Automated Customization of Cardiac Electrophysiology Models to Facilitate Patient-Specific Modeling

Darby I Cairns[1*], Maxfield R Comstock[1*], Flavio H Fenton[2], Elizabeth M Cherry[1]

[1]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2]School of Physics, Georgia Institute of Technology, Atlanta, GA, USA
[*]Both authors contributed equally to this work

Abstract

Models of cardiac electrophysiology can be useful for assessing the behavior of the heart when subjected to interventions like pacing, defibrillation, or drugs. However, for patient-specific predictions, models must be customized to match the electrophysiological properties of individuals. We present a browser-based tool for customizing models of cardiac action potentials by fitting parameter values to user-provided datasets. The tool uses a particle swarm optimization algorithm accelerated by the user's graphics card, which can support a large particle population and typically finds a low-error solution within a small number of iterations (10–32), computed within seconds. The interface allows all model parameters or a user-specified subset to be selected for fitting, and the user can set bounds for all parameters to constrain their values. We demonstrate the effectiveness of this tool by creating parameterizations of a four-variable human ventricular model to match human cardiac action potential data taken from tissue exhibiting Brugada syndrome. We expect this tool will be useful for tuning models to match data recorded from individual experiments and patients under normal and diseased conditions.

1. Introduction

The electrical dynamics of cardiac tissue vary based on tissue type, even across different regions of the same heart. Different patients' hearts will have distinct dynamics; medication can also affect this behavior. When applying models of cardiac action potentials (APs), it is necessary to tune them to best fit the data of interest by adjusting the model parameter values. Finding such parameter sets is a difficult nonlinear optimization problem, as the search space may have a large number of local extrema. For patient-specific AP models to be useful in a clinical setting, any method for generating such a parameterization must be fast, flexible, and easy to use.

Methods such as particle swarm optimization (PSO) and genetic algorithms have been used to solve similar nonlinear optimization problems [1, 2]. However, these methods are computationally expensive and challenging to implement, requiring detailed knowledge of both the models being parameterized and the optimization algorithms. The aim of our work is to provide a tool that uses the PSO algorithm to find parameterizations of cardiac AP models to fit membrane potential timeseries data. Toward this end, we present a browser-based tool that can be run on virtually any modern computer, with an interface allowing for quick and simple control of the algorithm's behavior. Our implementation automatically takes advantage of any available graphics hardware to parallelize the algorithm, resulting in accurate results in a matter of seconds even on consumer-grade laptops.

2. Methods

2.1. Data

We generated cardiac AP model parameterizations to fit human tissue data obtained from optical mapping of explanted human hearts. Individual positions in the optical mapping data were used to produce timeseries data for a single point. The tissue demonstrated hallmarks of Brugada syndrome, a genetic disorder characterized by abnormally long APs with characteristic shapes that can induce ventricular arrhythmias.

2.2. Model

For the present study, the PSO algorithm is used with a four-variable human ventricular model adapted for Brugada syndrome [3]. Of the 39 model parameters, we fit between 20 and 23 using PSO, with the remainder set from published model parameterizations. The pacing stimulus

for the model uses a biphasic current that reflects the effect of voltage diffusion between cells in cardiac tissue, which leads to better approximations of tissue behavior while retaining the computational advantages of a single-cell model.

2.3. Particle Swarm Optimization

Particle swarm optimization is a derivative-free optimization method that makes no assumptions about the problem it is optimizing. The algorithm functions by maintaining a pool of candidate solutions, referred to as particles, and iteratively choosing parameter sets that produce output more consistent with the data until an error threshold has been met or a maximum number of iterations has been reached. In our application, the position, $\vec{p}_i$, of particle i is a vector containing candidate values for each parameter being fit. Each particle also has an associated velocity $\vec{v}_i$, which is used to update its position during each iteration. Initially, the positions are drawn from a uniform random distribution defined by the bounds specified for each parameter, with random velocities as well. Each particle tracks the best (lowest error) set of parameters it has found during execution, $\vec{b}_i$, and has knowledge of the global best achieved by any particle, $\vec{b}_g$.

The position and velocity of each particle are updated according to the formulae

$$\vec{v}_i \leftarrow \chi\Big[\vec{v}_i + \vec{U}(0,\phi_1) \otimes \left(\vec{b}_i - \vec{p}_i\right) + \vec{U}(0,\phi_2) \otimes \left(\vec{b}_g - \vec{p}_i\right)\Big], \quad (1)$$

$$\vec{p}_i \leftarrow \vec{p}_i + \vec{v}_i, \quad (2)$$

where constriction coefficient χ is a parameter described in Ref. [4] and $\vec{U}(a,b)$ is a vector of uniformly distributed random numbers in the range $[a,b)$. The symbol $\otimes$ denotes component-wise multiplication. Each parameter value is restricted to a range, and if a particle contains any parameter outside its range, that value is reset to a randomly selected value in the nearest three-quarters of the range. As in Ref. [1], we chose $\phi_1 = \phi_2 = 2.05$ and $\chi = 2/(\phi - 2 + \sqrt{\phi^2 - 4\phi}) \approx 0.73$ as the default values for our PSO implementation, with $\phi = \phi_1 + \phi_2$.

For each particle, the model is run and the output is compared with the input data, with the sum of squared error used to quantify error for a given parameterization. As many aspects of the algorithm are randomized, the result of a particular run is not deterministic.

2.4. Software Interface

We implemented the PSO algorithm as a static web page (see Figure 1), which allows for an interactive user interface and GPU-accelerated parallelism using the WebGL API for JavaScript. The particle swarm optimization algorithm is well-suited to parallelization, as the update steps for each particle in a given iteration are independent. Other steps, such as computing the global best particle position, can take limited advantage of parallelism. Because the most significant computational requirement for the algorithm is the solution of the model for every particle at every iteration in order to evaluate the quality of each particle position, the update step where parallelism is easiest to apply is also the most relevant source of speedup.

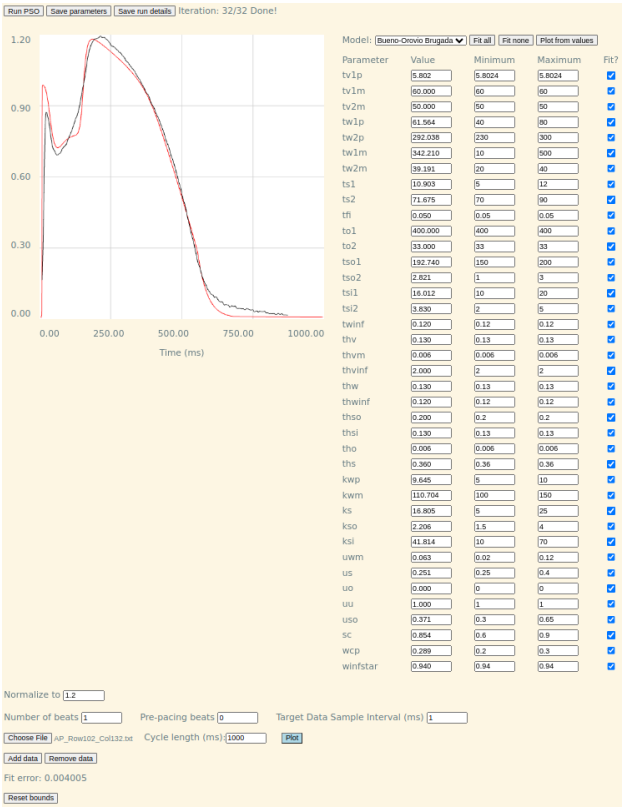

Figure 1. Interface of the PSO tool with representative Brugada fitting.

The PSO interface consists of several components, as shown in Figure 1. Buttons at the top of the page are used to initiate PSO runs and to save the results. The upper-left part of the page contains a plot that compares the fit found by PSO to the provided data. On the right is a section that allows the user to select a model and set parameter bounds, as well as to identify parameters not to be fit and their values. Once a fit is complete, the resulting parameterization is displayed here as well. Below, input data to the PSO algorithm for fitting the model is specified. Aside from normalization, most pre-processing of the data should occur before it is provided as an input to PSO. For instance, high-resolution data should be downsampled to a sample

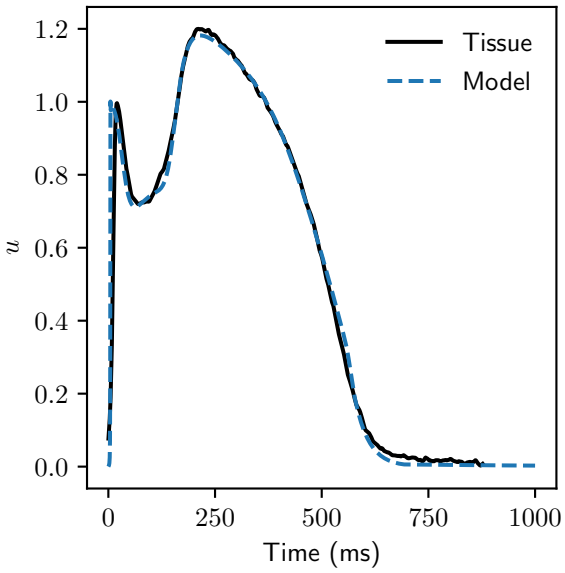

Figure 2. Fitting to a human ventricular AP with saddleback-type Brugada morphology.

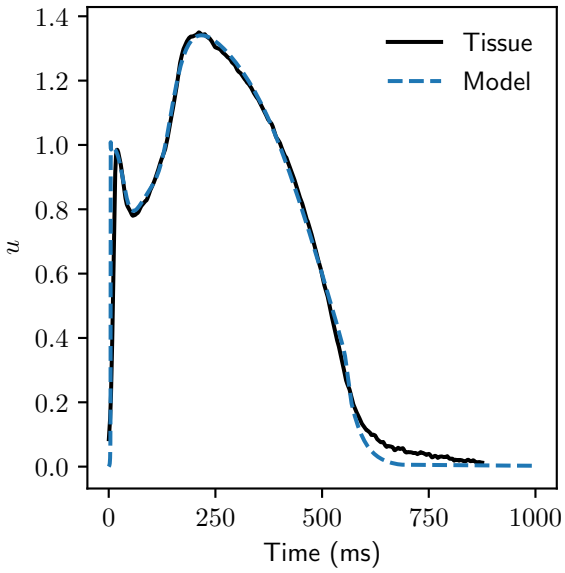

Figure 3. Fitting to a human ventricular AP with coved-type Brugada morphology.

rate of 1 ms or higher to reduce the memory consumed by the algorithm. The bottom of the page includes controls for the PSO hyperparameters, such as the number of particles and number of iterations.

As the WebGL API is supported in all modern web browsers, this software can be accessed either by visiting a website hosting the code or by downloading the files and running them as a local web server. No compilation or installation is required for the program to run. The user interface provides full access to the program's features without the need to modify any of the existing code.

3. Results

To test the effectiveness of the PSO algorithm, we use it to fit data taken from optical-mapping experiments on human cardiac tissue. In all cases, we fit a single action potential paced at a period of 1 s, recorded with a sample interval of 1 ms. All data is normalized by the PSO algorithm, with the maximum value chosen manually to produce the best fit. The PSO algorithm uses 32 iterations with 1024 particles in all cases.

We find that the PSO algorithm is able to fit the model to a variety of different action potential morphologies. Figure 2 shows a parameterization of the model targeting one of the Brugada datasets that features a saddleback AP morphology. In Figure 3, PSO was able to fit a more coved-type Brugada morphology from a different region of tissue, while fitting an additional two parameters compared to Figure 2. Figure 4 shows a third example of an action potential with a more normal shape taken from the same tissue, fitting an additional parameter when compared to Figure 3, for a total of 23 parameters being fit. All three action potential fittings closely capture the shapes of the underlying datasets.

The speed of the PSO software enables rapid results, making it easier to find good fittings quickly, and also facilitating the use of expert knowledge by tuning individual parameters and seeing the results in seconds. When run on a machine with an AMD Ryzen Threadripper 3960X processor and NVidia GeForce RTX 2080 Ti GPU, our software took less than two seconds to run 32 iterations with 1024 particles. Increasing the number of particles to 4096 did not change this result due to the extreme parallelism provided by the GPU. Increasing the number of iterations to 128 brought the running time up to just under six seconds. When tested on a laptop without a dedicated GPU, results still were obtained in less than a minute.

4. Discussion

Our implementation of the PSO algorithm is able to find parameterizations of a four-variable ventricular model that fit data taken from human tissue exhibiting Brugada syndrome. Despite the large parameter space and challenging nature of the optimization problem, solutions are found within seconds, and running the software does not require detailed knowledge of the model, implementation, or source code. Our software has the potential to facilitate using these models for patient-specific predictions in

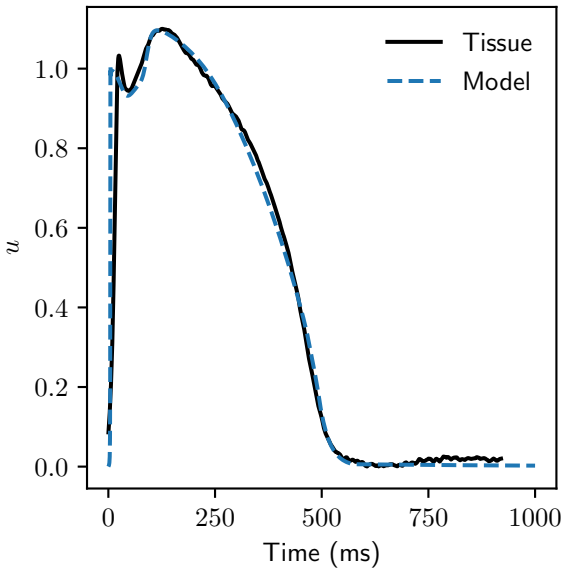

Figure 4. Fitting to a human ventricular AP with a normal morphology.

a clinical setting. In particular, our tool can be helpful for users with relatively little knowledge of the model dynamics and effects of individual parameters, but also provides experts the ability to hand-tune parameter values and bounds to constrain the solution.

Our PSO implementation has several limitations. In many cases, adjusting the normalization factor for the data can significantly improve the results of model fitting, but this procedure is not automated. This restriction is mitigated by the inclusion of an input field for the normalization value on the user interface. Another limitation occurs because individual PSO runs are not deterministic, so that re-running the algorithm will produce results that may differ in quality. Because each run is very fast, running PSO several times and choosing the best result may be the most pragmatic approach. In some cases, the PSO algorithm may overfit to the given data, resulting in a parameterization that leads to undesirable behavior. In these cases, imposing additional constraints on the parameter bounds may help alleviate the problem. A detailed study of the PSO hyperparameters and their effect on the behavior of the algorithm is necessary to ensure that the algorithm is exploited to its maximum potential. However, even with potentially suboptimal parameters, the current implementation produces useful results in the cases we tested.

In the future, use of a tissue simulation on a small domain may be considered to include the effects of diffusion, although our use of a biphasic stimulus mitigates some effects of single cells. However, such a change would increase the computational cost of the algorithm. The performance of other models, particularly those with larger numbers of parameters, still needs to be evaluated. In particular, as the number of parameters becomes large, the PSO algorithm may become inadequate to search the entire solution space, and other methods of constraining the parameters may become necessary. A possible way to improve upon the results from the PSO algorithm would be to use a second optimization method to find a local optimum near the solution provided by PSO, as in Ref. [1]. Another possible extension of this work would be to consider other types of cardiac data that do not give detailed knowledge of the action potential shape at a high time resolution. For example, iterative models of cardiac action potential duration could potentially be fit to much sparser data [5]. Even in the case of fitting continuous AP models, fits could possibly be improved by incorporating metrics in addition to the sum of squared error, such as action potential duration.

Acknowledgments

We acknowledge support for this study from NSF grants CMMI-2011280 and CNS-2028677 and from NIH grants T32GM142616 and 1R01HL143450.

References

[1] Loewe A, Wilhelms M, Schmid J, Krause MJ, Fischer F, Thomas D, Scholz EP, Dössel O, Seemann G. Parameter estimation of ion current formulations requires hybrid optimization approach to be both accurate and reliable. Frontiers in Bioengineering and Biotechnology 2016;3:209.

[2] Cairns DI, Fenton FH, Cherry E. Efficient parameterization of cardiac action potential models using a genetic algorithm. Chaos 2017;27(9).

[3] Bueno-Orovio A, Cherry EM, Evans SJ, Fenton FH, et al. Basis for the induction of tissue-level phase-2 reentry as a repolarization disorder in the Brugada syndrome. BioMed Research International 2015;2015.

[4] Clerc M, Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 2002;6(1):58–73.

[5] Qu Z, Shiferaw Y, Weiss JN. Nonlinear dynamics of cardiac excitation-contraction coupling: an iterated map study. Physical Review E 2007;75(1):011927.

Address for correspondence:

Elizabeth M. Cherry
School of Computational Science and Engineering
Georgia Institute of Technology
756 West Peachtree Street Northwest
Atlanta GA 30332-4017
elizabeth.cherry@gatech.edu