frontiers in Neuroinformatics

METHODS
published: 11 March 2016
doi: 10.3389/fninf.2016.00008

PyRhO: A Multiscale Optogenetics Simulation Platform

Benjamin D. Evans[1], Sarah Jarvis[2], Simon R. Schultz[2] and Konstantin Nikolic[1]*

[1] *Centre for Bio-Inspired Technology, Institute of Biomedical Engineering, Department of Electrical and Electronic Engineering, Imperial College London, London, UK,* [2] *Centre for Neurotechnology, Institute of Biomedical Engineering, Department of Bioengineering, Imperial College London, London, UK*

Optogenetics has become a key tool for understanding the function of neural circuits and controlling their behavior. An array of directly light driven opsins have been genetically isolated from several families of organisms, with a wide range of temporal and spectral properties. In order to characterize, understand and apply these opsins, we present an integrated suite of open-source, multi-scale computational tools called PyRhO. The purpose of developing PyRhO is three-fold: (i) to characterize new (and existing) opsins by automatically fitting a minimal set of experimental data to three-, four-, or six-state kinetic models, (ii) to simulate these models at the channel, neuron and network levels, and (iii) provide functional insights through model selection and virtual experiments *in silico*. The module is written in Python with an additional IPython/Jupyter notebook based GUI, allowing models to be fit, simulations to be run and results to be shared through simply interacting with a webpage. The seamless integration of model fitting algorithms with simulation environments (including NEURON and Brian2) for these virtual opsins will enable neuroscientists to gain a comprehensive understanding of their behavior and rapidly identify the most suitable variant for application in a particular biological system. This process may thereby guide not only experimental design and opsin choice but also alterations of the opsin genetic code in a neuro-engineering feed-back loop. In this way, we expect PyRhO will help to significantly advance optogenetics as a tool for transforming biological sciences.

Keywords: optogenetics, opsin, Python, Jupyter, PyRhO, spiking neurons, NEURON simulator, Brian simulator

OPEN ACCESS

Edited by:
Andrew P. Davison, Centre National de la Recherche Scientifique, France

Reviewed by:
Padraig Gleeson, University College London, UK
Emilia Entcheva, Stony Brook University, USA

****Correspondence:***
Benjamin D. Evans
benjamin.evans@imperial.ac.uk

Received: *04 November 2015*
Accepted: *19 February 2016*
Published: *11 March 2016*

Citation:
Evans BD, Jarvis S, Schultz SR and Nikolic K (2016) PyRhO: A Multiscale Optogenetics Simulation Platform. Front. Neuroinform. 10:8. doi: 10.3389/fninf.2016.00008

1. INTRODUCTION

Optogenetics is a biotechnology which renders excitable cells light-sensitive by inserting genes which, upon expression, create light-activated ion channels known originally as rhodopsins (Nagel et al., 2003; Boyden et al., 2005). Over the last 10 years optogenetics has found widespread application, initially in neuroscience (Zhang et al., 2006; Adamantidis et al., 2007; Arenkiel et al., 2007; Han and Boyden, 2007; Yizhar et al., 2011), but increasingly also in more distal areas of physiology such as cardiac science (Arrenberg et al., 2010; Boyle et al., 2013), intracellular signaling (Airan et al., 2009), and gene transcription (Konermann et al., 2013). Applications to date have included control of motor cortex (Aravanis et al., 2007), cortical circuit mapping (Wang et al., 2007; Zhang et al., 2007; Ayling et al., 2009; Petreanu et al., 2009; Klapoetke et al., 2014), optoelectronic neuroprosthetic devices (for example retinal Lagali et al., 2008; Degenaar et al., 2009; Busskamp and Roska, 2011 or cochlear prostheses, Hernandez et al., 2014), regulation of the symptoms

of neurodegenerative disorders (e.g., Parkinson's Gradinaru et al., 2009), closed loop control of epileptic seizures, peripheral nerve stimulation (Arlow et al., 2013), and novel cardiac pacemaker technology (Bruegmann and Sasse, 2015), to name just a few.

In an effort to develop more effective and tailored opsins, hybrids and genetic mutants are continually being created (Berndt et al., 2008; Lin et al., 2009; Hegemann and Moglich, 2011; AzimiHashemi et al., 2014). Experimentally characterizing these new variants is a lengthy process requiring substantial effort before they can be harnessed to address questions in neuroscience (Chow et al., 2010; Gunaydin et al., 2010; Lin, 2011; Chuong et al., 2014). The problem is further compounded when considering the number of combinations between opsins (with total variations now in the hundreds, Zhang et al., 2011) and target cell types. Experimentally testing each combination of opsin and target cell type of interest is practically impossible, effectively limiting the use of optogenetics as a tool.

Theoretical understanding of the underlying mechanisms of optogenetics has developed over the past 10 years (Hegemann et al., 2005; Feldbauer et al., 2009), which has led to a deeper understanding of the biophysical mechanisms of the photosensitization agents which form the foundations of optogenetics (Hegemann et al., 2005; Bamann et al., 2008; Ernst et al., 2008; Nikolic et al., 2009; Stehfest and Hegemann, 2010). Furthermore, the design and engineering of optogenetic devices must start with models of the underlying molecular mechanisms of opsin behavior in cells (Gradinaru et al., 2007; Nikolic et al., 2007; Shoham and Deisseroth, 2010; Foutz et al., 2012; Williams et al., 2013). Computational modeling is thus core to understanding how light induced ionic transport across cell membranes can be tailored for different applications: from probing cellular physiology to creating new treatments for neurological and psychiatric illnesses.

The quest to both expand and refine optogenetics as an effective tool for neuroscience and other areas of physiology requires multiple levels of analysis: from molecular modeling through kinetic models and even network level models. To aid in this effort we propose *PyRhO*; an integrated suite of open-source, multi-scale computational tools to characterize opsins, then rapidly develop and conduct virtual experiments with them *in silico*.

PyRhO offers several integrated computational tools for analysing and experimenting with (rhod)opsins in a virtual environment:

1. The first tool will automatically fit a choice of models to experimental data, extracting the parameters that describe the functional dynamics of the opsins.
2. The second tool can then take these extracted parameters (or alternatively use default values) and simulate a wide range of experimental protocols to reproduce the photo-response of the opsin of interest. These protocols are typically voltage-clamp experiments and include common engineering inputs such as steps, ramps, and chirps, along with more tailored protocols such as pairs of increasingly spaced pulses for evaluating the recovery process.
3. These models and protocols can be run on several simulation platforms spanning multiple scales (to model isolated opsins or transfected neurons) including:
   a. Pure Python for simple channel-level voltage clamp experiments;
   b. NEURON for morphologically detailed models of optogenetically transfected neurons;
   c. Brian2 for simulating whole networks with transfected groups of neurons.
4. A Graphical User Interface (GUI) for easy navigation through all tools, running of virtual experiments and sharing of results.

In this way, PyRhO allows the investigator to simulate opsin dynamics on multiple scales from sub-cellular channels, to individual neurons and finally the dynamics of whole networks. This will help to elucidate the link between the biophysics of opsins and the functional implications of their use in a particular biological system.

The tools are written in Python due to its rapidly growing popularity across the sciences, readability, modularity and large array of open-source modules (Muller et al., 2015). An accompanying GUI running in IPython/Jupyter (Pérez and Granger, 2007) has also been developed to facilitate more interactive exploration of the models for both experimental and pedagogic purposes, requiring virtually no programming experience. In addition to controlling the fitting routines, the GUI also exposes the integrated simulators (e.g., NEURON). Furthermore, this self-logging, notebook-based approach has been identified as a particularly promising medium for sharing models and reproducing results in computational neuroscience (Topalidou et al., 2015).

Simulations based on these virtual opsins will enable neuroscientists to gain insight into their behavior and rapidly identify the most suitable variant for application in a particular biological system, not only guiding choice, but also opsin development. Understanding gained from biologically realistic simulations may provide ideas of how to alter the opsin's genetic code to generate new mutants. These new variations can then be characterized and simulated within PyRhO to determine their suitability for a particular application.

Here, we describe the structure of PyRhO and demonstrate a sample of its capabilities, illustrated through snippets of code and its GUI. We demonstrate the use of PyRhO in fitting models to Channelrhodopsin-2 (ChR2) data and present results for typical illumination strategies and experimental protocols designed to tease apart the effects of key model parameters. We finish with a discussion of the main benefits of using PyRhO, its limitations to date and planned future developments to extend its capabilities.

2. MATERIALS AND METHODS

PyRhO is written as a Python module and released as an open-source project under the revised BSD license. Download and installation instructions can be found with the code at

PyRhO's GitHub repository (https://github.com/ProjectPyRhO/PyRhO), along with example notebooks and a link to the project's website containing further information. A virtual machine with all dependencies installed and examples ready to run is also available, such that the GUI can be used with a minimum of set-up and virtually no programming experience.

The module is comprised of several integrated components for fitting model parameters to experimental data and for simulating the models at multiple scales. Fitting data is an optional step since PyRhO is initialized with default parameters, allowing the user to immediately experiment with simulating the three types of opsin models in order to better understand their dynamics. If the required data are provided to the fitting algorithms however, the parameterized models may be run through the stimulation protocols to efficiently characterize the opsins *in silico*, or determine their suitability for a particular application based upon their dynamics.

2.1. Implementation

PyRhO is implemented as a Python package called `pyrho` which builds upon popular scientific Python modules including `scipy`, `numpy`, `matplotlib`, and `lmfit`. Additionally, if optogenetic simulations in detailed morphological models of individual (or a few) neurons are required, `NMODL` files (Hines and Carnevale, 2000) are provided for use with NEURON (Hines et al., 2009). Similarly, for network-level simulations PyRhO has been integrated with the Brian simulator (Goodman and Brette, 2008, 2009) and includes model descriptions suitable for use with Brian2.

The simulation architecture is designed around three layers of abstraction: models, protocols and simulators. These layers are illustrated in the work-flow schematic of **Figure 1** along with the other major components of PyRhO. Each layer contains families of classes to create a uniform interface for each subclass, for example, the differences in setting the light-dependent transition rates of the three models are shielded from the user by endowing each opsin model subclass with the method `setLight()`. A similar approach is taken with the other layers providing a common set of member variables and methods, making usage consistent and providing a framework for future development of new subclasses (i.e., additional kinetic models, stimulation protocols, and simulation platforms).

2.2. Photocurrent Model

A detailed understanding of how the channel is gated and the ions are conducted is still lacking, although some recent studies have elucidated important aspects of the pore formation and ionic transport (Feldbauer et al., 2009; Kuhne et al., 2014) We assume that all light-sensitive ion channel currents (I) can be expressed in the classic form:

$$I = g \cdot (v - E)\,, \tag{1}$$

where g is the channel conductance, v the membrane voltage and E is the reversal potential for the specific opsin type. Generally speaking the ionic conductance is a complex function of light flux ($\phi(t)$), wavelength (λ), and the opsin's photocycle, membrane voltage, temperature (T), and intracellular and extracellular pH (Gradmann et al., 2011). We use a simplified empirical form for the channel conductance, introduced by Hodgkin and Huxley, expressing it as a product of a constant (g_0, in our case this

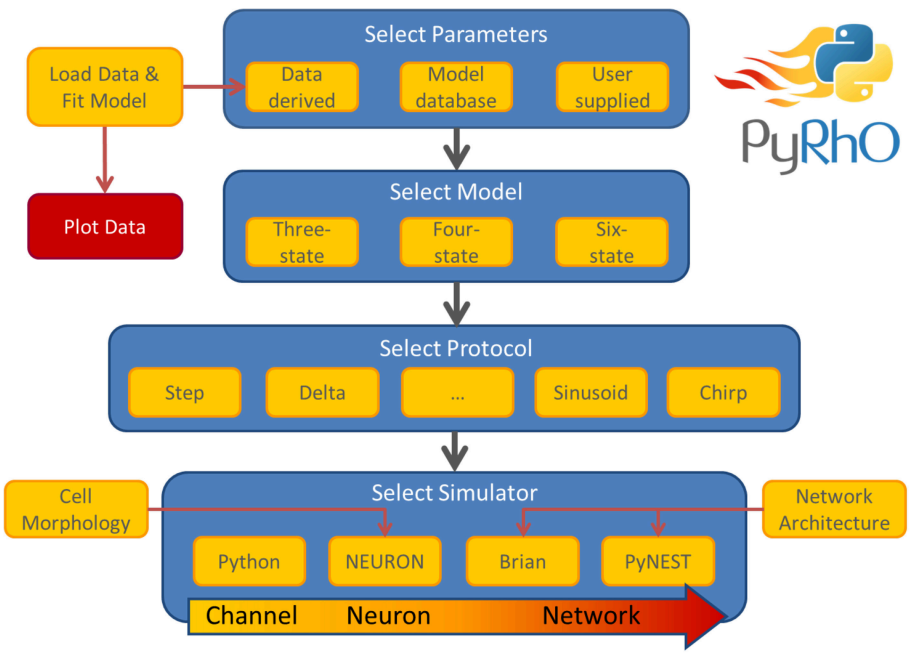

FIGURE 1 | Schematic of the PyRhO work-flow. Model parameters may be user supplied, initialized with defaults, or optionally derived from data. The user then selects the number of states in the model, the stimulation protocol and the simulation engine to start running virtual experiments.

TABLE 1 | Summary of opsin models.

States	Transitions	Parameters	Pros	Cons
3	3	11	Efficient analytic solution	Single exponential off-phase decay
4	7	17	Balance of detail and efficiency	Lacks short-pulse dynamics
6	9	19	Most detailed dynamics	Computationally expensive

is maximum conductance at $v = -70\,\text{mV}$), and a numerical coefficient ($f > 0$):

$$g = g(\phi, \lambda, v, T, pH, t) = g_0 \cdot f(\phi, \lambda, v, T, pH, t)\,, \qquad (2)$$

In this version of PyRhO we have implemented the photocycle and membrane voltage dependencies and assumed that these two contributions can be separated:

$$g = g_0 \cdot f_\phi(\phi, t) \cdot f_v(v)\,. \qquad (3)$$

These two dependences are considered to be the most relevant for physiological electrolyte conditions, when temperature and pH are considered to be fixed. Other dependencies will be implemented in the next version of PyRhO.

2.3. Photocycle Models

At the core of PyRhO are three functional Markov models of opsin kinetics, namely the three-, four- (Nikolic et al., 2009), and six-state (Grossman et al., 2013) models. We note that very similar models have been investigated in several other studies (Gradmann et al., 2002; Nagel et al., 2003; Hegemann et al., 2005; Ishizuka et al., 2006; Bamann et al., 2008; Ernst et al., 2008; Foutz et al., 2012; Williams et al., 2013) but used our earlier models as a starting point as we have since extended them and unified their notation. These models vary in complexity providing a range in the trade-off between biological accuracy and computational efficiency to choose from. The key features of these models, including an outline of their strengths and weaknesses, are summarized in **Table 1** with accompanying illustrations in **Figure 2**. Since their original formulation, the models have been extended to encompass additional parameter dependencies, better fit the experimental data and use a consistent notation, with the full model descriptions given in **Table 2**. An analytic solution for the three-state model was also calculated and is included in the Appendix.

Both four- and six-state models assume that there are two open states (O_1 and O_2, see **Figure 2**), with channel conductances g_{O_1} and g_{O_2}, respectively. The Photocycle factor in Equation (1) has the form:

$$f_\phi(\phi) = O_1 + \gamma O_2\,, \qquad (4)$$

where O_1 and O_2 are the fractions of opsins in two open states in the interval [0, 1], and $\gamma = g_{O_2}/g_{O_1}$. In contrast, the three-state model assumes only one open state (O) making the photocycle factor simply: $f_\phi(\phi) = O$.

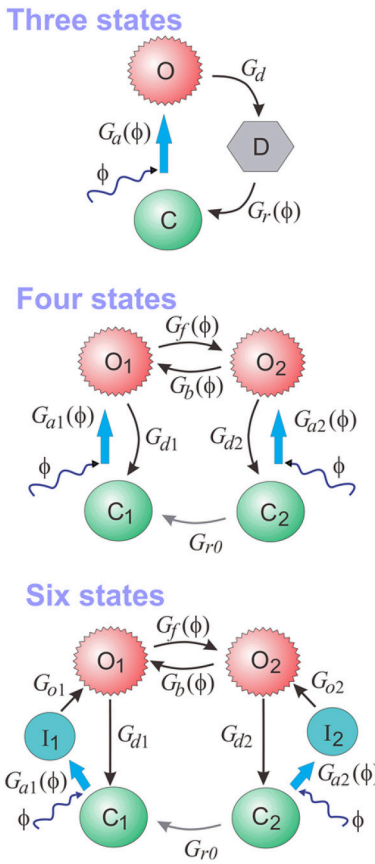

FIGURE 2 | The three-, four-, and six-state functional Markov models of opsins.

2.4. Voltage Dependence

Here, we assume that the membrane voltage affects only the ion-channel conductance but not the channel kinetics. By investigating experimental results for Channelrhodopsin-2 steady-state current vs. clamped voltage, the I–V curve shows inwardly rectifying behavior (Bamberg et al., 2008). We have previously found that an exponential function gives a good fit for this dependency (Grossman et al., 2011), therefore for the voltage factor in Equation (1) we adopt the form:

$$f_v(v) = \frac{v_1}{v - E} \cdot \left(1 - e^{-\frac{v-E}{v_0}}\right)\,, \qquad (5)$$

where v_0 and v_1 are fitting parameters, along with E, the channel's reversal potential. The exponential dependence on v transforms into a linear dependence for large values of v_0

TABLE 2 | Opsin model equations.

States	Functional state equations	Light-dependent transitions	Conductance factors
3	$\dot{C} = G_r(\phi)D - G_a(\phi)C$ $\dot{O} = G_a(\phi)C - G_d O$ $\dot{D} = G_d O - G_r(\phi)D$ $C + O + D = 1$	$G_a(\phi) = k_a \frac{\phi^p}{\phi^p + \phi_m^p}$ $G_r(\phi) = k_r \frac{\phi^q}{\phi^q + \phi_m^q} + G_{r0}$	$f_\phi(\phi) = O$ $f_v(v) = \frac{1 - e^{-(v-E)/v_0}}{(v-E)/v_1}$
4	$\dot{C_1} = G_{d1}O_1 + G_{r0}C_2 - G_{a1}(\phi)C_1$ $\dot{O_1} = G_{a1}(\phi)C_1 + G_b(\phi)O_2 - (G_{d1} + G_f(\phi))\,O_1$ $\dot{O_2} = G_{a2}(\phi)C_2 + G_f(\phi)O_1 - (G_{d2} + G_b(\phi))\,O_2$ $\dot{C_2} = G_{d2}O_2 - (G_{r0} + G_{a2}(\phi))\,C_2$ $C_1 + O_1 + O_2 + C_2 = 1$	$G_{a1}(\phi) = k_1 \frac{\phi^p}{\phi^p + \phi_m^p}$ $G_f(\phi) = k_f \frac{\phi^q}{\phi^q + \phi_m^q} + G_{f0}$ $G_b(\phi) = k_b \frac{\phi^q}{\phi^q + \phi_m^q} + G_{b0}$ $G_{a2}(\phi) = k_2 \frac{\phi^p}{\phi^p + \phi_m^p}$	$f_\phi(\phi) = O_1 + \gamma O_2$ $f_v(v) = \frac{1 - e^{-(v-E)/v_0}}{(v-E)/v_1}$
6	$\dot{C_1} = G_{d1}O_1 + G_{r0}C_2 - G_{a1}(\phi)C_1$ $\dot{I_1} = G_{a1}(\phi)C_1 - G_{o1}I_1$ $\dot{O_1} = G_{o1}I_1 + G_b(\phi)O_2 - (G_{d1} + G_f(\phi))\,O_1$ $\dot{O_2} = G_{o2}I_2 + G_f(\phi)O_1 - (G_{d2} + G_b(\phi))\,O_2$ $\dot{I_2} = G_{a2}(\phi)C_2 - G_{o2}I_2$ $\dot{C_2} = G_{d2}O_2 - (G_{r0} + G_{a2}(\phi))\,C_2$ $C_1 + I_1 + O_1 + O_2 + I_2 + C_2 = 1$	$G_{a1}(\phi) = k_1 \frac{\phi^p}{\phi^p + \phi_m^p}$ $G_f(\phi) = k_f \frac{\phi^q}{\phi^q + \phi_m^q} + G_{f0}$ $G_b(\phi) = k_b \frac{\phi^q}{\phi^q + \phi_m^q} + G_{b0}$ $G_{a2}(\phi) = k_2 \frac{\phi^p}{\phi^p + \phi_m^p}$	$f_\phi(\phi) = O_1 + \gamma O_2$ $f_v(v) = \frac{1 - e^{-(v-E)/v_0}}{(v-E)/v_1}$

which cause the exponent to be small and the expression in Equation (5) reduces to $f_v(v) \approx v_1/v_0 = const$, i.e., no direct dependence on membrane voltage, which may be a more appropriate form for some opsins. The expression given by Equation (5) therefore generalizes to both cases for appropriate choices of the parameters v_0 and v_1.

Furthermore, since the voltage dependence factor is defined to be equal to 1 at −70 mV ($f_v(-70\text{ mV}) := 1$), the value of v_1 is related to the other parameters through the following equation:

$$v_1 = \frac{70 + E}{e^{\frac{70+E}{v_0}} - 1} \tag{6}$$

This relationship is used as a constraint in the fitting procedures described below.

2.5. Model Fitting

PyRhO incorporates novel fitting algorithms with which each of the opsin models may be parameterized when given an appropriate set of data. The fitting algorithms use the `lmfit` module (Newville et al., 2014) which in addition to providing access to a variety of optimization algorithms, enables numerical bounds to be placed on parameter values as well as algebraic constraints. Parameters may also be manually fixed if, for example, they have been directly measured experimentally. Once the algorithm has finished, a `Parameters` object is returned, populated with the extracted parameter values which may then be saved or used as the basis for simulations. Plots of photocurrents simulated with these derived parameters are drawn over the experimental data (with residual error) to allow for visual comparison of the resultant fits.

2.5.1. Characterization Data

In order to characterize each model, a set of voltage-clamped photocurrents are required, ideally collected from HEK cells to eliminate the confounding effects of other ion channels which may be present in neurons. To capture all currently modeled variable dependencies, data from three stimulation protocols are necessary, listed below by the model properties which they reveal. In the event of scarce data or uncharacterized variables which the user does not intend to vary, we describe the minimum set of data for the fitting procedure below and discuss the implications for the resultant model.

- **Voltage dependence**: $\{E, v_0, v_1\}$. Long light pulse (fixed flux) to steady-state, vary voltage clamp potential (e.g., in steps of 30 mV: $V_{\text{clamp}} = \{-100, -70, -40, -10, 20, 50, 80\}$ mV, $n \geq 5$). Voltage clamp values should not be too close to E as this may cause distortions in the fitting algorithms. The software will automatically find the plateau values I_{ss}, plot I_{ss} vs. V_{clamp}, and find the fitting parameters for the function $f_v(v)$ given by Equation (5). An example is shown in **Figure 3**.
- **Recovery rate**: $\{G_{r0}\}$. Two long light pulses with varying inter-pulse-interval (IPI), Voltage clamp: −70 mV. Light on (first

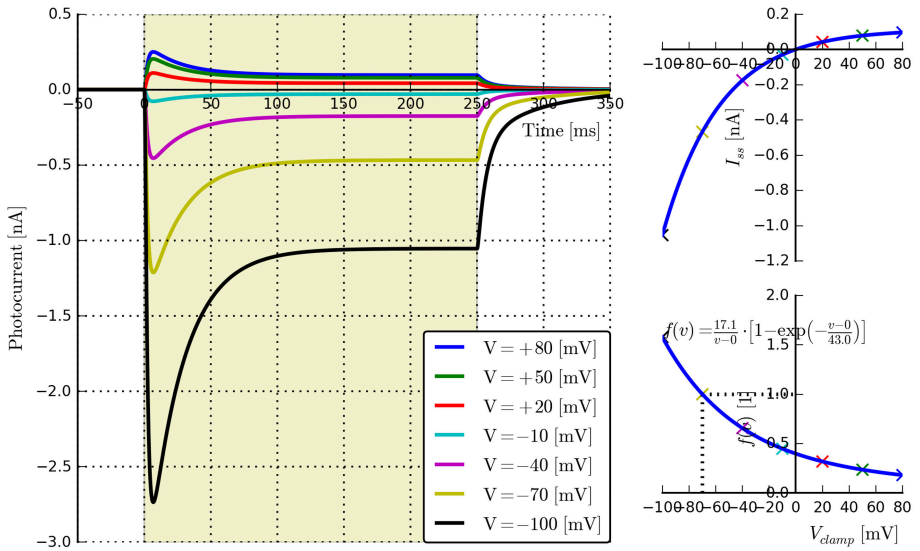

FIGURE 3 | Photocurrent plots from the "rectifier" protocol with accompanying steady-state current and fitted *f*(*v*) plots.

pulse)—light off for e.g., $t_{\mathrm{IPI}} = \{0.5, 1, 2.5, 5, 10\}$ s—light on (second pulse). The software will automatically find the peak values for each recording, align the data to the end of the first pulse and fit an appropriate exponential curve of the form $I_{\mathrm{peak}}(t) = I_{peak0} - a \cdot e^{-G_{r0} \cdot t_{\mathrm{IPI}}}$. We note here that this expression is strictly speaking correct only when both O_1 and O_2 states are empty. Consequently a is left as a free fitting parameter and very short values for t_{IPI} should be avoided to prevent the distortions caused by the faster transitions. An example for wild-type ChR2 is given in **Figure 4**, where $t_{\mathrm{IPI}} \gtrsim 100$ ms.

- **Flux dependence**: *Off-curve*: $\{G_{d(1,2)}, [G_{f0}, G_{b0}]\}$; *On-curve*: {All other parameters}. Voltage clamp (preferably): −70 mV, long pulse to steady-state, (e.g., $T \approx 500$ ms) plus decay of off-curve. Vary light intensity from near threshold to saturation (e.g., $\phi = \{0.1, 0.5, 1, 5, 10, 50, 100\}$ mW/mm^2, $n \geq 5$). The recorded off- and on-curves are automatically fitted. An example set is shown in **Figure 5** with more details of the algorithm given in Appendix Section (Model-Dependent Fitting Procedures).

Additionally the six-state model requires one or more very short pulses in order to characterize the opsin activation rates which model the lag in transitioning to conductive states upon light stimulation:

- **Opsin activation rate**: $\{G_{o1}, G_{o2}\}$ One or more *short* pulses, voltage clamp: −70 mV. Vary pulse length, e.g., 0.5, 1, 2, 3, varied up to 10 ms. PyRhO will automatically find the time of the peak current and use an iterative formula to estimate G_{o1}. We initially assume $G_{o2} = G_{o1}$. Further details of the algorithm are given in Appendix section Six-state opsin activation rate fitting (Step 1b.).

All light pulses should be "rectangular" (step functions) in that they have a sharp onset and offset. Examples of each protocol are included in PyRhO with illustrations provided in **Figures 3–7**. The duration of the on- and off-phases should also be kept approximately equal since the optimizer will effectively weight the contributions of each according to the relative numbers of data points. Additional parameter dependencies will be added in the future which may require additional data sets for a full characterization of the models.

2.5.2. Minimal Data Requirements

In general, the most important data are those described for characterizing the flux dependence, which may be considered to be the "minimal set." If this set consists of only a single photocurrent, the fitting algorithms will fix the parameters which model the flux dependence (ϕ_m, p and q) to the initial values supplied (along with fixing those describing other uncharacterized variables) and tune the remaining parameters to return a model fit for that specific flux. This is not recommended however, as the model is under-constrained by the data (typically resulting in a poorer fit than when using a whole set of photocurrents) and is unlikely to generalize well to new experimental conditions. For best results, the flux dependence photocurrents should be measured at light intensities spanning several orders of magnitude as described above.

If variations in other parameters or short pulses are of interest then the additional data should (ideally) be collected as described. However, if obtaining the data for a full characterization of the model is not possible, the pre-set default values should be adequate for most practical purposes.

2.5.3. Data Format

Each voltage-clamp recorded photocurrent should be loaded into a `PhotoCurrent` object as follows:

```
pc = PhotoCurrent(I=i0, t=t, pulses=[[t_on,
                  t_off]], phi=2e15, V=-70)
```

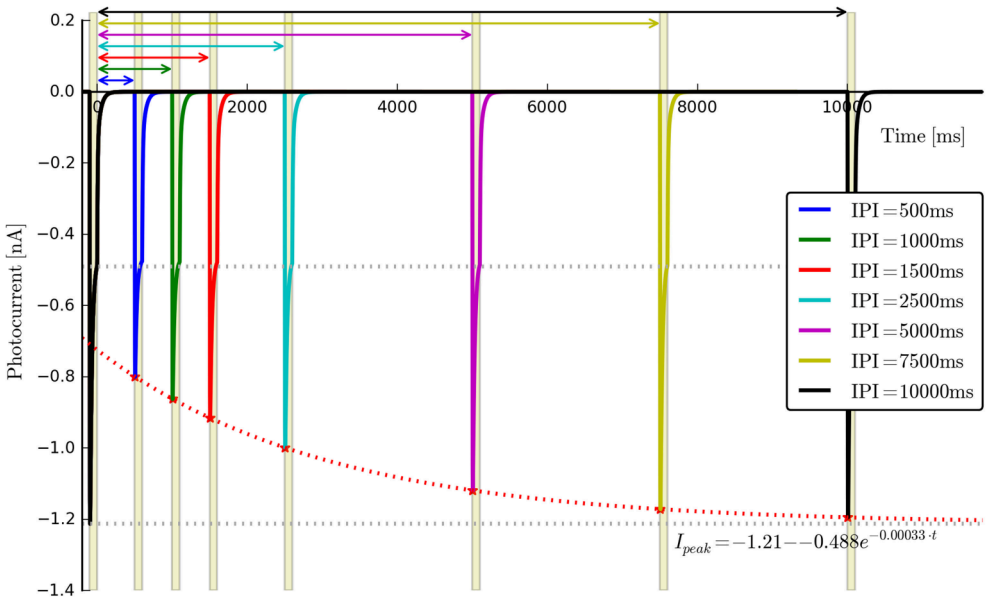

FIGURE 4 | Photocurrent plots from the "recovery" protocol with fitted peak recovery function.

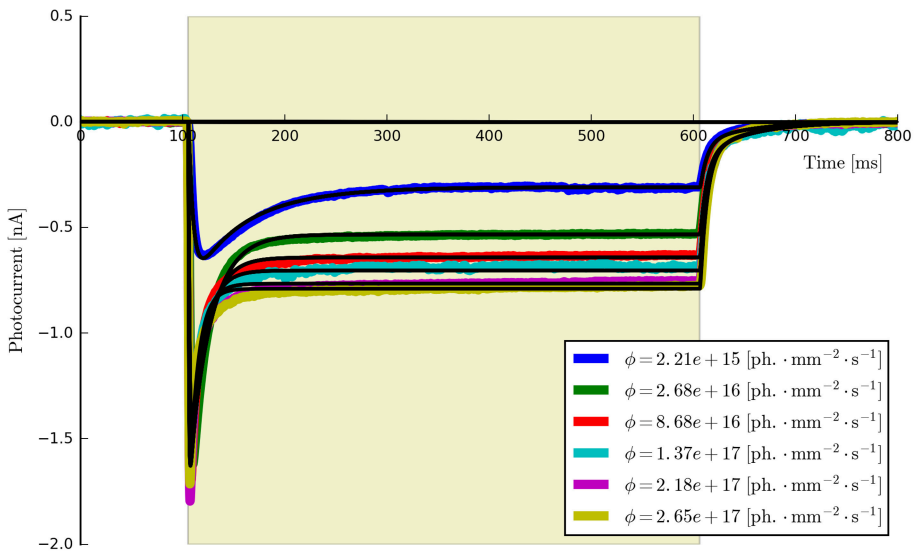

FIGURE 5 | The six-state model fit to a set of six ChR2 photocurrents using the same model parameters.

Here, `I` is the array of photocurrent values in nanoamperes, `t` is the corresponding array of recording times (or a scalar representing the time-step) in milliseconds, `pulses` is a nested list of n lists (or an $n \times 2$ array), where n corresponds to the number of pulses and each inner list contains the on-time and off-time for each pulse in milliseconds, `phi` represents the stimulating flux value in photons $\cdot$ mm$^{-2} \cdot$ s^{-1} and `V` is the clamp voltage in millivolts (or "`None`" if the voltage was not clamped).

The `PhotoCurrent` class contains methods which automatically check the data and extract the key features from it, which may then be accessed as properties of the object with the `.` operator. Any properties which are data-derived are suffixed with "_," for example, the peak and steady-state current are accessed with `pc.peak_` and `pc.ss_`, respectively. These photocurrents may easily be plotted, along with their main features and the light stimulus using the `plot()` method.

The `PhotoCurrent` objects are then combined into `ProtocolData` objects. These sets of photocurrents also provide several convenient methods for plotting and extracting parameters from the data set as a whole.

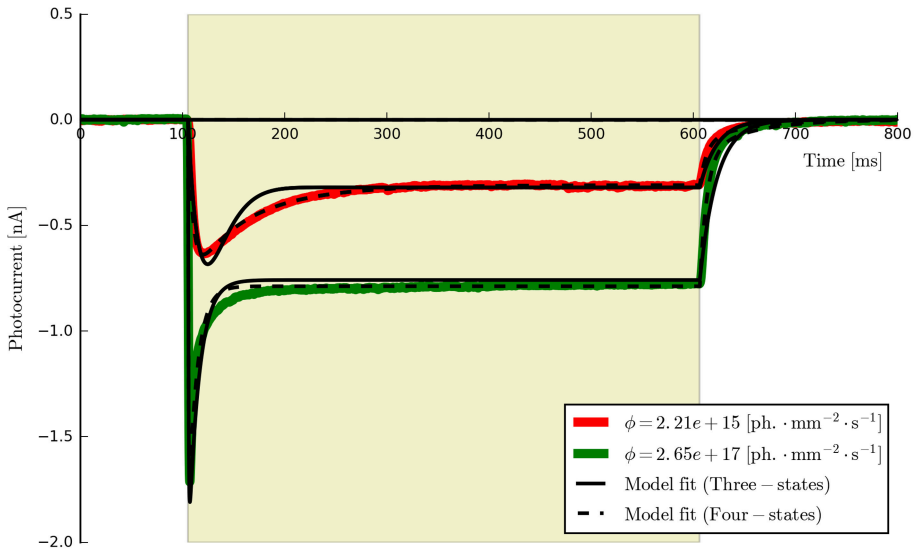

FIGURE 6 | The three- and four-state models fitted to the first two ChR2 photocurrents for comparison (the six-state fits are omitted since they are almost indistinguishable from the four-state fits at this time scale). While both models capture the major features of the data, the four-state (and six-state) models produce a higher quality of fit, particularly during the post-peak inactivation phase (where the three-state curves tend to over-shoot the data) and the deactivation (off) phases which are better described by a double rather than a single exponential decay process.

```
stepPD = ProtocolData(protocol="step",
   nRuns=1, phis=[1e14,1e15,1e16,1e17,1e18],
                                    Vs=[-70])

for iPhi, phi in enumerate(phis):
    for iV, V in enumerate(Vs):
        pc = PhotoCurrent(Is[iPhi][iV], t,
                          pulses, phi, V)
        stepPD.addTrial(pc)
```

Finally, the data sets are combined into a dictionary using the protocol names as keys:

```
ChR2DataSet = {"step" : stepPD,
               "recovery" : recovPD,
               "rectifier" : rectiPD,
               "shortPulse" : shortPD}
```

This dictionary contains all the data necessary to parameterize all three models, however, if only the three and four-state models are of interest then the `"shortPulse"` protocol may be omitted.

2.5.4. Fitting Procedure and Algorithms

Once the data have been loaded into the appropriate structures, the fitting algorithms may be called with the `fitModels()` function.

```
fp = fitModels(ChR2dataSet, nStates=6,
                  params=initialParams)
```

This procedure returns a `Parameters` object (from the `lmfit` module) with the calculated values and plots the resultant model fits over the experimental photocurrents. The entire set of ChR2 data are shown fitted to the six-state model for each flux value spanning two orders of magnitude ($\phi = [2.21 \times 10^{15},\ 2.65 \times 10^{17}]$ photons $\cdot$ mm$^{-2} \cdot$ s^{-1}) with the same set of parameters in **Figure 5**. The lowest and highest intensity photocurrents are also shown in **Figure 6** with the model fits for both the three- and four-state models for direct comparison. The six-state model fits are not replotted here as they only exhibit a significant difference to the four-state fits for short pulses, as illustrated in **Figure 7**.

Having fit a model, it may be easily characterized by plotting how the light-dependent transition rates vary as a function of flux (based on the Hill equation) along with light-independent transition rates as shown in **Figure 8**. An individual fit is shown in more detail with the residual error for $\phi = 2.21 \times 10^{15}$ photons $\cdot$ mm$^{-2} \cdot$ s^{-1} in **Figure 9**. To provide insight into the model's kinetics, PyRhO also offers state variable plots. The evolution of the six-state model corresponding to the fit in **Figure 9** is given in **Figure 10**.

In general terms, the fitting algorithm first finds the model-independent variables such as the dark recovery rate and voltage dependence factors, proceeding through "off-curve" parameters by fitting a double exponential decay function, optionally fitting opsin activation rates for the six-state model and finally optimizing across a set of "on-curves" to find any remaining parameters. Due to the inherent variability and imprecision in experimental measurements there is an optional second optimization phase over the entire set of photocurrents simultaneously. The values found for the dark parameters $\{G_{d(1,2)}, [G_{f0}, G_{b0}]\}$ (and opsin activation rates $G_{o(1,2)}$ if relevant) are used as the initial values, lower and upper bounds are calculated as 50 and 200% of these values, respectively (set by a hyperparameter) and the model is then re-optimized to achieve an overall better fit. The main sub-routines of the algorithm are

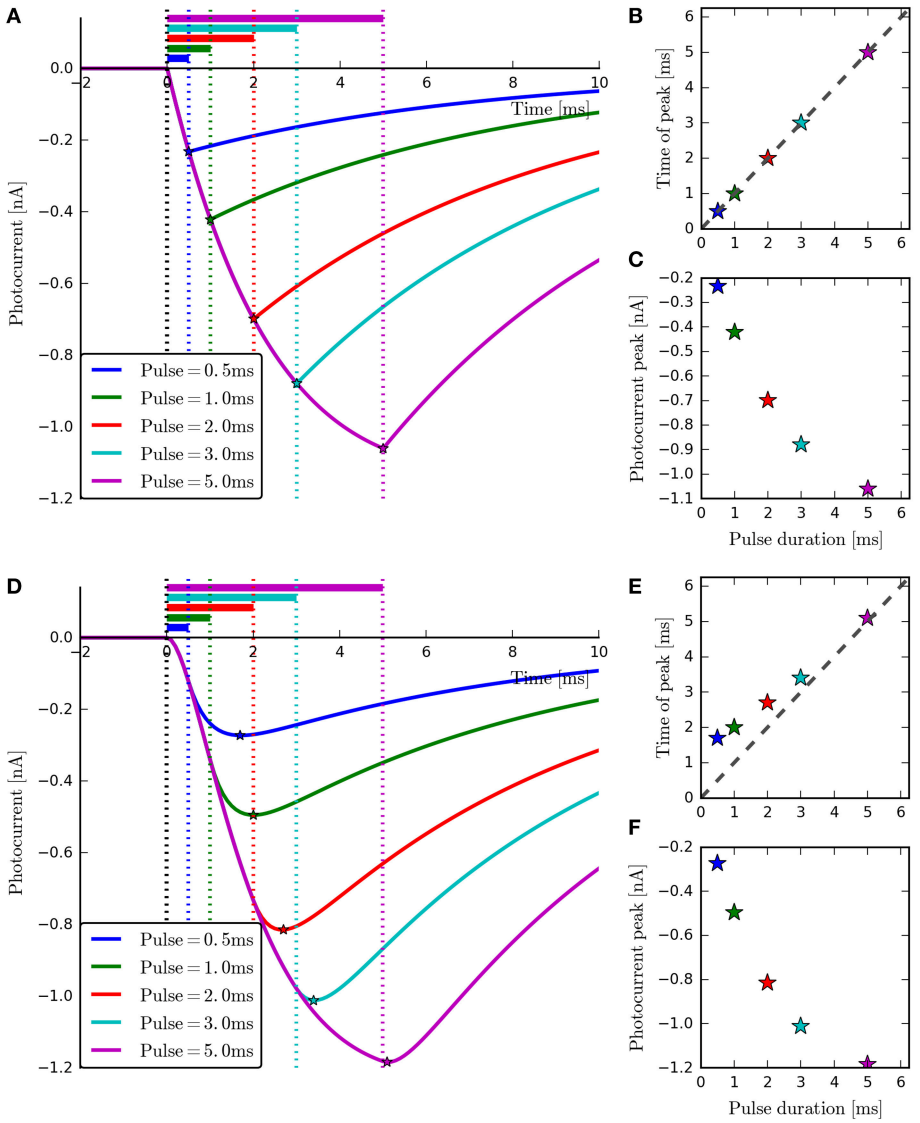

FIGURE 7 | The four-state and six-state models simulated with ChR2-derived parameters for short-duration pulses. With the four-state photocurrents **(A)**, the peak occurs at the end of the on-phase. This can be seen most clearly in the plot of Pulse duration vs. Time of peak **(B)** where all points lie on the diagonal. For the six-state photocurrents however, the peaks lag behind the end of the illumination periods slightly **(D)**, due to the transitions to and from the model's extra intermediate states. The inactivation phases **(A,D)** and magnitude of the peaks **(C,F)** can be seen to be very similar for both models. For the six-state model, the peak-lag effect can also be observed to diminish as the pulse duration increases **(E)**, demonstrating the practical equivalence of the two models for long stimulation periods.

given in **Algorithm 1** with more detail for each process given in the Appendix.

When fitting the three-state model, a double exponential is fit (with two corresponding decay rates G_{d1} and G_{d2}) which are then weighted by their coefficients (I_{slow} and I_{fast}) and combined to form a single exponential. The mean of these values is then calculated across a set of N photocurrents (Equation 7) and this value is then used in subsequent parts of the fitting algorithm.

$$G_d = \frac{1}{N}\sum_{n=1}^{N}\frac{I_{\text{slow}_n} \times G_{d1_n} + I_{\text{fast}_n} \times G_{d2_n}}{I_{\text{slow}_n} + I_{\text{fast}_n}}\,. \tag{7}$$

2.5.5. Verification

In order to test the algorithms, synthetic data was generated using the parameter values derived from fitting the six-state model to the ChR2 experimental data. The fitting procedure was then applied to these synthetic photocurrents to compare the newly derived parameters with the known values used to generate the synthetic data. The results of these two fitting processes using the "`powell`" optimization algorithm are shown in **Table 3** (along with the values used as initial estimates).

We first note that many of the computed parameters are very close to the true (original) values (all but two are within ±5% of the original values), especially in the context of the

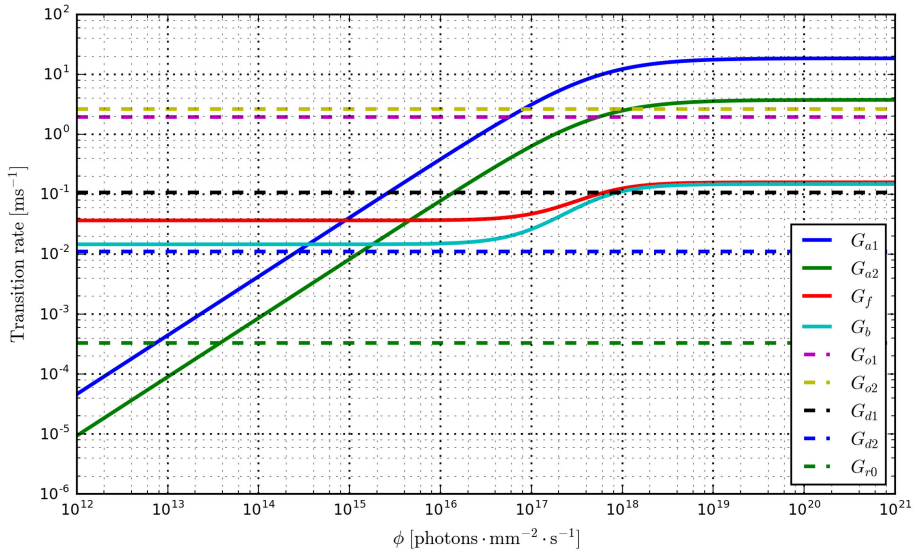

FIGURE 8 | Transition rate plots for the six-state model fit to ChR2 data (on log-log axes) where light-dependent transitions are shown with solid lines and light-independent transitions are shown with dashed lines.

Algorithm 1 Fitting algorithm

function FITMODEL({*DataSet*}, {*initParams*})
  if "*rectifier*" in {*DataSet*} **then**
    $(E, v_0, v_1) \leftarrow$ fitVoltageRectifier($\mathbf{V_c}$, $\mathbf{I_{ss}}$)
  $g_0 \leftarrow$ fitConductance(v, E, $\max_\phi(\mathbf{I_p})$)
  if "*recovery*" in {*DataSet*} **then**
    $G_{r0} \leftarrow$ fitPeakRecovery($\mathbf{t_p}$, $\mathbf{I_p}$)
  $(G_{d(1,2)}, [G_{f0}, G_{b0}]) \leftarrow$ fitOffCurves($\{I_\phi[t_{off}:]\}$)
  $(\phi_m, k_{(1,2)}, p, [G_{r1}, G_{f0}, k_f, q, G_{b0}, k_b, \gamma, G_{o(1,2)}])$
    $\leftarrow$ fitOnCurves($\{I_\phi[t_{on}:t_{off}]\}$)
  if postFitOptimization is `True` **then**
    ({*All parameters*}) $\leftarrow$ fitCurves($\{I_\phi\}$)

degree of experimental noise and measurement error which would typically accompany recordings of real neuronal data. One notable exception is G_{o2} which is hard to fit in a single-pulse protocol since all opsin models are assumed to be in their fully dark-adapted (ground) state i.e., $C_1 = 1$.

While there are other differences between some of the original and computed parameters, these may potentially be accounted for by numerical precision issues and the high-dimensional parameter space of the six-state model being under-constrained by the data. For example, a decrease in one parameter may be compensated for by an increase in another such that only latent variables are affected and the fit in the observable current is still good. Inspecting the model fit plots appears to confirm this, as the residual error is very low across the whole set of generated verification photocurrents—at most ±0.5% of the steady-state current and usually considerably less. The entire set of photocurrents fitted to the synthetic data are plotted in **Figure 11** and show a very close correspondence to the target (synthetic) photocurrents across the whole set of stimulus intensities.

2.6. Computational Simulation

2.6.1. Simulations Procedure

To programmatically simulate the opsin, a model, protocol and simulator object must first be created (see **Figure 1**). The model and protocol are then loaded into the simulator which configures the simulation environment for that particular choice of opsin and protocol (e.g., by setting the numerical time-step according to the shortest stimulation period). The simulator object can then be run and plotted with the appropriate methods as shown in the following example, where a six-state model is used with the ChR2 parameters and run through the rectifier protocol (described below). In this example the protocol is run on the NEURON simulator, (rather than the default Python simulator) and the default ChR2 parameters are loaded from the module's `modelFits` dictionary, which contains some pre-fit model parameter sets for several common opsins.

```
from pyrho import *

nStates = "6"       # 3, 4 or 6
ChR2params = modelFits[nStates]["ChR2"]
RhO = models[nStates](ChR2params)
Prot = protocols["rectifier"]()
Sim = simulators["NEURON"](Prot, RhO)

Sim.run()
Sim.plot()
```

Alternatively, when using the PyRhO GUI, the model, protocol and simulator are simply selected from drop-down lists, (optionally parameters may be changed),

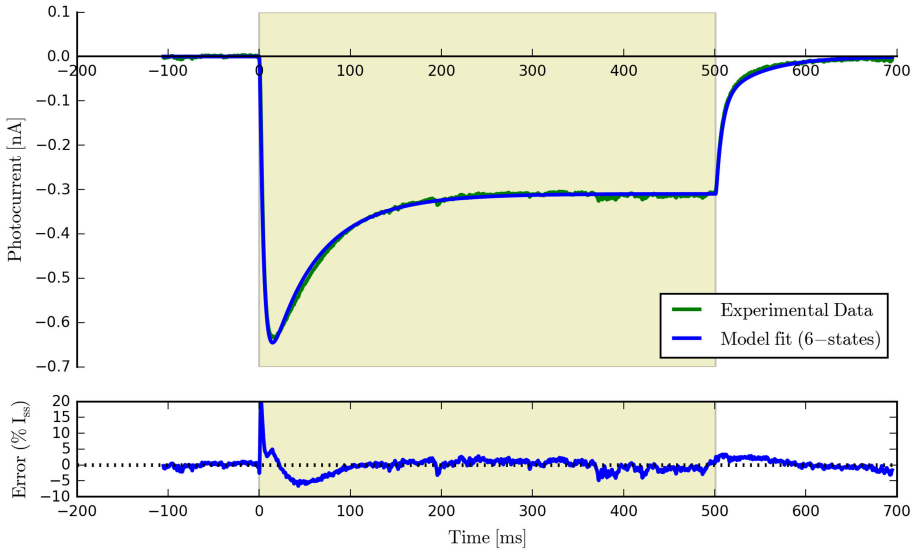

FIGURE 9 | An example six-state model fit to ChR2 data (at $\phi = 2.21 \times 10^{15}$ photons $\cdot$ mm$^{-2} \cdot$ s^{-1}) with the accompanying residual error expressed as a percentage of the steady-state current.

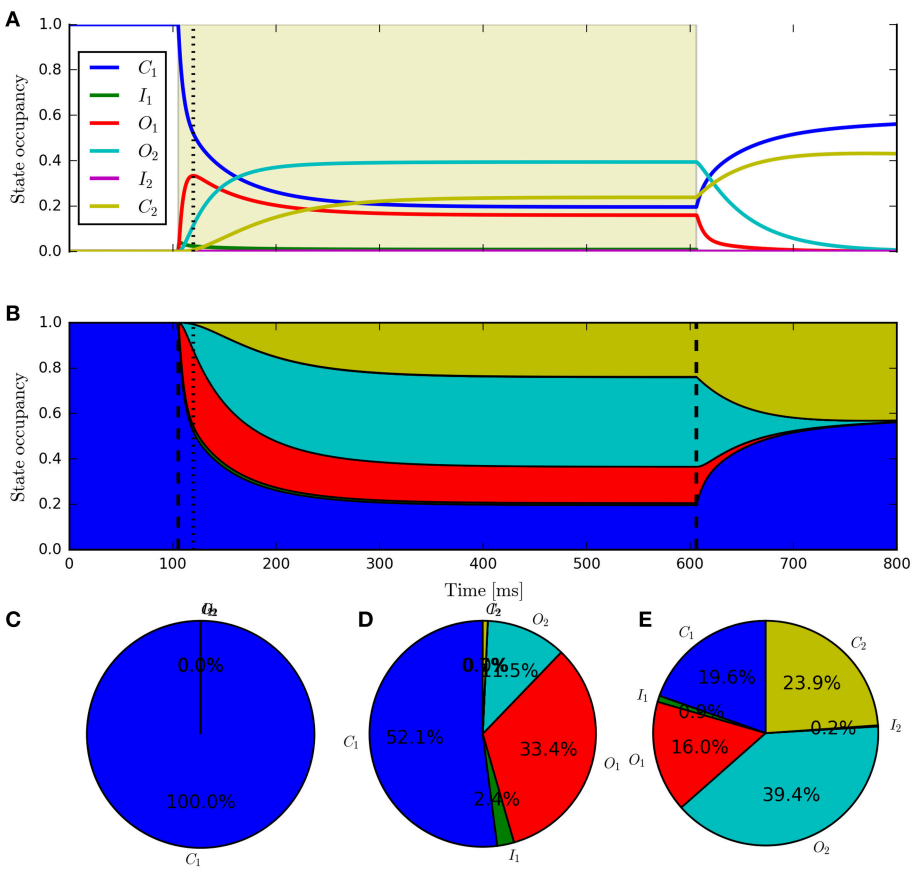

FIGURE 10 | The six-state model's internal states. This figure corresponds to the example fitting plot (**Figure 9**) and shows the evolution of the internal states through time **(A,B)** shows alternative representations of the same data) along with the occupancy proportions at the initial conditions **(C)**, peak **(D)** and steady-state **(E)**.

TABLE 3 | Comparison of parameters found in fitting to those used to generate the fitting data.

Parameter	Initial	Experimental	Computed	Difference
g_0 (pS)	2.5e4	2.76e+04	2.77e+04	+0.0464%
γ (1)	0.05	8.33e-16	0.00369	Δ: +0.00369
ϕm (ph. $\cdot$ mm^{-2} $\cdot$ s^{-1})	3.5e17	5.07e+17	5.02e+17	−0.983%
k_1 (ms^{-1})	10	18.5	18.2	−1.3%
k_2 (ms^{-1})	3	3.75	4.07	+8.68%
p (1)	1	0.982	0.981	−0.0921%
G_{f0} (ms^{-1})	0.04	0.0365	0.0365	+0.149%
k_f (ms^{-1})	0.1	0.121	0.121	−0.601%
G_{b0} (ms^{-1})	0.02	0.0146	0.0143	−1.99%
k_b (ms^{-1})	0.15	0.133	0.131	−1.31%
q (1)	1	1.45	1.45	−0.196%
G_{o1} (ms^{-1})	2	1.93	1.93	+0.0256%
G_{o2} ([ms^{-1})	2	2.65	3.38	+27.7%
G_{d1} (ms^{-1})	0.1	0.108	0.108	+0.453%
G_{d2} (ms^{-1})	0.01	0.0111	0.0115	+3.96%
G_{r0} (ms^{-1})	0.00033	0.00033	0.00033	−0.0585%
E (mV)	0	0	3.9e-08	Δ: +3.9e-08
v_0 (mV)	43	43	43	−2.16e-08%
v_1 (mV)	17.1	17.1	17.1	+0.00889%

then simulated and plotted by clicking the "Run" button.

Each type of object will be initialized with default parameters (in the form of `Parameters` objects) unless passed a different set upon initialization e.g., `RhO = models[nStates](params6s)`. Alternatively, parameters may be set after creation using methods such as `.setParams()` or `.updateParams()` for partial sets.

2.6.2. Protocols

PyRhO comes with several preconfigured and customizable simulation protocols for exploring the dynamics of the models. These include typical system analysis stimuli such as delta functions, step functions and sinusoids, as well as chirps and specialized protocols designed to probe particular features of the opsins including voltage-dependence (`rectifier`), opsin activation (`shortPulse`) and dark recovery (`recovery`).

2.6.3. Simulators

PyRhO's simulation layer serves to perform house-keeping tasks necessary to prepare different simulation environments to use a particular opsin model in a "system" of interest and apply a particular protocol to it. Currently, three simulators are available in PyRhO: Brian2 for neural networks, NEURON for detailed morphological neurons and (pure) Python for basic opsin channel dynamics. This selection of simulators provides PyRhO with a way to seamlessly span multiple scales of modeling with the same parameterized opsins; from individual channels to whole brain regions.

When using simulators other than "`Python`," additional parameters may be specified. For example, the `NEURON` simulator has additional parameters such as "`v_init`" (the cell membrane potential initialization value), "`CVode`" (a boolean value for activating variable time-step solvers), and "`cell`" (a hoc file specifying the neuron's morphology). This allows existing simulations created in `NEURON` to be conveniently transfected (augmented with opsins) and run within the PyRhO framework. While models may be fit to data and then seamlessly inserted into one of these simulators within the same environment, if desired the `NMODL` files and Brian equations may be accessed and exported as a starting point for creating stand-alone simulations.

An example of implementing opsins within the `NEURON` environment is shown in **Figure 12** where ChR2 expressing cells are illuminated with a 150 ms light pulse. The six-state equations were used to model the ChR2 additions and implemented via the `NMODL` file `RhO6.mod`, adapted from the description in Grossman et al. (2013).

PyRhO also incorporates the Brian2 spiking neural network simulator. The opsin is represented with a set of ODEs which use the parameters specified in the `RhodopsinModel` object. As an example we show simulation results for a neural network which consists of 140 leaky integrate-and-fire neurons, separated into three feed-forward layers. The first group has neurons which express ChR2 and that layer has a set of random connections with the next layer, with 20% connectivity and 1ms conduction delays (with the same specifications for connectivity between the second and third layers). **Figure 13** shows raster plots of the spiking neurons in all three layers.

2.7. Graphical User Interface

To make PyRhO usable without any programming background, a graphical user interface (GUI) has been written which runs in Jupyter (formerly IPython; Pérez and Granger, 2007). This runs in a browser-based notebook meaning that it could be easily configured as a server and made accessible to an entire laboratory or classroom without requiring local installations on each machine. Since both figures and text results appear embedded in the notebook after the code used to produce them, this makes the interface self-documenting and a particularly useful means of sharing models (Topalidou et al., 2015).

A screenshot of the GUI is given in **Figure 14** showing the simulation tab for the three-state model. In addition to the parameter fields, the model states diagram is also embedded in the GUI with the equations which describe the opsin's behavior rendered in LaTeX. Similarly, the parameters for each protocol and each simulator are shown on other tabs for easy modification of their values. On the fitting tab, there are also sub-tabs for each of the models, with their respective sets of parameters including tick-boxes to fix parameters and fields for numerical bounds and algebraic constraints to be passed to the fitting algorithms.

3. DISCUSSION

PyRhO has been written to be an integrated suite of intuitive, flexible, open-source and multi-scale computational tools for analysing and simulating opsins. In keeping with the open-source community's ethos, it builds upon existing libraries for numerical (NumPy), scientific (SciPy, lmfit) and plotting

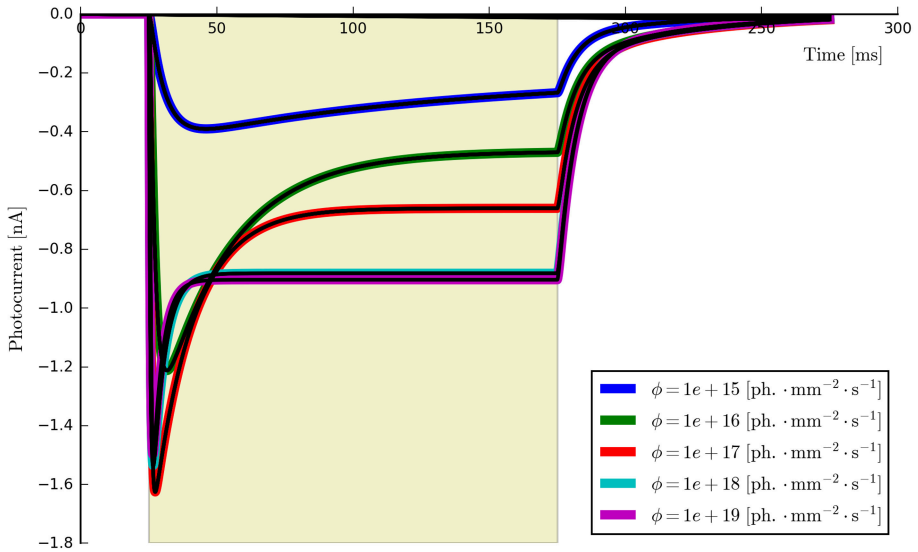

FIGURE 11 | The six-state model (re-)fitted to synthetic data generated from parameters derived from six ChR2 photocurrents. It can be seen that the model fits (black lines) almost perfectly fit each synthetic photocurrent using a unified parameter set.

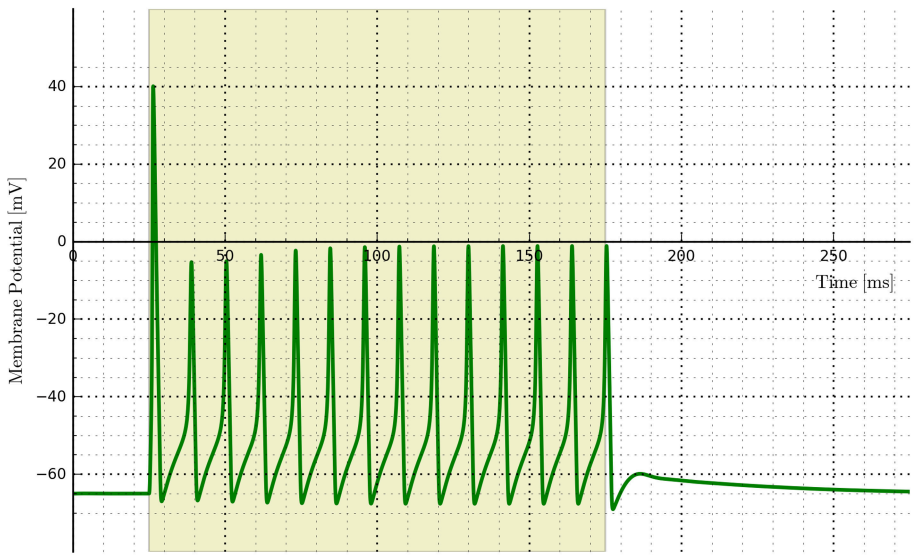

FIGURE 12 | Cell membrane potential for a simple Hodgkin–Huxley neuron transfected with a six-state opsin model of ChR2 and simulated in NEURON.

routines (Matplotlib). It also incorporates several freely available simulators, namely NEURON and Brian2, with work to incorporate PyNEST underway.

3.1. Classes of Opsin

While the models were developed with rhodopsins and the fitting and simulation demonstrations are illustrated using ChR2 data, in principle the tools should work just as well with other classes of opsin. Essentially the opsin models represent non-linear dynamical systems (second order for the three-state model and $n - 1^{th}$ order for the n-state model in general). As such they are capable of capturing the three main classes of dynamics in response to a step input: under-damped, over-damped and oscillatory (ringing) currents. While PyRhO can fit and simulate all three cases, interestingly, only the first two types of response have been observed so far (for low and high intensity stimulation, respectively).

For inhibitory opsins observed so far, while the sign of the current changes, the dynamics remain qualitatively the same, meaning that the fitting and simulation routines should be just

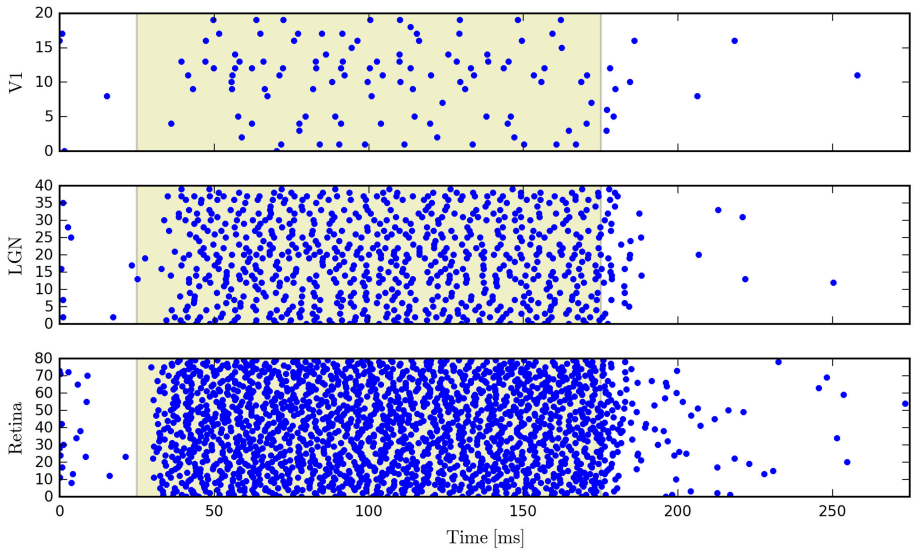

FIGURE 13 | Raster plots for a three-layer network of leaky integrate-and-fire neurons simulated with Brian2 where the first layer has been transfected with a six-state model of ChR2.

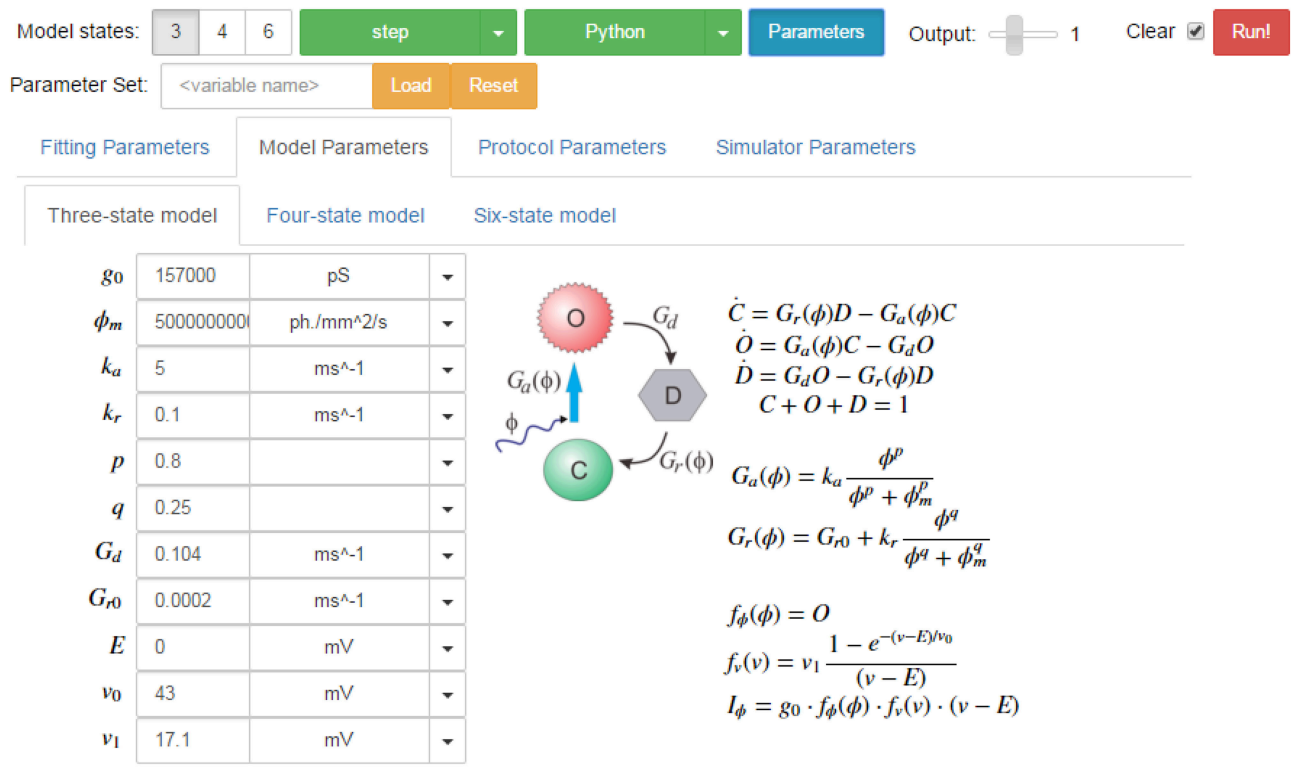

FIGURE 14 | Screen shot of the PyRhO GUI (in expanded view) showing the run bar at the top for running simulations and the models tab below where parameters may be adjusted.

as effective. However, for fitting, it may be necessary to adjust the starting values for some parameters to help the algorithms achieve good results. The main changes we anticipate would be in parameters that are extraneous to the the core dynamics (that is, those not described by the differential equations) such as the reversal potential, (E) which may be measured through standard

electro-physiological techniques and possibly the parameters tuning the voltage rectification curve (v_0 and v_1).

3.2. Extensibility

Given the way that PyRhO has been constructed around several layers of abstraction, extending the capabilities with more models, simulators or protocols is relatively straightforward. Nested classes (and sub-classes) act as templates guiding development while inheritance of methods and attributes facilitates code reuse in line with open-source software development best practices. As more data is collected, the library of parameterized models may be contributed to, making more characterized opsin variants available for other optogeneticists to simulate.

In broader terms, PyRhO has potential as a framework to be generalized to incorporate other types of stimulation, including for example electrical and magnetic. The simulator and protocol layers could potentially remain largely unaltered allowing it to relatively easily grow into a neural stimulation platform with neuro-engineering applications beyond the scope of just optogenetics.

3.3. Limitations

The fitting algorithms rely upon optimization methods to extract several parameters and hence are subject to the standard issues associated with such procedures, including sensitivity to initial conditions and settling in local minima. To ameliorate these issues, the `lmfit` module is used to provide several useful facilities including imposing bounds on the parameters and algebraic constraints. Individual parameters may also be fixed and the fitting procedures rerun to optimize over the remaining free parameters, possibly with a new set of initial conditions. Measuring and then fixing physiological parameters in this way, such as the reversal potential E, will help the optimization algorithms and considerably improve the resultant model fit.

Additionally there are limitations due to the more specific nature of the models used. For example, the routine for estimating g_0, the biological scaling parameter for the cell's conductance, is systematically under-estimated. This should be the maximum conductance of the cell (voltage clamped to −70 mV) assuming full occupancy of the (primary) open-state. However, in simulations this condition is never achieved in the models, with maximum occupancy dependent on the model parameters, typically found to be around 0.8 for wild-type ChR2 (Nikolic et al., 2009). We therefore calculate a conservative correction factor to yield a better (albeit imperfect) estimate and make the user aware of the issue so that they may override and fix the values returned from the fitting function as necessary.

We note however that these limitations are not major, especially in the context of experimental and measurement inaccuracies and so do not significantly detract from the usefulness of PyRhO's fitting algorithms.

3.4. Future Work

One natural development for PyRhO would be to extend the models (and fitting algorithms) to include additional parameter dependencies defined in the introduction, Equation (3), such as spectral absorption $f_\lambda(\lambda)$, temperature $f_T(T|Q_{10})$, and pH $f_{pH}(pH_{int}, pH_{ext})$ factors for the channel conductance, as well as the effects of temperature and pH on the photocycle kinetics. This would allow the simulations to capture more of the variation and enhance the tools' ability to engineer the response to a target outcome or compensate for adverse experimental conditions.

The unconstrained nature of the parameter G_{o2} also suggests that there may be scope for additional models. For example, a simpler five-state model (similar to the six-state model but without I_2 and its associated transitions) may be able to capture the experimental data as well as the six-state model with a less arduous fitting process and more stable resulting parameters (less sensitive to initial choices). In the first release however, we have used the six-state model for the sake of greater generality.

Another route for future development would be to incorporate other simulators for example PyNEST (Eppler et al., 2008), allowing greater flexibility for users to choose the simulator they are most comfortable with or which offers alternative features and performance benefits. Ultimately this process could be continued to interface with PyNN (Davison et al., 2009) and other simulator-agnostic model description languages such as NeuroML (Gleeson et al., 2010) and NineML (Gorchetchnikov et al., 2011). Furthermore, PyRhO could be combined with the software for optical pattern generation and data acquisition NeuroPG (Avants et al., 2015), to create a complete neural engineering tool for optogenetics.

While there is always scope to add additional protocols, a particularly interesting approach may be to allow networks of neurons to be transfected with several types of opsin and stimulated with multiple wavelengths of light (Han and Boyden, 2007). This could also be supplemented with a more detailed model of the optics of light-tissue interactions as previously implemented for an individual cell in NEURON (Foutz et al., 2012) or interfaced with OptogenSIM for whole brain simulations (Liu et al., 2015). These would be particularly useful additions to PyRhO's neuro-engineering capabilities, allowing stimuli to be more accurately sculpted. In the meantime, users of PyRhO should be mindful of the attenuating effects of light scattering and absorption (or equivalently a spectral shift) from passing through other tissue (particularly *in vivo*) which are not currently explicitly accounted for. These effects may result in a lower effective flux intensity than specified which may shift the "true" value of ϕ_m recovered from the fitting procedures.

In summary, we have presented and verified a new integrated suite of open-source computational tools for optogenetics. PyRhO has been demonstrated to characterize opsins from experimental photocurrents, fitting kinetic model parameters to yield a functional understanding, helping to guide opsin choice and development. These models have been demonstrated in simulations across multiple scales, from channels to networks, by harnessing popular simulators such as NEURON and Brian2. PyRhO is also provided with a Jupyter browser-based GUI to facilitate its use and aid in model sharing. We have outlined some of its chief strengths along with its limitations and plans for future improvements. By releasing these tools as open-source, we hope

that other computational neuroscientists will contribute features and expertise, accelerating progress in the rapidly growing field of optogenetics.

AUTHOR CONTRIBUTIONS

BE designed and wrote the software module and GUI. KN and BE developed the opsin models. BE and KN developed the model fitting algorithms. BE primarily wrote the article, ran the tests, analyzed the data and plotted the figures with participation from KN. BE, KN, SJ, and SS contributed to the project concepts, manuscript comments and discussion of results.

FUNDING

This work was supported by the UK Biotechnology and Biological Sciences Research Council (BBSRC) grant BB/L018268/1 and the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N002474/1.

ACKNOWLEDGMENTS

We would like to thank Matthew Grub and Juan Burrone for the ChR2 photocurrent data used to illustrate the fitting algorithms. We would also like to thank Pepe Herrero for designing the PyRhO logo.

REFERENCES

Adamantidis, A. R., Zhang, F., Aravanis, A. M., Deisseroth, K., and de Lecea, L. (2007). Neural substrates of awakening probed with optogenetic control of hypocretin neurons. *Nature* 450, 420–424. doi: 10.1038/nature06310

Airan, R. D., Thompson, K. R., Fenno, L. E., Bernstein, H., and Deisseroth, K. (2009). Temporally precise *in vivo* control of intracellular signalling. *Nature* 458, 1025–1029. doi: 10.1038/nature07926

Aravanis, A. M., Wang, L.-P., Zhang, F., Meltzer, L. A., Mogri, M. Z., Schneider, M. B., et al. (2007). An optical neural interface: *in vivo* control of rodent motor cortex with integrated fiberoptic and optogenetic technology. *J. Neural Eng.* 4, S143–S156. doi: 10.1088/1741-2560/4/3/s02

Arenkiel, B. R., Peca, J., Davison, I. G., Feliciano, C., Deisseroth, K., Augustine, G. J., et al. (2007). *In vivo* light-induced activation of neural circuitry in transgenic mice expressing channelrhodopsin-2. *Neuron* 54, 205–218. doi: 10.1016/j.neuron.2007.03.005

Arlow, R. L., Foutz, T. J., and McIntyre, C. C. (2013). Theoretical principles underlying optical stimulation of myelinated axons expressing channelrhodopsin-2. *Neuroscience* 248, 541–551. doi: 10.1016/j.neuroscience.2013.06.031

Arrenberg, A. B., Stainier, D. Y. R., Baier, H., and Huisken, J. (2010). Optogenetic control of cardiac function. *Science* 330, 971–974. doi: 10.1126/science.1195929

Avants, B. W., Murphy, D. B., Dapello, J. A., and Robinson, J. T. (2015). NeuroPG: open source software for optical pattern generation and data acquisition. *Front. Neuroeng.* 8:1. doi: 10.3389/fneng.2015.00001

Ayling, O. G. S., Harrison, T. C., Boyd, J. D., Goroshkov, A., and Murphy, T. H. (2009). Automated light-based mapping of motor cortex by photoactivation of channelrhodopsin-2 transgenic mice. *Nat. Methods* 6, 219–224. doi: 10.1038/nmeth.1303

AzimiHashemi, N., Erbguth, K., Vogt, A., Riemensperger, T., Rauch, E., Woodmansee, D., et al. (2014). Synthetic retinal analogues modify the spectral and kinetic characteristics of microbial rhodopsin optogenetic tools. *Nat. Commun.* 5, 1–12. doi: 10.1038/ncomms6810

Bamann, C., Kirsch, T., Nagel, G., and Bamberg, E. (2008). Spectral characteristics of the photocycle of channelrhodopsin-2 and its implication for channel function. *J. Mol. Biol.* 375, 686–694. doi: 10.1016/j.jmb.2007.10.072

Bamberg, E., Bamann, C., Feldbauer, K., Kleinlogel, S., Spitz, J., Zimmermann, D., et al. (2008). *Channelrhodopsins: Molecular Properties and Applications*. Washington, DC: Society for Neuroscience.

Berndt, A., Yizhar, O., Gunaydin, L. A., Hegemann, P., and Deisseroth, K. (2008). Bi-stable neural state switches. *Nat. Neurosci.* 12, 229–234. doi: 10.1038/nn.2247

Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G., and Deisseroth, K. (2005). Millisecond-timescale, genetically targeted optical control of neural activity. *Nat. Neurosci.* 8, 1263–1268. doi: 10.1038/nn1525

Boyle, P. M., Williams, J. C., Ambrosi, C. M., Entcheva, E., and Trayanova, N. A. (2013). A comprehensive multiscale framework for simulating optogenetics in the heart. *Nat. Commun.* 4, 1–9. doi: 10.1038/ncomms3370

Bruegmann, T., and Sasse, P. (2015). Optogenetic cardiac pacemakers: science or fiction? *Trends Cardiovas. Med.* 25, 82–83. doi: 10.1016/j.tcm.2014.10.016

Busskamp, V., and Roska, B. (2011). Optogenetic approaches to restoring visual function in retinitis pigmentosa. *Curr. Opin. Neurobiol.* 21, 1–5. doi: 10.1016/j.conb.2011.06.001

Chow, B. Y., Han, X., Dobry, A. S., Qian, X., Chuong, A. S., Li, M., et al. (2010). High-performance genetically targetable optical neural silencing by light-driven proton pumps. *Nature* 463, 98–102. doi: 10.1038/nature08652

Chuong, A. S., Miri, M. L., Busskamp, V., Matthews, G. A. C., Acker, L. C., Sørensen, A. T., et al. (2014). Noninvasive optical inhibition with a red-shifted microbial rhodopsin. *Nat. Neurosci.* 17, 1123–1129. doi: 10.1038/nn.3752

Davison, A. P., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., et al. (2009). PyNN: a common interface for neuronal network simulators. *Front. Neuroinform.* 2:11. doi: 10.3389/neuro.11.011.2008

Degenaar, P., Grossman, N., Memon, M. A., Burrone, J., Dawson, M., Drakakis, E., et al. (2009). Optobionic vision: a new genetically enhanced light on retinal prosthesis. *J. Neural Eng.* 6:035007. doi: 10.1088/1741-2560/6/3/035007

Eppler, J. M., Helias, M., Muller, E., Diesmann, M., and Gewaltig, M.-O. (2008). PyNEST: A convenient interface to the nest simulator. *Front. Neuroinform.* 2:12. doi: 10.3389/neuro.11.012.2008

Ernst, O. P., Sánchez Murcia, P. A., Daldrop, P., Tsunoda, S. P., Kateriya, S., and Hegemann, P. (2008). Photoactivation of channelrhodopsin. *J. Biol. Chem.* 283, 1637–1643. doi: 10.1074/jbc.M708039200

Feldbauer, K., Zimmermann, D., Pintschovius, V., Spitz, J., Bamann, C., and Bamberg, E. (2009). Channelrhodopsin-2 is a leaky proton pump. *Proc. Natl. Acad. Sci. U.S.A.* 106, 12317–12322. doi: 10.1073/pnas.0905852106

Foutz, T. J., Arlow, R. L., and McIntyre, C. C. (2012). Theoretical principles underlying optical stimulation of a channelrhodopsin-2 positive pyramidal neuron. *J. Neurophysiol.* 107, 3235–3245. doi: 10.1152/jn.00501.2011

Gleeson, P., Crook, S., Cannon, R. C., Hines, M. L., Billings, G. O., Farinella, M., et al. (2010). NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS Comput. Biol.* 6:e1000815. doi: 10.1371/journal.pcbi.1000815

Goodman, D., and Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Front. Neuroinform.* 2:5. doi: 10.3389/neuro.11.005.2008

Goodman, D. F. M., and Brette, R. (2009). The brian simulator. *Front. Neurosci.* 3, 192–197. doi: 10.3389/neuro.01.026.2009

Gorchetchnikov, A., Cannon, R., Clewley, R., Cornelis, H., Davison, A., De Schutter, E., et al. (2011). Nine*ML*: declarative, mathematically-explicit descriptions of spiking neuronal networks. *Front. Neuroinform. Conference Abstract: 4th INCF Congress of Neuroinformatics*. doi: 10.3389/conf.fninf.2011.08.00098

Gradinaru, V., Mogri, M., Thompson, K. R., Henderson, J. M., and Deisseroth, K. (2009). Optical deconstruction of parkinsonian neural circuitry. *Science* 324, 354–359. doi: 10.1126/science.1167093

Gradinaru, V., Thompson, K. R., Zhang, F., Mogri, M., Kay, K., Schneider, M. B., et al. (2007). Targeting and readout strategies for fast optical neural control *in vitro* and *in vivo*. *J. Neurosci.* 27, 14231–14238. doi: 10.1523/JNEUROSCI.3578-07.2007

Gradmann, D., Berndt, A., Schneider, F., and Hegemann, P. (2011). Rectification of the channelrhodopsin early conductance. *Biophys. J.* 101, 1057–1068. doi: 10.1016/j.bpj.2011.07.040

Gradmann, D., Ehlenbeck, S., and Hegemann, P. (2002). Modeling light-induced currents in the eye of chlamydomonas reinhardtii. *J. Mem. Biol.* 189, 93–104. doi: 10.1007/s00232-002-1006-8

Grossman, N., Nikolic, K., Toumazou, C., and Degenaar, P. (2011). Modeling study of the light stimulation of a neuron cell with channelrhodopsin-2 mutants. *IEEE Trans. Biomed. Eng.* 58, 1742–1751. doi: 10.1109/TBME.2011.2114883

Grossman, N., Simiaki, V., Martinet, C., Toumazou, C., Schultz, S. R., and Nikolic, K. (2013). The spatial pattern of light determines the kinetics and modulates backpropagation of optogenetic action potentials. *J. Comp. Neurosci.* 34, 477–488. doi: 10.1007/s10827-012-0431-7

Gunaydin, L. A., Yizhar, O., Berndt, A., Sohal, V. S., Deisseroth, K., and Hegemann, P. (2010). Ultrafast optogenetic control. *Nat. Neurosci.* 13, 387–392. doi: 10.1038/nn.2495

Han, X., and Boyden, E. S. (2007). Multiple-color optical activation, silencing, and desynchronization of neural activity, with single-spike temporal resolution. *PLoS ONE* 2:e299. doi: 10.1371/journal.pone.0000299

Hegemann, P., Ehlenbeck, S., and Gradmann, D. (2005). Multiple photocycles of channelrhodopsin. *Biophys. J.* 89, 3911–3918. doi: 10.1529/biophysj.105.069716

Hegemann, P., and Möglich, A. (2011). Channelrhodopsin engineering and exploration of new optogenetic tools. *Nat. Methods* 8, 39–42. doi: 10.1038/nmeth.f.327

Hernandez, V. H., Gehrt, A., Reuter, K., Jing, Z., Jeschke, M., Mendoza Schulz, A., et al. (2014). Optogenetic stimulation of the auditory pathway. *J. Clin. Invest.* 124, 1114–1129. doi: 10.1172/JCI69050

Hines, M., Davison, A. P., and Muller, E. (2009). Neuron and python. *Front. Neuroinform.* 3:1. doi: 10.3389/neuro.11.001.2009

Hines, M. L., and Carnevale, N. T. (2000). Expanding neuron's repertoire of mechanisms with nmodl. *Neural Comput.* 12, 995–1007. doi: 10.1162/089976600300015475

Ishizuka, T., Kakuda, M., Araki, R., and Yawo, H. (2006). Kinetic evaluation of photosensitivity in genetically engineered neurons expressing green algae light-gated channels. *Neurosci. Res.* 54, 85–94. doi: 10.1016/j.neures.2005.10.009

Klapoetke, N. C., Murata, Y., Kim, S. S., Pulver, S. R., Birdsey-Benson, A., Cho, Y. K., et al. (2014). Independent optical excitation of distinct neural populations. *Nat. Meth.* 11, 338–346. doi: 10.1038/nmeth.2836

Konermann, S., Brigham, M. D., Trevino, A. E., Hsu, P. D., Heidenreich, M., Le, C., et al. (2013). Optical control of mammalian endogenous transcription and epigenetic states. *Nature* 500, 472–476. doi: 10.1038/nature12466

Kuhne, J., Eisenhauer, K., Ritter, E., Hegemann, P., Gerwert, K., and Bartl, F. (2014). Early formation of the ion-conducting pore in channelrhodopsin-2. *Angewandte Chemie* 54, 4953–4957. doi: 10.1002/anie.201410180

Lagali, P. S., Balya, D., Awatramani, G. B., Münch, T. A., Kim, D. S., Busskamp, V., et al. (2008). Light-activated channels targeted to on bipolar cells restore visual function in retinal degeneration. *Nat. Neurosci.* 11, 667–675. doi: 10.1038/nn.2117

Lin, J. Y. (2011). A user's guide to channelrhodopsin variants: features, limitations and future developments. *Exp. Physiol.* 96, 19–25. doi: 10.1113/expphysiol.2009.051961

Lin, J. Y., Lin, M. Z., Steinbach, P., and Tsien, R. Y. (2009). Characterization of engineered channelrhodopsin variants with improved properties and kinetics. *Biophys. J.* 96, 1803–1814. doi: 10.1016/j.bpj.2008.11.034

Liu, Y., Jacques, S. L., Azimipour, M., Rogers, J. D., Pashaie, R., and Eliceiri, K. W. (2015). Optogensim: a 3d monte carlo simulation platform for light delivery design in optogenetics. *Biomed. Opt. Exp.* 6, 4859–4870. doi: 10.1364/BOE.6.004859

Muller, E., Bednar, J. A., Diesmann, M., Gewaltig, M.-O., Hines, M., and Davison, A. P. (2015). Python in neuroscience. *Front. Neuroinform.* 9:11. doi: 10.3389/fninf.2015.00011

Nagel, G., Szellas, T., Huhn, W., Kateriya, S., Adeishvili, N., Berthold, P., et al. (2003). Channelrhodopsin-2, a directly light-gated cation-selective membrane channel. *Proc. Natl. Acad. Sci. U.S.A.* 100, 13940–13945. doi: 10.1073/pnas.1936192100

Newville, M., Stensitzki, T., Allen, D. B., and Ingargiola, A. (2014). LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python. *Zenodo*. doi: 10.5281/zenodo.11813

Nikolic, K., Grossman, N., Grubb, M. S., Burrone, J., Toumazou, C., and Degenaar, P. (2009). Photocycles of channelrhodopsin-2. *Photochem. Photobiol.* 85, 400–411. doi: 10.1111/j.1751-1097.2008.00460.x

Nikolic, K., Grossman, N., Yan, H., Drakakis, E., Toumazou, C., and Degenaar, P. (2007). "A non-invasive retinal prosthesis - testing the concept," in *Engineering in Medicine and Biology Society, EMBS 2007. 29th Annual International Conference of the IEEE* (Lyon), 6364–6367.

Pérez, F., and Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Comput. Sci. Eng.* 9, 21–29. doi: 10.1109/MCSE.2007.53

Petreanu, L., Mao, T., Sternson, S. M., and Svoboda, K. (2009). The subcellular organization of neocortical excitatory connections. *Nature* 457, 1142–1145. doi: 10.1038/nature07709

Shoham, S., and Deisseroth, K. (2010). Special issue on optical neural engineering: advances in optical stimulation technology. *J. Neural Eng.* 7:040201. doi: 10.1088/1741-2560/7/4/040201

Stehfest, K., and Hegemann, P. (2010). Evolution of the channelrhodopsin photocycle model. *Chemphyschem* 11, 1120–1126. doi: 10.1002/cphc.200900980

Topalidou, M., Leblois, A., Boraud, T., and Rougier, N. P. (2015). A long journey into reproducible computational neuroscience. *Front. Comput. Neurosci.* 9:30. doi: 10.3389/fncom.2015.00030

Wang, H., Peca, J., Matsuzaki, M., Matsuzaki, K., Noguchi, J., Qiu, L., et al. (2007). High-speed mapping of synaptic connectivity using photostimulation in channelrhodopsin-2 transgenic mice. *Proc. Natl. Acad. Sci. U.S.A.* 104, 8143–8148. doi: 10.1073/pnas.0700384104

Williams, J. C., Xu, J., Lu, Z., Klimas, A., Chen, X., Ambrosi, C. M., et al. (2013). Computational optogenetics: empirically-derived voltage- and light-sensitive channelrhodopsin-2 model. *PLoS Comput. Biol.* 9:e1003220. doi: 10.1371/journal.pcbi.1003220

Yizhar, O., Fenno, L.E., Davidson, T. J, Mogri, M., and Deisseroth, K. (2011). Optogenetics in neural systems. *Neuron* 71, 9–34. doi: 10.1016/j.neuron.2011.06.004

Zhang, F., Vierock, J., Yizhar, O., Fenno, L. E., Tsunoda, S., Kianianmomeni, A., et al. (2011). The microbial opsin family of optogenetic tools. *Cell* 147, 1446–1457. doi: 10.1016/j.cell.2011.12.004

Zhang, F., Wang, L. P., Boyden, E. S., and Deisseroth, K. (2006). Channelrhodopsin-2 and optical control of excitable cells. *Nat. Methods* 3, 785–792. doi: 10.1038/nmeth936

Zhang, F., Wang, L. P., Brauner, M., Liewald, J. F., Kay, K., Watzke, N., et al. (2007). Multimodal fast optical interrogation of neural circuitry. *Nature* 446, 633–639. doi: 10.1038/nature05744

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX

Three-State Analytic Solution

The analytic solution for arbitrary initial conditions (C_0, O_0, D_0) is as follows:

$$C(t) = \frac{G_d G_r}{\lambda_1 \lambda_2} + \frac{(\lambda_1 - G_d - G_r)}{G_d \lambda_1 \xi} \cdot \zeta_1 \cdot e^{-\lambda_1 \cdot t} - \frac{(\lambda_2 - G_d - G_r)}{G_d \lambda_2 \xi} \cdot \zeta_2 \cdot e^{-\lambda_2 \cdot t} \quad \text{(A1)}$$

$$O(t) = \frac{G_a G_r}{\lambda_1 \lambda_2} - \frac{(\lambda_1 - G_r)}{G_d \lambda_1 \xi} \cdot \zeta_1 \cdot e^{-\lambda_1 \cdot t} - \frac{(\lambda_2 - G_r)}{G_d \lambda_2 \xi} \cdot \zeta_2 \cdot e^{-\lambda_2 \cdot t} \quad \text{(A2)}$$

$$D(t) = \frac{G_a G_d}{\lambda_1 \lambda_2} + \frac{1}{\lambda_1 \xi} \cdot \zeta_1 \cdot e^{-\lambda_1 \cdot t} - \frac{1}{\lambda_2 \xi} \cdot \zeta_2 \cdot e^{-\lambda_2 \cdot t} \quad \text{(A3)}$$

with the following definitions:

$$\xi = \sqrt{G_a^2 + G_d^2 + G_r^2 - 2 \cdot (G_a G_d + G_a G_r + G_d G_r)}$$

$$\lambda_{1,2} = \frac{1}{2} \cdot \left((G_a + G_d + G_r) \pm \xi \right)$$

$$\zeta_{1,2} = C_0 \cdot [G_d \cdot G_a] + O_0 \cdot \left[G_d \cdot (G_a - \lambda_{1,2}) \right] + D_0 \cdot \left[G_r \cdot (\lambda_{1,2} - G_a - G_d) \right]$$

Note also that $\lambda_1 \cdot \lambda_2 = G_a G_d + G_a G_r + G_d G_r$ i.e., the sum of products.

Model-independent Fitting Procedures

Prior to fitting the specific parameters of a chosen model, several fitting processes are performed to find parameters common to all models:

1. If a set of voltage clamp data are provided over a range of potentials, the steady-state currents are used to fit the rectification function, $f_v(v)$, (Equation 5) yielding the parameters E, v_0, and v_1:
   a. The I_{ss} values are first fit to the function $f_v(\mathbf{V_{clamp}}) \cdot (\mathbf{V_{clamp}} - E)$ to find an initial estimate for v_0 and a value for E where the polarity of the steady-state currents changes (along with a dummy parameter comprising $g_0 \cdot f_\phi \cdot v_1$, which is discarded).
   b. The reversal potential, E, is fixed and the steady-state currents are converted to conductances: $\mathbf{g} = \mathbf{I_{ss}}/(\mathbf{V_{clamp}} - E)$. These conductances are then normalized by dividing by the conductance calculated at $V_{clamp} = -70$ mV to produce an array of conductance factors.
   c. The conductance factors are fit to the function $f_v(V_{clamp})$ with the constraint $v_1 = (70 + E) / \left(e^{\frac{70+E}{v_0}} - 1 \right)$ to calculate and fix the parameters E, v_0 and v_1.
2. All photocurrents in the dataset recorded at −70 mV are scanned to find the highest peak. This value is used to calculate an initial estimate for g_0 (according to the formula $g_0 = I_{\text{peak,max}}/(-70 - E)$) to aid the optimization routines.
3. If the recovery protocol is included in the dataset (pairs of pulses with a range of inter-pulse-intervals) then the peak currents of the second pulses are extracted along with the times and the initial peak and steady-state current. These values are then aligned to the end of the first pulse (where the current is I_{ss0}) and fit to the exponential recovery function: $I_{\text{peak}}(t) = I_{peak0} - a \cdot e^{G_{r0} \cdot t}$. This yields the dark recovery rate constant, G_{r0}.

These parameters are then fixed (except for the estimate of g_0) and passed to the model-specific fitting routines.

Model-dependent Fitting Procedures

Each model is fit with broadly the same three-stage fitting procedure consisting of Off-curve parameters, then On-curve parameters and finally re-optimizing:

1a. Fit bi-exponential functions to the Off-curves to find parameter values for the light-insensitive decay rates: {G_{d1}, G_{d2}, G_{f0}, G_{b0}} for the four- and six-state models [detailed in Section Four- and Six-state Model Off-curve Fitting (Step 1a.)], or G_d for the three-state model (described in Equation 7),
1b. [Six-state only] Fix the values for the parameters from Step 1a then calculate activation rates G_{o1} and G_{o2} [detailed in Section Six-State Opsin Activation Rate Fitting (Step 1b.)],
2. Fix the values for the parameters found in Step 1 and fit the On-curves to find the remaining parameters governing the light-dependent transitions,
3. Using the values found in Steps 1 and 2 as initial values, relax all parameter values to freely vary within a range (by default between 50 and 200% of the initial values) and refit the model across the set of whole photocurrent curves (on- and off-phases).

Four- and Six-state Model Off-curve Fitting (Step 1a.)

The fitting equations for the Light-Off response curves can be calculated directly from the set of ODEs given in **Table 2**, for light flux zero, as shown in Nikolic et al. (2009):

$$O_{1\text{off}} = A_{11} \exp(-\Lambda_1 t) + A_{12} \exp(-\Lambda_2 t) \quad \text{(A4)}$$

$$O_{2\text{off}} = A_{21} \exp(-\Lambda_1 t) + A_{22} \exp(-\Lambda_2 t) \quad \text{(A5)}$$

where

$$\Lambda_{1,2} = b \mp c \,,$$

$b = (G_{\text{d1}} + G_{\text{d2}} + G_{\text{f0}} + G_{\text{b0}})/2$ and $c = \sqrt{b^2 - (G_{\text{d1}} G_{\text{d2}} + G_{\text{d1}} G_{\text{b0}} + G_{\text{d2}} G_{\text{f0}})}$. Two useful relationships follow from the previous equations:

$$\Lambda_1 + \Lambda_2 = G_{\text{d1}} + G_{\text{d2}} + G_{\text{f0}} + G_{\text{b0}} \quad \text{(A6)}$$

$$\Lambda_1 \cdot \Lambda_2 = G_{\text{d1}} G_{\text{d2}} + G_{\text{d1}} G_{\text{b0}} + G_{\text{d2}} G_{\text{f0}} \quad \text{(A7)}$$

The current after the light is turned off is given by:

$$\begin{aligned} I_{\text{off}} &= V(g_1 \cdot O_{1\text{off}} + g_2 \cdot O_{2\text{off}}) \\ &= I_{\text{slow}} \exp(-\Lambda_1 t) + I_{\text{fast}} \exp(-\Lambda_2 t), \end{aligned} \tag{A8}$$

where the expression for I_{slow} (corresponds to Λ_1) and I_{fast} (corresponds to Λ_2) components of the off-current can be calculated from the equation above and are given in Nikolic et al. (2009). Note also that $I_{\text{slow}} + I_{\text{fast}} = I_{\text{ss}}$ which is included as an additional constraint in the fitting procedure.

Six-State Opsin Activation Rate Fitting (Step 1b.)

From the differential equations describing the dynamics of a short pulse (**Table 2**, three-state model):

$$\frac{dO}{dt} = G_a(t)C - G_d O$$

where $G_a(t)$ is given in Nikolic et al., 2007, Appendix 1. If the light illumination occurs from $t = 0$ to $t = t_{\text{pulse}}$, then after the light goes off ($t > t_{pulse}$):

$$G_a(t) = G_a[e^{t_{pulse}/\tau_{opsin}} - 1] \cdot e^{-t/\tau_{opsin}}, \tag{A9}$$

where $\tau_{opsin}(= 1/G_o)$ is the time constant of the opsin complex activation (or conformal change) after the retinal isomerization (which happens very quickly upon photon absorption). Now if we use a very short pulse then $t_{pulse} \ll \tau_{opsin}$ so that $C \approx 1$ (and secondary cycles are not significantly activated), we may apply the standard approximation for small exponents $e^x \approx 1 + x$:

$$G_a(t) \approx G_a \frac{t_{pulse}}{\tau_{opsin}} \cdot e^{-t/\tau_{opsin}} = Q \cdot e^{-\frac{t}{\tau_{opsin}}}, \quad \text{where } Q = G_a \frac{t_{pulse}}{\tau_{opsin}}$$

and so

$$\frac{dO}{dt} \approx Q \cdot e^{-G_o t} - G_d O. \tag{A10}$$

The solution of Equation (A10) is the limit case for very short pulses (relative to τ_{opsin}) when $I(t)$ stops being dependent on the pulse duration t_{pulse}:

$$I(t) = g \cdot (v - E) \cdot \frac{Q}{(G_o - G_d)} \cdot e^{-t/\tau_d} \cdot (1 - e^{-(G_o - G_d)t}). \tag{A11}$$

Differentiating Equation (A11) we find the time of the peak current:

$$t_{maxlag} = \frac{\log(G_o/G_d)}{G_o - G_d} = \frac{\tau_{opsin}\tau_d}{\tau_d - \tau_{opsin}} \log\left(\frac{\tau_d}{\tau_{opsin}}\right),$$

where $\tau_d = 1/G_d$. Assuming that $\tau_{opsin} \ll \tau_d$ then,

$$t_{maxlag} \approx \tau_{opsin} \log\left(\frac{\tau_d}{\tau_{opsin}}\right). \tag{A12}$$

To find t_{maxlag} we can fit the following function to a series of short pulses:

$$t_{peak} = t_{pulse} + t_{maxlag} \cdot e^{-k \cdot t_{pulse}}.$$

Alternatively, estimate t_{maxlag} from the "delta" protocol as $t_{pulse} \to 0$. Once we have obtained an estimate for t_{maxlag} we can then solve for τ_{opsin} iteratively as follows:

$$\tau_{opsin,i+1} = \frac{t_{maxlag}}{(t_{maxlag}/\tau_d) - \log(\tau_{opsin,i}/\tau_d)}. \tag{A13}$$