Estimating kinetic mechanisms with prior knowledge II: Behavioral constraints and numerical tests

Marco A. Navarro,* Autoosa Salari,* Mirela Milescu, and Lorin S. Milescu

Division of Biological Sciences, University of Missouri, Columbia, MO

Kinetic mechanisms predict how ion channels and other proteins function at the molecular and cellular levels. Ideally, a kinetic model should explain new data but also be consistent with existing knowledge. In this two-part study, we present a mathematical and computational formalism that can be used to enforce prior knowledge into kinetic models using constraints. Here, we focus on constraints that quantify the behavior of the model under certain conditions, and on constraints that enforce arbitrary parameter relationships. The penalty-based optimization mechanism described here can be used to enforce virtually any model property or behavior, including those that cannot be easily expressed through mathematical relationships. Examples include maximum open probability, use-dependent availability, and nonlinear parameter relationships. We use a simple kinetic mechanism to test multiple sets of constraints that implement linear parameter relationships and arbitrary model properties and behaviors, and we provide numerical examples. This work complements and extends the companion article, where we show how to enforce explicit linear parameter relationships. By incorporating more knowledge into the parameter estimation procedure, it is possible to obtain more realistic and robust models with greater predictive power.

Journal of General Physiology

INTRODUCTION

To understand how ion channels and other proteins function at the molecular and cellular levels, one must decrypt their kinetic mechanism, defined as a set of interconvertible structural conformations, with transitions quantified by rate constants that depend on external variables (e.g., membrane potential, ligand concentration, etc.). Modeling molecular kinetics is not trivial, but sophisticated algorithms have been developed that can extract the rate constants for a given model from a variety of experimental data types, such as single-channel or whole-cell voltage-clamp currents (Colquhoun and Hawkes, 1982; Colquhoun and Sigworth, 1995; Qin et al., 1996, 2000; Venkataramanan and Sigworth, 2002; Colquhoun et al., 2003; Milescu et al., 2005; Csanády, 2006; Stepanyuk et al., 2011, 2014), single-molecule fluorescence (Weiss, 2000; Milescu et al., 2006a,b; Liu et al., 2010), or even current-clamp recordings (Milescu et al., 2008). Automated algorithms that can identify the model itself have also been attempted (Gurkiewicz and Korngreen, 2007; Menon et al., 2009). This abundance of data and analysis algorithms is great, but it raises an important issue: how do we make sure that a model is consistent with all these data, new and old? In other words, how do we extract a model that explains new experimental data but also satisfies existing knowledge?

In the first part of this study (see Salari et al. in this issue), we discussed the general principles of enforcing prior knowledge using model constraints. We identified two main types: parameter constraints and behavioral constraints. Parameter constraints represent explicit mathematical relationships between model parameters, which include the pre-exponential and exponential rate constant factors, allosteric and other multiplicative factors, and any external variables that describe the experimental data and the recording conditions. In part one, we presented a unified mechanism that handles both equality and inequality linear parameter constraints, using relatively simple linear algebra methods that convert the interdependent parameters of the model into a reduced set of independent ("free") parameters that can be passed to the optimization engine. Linear relationships can implement a surprisingly broad range of constraints, particularly after some model parameters are first transformed by the logarithm function. For example, one can scale one rate to another, parameterize allosteric relationships, enforce microscopic reversibility, restrict a parameter to a range of values, etc. Nevertheless, linear parameter constraints are not a universal solution.

We present here a complementary modality for enforcing prior knowledge, which can be used to enforce any type of model behavior, as well as arbitrary parameter relationships. For example, one can fit stationary single-channel data using a maximum likelihood method but simultaneously use constraints to enforce voltage de-

*M.A. Navarro and A. Salari contributed equally to this paper.
Correspondence to Lorin S. Milescu: milescul@missouri.edu

The Rockefeller University Press
J. Gen. Physiol. 2018 Vol. 150 No. 2 339–354
https://doi.org/10.1085/jgp.201711912

Table 1. **Model parameters and properties**

	Parameters									Properties	
	$k^0_{2,1}$ (s^{-1})	$k^1_{2,1}$ (mV^{-1})	$k^0_{2,3}$ (s^{-1})	$k^1_{2,3}$ (mV^{-1})	$k^0_{3,4}$ (s^{-1})	$k^0_{4,3}$ (s^{-1})	$k^1_{4,3}$ (mV^{-1})	a_1	N_C	P_O	f_R
True	100.00	−0.13	5,000.00	0.02	3,000.00	5.00	−0.01	2.0	5,000	0.4175	0.4292
Initial	100.00	−0.075	1,500.00	0.05	1,500.00	20.00	−0.10	3.0	3,000	0.3198	1.0
Run I	77.08	−0.14	4,841.62	0.02	3,102.51	11.75	0.01	1.99	5,073	0.4062	0.1843
Run II	91.18	−0.13	4,460.68	0.02	3,458.43	9.64	0.00	2.22	5,667	0.3723	0.3946
Run III	82.75	−0.13	3,976.55	0.02	4,129.05	7.24	0.00	2.05	6,623[a]	0.3125	0.3543
Run IV	95.39	−0.13	6,056.44	0.02	2,414.50	7.52	0.00	1.90	4,061	0.4992[a]	0.3186
Run V	98.88	−0.13	5,039.69	0.02	3,001.74	2.52	−0.03	1.81	4,919	0.4135	0.7991[a]
Run VI	112.61	−0.13	6,280.71	0.02	2,416.81	1.88	−0.04	1.72	4,055	0.4995[a]	0.7998[a]

The quantities refer to the kinetic model shown in Fig. 2 A. k^0_{ij} and k^1_{ij} represent rate constant parameters, as defined by Eq. 1 from the companion paper (Salari et al., 2018; rate constant $k_{ij} = k^0_{ij} \times e^{k^1_{ij} \times V}$), a_1 is an allosteric factor, and N_C is the channel count. P_o and f_R represent model properties, as defined in Fig. 2 B. The "true" parameter values were used to simulate the data shown in Fig. 3. The "initial" and the "run" values refer to the starting and the ending of optimization, respectively, as plotted in Figs. 4 and 5. Some rate constant parameters are not shown, because they are defined by constraints (e.g., $k^0_{1,2} = a_1 \times k^0_{2,3}$) and can be easily derived.
[a]Optimization runs where model parameters and properties were constrained away from the true values.

pendence or other nonstationary behavior, as obtained from other types of data or from the literature. The basic idea is to calculate for each applied behavioral constraint a penalty that represents the degree by which the model deviates from that constraint. The penalty is then added to the cost function that measures the error between the data and the prediction of the model. As the optimizer minimizes the cost function in search for an optimal solution, it will generate a model that will not only fit the data but will also satisfy all the prior knowledge.

We illustrate here the two types of model constraints (linear parameter constraints and behavioral constraints) and test their respective computational procedures with a simple ion channel modeling example. First, we simulate stochastic macroscopic data in response to a typical voltage-clamp protocol. Then, we fit these data while enforcing different combinations of model constraints. The calculations are explained step by step, and detailed numerical examples are given. All computational procedures were implemented in our freely available software (Milescu, 2015).

MATERIALS AND METHODS

All the mathematical and computational algorithms described in this study, as well as the simulation, data processing, and model optimization, were implemented, tested, and performed with the freely available MLab edition of the QuB program, running under the Microsoft Windows operating system.

Model parameters

To simulate the test data, the model shown in Fig. 2 A was tweaked by hand to generate macroscopic currents resembling voltage-dependent sodium currents (Fig. 3). The simulated data were fitted in multiple runs, with different sets of constraints applied to the model (Fig. 2 B). The model parameter values (true, initial, and estimated) are listed in Table 1.

Stochastic simulations

Ion channel macroscopic traces were simulated stochastically under the voltage-clamp paradigm, using established procedures (Milescu et al., 2005). To approximate the properties of sodium currents, the single-channel conductance was 10 pS and the reversal potential was +60 mV. Random Gaussian noise with zero average and 5-pA standard deviation was added to each trace to approximate whole-cell recording noise. To generate the activation/inactivation time course (Fig. 3 A) and activation/availability steady-state curves (Fig. 3 B), we used a typical voltage-clamp protocol: the channels were first equilibrated at −120 mV and then subjected to a 200-ms conditioning step at potentials ranging from −120 mV to +40 mV, followed by a 50-ms test step at 0 mV. The peak current from each conditioning step was extracted and converted to conductance (assuming a linear relationship), and the obtained values were used to construct the activation curve. Similarly, the peak current from the test step was extracted and used to construct the availability curve. Together, the currents evoked during the first 5 ms of the conditioning step in the −50 mV to +40 mV range (Fig. 3 A) and the activation and availability curves (Fig. 3 B) were used for model optimization.

Model optimization

The algorithms were tested by fitting the data shown in Fig. 3. Optimization trials were run on a dual eight-core 3.3 GHz Intel Xeon processor computer, running Windows 7 (64 bit). Each optimization run took less than 20 min to complete. The model was optimized by minimizing the cost function with a modified version of the Davidon–Fletcher–Powell optimizer (dfpmin; Fletcher and Powell, 1963; Press et al., 1992). For efficiency, the cost function was coded for parallel computation. The cost function was calculated as the sum of square errors between the data and the prediction of the model, normalized to the total number of

points, plus a penalty term for those optimization trials involving a behavioral constraint, as detailed in Results. The gradients of the cost function with respect to the free parameters were calculated numerically. The prediction of the model for a given set of parameters was obtained by simulating the deterministic response of the model to the same stimulation protocols as used for simulation. Then, the resulting traces were processed to extract the time course and the activation and availability curves, following the same procedure as for the simulated test data (Milescu et al., 2010; Salari et al., 2016).

RESULTS

Implementing prior knowledge with model constraints

A theoretical background was given in part one of this study (Salari et al., 2018), where we briefly introduced a few prerequisite topics (kinetic mechanisms, model topology, and parameter estimation). Then, we presented a mathematical formalism and computational procedure for enforcing linear parameter constraints. Here, in part two, we continue with a mechanism for enforcing model behavior and arbitrary parameter relationships. Then, we give a step-by-step numerical example that illustrates the implementation of all types of constraints. Because we will make multiple references to the first part, the equations introduced here are numbered in continuation of those in part one. The mathematical symbols that are not explicitly defined here have been introduced in part one.

Behavioral constraints and arbitrary parameter relationships. A good amount of prior knowledge about the channel can be expressed as linear relationships between model parameters, resulting in constraints that can be handled with relatively straightforward linear algebra methods. However, some channel behaviors and properties cannot be easily formulated as explicit functions of model parameters or they need nonlinear functions that are not so easily tractable. For voltage-gated channels, examples of important functional behavior include the open probability (P_O), the voltage dependence of activation or inactivation, or the use-dependent availability. These properties cannot be easily formulated as functions of rate constants, except for very simple kinetic mechanisms. Furthermore, they must be prescribed in the context of a specific experiment (e.g., a voltage-clamp step protocol).

Expanding the cost function. Without explicit parameter relationships, we cannot solve behavioral constraints simply by converting model parameters into free parameters, as we did for linear parameter constraints. Likewise, we cannot use that formalism to solve any parameter constraint that cannot be written as a linear relationship between the transformed model parameters, as captured by the generalized linear constraint Eqs. 32 and 33 (Salari et al., 2018). Instead, the solution we propose here for handling behavioral constraints and arbitrary parameter relationships is to include them into the cost function. Thus, the cost function F, which is minimized by the optimizer in search for an optimal solution, can be expanded to include multiple components, one for each set of experimental data and one for each constraint:

$$F = \sum_i \left(\alpha_i^Y \times F_i^Y\right) + \sum_j \left(\alpha_j^C \times F_j^C\right), \tag{60}$$

where F_i^Y represents the cost of data component i and F_j^C represents the cost of behavioral constraint j. The α quantities are relative weighting factors that multiply the cost function components. Including multiple components in the cost function is known in the optimization literature as multi-objective fitting (Druckmann et al., 2007; Bandyopadhyay and Saha, 2013; Fletcher, 2013). For example, F_i^Y could stand for newly acquired voltage-clamp data (e.g., the time course of activation and inactivation at different membrane potentials), whereas F_j^C could be data from the literature (e.g., steady-state activation and inactivation curves) or a hypothesized property (e.g., the open probability P_O). The cost function components that denote constraints should be formulated in such a way that they take a value of zero when the underlying constraint is satisfied, and a very large value (relative to the data cost components) when the constraint fails, as explained further.

Formulating behavioral constraints and arbitrary parameter relationships. Some behavioral constraints can be formulated as mathematical relationships involving simple properties of the channel. For example, we could constrain the maximum open probability reached during a depolarization step to take certain values or to fall within a range:

$$\begin{aligned} &P_O = 0.5, \text{ or} \\ &P_O \le 0.4, \text{ or} \\ &0.3 \le P_O \le 0.7. \end{aligned} \tag{61}$$

An example of a parameter constraint that cannot be processed with the formalism developed in part one is restricting a rate constant pre-exponential factor k_{ij}^0 to a range of values:

$$1{,}000 \le k_{ij}^0 \le 10{,}000. \tag{62}$$

This range constraint cannot be handled as two linear inequality relationships, because they would be mathematically redundant, where both cannot be simultaneously satisfied. Another example is parameterizing an exponential factor k_{ij}^1 as a product of more than one variable:

$$k_{ij}^{1} = C \times a \times b, \tag{63}$$

where a and b could stand for the δ_{ij} and z_{ij} parameters in Eq. 2 (Salari et al., 2018) and C is a constant equal to $F/(R \times T)$. Likewise, this equation cannot be handled with the formalism from part one because it is nonlinear.

Algebraically, any equality or inequality relationship can be converted to "=0" or "≥0," respectively. Thus, the above constraints could be rewritten as follows:

$$\begin{aligned} &P_O - 0.5 = 0, \\ &0.4 - P_O \geq 0, \\ &\left\{ \begin{matrix} P_O - 0.3 \geq 0 \\ 0.7 - P_O \geq 0 \end{matrix} \right\}, \\ &\left\{ \begin{matrix} k_{ij}^{0} - 1{,}000 \geq 0 \\ 10{,}000 - k_{ij}^{0} \geq 0 \end{matrix} \right\}, \\ &k_{ij}^{1} - C \times a \times b = 0. \end{aligned} \tag{64}$$

The cost function components F^C that correspond to the above equality and inequality relationships can be formulated as follows:

$$\begin{aligned} F^C &= \alpha \times (P_O - 0.5)^2, \\ F^C &= \alpha \times (0.4 - P_O)^2, \\ F^C &= \alpha \times \left[(P_O - 0.3)^2 + (0.7 - P_O)^2 \right], \\ F^C &= \alpha \times \left[\left(k_{ij}^{0} - 1{,}000 \right)^2 + \left(10{,}000 - k_{ij}^{0} \right)^2 \right], \text{ and} \\ F^C &= \alpha \times \left(k_{ij}^{1} - C \times a \times b \right)^2, \end{aligned} \tag{65}$$

where α is a weighting factor with the following properties:

$$\alpha > 0, \text{ for equality constraints,} \tag{66}$$

and

$$\left\{ \begin{matrix} \alpha = 0 \text{ if constraint} \geq 0 \\ \alpha > 0 \text{ if constraint} < 0 \end{matrix} \right\}, \text{ for inequality constraints,} \tag{67}$$

where "constraint" refers to the left-side term of a constraint equation (Eq. 37; Salari et al., 2018). Thus, these cost function components are equal to zero when the underlying constraints are exactly satisfied but take a positive and quadratically increasing value when the constraint relationships are not satisfied.

Nonparametric behavioral constraints. In principle, the same logic can be applied to any other model property. However, some model behaviors cannot be reduced to a single value or cannot be easily calculated theoretically. For example, many functional aspects, such as the recovery from inactivation or the use dependence, can be empirically fitted by one or two exponentials. Unfortunately, these apparent time constants cannot be directly and easily calculated from the model, which actually predicts a larger number of exponentials, equal to the state count minus one (Milescu et al., 2005; Salari et al., 2016). Likewise, the voltage-dependent activation curve can be well approximated and fitted by a Boltzmann equation with only two parameters, but calculating the half-activation and the sensitivity values directly from the model is generally not practical.

In cases like these, it is simpler to simulate the response of the channel to the same stimulation protocol as was used to obtain the experimental (or hypothesized) data. Then, a cost function component can be calculated as the sum of square differences between the simulated and the experimental data:

$$F^C = \alpha \times \frac{1}{N} \sum_i (y_i - x_i)^2, \tag{68}$$

where y_i and x_i are experimental and simulated data points, respectively, and N is the number of data points. In the above equation, one could use the raw data directly, point by point, or one could extract some properties from the raw data and use the points on that property curve. For example, when the stimulation protocol is designed to extract the time course of a macroscopic current, one would fit the raw data directly. In contrast, when the stimulation protocol is designed to extract a behavior, such as the recovery from inactivation, one would fit the property curve. Although extracting a property curve involves additional computation, it has the substantial benefit of concentrating the information on a very specific aspect of channel behavior. For example, in a curve that represents the recovery from inactivation, every data point informs directly on the apparent time constants of inactivation. Likewise, every data point in a voltage-dependent activation curve informs directly on the two parameters of the Boltzmann equation.

Whether the cost function for these nonparametric behavioral constraints is calculated from raw data or from property curves or is based on hypothetical values, one must consider the presence of random noise and other artifacts that contaminate the experimental data. Thus, even a perfect model would not generate zero cost for the constraints, which may confuse the optimization engine. A simple solution is to reformulate the problem as an inequality:

$$\frac{1}{N} \sum_i (y_i - x_i)^2 \leq \varepsilon, \tag{69}$$

where ε is a positive constant proportional to the noise content of the data. Then, the cost function component can be written as follows:

$$F^C = \alpha \times \left[\varepsilon - \frac{1}{N} \sum_i (y_i - x_i)^2 \right]^2, \tag{70}$$

where α is a weighting factor with the following property:

$$\left\{\begin{matrix}\alpha = 0 \ \text{ if} \frac{1}{N}\sum_i (y_i - x_i)^2 \le \varepsilon \\ \alpha > 0 \ \text{ if} \frac{1}{N}\sum_i (y_i - x_i)^2 > \varepsilon\end{matrix}\right\}. \quad (71)$$

Thus, if the sum of square errors between the simulated and the experimental data is less than ε, then the underlying constraint is considered to be satisfied. In other words, the model only needs to explain the constraining data components "well enough," as warranted by the inevitable noise and artifacts.

A computational framework for solving behavioral constraints and arbitrary parameter relationships. We presented above some examples of behavioral constraints and arbitrary parameter relationships. In general, the problem we must solve is to find a model that not only best explains the experimental data but also satisfies a set of equality or inequality constraints. Mathematically, the problem can be formulated as the minimization of a function subject to a set of nonlinear constraints:

$$\begin{aligned}&\text{minimize } F(\bar{\mathbf{X}}) \\ &\text{such that: } h_i(\bar{\mathbf{X}}) = 0,\ i = 1 \ldots N_E \\ &\qquad\qquad g_j(\bar{\mathbf{X}}) \ge 0,\ j = 1 \ldots N_I,\end{aligned} \quad (72)$$

where $\bar{\mathbf{X}}$ and $F(\bar{\mathbf{X}})$ are the vector of free parameters and the cost function, respectively, as defined in part one, and $h_i(\bar{\mathbf{X}})$ and $g_j(\bar{\mathbf{X}})$ are two sets of N_E equality and N_I inequality constraints, respectively. In the case of maximum likelihood methods, instead of maximizing the log-likelihood, one can equivalently minimize its negative.

As discussed in the previous section, one possible solution to this constrained function minimization is to add the constraints to the cost function (Eq. 60). This approach is equivalent to the method of penalties (Fletcher, 2013), which reformulates the problem as an unconstrained optimization, by adding a penalty term to the cost function $F(\bar{\mathbf{X}})$. Thus, the objective becomes minimizing a penalized cost function $F'(\bar{\mathbf{X}}, \alpha)$:

$$F'(\bar{\mathbf{X}}, \alpha) = F(\bar{\mathbf{X}}) + \alpha \times \sum_i \left[h_i(\bar{\mathbf{X}})\right]^2 + \sum_j \left\{\beta_j \times \left[g_j(\bar{\mathbf{X}})\right]^2\right\}, \quad (73)$$

where α and β are penalty factors with the following properties:

$$\begin{aligned}&\alpha > 0, \\ &\beta_j = \left\{\begin{matrix}0 \ \text{ if } g_j(\bar{\mathbf{X}}) \ge 0 \\ \alpha \ \text{ if } g_j(\bar{\mathbf{X}}) < 0\end{matrix}\right\}.\end{aligned} \quad (74)$$

Formulating the h_i and g_j expressions that correspond to any of the equality and inequality constraints above is straightforward. For example, the first two P_O constraints given above (Eq. 64) become:

$$\begin{aligned}h_i &= P_O - 0.5, \\ g_j &= 0.4 - P_O.\end{aligned} \quad (75)$$

The gradients of the penalized cost function might be required by the optimization engine. These could be calculated analytically, as follows:

$$\frac{\partial F'(\bar{\mathbf{X}}, \alpha)}{\partial \bar{x}_k} = \frac{\partial F(\bar{\mathbf{X}})}{\partial \bar{x}_k} + 2 \times \left\{\begin{matrix}\alpha \times \sum_i \left[h_i(\bar{\mathbf{X}}) \times \frac{\partial h_i(\bar{\mathbf{X}})}{\partial \bar{x}_k}\right] + \\ \sum_j \left[\beta_j \times g_j(\bar{\mathbf{X}}) \times \frac{\partial g_j(\bar{\mathbf{X}})}{\partial \bar{x}_k}\right]\end{matrix}\right\}. \quad (76)$$

The derivatives of h_i and g_j with respect to a free parameter $\bar{x}_k$ depend on the specific constraints used, and one may need to calculate them using the chain differentiation rule, as in the case of linear constraints. Ultimately, if the constraint functions are too complicated, the gradients can be approximated numerically. Whether the gradients are calculated analytically or numerically, one should keep in mind that inequality penalties are only semidifferentiable and may throw off the optimizer. If this is the case, then one possibility is to approximate the penalty into a differentiable function (Bertsekas, 1975). The variance of the estimates can also be calculated using the procedure described in part one.

The main advantage of the penalty method is that it can be used with any optimization algorithm that was originally designed for nonconstrained problems. The main issue, however, is the choice of the penalty parameter α. On the one hand, if α is too small, then the solution found by the optimizer will be pulled toward $F(\bar{\mathbf{X}})$ and the constraints defined by $h_i(\bar{\mathbf{X}})$ and $g_j(\bar{\mathbf{X}})$ may not be exactly satisfied. On the other hand, if α is very large, then the solution will satisfy the constraints (in principle). However, the optimizer engine may have a difficult time finding that solution, because the penalized cost function $F'(\bar{\mathbf{X}}, \alpha)$ may change very abruptly in the *n*-dimensional parameter space. Thus, although it is conceptually and computationally very simple, using the penalized cost function is not exactly a plug-and-play solution, as in the case of linear constraints.

A possible strategy is to find the solution iteratively, starting with a relatively small α, and increasing it until some convergence criteria are satisfied (Himmelblau, 1972). This is the approach we are taking here, as summarized in Fig. 1. Once a model topology is chosen, the workflow starts with defining the linear parameter constraints and the behavioral constraints (if any), including any other arbitrary parameter constraints. The next step is to define the cost function and the penalized cost function, according to the specific application (e.g., macroscopic fitting, single-channel maximum likelihood, etc.). Then, we choose a set of model parameters as the starting point, $\mathbf{K}_0$. Mathematically, these parameters do not need to satisfy either set of constraints (parameter or behavioral), but starting as close as possible is recommended. From the initial model parameters $\mathbf{K}_0$, we then calculate the initial set of free parameters, $\bar{\mathbf{X}}_0$. Finally, we initialize the penalty

parameter α to α_0, equal to a small positive number. In practice, this value can be chosen so as to make the data and the penalty components of the overall cost function be approximately equal.

Once these quantities are defined and initialized, we start the optimization procedure, which involves two embedded loops, as shown in Fig. 1. An outer loop, indexed by *p,* handles the schedule for updating the penalty parameter α_p that is used to calculate the penalized cost function $F'(\bar{\mathbf{X}}, \alpha_p)$, and an inner loop, indexed by *k,* handles model optimization for a given α_p. The penalty parameter α_p is progressively increased at each outer loop iteration, to increase the relative weight of the penalty component in $F'(\bar{\mathbf{X}}, \alpha_p)$. Thus, the behavioral constraints may be only loosely satisfied at the end of the first outer loop iteration, but they will get tighter each time α_p is increased. The outer loop can be run for a predefined number of iterations or can be terminated when the behavioral constraints are satisfied, if at all possible.

In principle, any type of optimization engine can be used in the inner loop. As explained, the optimizer is completely model and penalty blind. Essentially, the optimizer solves an unconstrained minimization problem, operating with a set of free parameters $\bar{\mathbf{X}}$. However, as it explores the free parameter space in search for a minimum, the optimizer will require, for a given $\bar{\mathbf{X}}_k$, the penalized cost function $F'(\bar{\mathbf{X}}, \alpha_p)$ and possibly its gradients. For this, the transformed model parameters $\mathbf{R}_k$ are calculated from $\bar{\mathbf{X}}_k$, and then the model parameters $\mathbf{K}_k$ are calculated from $\mathbf{R}_k$, as outlined in Fig. 3 of the companion article (Salari et al., 2018). The model optimization in the inner loop can be run for a predefined number of iterations or can be terminated when some convergence criteria are satisfied. Typically, convergence requires that there be no substantial changes in the free parameter values and in the cost function (and its gradients be close to zero) from one iteration to the next.

Testing the algorithm

To clarify the computational procedures described in both parts of this study, we give a step-by-step numerical example. For illustration purposes, we chose the model shown in Fig. 2 A, which is complex enough to accommodate an allosteric factor (Fig. 2 A, a_1), an external parameter representing the number of active channels in the recording (N_C), and several parameter and behavioral constraints. At the same time, the model is small enough to allow us to print the vectors and matrices used in the numerical computation. Readers who wish to implement their own code can use these examples for verification. Briefly, we tested the algorithms by fitting a stochastically simulated set of macroscopic data, generated in response to a typical voltage-clamp step protocol. We intentionally chose a relatively small dataset (the time course of activation and inactivation

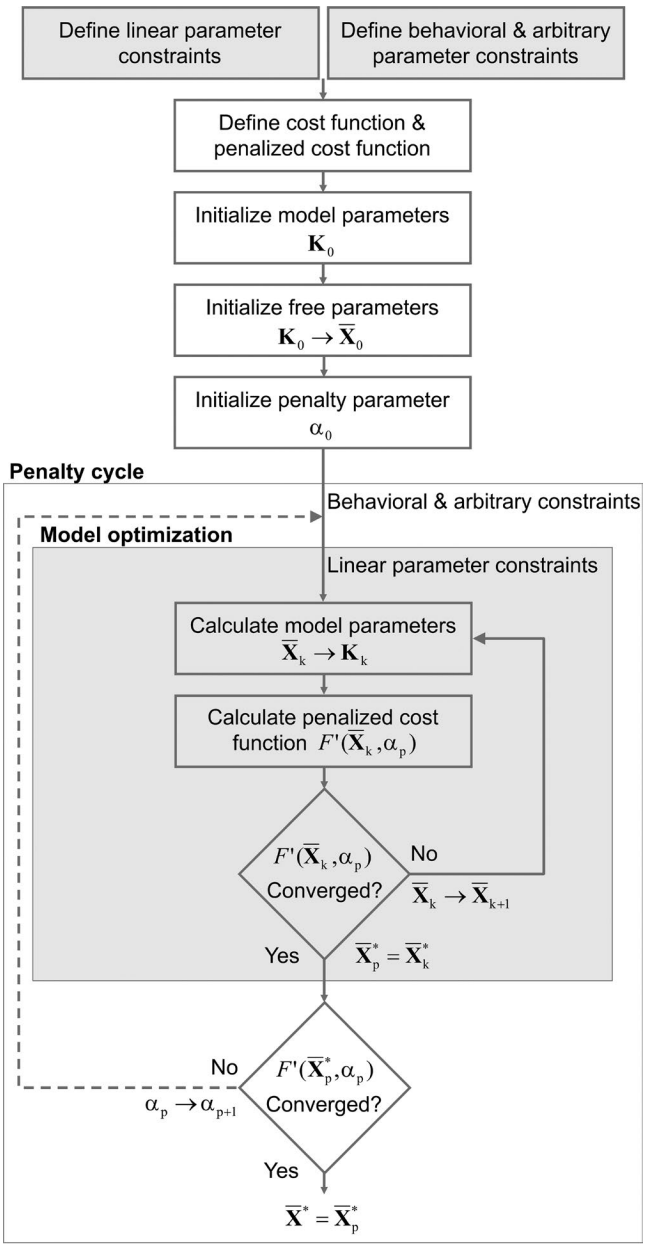

Figure 1. **Optimizing a constrained model.** The flowchart summarizes the computational steps needed to optimize a kinetic model, subject to parameter and behavioral constraints. Linear parameter constraints are implemented via linear algebra transformations between the model parameters **K** and the free parameters $\bar{\mathbf{X}}$, whereas behavioral constraints or arbitrary parameter relationships are handled by a penalized cost function $F'(\bar{\mathbf{X}}_k, \alpha_p)$ that measures the overall error of the model relative to the data and the constraints. The $\mathbf{K} \rightarrow \bar{\mathbf{X}}$ and $\bar{\mathbf{X}} \rightarrow \mathbf{K}$ transformations are detailed in Fig. 3 in the companion paper (Salari et al., 2018). To calculate the cost function, one needs to generate the response of the model (e.g., probability distributions and macroscopic currents) to the same stimulation protocols used to generate the experimental data and formulate the behavioral constraints. The inner computational loop, indexed by *k*, optimizes the model for a given penalty factor α_p, whereas the outer loop, indexed by *p*, gradually increases α_p, to more tightly satisfy the behavioral and arbitrary parameter constraints.

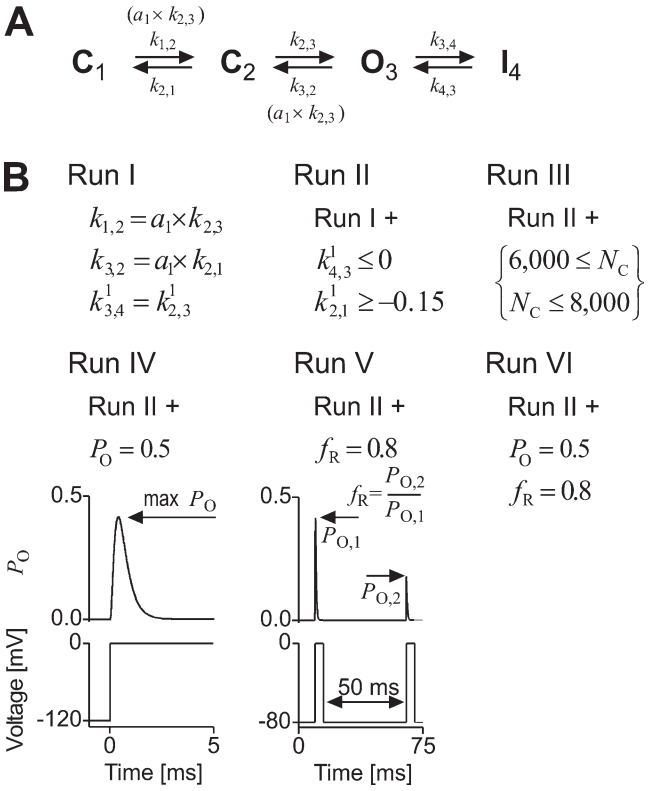

Figure 2. **A test model with different sets of constraints. (A)** A simple kinetic mechanism that generates voltage-gated sodium channel-like currents (see Fig. 3). All rate constants are as described by Eq. 1 in the companion paper (Salari et al., 2018; $k_{ij} = k^0_{ij} \times e^{k^1_{ij} \times V}$); a_1 is an allosteric factor, and N_C is the number of channels. **(B)** Six sets of constraints were applied to the model to test the algorithms (see Figs. 4 and 5). Runs I and II test linear parameter constraints implemented with linear algebra–based methods that convert model parameters into free parameters, and vice versa. Run I implements only linear relationships, whereas run II adds two inequalities. Runs III through VI test arbitrary parameter constraints and behavioral constraints implemented with the penalty mechanism. Run III tests a parameter range constraint, whereas runs IV through VI test constraints that enforce model properties and behavior: the maximum open probability during a depolarization step (P_O, run IV and VI) and the recovered fraction of available channels at 50 ms after a 5-ms inactivation step (f_R, runs V and VI). The P_O and f_R quantities are obtained as shown.

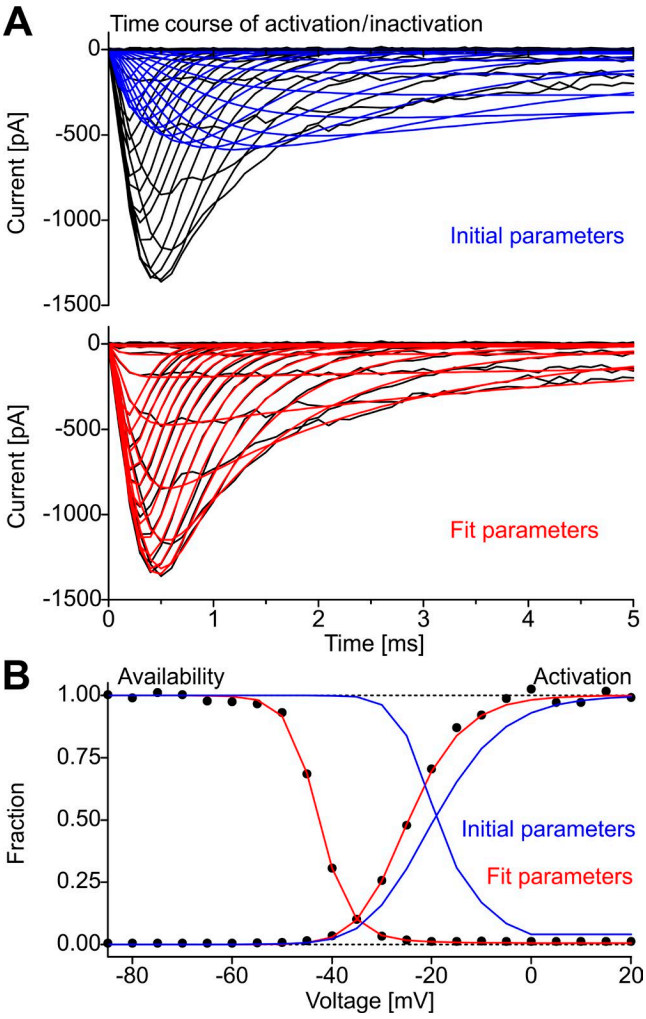

Figure 3. **Test data and model predictions. (A and B)** Whole-cell currents were simulated stochastically with the test model in Fig. 2 A, using a standard activation/inactivation protocol. The data were processed to extract the time course of activation/inactivation (black traces in A) and the steady-state activation and availability curves (black symbols in B). The time course and steady-state curves were fitted together (see Fig. 4). The predictions of the model at the beginning and at the end of optimization are shown by the blue and red traces, respectively. The fit curves correspond to run I in Fig. 2 B, but all runs resulted in virtually identical fits. The true, initial, and estimated parameters and properties of the model are shown in Table 1.

at different voltages and the voltage-dependent steady-state activation and inactivation curves, as shown in Fig. 3) to illustrate potential parameter identifiability issues and the effect of constraints. The data were fitted in multiple runs, with each run enforcing a different set of constraints, as outlined in Fig. 2 B. The simulation, data analysis, and fitting procedures are explained in Materials and methods.

We define the following set of model parameters **K**:

$$\mathbf{K} = \left\{k^0_{1,2}, k^1_{1,2}, k^0_{2,1}, k^1_{2,1}, k^0_{2,3}, k^1_{2,3}, k^0_{3,2}, k^1_{3,2}, k^0_{3,4}, k^1_{3,4}, k^0_{4,3}, k^1_{4,3}, a_1, q_1\right\}, \quad (77)$$

where a_1 is the allosteric factor and q_1 is the channel count. Thus, we have a total of 14 model parameters, with numerical values given in Table 1. The corresponding vector of transformed model parameters **R** is:

$$\mathbf{R} = \left\{\varepsilon^0_{1,2}, k^1_{1,2}, \varepsilon^0_{2,1}, k^1_{2,1}, \varepsilon^0_{2,3}, k^1_{2,3}, \varepsilon^0_{3,2}, k^1_{3,2}, \varepsilon^0_{3,4}, k^1_{3,4}, \varepsilon^0_{4,3}, k^1_{4,3}, \phi_1, \varphi_1\right\}, \quad (78)$$

where

$$\begin{aligned}\varepsilon^0_{ij} &= \ln(k^0_{ij}),\\ \phi_1 &= \ln(a_1),\\ \varphi_1 &= \ln(q_1).\end{aligned} \tag{79}$$

Applying linear parameter constraints. The test model has allosteric relationships that require two sets of linear parameter constraints. The first set applies to the forward transitions C_1 to C_2 and C_2 to C_3:

$$k_{1,2} = a_1 \times k_{2,3}. \tag{80}$$

As explained in part one (Salari et al., 2018), we apply the logarithm on both sides of Eq. 80 and obtain a set of two equality relationships:

$$\left\{\begin{aligned}\ln(k^0_{1,2}) &= \ln(a_1) + \ln(k^0_{2,3})\\ k^1_{1,2} &= k^1_{2,3}\end{aligned}\right\}. \tag{81}$$

The backward transitions C_3 to C_2 and C_2 to C_1 have a similar allosteric relationship, which results in another set of two equality relationships:

$$\left\{\begin{aligned}\ln(k^0_{3,2}) &= \ln(a_1) + \ln(k^0_{2,1})\\ k^1_{3,2} &= k^1_{2,1}\end{aligned}\right\}. \tag{82}$$

Altogether, we have a set of four linear mathematical relationships between the transformed model parameters in **R**:

$$\begin{Bmatrix}\varepsilon^0_{1,2} - \varepsilon^0_{2,3} - \phi_1 = 0\\ \varepsilon^0_{3,2} - \varepsilon^0_{2,1} - \phi_1 = 0\\ k^1_{1,2} - k^1_{2,3} = 0\\ k^1_{3,2} - k^1_{2,1} = 0\end{Bmatrix}. \tag{83}$$

Together, these four equations reduce the number of free parameters by four, from 14 down to 10. We must point out that the same allosteric relationships could be implemented just as well without the explicit use of an allosteric factor. Thus, we could write the following constraint equation:

$$\frac{k_{1,2}}{k_{2,3}} = \frac{k_{3,2}}{k_{2,1}}. \tag{84}$$

Again, after taking the logarithm, we obtain a set of two equations:

$$\left\{\begin{aligned}\ln(k^0_{1,2}) - \ln(k^0_{2,3}) &= \ln(k^0_{3,2}) - \ln(k^0_{2,1})\\ k^1_{1,2} - k^1_{2,3} &= k^1_{3,2} - k^1_{2,1}\end{aligned}\right\}. \tag{85}$$

The first equation in the set enforces the allosteric relationships at $V = 0$. However, the second equation is not sufficient to enforce the allosteric relationships at any arbitrary voltage. To do so, we must add either one of the following two equations:

$$\begin{aligned}k^1_{1,2} &= k^1_{2,3}, \text{ or}\\ k^1_{3,2} &= k^1_{2,1}.\end{aligned} \tag{86}$$

Altogether, this is equivalent to having a set of three mathematical relationships:

$$\left\{\begin{aligned}\ln(k^0_{1,2}) - \ln(k^0_{2,3}) &= \ln(k^0_{3,2}) - \ln(k^0_{2,1})\\ k^1_{1,2} &= k^1_{2,3}\\ k^1_{3,2} &= k^1_{2,1}\end{aligned}\right\}, \tag{87}$$

with the final form:

$$\begin{Bmatrix}\varepsilon^0_{1,2} + \varepsilon^0_{2,1} - \varepsilon^0_{2,3} - \varepsilon^0_{3,2} = 0\\ k^1_{1,2} - k^1_{2,3} = 0\\ k^1_{3,2} - k^1_{2,1} = 0\end{Bmatrix}. \tag{88}$$

This result can be easily verified: the same set of equations can be obtained by eliminating ϕ_1 between the first two equalities in Eq. 83. Without the explicit use of an allosteric factor, the model would have only 13 model parameters. However, there would be only three constraint equations in that case, which means that the number of free parameters would still be the same: 10. In conclusion, adding an allosteric factor does not necessarily increase the number of free parameters of a model. Instead, it provides a more intuitive way of formulating the relationships that may exist between rate constants.

Another assumption that we made about our test model is that the O_3 to I_4 transition has the same voltage sensitivity as the C_2 to O_3 transition. This results in one mathematical relationship:

$$k^1_{3,4} - k^1_{2,3} = 0. \tag{89}$$

With this relationship, the number of free parameters is down to nine. We note that this relationship follows from the actual model parameters used to simulate the data. However, even if the true model parameters were unknown, the savvy investigator would still enforce this constraint, motivated by the shape of the activation curve, which reaches a constant value toward the more positive voltages (Fig. 3 B). For this particular model, this aspect of the activation curve suggests that the rates of activation and inactivation increase by approximately the same factor with voltage. If, for example, the inactivation rate had a stronger voltage dependence than the activation rate ($k^1_{3,4} > k^1_{2,3}$), the activation curve would start turning down at more positive potentials.

The final assumptions we made involve inequality constraints. Thus, we constrained the rate of recovery from inactivation (I_4 to O_3) to have negative voltage dependence and the deactivation rates (O_3 to C_2 and C_2 to C_1) to have a voltage sensitivity greater than $-0.15\ \text{mV}^{-1}$:

$$\begin{aligned}k^1_{4,3} &\le 0,\\ k^1_{2,1} &\ge -0.15.\end{aligned} \tag{90}$$

Because the $k^1_{3,2}$ and $k^1_{2,1}$ factors are already constrained to be equal, we apply the inequality constraint only to $k^1_{2,1}$, to avoid redundancy. To handle these two inequality constraints, we add two slack variables, z_1 and z_2, and write two equality relationships:

$$\begin{aligned} k^1_{4,3} &= 0.0 - {z_1}^2, \\ k^1_{2,1} &= -0.15 + {z_2}^2. \end{aligned} \tag{91}$$

Thus, although we added two constraints, we also added two slack variables. As a result, the number of free parameters remains the same: nine.

We summarize here all the constraint equations:

$$\left\{\begin{aligned} &\varepsilon^0_{1,2} - \varepsilon^0_{2,3} - \phi_1 = 0 \\ &\varepsilon^0_{3,2} - \varepsilon^0_{2,1} - \phi_1 = 0 \\ &k^1_{1,2} - k^1_{2,3} = 0 \\ &k^1_{3,2} - k^1_{2,1} = 0 \\ &k^1_{3,4} - k^1_{2,3} = 0 \\ &k^1_{4,3} = 0.0 - {z_1}^2 \\ &k^1_{2,1} = -0.15 + {z_2}^2 \end{aligned}\right\}. \tag{92}$$

Linear algebra calculations. We can now formulate the constraint matrix **M** and vector **V**, as in Eq. 37 (Salari et al., 2018):

$$\mathbf{M} = \begin{array}{c} \begin{array}{cccccccccccccc} \varepsilon^0_{1,2} & k^1_{1,2} & \varepsilon^0_{2,1} & k^1_{2,1} & \varepsilon^0_{2,3} & k^1_{2,3} & \varepsilon^0_{3,2} & k^1_{3,2} & \varepsilon^0_{3,4} & k^1_{3,4} & \varepsilon^0_{4,3} & k^1_{4,3} & \phi_1 & \varphi_1 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{93}$$

$$\mathbf{V} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 - {z_1}^2 \\ -0.15 + {z_2}^2 \end{bmatrix}.$$

As **M** contains only constant values, it can now be decomposed with the singular value decomposition technique into three matrices, as in Eq. 40 (Salari et al., 2018):

$$\mathbf{U_M} = \begin{bmatrix} -0.707 & 0 & 0 & -0.707 & 0 & 0 & 0 \\ -0.707 & 0 & 0 & 0.707 & 0 & 0 & 0 \\ 0 & 0.707 & 0 & 0 & -0.707 & 0 & 0 \\ 0 & 0 & 0.851 & 0 & 0 & 0 & -0.526, \\ 0 & 0.707 & 0 & 0 & 0.707 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -0.526 & 0 & 0 & 0 & -0.851 \end{bmatrix} \tag{94}$$

$$\mathbf{S_M} = \begin{bmatrix} 2 \\ 1.732 \\ 1.618 \\ 1.414, \\ 1 \\ 1 \\ 0.618 \end{bmatrix} \tag{95}$$

$$\mathbf{V_M} = \begin{bmatrix} -0.354 & 0 & 0 & -0.5 & 0 & 0 & 0 & -0.791 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.408 & 0 & 0 & -0.707 & 0 & 0 & 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ 0.354 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0.158 & 0 & -0.192 & 0 & 0.580 & -0.476 & 0 \\ 0 & 0 & -0.851 & 0 & 0 & 0 & -0.526 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.354 & 0 & 0 & 0.5 & 0 & 0 & 0 & -0.474 & 0 & -0.067 & 0 & -0.173 & -0.604 & 0 \\ 0 & -0.817 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ -0.354 & 0 & 0 & 0.5 & 0 & 0 & 0 & -0.158 & 0 & -0.125 & 0 & 0.754 & 0.129 & 0 \\ 0 & 0 & 0.526 & 0 & 0 & 0 & -0.851 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.408 & 0 & 0 & 0.707 & 0 & 0 & 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & -0.316 & 0 & 0.067 & 0 & 0.173 & 0.604 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{96}$$

From $\mathbf{V_M}$, we can now obtain the **A** matrix, as shown in Eq. 41 (Salari et al., 2018):

$$\mathbf{A} = \begin{bmatrix} -0.791 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ 0.158 & 0 & -0.192 & 0 & 0.580 & -0.476 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.474 & 0 & -0.067 & 0 & -0.173 & -0.604 & 0 \\ 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ -0.158 & 0 & -0.125 & 0 & 0.754 & 0.129 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.559 & 0 & 0.108 & -0.093 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.316 & 0 & 0.067 & 0 & 0.173 & 0.604 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{97}$$

The $\mathbf{A}^{-1}$ matrix is simply obtained by transposing **A**, and we do not show it here. To obtain the **B** vector, we must first calculate the pseudoinverse of **M**, $\mathbf{M}^+$, as shown in Eq. 43 (Salari et al., 2018). First, we calculate the pseudoinverse of $\mathbf{S_M}$, $\mathbf{S_M}^+$:

$$\mathbf{S_M}^+ = \begin{bmatrix} 0.5 \\ 0.577 \\ 0.618 \\ 0.707 \cdot \\ 1 \\ 1 \\ 1.618 \end{bmatrix} \tag{98}$$

With $\mathbf{S_M}^+$, we can calculate $\mathbf{M}^+$:

$$\mathbf{M}^{+} = \begin{matrix} 0.375 & -0.125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.667 & 0 & -0.333 & 0 & 0 \\ 0.125 & -0.375 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -0.375 & 0.125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.333 & 0 & -0.333 & 0 & 0 \\ -0.125 & 0.375 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.333 & 0 & 0.667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -0.25 & -0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}. \quad (99)$$

The $\mathbf{M}^{+}$ matrix can now be used to calculate the $\mathbf{B}$ vector, as in Eq. 42 (Salari et al., 2018). However, when the model contains inequality constraints, the $\mathbf{V}$ vector will contain elements that depend on the slack variables z_1 and z_2. During optimization, the slack variables are changed freely by the parameter estimation engine. However, at the beginning of the optimization, they must be initialized by solving their corresponding constraint equation. In this case, z_1 is initialized as follows:

$$\begin{aligned} k_{4,3}^{1} &= 0 - {z_1}^2 \quad \Rightarrow \\ z_1 &= \sqrt{k_{4,3}^{1} - 0} = \sqrt{0.1} = 0.316, \end{aligned} \quad (100)$$

where 0.1 is the initial value of $k_{4,3}^{1}$. Likewise, z_2 is initialized as:

$$z_2 = 0.274. \quad (101)$$

With the z_1 and z_2 values, we can now calculate the initial $\mathbf{V}$ and $\mathbf{B}$ vectors:

$$\mathbf{V} = \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -0.1 \\ -0.075 \end{matrix}, \quad (102)$$

$$\mathbf{B} = \begin{matrix} 0 \\ 0 \\ 0 \\ -0.075 \\ 0 \\ 0 \\ 0 \\ -0.075 \\ 0 \\ 0 \\ 0 \\ -0.1 \\ 0 \\ 0 \end{matrix}, \quad (103)$$

To start the optimization, we must initialize the free parameters $\bar{\mathbf{X}}$. When the model constraints include inequalities, as we have here, $\bar{\mathbf{X}}$ is formed by the reunion of $\mathbf{X}$ and $\mathbf{Z}$ vectors (Eq. 44; Salari et al., 2018). $\mathbf{Z}$ contains the slack variables, which are initialized as shown above, whereas $\mathbf{X}$ is initialized from the initial set of model parameters $\mathbf{R}_0$, using Eq. 39 (Salari et al., 2018). Altogether, the initial free parameter values are:

$$\bar{\mathbf{X}}_0 = \begin{matrix} 10.640 \\ 7.313 \\ -3.708 \\ 2.996 \\ -4.537 \\ -5.626 \\ 8.006 \\ 0.316 \\ 0.274 \end{matrix}. \quad (104)$$

Each time the cost function is requested by the optimization engine, the transformed model parameters $\mathbf{R}$ are calculated from the free parameters $\bar{\mathbf{X}}$ with Eq. 40 (Salari et al., 2018). Then, the model parameters $\mathbf{K}$ are calculated from $\mathbf{R}$.

Applying arbitrary parameter constraints and behavioral constraints. In addition to linear parameter constraints, we also tested a few simple but useful constraints that cannot be implemented with the linear algebra formalism (Salari et al., 2018). First, we tested an arbitrary parameter constraint that restricts the channel count N_C to a range of values. The test data were simulated with $N_C = 5{,}000$. However, to test the algorithms under more realistic conditions, we enforced a range of values away from the true value (6,000–8,000). The same strategy was used with all the behavioral constraints introduced next. The constraint and the corresponding cost function component are the following:

$$6{,}000 \le N_C \le 8{,}000, \quad (105)$$

$$F_1^C = \beta_1 \times \left(\frac{N_C - 6{,}000}{6{,}000}\right)^2 + \beta_2 \times \left(\frac{8{,}000 - N_C}{8{,}000}\right)^2, \quad (106)$$

where β_1 and β_2 are numerical factors with the following properties:

$$\left\{\begin{matrix} \beta_1 = 0 \text{ if } N_C \ge 6{,}000 \\ \beta_1 = \alpha \text{ if } N_C < 6{,}000 \end{matrix}\right\}, \quad \left\{\begin{matrix} \beta_2 = 0 \text{ if } N_C \le 8{,}000 \\ \beta_2 = \alpha \text{ if } N_C > 8{,}000 \end{matrix}\right\}. \quad (107)$$

The normalization to 6,000 or 8,000 makes this penalty component numerically comparable with all the other penalty and data components.

The second is a behavioral constraint that enforces the maximum open probability reached during a brief depolarization step from −120 to 0 mV, as illustrated in Fig. 2 B (run IV). With the true parameter values, the model predicts a maximum P_O of ∼0.42, but we constrained it to 0.5. The constraint equation and the corresponding cost function component are the following:

$$P_O = 0.5, \tag{108}$$

$$F_2^C = \alpha \times (P_O - 0.5)^2. \tag{109}$$

Finally, we tested a behavioral constraint that enforces the time constant of recovery from inactivation. As discussed earlier, it would be rather difficult to calculate this quantity analytically. Instead, we use a surrogate value, extracted from a simulation in response to a two-pulse voltage-clamp protocol. As shown in Fig. 2 B (run V), we inactivate the channels with a brief voltage pulse, let them recover for 50 ms, and then apply a second pulse to test how many channels have recovered. The recovered fraction is defined as the maximum open probability reached during the second voltage pulse relative to the first pulse:

$$f_R = \frac{(P_O)_{pulse2}}{(P_O)_{pulse1}}. \tag{110}$$

Thus, if we want to enforce a specific recovery time constant τ_R, we can calculate the corresponding f_R for a recovery interval of arbitrary duration t and use that f_R value in the behavioral constraint:

$$f_R = \exp\left(-\frac{t}{\tau_R}\right). \tag{111}$$

Our test model predicts a recovered fraction f_R of ~0.43 with a recovery interval t = 50 ms, at −80 mV, but we constrained it to 0.8. The constraint equation and the corresponding cost function component are the following:

$$f_R = 0.8, \tag{112}$$

$$F_3^C = \alpha \times (f_R - 0.8)^2. \tag{113}$$

Optimizing the model. We illustrate the performance of the algorithms with six optimization runs, each implementing a different set of constraints, as described in Fig. 2 B. Together, these examples test the full range of constraints that the algorithms are designed to handle, as they are likely to occur in practical modeling applications: linear equality and inequality parameter constraints and model behavior and properties. Furthermore, we test all types of model parameters, as defined in the companion paper: rate constant parameters, multiplicative factors (a_1), and external parameters (N_C). The true parameter values, as well as the initial and the estimated values obtained in each optimization run, are given in Table 1.

In run I, we enforced only equality linear parameter constraints (Eq. 83). The cost function that was minimized by the optimizer had the following expression:

$$F(\bar{\mathbf{X}}) = F_1^D + F_2^D + F_3^D, \tag{114}$$

where F_1^D, F_2^D, and F_3^D are the cost components corresponding to the data shown in Fig. 3: time-course traces, activation curve, and availability curve, respectively. Each of these data components is the sum of square differences between the data and the prediction of the model, normalized by the total number of data points. The time-course component was also normalized to the peak current, as follows:

$$F_1^D = \frac{1}{N_V \times N_t} \sum_{V,t} \left(\frac{y_{V,t} - I_{V,t}}{y_{Peak}}\right)^2, \tag{115}$$

where N_V is the number of traces, N_t is the number of samples in each trace, $y_{V,t}$ and $I_{V,t}$ are the data point and the predicted current, respectively, at voltage V and time t, and y_{peak} is the largest negative peak current in the entire dataset. With these normalizations, all three data cost components take comparable values. We have not done it here but, in principle, one should further normalize the data to account for potentially different levels of noise, such as between the time course traces and the activation and availability curves. One possibility would be to multiply each cost component by a factor inversely proportional to its normalized variance, to ensure that less noisy datasets will be fitted more tightly by the model. This variance can be approximated through fitting each dataset with an appropriate mathematical function (e.g., a sum of exponentials for the time course data and a Boltzmann for the activation and inactivation curves).

In run II, we used the same conditions as for run I, but we added the inequality linear parameter constraints (Eqs. 90 and 91). In runs III through VI, we applied the same linear parameter constraints as in run II, but in each of these runs, we added different constraints that were implemented via the penalty mechanism: an arbitrary parameter constraint that restricts N_C to a range of values (run III) and behavioral constraints that enforce P_O (run IV), the recovered fraction f_R (run V), or both P_O and f_R simultaneously (run VI). In runs III through VI, the optimizer minimized a penalized cost function with the following expression:

$$F(\mathbf{X}, \alpha) = F_1^D + F_2^D + F_3^D + F^C, \tag{116}$$

where F^C stands for either F_1^C (run III), F_2^C (run IV), F_3^C (run V), or $F_2^C + F_3^C$ (run VI).

The optimization results shown in Fig. 4 demonstrate the proper functioning of the algorithm with all types of constraints. To test the convergence of the optimizer, we intentionally chose starting parameters (Table 1) that generate prediction curves that deviate substantially from the data, as shown by the blue traces in Fig. 3. In all cases, the cost function virtually settled in ~30 iterations (Fig. 4 A, left), after which most model parameters changed little (Fig. 4 B). For run I, the final parameter

values are within ∼10% of the true values (Table 1), which is to be expected under these conditions (Milescu et al., 2005). For the other runs, the constraints push some of the parameters away from their true values, as intended. Although the final parameter values (Table 1) vary across the six runs, they all predict virtually identical fit curves, all represented by the red traces in Fig. 3 (A and B).

The effect of inequality linear parameter constraints can be observed by comparing runs I and II. In run I, the $k^1_{4,3}$ parameter is unconstrained and meanders to values as large as +0.12 mV^{-1}, finally converging to a slightly positive value, even though the true value is slightly negative (−0.05 mV^{-1}). The convergence to a positive value for $k^1_{4,3}$ is not a failure of the search engine but simply a result of the stochastic nature of the data. In run II, $k^1_{4,3}$ is constrained to a negative range and, as expected, converges to a final value of zero. The convergence to a value that lies on the edge of the constrained range would suggest that this solution is suboptimal, compared with the solution found in run I. Indeed, the cost function value is nominally larger: 0.000533 for run II versus 0.000392 for run I, although the difference is imperceptible. The $k^1_{2,1}$ parameter is also constrained with an inequality in run II. However, $k^1_{2,1}$ hovers comfortably above its limit in run I, and, as expected, the constraint applied in run II has no effect.

In runs III through VI, the cost function is replaced by a penalized cost function, which adds penalty components (Eq. 73). In all of these cases, the penalty function quickly drops by four or five orders of magnitude during the optimization (Fig. 4 A, right). In run III, where the penalty mechanism enforces a range of values for N_C, the penalty function occasionally drops to zero (Fig. 4 A, right, orange trace), whenever N_C is within the allowed range and the constraint is exactly satisfied (Fig. 5 A). Although the initial value of N_C (3,000) was outside the acceptable range, the optimizer quickly brought N_C within the range, in just a few iterations. We find it interesting that the convergence value of N_C does not lie on the edge of its allowed range (6,000), as close as possible to the convergence value found in run II (5,500). This suggests the existence of multiple solutions that predict identical fits.

In runs IV through VI, the penalty mechanism was used to enforce equality relationships for P_O and f_R. Like with N_C in run III, the initial values of P_O and f_R were quite different from their enforced values. However, a few iterations were sufficient to bring P_O or f_R close to their enforced values, as illustrated in Fig. 5 (B and C, green and magenta traces). In contrast to run III, the penalty function approaches a small value but does not reach zero (Fig. 4 A, right, green, blue, and magenta traces). Accordingly, the enforced quantities hover in a small neighborhood centered on their enforced values (Fig. 5, B and C). The size of this neighborhood depends on the numerical value of the penalty parameter α_p: the larger the α_p, the smaller the neighborhood. In principle, enforcing the penalty might require several cycles, where each cycle increases the value of α_p, as illustrated in Fig. 1, and tightens the constraint. However, for these relatively simple optimization examples, we initialized the penalty factor as $\alpha_0 = 1$, which enforced the constraints tightly enough in a single penalty cycle.

As expected, adding these constraints that push P_O and f_R away from their true values also results in slightly suboptimal fits in runs IV through VI, compared with runs I through III. Furthermore, these constraints expose correlations between properties of the model (P_O and f_R) and certain model parameters. Thus, P_O is inversely correlated with N_C. Without any constraint, P_O and N_C are estimated as ∼0.42 and 5,100, respectively. In contrast, when P_O is constrained (runs IV and VI), the N_C estimate is lowered to 4,000 (Fig. 5 A). Vice versa, when N_C is constrained to a larger value, the estimated parameters predict a lower P_O (Fig. 5 B). Likewise, f_R is correlated with the rate of recovery from inactivation (the I_4 to O_3 transition). Thus, enforcing f_R to a larger value (0.8) than the true value (0.43) results in a smaller estimate for $k^0_{4,3}$ and in a more negative estimate for $k^1_{4,3}$ (Fig. 4 B, runs V and VI). Considering these potential correlations between different parameters or model properties, one should be careful not to apply contradictory constraints.

DISCUSSION

We have presented here a set of mathematical and computational tools that can be used to estimate kinetic mechanisms that explain new data but also satisfy user-defined prior knowledge. In part one of this study (Salari et al., 2018), we derived a procedure for enforcing explicit linear equality and inequality parameter relationships. Here, in part two, we introduced a procedure for enforcing arbitrary model properties and behaviors, as well as arbitrary parameter relationships. Together, these methods are capable of handling virtually all types of model constraints that are likely to arise in practical situations. To demonstrate our approach, we provided a step-by-step numerical example. Interested readers can use these examples to implement the constraint algorithms in their own software and to verify correctness. We also implemented these algorithms in the freely available QuB software, as maintained by our laboratory (Milescu, 2015).

Compatibility with existing optimization frameworks

The procedures described here can be easily adapted into a typical optimization package. As illustrated by the workflow diagram in Fig. 1, only a few modifications would be required: adding a function that converts between free parameters and model parameters ($\bar{\mathbf{X}}_\mathbf{k} \rightarrow \mathbf{K}_\mathbf{k}$) and vice versa, modifying the cost function

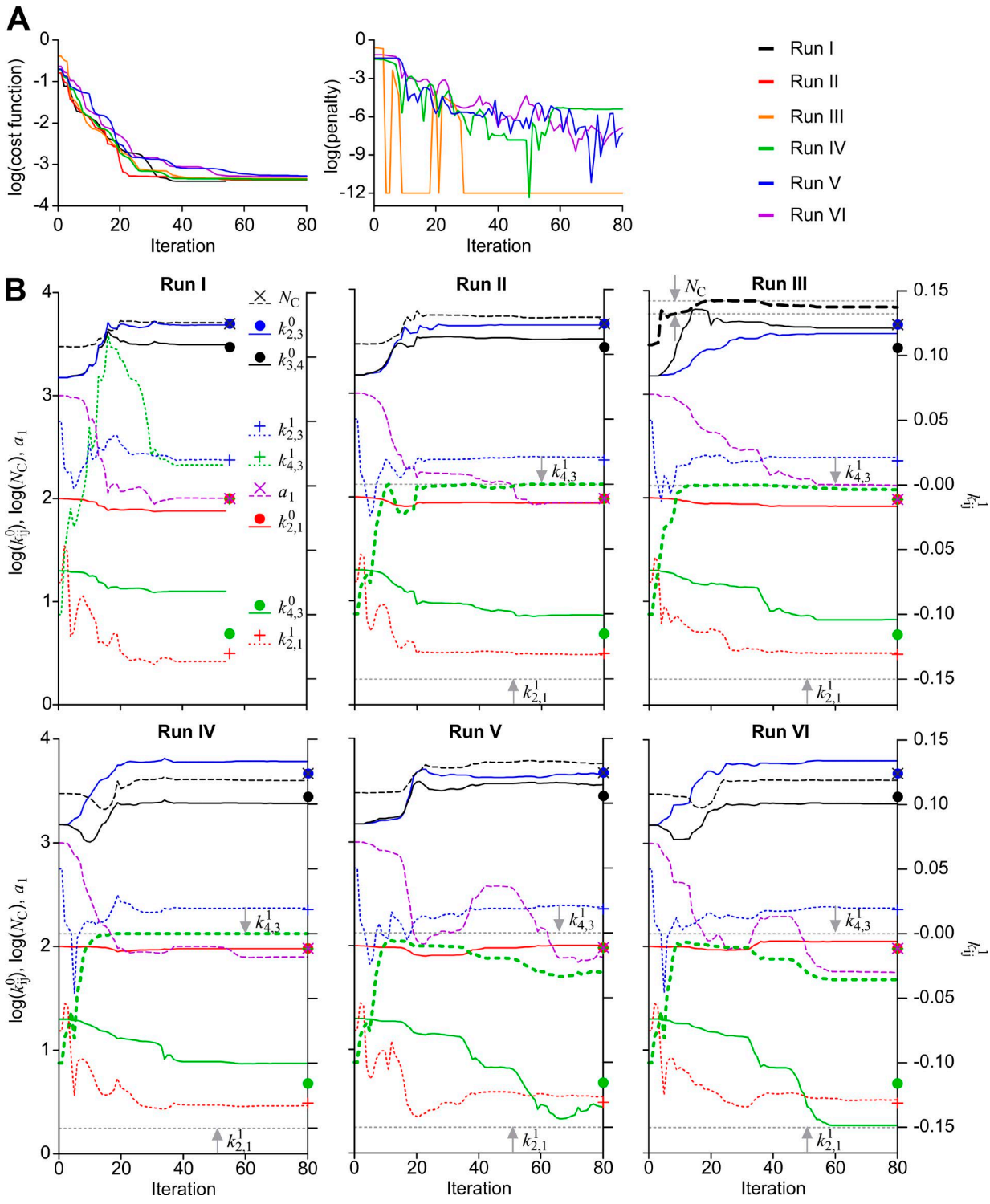

Figure 4. **Testing optimization with model constraints.** The model shown in Fig. 2 A was optimized to fit the data in Fig. 3 (time course and steady-state curves), subject to the six sets of constraints shown in Fig. 2 B. **(A)** The convergence of the overall cost function (left) and penalty component (right). **(B)** Parameter convergence in each of the six test runs. Only the model parameters **K** are shown, but note that the optimizer searches in the free parameter space defined by $\tilde{\mathbf{X}}$. To reduce clutter, some model parameters are not displayed, as they are defined by constraints (e.g., $k^0_{1,2} = a \times k^0_{2,3}$). For better visualization, the exponential factors k^1_{ij} are plotted on the right axis (dotted lines), whereas all the other quantities are on the left axis: preexponential factors k^0_{ij} (log scale, solid lines), channel count N_C (log scale, dashed black line), and allosteric factor a_1 (dashed magenta line). The dashed gray horizontal lines and arrows indicate the boundaries of inequality linear constraints for $k^1_{2,1}$ and $k^1_{4,3}$ (runs II through IV) and the boundaries of the range constraint for N_C (run III). Note how $k^1_{4,3}$ is estimated as a positive value in run I, but it remains less than zero under the inequality constraint in runs II through VI. In each panel, the symbols aligned with the last iteration mark the true parameter values.

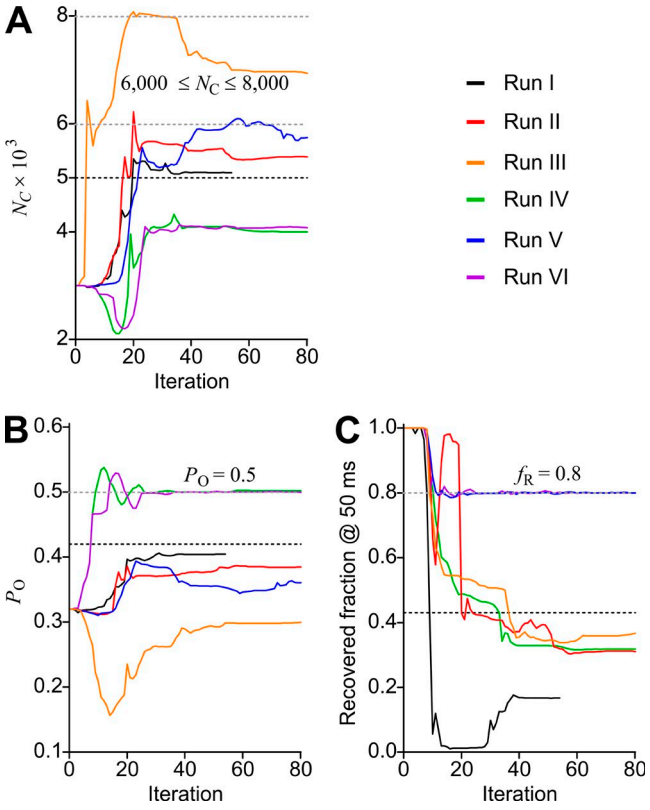

Figure 5. **Enforcing model properties and behavior. (A–C)** The convergence of the N_C estimate (A) and calculated P_O (B) and f_R (C) quantities are shown for each of the six optimization runs described in Fig. 2 B. The constrained and the true values are indicated by the gray and black dashed lines, respectively. As expected, N_C and P_O are inversely correlated; when one is constrained to be larger or smaller than the true value, the other one becomes smaller or larger, respectively. Likewise, f_R is correlated with the rate of recovery from inactivation; when f_R is constrained to be larger than the true value, the $k^0_{4,3}$ and $k^1_{4,3}$ estimates become smaller and more negative, respectively. All the enforced quantities quickly reach their enforced range (N_C) or value (P_O and f_R). Note that without being constrained, f_R is not well defined by the test data and does not converge to the true value. In contrast, N_C and P_O are well defined.

to calculate and add the penalty, and implementing a schedule to progressively increase the penalty parameter. The first two modifications are trivial because every optimizer will have a callback function where the user writes custom code to calculate the cost function for a given set of free parameters. The third modification is potentially more involved, but a simple solution would be to increase the penalty parameter by hand and restart the optimizer with the parameter values obtained in the previous iteration.

Constrained fitting versus multiobjective fitting

There is a certain similarity between constrained fitting and simply including those data that underlie the constraints into a more comprehensive dataset to be fitted. The second approach is generally described as multiobjective fitting (Druckmann et al., 2007; Bandyopadhyay and Saha, 2013). Although it is not a substitute for the reduction method that is used to enforce linear parameter constraints, it could be a substitute for the penalty method. As the name implies, in this case the optimizer would need to find a solution that satisfies multiple objectives (i.e., datasets). This is conceptually equivalent to constrained fitting, but there is also one important difference: in multiobjective fitting, the optimal solution found by the search engine may actually explain poorly each and all of the individual datasets, as long as it is the best overall compromise. Moreover, to find this compromise solution, one must choose a set of weighting factors that encode how much each dataset is worth to the model, which is not trivial.

In contrast, the constraining mechanism described in this study will give the highest priority to the constraints and satisfy them exactly (the linear parameter constraints, via the reduction method) or at least very closely (all other constraints, via the penalty method). Only after the constraints are satisfied will the model adapt to explain the data (in as much as it is possible). Nevertheless, as we explained in the paper, a certain margin of error can be built into the constraints to accommodate noise and potential artifacts, but the constraints will stay tightly within this margin. Then, one advantage to the constraint approach is that one can more easily detect when a model is incompatible with the data. Furthermore, one could also detect inconsistent knowledge, as signaled by incompatible constraints.

Model behavior: To enforce or not?

The need for enforcing explicit parameter relationships is obvious, if only to consider microscopic reversibility or the ratio of sequential activation rates. However, it may be less clear to the reader why model behavior and properties need to be enforced. Why not derive them directly from the data? After all, once model parameters are estimated, they can be used to predict any model property or behavior. The problem resides in the potential lack of model and parameter identifiability. In an ideal case, the model would be uniquely identifiable, which means that no other topology exists that can explain the data equally well (Kienker, 1989; Bruno et al., 2005). Furthermore, the data would be noise and artifact free and the model parameters would be fully identifiable, which means that the model admits a unique solution and the optimizer is able to find it from the data. If this were the case, then it would make little sense to enforce a model behavior or property except to test the sensitivity of the parameters with respect to that behavior.

In reality, however, the true model may never be known, and the working model may be just one out of many equivalent topologies. Furthermore, the parameters may not be fully identifiable, either because the

model admits multiple solutions (theoretical parameter identifiability) or because the data are corrupted by noise and artifacts that flatten the cost function surface (practical parameter identifiability; Milescu et al., 2005; Raue et al., 2009; Siekmann et al., 2012; Hines et al., 2014; Middendorf and Aldrich, 2017). Thus, estimating the kinetic mechanism from limited data may result in a parameter set that is just one out of many possible solutions and potentially one with poor predictive power.

This is actually the case with our numerical example: in all runs, the estimates obtained by the optimizer are close to the true values (Table 1), except when otherwise constrained (e.g., N_C in run III). However, the estimates differ across runs, even though the fits are virtually identical between runs and follow closely the data (Fig. 3, red lines). How can different sets of parameters produce the same solution? The explanation for this apparent contradiction is that the parameters are not uniquely identifiable given the reduced dataset. Clearly, adding more constraints or enforcing other model behaviors would improve parameter identifiability and would select only those parameter solutions that are compatible with that behavior. Furthermore, it would also improve model identifiability, making it easier to discover the correct model. Short of the true model, we would at least obtain more robust models, as nature always does.

ACKNOWLEDGMENTS

We thank the members of the Milescu laboratories for their constructive comments and suggestions.

This work was supported by the American Heart Association (grants 13SDG16990083 to L.S. Milescu and 13SDG14570024 to M. Milescu) and the Graduate Assistance in Areas of National Need Initiative/Department of Education (training grant fellowship to M.A. Navarro).

The authors declare no competing financial interests.

Author contributions: L.S. Milescu developed the mathematical and computational algorithms and implemented them in software. All authors contributed to designing and testing the algorithms and software and writing the manuscript.

Richard W. Aldrich served as editor.

Submitted: 28 September 2017
Accepted: 6 December 2017

REFERENCES

Bandyopadhyay, S., and S. Saha. 2013. Some single-and multiobjective optimization techniques. *In* Unsupervised Classification. Springer, Berlin. 17–58. https://doi.org/10.1007/978-3-642-32451-2_2

Bertsekas, D.P. 1975. Nondifferentiable optimization via approximation. *In* Nondifferentiable Optimization. Springer, New York. 1–25. https://doi.org/10.1007/BFb0120696

Bruno, W.J., J. Yang, and J.E. Pearson. 2005. Using independent open-to-closed transitions to simplify aggregated Markov models of ion channel gating kinetics. *Proc. Natl. Acad. Sci. USA*. 102:6326–6331. https://doi.org/10.1073/pnas.0409110102

Colquhoun, D., and A.G. Hawkes. 1982. On the stochastic properties of bursts of single ion channel openings and of clusters of bursts. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 300:1–59. https://doi.org/10.1098/rstb.1982.0156

Colquhoun, D., and F. Sigworth. 1995. Fitting and statistical analysis of single-channel records. *In* Single-channel recording. B. Sakmann, and E. Neher, editors. Plenum Press, New York. 483–587. https://doi.org/10.1007/978-1-4419-1229-9_19

Colquhoun, D., C.J. Hatton, and A.G. Hawkes. 2003. The quality of maximum likelihood estimates of ion channel rate constants. *J. Physiol.* 547:699–728. https://doi.org/10.1113/jphysiol.2002.034165

Csanády, L. 2006. Statistical evaluation of ion-channel gating models based on distributions of log-likelihood ratios. *Biophys. J.* 90:3523–3545. https://doi.org/10.1529/biophysj.105.075135

Druckmann, S., Y. Banitt, A. Gidon, F. Schürmann, H. Markram, and I. Segev. 2007. A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. *Front. Neurosci.* 1:7–18. https://doi.org/10.3389/neuro.01.1.1.001.2007

Fletcher, R. 2013. Practical Methods of Optimization. John Wiley & Sons, New York.

Fletcher, R., and M.J.D. Powell. 1963. A rapidly convergent descent method for minimization. *Comput. J.* 2:163–168. https://doi.org/10.1093/comjnl/6.2.163

Gurkiewicz, M., and A. Korngreen. 2007. A numerical approach to ion channel modelling using whole-cell voltage-clamp recordings and a genetic algorithm. *PLOS Comput. Biol.* 3:e169. https://doi.org/10.1371/journal.pcbi.0030169

Himmelblau, D.M. 1972. Applied Nonlinear Programming. McGraw-Hill, New York.

Hines, K.E., T.R. Middendorf, and R.W. Aldrich. 2014. Determination of parameter identifiability in nonlinear biophysical models: A Bayesian approach. *J. Gen. Physiol.* 143:401–416.

Kienker, P. 1989. Equivalence of aggregated Markov models of ion-channel gating. *Proc. R. Soc. Lond. B Biol. Sci.* 236:269–309. https://doi.org/10.1098/rspb.1989.0024

Liu, Y., J. Park, K.A. Dahmen, Y.R. Chemla, and T. Ha. 2010. A comparative study of multivariate and univariate hidden Markov modelings in time-binned single-molecule FRET data analysis. *J. Phys. Chem. B.* 114:5386–5403. https://doi.org/10.1021/jp9057669

Menon, V., N. Spruston, and W.L. Kath. 2009. A state-mutating genetic algorithm to design ion-channel models. *Proc. Natl. Acad. Sci. USA*. 106:16829–16834. https://doi.org/10.1073/pnas.0903766106

Middendorf, T.R., and R.W. Aldrich. 2017. Structural identifiability of equilibrium ligand-binding parameters. *J. Gen. Physiol.* 149:105–119. https://doi.org/10.1085/jgp.201611702

Milescu, L.S. 2015. QuB: The Mlab version. Available at: https://milesculabs.biology.missouri.edu/QuB.html

Milescu, L.S., G. Akk, and F. Sachs. 2005. Maximum likelihood estimation of ion channel kinetics from macroscopic currents. *Biophys. J.* 88:2494–2515. https://doi.org/10.1529/biophysj.104.053256

Milescu, L.S., A. Yildiz, P.R. Selvin, and F. Sachs. 2006a. Maximum likelihood estimation of molecular motor kinetics from staircase dwell-time sequences. *Biophys. J.* 91:1156–1168. https://doi.org/10.1529/biophysj.105.079541

Milescu, L.S., A. Yildiz, P.R. Selvin, and F. Sachs. 2006b. Extracting dwell time sequences from processive molecular motor data. *Biophys. J.* 91:3135–3150. https://doi.org/10.1529/biophysj.105.079517

Milescu, L.S., T. Yamanishi, K. Ptak, M.Z. Mogri, and J.C. Smith. 2008. Real-time kinetic modeling of voltage-gated ion channels

using dynamic clamp. *Biophys. J.* 95:66–87. https://doi.org/10.1529/biophysj.107.118190

Milescu, L.S., T. Yamanishi, K. Ptak, and J.C. Smith. 2010. Kinetic properties and functional dynamics of sodium channels during repetitive spiking in a slow pacemaker neuron. *J. Neurosci.* 30:12113–12127. https://doi.org/10.1523/JNEUROSCI.0445-10.2010

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. 1992. Numerical recipes in C. Cambridge University Press, Cambridge, MA.

Qin, F., A. Auerbach, and F. Sachs. 1996. Estimating single-channel kinetic parameters from idealized patch-clamp data containing missed events. *Biophys. J.* 70:264–280. https://doi.org/10.1016/S0006-3495(96)79568-1

Qin, F., A. Auerbach, and F. Sachs. 2000. A direct optimization approach to hidden Markov modeling for single channel kinetics. *Biophys. J.* 79:1915–1927. https://doi.org/10.1016/S0006-3495(00)76441-1

Raue, A., C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. 2009. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics.* 25:1923–1929. https://doi.org/10.1093/bioinformatics/btp358

Salari, A., M.A. Navarro, and L.S. Milescu. 2016. Modeling the kinetic mechanisms of voltage-gated ion channels. *In* Advanced Patch-Clamp Analysis for Neuroscientists. Humana Press, New York. 267–304. https://doi.org/10.1007/978-1-4939-3411-9_13

Salari, A., M.A. Navarro, M. Milescu, and L.S. Milescu. 2018. Estimating kinetic mechanisms with prior knowledge: I. Linear parameter constraints. *J. Gen. Physiol.* https://doi.org/10.1085/jgp.201711911

Siekmann, I., J. Sneyd, and E.J. Crampin. 2012. MCMC can detect nonidentifiable models. *Biophys. J.* 103:2275–2286. https://doi.org/10.1016/j.bpj.2012.10.024

Stepanyuk, A.R., A.L. Borisyuk, and P.V. Belan. 2011. Efficient maximum likelihood estimation of kinetic rate constants from macroscopic currents. *PLoS One.* 6:e29731. https://doi.org/10.1371/journal.pone.0029731

Stepanyuk, A., A. Borisyuk, and P. Belan. 2014. Maximum likelihood estimation of biophysical parameters of synaptic receptors from macroscopic currents. *Front. Cell. Neurosci.* 8:303. https://doi.org/10.3389/fncel.2014.00303

Venkataramanan, L., and F.J. Sigworth. 2002. Applying hidden Markov models to the analysis of single ion channel activity. *Biophys. J.* 82:1930–1942. https://doi.org/10.1016/S0006-3495(02)75542-2

Weiss, S. 2000. Measuring conformational dynamics of biomolecules by single molecule fluorescence spectroscopy. *Nat. Struct. Biol.* 7:724–729. https://doi.org/10.1038/78941