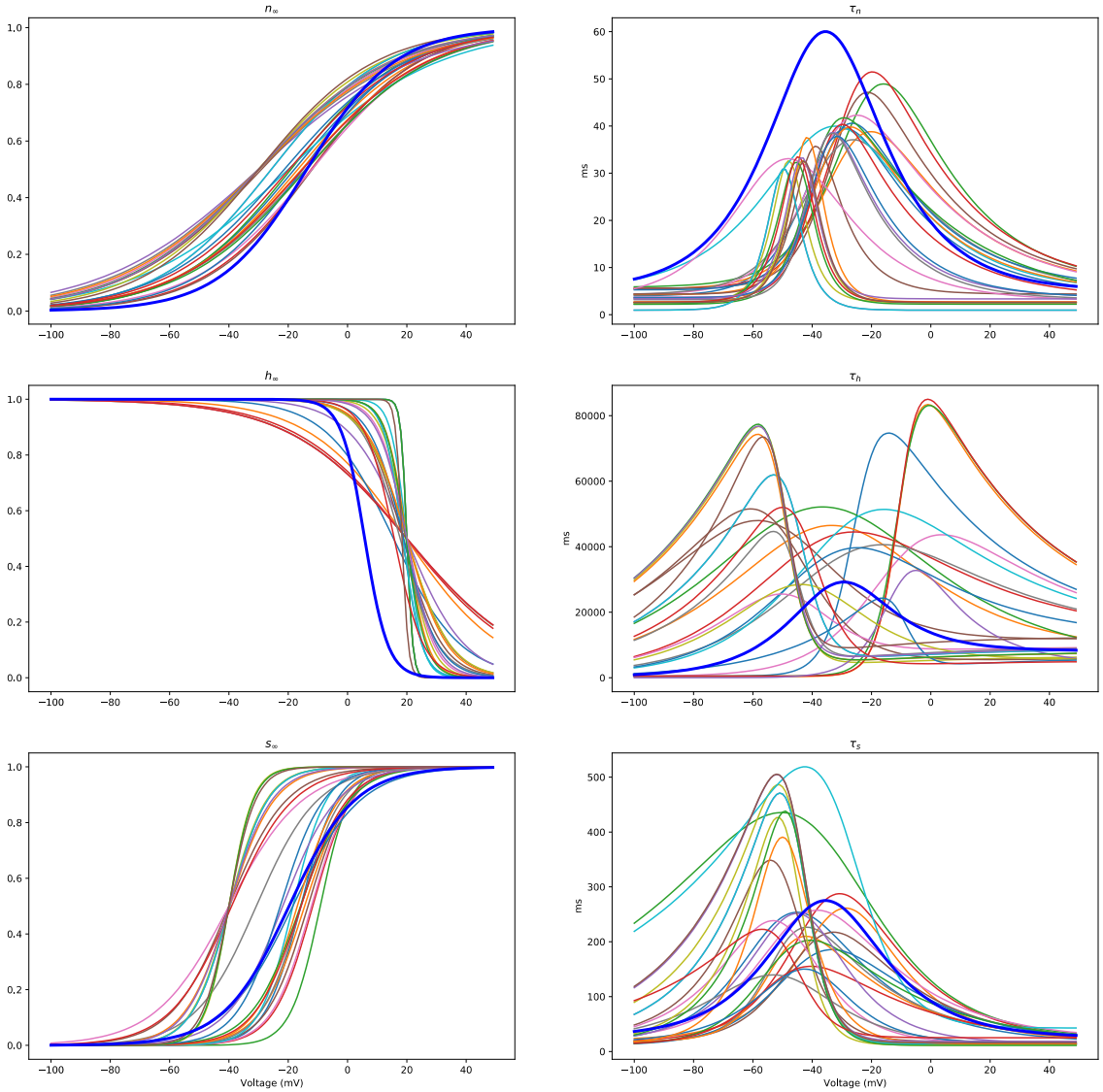


Figure 3.16: Kv2.1x full-record kinetic fits (n=27). The thick blue line represents the default kinetics (see Table 3.2), which were used as the initial guess for fitting.

3.5.4 Two-component Kv1.4x

Although the single-component Kv1.4 fits were less obviously inadequate than those to the Kv2.1 currents, I decided to try adding a second component to better fit the current peak. There is admittedly less biophysical justification to do so – I am not aware of any work similar to (Gordon *et al.*, 2006) looking specifically

at Kv1.4, though similar concentration-dependent effects have been seen in other potassium channels, including Kv1.2 (Guillemare *et al.*, 1992), Kv1.3 (Honore *et al.*, 1992), and others (Moran *et al.*, 1992).

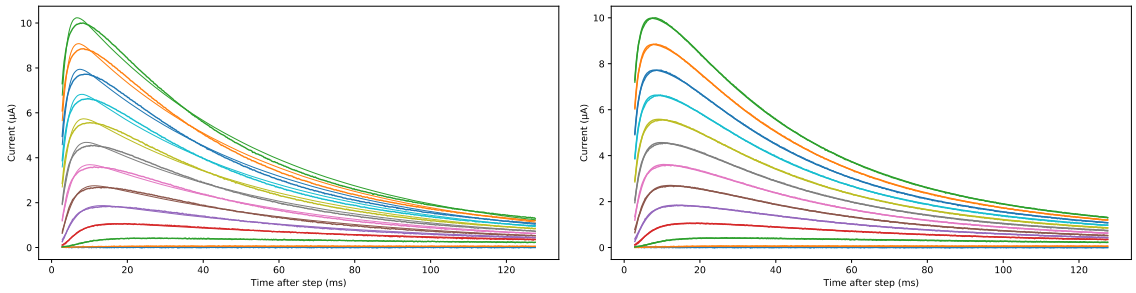


Figure 3.17: Kv1.4 single-component fit (left) and two-component fit (right). The single-component mismatch is less pronounced than with Kv2.1 currents, but the improvement in moving from one to two components is equally clear and remarkable.

Since inactivation is much more pronounced in Kv1.4, I posited the second component to also inactivate, such that both components take the form $g_c(V) = \bar{g}_c n_c^4 h$. Similar to Kv2.1x, I found that the two-component model fit much better (see Figure 3.17 and Table 3.1), with the components again consistently splitting up into a slower and a faster component (Figure 3.18) with only slight nudging via initial parameter guesses.

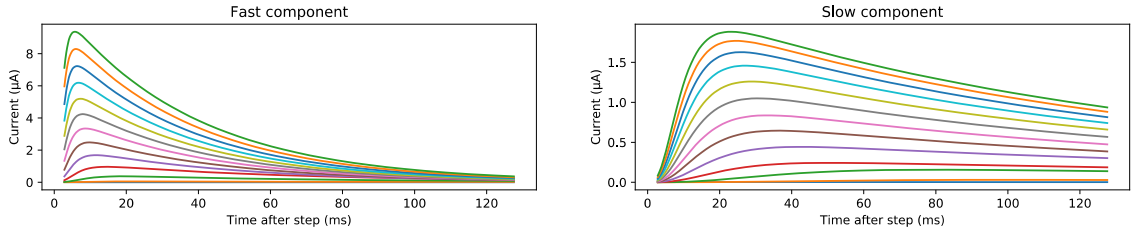


Figure 3.18: Kv1.4x fast and slow components that make up the two-component fit in Figure 3.17. The two components are remarkably well separated and both look like credible currents in their own right.

With the activation protocol fits in hand, I first formalised the voltage-dependent functions for inactivation. Following the literature-based model, both inactivation gates were modelled with an inactivating fraction h_a, such that

$$h_\infty(V) = h_a + \frac{1 - h_a}{1 + \exp((h_{mid} - V)/h_{slope})} \qquad \text{(Equation 3.20)}$$

Here, too, the data did not allow a distinction between the two components, so I hand-fit Equation 3.20 with $h_a = 0.02$, $h_{mid} = -60\ mV$, and $h_{slope} = -7\ mV$, see Figure 3.19.

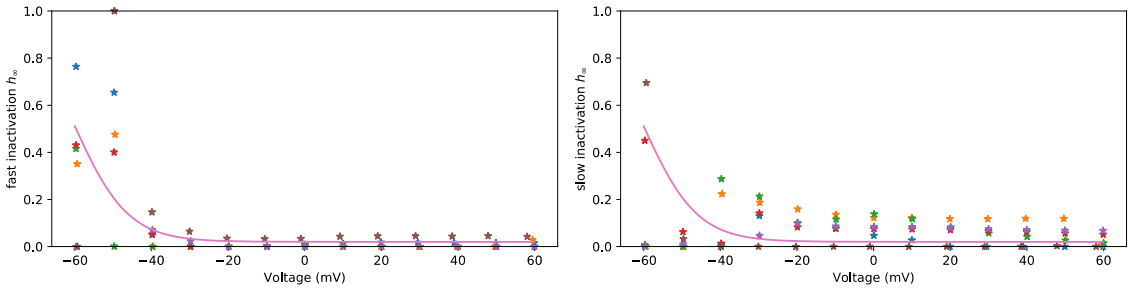


Figure 3.19: Kv1.4x fast and slow steady state inactivation. Each marker colour represents one of the n=6 recordings.

The time constants for inactivation I adjusted from the literature-based model (Equation 3.11) to the few available data points by hand. Unlike in Kv2.1, there was no need to exclude any of the data, as inactivation was clearly present throughout the probed voltages. The resulting equations are

$$\tau_{h,fast} = 35 + \frac{5 * 10^3}{1 + \exp((-72 - V)/-7)} \qquad \text{(Equation 3.21)}$$

and

$$\tau_{h,slow} = 120 + \frac{10^4}{1 + \exp((-72 - V)/-8)} \qquad \text{(Equation 3.22)}$$

with the data and fitted curve shown in Figure 3.20.

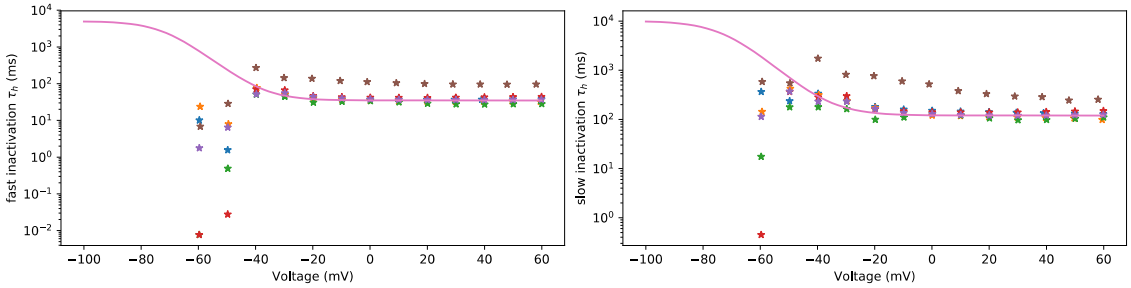


Figure 3.20: Kv1.4x fast and slow inactivation time constants.

Next, I turned to the activation variables. I modelled the fast and slow steady-state nf_∞ and ns_∞ with the usual sigmoid function,

$$n_\infty(V) = \frac{1}{1 + \exp((n_{mid} - V)/n_{slope})} \qquad \text{(Equation 3.23)}$$

Due to the small number of recordings I had (of my seven records, one was discarded due to low expression, leaving me with 6 viable fits), I could not reliably fit a steady-state sigmoid by least-squares, and so decided to hand-fit them instead. As with the inactivation steady-state, there appeared to be no clear difference between the components, so I decided to model both with the values $n_{mid} = -47\ mV$ and $n_{slope} = 15\ mV$. Results are shown in Figure 3.21.

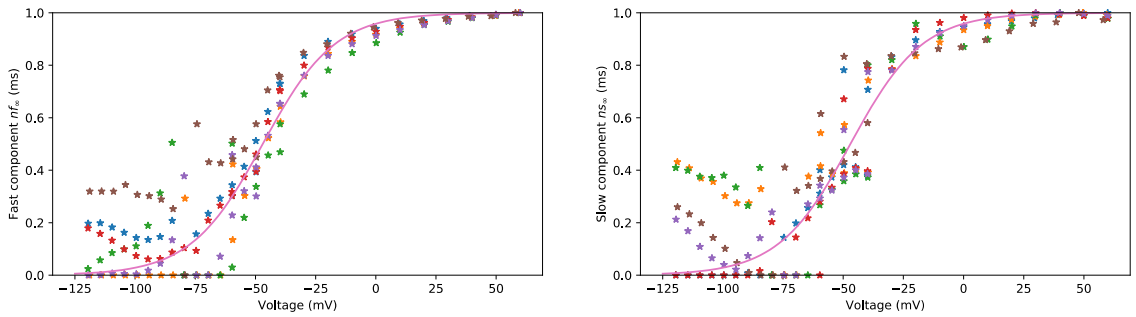


Figure 3.21: Kv1.4x fast and slow component activation steady-state fits, including data from both activation and tail current protocols. Note that there is some overlap in the probed voltages between -60 mV and -40 mV which is fit consistently. The sigmoid curve was not adjusted to the tail current data points (lower voltages), as I considered it unnecessary and potentially misleading.

For the activation time course, I hand-fit curves of the form used in the single-component model (Equation 3.10) to the data in order to run a tail current fit as described in section 3.4.3. Using data from both the activation protocol fits and the tail current fits, I then hand-fit the time constant equations as

$$\tau_{n,fast} = 1 + \frac{20}{\exp((-65 - V)/-20) + \exp((-65 - V)/32)}$$

(Equation 3.24)

and

$$\tau_{n,slow} = 4 + \frac{62}{\exp((-85 - V)/-28) + \exp((-85 - V)/20)}$$

(Equation 3.25)

The resulting curves, along with data from both protocols fits, are shown in Figure 3.22.

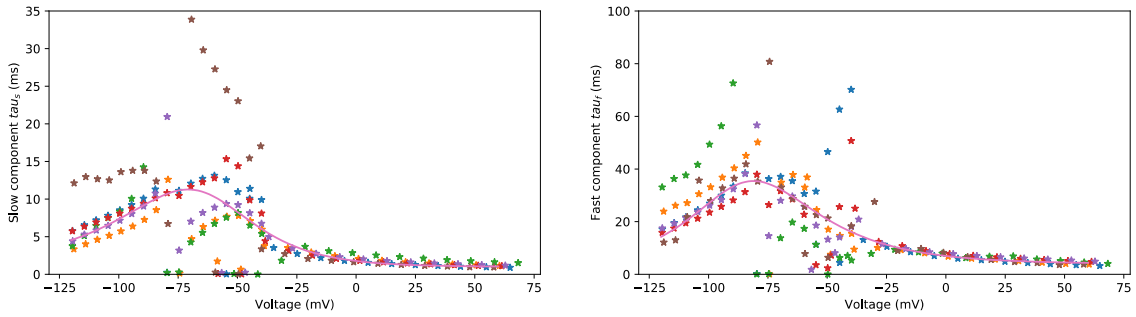


Figure 3.22: Kv1.4x fast and slow activation time constants. Two obvious outlier data points are not shown. The recording producing the brown markers does not distinguish itself in any obvious way other than the incongruous fitting results (note also its apparent outlier status in other figures in this section) and was therefore not removed, but simply ignored during hand-tuning.

Finally, because I did not quite trust my hand-tuning of the activation sigmoids described above, I fit the tail protocols in much the same way as described in section 3.4.3, but fitting nf_∞ and ns_∞ while drawing $\tau_{n,fast}$ and $\tau_{n,slow}$ from Equations 3.24 and 3.25. Somewhat surprisingly, although the tail current fits are of course noisier, the sigmoids matched the data reasonably well, as seen in Figure 3.21, and were not adjusted any further.

3.6 Conclusion

In this chapter, we have seen how, starting from presumably good characterisations in literature, I built models of two potassium channels. Upon expressing these channels in *Xenopus* oocytes – the exact environment the characterisations had emerged from – closer inspection of the expressed currents showed a clear mismatch to the model expectation, even when the kinetic aspects of the models were left entirely unconstrained. In the case of Kv2.1, this mismatch is entirely

explained by the interaction with endogenous proteins in oocytes; in Kv1.4, we can hypothesise a similar effect, though tangible evidence is lacking. Separating the channel models into two components, I have shown a greatly improved fit; thus, turning to the proposed MOSTIPS method in the next chapter, I have used these two-component models exclusively when working with real data.

I hasten to point out that, despite my relatively detailed analysis of the available data, neither of the extended models is quite beyond doubt. The stimulus protocols that were used are suited to illuminate only parts of the channel kinetics, but leave others – in particular, the time constants of inactivation and its recovery at lower voltages – ill constrained. A full characterisation would have to include inactivation probing, but was omitted due to time constraints: Given the 10-second holding period between each stimulus, the classical protocols shown in this chapter alone take approximately five minutes. Of course, in addition to those, I also had a number of MOSTIPS stimulus sets to record, raising the total recording time closer to 20 minutes per oocyte in some conditions. Although oocytes are remarkably robust under voltage clamp, prolonged experimentation is not conducive to the quality of the data received, such that I rejected further extending the set of recordings.

The difficulty of accurately modelling even a single, well known, well characterised channel in a highly controlled environment suggests, I would argue, that conductance-based neuron models are generally much more abstract than one might assume. We have seen here the result of incidental interactions between single channel proteins and non-neuronal factors; there can be no doubt that in neurons, these sorts of interactions are multiplied many times over, with heteromerisation between channel pore subunits, targeted interactions with auxiliary subunits, modulations mediated by other ligands such as calcium, as well as modulations driven by more complex processes as they are known e.g. in the regulation of synaptic channel proteins (Song & Huganir, 2002; Derkach *et al.*, 2007; Diering & Huganir, 2018). To accurately model all of these effects, along

with the effects of cell morphology and the distribution of channels among distinct membrane compartments, is impossible with the kinds of models, and the level of complexity, that are in use today.

While I do not believe that this invalidates the aim of this thesis, it does suggest a certain humility: If we look closely enough, any model, and any parametrisation thereof, must fall far short of the true complexity of a neuron. Thus, as we move into the next chapter, where I show how well the MOSTIPS method works for fitting these – ultimately rather crude – models to data, it is perhaps instructive to expect mismatch, rather than be surprised by it.

4 Empirical proof of concept

4.1 Introduction

Developing a method to measure previously unmeasurable quantities presents a bootstrapping problem: We want to make sure that the method works, but we cannot test it in its entirety, because no reference method exists against which we could calibrate our work. Therefore, it is of critical importance to tread carefully and verify the novel method's functionality at every step. In order to do this, I have tested the methods developed in chapter 2 against two types of model systems whose properties should be known well enough to serve as calibration.

The first of these systems is a simulated optimisation target. This offers the opportunity of perfect knowledge and, by using the simulated model as a starting point for the optimisation algorithm, of perfect correspondence between assumed and target model structure. Although the optimisation problem is by no means trivial even in this idealised scenario, it is a relatively low bar for the method to pass. In addition, fitting against a simulated target allows us to look at how well the method can cope with such added difficulties as instrument noise or target parameter values that are different from the original assumption. An exploration of these themes is presented in section 4.3.

The second model system that I have used for calibration is a more life-like test of optimisation against real, active membranes. Ideally, we would like to use a model system that is characterised well enough that we can obtain validation data that fully qualifies the success of the method. I know of no neurons that fit this description, since their morphology and ion channel composition is too complex to allow a full characterisation on a single cell. If, instead, we can turn a passive, geometrically simple cell into an active membrane compartment by introducing well-known ion channels into it, we should have an easily characterised biological system. This is made possible by using *Xenopus* oocytes, a commonly

used expression system that will readily translate any cRNA injected into its cytoplasm. Using this feature, I have gathered electrophysiological data from real, active membranes, without having to worry about geometry, in an effort to determine how well the method works with such data. We have already seen the difficulty of accurate modelling and model fitting by classical methods in the previous chapter; in section 4.4, we will look at how well my algorithmic solution fares.

4.2 Methods

First, however, I need to elucidate details of the experiments, simulations and algorithms I've used to produce my results, beyond the wet-lab experimental methods described in chapter 3. Although the working principle of the algorithms that comprise the MOSTIPS method is described in chapter 2, some details, as well as many metaparameters (i.e., parameters governing the algorithms themselves), require additional explanation. Many of these metaparameters were chosen judiciously, though without any detailed exploration of the methods' sensitivity to their choice, not least because there are too many metaparameters to investigate their effects in a principled manner.

4.2.1 Models

The models I used to generate the results in this chapter are, in an approximate order of increasing complexity, a single-component delayed rectifier potassium current (Kv2.1), its two-component analogue (Kv2.1x), a two-component A-type potassium current (Kv1.4x), a combination of the latter two (Kboth), the original Hodgkin-Huxley squid axon model (HH), a model of the B1 motor neuron in the *Lymnaea stagnalis* buccal ganglia (B1), and finally a version of the two-component delayed rectifier current where I fitted the kinetic parameters (Kv2.1k).

All models take the generic form shown in Equation 4.1, where C is the membrane capacitance, I_{inj} is injected (clamp) current, g_l and E_l define a passive leak

current, and the active conductances I_{ion} are, of course, specific to the model and described below.

$$C\frac{dV}{dt} = I_{inj} - g_l(V - E_l) - \sum I_{ion}(t, V) \qquad \text{(Equation 4.1)}$$

A small number of hyperparameters for the MOSTIPS algorithm were chosen differently between models, and are listed in Table 4.1.

	unit	All Kv	HH	B1
Sampling interval	ms	0.1	0.25	0.25
Settling duration	s	10	1	1
Clamp gain		2000	1000	1000
Access resistance	MΩ	0.6	5	15
Current limit for stimulus generation	µA	10-20	10	1
Holding potential	mV	-80	-70	-60

Table 4.1: Model-specific hyperparameters for stimulus generation, robustness screening, and model optimisation. Current limits for the potassium channel models were, for Kv2.1 and Kv2.1k, 10 µA; for Kv2.1x, 20 µA; and for Kv1.4x, 15µA.

4.2.1.1 Kv2.1

The details of the single-component Kv2.1 model are shown in section 3.3.1, and parameter details specific to MOSTIPS are listed in Table 4.2.

	unit	value	range	sigma	type
gK	µS	60	[10, 200]	0.05	*
EK	mV	-80	[-100, -50]	1	+
gl	µS	0.1	[0.0001, 10]	0.01	*
El	mV	-10	[-30, 10]	1	+
C	nF	150	[50, 200]	0.01	*

Table 4.2: Base value, range, and MOSTIPS metaparameters for the Kv2.1 model.

4.2.1.2 Kv2.1x and Kv2.1k

The two-component model of Kv2.1 was used in two incarnations: Firstly, with fixed kinetics (as Kv2.1x), where only passive parameters (C, gl, El, EK) and maximum conductances (gK_fast, gK_slow) are fitted; and secondly, where in addition, all kinetic parameters were adjustable with MOSTIPS (as Kv2.1k). The model structure, i.e. the form of the equations, was identical between these two, and is detailed in section 3.5.2. The kinetic parameters for Kv2.1x were the default values listed in Table 3.2, which also contains the MOSTIPS metaparameters for both models. As mentioned in the legend to that table, the parameter range for the non-kinetic parameters differed between stimulus generation and model optimisation, as the data included oocytes that lay beyond what I had considered the likely parameter range.

4.2.1.3 Kv1.4x

The two-component model of Kv1.4 is described in detail in section 3.5.4, with the MOSTIPS metaparameter listed in Table 4.3.

	unit	value	range (sg)	range (opt)	sigma	type
gA_slow	µS	20	[5, 50]	[1, 500]	0.05	*
gA_fast	µS	80	[10, 200]	[1, 500]	0.05	*
EK	mV	-80	[-100, -50]	[-130, -30]	0.1	+
gl	µS	0.1	[0.0001, 10]	[0.01, 100]	0.01	*
El	mV	-10	[-30, 10]	[-60, 40]	0.1	+
C	nF	150	[50, 200]	[50, 500]	0.1	*

Table 4.3: Base value, ranges, and MOSTIPS metaparameters for the Kv1.4x model.

4.2.1.4 Kboth

Oocytes expressing both Kv1.4 and Kv2.1 were modelled with each of the components of Kv1.4x and Kv2.1x and a common potassium equilibrium potential EK for both channels. For details of the channel models, see sections 3.5.2 and 3.5.4. The details of the parameters under optimisation are shown in Table 4.4.

	unit	value	range (sg)	range (opt)	sigma	type
gK_slow	µS	60	[1, 200]	[1, 500]	0.05	*
gK_fast	µS	20	[1, 100]	[1, 500]	0.05	*
gA_slow	µS	20	[5, 50]	[1, 500]	0.05	*
gA_fast	µS	80	[10, 200]	[1, 500]	0.05	*
EK	mV	-80	[-100, -50]	[-130, -30]	0.1	+
gl	µS	0.1	[0.0001, 10]	[0.01, 100]	0.01	*
El	mV	-10	[-30, 10]	[-60, 40]	0.1	+
C	nF	150	[50, 200]	[50, 500]	0.1	*

Table 4.4: Base value, ranges, and MOSTIPS metaparameters for the Kboth model combining Kv1.4x and Kv2.1x.

4.2.1.5 HH (Squid axon)

The squid axon model, besides being a simple, experimentally validated model of a neuronal membrane compartment, was chosen for its place in the history of conductance-based neuron modelling. It is an implementation of the squid giant axon model described in (Hodgkin & Huxley, 1952d). The model contains two active conductances, representing sodium and potassium current.

The sodium current is modelled as $I_{Na} = \bar{g}_{Na} m^3 h (V - E_{Na})$, with the activation variable m following

$$\begin{aligned} \frac{dm}{dt} &= \alpha(1-m) - \beta m \\ \alpha &= \frac{-0.1(V+45)}{1-\exp((V+45)/-10)} \\ \beta &= 4\exp((V+70)/-18) \end{aligned} \qquad \text{(Equation 4.2[4])}$$

and the inactivation variable h following Equation 4.3:

4 Note that here and in subsequent equations, α and β are dimensionless values implicitly expressed in ms^{-1}, and V in these equations is the dimensionless value of the membrane potential expressed in mV.

$$\begin{aligned} \frac{dh}{dt} &= \alpha(1-h) - \beta h \\ \alpha &= 0.07\exp((V+70)/-20) \\ \beta &= \frac{1}{1+\exp((V+40)/-10)} \end{aligned} \qquad \text{(Equation 4.3)}$$

The potassium current is modelled as $I_K = \bar{g}_K n^4 (V - E_K)$, with the gating variable n following

$$\begin{aligned} \frac{dn}{dt} &= \alpha(1-n) - \beta n \\ \alpha &= \frac{-0.01(V+60)}{\exp((V+60)/-10)-1} \\ \beta &= 0.125\exp((V+70)/-80) \end{aligned} \qquad \text{(Equation 4.4)}$$

In this model, the kinetic parameters were held constant, leaving only the parameters in Table 4.5 to be fitted by MOSTIPS.

	unit	value	range	sigma	type
gNa	µS	120	[60, 240]	0.05	*
ENa	mV	45	[25, 65]	0.1	+
gK	µS	36	[18, 72]	0.05	*
EK	mV	-82	[-102, -62]	0.1	+
gl	µS	0.3	[0.15, 0.6]	0.01	*
El	mV	-60	[-80, -40]	0.1	+
C	nF	1	[0.5, 2]	0.1	*

Table 4.5: Base value, range, and MOSTIPS metaparameters for the HH model.

*4.2.1.6 B1 (*Lymnaea stagnalis *buccal motor neuron)*

The B1 model was intended to be a live experimental target within this project, had I progressed beyond oocytes. Although I did not quite get there, I did use a version of this model with slightly adjusted kinetics for the preliminary closed-loop work documented in section 5.2. The model described here mirrors that proposed by (Vehovszky *et al.*, 2005). Besides a leak conductance, it contains a

sodium current I_{Na}, a delayed rectifier potassium current I_K with two components, and an A-type potassium current I_A, defined as shown in Equation 4.5:

$$\begin{aligned} I_{Na} &= \bar{g}_{Na} m^3 h (V - E_{Na}) \\ I_K &= (\bar{g}_{KA} n_a^2 + \bar{g}_{KB} n_b)(V - E_K) \\ I_A &= \bar{g}_A a^4 b (V - E_K) \end{aligned} \qquad \text{(Equation 4.5)}$$

All currents' gating variables are formulated by their steady-state and time constants, i.e., they follow the formulation

$$\frac{dx}{dt} = \frac{x_\infty - x}{\tau_x} \qquad \text{(Equation 4.6)}$$

The sodium current's activation kinetics are defined as

$$\begin{aligned} m_\infty &= \frac{1}{1 + \exp((V + 24)/-8)} \\ \tau_m &= 0.3 + \frac{8}{1 + \exp((V + 40)/2)} \end{aligned} \qquad \text{(Equation 4.7)}$$

and its inactivation kinetics as

$$\begin{aligned} h_\infty &= \frac{1}{1 + \exp((V + 29)/3.8)} \\ \tau_h &= 2.4 + \frac{15}{1 + \exp((V + 24.3)/3.8)} \end{aligned} \qquad \text{(Equation 4.8)}$$

The two components of the delayed rectifier current activate with

$$\begin{aligned} n_{a\infty} &= \frac{1}{1 + \exp((14.9 - V)/16.6)} \\ \tau_{na} &= 38.3 - 0.41V \end{aligned} \qquad \text{(Equation 4.9)}$$

and

$$\begin{aligned} n_{b\infty} &= \frac{1}{1 + \exp((8.6 - V)/14.6)} \\ \tau_{nb} &= 5.7 - 0.07V \end{aligned} \qquad \text{(Equation 4.10)}$$

Finally, the A-type current's activation kinetics are

$$\begin{aligned} a_\infty &= \frac{1}{1 + \exp((12.4 + V)/-14.1)} \\ \tau_a &= 1.8 - 0.03V \end{aligned} \qquad \text{(Equation 4.11)}$$

and its inactivation kinetics,

$$b_\infty = \frac{1}{1 + \exp((71 + V)/6.6)}$$
$$\tau_b = 26.0 - 0.22V \qquad \text{(Equation 4.12)}$$

As with the squid axon model, I have kept the kinetic parameters constant, fitting only the passive parameters and maximum conductances as shown in Table 4.6.

	unit	value	range	sigma	type
gNa	µS	7	[0.1, 25]	0.03	*
ENa	mV	35	[20, 50]	1.14	+
gKA	µS	1.44	[0.01, 10]	0.008	*
gKB	µS	2.88	[0.01, 10]	0.002	*
gA	µS	12	[0.1, 40]	0.016	*
EK	mV	-67	[-82, -52]	0.1	+
gl	µS	0.02	[0.0001, 1]	0.1	*
El	mV	-20	[-50, 10]	3.4	+
C	nF	3.5	[1, 6]	0.029	*

Table 4.6: Base value, range, and MOSTIPS metaparameters for the B1 model.

4.2.2 Simulations

All simulations were performed with Runge-Kutta-Fehlberg 4/5 integration with a minimum step size equal to one twentieth of the observation sampling interval noted in the respective model description, and a maximum step size extending to the next time point at which a value needed to be reported. In order to arrive at a steady state, models were settled by imposing the holding potential of the next stimulus for 10 s (potassium channel models) or 1 s (HH and B1 models). During observed periods, clamp current and any values derived from it (e.g. elementary effects) were recorded at the temporal resolution noted in the model descriptions. Simulations generating synthetic target traces (section 4.3) were executed on the CPU, using functionally identical code to that produced for GPU simulations. Target traces were produced only once and then cached for repeated use

during fitting. All other simulations as required throughout the MOSTIPS stimulus generation and model fitting process were executed on the GPU, parallelised across 8192 model instances.

For stimulus generation and robustness testing, the simulated voltage clamp was parametrised in line with expected experimental circumstances. For the invertebrate neuron models, the gain and access resistance were set to 1000-fold and between 5 and 15 MΩ, respectively, in anticipation of sharp-electrode voltage clamp. For the potassium channel models, which were constructed for oocyte experiments, clamp and access resistance were 2000-fold and 0.6 MΩ, respectively. When fitting against synthetic data, these parameters were left unchanged; when fitting against real data, they were adjusted to the true experimental values that had been used during data collection.

Since voltage clamp is experimentally a little trickier than mere current clamp recordings, it is easily polluted by artefacts specific to the equipment used, to saline and noise levels, and many other factors that are difficult to control. The largest impact of these artefacts appears immediately after a command voltage step, as the equipment is pushed to its limits by the capacitive current. To avoid fitting against such artefactual data, I categorically excluded all data within 5 ms after a command voltage step at all points in the process.

Lastly, all simulations were run using single-precision (32-bit) floating point numbers for parameters, state variables and other internals of the simulation and related code, with the notable exception of variables that accumulate values (e.g. current residuals) over an entire stimulus, which were implemented as double precision (64-bit) floating point numbers to reduce the cumulative effect of numeric imprecisions and rounding.

4.2.3 Stimulus generation

Voltage clamp stimuli were constrained to last 1 second, to start at a holding potential of -80 mV (potassium channel models), -70 mV (HH) or -60 mV (B1), and

to have between 1 and 10 steps or ramps with command potentials reaching between -120 mV and +60 mV. At the outset of the stimulus generation algorithms, 3200 stimuli were randomly generated within these constraints. Once this initial set was exhausted, existing stimuli were chosen uniformly from within the archive and mutated as described in Table 2.1.

The parameter sets used during stimulus generation were initialised once, at the start of the search, such that all stimuli were scored against the same parameter sets. In most cases, I used 32 starting points in parameter space, including the model's initial parameter set, and other starting points drawn uniformly within the ranges shown in each model's parameter table. Each starting point was detuned by adding the appropriate σ to each successive parameter to yield a spiral trajectory of length 8. Since a single epoch contained 8192 parallel model simulations, this allowed for 32 stimuli to be evaluated per epoch.

An exception to this was the 28-parameter Kv2.1 model with adjustable kinetics (Kv2.1k), for which only the initial parameter set was used as a starting point for a single spiral detuning trajectory of length 32. This was done to avoid over-reliance on the particular randomly chosen starting points, while keeping the computational demands relatively low by dodging the issue of combinatorial explosion. Thus, with only a single trajectory, the Kv2.1k model allowed for 256 stimuli to be evaluated per epoch.

The MAP-Elites archive used the duration and mean current of the scored observation as outcome dimensions (see Table 2.2) with 32 bins each, increasing to 64 after 1000 epochs. Following the observation that the observed current was generally concentrated near the lower end of the total possible current in each model, I applied a current limit, with observations yielding a higher mean current binned to the highest available bin. This limit was model-specific and is listed in Table 4.1. In addition to duration and mean current, I used the index of the parameter for which fitness is evaluated and, for cluster-based MAP-Elites searches, the number of valid clusters found. These latter two dimensions are limited by

the model and implementation details, respectively, and were therefore unaffected by the resolution increase in epoch 1000.

To estimate the average elementary effect size $\bar{\Delta}$, which is used to normalise the effect of detuning (see section 2.2.2.1), 100 epochs of purely randomly generated stimuli were simulated and the elementary effects across all stimuli and all time points averaged for each parameter.

In order to keep processing time within manageable limits, the cluster and bubble algorithms did not operate on individual current samples, but instead on the sample average in 500 µs windows. In addition to a performance boost, this may also have had the effect of reducing the influence of floating point inaccuracies by averaging them out.

While the bubble algorithm was left free to explore even the shortest possible "bubbles" of 500 µs, the cluster algorithm was restricted to disregard observations of less than 5 ms duration for greater robustness. In addition, implementation details limited the number of bubbles per evaluated stimulus to a single highest-fitness bubble for each parameter, and the number of clusters to the longest 32 (total, regardless of model size). The cluster similarity threshold was set to 0.98.

The MAP-Elites algorithm was run for 10,000 epochs in search of bubbles, and for 100,000 epochs when searching for clusters. The number of epochs was chosen somewhat arbitrarily, based on anecdotal observations of the search process stalling with a largely saturated archive. Several factors play into saturation, the most obvious being the archive size, which was 32-times larger for clusters due to the additional outcome dimension (number of clusters, see above). A second factor is that a given stimulus, returning up to 32 clusters, can potentially occupy many more niches in a cluster archive than a bubble archive, which translates into a smaller number of unique stimuli relative to the size of the archive. Finally, because they are restricted to contiguous intervals, bubbles for many

parameters are less likely to extend over large durations, leaving parts of the archive effectively unreachable. Although I have not investigated this in detail (fitness considerations suggest that a superficial investigation via the number of unique stimuli would be insufficient), I suspect that, given the epoch numbers above, the cluster archives were left comparatively underexplored, yet may have yielded more valuable stimuli due to the greater freedom that the search was given.

4.2.4 Stimulus screening and selection

The theoretical maximum number of stimuli per parameter under the conditions outlined above is 4096 (64^2) for bubble searches, and 131072 (64^2*32) for cluster searches. With some areas of stimulus space effectively unreachable, the actual number of candidates was somewhat smaller, on the order of 2,500 and 90,000 total stimulus/observation pairs per parameter for bubble and cluster searches, respectively. A notable exception to this was the Kv1.4x model, which, due to its swiftly inactivating current, was unable to produce long, high-current stimuli, which limited the archive to around 1,000 and 20,000 stimuli per parameter for bubble and cluster searches, respectively.

The general process of selecting a promising subset of stimuli from an archive is described in section 2.2.5, and illustrated with two examples in Figure 4.1. Briefly, the archive was split into parameter-specific slices, and for cluster searches, the third outcome dimension (number of clusters) was squeezed out, leaving duration and mean current of the observation as the two outcome axes. To achieve a reasonable signal-to-noise ratio, observed current was limited to >1 µA (>100 nA for B1 models) and observed duration to >30 ms. If the remaining solutions numbered less than 1000, they were all selected for robustness screening; otherwise, a tolerant Pareto optimality criterion was applied, with the tolerance factor ϑ at 5% of maximum observed fitness. For the Kv2.1 model, no further selection was applied, while for the HH and B1 models, I selected for

high duration and high current with ϑ along both axes at 8 bins (i.e., 125 ms and 12.5% of the current range). The selection strategies for the oocyte models were slightly different, see section 4.2.5 below.

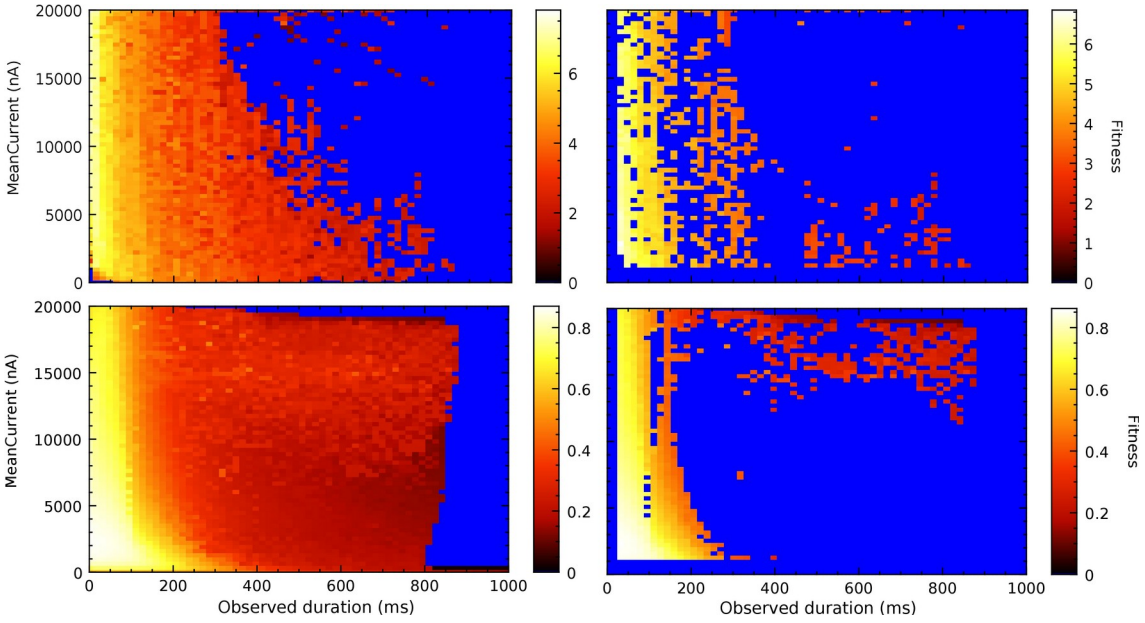


Figure 4.1: Example MAP-Elites archive selection, from archives generated for the Kboth model. Colour indicates candidate fitness, with blue for no candidate. Top: Bubble search, candidates for capacitance (C). Bottom: Cluster search, candidates for the Kv1.4x slow component conductance (gA_slow). Left, full archive for the given parameter; right: candidates selected by lower boundaries and a tolerant Pareto front as described in the text.

The selected stimulus/observation pairs were then run through the robustness screening algorithm using unpaired parameter space sampling (see section 2.2.4.3), yielding distance-to-error correlations $\rho_{weighted}$, $\rho_{target-only}$ and $\rho_{unweighted}$ and corresponding scores ζ for each candidate. Finally, picking the stimulus/observation pair with the highest ζ in each parameter, I arrived at one stimulus set for each search modality (cluster and bubble) and weighting scheme (weighted, target-only, and unweighted). All stimulus sets used in this chapter are presented without further comment in Appendix 1.

4.2.5 Oocyte stimuli and recordings

The experimental protocols for oocyte preparation and recording are detailed in chapter 3. The data presented in this chapter are derived from a total of 23 cells with joint expression of Kv2.1 and Kv1.4, 8 cells expressing Kv1.4, and 12 cells expressing Kv2.1. As noted in chapter 3, each cell was first probed with the classical stimulus protocols for validation purposes. Then, MOSTIPS stimuli were used according to the expression profile of the oocyte (i.e., according to what cRNA was injected, as long as any potassium current was seen at all; no attempt was made to check for successful joint expression). Each set of stimuli generated a single data file that was either saved for fitting or, in case the recording deteriorated beyond the point of usefulness during stimulation, discarded along with the affected oocyte, such that no incomplete stimulus sets were used for fitting.

Due to time constraints, I had not quite finalised the stimulus selection process yet when I started recording the data presented in section 4.4. Therefore, the stimuli used for oocyte recordings differ in two minor details: Firstly, stimuli were selected for robustness screening following the principles outlined above, but with some discretion applied with regards to the Pareto front and its tolerance limits. No Pareto front was used where few stimuli were found; where Pareto front selection was used, the tolerance values lay between 5-10% fitness, with further selection for high current and high observation duration where many (>1000) candidates remained. Secondly, after robustness screening, the values of ρ were not consistently normalised, with ρ_{min} fixed at 0 or -1 manually, and in a few cases mistakenly in contravention to the principle set out section 2.2.5. However, these discrepancies should not have a significant impact on the outcomes, and are indeed only partly reflected in the selected stimuli. For the Kv2.1x and Kboth models, several sets of stimuli were selected based on shifting criteria; for analysis, these details are ignored.

Finally, since the oocytes expressing Kv2.1 were probed for two models, Kv2.1x and Kv2.1k, and since a single stimulus set for the latter includes 28 stimuli (one

per parameter) with the membrane potential held at -80 mV for 10 s between each, only two stimulus sets were used for each of these models, selected from cluster and bubble archives using a $\rho_{weighted}$ criterion. Unweighted and target-only fits to Kv2.1-expressing oocytes in section 4.4 were completed using recordings from these weighted stimulus sets.

4.2.6 Model optimisation

Model optimisation attempts using both a genetic algorithm (GA, section 2.3.2.1) and differential evolution (DE, section 2.3.2.2) were performed over 500 epochs in bundles of 8 separate, parallel populations of 1024 total candidate parameter sets each. Given a settling period of 10 s before each stimulus and a stimulus duration of 1 s, this would take some 90 minutes if simulated at real time; in reality, since data collection and model optimisation were decoupled, simulations generally completed much faster, taking only 10-15 minutes in most instances. In GA fits, 32 candidates were retained across epochs as elites, and a further 32 were reinitialised from scratch, while the remaining 960 candidates were generated by mutation with a 30% crossover probability (xGA) or without crossover (mGA). The parameter-specific mutation step sizes are listed as “sigma” in the model description tables; these were multiplied by 5 and decayed with a half-life of 100 epochs. In DE fits, the decay period λ used for self-adapting method and crossover probabilities was set to 10 epochs, allowing the algorithm to quickly respond to changing circumstances as the population converges on a solution.

In general, the mutation scheme (unweighted, weighted, target-only) was chosen to reflect the stimulus selection method, except for the Kv2.1x and Kv2.1k models where, as noted above, only data from weighted stimulus sets were available, which were used with all three mutation schemes to complete the picture. While I appreciate that lumping an aspect of stimulus selection and the mutation scheme together in this manner may make it slightly more difficult to interpret the results, I would argue that the two concepts are linked closely

enough to justify this. Moreover, since the different weighting schemes are closely related, there was considerable overlap between stimulus sets, such that any differences in performance are more likely to arise from the mutation scheme than from the stimulus selection criteria.

4.3 Fitting against synthetic data

Before taking the proposed method into a lab setting, it is sensible to run a proof of concept under controlled conditions. To do this, I have used synthetic data – i.e., data generated by model simulations – as easily collected and manipulated target data for an early proof of concept. Since almost all parts of the MOSTIPS method have changed since that initial groundwork, however, I present here data gathered at a later stage of my project, using the algorithms in the form described in this thesis. For this demonstration, I have chosen two neuron models (HH and B1) as well as the single-component Kv2.1 model.

In this section, we will look at the method's performance in fitting against these three models in an environment of increasing difficulty, arising from the manner of generating the reference data. The easiest target would seem to be one that can be matched exactly. I present results from fitting against simulations of the original models, both with the original parameter values that were used during stimulus generation, and with randomised parameter sets. Since the original parameter set is always considered during stimulus generation, while randomly parametrised models may lie in areas of parameter space that were not explored, I would expect the selected stimuli to be less well adapted to fitting against randomised models, making those more difficult to match precisely. Then, to make the fitting both more difficult and more realistic, we will look at how adding white noise to the synthetic data changes the fitting performance.

The MOSTIPS algorithm as described in chapter 2 and sections 4.2.3 to 4.2.6 leaves a number of choices open. There are two stimulus generation algorithms (cluster and bubble), three robustness correlation and mutation weighting

schemes (unweighted, weighted, target-only) and three optimisation algorithms (DE, xGA and mGA). In addition, each model has different parameters with distinct scales, and the models themselves have characteristic scales e.g. of voltage clamp current. In the following I will first give examples of how fitting unfolds over time in specific examples, and then summarize performance with respect to the different algorithm choices.

4.3.1 Convergence within populations

4.3.1.1 Introduction

First, we will look at how well the fitting algorithms converge on a particular solution. To make this a little more tangible, I present Figure 4.2 as an example. It shows the median and interquartile range of the parameter standard deviation in each individual model population of fits against a noise-free, originally parametrised B1 model. Each colour represents one fitting method and summarises fits across all weighting schemes and stimulus generation methods, each producing eight independent model populations. We can see that the general trend is a roughly exponential decrease in the within-population variance. The time scales differ somewhat between the fitting methods, with DE populations clearly contracting more slowly, but reaching a more tightly constrained region of parameter space than the GA variants in most cases. There are also clear differences between parameters, with some proving apparently more difficult to constrain than others.

Before we take a closer look at more data, there are a few things worth noting here. Firstly, what we are looking at in this section – convergence within populations – means neither convergence across populations, nor convergence with the reference parameter set, which are tackled in the next section. It also does not necessarily imply that the populations are stationary; it is possible in principle for a population to move around parameter space while remaining internally convergent. This possibility will not be further explored, but is worth keeping in mind. I

also do not discuss the question of the time to converge, partly because the fitting algorithms are not my central concern, and partly because there is no obvious and unbiased way to control for the initial range of each parameter.

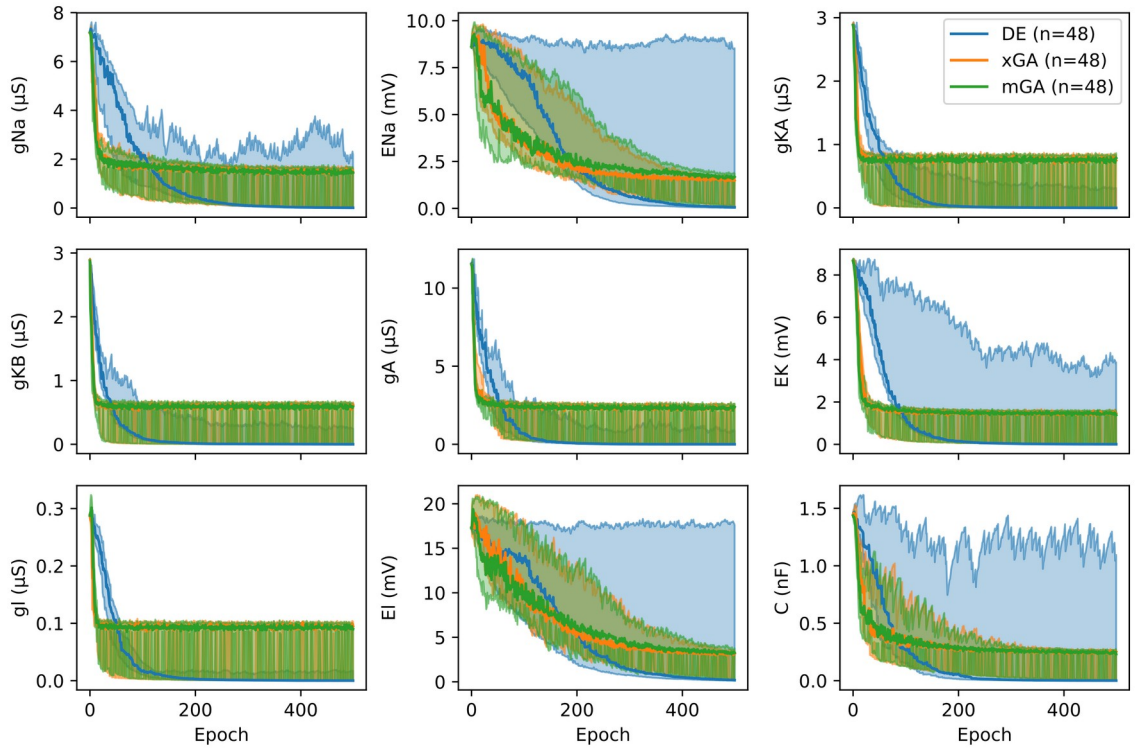


Figure 4.2: Example convergence for the B1 model, split by parameter (subplots) and fitting algorithm (colours). The reference model was parametrised with its original parameter values, and no noise was added to its simulation. Plotted are the median (strong lines) and interquartile range (shaded) of the standard deviations of the parameter values in question within the algorithm's model populations as a function of fitting epoch.

Secondly, for clarity and to avoid tedious repetition, I will compress data for further analysis in two ways. In the first instance, since I am much more interested in the final outcomes than in the fitting algorithms per se, I ignore the progress of fitting, limiting the data shown instead to only the final 10% of the fitting runs (epochs 450 to 500). Convergence in these epochs is averaged to even out effects related to parameter-specific mutation operators. In the second instance, I will present results in terms of parameter space distance, rather than broken

down by parameter. The latter is not a trivial step and deserves a detailed explanation, as follows.

4.3.1.2 Parameter space distance

The problem, in moving from parameter-specific data to a single representative value, is that each model parameter has a characteristic scale, and a distinct way of affecting its model. As noted previously, some parameters have a multiplicative effect: Doubling e.g. a maximum conductance will approximately double the clamp current under some circumstances. Other parameters are more additive, such as equilibrium potentials, where each additive change will have roughly the same effect regardless of the parameter's magnitude. In addition, not all parameters are equally influential on the model as a whole – the leak equilibrium potential, for example, is relatively insignificant (and therefore also difficult to estimate with any certainty), unless the leak conductance is unusually high. As a consequence, it is not obvious what it means for one parameter set to be close to another. To normalise parameter magnitudes, one could express deviations between models in terms of relative magnitude, but would risk e.g. overestimating the impact of additive parameters that happen to be close to zero. To combine parameter differences into a single value, one could use Euclidean distance in parameter space, but would risk overestimating the impact of relatively inconsequential parameters.

My solution is to use a measure of parameter influence that I already have, namely, the $\bar{\mathbf{\Delta}}$ introduced in section 2.2.2.1, which describes the average current shift caused by changing a given parameter by its detuning step size σ_i. Under the assumption that this is linear and independent of other parameters, we can use this measure to assign to any parameter deviation its expected current shift. Of course, neither of those assumptions hold up particularly well, but for an impact-weighted distance metric, I consider it to be more valid than any of the alternatives considered. Formally, I calculate the parameter space distance de-

scribed by a parameter vector $\mathbf{d}$ as $PD = \sum_i d_i \frac{\bar{\Delta}_i}{\sigma_i}$. Although PD could technically be considered a measure of current, I consider it safer to use it as a dimensionless quantity, given the noted violation of assumptions. This choice also makes sense in light of the very different PD scales seen in the three models used here, which can only partly be attributed to the characteristic scale of the model currents.

4.3.1.3 Results

Using this approach, we will take a closer look at convergence across the three proof-of-concept models. Figure 4.3 shows convergence in the perfect match condition, where the target traces are generated by models with the original parameter sets and not polluted with noise. The data invite a discussion of a few points of interest.

Firstly, there is a very clear trend for the GA populations (left and central groups) to remain more varied than the DE populations. This reflects a fundamental difference between the two types of algorithm: In a GA, both with and without crossover, parameters are perturbed by the mutation size σ, which is independent of the population. Thus, even if a population were to converge on a single point in parameter space, the mutation operator would scatter it for the next epoch, thereby forming a lower bound on expected population variance. Only in target-only weighting, where only one parameter is perturbed in each epoch, can a GA population contract to a smaller region of parameter space, as is clearly the case in almost all cases shown (green markers). In contrast, the perturbation in the DE algorithm is directly derived from the population itself, meaning that, given a scaling factor $F < 1$, a population will tend to contract indefinitely even in the absence of a cost function.

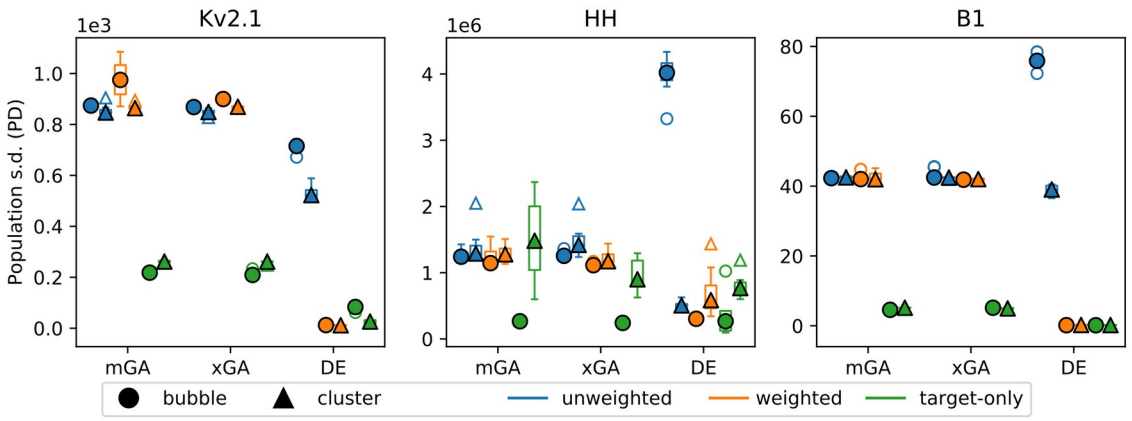


Figure 4.3: Convergence within populations of fits to noise-free reference traces generated by originally parametrised models. Each box and whiskers summarises the standard deviations of parameters in n=8 populations, averaged over the final 50 epochs of fitting, measured in terms of parameter space distance (see section 4.3.1.2). The box represents upper and lower quartiles, and whiskers delimit the full data range, excluding individually plotted outliers. Outliers are defined as data points more than 1.5 times the interquartile range beyond their respective quartile boundary. The filled symbols are placed at the median of each group. Fits are split into groups by fitting algorithm (separated spatially), by weighting scheme (colour) and by stimulus generation method (symbols). Note the different vertical scales, reflecting the characteristic scales of the models in question.

Secondly, among the DE fits, there is a very clear trend for the unweighted scheme to converge much less than the weighted and target-only schemes. This is not entirely surprising, since unweighted fitting ignores the information we have about parameter-specific influences. Mutating the candidates without accounting for parameter sensitivity essentially scatters the population along axes in parameter space that the subsequently applied cost function is ill equipped to constrain. Thus, in unweighted fitting, the pressure of the cost function is effectively reduced. I would argue that this is more apparent in DE fits because, unlike a GA, DE does not discard inadequate candidate solutions outright, unless their own offspring is better; thus, in absence of consistent selection pressure, population-level variance particularly in relatively uninfluential parameters will tend to remain high.

Finally, despite fits within a group being entirely independent, the convergence is remarkably similar across fits in most conditions. Between-group differences largely outweigh any fluctuations due to the random nature of the fitting procedure. Likewise, the relative magnitude of convergence is largely unchanged between the three models, with only few differences. Due to numeric instabilities inherent in the HH model, the results there are less clear, but the general trends seen in the other two models are discernible there, too.

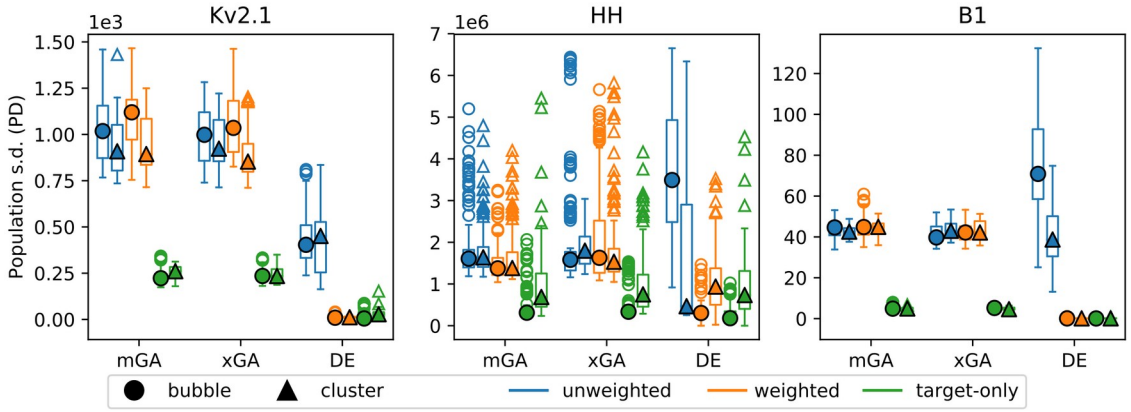


Figure 4.4: Convergence within populations of fits to noise-free reference traces generated by randomly parametrised models, processed and displayed as described in Figure 4.3. Each group summarises fits against 20 independently drawn random parametrisations, plus the original parametrisation, with 8 populations per parameter set, for a total of n=168 fits per group.

Next, I investigated fitting performance when the reference model uses parametrisations not encountered during stimulus generation. In each condition (model x fitting algorithm x weighting scheme x stimulus generation method), I independently drew 20 random parametrisations uniformly from the parameter range of the respective model. These randomised models were used to generate reference traces with the same stimulus sets as above, which were then fed to the fitting algorithms. Since each fitting run was parallelised over 8 populations (see section 4.2.6), each of the randomised models was thus fit 8 times. The results of these fits, together with the original parametrisations, are shown in Figure 4.4.

The general trends described above are largely conserved. Most DE fits converge more strongly than the GA fits; within GA fits, target-only convergence is greater than that of the weighted and unweighted schemes; the relative magnitudes of convergence are approximately the same across all three models; and the unweighted DE fits converge less reliably than their weighted counterparts.

The only major differences are in the DE/bubble fits, with the unweighted convergence edging much closer to its cluster counterpart on average, due to better convergence in the bubble fits and worse convergence in the cluster fits in roughly equal measure. Again, the HH model's results are less clear, with large numbers of outliers due mostly to parameter combinations that appear to be detrimental to the model's numeric performance.

More generally, the fact that convergence remains comparable between the original parametrisation and other, randomly chosen parameter sets as reference indicates that the stimuli are applicable across the full parameter range, fulfilling one of the chief goals of the stimulus generation process.

Finally, I repeated the random drawing of parameter sets, but added Gaussian white noise with a standard deviation of 50 nA (Kv2.1, HH) or 5 nA (B1) to the reference current traces. As before, each stimulus only produced a single reference trace that was cached and reused throughout each fit, and across all eight otherwise independent parallel populations in a given run, mirroring the situation with recorded oocyte data (section 4.4).

The results of these fits are shown in Figure 4.5 and very closely resemble those in the noise-free condition, both in terms of the relative distribution of convergence values, and in terms of their absolute values. I conclude that the impact of noise on convergence is very limited.

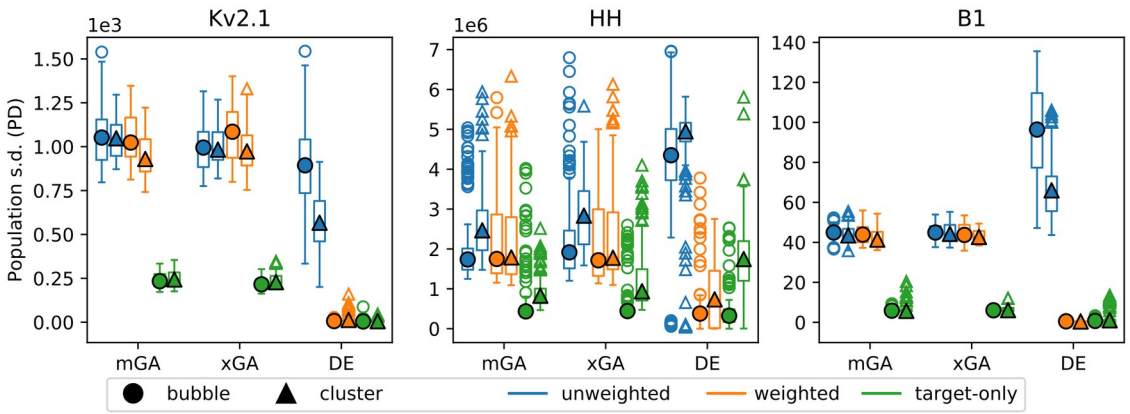


Figure 4.5: Convergence within populations of fits to noisy reference traces generated by randomly parametrised models, processed and displayed as described in Figure 4.3. Again, each group summarises fits against 20 independently drawn random parametrisations, plus the original parametrisation, with 8 populations per parameter set, for a total of n=168 fits per group.

4.3.2 Distance to reference parameters

4.3.2.1 Introduction

A crucial measure of the success of the MOSTIPS method is, of course, its ability to find the true parameter values. In fitting against synthetic data, we have this information readily available and can thus straightforwardly evaluate the accuracy of the various versions of the algorithm. (Reasons for not calling this measure “accuracy” will become apparent with the oocyte results, section 4.4.) Again, to make this analysis more tangible, Figure 4.6 shows an example of the progress of fitting over time. Here, because we are interested in how well the reference model is approximated by fitting, I show the candidate parameter set with the lowest cost in each epoch and population.

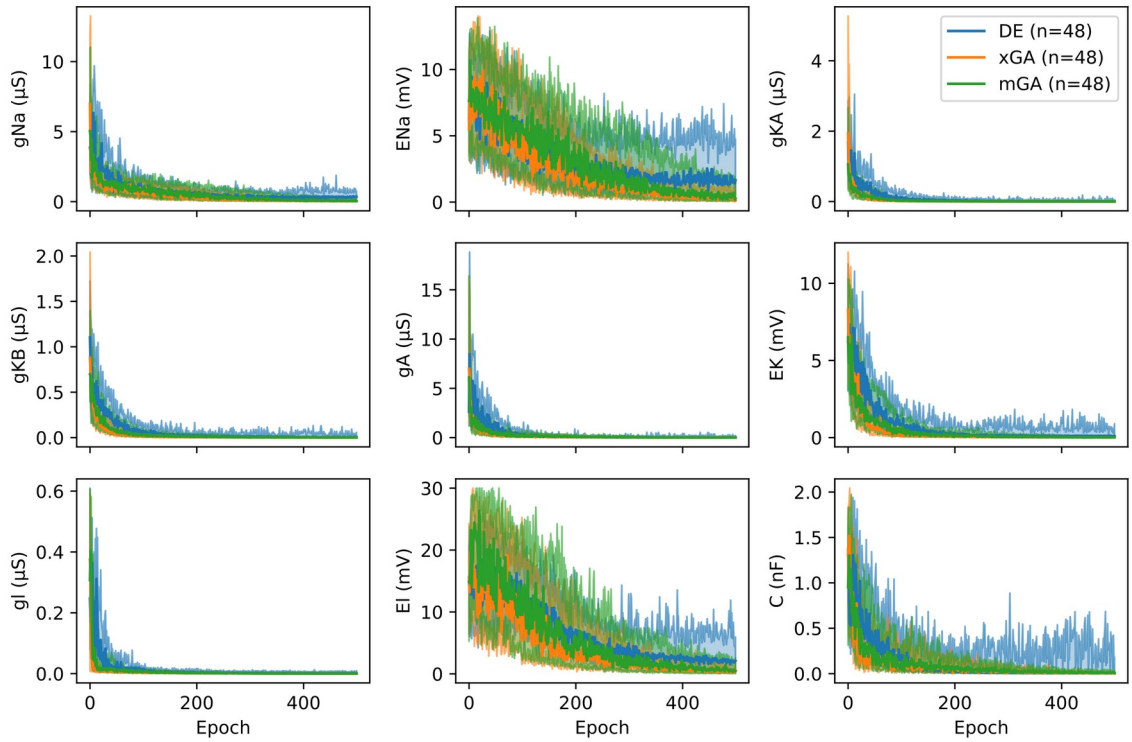


Figure 4.6: Example distance to reference for the B1 model, split by parameter (subplots) and fitting algorithm (colours). The reference model was parametrised with its original parameter values, and no noise was added to its simulation. Plotted are the median (strong lines) and interquartile range (shaded) of the absolute difference between the reference parameter values and every population's best-fit model in each epoch.

As with convergence, the detailed analysis below is presented in terms of overall parameter space distances as defined in section 4.3.1.2 above. I also again ignore the temporal aspect of fitting, being more interested in the final result. To find the final best-fit parameter set from a population independently of the last fitness function applied, I evaluate the algorithm's entire population in the final epoch against all stimuli used during fitting, collecting the error not just in the respective observation windows, but throughout the stimuli, excluding capacitive current artefacts. Then, I select the parameter set with the lowest total error as the algorithm's final output, and plot the parameter space distance between these final models and their respective reference models.

4.3.2.2 Results

Again, we start with the ostensibly easiest case, the noise-free fits against the originally parametrised models. The results are shown in the top row of Figure 4.7, organised in the same manner as above, but using violin plots for distinctiveness. Perhaps the most immediately striking feature of the results is how much less orderly and systematic they are than the convergence shown above. Performance appears to depend not just on the combination of methods, but also on the model used, such that no general trends can be seen that hold across models. Within models, there are clear trends, particularly in the Kv2.1 model with the separation between relatively ill-fitting bubble stimuli and the comparatively much better fitting cluster stimuli, and in the HH model with the cluster/target-only fits approximating the reference model notably less closely than other methods.

With increasing difficulty, however, these trends progressively wash out. Fitting against randomised parameter sets (Figure 4.7, middle row), the disadvantage of the HH cluster/target-only fits disappears, and further adding noise (Figure 4.7, bottom row) largely removes any discrepancies between the Kv2.1 bubble and cluster stimuli. In both cases, direction of change is towards the methods performing worse in the easiest environment, which suggests that there may be a trade-off between performance in specific cases (initial parameters; noise-free data) on the one hand, and performance in the general case (novel parameters; noisy data) on the other hand, with stimuli or methods performing below average in specific cases being more robust to added difficulty.

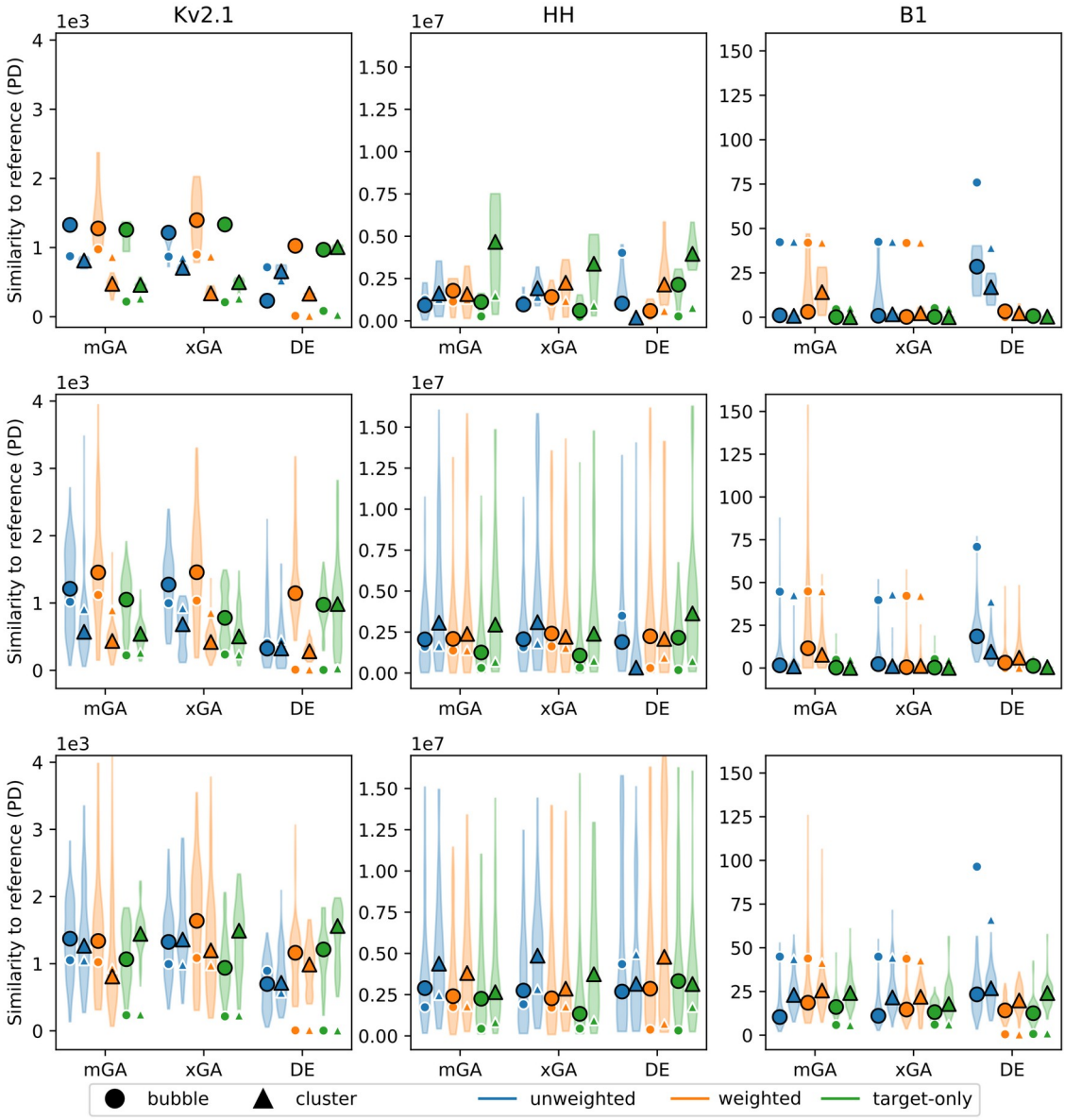


Figure 4.7: Distance to reference for fits against synthetic data. Top row: Fits to noise-free reference traces generated by originally parametrised models (n=8 fits per group). Middle row: Fits to noise-free reference traces generated by randomly parametrised models (n=168 fits per group). Bottom row: Fits to noisy reference traces generated by randomly parametrised models (n=168 fits per group). Each "violin" summarises the parameter space distance (as defined in section 4.3.1.2) to the respective reference model of the final solutions of all populations in a group. The shape (width) of the violins represent a histogram, or the density of solutions at a given value. The large filled symbols are placed at the median of each group, while the small filled symbols represent the respective median convergence, for comparison. Note that the PD scale is model-specific and can not be compared between models.

In order to make a little more sense of the scale of parameter (mis-)match, I report the median distance to reference in each of the three conditions, broken down by parameter, in Tables 4.7-4.9. Keeping the caveats noted in section 4.3.1.2 in mind – neither absolute nor relative terms are particularly clear indicators of the goodness of fit, because the reference parameter values differ in the manner and magnitude of their influence – we can see that the distance to reference differs greatly between parameters.

Perhaps surprisingly, particularly the passive leak conductance (gl, El) and capacitance (C) are not well constrained. A likely cause for this is that these parameters are relatively uninfluential compared to those governing active conductances, which makes it difficult for the fitting algorithm to reliably separate good estimates from bad in models that show any mismatch in other parameters. However, since these parameters are relatively simple to estimate with other methods (as demonstrated in chapter 3), they are also the least important to "get right" – in an experimental situation, it would be sufficient to briefly fit these by classical methods in advance, then let the MOSTIPS fitting algorithm either use such estimates as fixed values, or perhaps adjust them within a narrow range.

Of the remaining parameters, those governing potassium conductances are largely very well constrained, whereas the sodium conductances are evidently more difficult. This is not a great surprise, given that the sodium conductances inactivate much faster, giving the fitting algorithm much less data to work with than for the more slowly (or not at all) inactivating potassium conductances.

Kv2.1	gK (µS)	EK (mV)	gl (µS)	El (mV)	C (nF)
Noise-free, initial parameters (n=144)	1.35E+00	1.66E+00	5.52E-01	1.36E+01	5.00E+01
	2.25%	2.07%	551.61%	136.36%	33.34%
	0.4	1.7	1.5	3.9	6.2
Noise-free, random parameters (n=3024)	1.35E+00	1.35E+00	9.38E-01	6.98E+00	3.19E+01
	1.68%	1.77%	23.73%	97.50%	27.40%
	0.3	1.1	2.3	3.5	5.6
Noisy, random parameters (n=3024)	3.73E+00	3.34E+00	1.55E+00	1.39E+01	4.77E+01
	4.05%	4.30%	32.28%	141.09%	42.15%
	0.7	2.5	3.7	6.6	6.3

Table 4.7: Kv2.1 distance to reference by parameter in terms of absolute values (top rows), percentage of the reference value (middle rows) and z-score (bottom rows). Reported are the median values across fits in all method combinations. Relative values are calculated using the respective reference parameter values of each fit and are therefore not consistent across conditions. Z-scores are calculated per fit as the fraction of the distance to reference and the standard deviation of the respective candidate population in the final 50 epochs.

HH	gNa (µS)	ENa (mV)	gK (µS)	EK (mV)	gl (µS)	El (mV)	C (nF)
Noise-free, initial parameters (n=144)	1.32E+01	9.96E+00	9.07E-02	2.24E-01	5.64E-03	8.09E-01	1.25E-01
	10.99%	22.14%	0.25%	0.27%	1.88%	1.35%	12.53%
	1.4	3.4	0.1	0.3	0.5	0.5	1.2
Noise-free, random parameters (n=3024)	1.75E+01	1.02E+01	2.18E-01	5.12E-01	1.05E-02	1.17E+00	5.96E-02
	12.36%	22.68%	0.53%	0.63%	3.10%	2.02%	5.04%
	1.9	3.4	0.3	0.7	0.7	0.8	0.5
Noisy, random parameters (n=3024)	2.58E+01	1.23E+01	7.40E-01	1.91E+00	4.73E-02	4.43E+00	2.64E-01
	18.31%	27.46%	1.88%	2.37%	13.96%	7.84%	21.63%
	1.9	3.1	0.7	1.7	1.6	1.6	1.6

Table 4.8: HH distance to reference by parameter in terms of absolute values (top rows), percentage of the reference value (middle rows) and z-score (bottom rows). Reported are the median values across fits in all method combinations. Relative values are calculated using the respective reference parameter values of each fit and are therefore not consistent across conditions. Z-scores are calculated per fit as the fraction of the distance to reference and the standard deviation of the respective candidate population in the final 50 epochs.

B1	gNa (µS)	ENa (mV)	gKA (µS)	gKB (µS)	gA (µS)	EK (mV)	gl (µS)	El (mV)	C (nF)
Noise-free, initial parameters (n=144)	7.11E-02	4.45E-01	7.22E-04	9.67E-04	5.45E-03	2.00E-02	2.24E-04	6.97E-01	4.50E-03
	1.02%	1.27%	0.05%	0.03%	0.05%	0.03%	1.12%	3.48%	0.13%
	0.08	0.39	0.00	0.01	0.01	0.04	0.01	0.37	0.06
Noise-free, random parameters (n=3024)	1.90E-01	1.06E+00	3.51E-03	3.70E-03	1.70E-02	2.64E-02	9.91E-04	2.04E-01	9.71E-03
	2.30%	3.10%	0.09%	0.08%	0.10%	0.04%	0.30%	1.28%	0.30%
	0.24	0.73	0.02	0.03	0.02	0.05	0.06	0.15	0.11
Noisy, random parameters (n=3024)	1.86E+00	7.50E+00	1.76E-01	1.98E-01	5.96E-01	5.72E-01	2.03E-02	2.52E+00	4.19E-01
	18.20%	22.86%	3.66%	4.38%	4.00%	0.83%	5.45%	17.27%	11.29%
	2.23	5.08	0.62	0.57	0.53	0.60	0.77	1.36	2.24

Table 4.9: B1 distance to reference by parameter in terms of absolute values (top rows), percentage of the reference value (middle rows) and z-score (bottom rows). Reported are the median values across fits in all method combinations. Relative values are calculated using the respective reference parameter values of each fit and are therefore not consistent across conditions. Z-scores are calculated per fit as the fraction of the distance to reference and the standard deviation of the respective candidate population in the final 50 epochs.

4.3.3 Conclusions

In conclusion, these results are a promising start to this challenging problem. While not all parameters are well constrained to the known reference value, the discrepancies are generally small to medium in size, suggesting that the method is capable of achieving reasonable fits.

Although there are clear differences between the various combinations of methods in terms of convergence within candidate populations, these differences largely disappear when considering the accuracy of the parameter estimates. One could express fitting performance in terms of posterior probabilities (i.e., how likely does the algorithm "think" the true solution is, considering the location and spread of the candidate population in parameter space), thus making use of both convergence and distance to target and ostensibly gaining information about the algorithm's precision. However, most major differences between method combinations in convergence are essentially artefacts of the way the fitting algorithms work. Therefore, the level of convergence cannot be reliably interpreted as indicative of the algorithm's precision, particularly not when comparing between methods. Since all method combinations seem to perform roughly on par with each other, going forward into real electrophysiological data, we will not discard any of them, instead repeating the same pattern of analysis, with added checks to thoroughly test the MOSTIPS method.

4.4 Fitting against oocytes expressing ion channels

At last, we come to the proof of concept with real electrophysiological data. In the following, I will present a similar analysis to that above, with some obvious differences. Here, there is no question of artificially adding noise to the data, since that's already included at a magnitude on the order of 10-100 nA at holding potential. Likewise, there is natural variation in the reference parameter values, since channel expression can only be partially controlled by adjusting the amount

of cRNA injected, and I did not attempt to explicitly vary expression levels. Finally, the reference parameter values themselves are merely estimates, derived from the classical protocol data as described in chapter 3.

In addition to analysing convergence and similarity to reference as above, we will look at the data and fits in two additional ways. The first is a cross-validation analysis explained in more detail in the relevant section (4.4.4) below, while the second takes the form of a randomised control, as follows.

4.4.1 Random observation controls

A question I have so far left entirely unexamined is whether the stimulus generation approach that I have taken actually does anything useful. After all, despite having thought deeply about how to find good stimuli and observations, and indeed about what “good” means in this context, I have yet to show that my stimulus generation methods are functional. One might argue that successful fitting is a sufficiently positive answer; however, it is conceivable that any fitting success is in fact exclusively attributable to the fitting algorithm, rather than the stimuli or observations. To control for this, I have chosen to reuse the stimuli and their recordings, but with a twist: Rather than use the algorithmically chosen observation windows, the observation windows were scattered across their stimulus set, as follows.

Remember that each stimulus/observation is associated with a target parameter. The association between parameter and stimulus was randomly reshuffled within a given stimulus set, while the observation windows were retained with their original parameter, but placed into the new stimulus at random time points. In other words, each parameter retained the duration of its observation(s), but the observations were placed at random times in a randomly selected stimulus. These new stimulus/observation pairings were then used in two ways. Firstly, to control for the effect of selecting appropriate observation windows, I performed target-only fitting. Secondly, to control for the effect of stimulus design, I evaluated the new

stimulus/observation pairings under the cluster algorithm, and used the resulting sensitivity vector for weighted fitting. This is an effective control, because, while the weighting reflects the parameter sensitivities in the observed data, the stimuli are not optimised to emphasise any particular parameter. Finally, unweighted fitting was not attempted with randomised observations, as it is unclear how such fits could be interpreted, and the fitting algorithms used were limited to xGA and DE. I note that each set of 8 fits (i.e., a single fitting run using a single fitting algorithm, stimuli, weighting scheme, and set of recorded current traces) used an independently randomised stimulus/observation pairing.

4.4.2 Convergence within populations

As for synthetic data, we will first consider the convergence within populations of the various fitting algorithms. Figure 4.8 shows the time course of convergence for the Kv1.4x model, which is largely representative of the progress also in Kv2.1x and Kboth. For most parameters, convergence is largely complete within 200 epochs. Again, the DE populations converge more slowly, but much more thoroughly, while substantial variance remains in the GA populations. The Kv2.1k model with its 28 parameters (Figure 4.9) shows a similar trajectory, though convergence is both slower and less pronounced, particularly in DE fits and for the parameters to the kinetic equations.

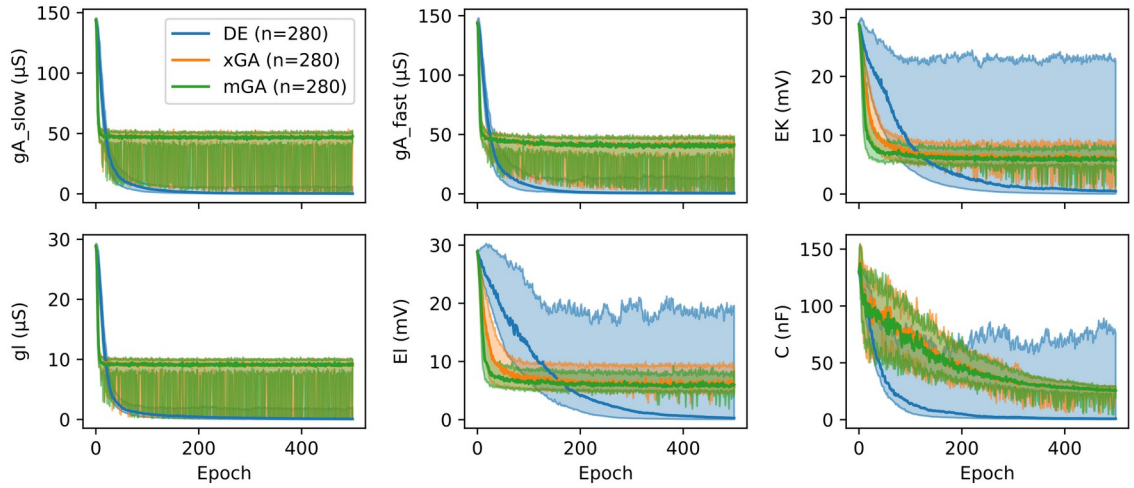


Figure 4.8: Example convergence for the Kv1.4x model, split by parameter (subplots) and fitting algorithm (colours). Plotted are the median (strong lines) and interquartile range (shaded) of the standard deviations of the parameter values in question within the algorithm's model populations as a function of fitting epoch. For a description of this model and its parameters, refer to sections 4.2.1.3 and 3.5.4.

Moving to parameter-space distance and separating convergence by the combination of methods in each fit (Figure 4.10), we again find pronounced differences along similar lines as shown for synthetic data. Firstly, weighted and target-only DE fits converge more strongly than most of the GA fits. In contrast, unweighted DE populations remain spread out, both relative to the other DE weighting schemes, and – except in Kv1.4x – also relative to the GA populations. In the Kv2.1k fits using unweighted DE, I observe that populations appear to be driven into the corners of parameter space (i.e., the distribution of values along any given parameter axis is bimodal, with peaks at either end of the permitted scale and only a small number of models in between). This effect is very consistent, but I have not been able to find its root cause, and have not pursued the issue further because I consider the unweighted scheme a null control; if it fails, as it does in this particular case, this serves to show that scaling mutations by parameter influence does indeed have a positive effect.

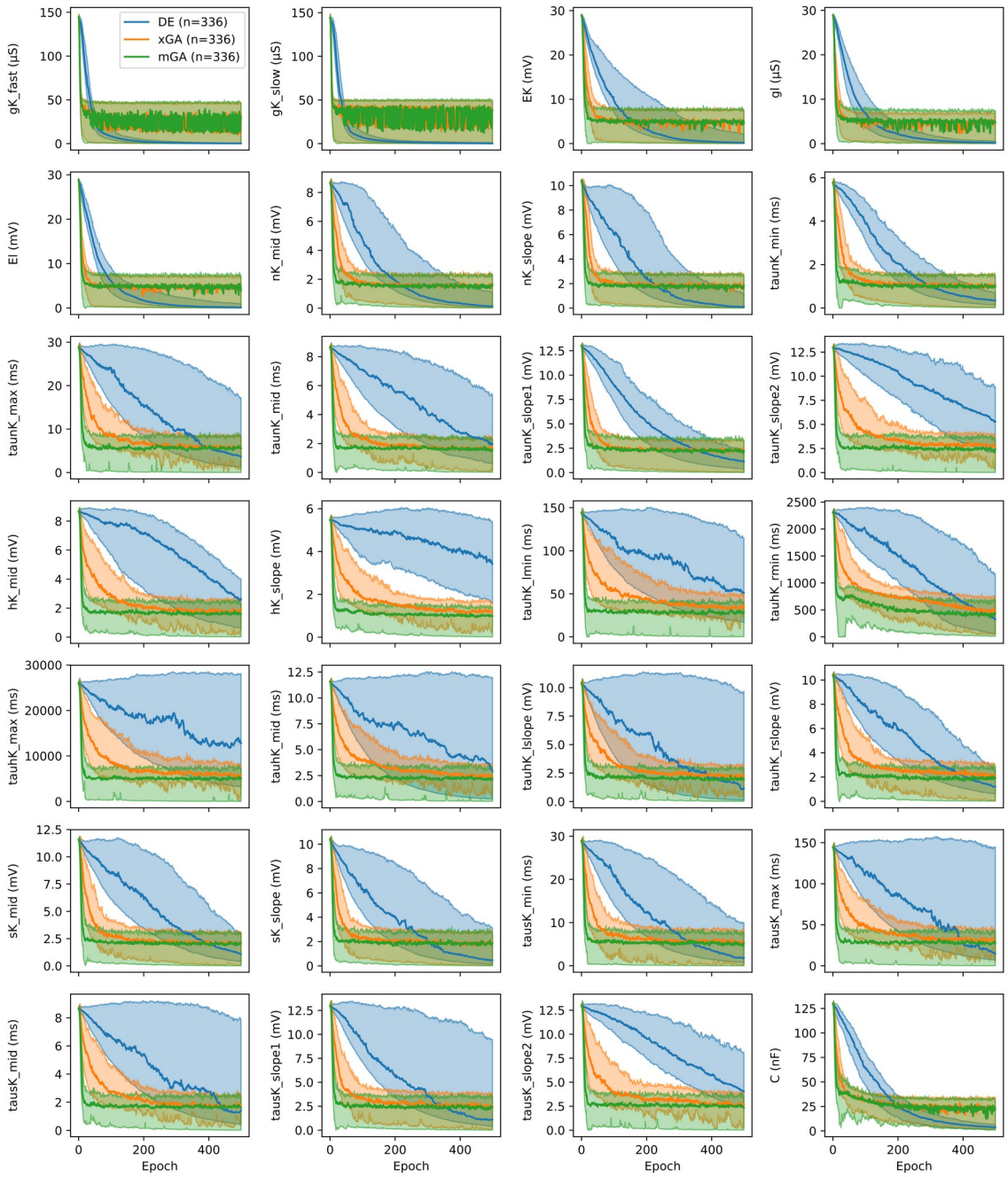


Figure 4.9: Convergence in the Kv2.1k model, including cluster and bubble stimuli fitted with weighted and target-only mutation schemes. Unweighted fits did not converge and are excluded. Note the relatively fast convergence for the passive parameters (potassium and leak conductances and equilibrium potentials in the first five subplots, and capacitance in the bottom right corner), in contrast to the kinetic parameters, particularly in the DE algorithm. For a description of this model and its parameters, refer to sections 4.2.1.2, 3.5.2, and Table 3.2.

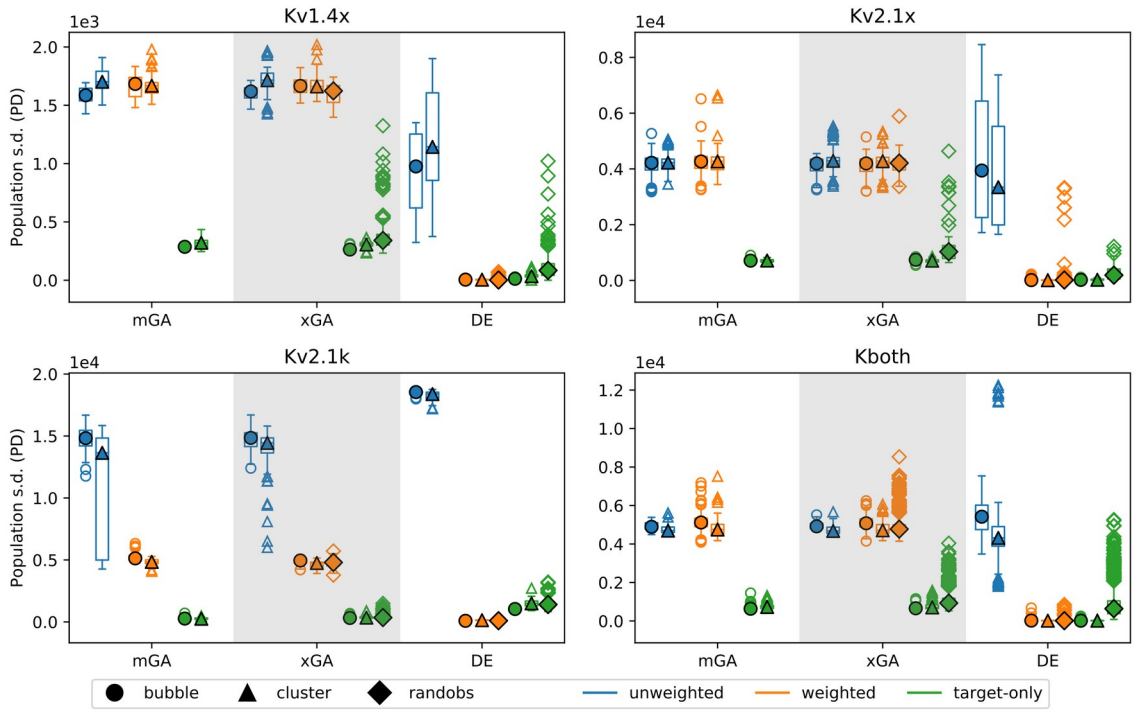


Figure 4.10: Convergence within populations of fits against oocyte data. Each box and whiskers summarises the standard deviations of parameters in a given population, averaged over the final 50 epochs of fitting, measured in terms of parameter space distance (see section 4.3.1.2). The box represents upper and lower quartiles, and whiskers delimit the full data range, excluding individually plotted outliers. Outliers are defined as data points more than 1.5 times the interquartile range beyond their respective quartile boundary. The filled symbols are placed at the median of each group. Fits are split into groups by fitting algorithm (separated spatially), by weighting scheme (colour) and by stimulus/observation pair generation method (symbols). The method labelled "randobs" (diamonds) refers to the randomised stimulus/observation pairings used as controls, generated as described in the introduction section 4.4 above.

Secondly, there is strong convergence in target-only GA fits across both stimulus generation algorithms. As noted in the previous section on synthetic data, the reason for this is that the unweighted and, to a lesser extent, weighted mutation operator keeps the population variance above a lower bound set by the mutation step σ, whereas in target-only fitting, the off-target parameters are not perturbed, and thus not prevented from converging to a narrower region. Interestingly, in the Kv2.1k model, populations in the weighted scheme converge much more readily than those in the unweighted scheme. A potential explanation is

that the average weighting of a parameter across stimuli is lower in this model, due to the high specificity of many of the parameters. That is, kinetic parameters are, at least theoretically, more separable, because their effects are limited to a small range of possible stimulus/observation pairs. To see this, consider e.g. the tau_max parameters, governing the amplitude of the τ curve of their respective gating variable (Figure 3.15). To experience any significant influence from this parameter, a stimulus needs to both hit a relatively narrow voltage range, corresponding to the width of the τ curve, dwell there for an appropriate duration corresponding to the curve's amplitude, and then find a stimulation regime that reveals the resulting gating state. Since all kinetic parameters have similarly convoluted stimulus/observation requirements, it should not be surprising that they are well separated, thus leading GA fits with the corresponding weight vectors to converge more strongly than their unweighted analogues. Conversely, static parameters (conductance, equilibrium potentials) have a much more diffuse effect, and are thus less separable, and much more likely to exert an influence over a very wide range of stimuli.

Finally, we see that the randomised control observations (diamonds in Figure 4.10) converge almost as well as the effective stimulus/observation pairs, regardless of the weighting scheme used. This seems to indicate that neither the applied stimulus nor the observation of a particular chunk of data have a notable impact on convergence within populations. Instead, the pattern of convergence we see across non-control fits appears to be largely dictated by the combination of fitting algorithm and weighting scheme, but not by the quality of the stimulus or observation. In the weighted controls, one might argue that the often short observations allow a sensible choice of weight vectors (i.e., weights that are in fact reflective of parameter influence in the observed period) even in the absence of a targeted choice of stimulus or observation, and so the comparable result is perhaps not too surprising. Conversely, the target-only controls evaluate parameters on data that is potentially (and in the case of kinetic parameters, as ar-

gued above, likely) not strongly influenced by that parameter. Even so, however, unless a given observation is entirely insensitive to its randomly associated parameter, there will be a fitness gradient, and since neither of the two fitting algorithms make use of the value of the gradient (GA uses population-wide rank order, DE only refers to the lowest-cost individual in a population and relative fitness between parent and offspring), we should not expect any difference in convergence locally, i.e. in the context of similar parameter sets moving from one epoch to the next. Globally, however – across disparate parameter sets and over the course of many epochs – we might expect that non-optimal observations are unstable, i.e. that their gradient points in inconsistent directions, which would hinder convergence. Given that most target-only fits converge no worse under control conditions than with optimised observations, it would appear that either random observations are unexpectedly stable, or the observations to come out of the bubble and cluster algorithms are less so.

4.4.3 Distance to reference parameters

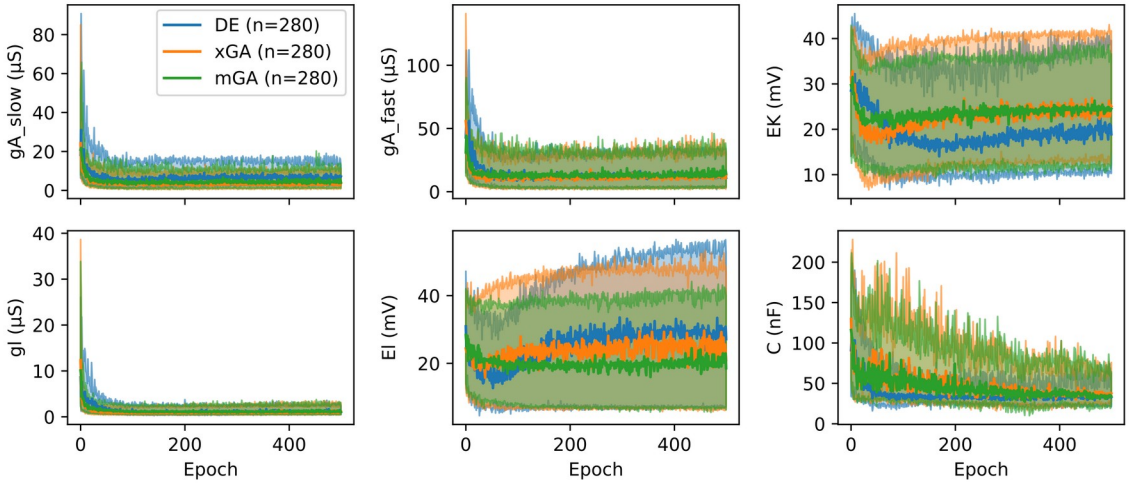


Figure 4.11: Example distance to reference for the Kv1.4x model, split by parameter (subplots) and fitting algorithm (colours). Plotted are the median (strong lines) and interquartile range (shaded) of the absolute difference between the reference parameter values and every population's best-fit model in each epoch.

Next, we will look at the distance between fitted models and their reference parameter sets. As a reminder, the reference parameters are derived from the combined measuring and fitting protocols described in chapter 3, and should therefore be seen not as a ground truth as such, but only as a reliable second opinion. As in the previous section, Figure 4.11 shows parameter-wise distance to reference for the Kv1.4x model, with the trajectories following similar patterns for the Kv2.1x and Kboth models. Here, the fitting algorithm does not seem to have a significant effect on the outcome. I note that the median lines' reversal in the early epochs of the EK and El parameters, which might be interpreted as overfitting, is in fact merely an artefact of presenting the absolute distance. In Figure 4.12, we see the parameter-wise distance to reference of Kv2.1k fits, which show very considerable discrepancies that, for many of the kinetic parameters, tend to increase as fitting proceeds. While there are some differences in performance between the fitting algorithms in individual parameters, these are neither significant nor systematic.

Examining distance to reference in terms of parameter-space distance and splitting the results by the combination of methods used, we arrive at the data shown in Figure 4.13. Unfortunately, unlike convergence within populations, there is very little systematic difference in performance between any of the models. The one notable systematic effect is a high distance to reference in the Kv2.1k model, i.e., in fits that failed to properly converge; however, the effect is much smaller than might be expected, and there is considerable overlap between these and the weighted and target-only fits. Even the control stimulus/observation pairings (diamonds) achieve similar performance.

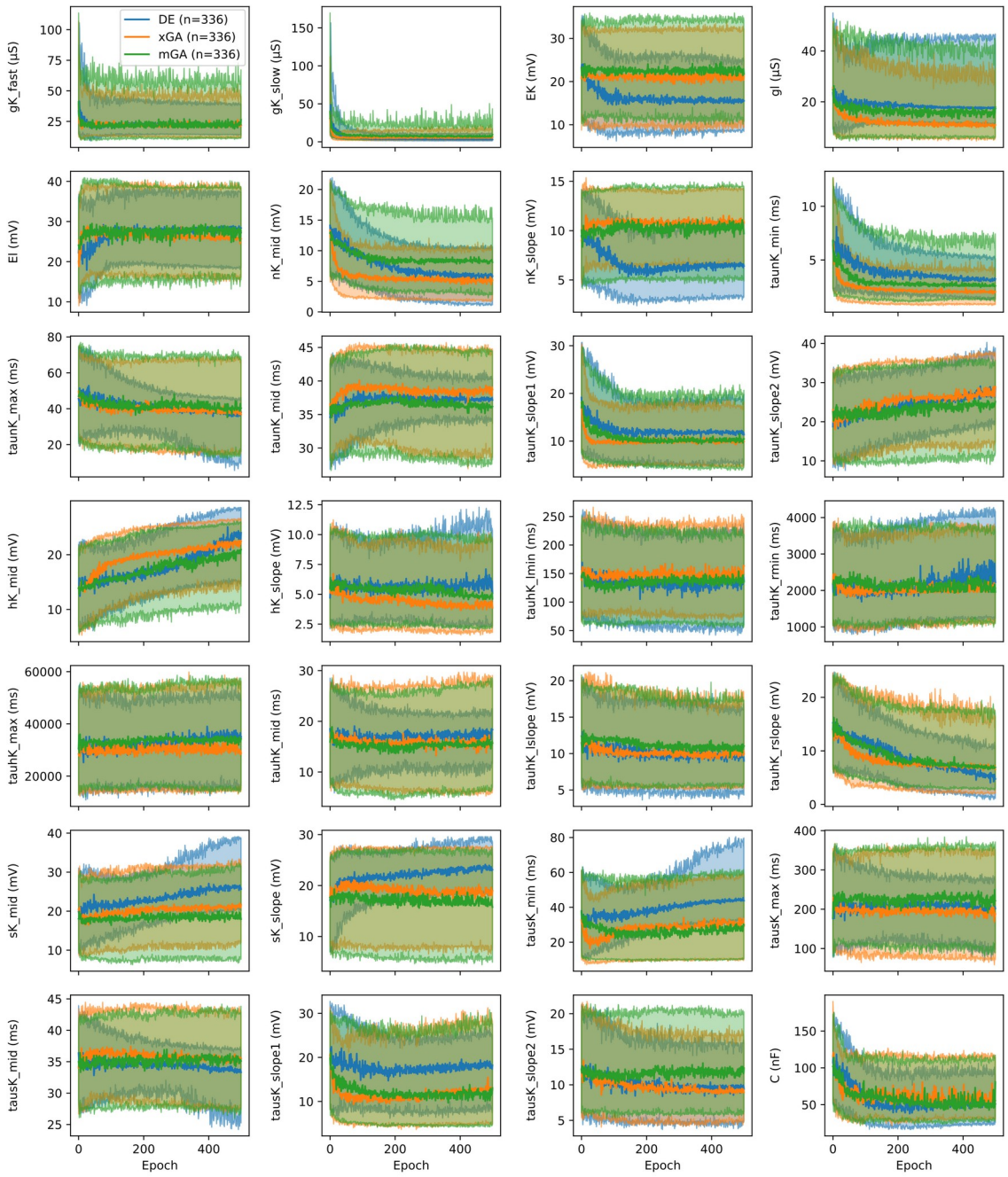


Figure 4.12: Distance to reference in the Kv2.1k model, again including weighted and target-only, but not unweighted fits.

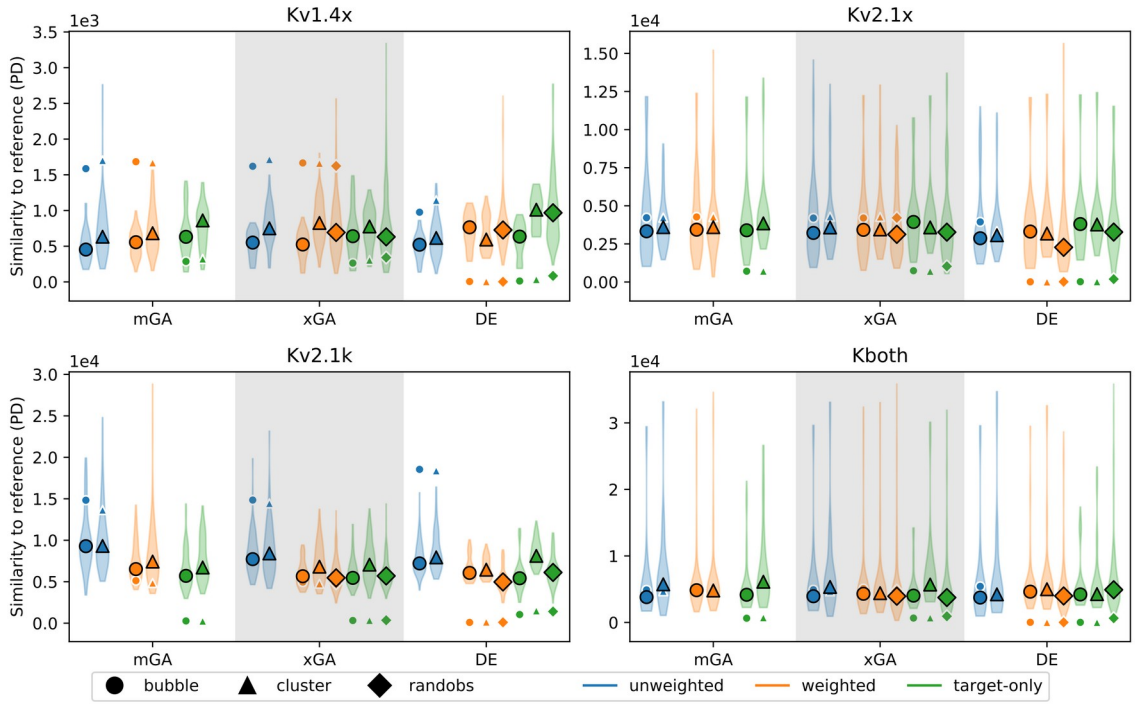


Figure 4.13: Distance to reference for fits against oocyte data. Each "violin" summarises the parameter space distance (as defined in section 4.3.1.2) to the respective reference model of the final solutions of all populations in a group. The shape (width) of the violins represent a histogram, or the density of solutions at a given value. The large filled symbols are placed at the median of each group, while the small filled symbols represent the respective median convergence, for comparison. Note that the PD scale is model-specific and can not be compared between models. The method labelled "randobs" (diamonds) refers to the randomised stimulus/observation pairings used as controls.

This – admittedly rather boring – outcome mirrors the relative indifference of fitting results to method combination when noisy synthetic data were used as targets (section 4.3.2.2). Given that the parameter space distances correspond to rather large deviations from the reference value (cf. Figures 4.11 and 4.12), and that the variance within populations is of a similar order of magnitude for most fits, it is perhaps instructive to consider whether we are seeing a failure to fit at all. If so, we would expect fits using a given method combination and data set to disagree amongst each other, too.

Figure 4.14 shows that this is partly the case, particularly with the unweighted Kv2.1k fits, but also in several other conditions, including all Kv2.1k GA fits,

weighted and unweighted mGA in all models, and to a lesser extent weighted and unweighted xGA. Conversely, weighted and target-only DE fits almost universally agree much better among each other than with the reference model. In this last point, and indeed across all three weighting schemes for DE, these results closely resemble the convergence within populations. This suggests that convergence within populations could indeed be interpreted as a measure of the uncertainty of the DE algorithm's results. It also shows that, unlike most of the GA fits, weighted and target-only DE produces internally consistent estimates. This consistency, however, does not extend beyond the boundaries of a given group: Convergence across different combinations of weighting scheme and stimulus generation method is on the same order of magnitude as similarity to reference (data not shown).

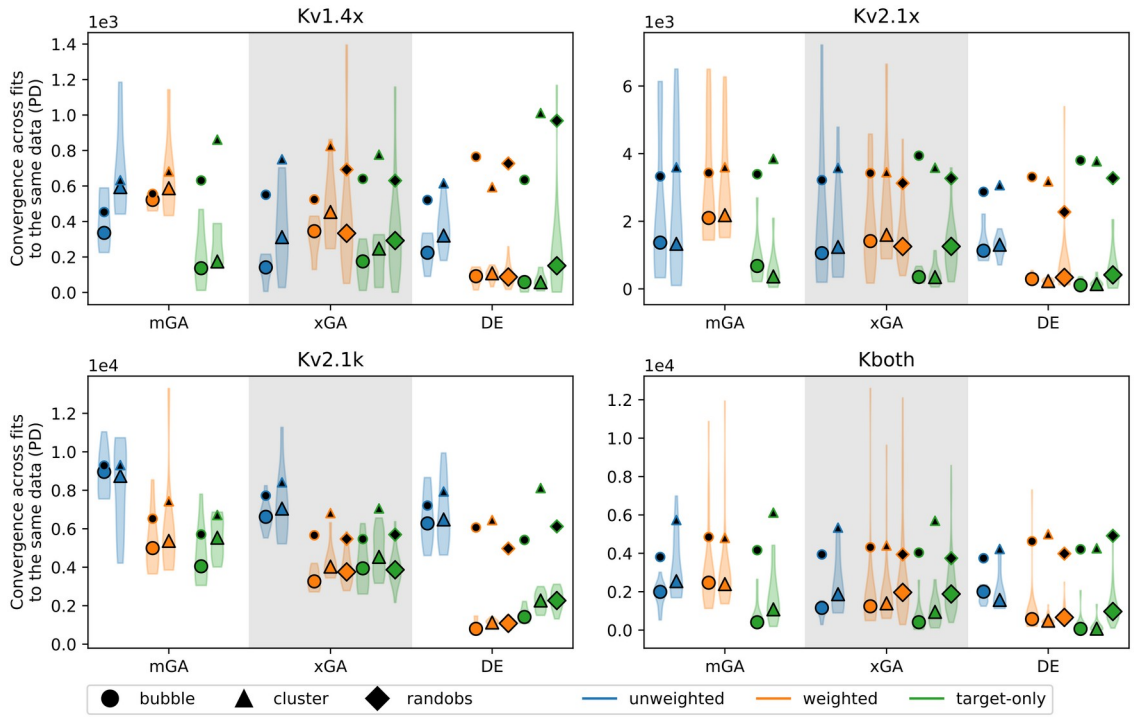


Figure 4.14: Convergence across fits to the same data, calculated as the PD of the standard deviation of the final model of all fits in a given group that were made using a given data set. The small symbols filled with black reproduce the median similarity to reference from Figure 4.13.

Disappointingly, the control observations do not show significantly greater variability within fits to a given data set nor greater distance to reference, suggesting

that stimulus and observation selection is not a strong factor in determining the result of fitting.

4.4.4 Cross-validation

Finally, we will examine the fitting results in terms of cross-validation, a common machine learning practice and a measure of how well the chosen model generalises to previously unseen data. Remember that each oocyte yielded several different data sets, including both the classical and various MOSTIPS stimulation protocols. In addition, even in fitting against data from a given MOSTIPS stimulus set, the fitting algorithms' cost function is only exposed to the observed regions, typically only a small part of the total data collected. Thus, for cross-validation, I used the full set of recordings taken from a given cell, reporting the root mean squared clamp current error of a fitted model. It may be worth noting that this lends an advantage to the classically fitted models, which were fitted to large portions of the activation and tail current protocol data, and have therefore been exposed to a much larger fraction of the total recordings than the MOSTIPS fits.

In reporting the outcome of cross-validation, I have chosen to relate the MOSTIPS results directly to their respective reference fits' results, presenting an error ratio. Note that this does not introduce bias to the analysis, since the reference fits are shared across all recordings of a given cell. The mean raw error values across all method combinations are recorded in Table 4.10. Leaving aside the Kv1.4x and Kboth fits (see below for a closer analysis of those), it is interesting to note that the Kv2.1k parameter sets, fitted both by classical and by MOSTIPS methods, generalise less well overall than the Kv2.1x fits, even though they are generated from the exact same data (reference fits) or at least using data from the same cells (MOSTIPS fits). This suggests that fitting the kinetic parameters leads to some amount of overfitting. I note this with a mixture of satisfaction and disappointment, as it both vindicates my choice of (fixed) kinetic parameters for

the Kv2.1x model, but also shows that none of the systematic methods I used, classical or MOSTIPS, could outperform my careful manual tuning.

	Kv1.4x	Kv2.1x	Kv2.1k	Kboth
Reference error	264	1141	2086	799
MOSTIPS error	348	1164	1394	1183
MOSTIPS error (excluding unweighted fits)	350	1176	1226	1199

Table 4.10: Cross-validation error of the reference parameter sets (ϵ_0) and the MOSTIPS fits (ϵ) in nA. Note that the Kv2.1 currents are, due to their slow inactivation, greater than the Kv1.4 currents even at comparable expression levels, which leads to naturally higher error values.

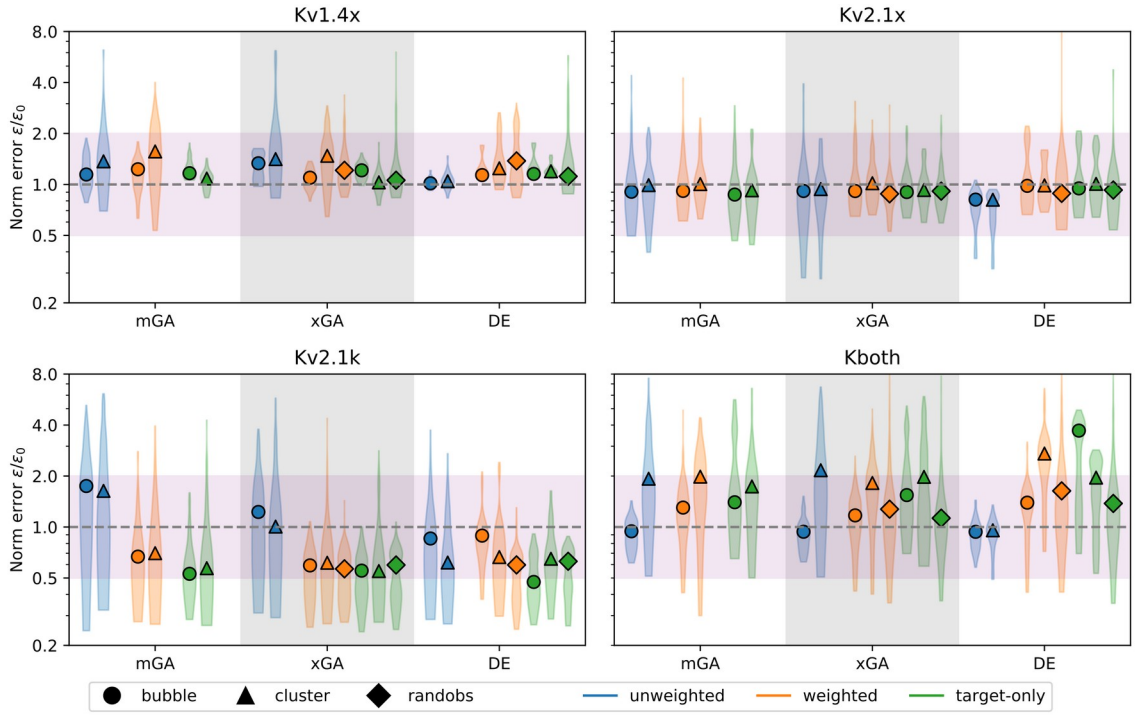


Figure 4.15: Cross-validation error of the final parameter selections (ϵ), compared to that of the reference parameter set (ϵ_0). Values below the dotted line indicate an improvement over the reference, with the purple shaded area containing all MOSTIPS results that generalise between half and twice as well (upper and lower boundary, respectively) as the reference parameter sets.

In Figure 4.15, we see more clearly how strongly the outcome of cross-validation differs between models. The Kv1.4x and Kboth model MOSTIPS fits both generalise worse, overall, than their classical counterparts, with the Kboth model

showing much greater variability. In the Kv1.4x fits, there is no systematic pattern relating the cross-validation performance to any of the method combinations. In the Kboth fits, conversely, there is a noticeable difference between bubble and cluster fits, with the latter performing worse in combination with either of the GA fitting methods.

In contrast to the poor performance of unweighted fitting in terms of convergence, similarity to reference, and even consistency across fits to the same data, we see surprisingly good unweighted performance in Kboth and Kv2.1x cross-validation, with particularly unweighted DE in both models outperforming the classical fits.

Both Kv2.1x and Kv2.1k MOSTIPS fits largely generalise better than the classical fits, with the exception of the Kv2.1k unweighted fits. Ignoring those, the Kv2.1k cross-validation results are by far the most promising, very clearly outperforming the classical reference values. As noted above, this is at least in part driven by overfitting in the classical case, which the MOSTIPS fits appear to be more resistant to.

Finally, I note that the control observations perform no worse than the algorithmically designed ones, regardless of weighting scheme, fitting method, or model used.

4.4.5 Conclusions

In the sections above, I have detailed how well MOSTIPS fits against oocyte data converge within populations, how well each population's best solution agrees with the reference parameter values, as well as with other fits using the same data and method, and how well the resulting parameter sets generalise to other data harvested from the same cells. I have shown that, while there are clear differences in the population trajectories between the various method combinations, the final outcomes, both in terms of similarity to reference and of cross-validation, do not appear to depend much on the particular methods chosen.

The only clear exception to this are the unweighted Kv2.1k fits, which failed to converge, thereby also impeding the refinement of any high-quality models present in the population. This failure is likely driven by the sheer size of parameter space, both in terms of its dimensionality and in terms of the relatively large range of parameter values. It is likely that narrower value ranges would have aided the fitting process by restricting the search to a more manageably sized area of parameter space. However, considering that the weighted and target-only fits for the same model produced results that compare favourably to the reference values, the loose limits I have chosen are clearly an impediment only to a blind search, but not to a search guided by added information about the observed data.

At face value, fitting with just about any MOSTIPS method appears to yield comparable results to the classical fitting method. There are two main points of contention to this, however. Firstly, the controls that I have chosen – arbitrarily placed observations – performed as well as the optimised observations on all measures investigated, both in weighted mode, where parameter-specific mutation sizes were related to the predicted parameter sensitivity in the observation windows, but also in target-only mode, where the relationship between observation and mutated parameter was entirely arbitrary. The unexpected success of these controls appears to suggest that the entire stimulus generation algorithm, including both the evolution of particular stimuli and the selection of suitable observations, is quite unnecessary, and that randomly observing chunks of data generated by an arbitrary process could yield similar results.

While I cannot dismiss that possibility, I note on the one hand that the stimuli produced, and used for both effective and control observations, are of course a particular subset of all possible stimuli; they are designed to perform parameter separation. This may mean that observations of data generated by these stimuli, even at arbitrary time points, are likely to be unusually sensitive (or insensitive) to certain parameters, compared to arbitrary stimuli. Conversely, robustness

screening and the stimulus selection procedure, relying strictly on the observation windows suggested by the stimulus generation algorithm, clearly does not have a favourable effect on the fitting outcomes. On the other hand, the results suggest that alternating the mutated parameters or differently modulating the relative size of mutations across epochs may be a powerful way to constrain models that are otherwise difficult to fit, and thus vindicates the key idea of parameter separation on the fitting side of the process, if not on the observation side.

Secondly, I note that both the amount of data and the amount of wall-clock time used are considerably less for MOSTIPS fitting than for the classical, least-squares fits. Conversely, the kinds of data observed are much more varied, including ramps as well as constant holding voltages, and exhibiting a wider range of voltages during and, particularly, before observation. These stimulus acrobatics are encouraged by the stimulus generation cost functions, since they drive the model to unusual places in state space. That even very short observations of this kind are sufficient for high-quality fitting reinforces the notion that forcing the system into unusual states is a useful method of learning about the system even in the absence of a targeted observation strategy, as noted above.

5 Closed-loop approaches

5.1 Introduction

The ultimate goal of optimising models to individual cells is, of course, not to parametrise models of genetically engineered oocytes, but of real neurons. In the first part of this chapter, I present a model optimisation method that builds on the work covered in the preceding chapters, taking some of the core ideas of the MOSTIPS approach, but transposing them into the current clamp domain and closing the loop between neurons and models. I will elucidate the guiding principles of the method and present results from a small number of pilot experiments.

In the second part, we will leave the model fitting world behind and cover some new ground in dynamic clamp techniques. Specifically, I have expanded the feature range of the Windows-based dynamic clamp software StdpC, and report here the major additions and improvements made, along with an account of their experimental application.

5.2 Closed-loop model fitting

5.2.1 Motivation

The MOSTIPS method, as described in chapters 2-4, was developed mainly in the expectation of voltage clamp data (see section 2.2.1 for a justification). From an experimenter's perspective, however, voltage clamp is significantly more difficult to use: Even under ideal conditions, the method causes non-negligible artefacts due to its sensitivity to details in instrumentation. In neurons, there is the additional difficulty of achieving what is called "space clamp", or full control over the entire membrane; there are physical limits to how far, and how fast, the injected current can spread into axons and dendrites. In cells with active compartments

outside the soma, this can make robust control even of somatic membrane potential impossible, as e.g. action potentials may arise in uncontrolled distal compartments and systematically distort the detected clamp current.

On the other hand, the MOSTIPS methods as presented in this thesis is not well suited to current clamp data without some modification. In particular, the cost function in voltage clamp – root mean squared (current) trace error – would yield a very heterogeneous fitness landscape in current clamp, trapping any search algorithm in one of many local optima. While developing a better cost function is certainly possible, as shown below, the stimulus generation and selection steps are of course also sensitive to such a cost function, making its development and testing a potentially very long-winded process.

More importantly, the parameter separation idea at the core of the stimulus generation process is less obviously applicable to current clamp data, regardless of the cost function. Without control over the model state, the task of finding stimulation/observation pairs that robustly highlight a given parameter – challenging enough in voltage clamp – is likely to be prohibitively difficult, if not impossible, in current clamp.

5.2.1.1 An alternative approach

As an alternative, I decided to try a different approach, distancing myself from the focus on individual parameters and instead embracing the possibilities afforded by immediate interactions between a living neuron and a population of candidate models. The key idea is that a population of candidate models can be treated as an estimate of the true parameter values, and that this estimate can be refined in a targeted manner. In particular, given that a model population has a certain variability in its behaviour, refining the estimate can be likened to searching for a certain behaviour or set of behaviours in the population. A possible way of doing this would be to record a reference behaviour (e.g. the voltage trace in response to a certain current stimulus), then score candidates by how well they

match this behaviour, and iterate with a GA or similar algorithm. But what should the stimulus be? Without knowing in advance what the reference behaviour is, a good stimulus is one, I would argue, that is "contentious" in the candidate model population, i.e., that produces a wide range of behaviours. The wider the spread of responses from the candidate models, the more information we gain from comparing the model responses to the reference. Conversely, of course, a stimulus that elicits a homogeneous response from the candidate population is quite unsuitable to provide any information about how well the candidates represent their reference.

Following this intuition, I propose the following iterated optimisation process. First, generate a number of candidate stimuli, and evaluate them against the candidate model population, keeping track of some outcome measure. Next, calculate for each stimulus the variance of said outcome measure across models, and use the stimulus with the highest variance to gather reference data. Finally, score the candidate models' responses to the same stimulus by the measure used during stimulus evaluation, conduct a selection and procreation round, and advance to the next epoch.

5.2.2 Implementation

I implemented the above approach on top of the software built for MOSTIPS in an effort not to complete a full investigation of the method, but to gain some pilot data as proof of concept. For this reason, and due to time constraints, I implemented a simplified version of the above as follows.

5.2.2.1 Cost function

The cost function λ, likely the most important part of this algorithm, was designed to provide a good measure for trace alignment without falling into the trap of very high cost for only slightly misaligned action potentials. It compares

the evaluated voltage trace $V_e(t)$ to a reference trace $V_{ref}(t)$ at each time point using two component functions λ_1 and λ_2 in root mean squares fashion, i.e.,

$$\lambda = \sqrt{\sum_t \lambda_1(t)^2 + \sum_t \lambda_2(t)^2} \qquad \text{(Equation 5.1)}$$

The first component, applied only outside of spikes, is the squared voltage difference as formalised in Equation 5.2.

$$\lambda_1(t) = \begin{cases} V_e(t) - V_{ref}(t) & \text{if spike} \\ 0 & \text{no spike} \end{cases} \qquad \text{(Equation 5.2)}$$

The second component is a spike density function, calculated iteratively starting from $\lambda_2(0) = 0$, decaying with a factor $D < 1$, and increasing by a set amount R during spikes:

$$\lambda_2(t + \Delta t) = \begin{cases} D\lambda_2(t) + R & \text{if spike} \\ D\lambda_2(t) & \text{no spike} \end{cases} \qquad \text{(Equation 5.3)}$$

The spike detection mechanism is implemented on top of a decaying average δ_V of the discrete voltage derivative of the evaluated trace, i.e.,

$$\delta_V(t + \Delta t) = 0.99\delta_V(t) + V(t + \Delta t) - V(t) \qquad \text{(Equation 5.4)}$$

The spike flag is set when $\delta_V(t)$ rises above a threshold T_δ, and cleared when $\delta_V(t)$ drops below 0.

For the experiments described below, the threshold $T_\delta = 40\ \mathrm{mV/ms}$ was set empirically to detect spikes in the reference trace as early as possible without false positives from recording noise. The spike density decay rate was set to $D = 0.999$, and the increment to $R = 100$, both applied at a sampling interval of 240 µs.

Due to an oversight on my part, while the spike detection mechanism's decaying voltage derivative δ_V was calculated as presented for the experimental reference trace, it was accidentally inverted for simulated traces. That is, for stimulus selection and for candidate traces during model optimisation (see below), Equation

5.4 was implemented as $\delta_V(t+\Delta t) = 0.99\delta_V(t) - V(t+\Delta t) + V(t)$, thus triggering the spike flag during the repolarisation phase of action potentials and resetting shortly thereafter. While largely inconsequential for the spike density function λ_2, the direct trace error λ_1 therefore included action potentials, making the cost function more sensitive to spike misalignment than intended.

5.2.2.2 Stimulus selection

In each epoch, 64 current stimuli were randomly generated, constrained to last 1 second, contain between 1 and 3 steps or ramps of no less than 2 ms duration, with a current starting at zero and limited to ±2 nA. Then, to speed up evaluation, 128 of the 8192 candidate models were randomly[5] chosen to evaluate each stimulus. Then, rather than compare traces all-to-all within these chosen models, I chose to only make pairwise comparisons, with each model comparing its trace to its preceding neighbour, wrapping around in sets of 32 models. This allowed me to make these comparisons and accumulate λ iteratively during simulation, greatly speeding up both my implementation and the evaluation. Then, for each stimulus, the variance of λ across all models evaluating that stimulus (i.e. across all pairwise comparisons between neighbouring models) was calculated. Then, the 2 stimuli with the highest variance were selected for evaluation against the reference neuron, see below.

5.2.2.3 Model optimisation

The candidate model population was optimised with a genetic algorithm as described in section 2.3.2.1, using $n_E = 100$ elites, $n_R = 100$ reinitialised models, and crossover rates Cr between 0 and 0.3. The parameter mutation rates (see section 5.2.3.2 below) were decayed with a half-life of 700 epochs. The cost

5 The choice of subset was effectively random for most models by virtue of the randomised procreation mechanism. This excludes any GA elites (see below), which would have remained within (or without) the subset as long as they retained their elite status. This was deemed acceptable, since on average, only 1-2 elites should fall within the subset.

function was used as described in section 5.2.2.1 above, accumulating the squared error observed during the full 1 second response to each of the two stimuli selected in a given epoch. For validation, a set of four stimuli was used, including single steps of 1 s duration to +1 nA and +2 nA, and ramps over 1 s from 0 nA to +1 nA and +2 nA. Reference responses were collected with this stimulus set every 10 epochs.

The sampling interval for recording and evaluation, both in model optimisation and in stimulus selection, was set to 240 µs, and integration was performed with a Runge-Kutta-Fehlberg 4/5 method with a minimum step size of 12 µs. To control the initial state of candidate models, they were settled with a 1 second null stimulus prior to both stimulus selection and each model optimisation stimulus.

5.2.3 Application

I applied this method to gather pilot data in collaboration with Dr. Rafael Levi, a visiting researcher invited for this purpose and supported by a Research Development Fund grant to Prof. Thomas Nowotny and Prof. George Kemenes. In this section, I will briefly discuss our experiments and their results. We used the pond snail *Lymnaea stagnalis* as a source of neurons, targeting the B1 motor neuron, for which we had a relatively trustworthy model in hand (Vehovszky *et al.*, 2005). In these experiments, Rafael Levi handled dissection and electrophysiology, Thomas Nowotny provided the adjusted B1 model, and Rafael Levi and myself jointly performed stimulation and recording.

5.2.3.1 Methods

Snail brains were dissected in normal (physiological) saline (in mM: NaCl 50, KCl 1.6, $MgCl_2$ 2, $CaCl_2$ 3.5, HEPES 10, pH 7.9 adjusted with NaOH) and fixed onto a Sylgard substrate with fine pins. The outer sheath covering the buccal ganglia was removed with forceps, and since the left and right B1 neurons are electrically coupled, the buccal commissure was crushed to disrupt this connection.

Then, a small amount of protease was topically applied to the buccal ganglia for 1-2 minutes to soften the inner sheath. After washing, the preparation was again submerged in physiological saline and transferred to the electrophysiology setup.

Electrodes were pulled from borosilicate glass capillaries and filled with 5 M potassium acetate, yielding tip resistances of 5-10 MΩ for the current electrode, and 20-25 MΩ for the voltage electrode. B1 neurons were visually identified and impaled with both electrodes. An AxoClamp 2B device was used in current clamp mode to read the voltage and inject current. For data collection, we used the Dell Poweredge computer and National Instruments PCIe-6251 data acquisition card described in section 2.4.1 to perform the closed-loop algorithm, to control stimulation, and to record data during stimulation episodes. In parallel to this, we used a Dell workstation running Windows XP with a Digidata 1320A to passively record all input and output data at 10 kHz in pClamp 8.

5.2.3.2 Model

The published B1 model (Vehovszky *et al.*, 2005), which I used in its original form in chapter 4 (see section 4.2.1.6), did not fit closely enough with data from our own investigations. We therefore adjusted its kinetics to more closely approximate the action potential shape and other characteristics. The model currents are unchanged from the original model, and include a sodium current I_{Na}, a delayed rectifier potassium current I_K, and an A-type potassium current I_A as described in Equation 4.5 on page 80. The gating variables of all currents are formulated by their steady-state and time constants, i.e., they follow the formulation

$$\frac{dx}{dt} = \frac{x_\infty - x}{\tau_x} \qquad \text{(Equation 5.5)}$$

The kinetics equations for all currents were standardised to the form

$$\begin{aligned} x_\infty &= \frac{1}{1+\exp((V_x - V)/s_x)} \\ \tau_x &= \tau_{0x} + \frac{\tau_{Ax}}{1+\exp((\tau_{Vx} - V)/\tau_{sx})} \end{aligned} \qquad \text{(Equation 5.6)}$$

where x is m and h for sodium activation and inactivation, NA and NB for the delayed rectifier components, and a and b for the A-type potassium current. The central parameter values were set as shown in Table 5.1. The equilibrium potential parameters ENa, EK, El, as well as the offset parameters Vx and tVx were treated as additive, with a sigma of 0.25 mV and a range of ±20 mV around the central value. All other parameters were treated as multiplicative, with a sigma of 0.01 and a range of 0.5 to 1.5 times the central value. The leak conductance gl was allowed to increase to up to 1 µS to account for varying experimental conditions. Due to a bug in my code, all negative multiplicative parameters (tsm, sh, tsh, tsNA, tsNB, tsa, sb) were restricted to exactly 1.5 times the central value during optimisation.

parameter	**value**		**parameter**	**value**		**parameter**	**value**	
* gNa	18.48	µS	+ ENa	36.20	mV	* gKA	4.71	µS
* gKB	1.58	µS	* gA	12.36	µS	+ EK	-66.29	mV
* gl	0.0185	µS	+ El	-20.82	mV	* C	1.0	nF
+ Vm	-25.27	mV	* sm	9.37	mV	* t0m	0.346	ms
+ tVm	-37.98	mV	* tsm	-1.99	mV	* tAm	8.465	ms
+ Vh	-29.43	mV	* sh	-3.96	mV	* t0h	2.45	ms
+ tVh	-25.12	mV	* tsh	-3.65	mV	* tAh	16.23	ms
+ VNA	13.99	mV	* sNA	32.54	mV	* t0NA	8.28	ms
+ tVNA	-19.65	mV	* tsNA	-46.33	mV	* tANA	67.68	ms
+ VNB	8.63	mV	* sNB	15.50	mV	* t0NB	1.44	ms
+ tVNB	-19.98	mV	* tsNB	-43.17	mV	* tANB	11.07	ms
+ Va	-12.28	mV	* sa	14.11	mV	* t0a	0.055	ms
+ tVa	-20.60	mV	* tsa	-34.76	mV	* tAa	5.70	ms
+ Vb	71.26	mV	* sb	-7.15	mV	* t0b	4.22	ms
+ tVb	-20.92	mV	* tsb	35.23	mV	* tAb	25.27	ms

Table 5.1: Adjusted B1 model parameters' central values. The prefix "+" or "" indicates additive or multiplicative status of the parameter, respectively.*

5.2.3.3 Results

We conducted a number of experiments exploring the space of possible implementations of the general idea of closed-loop model optimisation, and of possible cost functions, the results of most of which I will not present. In the following, I present results from three B1 cells that reflect the most recent state of our evolving understanding of how to apply this method.

In Figure 5.1, I demonstrate the progress of fitting with snapshots taken every 20 epochs. The plotted models (blue traces) are the best-fit parameter sets of the epoch immediately preceding the recording of the respective validation trace. The models are therefore not selected for an optimal fit against the plotted response, but against algorithmically selected stimuli. Despite this, we see a clear trend for the excitability of the models to track the excitability of the cell, which varies considerably across the approximately 10 minutes elapsing from first to last trace. There is also a slight trend towards closer approximation of the resting membrane potential, particularly in the two fits on the left. Conversely, spike shape does not appear to improve over time, which suggests that capturing spike shape in the cost function may be worth investigating, e.g. using a phase plane approach.

In Figure 5.2, I show a principal components analysis (PCA) of the complete model populations corresponding to the plots in Figure 5.1. Ignoring the 100 freshly randomised models dotted around the middle of each plot, we see that the populations tend to cluster very early in each fit, with successive snapshots showing an ongoing refinement from many to 2-4 clusters. This tendency to not collapse into a single cluster may be due to an interaction with the stimulus selection algorithm. Of course, once the model population is converged to a small number of clusters, stimuli are selected to differentiate between these, rather than between models and the reference. I speculate that this may impede refinement of the model population towards more accurate parameter sets, e.g. if neither cluster fits the reference traces particularly well. My use of two stimuli in

each epoch, rather than only one, may also impact this development, in that it takes twice as many stimuli exposing a consistent error gradient between two clusters for the cluster with higher error to go extinct. That there are several clusters throughout most fits therefore indicates that the true model is either well hidden (e.g. in a narrow global optimum) or does not exist; in either case, I would argue that maintaining several clusters is in fact more informative to us and keeps the algorithm more flexible.

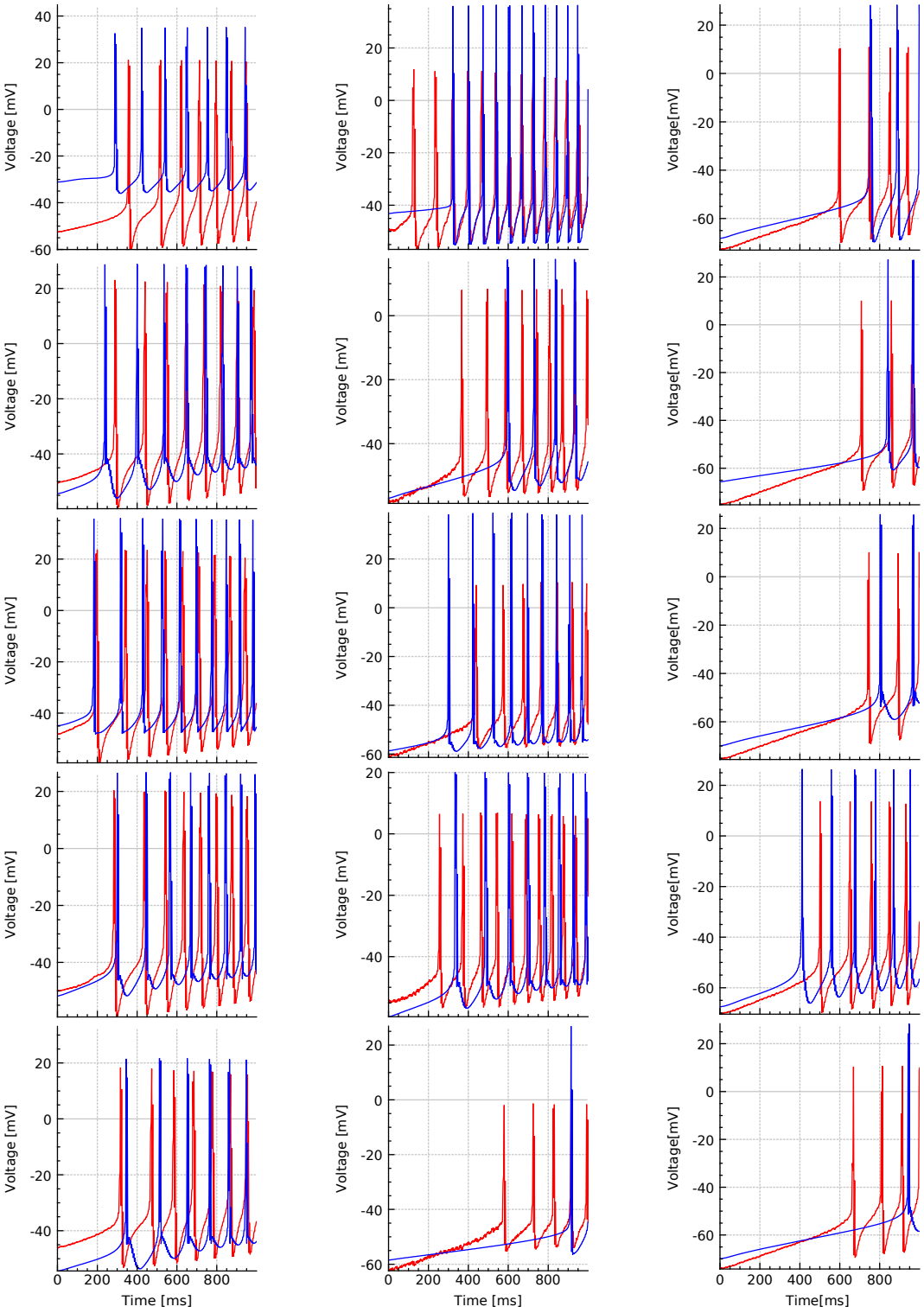


Figure 5.1: Closed-loop fitting progress in three B1 cells, demonstrated with snapshots from epochs 10, 30, 50, 70 and 90 (top to bottom; each column is a separate cell and fit). Shown are the validation traces (red) and the best-fit model (blue) responding to a ramp from 0 to +2 nA. Fits clearly track excitability changes.

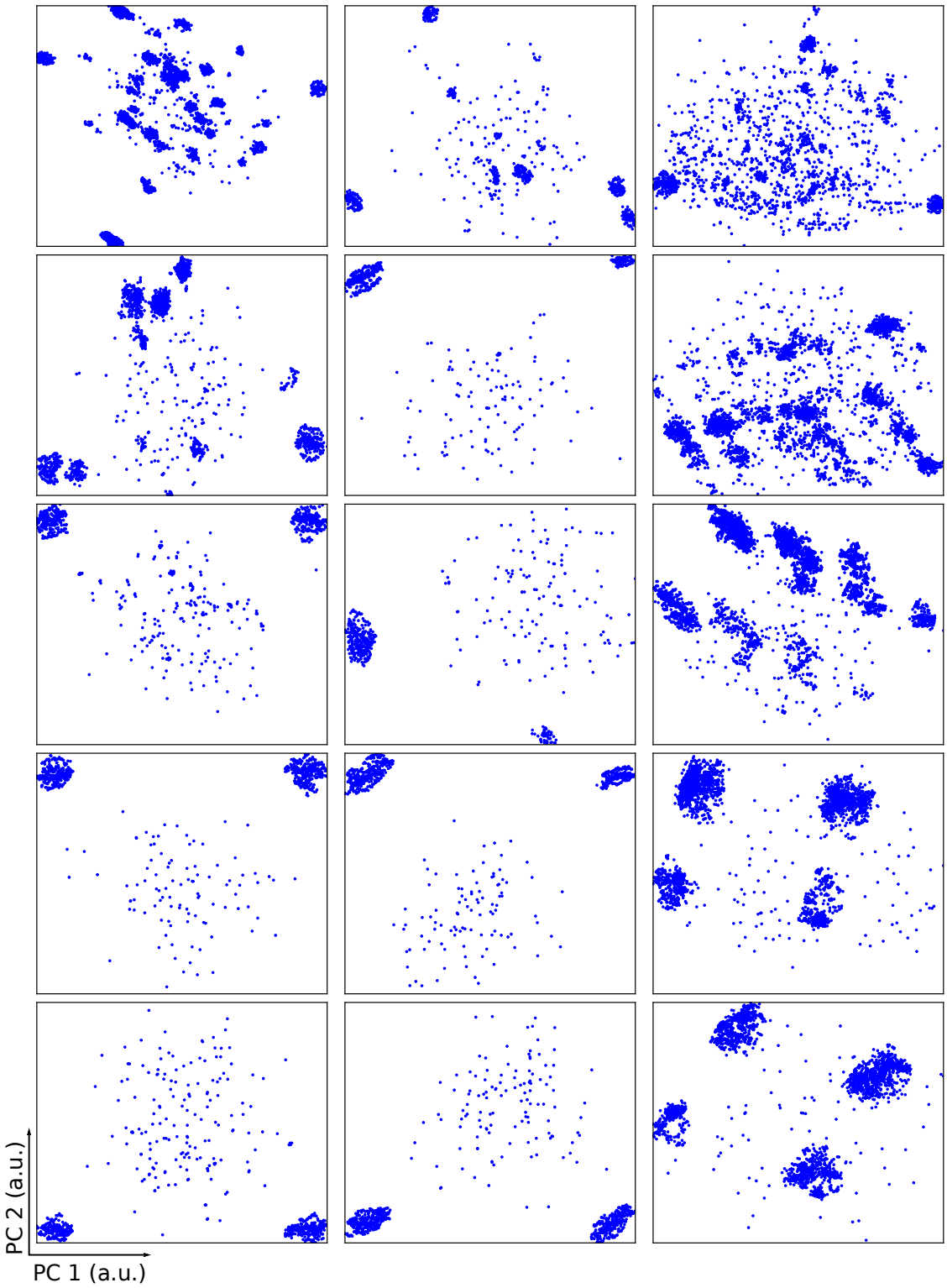


Figure 5.2: PCA of the candidate parameter sets, using the same fits as in Figure 5.1, arranged in the same way. The initially widely dispersed candidate parameter sets appear to quickly converge onto a small number of clusters, though the algorithm seems unable to differentiate between these. Note, the widely dispersed points are randomly initialised models.

5.2.4 Conclusion

Despite these issues, I am confident that the results presented here are indicative of this method's potential. By stimulating neurons in current clamp instead of voltage clamp, the kinds of data we receive are more naturalistic, i.e. more representative of the neuron's typical behaviour. While current clamp stimulation does not offer the opportunity of isolating, and thus precisely estimating, individual parameters, it may be a better tool to model the behaviourally relevant aspects of a neuron's activity patterns. This contrasts with voltage clamp data, which, in addition to the experimental difficulties outlined above, is also a very unnatural manipulation that may affect the cell in unexpected ways, e.g. by activating homoeostatic pathways related to the regulation of the membrane potential or to excitability.

Aside from fixing the bugs mentioned above and perhaps further exploration of the relevant cost functions for both models and stimuli, there are other relatively easy improvements that could be made to the method as described above. For example, I generated the stimuli completely randomly at each epoch, but it might be interesting to co-evolve a population of stimuli alongside the model population, allowing progressive refinements in one population to tighten the focus of the search in the other. Since the fitness goal of the stimuli (high variability in the model responses) is directly opposed to that of the models (close approximation of the reference trace), a co-evolution approach could be made to self-enhance, similar to the idea of generative adversarial networks in the deep learning literature (Goodfellow *et al.*, 2014).

Other possible improvements, such as a more refined fitting algorithm, more varied stimuli, etc. are shared with the voltage clamp MOSTIPS and discussed in the final chapter. Before we get there, however, there is one final aspect to my thesis, namely the continued development of the dynamic clamp software StdpC.

5.3 Extensions to StdpC

Dynamic clamp (Robinson & Kawai, 1993; Sharp *et al.*, 1993; Prinz, Abbott, *et al.*, 2004) is a method of manipulating a neuron's membrane potential based on conductance models e.g. of ion channels or synapses. That is, rather than injecting a current independently of the membrane voltage – as in open-loop current injection – dynamic clamp simulates a conductance, injecting a current that corresponds to the state of that conductance and the membrane voltage at any given time. Since closing the loop in this manner requires a very fast cycle of measurement, model calculation, and current injection, dynamic clamp is typically implemented either in hardware, e.g. (Desai *et al.*, 2017), or in real-time operating systems, e.g. (Linaro *et al.*, 2014; Amaducci *et al.*, 2019), where the implementation has complete control over the timing of the clamp cycle. However, these solutions are technically involved, requiring specialised hardware and/or software that might dissuade less technically oriented experimental neuroscientists from attempting to use them.

StdpC is an open source dynamic clamp software originally developed by Prof. Thomas Nowotny (Nowotny *et al.*, 2006; Kemenes *et al.*, 2011; Samu *et al.*, 2012). Unlike typical dynamic clamp solutions, StdpC is designed to run on Windows, likely the most commonly used operating system among experimental neuroscientists, given that most other electrophysiology software (e.g. Spike, Signal, pClamp) is also Windows-based. The principal difficulty in using Windows, a non-real-time operating system, as a platform for dynamic clamp is that the clamp cycle timing is not under the software's control, and the duty cycle varies considerably as the operating system schedules other tasks to run alongside the dynamic clamp computations. Of course, any processes worth implementing in dynamic clamp are dependent on both the membrane voltage and on time. To calculate its models accurately, StdpC must therefore refer to the system clock rather than a fixed time step for information about the time elapsed since the previous computation, as well as work as fast as possible to minimise

the temporal shift between a current injection command and the voltage reading that it is based on.

StdpC has become a key tool for a number of labs around the world and has been successfully used to perform dynamic clamp experiments in various systems, including in invertebrates such as lobsters (Reyes *et al.*, 2008), the pond snail *Lymnaea* (Kemenes *et al.*, 2011; Samu *et al.*, 2012) and the nudibranchs *Melibe*, *Dendronotus* (Sakurai & Katz, 2016, 2017, 2019) and *Tritonia* (Sakurai *et al.*, 2014), in rodent brain slice preparations, targeting thalamus (Hong *et al.*, 2014; Amarillo *et al.*, 2018), amygdala (Szűcs *et al.*, 2010, 2012; Szűcs & Huerta, 2015; Francesconi *et al.*, 2017), hippocampus (Vaidya & Johnston, 2013; French *et al.*, 2015; Szűcs *et al.*, 2017; Sokolova *et al.*, 2019), and cortex (Huang *et al.*, 2018), as well as in resected human cortical tissue (Szegedi *et al.*, 2019). During my early MOSTIPS work, Thomas Nowotny was approached by Naoki Kogo, who wanted to use StdpC for an ambitious project that required additional features. Having used StdpC in a previous project myself, and already being set up with all the relevant equipment and software development skills, I offered to take the extension in hand. In the following, I will summarise the major features added to the software in the meantime, both for Dr. Kogo and for other collaborators.

Most of the feature development detailed below is entirely my own work. Thomas Nowotny developed the software-defined voltage clamp (section 5.3.6), step generator and wire (section 5.3.7) tools. Software-defined voltage clamp testing was conducted in collaboration with Dr Rafael Levi.

5.3.1 Existing features

StdpC was built as a fairly general-purpose dynamic clamp software with a comprehensive graphical user interface (GUI) with the following features before my extensions. As data sources, StdpC supported DigiData 1200/A devices, National Instruments data acquisition cards, and a “simulated” data acquisition device (labelled SimulDAQ) which used text files with time series data as input and output

channels. Only one data source could be active at a given time. A hard-coded total of six synaptic conductances (including several formulations for chemical synapses, as well as a gap junction model) and six ionic conductances (including both x_∞/τ_x and α/β formulations) could be connected to the input (voltage) and output (current) channels of the data sources. Each of these models provided extensive opportunities for parametric customisation. Lastly, StdpC provided a "spike generator" producing a voltage time series defined by exponentially rising and decaying spikes of a parametrically defined width, amplitude, and timing structure, which could be run in a loop or triggered by threshold crossings on an input channel.

In addition to these core features, StdpC provided a rudimentary graphing interface to visualise the data that the software was reading from and writing to the data source channels, intended mostly for testing purposes, as well as a "data saving" feature that allowed the samples from any channel to be written to a file on disk. StdpC allowed saving a graphically set up configuration to a text file, and of course loading the same. The format of configuration files consisted of human-readable key-value pairs, which – together with a time stamp – could also be used for scripting, where parameters are changed at specific times while the dynamic clamp process is running.

In addition to the flexibility afforded by the highly parametric models and the scripting interface, StdpC was, and is, an open source project written in C++, which means that users are free to change the source code or add their own models as required.

5.3.2 Hybrid networks with complete neuron models

5.3.2.1 Motivation

Hybrid networks, composed of both real and model neurons, were requested by Naoki Kogo. His project aims to examine the neural substrate for bistable per-

ception, as e.g. in binocular rivalry (Levelt, 1965; Laing & Chow, 2002; Noest *et al.*, 2007; Shpiro *et al.*, 2009). To do this, he hoped to form a mutually inhibitory circuit between pairs of real cortical pyramidal neurons, manipulating the properties of the inhibitory interneurons and their synaptic interactions with the pyramidal neurons, the presence and strength of background noise in the circuit (see section 5.3.4 below), and the strength of input to the pyramidal neurons.

5.3.2.2 Implementation

To support this, StdpC needed a way to model not just ionic conductances, but entire neurons. This has been done with the StdpC code base previously, though in an ad-hoc fashion that did not find its way into the released software (Brochini *et al.*, 2011). In the spirit of reusing existing features and minimising redundancy, I decided to exploit the existing ionic conductance models by connecting them to a model of a passive membrane. Conceptually, a membrane model – defined by capacitance and a leak conductance – can be viewed as an input-output channel pair, much like a real membrane accessed with an electrode. By connecting models of active conductances to these channels, we can then build a complete neuron model in a completely modular fashion.

The mathematical implementation of such a model is straightforward, with the update rule for the membrane potential derived from the familiar Equation 5.7, with the "injected" current I_{inj} now produced by the combined conductance models.

$$C\frac{dV}{dt} = I_{inj} - g_l(V - E_l) \qquad \text{(Equation 5.7)}$$

The programmatic implementation, however, was a little more involved. Since in the existing StdpC code, only one data source (data acquisition card or simulation thereof) could be active at a time, the concept of input and output channels was immediately linked to the presently active data source, with an exception in

place for the spike generator outputs. Instead of adding to the exceptions, I therefore decided to rework the way data sources and channels were handled.

First, I modularised the code for the existing data source classes, allowing an arbitrary number of data source objects to be active simultaneously. Next, I reworked the way channels were identified. Previously, to assign conductances to specific channels, addressing was done using an integer index, with -1 specifying the spike generator's output. With data sources no longer defined as a single, uniquely active object, this needed to be replaced with a hierarchical addressing method, specifying the type of data source, its index (since there could be multiple instances of a given type, e.g. two separate data acquisition cards), and the index of the channel within that instance. Finally, I turned the spike generator and the newly created passive membrane model into data source classes of their own.

Since there were several configuration options associated with channels (specifically, flags to denote them as active and to mark them for data saving, data limits, a constant bias term for output, i.e. current, channels, and spike detection settings for input, i.e. voltage, channels), I decided to keep the basic data source parameters (e.g. leak conductance for the membrane model, or the spike timings for the spike generator) separate from the channel settings, allowing each of these new data sources to expose an arbitrary number of channels or channel pairs, each with their own configuration. Finally, the interface to assign channels to conductances was likewise modularised, allowing a given conductance model to be assigned to an arbitrary number of channel combinations.

To allow users to make full use of the flexibility afforded by unlimited data sources, I further lifted the limitation on the number of synaptic and ionic conductance models, turning the previously static list of six models of each type into a dynamically sized list, thereby removing any artificial limitations on the complexity of experiments conducted with StdpC.

With all these changes implemented, users could now define neuron models of arbitrary complexity, within the limitations of the formalisms provided by the conductance models. Since the synapse models include a gap junction model, multi-compartment models could also be built in principle by creating a separate membrane model for each compartment and linking them with gap junctions. In addition, any synapse model could also be used to build synaptic connections amongst model neurons as well as between model neurons and live neurons connected to StdpC via electrodes and data acquisition cards, thus making it possible to build hybrid circuits of both real and modelled neurons.

Upon testing, however, we noticed that model neurons had a tendency to become numerically unstable due to the following problem. The ionic conductance models used a simple forward Euler integration method to calculate their state evolution. In a purely dynamic clamp context, where a conductance model only contributes a small part of the overall membrane current, this is sufficiently precise, with any imprecisions washing out in the noise of the recording and the membrane's intrinsic dynamics. In a membrane model, however, where the entire current is defined by numeric integration, any imprecisions in the integration add up, leading the model to behave in unexpected ways. This problem is exacerbated with StdpC's variable clamp cycle timing and occasional very long time steps as the operating system schedules other tasks alongside the clamp cycle process.

In part, this problem was fixed by enforcing limits on the models' gating variables and the membrane model's voltage, catching invalid values and resetting affected models on the fly, thereby preventing integration from getting stuck in infinities. However, while this prevented complete breakdowns of the models, the fundamental problem of insufficiently precise numeric integration also needed to be addressed to keep models behaving as expected.

To do this, I turned to a fourth-order Runge-Kutta integration scheme, which approximates the model state at time $t + h$ much more precisely than the forward

Euler scheme by progressively estimating the state and resulting state gradient at intermediate time points. Of course, in normal dynamic clamp, the state of the system is largely externally defined, thus Runge-Kutta integration is not possible. Therefore, during the setup phase of the clamp cycle, I separated conductances into three categories, the first including all conductances that take input from analogue data sources (e.g. normal dynamic clamp conductances, or synapse models connecting from analogue input to model neurons), the second including all conductances with no connection to analogue sources (e.g. model neurons' membrane conductances), and the third including any remaining conductances (e.g. synapses connecting from model neurons to analogue output). While the first and third category could be integrated with a forward Euler scheme without producing artefacts, the second category was entered into a Runge-Kutta integration loop, providing robust numeric integration with minimal computational cost.

5.3.2.3 Application

The new feature was successfully applied by Naoki Kogo in experiments. Pyramidal neurons were recorded with patch pipettes in mouse occipital brain slices, ensuring no existing mono- or disynaptic connections. In StdpC, interneuron models were set up using literature-derived models of sodium and delayed rectifier potassium channels (Hodgkin & Huxley, 1952d), as well as Kv3 channels (Lien & Jonas, 2003), using passive membrane properties as described in (Pospischil *et al.*, 2008). Excitatory synaptic connections from the pyramidal cells to the model inhibitory neurons, and inhibitory connections from the model inhibitory neurons to the pyramidal cells, were set up using StdpC's ChemSyn model with parameters hand-tuned to consistently evoke sufficiently large post-synaptic potentials, giving rise to a hybrid four-neuron mutual inhibition circuit as shown in Figure 5.3. For more detail on the investigation of bistability, refer to section 5.3.2.

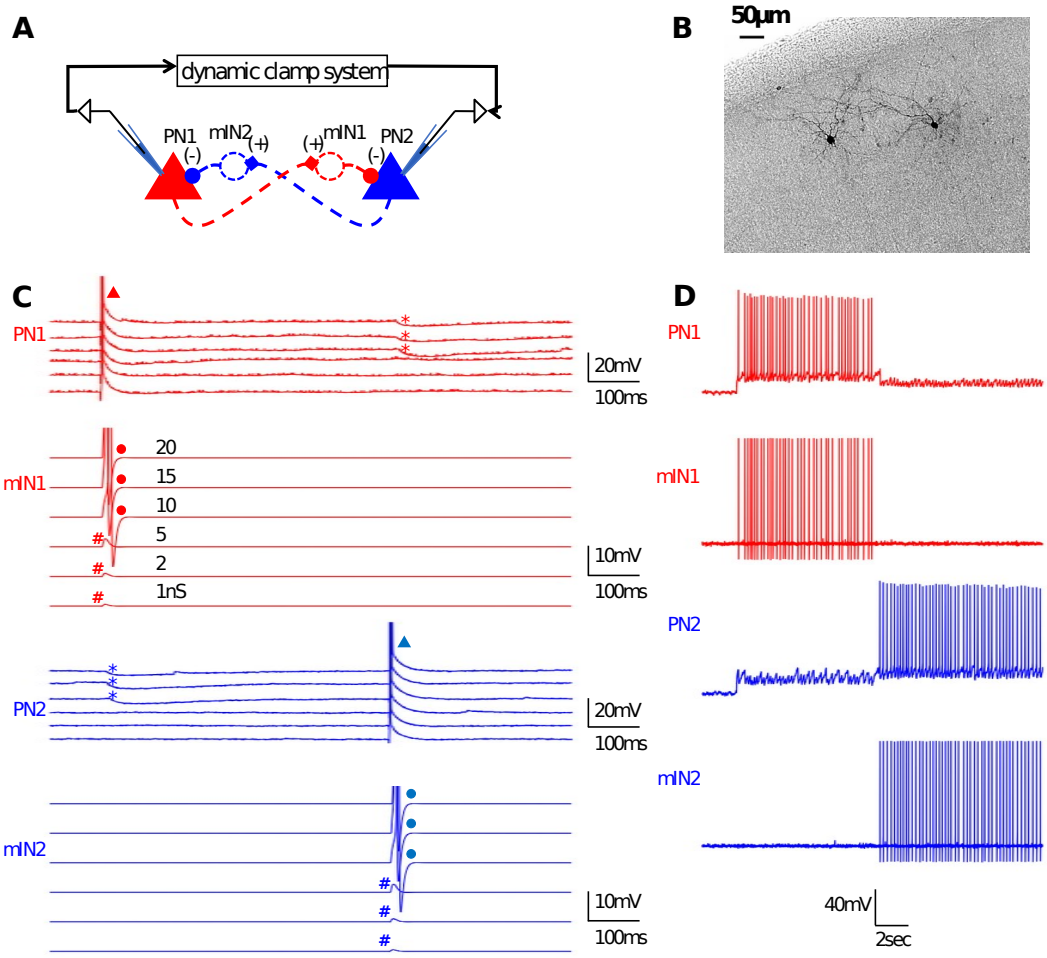


Figure 5.3, adapted from Kogo, Kern et al. *(under review): A hybrid circuit built from live pyramidal neurons (PN) and model inhibitory neurons (mIN). A: Circuit diagram. Dotted features are simulated in StdpC. B: Two biocytin-filled pyramidal neurons in mouse visual cortex, layer 2/3. C: The strength of the excitatory synapse (PN to mIN) had to be adjusted manually to consistently evoke action potentials in the mIN (bullet symbols), giving rise to inhibitory post-synaptic potentials in the other PN (asterisks). D: Mutual inhibition causes bistable spiking, with activity periodically switching between PN1 and PN2.*

5.3.3 Synaptic delay

One of the ingredients of a realistic circuit, hybrid or otherwise, is the existence of temporal conduction delays. StdpC had no such feature, instead calculating synaptic conductances based on the most recent measured (or model-produced) voltage sample. This feature was requested by Naoki Kogo as part of the hybrid

circuit extension, since his proposed circuit structure (see above) is a closed loop and could have been affected by unexpected feedback behaviour if all synapses were instantaneous. Implementation was again complicated by the variable time step of the StdpC clamp cycle. With a fixed time step, a temporal delay could be expressed in terms of number of cycles, making it straightforward to keep a buffer with the required number of (pre-synaptic voltage) samples, retrieving these with a delay to calculate the post-synaptic conductance.

To enable synaptic delays with StdpC's variable time step, I set up a buffer for voltage samples, sized generously to accommodate acquisition rates up to 100 kHz, and in parallel to that, a buffer keeping track of the time at which each sample was collected. Delayed samples could then be retrieved from the sample buffer by first looking up the time point closest to the requested time, then using the index of that point in the time buffer to locate the corresponding voltage sample.

To make this lookup process as fast as possible, I applied a number of tricks and optimisations. Firstly, each buffer was implemented as a ring buffer, i.e. a contiguous memory area with a pointer to the most recent sample, advancing with each sample and looping back to the start as it reaches the end of the memory area. Secondly, since data acquisition is synchronous (i.e., every channel produces exactly one sample in each cycle), I used a single time buffer for all delayed synapses, sized to the longest requested delay. Likewise, any channel requiring delayed samples is set up with just one buffer. Then, each delayed synapse is assigned two variables, one containing its offset into the time buffer, the other the exact delay requested. At the start of each clamp cycle, all offsets are advanced as necessary to point to the time point closest to the requested time, and these updated offsets are then used during the conductance calculation to retrieve the appropriate voltage samples.

The synaptic delay functionality has been used in the work presented in section 5.3.2 above. Since the feature is already contained in the most recent public re-

lease of the software (StdpC 2017), it appears to also have been used in unrelated research investigating microcircuits in human cortical interneurons (Szegedi *et al.*, 2019).

5.3.4 Synaptic background noise

5.3.4.1 Motivation

As part of his project to investigate the neural basis of bistable perception (see section 5.3.2), Naoki Kogo needed to differentially stimulate the pyramidal neurons in the circuit to reveal how such stimulation – analogous to e.g. the level of contrast of two competing images – affects the alternation rate, the relative duration of activity states, etc. One way to do this would be to inject a constant current into the cells. However, it has been argued in related work (Brascamp *et al.*, 2006; Kim *et al.*, 2006; Moreno-Bote *et al.*, 2007; Huguet *et al.*, 2014; Pisarchik *et al.*, 2014; Baker & Richard, 2019) that noise or stochasticity is a key component of the dynamics of this system. Additionally, pyramidal neurons in a slice preparation are already deprived of input from their many presynaptic partners. Therefore, we decided to inject, and systematically manipulate, a stochastic current that mimics the synaptic bombardment a pyramidal cell might receive in vivo.

5.3.4.2 Implementation

Mathematically, we followed the model proposed by (Destexhe *et al.*, 2001), which recreates the statistical properties of synaptic background noise with a decaying Ornstein-Uhlenbeck process. An individual (inhibitory or excitatory) synaptic background noise current with a reversal potential V_{rev} is calculated in a discrete time environment as described in Equation 5.8.

$$\begin{aligned} I &= g(t)(V_{rev} - V) \\ g(0) &= \overline{g} \\ g(t + \Delta t) &= \overline{g} + (g(t) - \overline{g})e^{-\Delta t/\tau} + R \end{aligned} \qquad \text{(Equation 5.8)}$$

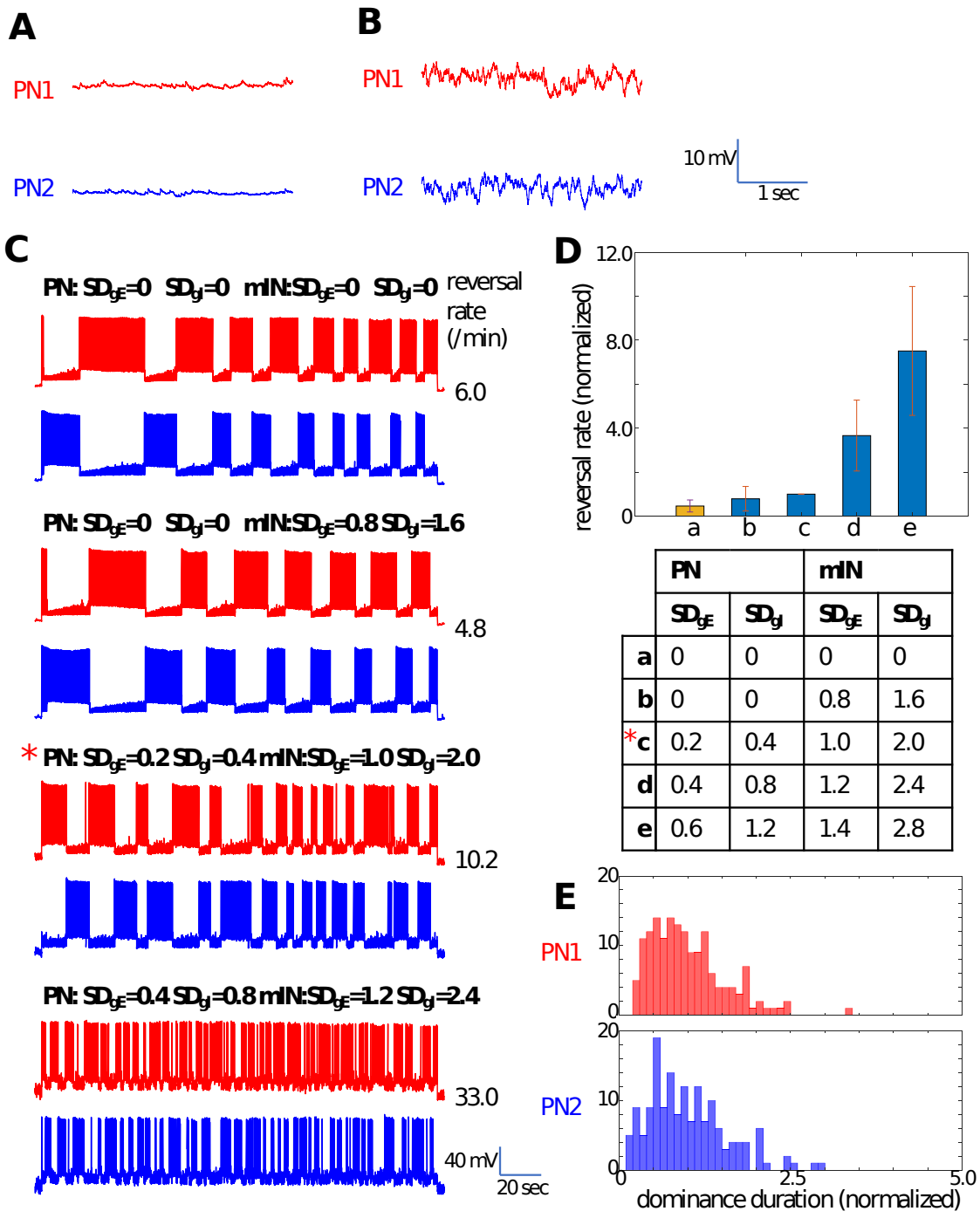
Here, $\bar{g}$ is the mean conductance around which the numeric value fluctuates, τ is the decay time constant, and R is a random variable drawn from a zero-mean, unit standard deviation normal distribution and scaled as shown in Equation 5.9, using a noise diffusion coefficient $D = \frac{2\sigma^2}{\tau}$, where σ^2 is the variance of the noise.

$$R = \sqrt{\frac{D\tau}{2}(1 - e^{-2\Delta t/\tau})}\ N(0,1) \qquad \text{(Equation 5.9)}$$

In StdpC, synaptic background noise is implemented in the same manner as an ionic current, reading voltage from an input channel and adding its resulting current into its assigned output channel. When a conductance of this type is injected into a model membrane and therefore integrated with a Runge-Kutta scheme, random samples are drawn only for the end of a complete step, with the intermediate estimates linearly interpolated. This both reduces the load of the pseudo-random number generator and keeps the stochastic nature of the conductance from interfering with the deterministic estimation process of other current models.

5.3.4.3 Application

The synaptic background noise model was employed in the set of experiments described in section 5.3.2, injecting in vivo-like synaptic bombardment into both the pyramidal neurons and the interneuron models. The noise standard deviation was varied systematically, and the duration and reversal rate of dominance periods, defined as periods during which one of the pyramidal neurons is exclusively active, were analysed, see Figure 5.4. The results demonstrate that the simulated noise produces a realistic pattern in the membrane potential of both real and model neurons (A,B), and that the dominance durations (E) follow a similarly skewed distribution as seen in behavioural data in bistable perception.



	PN		mIN	
	SD_{gE}	SD_{gI}	SD_{gE}	SD_{gI}
a	0	0	0	0
b	0	0	0.8	1.6
***c**	0.2	0.4	1.0	2.0
d	0.4	0.8	1.2	2.4
e	0.6	1.2	1.4	2.8

Figure 5.4, from Kogo, Kern et al. *(under review): Model excitatory and inhibitory synaptic noise (random changes of excitatory and inhibitory conductance, gE and gI, respectively) was applied to the pyramidal neurons and the inhibitory neurons through the dynamic clamp system. A-B: Baseline membrane potentials at -60mV without (A) and with (B) the model noise. C: Effect of changing the noise level systematically. Increase of the noise resulted in increase of reversal rate (from top to bottom). Noise levels are indicated as standard deviations (SD) of gE and gI (in nS). Asterisk: Data with the "standard" parameter set.*

Figure 5.4 (continued): D: Pooled data of the effect of noise to the reversal rates from 15 pairs. The reversal rates from the individual pair are normalized by the value at the standard noise parameters (c) before pooling. Orange bar (a) indicates the data with no model noise. The noise parameter sets for a, b, c (standard noise parameters), d and e are shown in the table below. The noise level is increased linearly from b to e. E: Histogram of dominance durations for PN1 and PN2 from 10 minutes continuous recording (with the "standard" noise parameters).

5.3.5 Synaptic stochasticity

For the next step in his project, Naoki Kogo aims to increase the realism of the synapses in the hybrid circuit. In particular, StdpC's synapse models are entirely deterministic, always evoking a post-synaptic conductance (PSC) of a given size for a pre-synaptic spike. While useful for predictable simulations, this contrasts with the situation in real synapses, which release their neurotransmitters in a probabilistic and quantal fashion (Katz & Miledi, 1965; Barrett & Stevens, 1972; Bekkers *et al.*, 1990; Redman, 1990; Stevens, 1993). A more realistic synapse model should therefore exhibit stochastic PSCs.

Of the three synapse models in StdpC, two (the "Destexhe" and "Alpha-Beta" models) are integrated continuously as a function of the presynaptic membrane potential; in contrast, the "Chemical" synapse model's PSCs are functionally independent of presynaptic voltage below a certain threshold. Its post-synaptic current I_{syn} is modelled as shown in Equation 5.10 and 5.11, where V_{pre} and V_{post} are pre- and post-synaptic membrane voltage, respectively, and V_{syn}, g_{syn}, τ_{syn}, V_{Thresh} and V_{Slope} are user-defined parameters.

$$I_{syn} = g_{syn} S(t)(V_{syn} - V_{post})$$
$$\frac{dS}{dt} = \frac{S_\infty - S}{(1 - S_\infty)\tau_{syn}} \qquad \text{(Equation 5.10)}$$

$$S_\infty(V_{\text{pre}}) = \begin{cases} \tanh\left[\frac{V_{pre} - V_{\text{Thresh}}}{V_{\text{Slope}}}\right] & \text{if } V_{\text{pre}} > V_{\text{Thresh}} \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Equation 5.11)}$$

Because of the explicit threshold in this model, it is possible to efficiently add stochasticity to the PSC size upon threshold crossings, as follows.

Following the analysis in (Redman, 1990), I added a set of parameters to the model to control the number of independent neurotransmitter release sites n_{rel}, the release probability per site, p_{rel}, as well as the standard deviation σ_q of the quantal amplitude, that is, of the PSC size of a single release event. When a PSC is evoked (i.e., the presynaptic voltage exceeds the threshold value), the number of release events is drawn from a binomial distribution as $n \sim B(n_{rel}, p_{rel})$. If any release occurs, i.e. $n > 0$, then an amplitude modifier $q \sim N(n, \frac{n\sigma_q}{n_{rel}p_{rel}})$ is drawn from a normal distribution, with negative values discarded. This modifier is then multiplied in with the dynamic PSC amplitude variable $S \in [0, 1]$ to yield an appropriately scaled rising and decaying post-synaptic potential.

Since there is no guarantee that a given PSC is fully decayed before the next one starts, the implementation in code was a little more complex. Without accounting for overlapping PSCs, the resulting current would immediately jump to a new value based on the latest value of q every time the presynaptic potential crosses the threshold. To prevent this, the PSC amplitude S, rather than being tracked with a single variable, is separated into a list of S_i with corresponding stochastic factors q_i for every threshold crossing event i. Only the most recent S_i is updated with S_∞ evolving according to Equation 5.11, while previous entries are decayed with $S_\infty = 0$ regardless of the presynaptic voltage, and removed from the list once they are sufficiently small. The total PSC is then calculated with $S = \sum_i S_i q_i$. Thus, each PSC is entirely independent from other events and scaled by its private stochastic amplitude.

5.3.6 Software-defined voltage clamp

5.3.6.1 Motivation

Voltage clamp (VC) is usually considered a hardware-based method, implemented with feedback amplifiers and supporting electronic circuitry. The obvious advantage of the hardware implementation is its continuous function and poten-

tially very high gain. However, hardware-defined VC produces artefacts both at voltage steps, in the form of current spikes and – hopefully damped – oscillations after the step onset, and at steady state, in the form of a constant voltage offset proportional to the injected current. Thus, in an attempt to provide a tool without these limitations, Thomas Nowotny and I implemented a software-defined VC in StdpC.

5.3.6.2 Implementation

At its heart, the control principle of hardware-defined VC is a simple proportional gain applied to the difference between command and membrane voltage; the amplified voltage $V_o = A(V_{cmd} - V_m)$ is applied to the injecting electrode, but split between the "access" or electrode resistance R_a and the membrane, i.e., $V_o = V_m + R_a I_{inj}$ (Halliwell *et al.*, 1994). Substituting V_o and rearranging, we see that $V_m = \frac{AV_{cmd} - R_a I_{inj}}{A+1}$, which, at the high gain values typical of hardware-defined VC, is approximately $\frac{R_a I_{inj}}{A}$ offset from V_{cmd}. Implementing VC in software, we could augment the proportional gain of hardware-defined VC with an integral component to better control the achieved membrane voltage and bring it closer to the intended value V_{cmd}. A software implementation also provides an opportunity to add a differential component for a complete PID control setup.

Presently, our implementation is a very simple, user-adjusted system, calculating a total clamp current from proportional, integral, and differential components independently. The proportional component I_p is computed based on the user-defined proportional gain g_p by Equation 5.12.

$$I_p = g_p(V_{cmd} - V_m) \qquad \text{(Equation 5.12)}$$

For the integral component, we accumulate running averages of the membrane and command voltage $V_{x,avg}$, decaying with a user-defined constant $d < 1$, such that $V_{x,avg}(t + \Delta t) = V_{x,avg}(t)d + V_x$. The component's current is then defined with a user-defined gain g_i as shown in Equation 5.13.

$$I_i = g_i(V_{cmd,avg} - V_{m,avg}) \qquad \text{(Equation 5.13)}$$

Finally, for the differential component, we keep a record of the last n values of command and membrane potential, approximating dV_{cmd}/dt and dV_m/dt with their discretised versions based on the difference between the current values and the values n clamp cycles earlier. With user-defined n and g_d, the component is then calculated as

$$I_d = g_d(\frac{dV_{cmd}}{dt} - \frac{dV_m}{dt}) \qquad \text{(Equation 5.14)}$$

I note that both the decay of the integral component's running average as well as the approximation of the differentials is, of course, dependent on the size of the actual time steps, and thus perhaps less than perfect in reality.

5.3.6.3 Application

In testing this feature, we encountered several difficulties. Firstly, tuning the control parameters is no trivial task. PID tuning (Zhuang & Atherton, 1993; Cominos & Munro, 2002; Ang *et al.*, 2005; Lennartson & Kristiansson, 2009) is a lively research field with many different strategies for automatic tuning under current investigation, e.g. (Wang, 2017; Freire *et al.*, 2018; Pinheiro de Moura *et al.*, 2019; Qi *et al.*, 2019). In addition to being a difficult problem, PID tuning is made more cumbersome still by StdpC's strict separation of the graphical user interface, where parameter adjustments are made, and the clamp cycle process; for changes to be applied, the clamp cycle has to be restarted.

Secondly, much like hardware-defined VC, software-defined VC has the potential to "ring", or cause uncontrolled oscillations that cause rapid cell death. This was in part controlled by a fade-in period during which the total PID current is multiplied with a factor ramping slowly from 0 to 1. Ringing, detected as a total current amplitude above a user-defined threshold, automatically deactivates the voltage clamp module, thereby preserving the integrity of the neuron under experimentation.

Thirdly, there are hard limits to the total current we can produce with common digitiser and amplifier equipment. In particular, there is both a limit to the voltage output of a digitiser or digital-to-analog signal converter (National Instruments boards, for example, are typically limited to ±10 V analog output), and to the input of typical amplifiers, enforced by their specification and electronics. This is in contrast to the very high transient currents amplifiers can produce in voltage clamp mode. Thus, even with well-tuned parameters, the step response of a software-defined VC with PID may be less faithful than that of a proportional-only VC amplifier, and depending on the size and morphology of the neuron under investigation, controlling the entire membrane (known as “achieving space clamp”) may be significantly more challenging, if not impossible.

Finally, since StdpC’s sampling step size varies, aligning data e.g. from several voltage steps can be tricky, and there is some jitter to the timing e.g. of pre-pulses. Though not prohibitive, these difficulties add to the cumbersome nature of our software-defined VC.

5.3.7 Additional tools

A number of other tools were added during development of the features detailed above, presented here in no particular order.

To augment the software-defined voltage clamp module, we added a simple step generator tool that outputs a command voltage time series according to a parametric sequence of single steps away from a holding potential.

Building on the modularised data source interface, we added a “Wire” module, which copies data from an input channel to an output channel. The particular use case we had in mind was to inject a predefined current into a cell by reading it from a file to the SimulDAQ module, but the generic implementation certainly allows other creative uses.

In a similar vein, I added an “input conductance” tool, which interprets data read from an input channel as a conductance value, and applies this conductance to an input/output channel pair. This allows e.g. an arbitrary conductance time series to be fed into the SimulDAQ module and replayed onto a patched membrane. This feature was requested by Dr. Attila Szűcs, who is currently using it to replay pre-recorded post-synaptic conductances in a dynamic clamp setup.

In support of the hybrid network extension, I added a preparatory phase to the dynamic clamp cycle. The state of model neurons is not well defined at initialisation, since there are no parameters to set e.g. the gating state of channel models. Instead, state variables are generally initialised to 0 and allowed to evolve from there. While this is unproblematic in standard dynamic clamp, model neurons may take a while to settle into a sensible resting state. To allow this settling to occur without interfering with the rest of the system (i.e. other models, or live neurons), I augmented the run controls such that, rather than starting the clamp cycle with all models active, users can choose to enter a “settling” mode, during which only selected models are being integrated. The transition from settling mode to the full clamp cycle is triggered either by a timer or by a direct command from the user.

Although StdpC has features to record data, many users choose to employ a second computer for record keeping in an effort to maximise clamp cycle performance. Starting a recording therefore requires user input on two separate machines. To allow users to consolidate their input, I added a simple “trigger” functionality which, when primed, waits for a signal on a digital input channel to enter the clamp cycle, or to move from settling mode to full clamp cycle mode.

Finally, in the spirit of liberating all modules from their hard-coded restrictions on number, simultaneous activity, etc., I extended the spike generator tool. Originally, this tool produced a set of 10 user-defined spike times, repeated (e.g. triggered by a threshold crossing on a given input channel) without variation. I replaced this with an arbitrary number of “bursts”, or lists of spike times of arbit-

rary length. Each triggering event now iterates through one burst, with successive trigger or repetition cues cycling through the list of bursts. Additionally, there is an interface to read and/or write spike timings from and to separate files, allowing programmatic generation of arbitrarily complex burst sequences.

5.3.8 Interface improvements

To support all of these new features, the graphical user interface of the software also received a large overhaul. The main control panel (see Figure 5.5), which had previously housed two rows of six model access panels (synapses and ionic conductances, respectively), was extended to four scrolling, collapsible rows, containing access panels for data sources, synaptic conductances, ionic conductances, and miscellaneous tools. The model configuration dialogs, for the most part, were left largely unchanged, with only minor additions as necessary to support added features such as synaptic stochasticity, delay, or multiple channel assignments. To aid navigation and readability of the interface, modules support user-defined labelling, in addition to the default type-and-index labelling scheme.

Perhaps the most striking change to the interface is the addition of a full-blown graph tool in a separate tab, see Figure 5.6. The previous graphing interface was implemented as a basic line-drawing with no interactive capability. While sufficient, if cumbersome, for basic sanity checks of input and output data, this tool was clearly due an upgrade once model neurons were implemented. For this, I harnessed the QCustomPlot library, a tool that supports interactive data display with zooming, panning, plotting on multiple axes, and other nifty features. The graph tab makes generous use of a wide range of these features, providing scrolling time series plots of an arbitrary number of relevant variables. Several plots with synchronised time axes can be stacked vertically, or multiple time series plotted on the same axis. Plottable values include all input and output channels, as well as the conductance value of any of the conductance models.

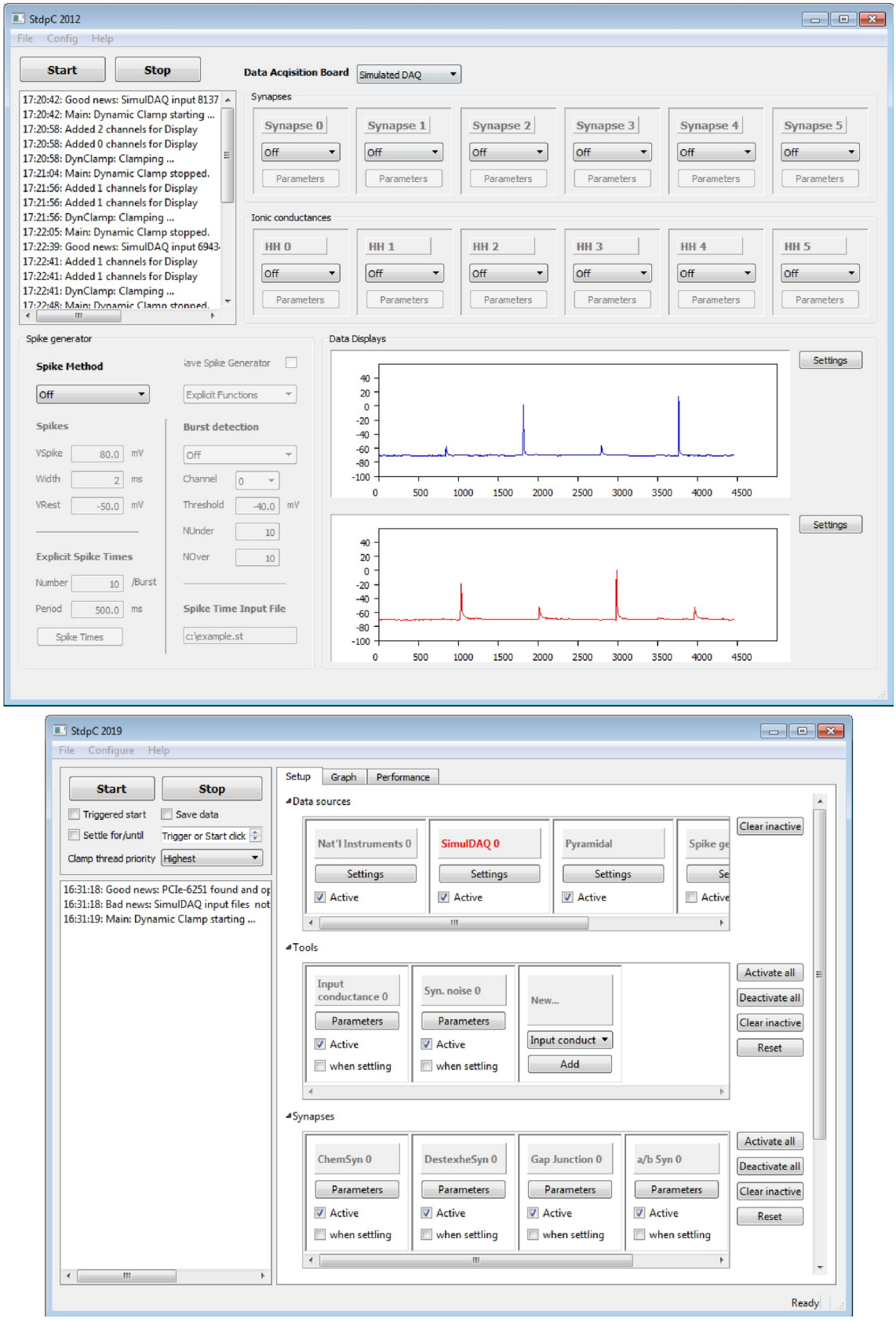


Figure 5.5: StdpC interface comparison, old version (top) and new version (bottom).

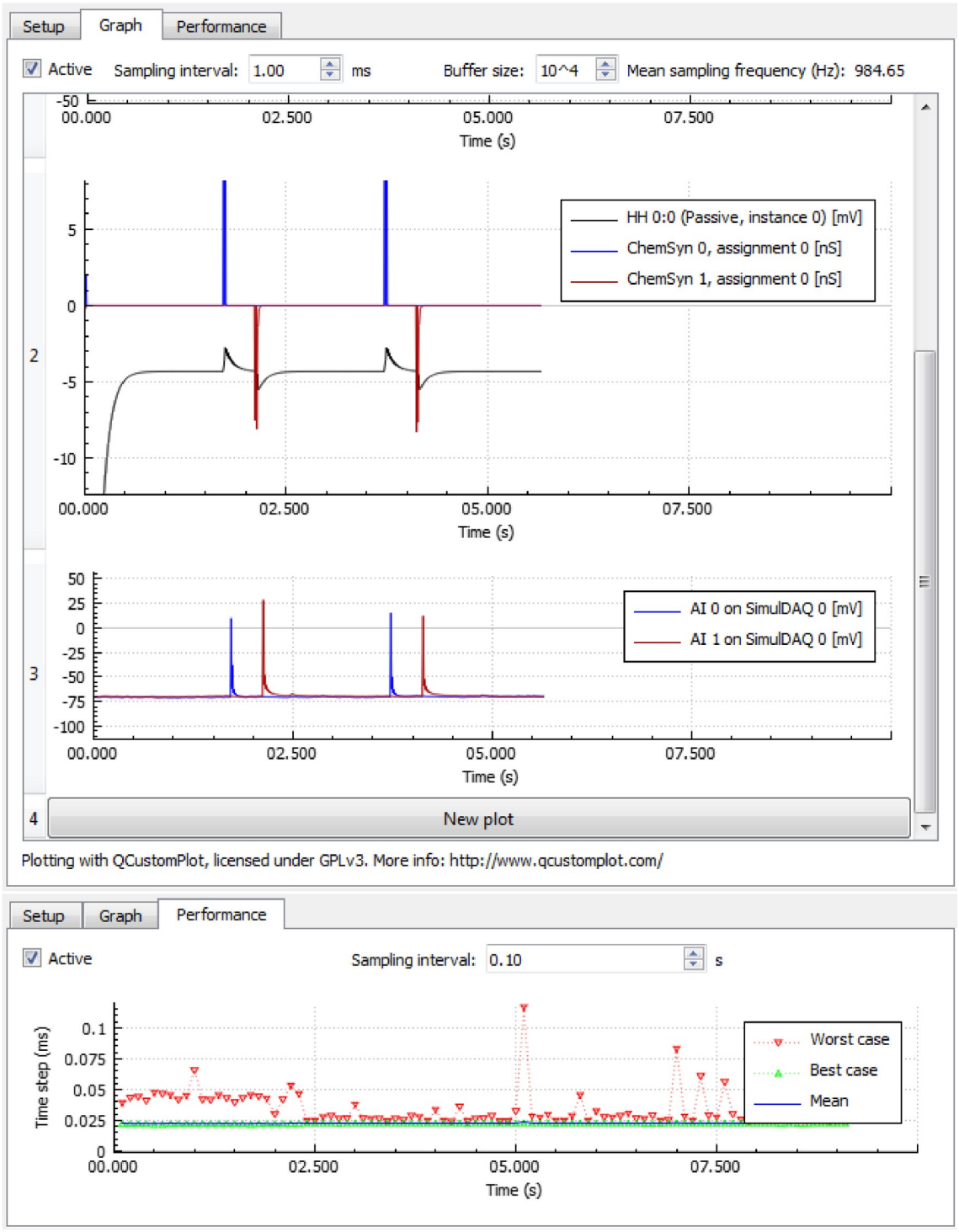


Figure 5.6: StdpC graph interface (top) and performance monitor (bottom). All axes are intuitively controllable by mouse actions. The time axes of all plots are linked, moving and rescaling together. During active use, data is displayed immediately, scrolling along as needed. The graph interface allows an arbitrary number of graphs per plot, with colour, unit and scaling options, and an arbitrary number of plots. Other mouse-based interactions such as adjusting the height of each plot or temporarily hiding graphs by clicking their legend entry are also supported.

The only major limitation of the graph interface is that plotted variables have to be selected before starting the clamp cycle. The reason for this is a performance consideration: Plotting, being a user interface task, is not performed in the clamp cycle thread. Thus, to safely display the values of variables owned and manipulated by the clamp cycle thread, these values have to be explicitly (and asynchronously) transferred to the plotting functions. Although not computationally heavy, this transmission could have a performance impact on the clamp cycle, and is therefore limited to the initially selected variables, and to a user-defined sampling frequency. Graphing can, of course, be turned off entirely, effectively removing any performance impact this feature might have.

Next, I added three tools related to performance itself. The first is a performance indicator in the main window's status bar, updating once per second with the latest clamp cycle frequency, calculated as the number of cycles over the past second. This tool is computationally very light, involving only a counter and a single data transmission per second in the clamp cycle thread, and is therefore always on. A second, more detailed performance monitoring tool transmits the counter value at a user-defined sampling frequency, keeping track also of minimum and maximum cycle period. These values can be displayed as a graph in a separate tab, see Figure 5.6, and are not collected when the feature is turned off. It may aid users in troubleshooting performance issues. Thirdly, users are given the option of changing the priority of the clamp cycle thread, that is, of asking the operating system to give a certain precedence to its operations over other tasks. However, in my experience on a high-end computer (with 16 CPU cores running at 3 GHz, and 32 GB of RAM), the primary limitation to clamp cycle frequency is the data acquisition rate, that is, the time required by the National Instruments or Digidata board to return the requested samples. Even with complex models running, I typically see clamp cycle frequencies of around 20 kHz with analog data acquisition, and 40-50 kHz without.

Finally, the data output feature, which saves samples from any selected input or output channel to a file at a user-defined frequency, was made much more flexible. It now allows data to be written as text or binary output, with the output format specified by the user. In binary format, each channel is written to a separate file, with an additional JSON (javascript object notation) file annotating the dataset with metadata such as units, data format, etc. In addition, file naming is dynamic, offering the capability to augment file names with time, date, or a consecutive index.

5.3.9 Outlook

StdpC has come a long way, with many of my improvements already seeing good use in active research, including independent projects by Naoki Kogo and Attila Szűcs (personal communication). There are, however, a few goals I would like to achieve in the short or medium term. Firstly, the most recent public release of the software is some two years old and includes only a part of the features detailed above. The main reason for not having released the rest of the features in a new version is that I have not found the time to adequately document the new features. StdpC includes a manual for non-technical users that should be up to date with any official release. I hope to get around to this once this thesis is submitted.

Next, in the course of making the module selection flexible, I have disentangled most of the module-specific code from the generic parts of the software. In principle, this could eventually lead to a plugin architecture, where modules are supplied as dynamically loaded library files rather than being hard-coded into the main executable. This would open up the possibility for technically advanced users to write, use, and share their own modules without needing to alter the main body of the software. Though new territory for me, I believe that only a few additions would need to be made to the code to support this.

Then, we have had repeated requests to support newer Digidata boards, which are not uncommon on electrophysiology setups. We have, in fact, attempted to write drivers for the Digidata 1300 and 1400 series, but have not been able to make satisfactory progress for several reasons. On the one hand, we only have access to an almost 20 year old 1300-series board in the lab, so our ability to test our drivers is limited. On the other hand, documentation of the driver programming interface (API) is very sparse, effectively being limited to undocumented so-called “test bed” code, written for a specific (and long obsolete) compiler tool chain. Having failed personally and seen a computer science MSc student fail to fully understand and use the API, I consider the future of Digidata support uncertain.

Finally, as the closed-loop model fitting work matures, I would very much like to integrate its functionality with StdpC. To date, the two projects have been entirely separate, one running exclusively under Windows, the other exclusively under Linux. Merging them, or possibly interfacing between the two projects at runtime, could e.g. allow fitted models to be used immediately in a hybrid circuit. To get there from the current state of the two pieces of software is not entirely trivial, as both rely in part on specific operating system functionality; however, it is certainly a possibility in the medium term future.

6 Discussion

In this thesis, I have presented two new approaches to neuron model optimisation, as well as a number of technical innovations in the area of dynamic clamp.

The majority of the focus of my work, and of this thesis, has been on the development of MOSTIPS, a model optimisation method built around parameter separation using specific stimulus patterns. At its core, this approach is nothing new: Manipulating the membrane under investigation and making targeted observations to isolate the contribution of individual parameters from the system's overall behaviour has been the essence of model building and optimisation routines from the very beginning of conductance-based modelling with Hodgkin and Huxley (1952a-d).

The fundamental innovations of the approach presented here are twofold: Firstly, the manipulation of the membrane is limited to electrical means, eschewing the chemical or pharmacological means – channel blockers, ion replacements, etc. – used in other methods. While this limits the separability of parameters, it prevents the problem of having to use several, and potentially not identical, cells to gather a complete data set, and is thus a critical ingredient to building models of individual cells. In contrast to the one-channel-at-a-time approach followed by Milescu et al. (2008), which could be considered a single-cell analogue to the classical channel blocker-based parameter separation approach, MOSTIPS does not perturb any other cells e.g. in a linked circuit, and leaves the reference neuron intact for use after successful model optimisation.

Secondly, the particular form of the manipulation used, i.e. the set of voltage clamp stimuli, is not designed by hand based on the experimenter's knowledge of the system. Instead, using the algorithms described in chapter 2, stimuli are derived from the structure and expected parameters of the model itself. This is necessary in order to regain the parameter separation that was lost with the re-

moval of pharmacological manipulations. Automating this process further makes it possible in principle to apply this method to intractably complex models that resist an intuitive understanding.

Algorithmic stimulus generation, targeted at achieving particular effects in particular models in service to parameter optimisation, has to my knowledge not been attempted before. It is not entirely clear what the role of the generated stimuli is in the success of the method – that is, whether parameter separation is achieved as intended due to the particular, parameter-specific form of the stimuli, or whether fitting success is mainly due to the wide range of model states traversed by the stimuli, as the control results shown in chapter 4 suggest. However, the results reported in this thesis suggest that the general idea of optimising stimuli to support parameter optimisation is a fruitful avenue of research that should be further pursued.

6.1 Delineating the method's applicability

The primary goal of the method was to be able to optimise models to particular expressions of a given a system, e.g. to individual neurons. To show that this is possible, I have used two types of model systems, harnessing both computer simulations of a variety of accurately controllable models and ectopic expression of two previously characterised potassium channels in *Xenopus* oocytes to produce reference data. While the former showed promising outcomes, the latter proved more difficult than expected, as the characterisation in literature did not match the expressed currents satisfactorily. While channel modification effects – through auxiliary subunits, primary subunit heteromerisation, or other pharmacological interactions – are not unknown, they are often not taken into account in neuron modelling. Viewed at a high enough resolution, however, these kinds of effects are likely to have a detrimental impact on the goal of neuron-specific modelling.

While I consider the model of a neuron type to be too low-resolution a picture, unable to capture the necessary detail for some applications as laid out in the introduction, the existence of variety at the level of channel protein complexes that affects macroscopic current properties such as their kinetics raises the question what the appropriate level of detail should be. Given the number of ion channel subtypes and the plethora of regulation mechanisms present in neurons, there are so many possible combinations that the mapping between ion channels and model currents must necessarily be an approximation. When optimising models to actual neurons e.g. to investigate individual variability or non-synaptic plasticity, I expect the appropriate level of detail to be similar to what I have used, that is, close to the level of detail used to describe neuron types. Using this level of specificity helps both to maintain continuity with existing models, and to prevent overfitting to irrelevant details.

6.2 K^+ channels have two components in oocytes

For the purposes of testing the MOSTIPS method, I had chosen two potassium channel constructs that, to the best of my literature search, had been reasonably well characterised. As I have detailed in chapter 3, however, a close investigation of the evoked voltage clamp currents showed a clear separation into two components in both Kv2.1 and Kv1.4. While a more detailed investigation of this effect exists for Kv2.1 in *Xenopus* oocytes (McCrossan *et al.*, 2009), I am aware neither of an equivalent report in Kv1.4, nor of a complete model of either channel separated into these components. Thus, while not a central part of my work, I do consider the development of the two-component models a significant contribution worth highlighting.

Although I have not investigated this further, I consider it possible that in oocytes simultaneously expressing both Kv2.1 and Kv1.4, a yet more varied picture may emerge upon closer inspection, for example due to heteromerisation of subunits or lateral interactions between channels. I chose to avoid this level of complexity,

however, as it would have made interpreting the results more difficult and potentially raised issues of overfitting.

6.3 MOSTIPS is competitive in fitting predictive models

At this resolution, I have shown that the MOSTIPS method is capable of producing optimised models that are no less predictive of behaviour not seen during optimisation than those optimised by a more common fitting approach. Encouragingly, the most difficult fit, optimising not just conductances and equilibrium potentials, but also kinetic parameters, showed the clearest advantage of the new method over the old, indicating that MOSTIPS is capable of successfully navigating a complex high-dimensional parameter space. That it is able to do so seems to be due to a separation of parameters, not as expected in terms of the stimuli and observations, but in terms of the actual fitting; one of the clearest results is that fitting all parameters at once is often less effective than restricted fitting, in which the directions along with candidate models can move through parameter space are constrained.

The particular form of this constraint, however, does not appear to have a material impact on fitting, seeing as both targeted stimulus/observation pairs and randomised observations scattered across unrelated stimuli produced similar results on most relevant measures. While this casts doubt over the notion that the stimuli designed by the MOSTIPS algorithm are capable of isolating parameters, it is worth remembering that even the random observations tested are made in the context of such stimuli. Since the MOSTIPS-derived stimuli are designed to drive the system into very particular locations in state space, any observation made during such stimulation is likely to encounter unusual, and thus highly informative, system states, which supports adequate fitting even in the absence of appropriate parameter targeting. In other words, although the stimulus generation pipeline failed to produce uniquely targeted stimulus/observation pairs that out-

perform random observations within the same stimuli, I suspect that the stimuli themselves generate more informative observations than e.g. single steps of comparable duration, and thus allow the fitting algorithms to perform adequately.

This also suggests that part of the stimulus generation pipeline – namely, the robustness screening and selection procedures – may not be necessary at all to produce comparable results. If so, this highlights the importance of the first step of the pipeline, that is, of the cost function and stimulus search algorithm, and vindicates the pains I have taken to hone these. In particular, it was the choice of an illuminating algorithm like MAP-Elites that gave rise to stimuli that worked, as opposed to earlier attempts using less stable novelty search or single-objective genetic algorithms, the results of which were far less encouraging.

While parametrisations produced with MOSTIPS generalise similarly well as classically fitted models, the novel method has two key advantages over the classical approach. Firstly, the amount of data required for MOSTIPS fitting is much smaller, meaning that there is less interference in cellular activity, and higher information gain per acquired sample. This is in contrast to classical methods where, even with the three lengthy protocols I chose to use as shown in chapter 3, many kinetic parameters remained ill constrained and had to be fit with computationally expensive least-squares methods. Secondly, due to the highly parallelised nature of the algorithm used, fitting is faster than the iterative approach employed classically. By reducing both data and time requirements, MOSTIPS or similar approaches for parameter optimisation may be useful e.g. for high-throughput screening of the effects of pharmacological agents on ion channel function.

6.4 MOSTIPS does not solve model specificity

However, the goal of the MOSTIPS project was, of course, not to produce merely predictive models, but to produce models whose parameters directly re-

flect the underlying system. In this respect, the results have fallen short of my expectations. Neither in optimising against simulated targets, where we know the true parameter values and can compare directly, nor in optimising against oocytes, where the reference parameter values were derived from a careful alternative fitting approach, did the MOSTIPS-optimised models very closely resemble their reference. This casts doubt on whether this method a suitable tool to investigate e.g. individual variability: without strong evidence that the method reveals the "true" parameter values, i.e. values that reflect an underlying reality in e.g. current densities or kinetics, its usefulness to study these underlying realities is very limited. Thus, in its current state, I cannot recommend using the MOSTIPS approach as a tool to investigate either individual variability or non-synaptic plasticity.

6.5 Closed-loop fitting as a way forward

The second method, presented in chapter 5, uses closed-loop feedback between the reference neuron and a large population of candidate models. Based on a more gentle stimulation approach, it is more suitable for optimisation to neurons that form part of a circuit. Unlike the closed-loop approach followed by Reyes-Sanchez et al. (2018), however, it does not make any reference to circuit-level properties, and indeed ignores any interactions the reference neuron might have with coupled or pre-synaptic partners. Also, unlike the MOSTIPS approach, my closed-loop method makes no direct attempt to achieve model specificity. On the other hand, given the interaction between stimulus selection and the cost function for candidate models, we should expect the method to reduce discrepancies between reference and model not only in the input-output relationship, but also in the internal dynamics. If so, the model specificity question would solve itself by virtue of an ever-increasing resolution, driving parameter accuracy, and revealing structural deficits in the model used with an inability to converge.

While the method is likely not quite refined enough yet to achieve this, the early results presented here are very encouraging.

6.6 More tools for the field to use

The model optimisation approaches described in this thesis, as well as the extensions to the dynamic clamp software StdpC, are tools in varying stages of development and maturity. While neither of the model optimisation methods fully satisfies the ideal aspirations outlined in the introduction – i.e., fast optimisation, low impact on the target system, and accurate single-neuron specificity – they both present promising approaches. Both methods, it seems to me, have the potential to be or to become very valuable tools in the investigation of intrinsic variability and plasticity of neurons, and their interactions with circuit dynamics. I do believe, however, that more work is required to improve the methods, and to adapt them to specific systems and methods of inquiry. In contrast, most of the various tools I have developed for dynamic clamp are already in active use. With the updated interface and added flexibility, StdpC is well placed to play a major role in making dynamic clamp a commonly used and easily accessible tool for any neuroscientist working with single cells or small circuits.

6.7 Future work

There are several avenues of further research that could build on the model optimisation approaches presented in thesis. Firstly, it would be useful to more closely examine how the stimuli lead to successful model fitting, and thereby gain better insights into which parts of the stimulus generation pipeline are truly fit for service, and which could be either omitted or improved.

Then, while I have focused here on piecewise linear stimuli, the optimisation particularly of kinetic parameters might be improved by augmenting the stimulus set e.g. with superpositions of wave functions. Such stimuli were successfully used by (Beattie *et al.*, 2018) to optimise the kinetic parameters of single channels.

More generally, stimulation based on wave functions opens the possibility of driving the membrane in temporally much more complex patterns without substantially increasing the search space, i.e., without having to reinvent the stimulus generation algorithm suggested here.

Next, I have left a large space of territory unexplored in the area of model optimisation itself. The most important lesson learned in stimulus generation – that it can be very beneficial to optimise not for a single best result, but to broaden the perspective and allow many different solutions that are optimal within certain limits – was not applied at all in model optimisation, where my underlying goal remained that there should be one definitive parametrisation. However, not only is this a flawed assumption, since part of the problem of model fitting in general is precisely the degeneracy of the model structures employed, but in addition, the use of algorithms that illuminate the search space, such as novelty search (Lehman & Stanley, 2011) and MAP-Elites (Mouret & Clune, 2015), has benefits beyond the plurality of their solutions. In particular, as argued by (Lehman & Stanley, 2011), optimising for fitness alone risks the search getting stuck in local optima, while illuminating algorithms explore the search space more widely, can cross-pollinate between a variety of good solutions, and are thus more likely to find even well-hidden paths to global optima. Using such an algorithm for model optimisation would likely result in higher quality results, as well as potentially offer opportunities for the experimenter to intervene, select appropriate subsets of solutions, or better understand the solutions proposed by the algorithm.

Finally, to gain better insight into how well a given parameter is constrained, it would be very useful not only to track the point estimates of parameter values, but rather their likely distribution in parameter space, using for example covariance matrix adaptation (Hansen & Ostermeier, 2001; Vavoulis *et al.*, 2012). The additional information gained about how well, or how uncertainly, parameters are constrained could be used both as a result in itself, and as a feedback about potential mismatch between the model and the neuron investigated.

References

Amaducci, R., Reyes-Sanchez, M., Elices, I., Rodriguez, F.B., & Varona, P. (2019) RTHybrid: A standardized and open-source real-time software model library for experimental neuroscience. *Front Neuroinform*, **13**.

Amarillo, Y., Tissone, A.I., Mato, G., & Nadal, M.S. (2018) Inward rectifier potassium current IKir promotes intrinsic pacemaker activity of thalamocortical neurons. *Journal of Neurophysiology*, **119**, 2358–2372.

Ang, K.H., Chong, G., & Li, Y. (2005) PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, **13**, 559–576.

Antonov, I., Antonova, I., Kandel, E.R., & Hawkins, R.D. (2001) The contribution of activity-dependent synaptic plasticity to classical conditioning in Aplysia. *J. Neurosci.*, **21**, 6413–6422.

Armstrong, C.M. (1969) Inactivation of the potassium conductance and related phenomena caused by quaternary ammonium ion injection in squid axons. *The Journal of General Physiology*, **54**, 553–575.

Baker, D.H. & Richard, B. (2019) Dynamic properties of internal noise probed by modulating binocular rivalry. *PLOS Computational Biology*, **15**, e1007071.

Barrett, E.F. & Stevens, C.F. (1972) Quantal independence and uniformity of presynaptic release kinetics at the frog neuromuscular junction. *The Journal of Physiology*, **227**, 665–689.

Beattie, K.A., Hill, A.P., Bardenet, R., Cui, Y., Vandenberg, J.I., Gavaghan, D.J., Boer, T.P. de, & Mirams, G.R. (2018) Sinusoidal voltage protocols for rapid characterisation of ion channel kinetics. *The Journal of Physiology*, **596**, 1813–1828.

Bekkers, J.M., Richerson, G.B., & Stevens, C.F. (1990) Origin of variability in quantal size in cultured hippocampal neurons and hippocampal slices. *PNAS*, **87**, 5359–5362.

Bell, N. & Hoberock, J. (2012) Chapter 26 - Thrust: A productivity-oriented library for CUDA. In Hwu, W.W. (ed), *GPU Computing Gems Jade Edition*, *Applications of GPU Computing Series*. Morgan Kaufmann, Boston, pp. 359–371.

Bhalla, U.S. & Bower, J.M. (1993) Exploring parameter space in detailed single neuron models: simulations of the mitral and granule cells of the olfactory bulb. *Journal of Neurophysiology*, **69**, 1948–1965.

Brascamp, J.W., van Ee, R., Noest, A.J., Jacobs, R.H.A.H., & van den Berg, A.V. (2006) The time course of binocular rivalry reveals a fundamental role of noise. *Journal of Vision*, **6**, 8–8.

Brochini, L., Carelli, P.V., & Pinto, R.D. (2011) Single synapse information coding in intraburst spike patterns of central pattern generator motor neurons. *J. Neurosci.*, **31**, 12297–12306.

Brookings, T., Goeritz, M.L., & Marder, E. (2014) Automatic parameter estimation of multicompartmental neuron models via minimization of trace error with control adjustment. *Journal of Neurophysiology*, **112**, 2332–2348.

Buhry, L., Pace, M., & Saïghi, S. (2012) Global parameter estimation of an Hodgkin–Huxley formalism using membrane voltage recordings: Application to neuro-mimetic analog integrated circuits. *Neurocomputing*, **81**, 75–85.

Catterall, W.A., Raman, I.M., Robinson, H.P.C., Sejnowski, T.J., & Paulsen, O. (2012) The Hodgkin-Huxley Heritage: From Channels to Circuits. *J. Neurosci.*, **32**, 14064–14073.

Comer, M.B., Campbell, D.L., Rasmusson, R.L., Lamson, D.R., Morales, M.J., Zhang, Y., & Strauss, H.C. (1994) Cloning and characterization of an Ito-like potassium channel from ferret ventricle. *American Journal of Physiology - Heart and Circulatory Physiology*, **267**, H1383–H1395.

Cominos, P. & Munro, N. (2002) PID controllers: recent tuning methods and design to specification. *IEE Proceedings - Control Theory and Applications*, **149**, 46–53.

Das, S. & Suganthan, P.N. (2011) Differential Evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, **15**, 4–31.

De Jong, K.A. (2006) *Evolutionary Computation: A Unified Approach*. MIT press.

de Jonge, M.C., Black, J., Deyo, R.A., & Disterhoft, J.F. (1990) Learning-induced afterhyperpolarization reductions in hippocampus are specific for cell type and potassium conductance. *Exp Brain Res*, **80**, 456–462.

De Schutter, E. & Bower, J.M. (1994) An active membrane model of the cerebellar Purkinje cell. I. Simulation of current clamps in slice. *Journal of Neurophysiology*, **71**, 375–400.

Debay, D., Wolfart, J., Le Franc, Y., Le Masson, G., & Bal, T. (2004) Exploring spike transfer through the thalamus using hybrid artificial-biological neuronal networks. *Journal of Physiology-Paris*, Decoding and interfacing the brain: from neuronal assemblies to cyborgs, **98**, 540–558.

Derkach, V.A., Oh, M.C., Guire, E.S., & Soderling, T.R. (2007) Regulatory mechanisms of AMPA receptors in synaptic plasticity. *Nature Reviews Neuroscience*, **8**, 101–113.

Desai, N.S., Gray, R., & Johnston, D. (2017) A dynamic clamp on every rig. *eNeuro*, **4**.

Desai, N.S., Rutherford, L.C., & Turrigiano, G.G. (1999) Plasticity in the intrinsic excitability of cortical pyramidal neurons. *Nat Neurosci*, **2**, 515–520.

Destexhe, A. & Huguenard, J.R. (2000) Nonlinear thermodynamic models of voltage-cependent currents. *J Comput Neurosci*, **9**, 259–270.

Destexhe, A., Rudolph, M., Fellous, J.-M., & Sejnowski, T.J. (2001) Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience*, **107**, 13–24.

Diering, G.H. & Huganir, R.L. (2018) The AMPA receptor code of synaptic plasticity. *Neuron*, **100**, 314–329.

Dixon, J.E. & McKinnon, D. (1994) Quantitative analysis of potassium channel mRNA expression in atrial and ventricular muscle of rats. *Circ. Res.*, **75**, 252–260.

Druckmann, S., Berger, T.K., Schürmann, F., Hill, S., Markram, H., & Segev, I. (2011) Effective stimuli for constructing reliable neuron models. *PLOS Comput Biol*, **7**, e1002133.

Economo, M.N., Fernandez, F.R., & White, J.A. (2010) Dynamic Clamp: Alteration of response properties and creation of virtual realities in neurophysiology. *J. Neurosci.*, **30**, 2407–2413.

Elices, I., Levi, R., Arroyo, D., Rodriguez, F.B., & Varona, P. (2019) Robust dynamical invariants in sequential neural activity. *Sci Rep*, **9**, 1–13.

Erö, C., Gewaltig, M.-O., Keller, D., & Markram, H. (2018) A cell atlas for the mouse brain. *Front. Neuroinform.*, **12**.

Fehlberg, E. (1970) Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme. *Computing*, **6**, 61–71.

Fitzhugh, R. (1962) Computation of impulse initiation and saltatory conduction in a myelinated nerve fiber. *Biophysical Journal*, **2**, 11–21.

Foster, W.R., Ungar, L.H., & Schwaber, J.S. (1993) Significance of conductances in Hodgkin-Huxley models. *Journal of Neurophysiology*, **70**, 2502–2518.

Francesconi, W., Szűcs, A., Berton, F., Koob, G.F., Vendruscolo, L.F., & Sanna, P.P. (2017) Opiate dependence induces cell type-specific plasticity of intrinsic membrane properties in the rat juxtacapsular bed nucleus of stria terminalis (jcBNST). *Psychopharmacology*, **234**, 3485–3498.

Frech, G.C., VanDongen, A.M.J., Schuster, G., Brown, A.M., & Joho, R.H. (1989) A novel potassium channel with delayed rectifier properties isolated from rat brain by expression cloning. *Nature*, **340**, 642.

Freire, E.O., Rossomando, F.G., & Soria, C.M. (2018) Self-tuning of a neuro-adaptive PID controller for a SCARA robot based on neural network. *IEEE Latin America Transactions*, **16**, 1364–1374.

French, C.R., Zeng, Z., Williams, D.A., Hill-Yardin, E.L., & O'Brien, T.J. (2015) Properties of an intermediate-duration inactivation process of the voltage-gated sodium conductance in rat hippocampal CA1 neurons. *Journal of Neurophysiology*, **115**, 790–802.

Gal, A., Eytan, D., Wallach, A., Sandler, M., Schiller, J., & Marom, S. (2010) Dynamics of excitability over extended timescales in cultured cortical neurons. *J. Neurosci.*, **30**, 16332–16342.

Gämperle, R., Müller, S.D., & Koumoutsakos, P. (2002) A parameter study for differential evolution. *Advances in intelligent systems, fuzzy systems, evolutionary computation*, **10**, 293–298.

Gluzman, Y. (1981) SV40-transformed simian cells support the replication of early SV40 mutants. *Cell*, **23**, 175–182.

Goldman, M.S., Golowasch, J., Marder, E., & Abbott, L.F. (2001) Global structure, robustness, and modulation of neuronal models. *J. Neurosci.*, **21**, 5229–5238.

Golowasch, J., Goldman, M.S., Abbott, L.F., & Marder, E. (2002) Failure of averaging in the construction of a conductance-based neuron model. *Journal of Neurophysiology*, **87**, 1129–1131.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014) Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., & Weinberger, K.Q. (eds), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pp. 2672–2680.

Gordon, E., Roepke, T.K., & Abbott, G.W. (2006) Endogenous KCNE subunits govern Kv2.1K+ channel activation kinetics in Xenopus oocyte studies. *Biophysical Journal*, **90**, 1223–1231.

Guillemare, E., Honore, E., Pradier, L., Lesage, F., Schweitz, H., Attali, B., Barhanin, J., & Lazdunski, M. (1992) Effects of the level of mRNA expression on biophysical properties, sensitivity to neurotoxins, and regulation of the brain delayed-rectifier K+ channel Kv1. 2. *Biochemistry*, **31**, 12463–12468.

Günay, C., Edgerton, J.R., & Jaeger, D. (2008) Channel density distributions explain spiking variability in the globus pallidus: A combined physiology and computer simulation database approach. *J. Neurosci.*, **28**, 7476–7491.

Guzman, S.J., Schlögl, A., & Schmidt-Hieber, C. (2014) Stimfit: quantifying electrophysiological data with Python. *Front. Neuroinform.*, **8**.

Hagiwara, K., Nunoki, K., Ishii, K., Abe, T., & Yanagisawa, T. (2003) Differential inhibition of transient outward currents of Kv1.4 and Kv4.3 by endothelin. *Biochemical and Biophysical Research Communications*, **310**, 634–640.

Halliwell, J.V., Plant, T.D., Robbins, J., & Standen, N.B. (1994) Voltage clamp techniques. In *Microelectrode Techniques. The Plymouth Workshop Handbook*, 2nd edn. The Company of Biologists, Ltd. Cambridge, pp. 17–35.

Hansen, N. & Ostermeier, A. (2001) Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, **9**, 159–195.

Hashimoto, Y., Nunoki, K., Kudo, H., Ishii, K., Taira, N., & Yanagisawa, T. (2000) Changes in the inactivation of rat Kv1.4 K+ channels induced by varying the number of inactivation particles. *J. Biol. Chem.*, **275**, 9358–9362.

Hille, B. (1991) *Ionic Channels of Excitable Membranes*, 2nd ed. edn. Sinauer, Sunderland, Mass.

Hobbs, K.H. & Hooper, S.L. (2008) Using complicated, wide dynamic range driving to develop models of single neurons in single recording sessions. *Journal of Neurophysiology*, **99**, 1871–1883.

Hodgkin, A.L. & Huxley, A.F. (1952a) Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. *The Journal of Physiology*, **116**, 449.

Hodgkin, A.L. & Huxley, A.F. (1952b) The components of membrane conductance in the giant axon of Loligo. *The Journal of Physiology*, **116**, 473–496.

Hodgkin, A.L. & Huxley, A.F. (1952c) The dual effect of membrane potential on sodium conductance in the giant axon of Loligo. *The Journal of Physiology*, **116**, 497.

Hodgkin, A.L. & Huxley, A.F. (1952d) A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, **117**, 500–544.

Hodgkin, A.L., Huxley, A.F., & Katz, B. (1952) Measurement of current-voltage relations in the membrane of the giant axon of Loligo. *The Journal of Physiology*, **116**, 424–448.

Holland, J.H. (1992) *Adaptation In Natural And Artificial Systems: An Introductory Analysis With Applications To Biology, Control, And Artificial Intelligence*.

Hong, S.Z., Kim, H.R., & Fiorillo, C.D. (2014) T-type calcium channels promote predictive homeostasis of input-output relations in thalamocortical neurons of lateral geniculate nucleus. *Front. Comput. Neurosci.*, **8**.

Honore, E., Attali, B., Romey, G., Lesage, F., Barhanin, J., & Lazdunski, M. (1992) Different types of K+ channel current are generated by different levels of a single mRNA. *The EMBO journal*, **11**, 2465–2471.

Huang, S., Bridi, M.S., & Kirkwood, A. (2018) Dynamic recovery from depression enables rate encoding in inhibitory synapses. *bioRxiv*, 379081.

Hughes, S.W., Cope, D.W., & Crunelli, V. (1998) Dynamic clamp study of Ih modulation of burst firing and δ oscillations in thalamocortical neurons in vitro. *Neuroscience*, **87**, 541–550.

Huguet, G., Rinzel, J., & Hupé, J.-M. (2014) Noise and adaptation in multistable perception: Noise drives when to switch, adaptation determines percept choice. *Journal of Vision*, **14**, 19–19.

Hunter, J.D. (2007) Matplotlib: A 2D graphics environment. *Computing in Science Engineering*, **9**, 90–95.

Huys, Q.J.M. & Paninski, L. (2009) Smoothing of, and parameter estimation from, noisy biophysical recordings. *PLOS Comput Biol*, **5**, e1000379.

Katz, B. & Miledi, R. (1965) The measurement of synaptic delay, and the time course of acetylcholine release at the neuromuscular junction. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, **161**, 483–495.

Kemenes, I., Marra, V., Crossley, M., Samu, D., Staras, K., Kemenes, G., & Nowotny, T. (2011) Dynamic clamp with StdpC software. *Nat. Protocols*, **6**, 405–417.

Kemenes, I., Straub, V.A., Nikitin, E.S., Staras, K., O'Shea, M., Kemenes, G., & Benjamin, P.R. (2006) Role of delayed nonsynaptic neuronal plasticity in long-term associative memory. *Current Biology*, **16**, 1269–1279.

Kerschensteiner, D. & Stocker, M. (1999) Heteromeric assembly of Kv2.1 with Kv9.3: Effect on the state dependence of inactivation. *Biophysical Journal*, **77**, 248–257.

Kim, Y., Yang, G.R., Pradhan, K., Venkataraju, K.U., Bota, M., García del Molino, L.C., Fitzgerald, G., Ram, K., He, M., Levine, J.M., Mitra, P., Huang, Z.J., Wang, X.-J., & Osten, P. (2017) Brain-wide maps reveal stereotyped cell-type-based cortical architecture and subcortical sexual dimorphism. *Cell*, **171**, 456-469.e22.

Kim, Y.-J., Grabowecky, M., & Suzuki, S. (2006) Stochastic resonance in binocular rivalry. *Vision Research*, **46**, 392–406.

Klemic, K.G., Shieh, C.-C., Kirsch, G.E., & Jones, S.W. (1998) Inactivation of Kv2.1 potassium channels. *Biophysical Journal*, **74**, 1779–1789.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & al, et (2016) Jupyter Notebooks - a publishing format for reproducible computational workflows. In *ELPUB*.

Kogo, N., Kern, F.B., Nowotny, T., van Ee, R., van Wezel, R., & Aihara, T. (under review) Dynamics of a mutual inhibition circuit between two visual cortical neurons, resembling human perceptual competition.

Kostuk, M., Toth, B.A., Meliza, C.D., Margoliash, D., & Abarbanel, H.D.I. (2012) Dynamical estimation of neuron and network properties II: path integral Monte Carlo methods. *Biol Cybern*, **106**, 155–167.

Kurata, H.T. & Fedida, D. (2006) A structural interpretation of voltage-gated potassium channel inactivation. *Progress in Biophysics and Molecular Biology*, **92**, 185–208.

Laing, C.R. & Chow, C.C. (2002) A spiking neuron model for binocular rivalry. *Journal of computational neuroscience*, **12**, 39–53.

Le Masson, G., Renaud-Le Masson, S., Debay, D., & Bal, T. (2002) Feedback inhibition controls spike transfer in hybrid thalamic circuits. *Nature*, **417**, 854–858.

Lee, J., Smaill, B., & Smith, N. (2006) Hodgkin–Huxley type ion channel characterization: An improved method of voltage clamp experiment parameter estimation. *Journal of Theoretical Biology*, **242**, 123–134.

Lehman, J. & Stanley, K.O. (2011) Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, **19**, 189–223.

Lein, E.S., Hawrylycz, M.J., Ao, N., Ayres, M., Bensinger, A., Bernard, A., Boe, A.F., Boguski, M.S., Brockway, K.S., Byrnes, E.J., Chen, L., Chen, L., Chen, T.-M., Chin, M.C., Chong, J., Crook, B.E., Czaplinska, A., Dang, C.N., Datta, S., Dee, N.R., Desaki, A.L., Desta, T., Diep, E., Dolbeare, T.A., Donelan, M.J., Dong, H.-W., Dougherty, J.G., Duncan, B.J., Ebbert, A.J., Eichele, G., Estin, L.K., Faber, C., Facer, B.A., Fields, R., Fischer, S.R., Fliss, T.P., Frensley, C., Gates, S.N., Glattfelder, K.J., Halverson, K.R., Hart, M.R., Hohmann, J.G., Howell, M.P., Jeung, D.P., Johnson, R.A., Karr, P.T., Kawal, R., Kidney, J.M., Knapik, R.H., Kuan, C.L., Lake, J.H., Laramee, A.R., Larsen, K.D., Lau, C., Lemon, T.A., Liang, A.J., Liu, Y., Luong, L.T., Michaels, J., Morgan, J.J., Morgan, R.J., Mortrud, M.T., Mosqueda, N.F., Ng, L.L., Ng, R., Orta, G.J., Overly, C.C., Pak, T.H., Parry, S.E., Pathak, S.D., Pearson, O.C., Puchalski, R.B., Riley, Z.L., Rockett, H.R., Rowland, S.A., Royall, J.J., Ruiz, M.J., Sarno, N.R., Schaffnit, K., Shapovalova, N.V., Sivisay, T., Slaughterbeck, C.R., Smith, S.C., Smith, K.A., Smith, B.I., Sodt, A.J., Stewart, N.N., Stumpf, K.-R., Sunkin, S.M., Sutram, M., Tam, A., Teemer, C.D., Thaller, C., Thompson, C.L., Varnam, L.R., Visel, A., Whitlock, R.M., Wohnoutka, P.E., Wolkey, C.K., Wong, V.Y., Wood, M., Yaylaoglu, M.B., Young, R.C., Youngstrom, B.L., Yuan, X.F., Zhang, B., Zwingman, T.A., & Jones, A.R. (2007) Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, **445**, 168–176.

Lennartson, B. & Kristiansson, B. (2009) Evaluation and tuning of robust PID controllers. *IET Control Theory Applications*, **3**, 294–302.

Levelt, W.J. (1965) On binocular rivalry (PhD Thesis).

Li, X.-T., Li, X.-Q., Hu, X.-M., & Qiu, X.-Y. (2015) The inhibitory effects of Ca2+ channel blocker nifedipine on rat Kv2.1 potassium channels. *PLoS One*, **10**.

Lien, C.-C. & Jonas, P. (2003) Kv3 potassium conductance is necessary and kinetically optimized for high-frequency action potential generation in hippocampal interneurons. *J. Neurosci.*, **23**, 2058–2068.

Linaro, D., Couto, J., & Giugliano, M. (2014) Command-line cellular electrophysiology for conventional and real-time closed-loop experiments. *Journal of Neuroscience Methods*, **230**, 5–19.

Liu, Z., Golowasch, J., Marder, E., & Abbott, L.F. (1998) A model neuron with activity-dependent conductances regulated by multiple calcium sensors. *J. Neurosci.*, **18**, 2309–2320.

Ma, M. & Koester, J. (1996) The role of K+ currents in frequency-dependent spike broadening in Aplysia R20 neurons: A dynamic-clamp analysis. *J. Neurosci.*, **16**, 4089–4101.

MacLean, J.N., Zhang, Y., Johnson, B.R., & Harris-Warrick, R.M. (2003) Activity-independent homeostasis in rhythmically active neurons. *Neuron*, **37**, 109–120.

Marder, E. & Calabrese, R.L. (1996) Principles of rhythmic motor pattern generation. *Physiological Reviews*, **76**, 687–717.

Marder, E. & Goaillard, J.-M. (2006) Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, **7**, 563–574.

McCrossan, Z.A., Roepke, T.K., Lewis, A., Panaghie, G., & Abbott, G.W. (2009) Regulation of the Kv2.1 Potassium Channel by MinK and MiRP1. *J Membrane Biol*, **228**, 1–14.

McIntosh, P., Southan, A.P., Akhtar, S., Sidera, C., Ushkaryov, Y., Dolly, J.O., & Robertson, B. (1997) Modification of rat brain Kv1.4 channel gating by association with accessory Kvβ1.1 and β2.1 subunits. *Pflügers Arch*, **435**, 43–54.

Meliza, C.D., Kostuk, M., Huang, H., Nogaret, A., Margoliash, D., & Abarbanel, H.D.I. (2014) Estimating parameters and predicting membrane voltages with conductance-based neuron models. *Biol Cybern*, **108**, 495–516.

Migliore, M. & Shepherd, G.M. (2002) Emerging rules for the distributions of active dendritic conductances. *Nature Reviews Neuroscience*, **3**, 362.

Migliore, R., Lupascu, C.A., Bologna, L.L., Romani, A., Courcol, J.-D., Antonel, S., Geit, W.A.H.V., Thomson, A.M., Mercer, A., Lange, S., Falck, J., Rössert, C.A., Shi, Y., Hagens, O., Pezzoli, M., Freund, T.F., Kali, S., Muller, E.B., Schürmann, F., Markram, H., & Migliore, M. (2018) The physiological variability of channel density in hippocampal CA1 pyramidal cells and interneurons explored using a unified data-driven modeling workflow. *PLOS Computational Biology*, **14**, e1006423.

Milescu, L.S., Yamanishi, T., Ptak, K., Mogri, M.Z., & Smith, J.C. (2008) Real-time kinetic modeling of voltage-gated ion channels using dynamic clamp. *Biophysical Journal*, **95**, 66–87.

Moran, O., Schreibmayer, W., Weigl, L., Dascal, N., & Lotan, I. (1992) Level of expression controls modes of gating of a K+ channel. *FEBS letters*, **302**, 21–25.

Moreno-Bote, R., Rinzel, J., & Rubin, N. (2007) Noise-induced alternations in an attractor network model of perceptual bistability. *Journal of Neurophysiology*, **98**, 1125–1139.

Morris, M.D. (1991) Factorial sampling plans for preliminary computational experiments. *Technometrics*, **33**, 161–174.

Mouret, J.-B. & Clune, J. (2015) Illuminating search spaces by mapping elites. *arXiv:1504.04909 [cs, q-bio]*,.

Moyer, J.R.Jr., Thompson, L.T., & Disterhoft, J.F. (1996) Trace eyeblink conditioning increases CA1 excitability in a transient and learning-specific manner. *J. Neurosci.*, **16**, 5536–5546.

Mozzachiodi, R. & Byrne, J.H. (2010) More than synaptic plasticity: Role of nonsynaptic plasticity in learning and memory. *Trends Neurosci*, **33**, 17.

Murakami, T.C., Mano, T., Saikawa, S., Horiguchi, S.A., Shigeta, D., Baba, K., Sekiya, H., Shimizu, Y., Tanaka, K.F., Kiyonari, H., Iino, M., Mochizuki, H., Tainaka, K., & Ueda, H.R. (2018) A three-dimensional single-cell-resolution whole-brain atlas using CUBIC-X expansion microscopy and tissue clearing. *Nat Neurosci*, **21**, 625–637.

Murakoshi, H. & Trimmer, J.S. (1999) Identification of the Kv2.1 K+ channel as a major component of the delayed rectifier K+ current in rat hippocampal neurons. *J. Neurosci.*, **19**, 1728–1735.

Nikitin, E.S., Balaban, P.M., & Kemenes, G. (2013) Nonsynaptic plasticity underlies a compartmentalized increase in synaptic efficacy after classical conditioning. *Current Biology*, **23**, 614–619.

Nikitin, E.S., Vavoulis, D.V., Kemenes, I., Marra, V., Pirger, Z., Michel, M., Feng, J., O'Shea, M., Benjamin, P.R., & Kemenes, G. (2008) Persistent sodium current is a nonsynaptic substrate for long-term associative memory. *Current Biology*, **18**, 1221–1226.

Noest, A.J., Van Ee, R., Nijs, M.M., & Van Wezel, R.J.A. (2007) Percept-choice sequences driven by interrupted ambiguous stimuli: A low-level neural model. *Journal of vision*, **7**, 10–10.

Nowotny, T., Levi, R., & Selverston, A.I. (2008) Probing the dynamics of identified neurons with a data-driven modeling approach. *PLOS ONE*, **3**, e2627.

Nowotny, T., Szűcs, A., Pinto, R.D., & Selverston, A.I. (2006) StdpC: A modern dynamic clamp. *Journal of Neuroscience Methods*, **158**, 287–299.

Nunoki, K., Ishii, K., Okada, H., Yamagishi, T., Murakoshi, H., & Taira, N. (1994) Hybrid potassium channels by tandem linkage of inactivating and non-inactivating subunits. *J. Biol. Chem.*, **269**, 24138–24142.

Oliphant, T.E. (2007) Python for scientific computing. *Computing in Science Engineering*, **9**, 10–20.

Oprisan, S.A., Prinz, A.A., & Canavier, C.C. (2004) Phase resetting and phase locking in hybrid circuits of one model and one biological neuron. *Biophysical Journal*, **87**, 2283–2298.

Phillips, J.W., Schulmann, A., Hara, E., Liu, C., Wang, L., Shields, B.C., Korff, W., Lemire, A.L., Dudman, J., Nelson, S.B., & Hantman, A. (2018) A single spectrum of neuronal identities across thalamus. *bioRxiv*, 241315.

Pinheiro de Moura, J., da Fonseca Neto, J.V., & Rêgo, P.H.M. (2019) A neuro-fuzzy model for online optimal tuning of PID controllers in industrial systems applications to the mining sector. *IEEE Transactions on Fuzzy Systems*, 1–1.

Pisarchik, A.N., Jaimes-Reátegui, R., Magallón-García, C.D.A., & Castillo-Morales, C.O. (2014) Critical slowing down and noise-induced intermittency in bistable perception: bifurcation analysis. *Biol Cybern*, **108**, 397–404.

Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., & Destexhe, A. (2008) Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biol Cybern*, **99**, 427–441.

Prinz, A.A., Abbott, L.F., & Marder, E. (2004) The dynamic clamp comes of age. *Trends in Neurosciences*, **27**, 218–224.

Prinz, A.A., Billimoria, C.P., & Marder, E. (2003) Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of Neurophysiology*, **90**, 3998–4015.

Prinz, A.A., Bucher, D., & Marder, E. (2004) Similar network activity from disparate circuit parameters. *Nat Neurosci*, **7**, 1345–1352.

Qi, Z., Shi, Q., & Zhang, H. (2019) Tuning of digital PID controllers using particle swarm optimization algorithm for a CAN-based DC motor subject to stochastic delays. *IEEE Transactions on Industrial Electronics*, 1–1.

Qin, A.K., Huang, V.L., & Suganthan, P.N. (2009) Differential Evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, **13**, 398–417.

Ranjan, R. & Khanna, N. (2019) Channelpedia: Kv2.1 [WWW Document]. URL https://channelpedia.net/ionchannels/9

Ranjan, R., Schartner, M., & Khanna, N. (2019) Channelpedia: Kv1.4 [WWW Document]. URL https://channelpedia.net/ionchannels/4

Rasmusson, R.L., Morales, M.J., Castellino, R.C., Zhang, Y., Campbell, D.L., & Strauss, H.C. (1995) C-type inactivation controls recovery in a fast inactivating cardiac K+ channel (Kv1.4) expressed in Xenopus oocytes. *The Journal of Physiology*, **489**, 709–721.

Redman, S. (1990) Quantal analysis of synaptic potentials in neurons of the central nervous system. *Physiological Reviews*, **70**, 165–198.

Reeves, C. & Rowe, J.E. (2002) *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Springer Science & Business Media.

Rettig, J., Heinemann, S.H., Wunder, F., Lorra, C., Parcej, D.N., Oliver Dolly, J., & Pongs, O. (1994) Inactivation properties of voltage-gated K+ channels altered by presence of β-subunit. *Nature*, **369**, 289–294.

Reyes, M.B., Huerta, R., Rabinovich, M.I., & Selverston, A.I. (2008) Artificial synaptic modification reveals a dynamical invariant in the pyloric CPG. *Eur J Appl Physiol*, **102**, 667–675.

Reyes-Sanchez, M., Amaducci, R., Elices, I., Rodríguez, F.B., & Varona, P. (2018) Automatic adaptation of model neurons and connections to build hybrid circuits with living networks. *bioRxiv*, 419622.

Rinzel, J. (1990) Discussion: Electrical excitability of cells, theory and experiment: Review of the Hodgkin-Huxley foundation and an update. *Bltn Mathcal Biology*, **52**, 3–23.

Robinson, H.P.C. & Kawai, N. (1993) Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons. *Journal of Neuroscience Methods*, **49**, 157–165.

Rohatgi, A. (2019) WebPlotDigitizer [WWW Document]. URL https://apps.automeris.io/wpd/

Saar, D., Grossman, Y., & Barkai, E. (1998) Reduced after-hyperpolarization in rat piriform cortex pyramidal neurons is associated with increased learning capability during operant conditioning. *European Journal of Neuroscience*, **10**, 1518–1523.

Sakurai, A. & Katz, P.S. (2016) The central pattern generator underlying swimming in Dendronotus iris: a simple half-center network oscillator with a twist. *Journal of Neurophysiology*, **116**, 1728–1742.

Sakurai, A. & Katz, P.S. (2017) Artificial synaptic rewiring demonstrates that distinct neural circuit configurations underlie homologous behaviors. *Current Biology*, **27**, 1721-1734.e3.

Sakurai, A. & Katz, P.S. (2019) Command or obey? homologous neurons differ in hierarchical position for the generation of homologous behaviors. *J. Neurosci.*, **39**, 6460–6471.

Sakurai, A., Tamvacakis, A.N., & Katz, P.S. (2014) Hidden synaptic differences in a neural circuit underlie differential behavioral susceptibility to a neural injury. *Elife*, **3**, e02598.

Saltelli, A. & Annoni, P. (2010) How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, **25**, 1508–1517.

Samu, D., Marra, V., Kemenes, I., Crossley, M., Kemenes, G., Staras, K., & Nowotny, T. (2012) Single electrode dynamic clamp with StdpC. *J Neurosci Methods*, **211**, 11–21.

Schulz, D.J., Goaillard, J.-M., & Marder, E. (2006) Variable channel expression in identified single and electrically coupled neurons in different animals. *Nat Neurosci*, **9**, 356–362.

Sharp, A.A., O'Neil, M.B., Abbott, L.F., & Marder, E. (1993) Dynamic clamp: computer-generated conductances in real neurons. *Journal of Neurophysiology*, **69**, 992–995.

Shi, G., Kleinklaus, A.K., Marrion, N.V., & Trimmer, J.S. (1994) Properties of Kv2.1 K+ channels expressed in transfected mammalian cells. *J. Biol. Chem.*, **269**, 23204–23211.

Shpiro, A., Moreno-Bote, R., Rubin, N., & Rinzel, J. (2009) Balance between noise and adaptation in competition models of perceptual bistability. *Journal of computational neuroscience*, **27**, 37.

Sieling, F.H., Canavier, C.C., & Prinz, A.A. (2009) Predictions of phase-locking in excitatory hybrid networks: excitation does not promote phase-locking in pattern-generating networks as reliably as inhibition. *Journal of Neurophysiology*, **102**, 69–84.

Sokolova, I.V., Szűcs, A., & Paolo Sanna, P. (2019) Reduced intrinsic excitability of CA1 pyramidal neurons in human immunodeficiency virus (HIV) transgenic rats. *Brain Research*, 146431.

Song, I. & Huganir, R.L. (2002) Regulation of AMPA receptors during synaptic plasticity. *Trends in Neurosciences*, **25**, 578–588.

Soofi, W., Goeritz, M.L., Kispersky, T.J., Prinz, A.A., Marder, E., & Stein, W. (2014) Phase maintenance in a rhythmic motor pattern during temperature changes in vivo. *Journal of Neurophysiology*, **111**, 2603–2613.

Sorensen, M., DeWeerth, S., Cymbalyuk, G., & Calabrese, R.L. (2004) Using a hybrid neural system to reveal regulation of neuronal network activity by an intrinsic current. *J. Neurosci.*, **24**, 5427–5438.

Stephens, G.J., Owen, D.G., & Robertson, B. (1996) Cysteine-modifying reagents alter the gating of the rat cloned potassium channel Kv1.4. *Pflugers Arch.*, **431**, 435–442.

Stevens, C.F. (1993) Quantal release of neurotransmitter and long-term potentiation. *Cell*, **72**, 55–63.

Storn, R. & Price, K. (1997) Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **11**, 341–359.

Strassberg, A.F. & DeFelice, L.J. (1993) Limitations of the Hodgkin-Huxley formalism: effects of single channel kinetics on transmembrane voltage dynamics. *Neural Computation*, **5**, 843–855.

Stühmer, W., Ruppersberg, J.P., Schröter, K.H., Sakmann, B., Stocker, M., Giese, K.P., Perschke, A., Baumann, A., & Pongs, O. (1989) Molecular basis of functional diversity of voltage-gated potassium channels in mammalian brain. *EMBO J*, **8**, 3235–3244.

Svensson, C.-M., Coombes, S., & Peirce, J.W. (2012) Using evolutionary algorithms for fitting high-dimensional models to neuronal data. *Neuroinform*, **10**, 199–218.

Szegedi, V., Paizs, M., Baka, J., Barzo, P., Molnar, G., Tamas, G., & Lamsa, K. (2019) Robust perisomatic GABAergic self-innervation inhibits basket cells in the human and mouse supragranular neocortex. *bioRxiv*, 760983.

Szűcs, A., Berton, F., Nowotny, T., Sanna, P., & Francesconi, W. (2010) Consistency and diversity of spike dynamics in the neurons of bed nucleus of stria terminalis of the rat: A dynamic clamp study. *PLOS ONE*, **5**, e11920.

Szűcs, A., Berton, F., Sanna, P.P., & Francesconi, W. (2012) Excitability of jcBNST neurons is reduced in alcohol-dependent animals during protracted alcohol withdrawal. *PLOS ONE*, **7**, e42313.

Szűcs, A. & Huerta, R. (2015) Differential effects of static and dynamic inputs on neuronal excitability. *Journal of Neurophysiology*, **113**, 232–243.

Szűcs, A., Rátkai, A., Schlett, K., & Huerta, R. (2017) Frequency-dependent regulation of intrinsic excitability by voltage-activated membrane conductances, computational modeling and dynamic clamp. *European Journal of Neuroscience*, **46**, 2429–2444.

Taylor, A.L., Hickey, T.J., Prinz, A.A., & Marder, E. (2006) Structure and visualization of high-dimensional conductance spaces. *Journal of Neurophysiology*, **96**, 891–905.

Tomaiuolo, M., Bertram, R., Leng, G., & Tabak, J. (2012) Models of electrical activity: calibration and prediction testing on the same cell. *Biophysical Journal*, **103**, 2021–2032.

Toth, B.A., Kostuk, M., Meliza, C.D., Margoliash, D., & Abarbanel, H.D.I. (2011) Dynamical estimation of neuron and network properties I: variational methods. *Biol Cybern*, **105**, 217–237.

Traub, R.D., Wong, R.K., Miles, R., & Michelson, H. (1991) A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *Journal of Neurophysiology*, **66**, 635–650.

Turrigiano, G., Abbott, L.F., & Marder, E. (1994) Activity-dependent changes in the intrinsic properties of cultured neurons. *Science*, **264**, 974–977.

Vaidya, S.P. & Johnston, D. (2013) Temporal synchrony and gamma-to-theta power conversion in the dendrites of CA1 pyramidal neurons. *Nature Neuroscience*, **16**, 1812–1820.

Van Geit, W., De Schutter, E., & Achard, P. (2008) Automated neuron model optimization techniques: a review. *Biol Cybern*, **99**, 241–251.

VanDongen, A.M.J., Frech, G.C., Drewe, J.A., Joho, R.H., & Brown, A.M. (1990) Alteration and restoration of K+ channel function by deletions at the N- and C-termini. *Neuron*, **5**, 433–443.

Vanier, M.C. & Bower, J.M. (1999) A comparative survey of automated parameter-search methods for compartmental neural models. *J Comput Neurosci*, **7**, 149–171.

Vavoulis, D.V., Straub, V.A., Aston, J.A.D., & Feng, J. (2012) A self-organizing state-space-model approach for parameter estimation in Hodgkin-Huxley-type models of single neurons. *PLOS Comput Biol*, **8**, e1002401.

Vehovszky, A., Szabo, H., & Elliott, C.J.H. (2005) Octopamine increases the excitability of neurons in the snail feeding system by modulation of inward sodium current but not outward potassium currents. *BMC Neurosci.*, **6**, 70.

Wang, L. (2017) Automatic tuning of PID controllers using frequency sampling filters. *IET Control Theory Applications*, **11**, 985–995.

Wickenden, A.D., Tsushima, R.G., Losito, V.A., Kaprielian, R., & Backx, P.H. (1999) Effect of Cd2+ on Kv4.2 and Kv1.4 expressed in Xenopus oocytes and on the transient outward currents in rat and rabbit ventricular myocytes. *CPB*, **9**, 11–28.

Wilders, R., Verheijck, E.E., Kumar, R., Goolsby, W.N., van Ginneken, A.C., Joyner, R.W., & Jongsma, H.J. (1996) Model clamp and its application to synchronization of rabbit sinoatrial node cells. *American Journal of Physiology-Heart and Circulatory Physiology*, **271**, H2168–H2182.

Willms, A.R., Baro, D.J., Harris-Warrick, R.M., & Guckenheimer, J. (1999) An improved parameter estimation method for Hodgkin-Huxley models. *J Comput Neurosci*, **6**, 145–168.

Yang, T., Wathen, M.S., Felipe, A., Tamkun, M.M., Snyders, D.J., & Roden, D.M. (1994) K+ currents and K+ channel mRNA in cultured atrial cardiac myocytes (AT-1 cells). *Circ Res*, **75**, 870–878.

Yavuz, E., Turner, J., & Nowotny, T. (2016) GeNN: a code generation framework for accelerated brain simulations. *Scientific Reports*, **6**, 18854.

Zhang, W. & Linden, D.J. (2003) The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, **4**, 885–900.

Zhang, Y., Oliva, R., Gisselmann, G., Hatt, H., Guckenheimer, J., & Harris-Warrick, R.M. (2003) Overexpression of a hyperpolarization-activated cation current (Ih) channel gene modifies the firing activity of identified motor neurons in a small neural network. *J. Neurosci.*, **23**, 9059–9067.

Zhuang, M. & Atherton, D.P. (1993) Automatic tuning of optimum PID controllers. *IEE Proceedings D (Control Theory and Applications)*, **140**, 216–224.

Appendix

The following pages display the MOSTIPS stimuli used in chapter 4. With the exception of the final two figures (A.15 and A.16), which contain one Kv2.1k stimulus set each, figures are organised in column-wise fashion. Each column contains the full stimulus set for a model, stimulus generation algorithm, and weighting scheme used during selection as indicated in the figure legend. Displayed are the command voltages with a solid line plotted against time, with shading indicating the observation window(s) chosen by the stimulus generation algorithm. The stimulus sets in figures A.7, A.8, A.11 and A.13 were each selected from one MAP-Elites archive, using the indicated weighting scheme, but with slight differences in how the robustness correlation coefficient ρ was normalised. Results in chapter 4 were combined across recordings from all three stimulus sets in these cases.

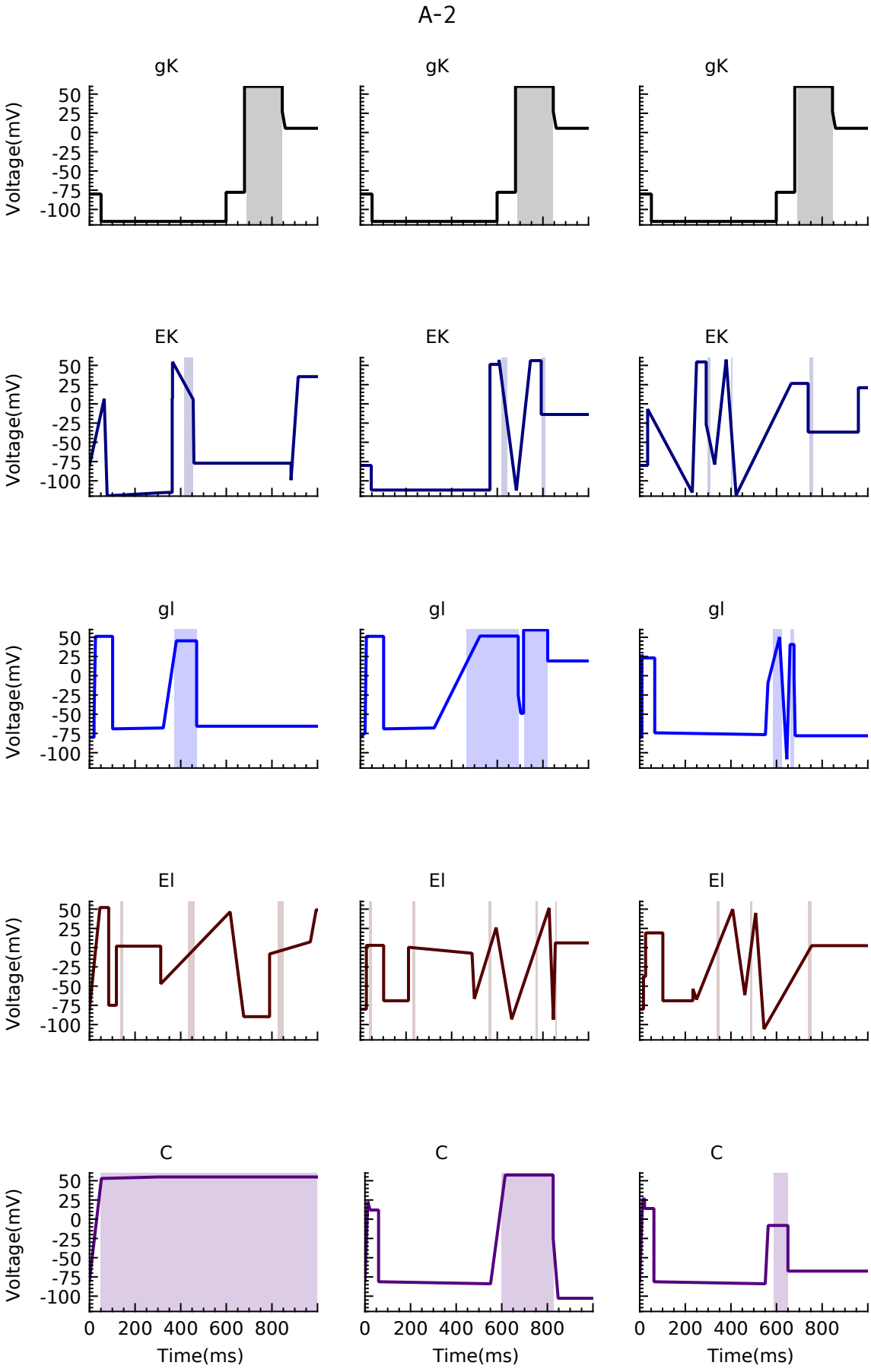


Figure A.1: Kv2.1 stimuli, cluster, left to right: weighted, unweighted, target-only

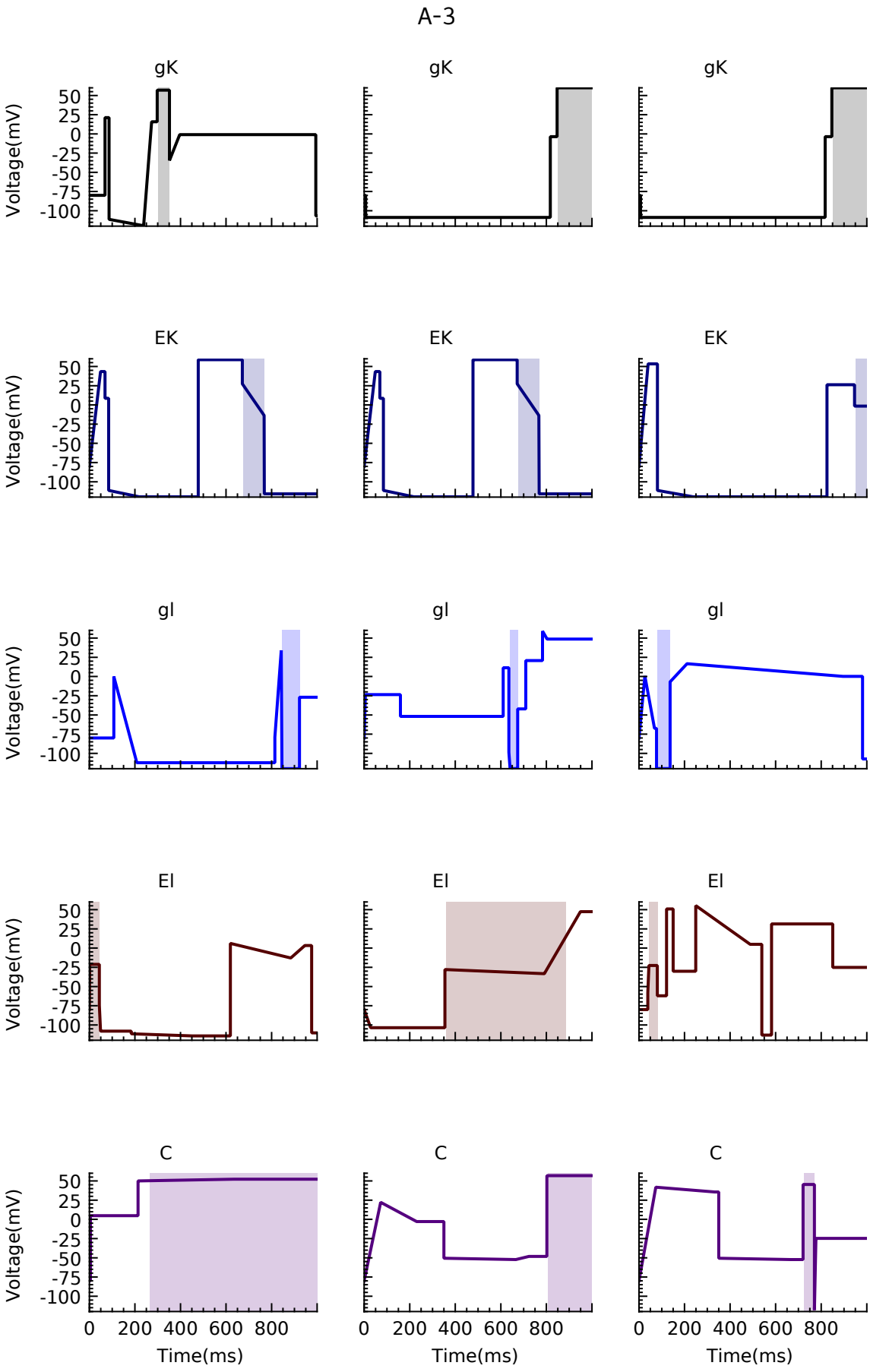


Figure A.2: Kv2.1 stimuli, bubble, left to right: weighted, unweighted, target-only

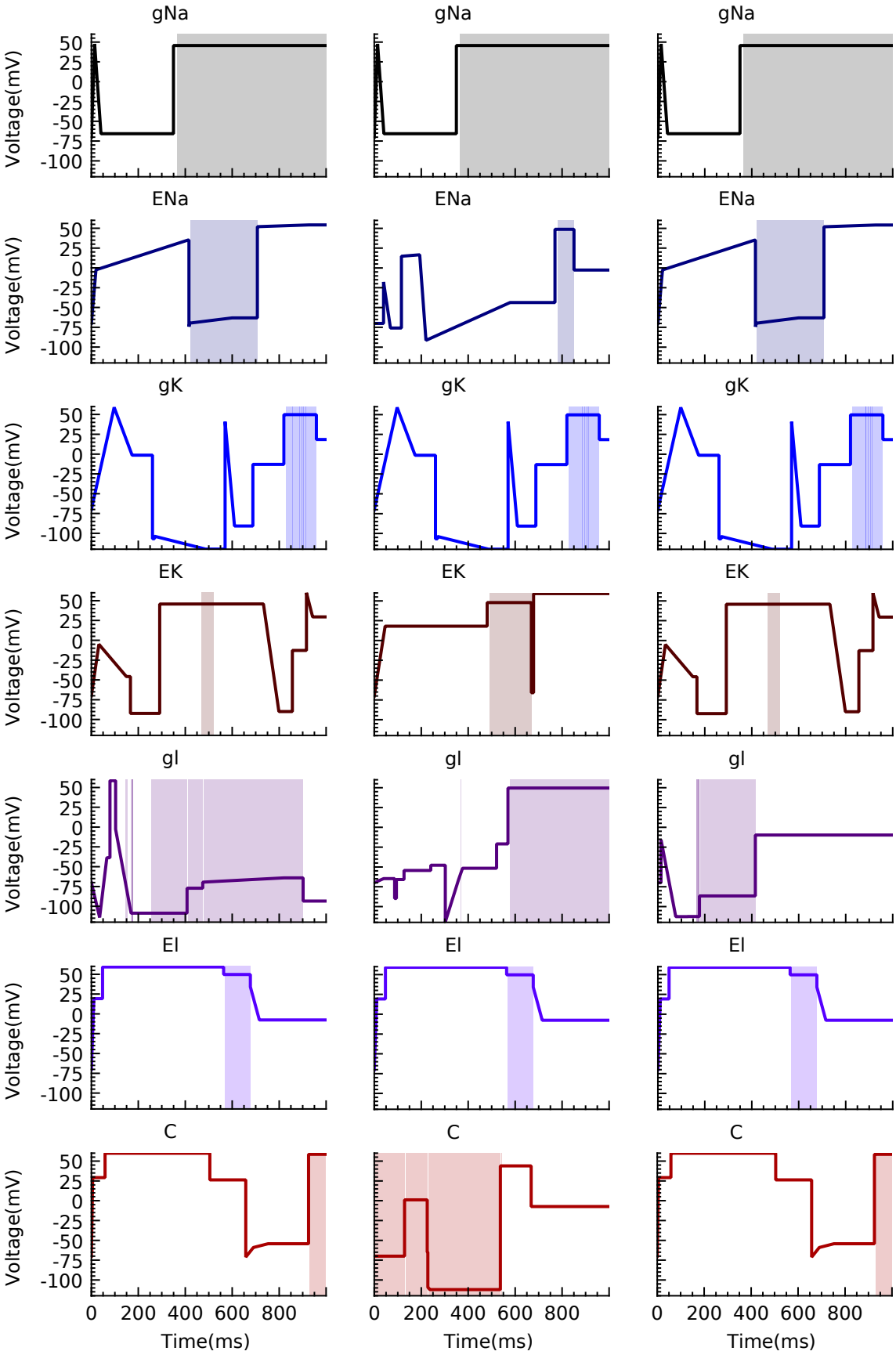


Figure A.3: HH stimuli, cluster, left to right: weighted, unweighted, target-only

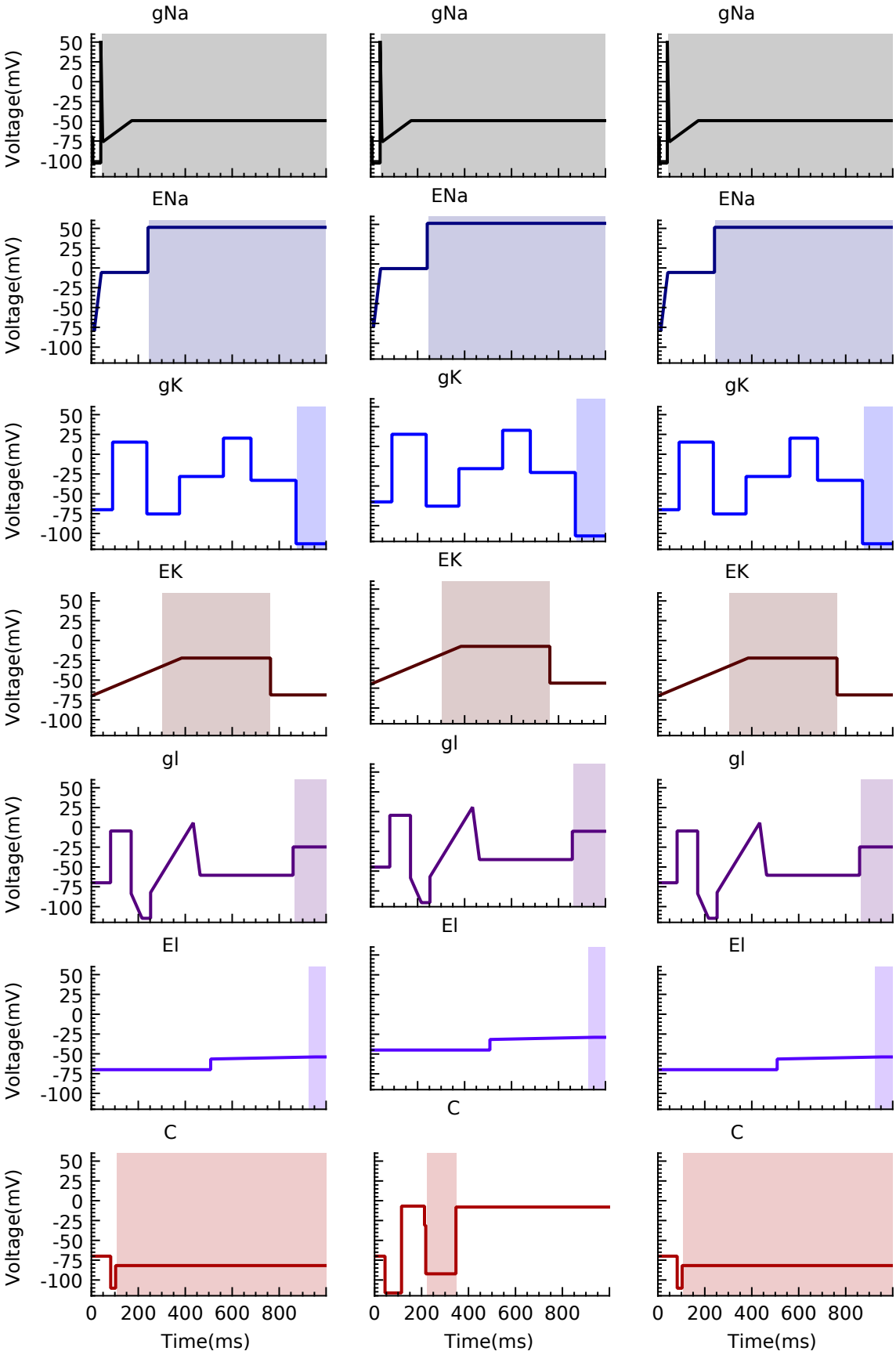


Figure A.4: HH stimuli, bubble, left to right: weighted, unweighted, target-only

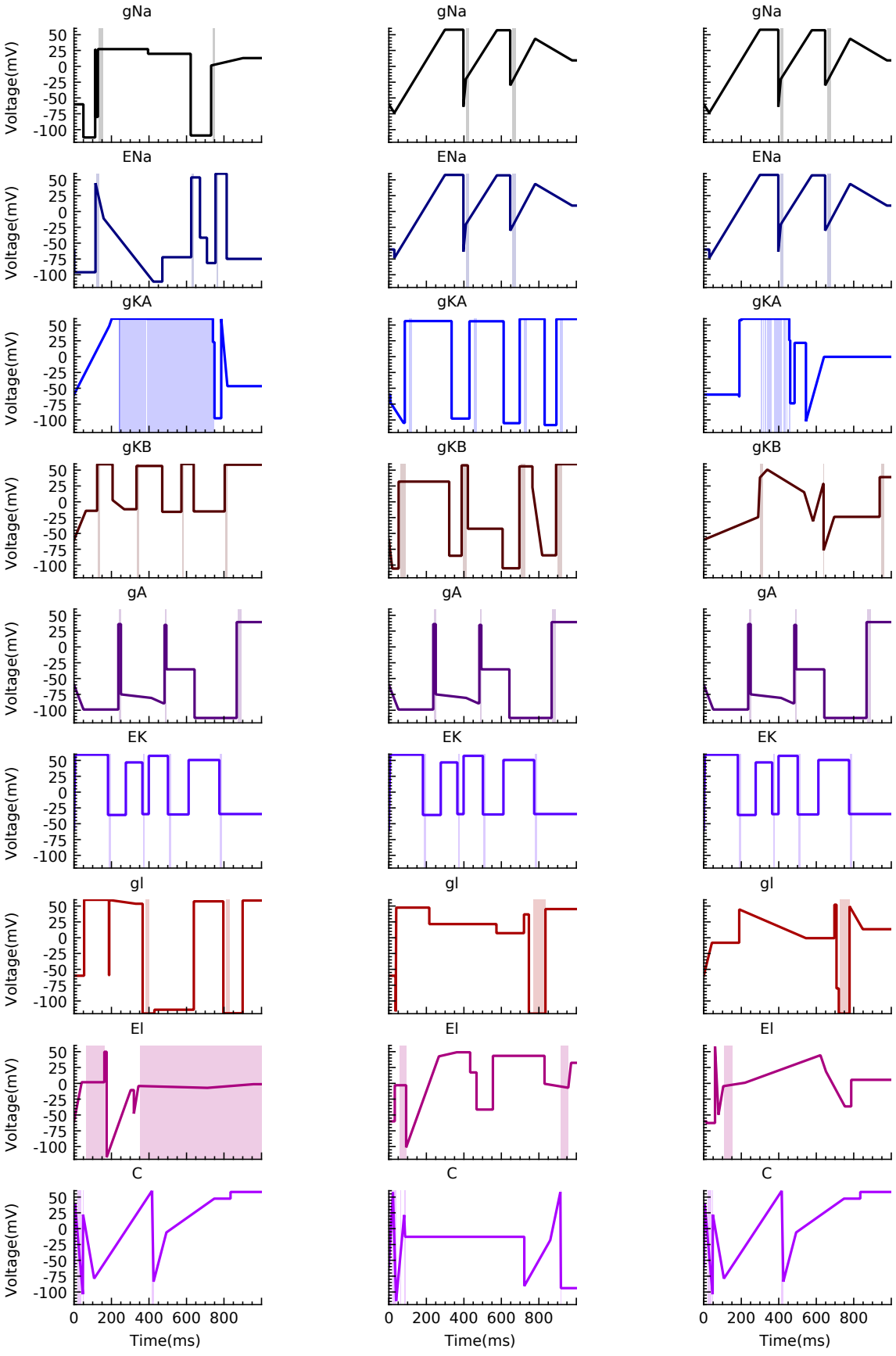


Figure A.5: B1 stimuli, cluster, left to right: weighted, unweighted, target-only

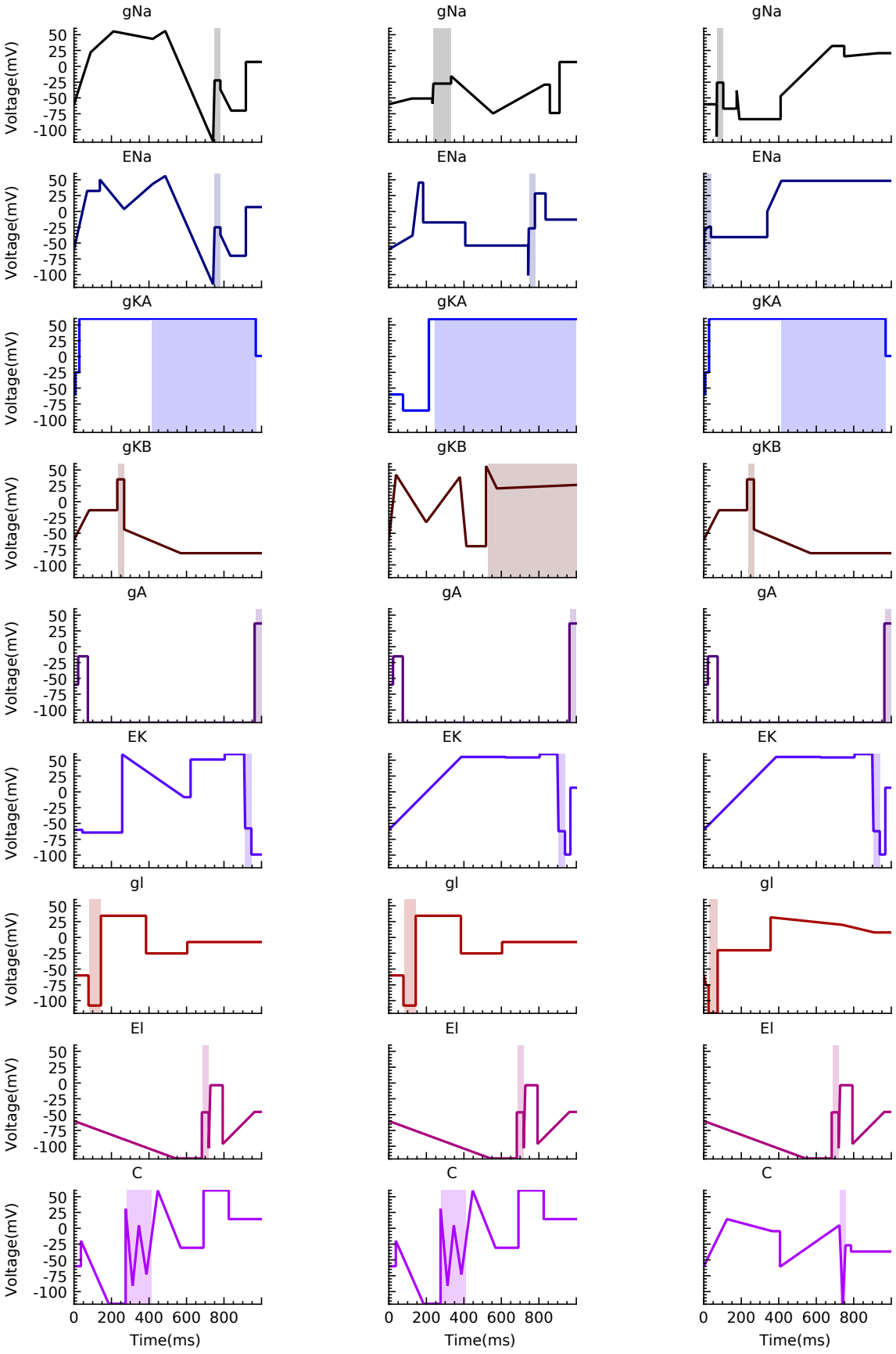


Figure A.6: B1 stimuli, bubble, left to right: weighted, unweighted, target-only

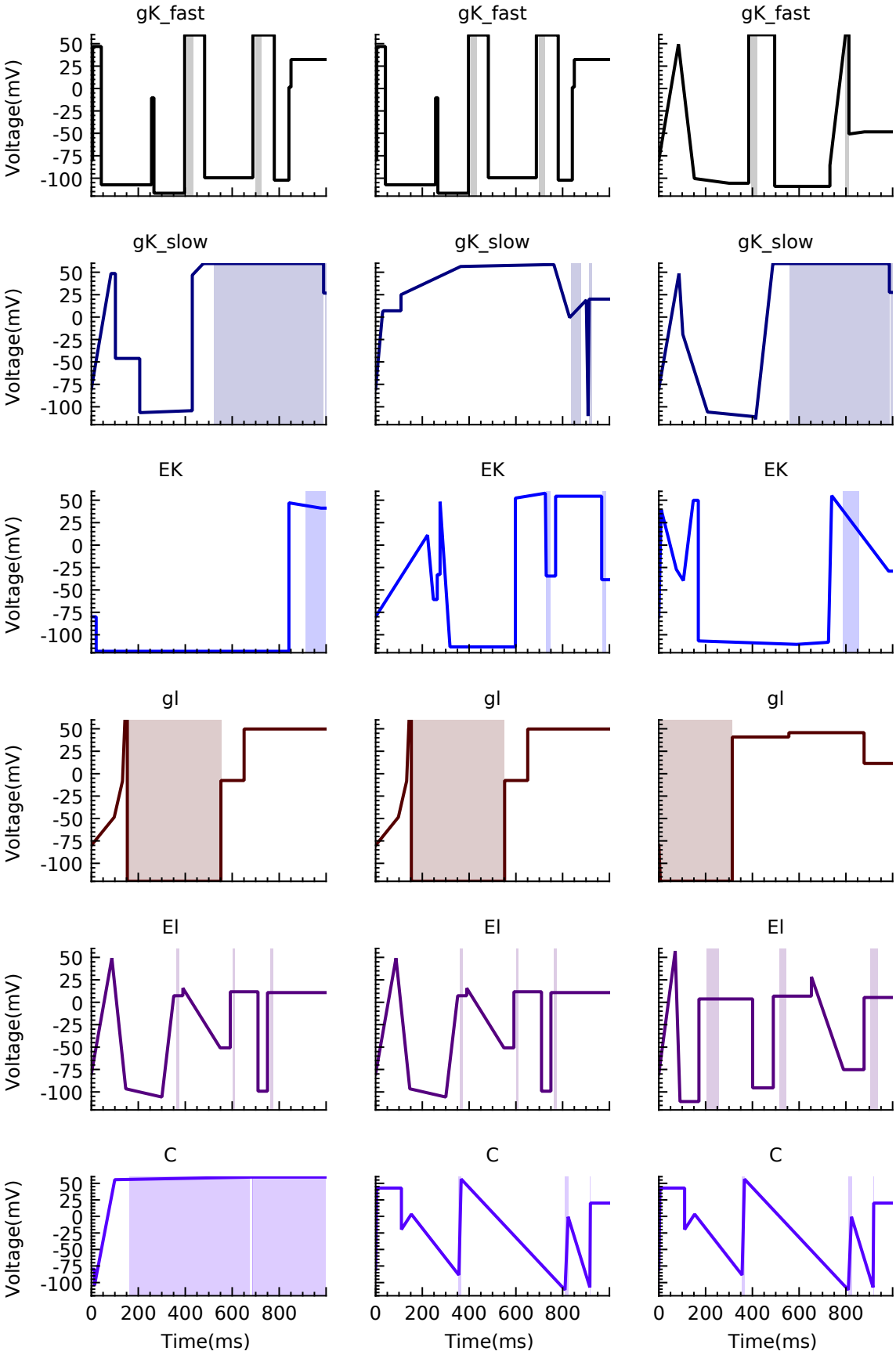


Figure A.7: Kv2.1x stimuli, cluster, weighted

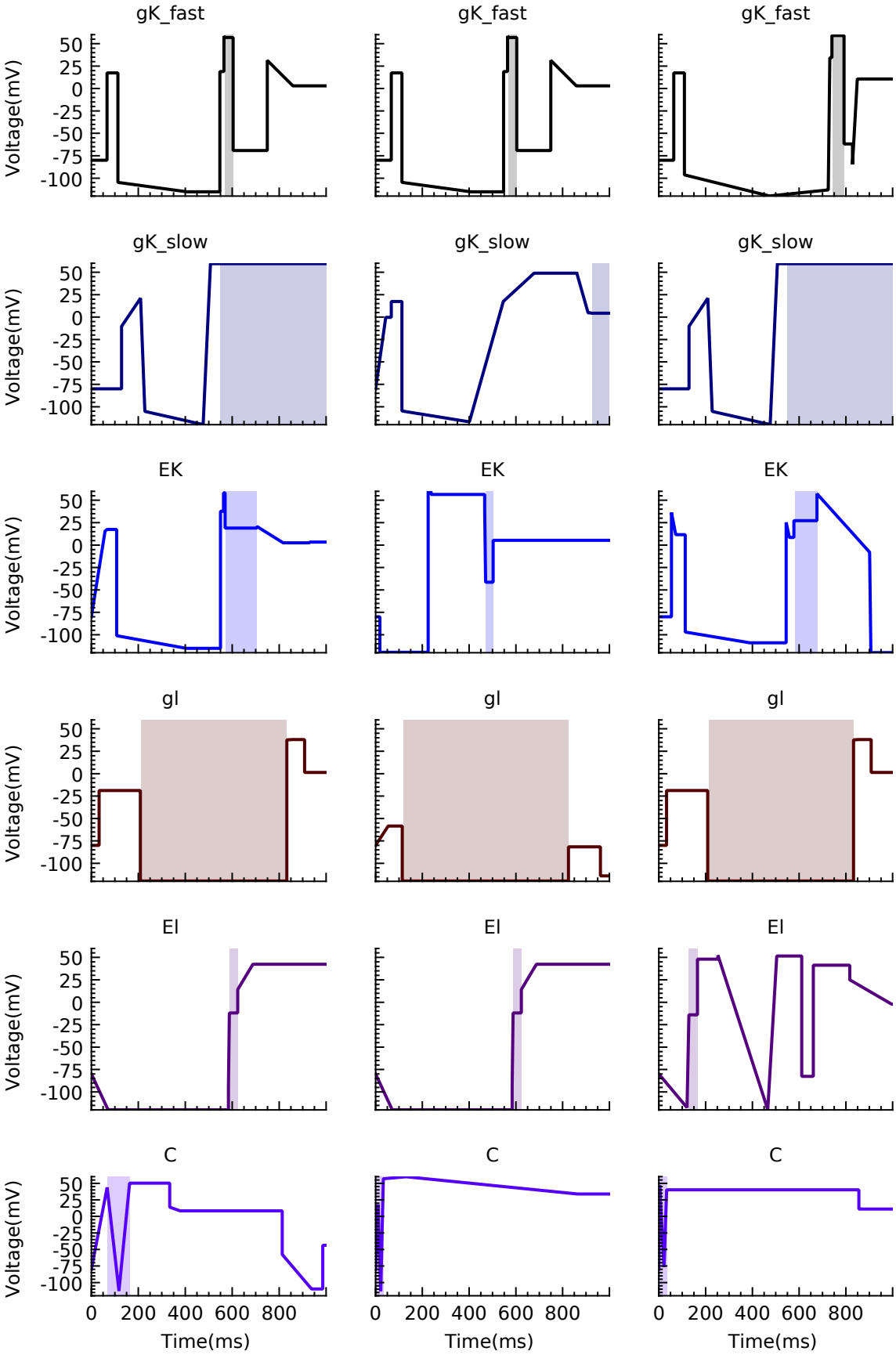


Figure A.8: Kv2.1x stimuli, bubble, weighted

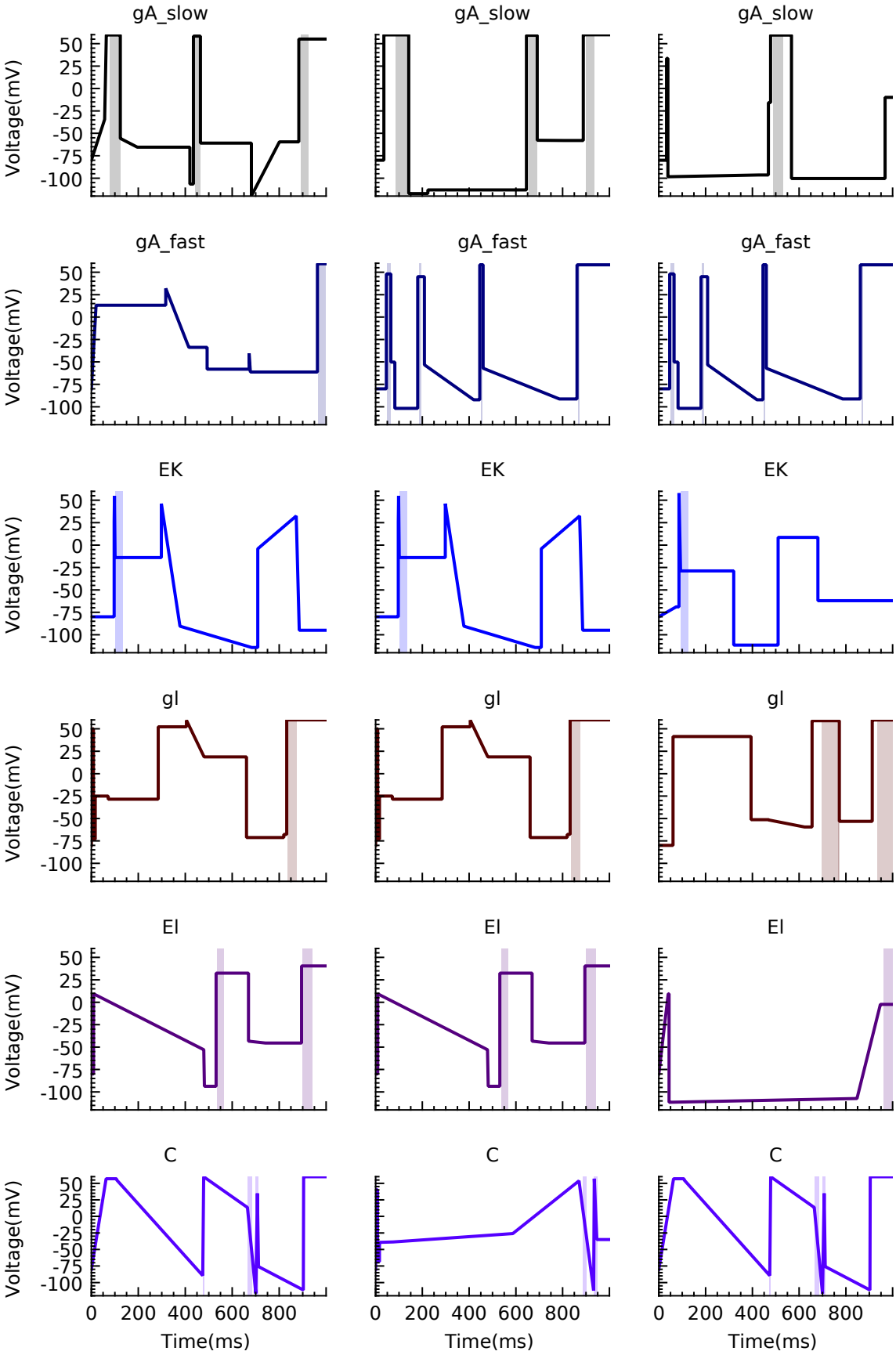


Figure A.9: Kv1.4x stimuli, cluster, left to right: weighted, unweighted, target-only

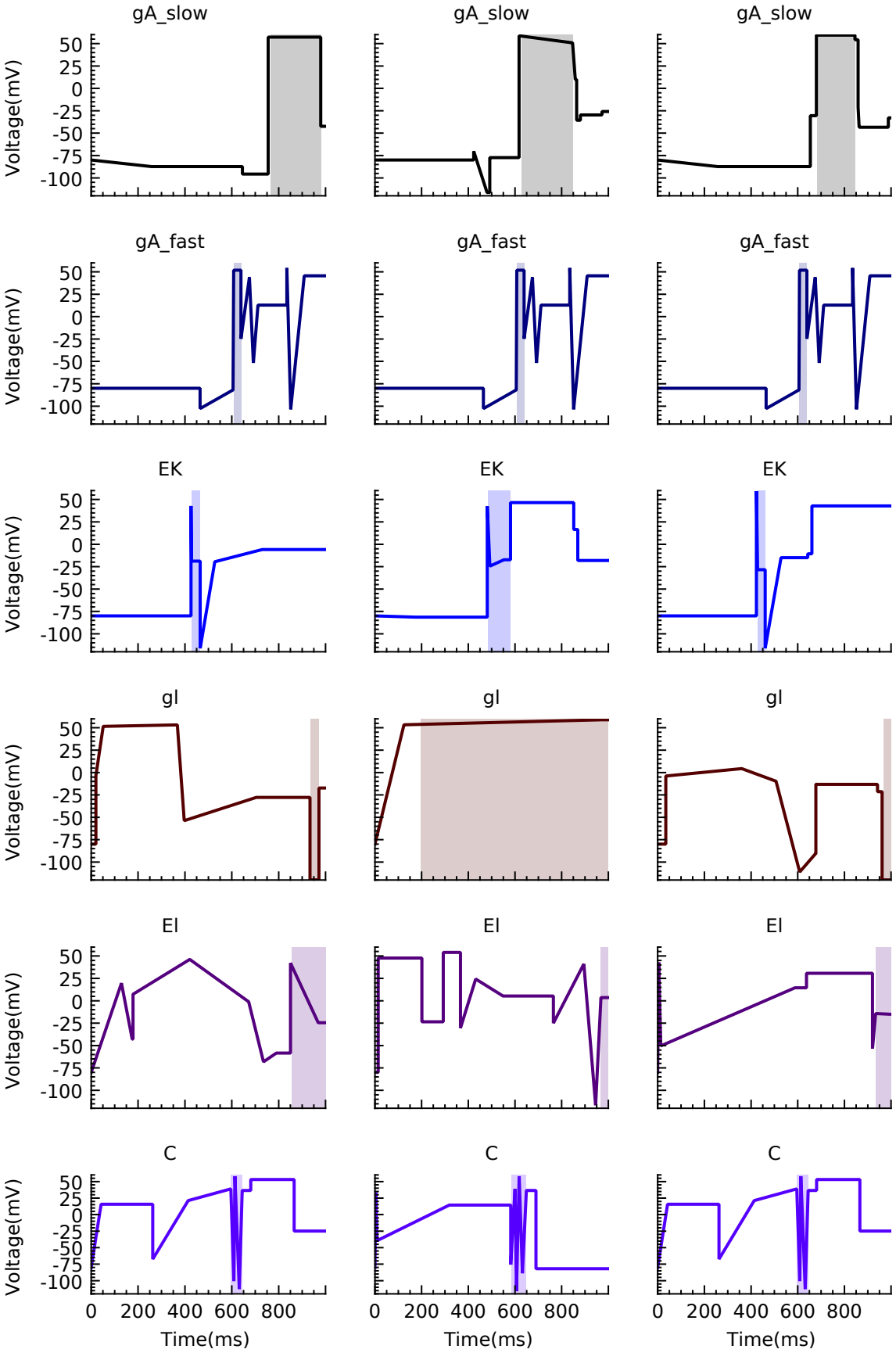


Figure A.10: Kv1.4x stimuli, bubble, left to right: weighted, unweighted, target-only

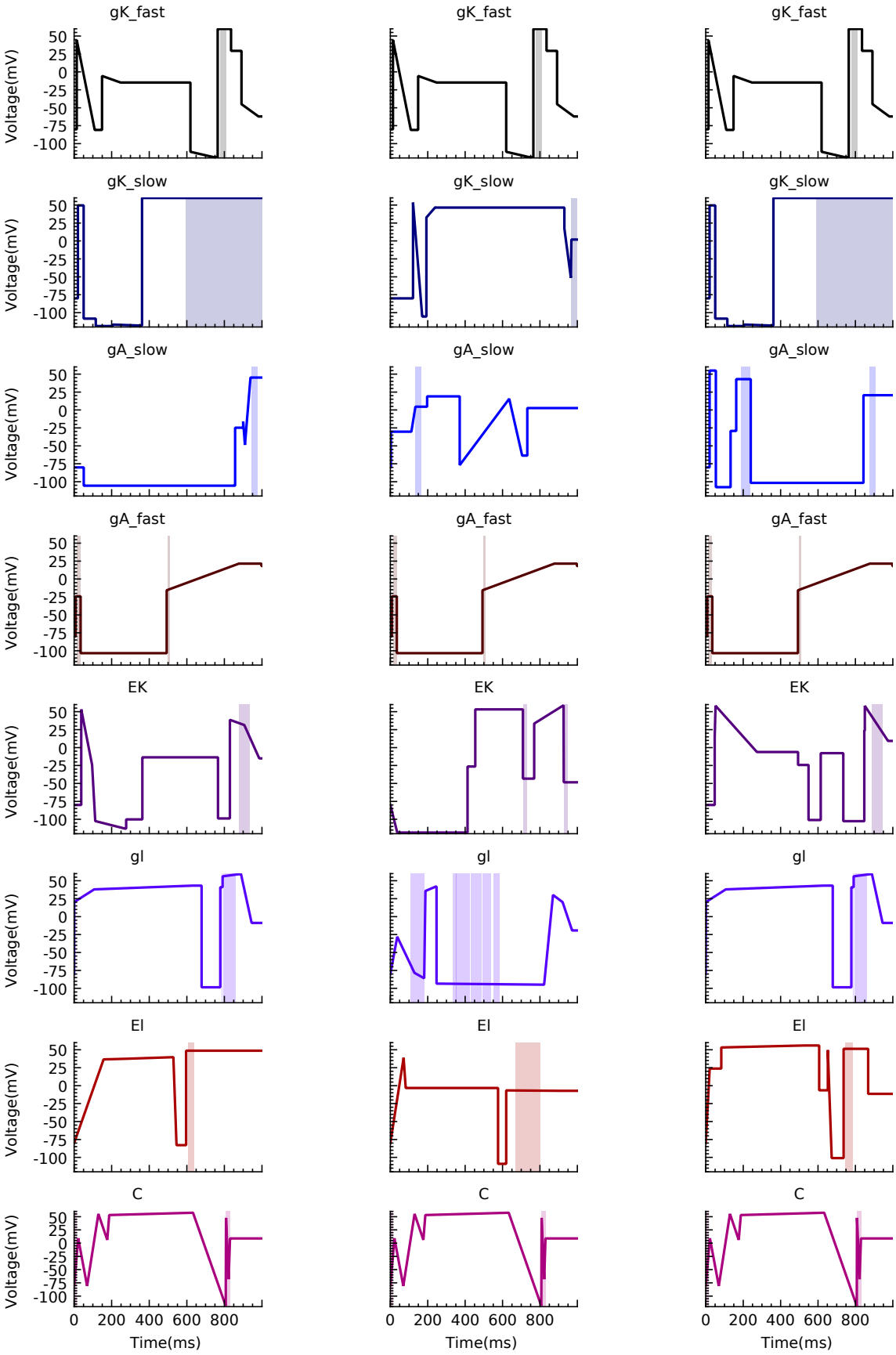


Figure A.11: Kboth stimuli, cluster, weighted

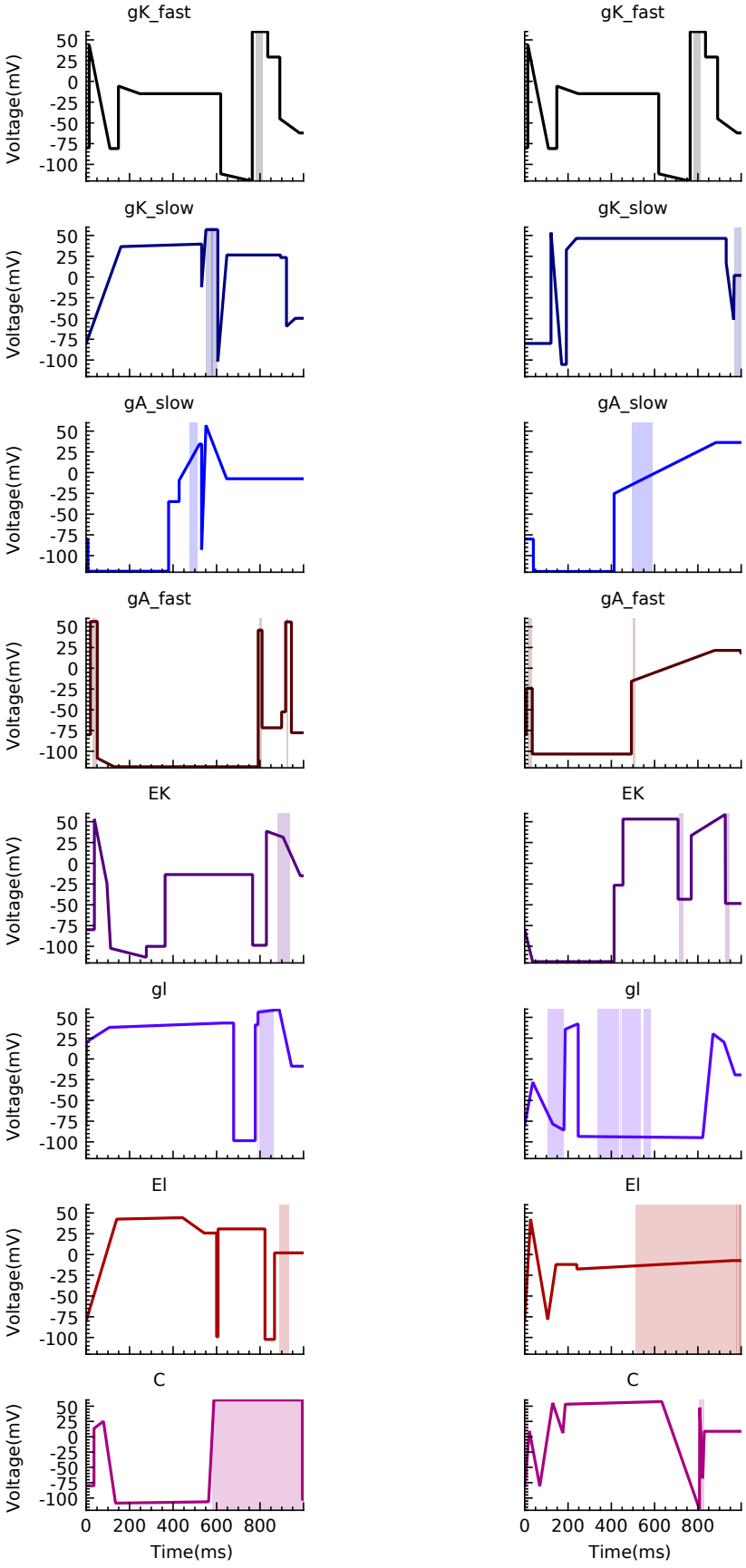


Figure A.12: Kboth stimuli, cluster, left: unweighted, right: target-only

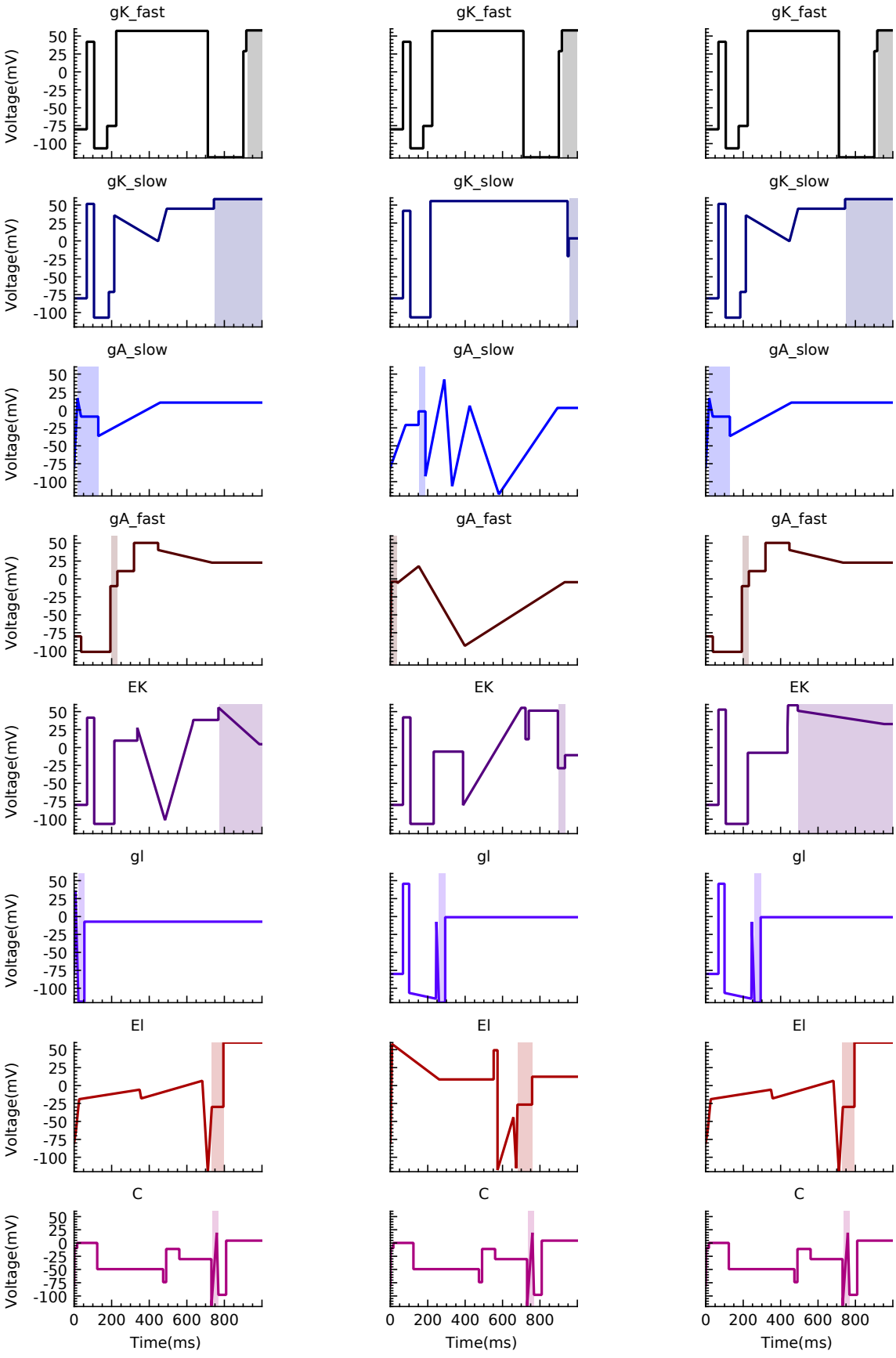


Figure A.13: Kboth stimuli, bubble, weighted

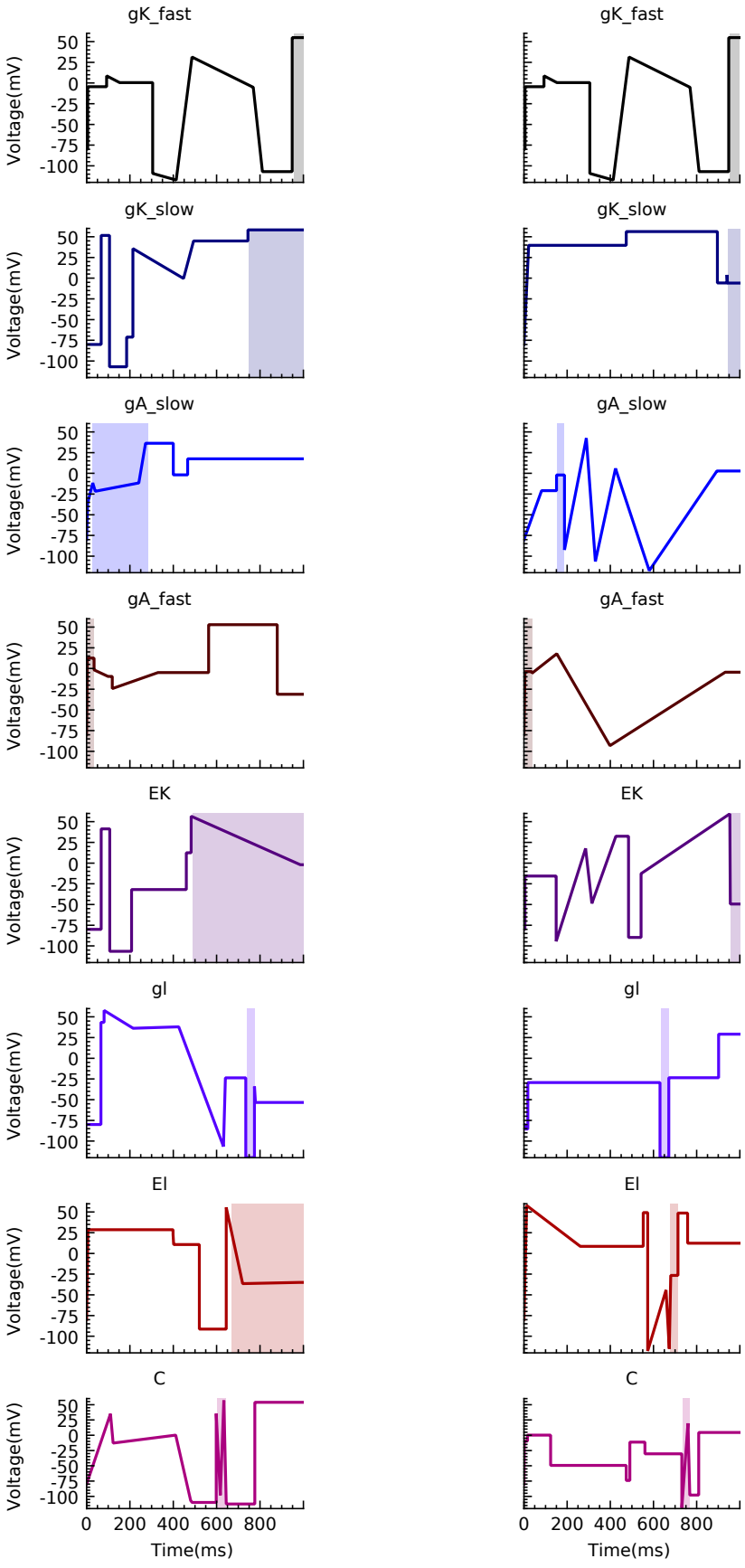


Figure A.14: Kboth stimuli, bubble, left: unweighted, right: target-only

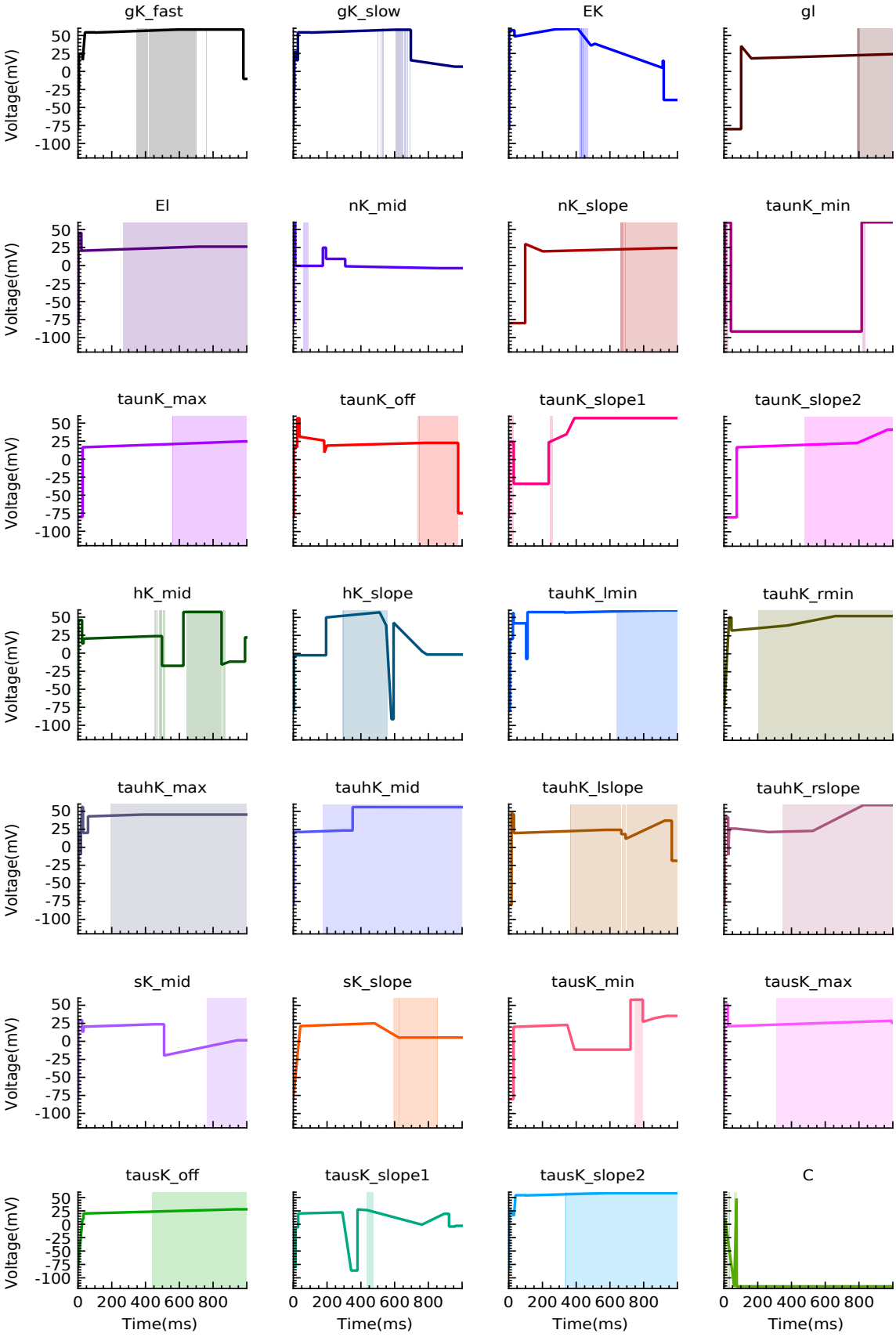


Figure A.15: Kv2.1k stimuli, cluster, weighted

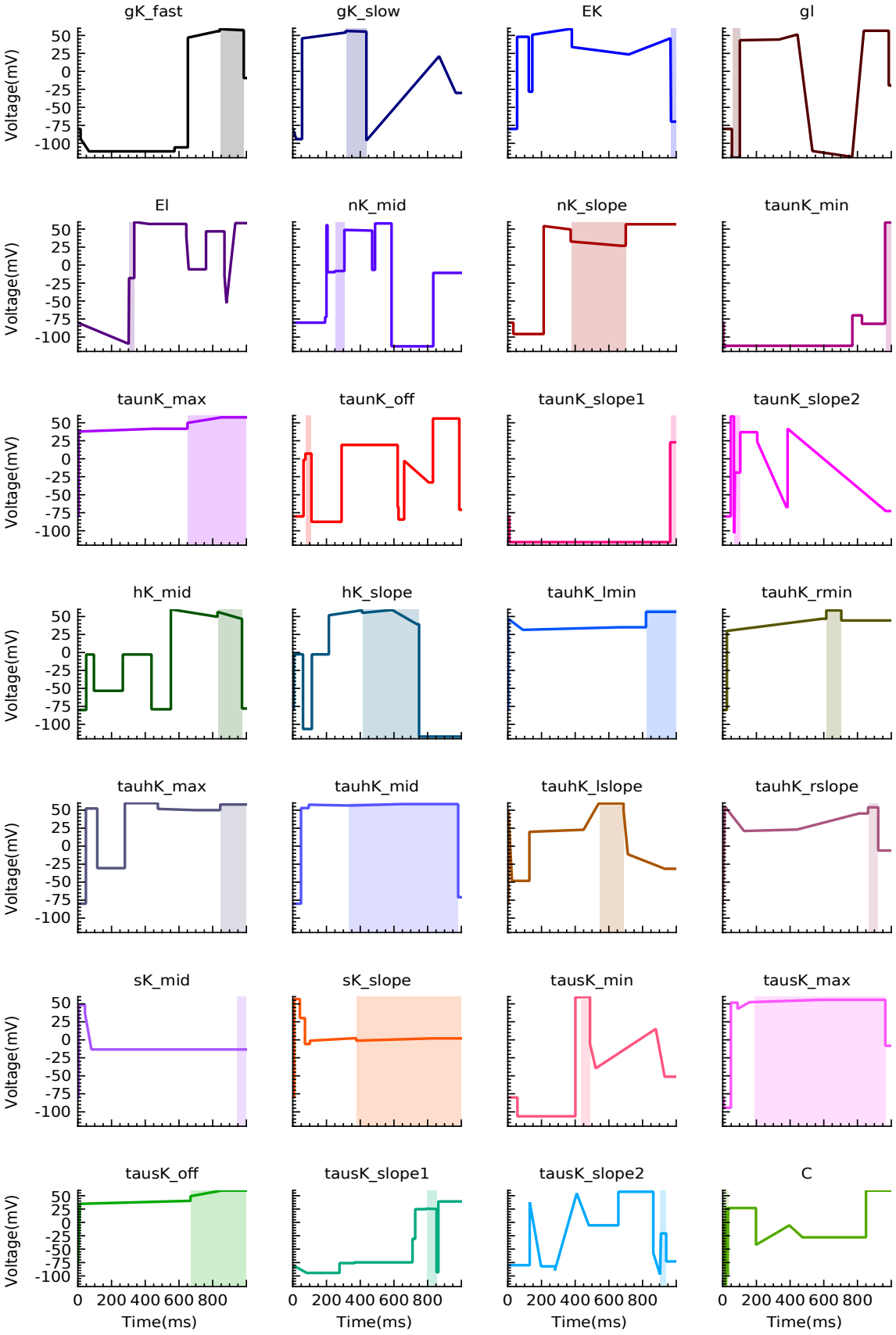


Figure A.16: Kv2.1k stimuli, bubble, weighted