Derivative-based Inference for Cell and Channel Electrophysiology Models

Michael Clerx[1], David Augustin[2], Alister R Dale-Evans[2], Gary R Mirams[1]

[1]University of Nottingham, UK. [2]University of Oxford, UK

Abstract

Models of ionic currents or of the cardiac action potential (AP) are frequently calibrated by defining an error function that quantifies the mismatch between simulations and data, and using numerical optimisation to find the parameter values that minimise this function. Many optimisation algorithms assume knowledge of the derivatives of the error function with respect to the parameters, but for models formulated as differential equations these are typically unknown.

In this study we extend our simulation tool, Myokit, with the capability to rapidly calculate derivatives of simulation output and couple it to our inference tool, PINTS, to calculate the derivatives of the error function. We measure the added overhead of the sensitivity calculations in a model of the ion current I_{Kr} and in a model of a stem-cell AP. Next we compare the performance of a state-of-the art derivative-free optimiser with that of a popular derivative-using method. For both problems, the derivative-based method requires fewer function evaluations, but this is offset by a significant increase in the computational cost of each evaluation. The derivative-free method is much faster for the I_{Kr} case, while the derivative-using method outperforms on the AP case. However, the derivative-free method is more robust on both problems: providing the correct answer on a greater percentage of runs.

1. Introduction

Models of cardiac cell and channel electrophysiology are commonly formulated as systems of ordinary differential equations (ODEs)

$$\frac{dy}{dt} = f(y, t, p), \quad y(t = 0) = y_0, \tag{1}$$

where y is a vector of state variables, t is time, and p is a vector of model parameters. Values for p can be found by defining an error function, $E(p)$, to quantify the mismatch between simulation and experiment, and then minimising E using numerical optimisation [1, 2]. A simple choice is

$$E(p) = \sum_i \left(m(t_i) - v_i\right)^2, \tag{2}$$

where v_i are experimental measurements taken at time t_i, while $m(t_i)$ is the corresponding simulation output. For example, y may contain variables describing ion channel states and ionic concentrations, while m is a current calculated from y and p.

Many classical algorithms for numerical optimisation utilise the derivatives $\partial E/\partial p$, but because evaluating E requires solving an initial value problem (running a simulation) these are not usually available. In this study we extend existing modelling tools with methods to calculate $\partial E/\partial p$, measure the overhead of the extra calculations, and compare the performance of a method that uses derivatives and a derivative-free one. Two test cases are used, each based on real experiments but with synthetically generated data.

2. Methods

All experiments were run using Python 3.9.13. Results, simulation code, and figure-generating scripts are provided at https://github.com/CardiacModelling/fitting-with-derivatives-cinc.

2.1. Test case: I_{Kr}

The first test case is a model of the ionic current I_{Kr} as a function of membrane potential (V), as detailed in [2]. Here y consists of two states, each described by a forward and backward reaction rate of the form $k_i = a_i \exp(b_i V)$ where a_i and b_i are parameters. The output m is a function of y and a conductance parameter, so that there are 9 parameters in total. The input V is a time-variant signal based on the "staircase protocol" from [3]. For simplicity, the initial state y_0 is assumed known.

Lower and upper limits are defined on the individual parameters and on the rates, optimisation is performed in a space where the a_i parameters are log-transformed, and starting points for each optimisation are sampled randomly from within the (log-transformed) boundaries [2]. The "experimental data" to fit to was generated by running a simulation with parameters based on [2] and adding normally distributed noise with $\sigma = 0.025$A/F.

2.2. Test case: AP

The second test case is a model of the action potential (AP) in human induced pluripotent stem cells [4]. Here y contains two internal calcium concentrations and several state variables for ion current models. The simulations mimic a perforated patch experiment [5] in which V is fixed to a time-varying input signal, all external concentrations are set to the bath concentrations, and the cytosolic concentrations of monovalent ions are assumed equal those in the pipette. The output m is the sum of 10 ionic transmembrane currents, each multiplied by a dimensionless scaling parameter, and 2 fixed-size "background" currents. For simplicity the initial state y_0 is assumed known.

Lower and upper limits on the parameters are set as 10^{-3} and 10^3 respectively, all 10 parameters are log-transformed during optimisation, and starting points are sampled uniformly from within the log-transformed boundaries. The data to fit to was generated by simulating with all scaling parameters set to 1 and then adding normally distributed noise with $\sigma = 0.1$A/F.

2.3. Calculating sensitivities

We adapted our modelling tool, Myokit [6], to use CVODES [7] to calculate both the solution to initial value problems, $y(t)$ and the *sensitivities*, defined as $s_i(t) \equiv \partial y(t)/\partial q_i$, where q_i is either a parameter or an initial value. To calculate $s_i(t)$ CVODES integrates

$$\dot{s}_i(t) = \frac{\partial f}{\partial y} s_i(t) + \frac{\partial f}{\partial q_i} y(t), \tag{3}$$

along with f, using finite difference approximations to estimate $\partial f/\partial y$ and $\partial f/\partial q_i$ [7]. We added symbolic differentiation functionality to Myokit to then derive $\partial m/\partial p$ and the final step of calculating $\partial E/\partial p$ was handled by the inference software PINTS [8].

2.4. Numerics & benchmarking

Based on previous successes we used CMA-ES [9] as a derivative-free optimiser [2, 3]. The implementation used was published by the method's authors (although we accessed it via a wrapper in PINTS [8]). As a derivative-based method we chose iRprop-[10], an implementation of which was added to PINTS for this study. Although this is a relatively simple method we found it outperformed other derivative-based algorithms.

Benchmarking was performed on a laptop with an Intel Core i9-10885H CPU with 8 true 2.4 GHz cores (sometimes advertised as 16 cores with "hyperthreading"). Up to 8 experiments were run simultaneously, but none of the individual experiments were parallelised so that each experiment had access to a single (true) core.

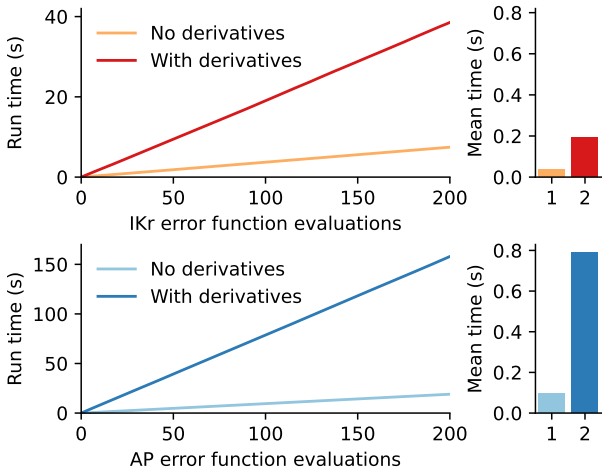

Figure 1. The time taken for 200 evaluations of E with and without derivatives, for the I_{Kr} (top) and AP (bottom) cases. Mean time per evaluation is shown on the right.

3. Results

Example simulations for both test cases can be viewed in the repository accompanying this paper.

To measure the overhead of calculating $\partial E/\partial p$, we evaluated each error function 200 times on parameters sampled randomly from within the boundaries (Figure 1). The I_{Kr} error function was 5.2 times slower when derivatives were calculated, while for the AP error function this rose to a factor 8.3.

Next, we ran 100 optimisations for each case and method, again starting at points sampled randomly from within the problem boundaries. Results for the I_{Kr} test case are shown in Figure 2. The derivative-using method iRprop- uses fewer evaluations, but their increased run-time makes CMA-ES the faster option.

Results for the AP test case are shown in Figure 3. The derivative-using iRprop- method uses far fewer evaluations, compensating for the 8 times slower run-time per evaluation and leading to a moderate improvement over CMA-ES in speed.

Next, we considered the accuracy and reliability of the obtained results. On synthetic problems we can check whether the obtained solution is the intended one, but in a realistic scenario the true solution is unknown so that other measures must be used. Here, we ordered the 100 results for each case-and-method combination from low (best) to high error (worst) and calculated the error relative-to-best-obtained in each run i as $(E_i - E_0)/E_i$.

This relative error is plotted for the I_{Kr} and AP test cases in the first 1st and 3d row of Figure 4 respectively. For I_{Kr} we can see that CMA-ES is more robust, returning an error within 5% of E_0 in 80 out of 100 runs, while for iRprop-

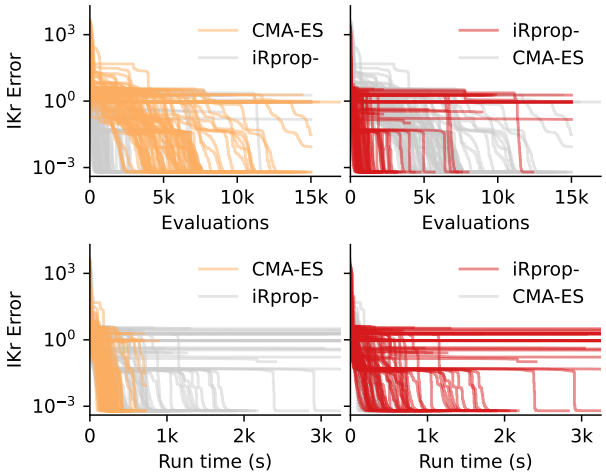

Figure 2. (Top) The error as a function of the number of function evaluations made by CMA-ES (left) and iRprop- (right). For ease of comparison, data for the competing method is shown in grey in the background. Both methods find the optimum in most, but not all runs, and iRprop- converges quicker. (Bottom) Viewed as a function of runtime the advantage for iRprop- disappears.

this figure was reduced to 57. On the AP problem CMA-ES returned a similar error on 95 runs, compared to 83 for iRprop-.

Disparate points can have similar errors, so we also defined a measure of the distance from the obtained parameters to the lowest-error parameters. Writing p_{ij} for value of parameter j obtained in run i, we define the maximum relative parameter error in run i as $\max_j |p_{ij} - p_{0j}|/p_{0j}$. The resulting parameter errors are plotted in rows 2 and 4 of Figure 4, and show that low errors corresponded to similar solutions in all 4 experiments.

Finally, we tried several other derivative-using methods but without success. For example, none of the methods included in SciPy (BFGS, CG, SHGO with SLSQP as local optimiser) or methods we implemented for this project (Adam, AdaGrad) failed to converge, even after attempts to manually 'tune' their performance.

We also experimented with MCMC sampling methods. Here we were successful with derivative-free adaptive methods such as ACMC, while derivative-using methods such as Hamiltonian MC and NUTS failed to converge within a reasonable time frame.

4. Discussion

We tested a derivative-using and a derivative-free method on two realistic inference problems. On both problems the derivative-using method required fewer evaluations, but the increased cost per evaluation led to an in-

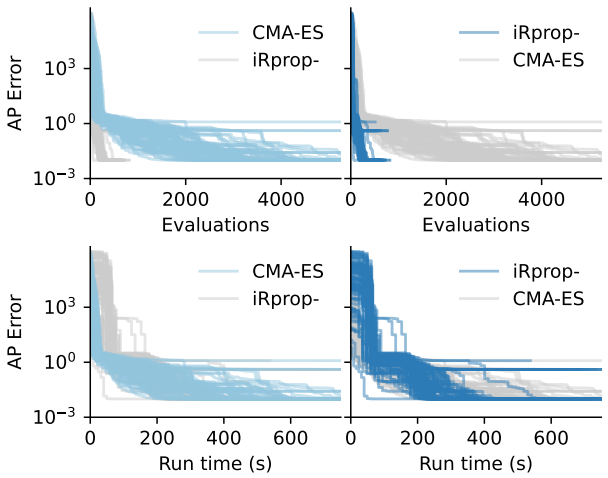

Figure 3. As Figure 2 but for the AP case. Here the reduced number of evaluations for iRprop- outweights the increased time per evaluation, making it the faster method.

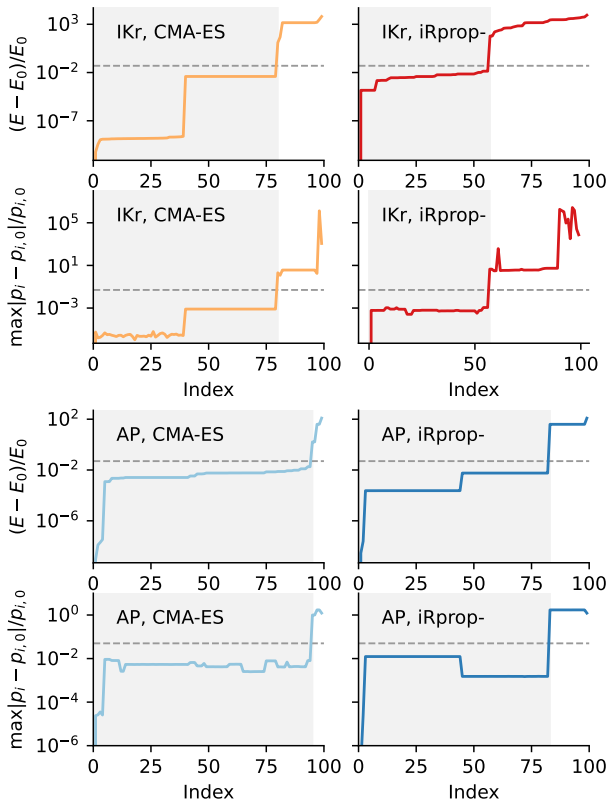

Figure 4. Error relative to best-obtained error (rows 1 & 3), and maximum relative difference between parameters and best-obtained-error parameters (row 2 & 4). Runs are ordered from lowest error (E) to highest, so that the plots for E are monotonically non-decreasing. The dotted line and the gray shaded areas indicate the percentage of results within 5% of the best obtained result.

creased run-time for the I_{Kr} case and an only slightly decreased run-time for the AP problem. In all tests the derivative-free method CMA-ES was more robust, returning the "correct" solution more often than derivative-using iRprop-. In this discussion we will focus first on the question of how (or if) we can improve the run-time of derivative calculations, and then on the larger questions of whether gradient information is useful for these problems and why so many optimisers failed to find a solution at all.

For both problems studied, we integrated and stored sensitivities for n parameters of interest and m states at k points in time, before using this set of $n \times m \times k$ values to calculate just n derivatives $\partial E/\partial p$. The memory allocation overhead associated with this process can partially be avoided by using the *adjoint method* of calculating $\partial E/\partial p$ [7]. Speed-ups might also be obtained by symbolically deriving expressions for $\partial f/\partial y$ and $\partial f/\partial q_i$, avoiding finite-difference approximation. Under specific conditions, analytical solutions exist for the I_{Kr} problem, from which derivative equations can be derived. This would speed up derivative-free and derivative-using methods, but may give a competitive advantage to the derivative-using case. We avoided parallelisation in this study, but it is worth noting that many derivative-free methods are trivially parallelisable, while derivative-using methods typically are not.

Gradients are usually regarded as an invaluable tool for optimisation. Compared to derivative-based optimisation, derivative-free methods are relatively new, and much less is known about their convergence properties. So how can we explain the observed lack of improvement on these cases?

The simplest explanation is that there is some error in our $\partial E/\partial p$ calculations, although tests comparing to finite difference approximations seem to indicate this is not the case. Next, it may be that a different choice of derivative-using optimisers (or their implementations) could yield a better result, or that there are some settings that require adjusting. Methods have been proposed to do this automatically (hyperparameter optimisation) and are a possible direction for future work.

An interesting property of both CMA-ES and iRprop- is that they automatically adjust to the scale of each parameter. CMA-ES does this by learning a covariance matrix that characterises the search space, while iRprop- maintains a separate learning rate for each parameter. As a result, they can find solutions even when E changes very rapidly in one parameter but very slowly in another. By contrast, typical gradient-descent methods or line-search methods like BFGS assume that $\partial E/\partial p_i$ has a similar scale for all parameters p_i. It may be possible then, that further transformations of the search space can condition the problem so that other derivative-using methods achieve good performance.

Finally, it is worth considering that, especially on noisy and non-linear problems the gradient at a single point may be a poor predictor of the location of the global optimum. We may speculate that methods like CMA-ES, which sample several points before taking a step, are actually using more reliable (more "global") information. Further work is needed to test these ideas.

In conclusion, derivative-based methods can provide performance improvements, but the extra work needed to implement and use them may not be worth the effort for cell and channel electrophysiology models.

References

[1] Whittaker DG, Clerx M, Lei CL, Christini DJ, Mirams GR. Calibration of ionic and cellular cardiac electrophysiology models. Wiley Interdisciplinary Reviews Systems Biology and Medicine 2020;12(4):e1482.

[2] Clerx M, Beattie KA, Gavaghan DJ, Mirams GR. Four ways to fit an ion channel model. Biophysical Journal 2019; 117:2420–2437.

[3] Lei C, Clerx M, Gavaghan DJ, Polonchuk L, Mirams GR, Wang K. Rapid characterisation of hERG channel kinetics I: using an automated high-throughput system. Biophysical Journal 2019;117:2438–2454.

[4] Kernik DC, Morotti S, Wu H, Garg P, Duff HJ, Kurokawa J, Jalife J, Wu JC, Grandi E, Clancy CE. A computational model of induced pluripotent stem-cell derived cardiomyocytes incorporating experimental variability from multiple data sources. The Journal of physiology 2019; 597(17):4533–4564.

[5] Clark AP, Wei S, Kalola D, Krogh-Madsen T, Christini DJ. An in silico–in vitro pipeline for drug cardiotoxicity screening identifies ionic pro-arrhythmia mechanisms. British Journal of Pharmacology 2022;1–15.

[6] Clerx M, Collins P, de Lange E, Volders PGA. Myokit: A simple interface to cardiac cellular electrophysiology. Progress in Biophysics and Molecular Biology 2016; 120(1–3):100–114. ISSN 0079-6107.

[7] Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software September 2005; 31(3):363–396.

[8] Clerx M, Robinson M, Lambert B, Lei CL, Ghosh S, Mirams GR, Gavaghan DJ. Probabilistic Inference on Noisy Time Series (PINTS). Journal of Open Research Software 2019;7(1):23.

[9] Hansen N. The CMA evolution strategy: A tutorial. arXiv 2016;abs/1604.00772.

[10] Igel C, Hüsken M. Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 2003;50:105–123.

Address for correspondence:

michael.clerx@nottingham.ac.uk School of mathematical sciences, University of Nottingham, NG7 2RD, UK.