Novel Error Function Computations for Dynamically Modeling Spike Train Data of the Aplysia Buccal Ganglion Involved in the Feeding Circuit

Zachary P. Kilpatrick[†] Steven J. Cox[†] John Bachir[†]

Randall D. Hayes[II]

[†] Computational and Applied Mathematics
Rice University
6100 Main Street, Houston, TX, 77005

[II] Randall's address

Abstract

Spike trains in timecourse plots of neuronal membrane voltage potential are a series of steep departures and nearly instantaneous returns to baseline potential. Dynamical systems that attempt to model the time courses of a system of synaptically connected neurons' voltage can be stiff, nonlinear, coupled ordinary differential equations (ODEs). Once dynamical systems are characterized, their free parameter constants must be determined so that the model accurately reproduces the data points of the true system. Using Euler's method for computational integration of these systems along with a finite difference method for objective function computation can be time consuming and inaccurate. More robust methods and smoother objective functions are necessary so that nonlinear least squares optimization can pinpoint a global minimum for the objective function with satisifying brevity. We propose using the difference in running averages of voltages as the error function onto which will be augmented a Lagrange multiplier of the dynamical model system. Rather than using finite differences to computes the gradient of this function and the direction of steepest descent, we use the adjoint method which requires drastically fewer evalutations of the objective function. When applied to a model ODE system of neurons, approximating time data of Aplysia ganglion feeding cicuit neural cells, results are drastically better than the crude, standard approach.

Contents

0.1 Introduction to Fitting Spike Trains 2

0.1.1 Motivation 2

0.1.2 Common Modeling Techniques 3

0.1.3 Single Cell Example – B34 4

0.2 Modeling the Neural Network Responsible for Aplysia Feeding . 10

0.2.1 Neurons Responsible for Determining the Feeding Circuit 10

0.2.2 Writing and Solving Equations for Cell and Synapse Dynamics 12

.1 Analytical Explanation 14

.2 Adjoint Method in MATLAB 17

0.1 Introduction to Fitting Spike Trains

0.1.1 Motivation

Such a class of data is specific to the shape that a time course has within, once integrated by a stiff enough ODE solver. These voltage signals are a means for neurons to signal each other, either through chemical or electrical synapses. In a chemical synapse, a rise in the membrane potential of a neuron lead to the release of neurotransmitter (APCD for Aplysia) [9]. Rise in the concnetration of this chemical in the vicinity of the postsynaptic cell leads to a an eventual rise in membrane potential if an appropriate threshold of depolarization in reached on account of the transmitter. Even more rapid is an electrical synapse that effectively receives current from its neighboring cell the way that a capacitor in a circuit does. With enough of these two types of connections and a few cells, a network can perform simple tasks or transition a creature between different

states of tasks. Mathematical models of these networks allow one to recover the "switch" components, cells that shift the network in between different states; the most important cells, that when removed cause the network to malfunction or terminate operation; and the unimportant cells, that can be done without to an extent. Akin to all information attained from mathematically approximatory models, the results' reliability is constrained by the quality of the model. Model quality is determined by how accurately it can reproduce accurate true data, which is only possible if there is an available method for determining all aspects of a mathematical model.

0.1.2 Common Modeling Techniques

Many approaches exist for modeling the beautifully synchorized behavior of neural network membrane potential time courses. Hodgkin and Huxley were the first to propose a gating variable approach to handle ionic flow that affected membrane voltage potential. Based on experiments run on a giant squid axon, appropriate parameters were recovered for their model with experimental data [8]. Single neurons can be modeled with the following system of ODEs.

$$\frac{dV}{dt} = -G_{Na}m^3h(V - V_{Na}) - G_K n^4(V - V_K) - G_{Cl}(V - V_{Cl}) + I_{stim} \quad (1)$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n \quad (2)$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \quad (3)$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h \quad (4)$$

Here, we have V as voltage of the cell; G_{Na}, G_K, and G_{Cl} are the conductance/area of open sodium, potassium, and chloride channels respectively; m and h are the activation and inactivation rate of sodium channels; V_{Na}, V_K, and V_{Cl} are the "half potentials" of sodium, potassium, and chloride channels; n is the activation rate of potassium channels. Finally, the system is stimulated by some current I_{stim} divided by the surface area of the cell. The α's and β's are constants determined for specific cell types: like squid, octopus, Aplysia, rat, or human. Another method, used when many cells comprise a neural network is integrate-and-fire [4], [6], [18].

Whereas the Hodkin-Huxley equations are four per cell - or even more depending on the variety of channels present - integrate-and-fire (IF) essentially uses one ordinary differential equation, if space is not considered. Consider the following template for an IF model. Spikes are simulated using leakage of potential and an incoming current. This replaces all of the ionic currents present

in the Hodgkin-Huxley model [18].

$$\frac{dV(t)}{dt} = -\frac{V(t)}{\tau_1} + c_1 I(t), \tag{5}$$

where $\tau_1 > 0$ is the membrane time constant and c_1^{-1} is a capacity. V is reset when it gets to be above some threshold, say F, and the intial conditon will be that $V(0) = 0$ so consider this the baseline and shift to fit data accordinly [18]. Of course, IF models can be written more extensively to include more dynamics of a cell system, but for the purposes of this research [4], citegolerm1, Hodgkin-Huxley like models were used for their robust capturing of spike train properties in real time in neuronal cell cicuits [1].

Dynamical systems are common templates for mathematical modeling of potential spike train data in neuronal circuits [1], [3]. Usually, a first order nonlinear coupled system of differential equations does the trick. Spiking cells can indeed have their voltage, ion flow, and synapses modelled by some adaptiation of the Hodgkin-Huxley equations, where freedom is given to constant parameters after the number of cell sand connections in the netwoks have been determined.

$$y'(t) = F(t, y(t), p), \quad y(0) = y_0 \tag{6}$$

Assume that we will know F entirely down to the freedom of changing the parameter vector p. Often this problem arises in modeling and applied mathematics refers to it as a coefficient inverse problem. All aspects of the model are known excepts for constants. Recovery of these elements seems a task simple enough for today's prerolled model fit routines. However, standard methods stall or are inaccurate as they are not adapted to handle *stiff* ODE systems, which include proportionally large changes in variables over short periods of time followed by longer time spans of little change. Necessary it is then to account for the unique stiffness of this dynamical ODE system. MATLAB has produced many ODE solvers that are suitable to handling the difficulty of stiff ODE systems [16]. However, standard error functions like point to point comparison take too long and thus an adaptive step running average needs to be implemented to quickly recover appropriate parameters.

0.1.3 Single Cell Example – B34

Aplysia calfornica have around $20,000$ neurons in their ganglion and as they are invertibrates, these are the controlling cerebral force in their bodies responsible for all motor and life functions. Cells dictating interneuronal, which does not involve motor function direction, activity of the seahare's feeding circuit activity. Feeding involves chewing, ingestion, and egestion. Toothed radula, which are part of the creature's mouth protract to receive food and then retract to pull it

in. Rather than excretion from a different cavity, waste material exits through the same aperture through which nourishment enters. Retraction prepares the food within the gut and protraction expells the waste. Neural cells switch the motor functions of the animal in between different states of the feeding circuit at large as well as between protraction and retraction of the toothed radula. One of these neurons is B34 which has actually been shown as a plastic switch between protraction and retraction states depending on the chemical and electrical input it receives from different interneuronal cells [9]. Consider the following diagram taken from a paper documenting experiments done upon the *Aplysia* to recover different cells and their affects on B34 during a number of feeding stages. B34 fires only during egestion like programs, but it is questionable whether it affects the toothed radula so that they will protract or retract [9]. This could be determined with an accurate model of the associations between B34 and the neurons that chemically or electrically synapse onto or from the neuron. Below is an experimental determination.

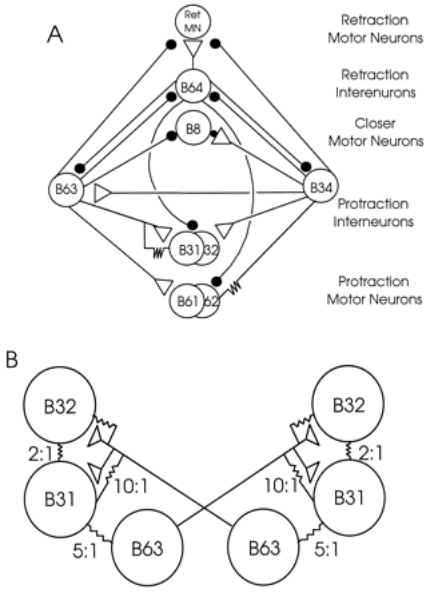

Figure 1: ***A***: diagram summarizing synaptic connections between neurons discussed in this paper. Note that weak coupling between B34 and B31/B32 [17] is not shown. Also, amplitude of connection is generally not indicated. One exception is that mixed excitatory-inhibitory synapse from B34 to B8 is shown with small inhibition and large excitation to emphasize that connection is primarily excitatory. In addition, there is no indication of whether connection is ipsilateral, contralateral, or bilateral. ***B***: diagram showing relationships between bilateral B63 and B31/B32 neurons. Open triangles: excitation. Filled circles: inhibition. Electrical connections are shown by resistors.

Were one to attain an accurate mathematical model, the affects of depressing or sensitizing certain cells in this network would become apparent. Removing a

single cell may have an important or unimportant affect on the cell cycle, but the result could be determined without requiring wet lab research. This begins with the modeling of single cells. As the B34 cell was already well determined by the Byrne group through a series of experiments and model fits. The task of recovering the appropriate parameters from some discrete set of data for this single cell would be a way to kick the tires on the algorithm presented in the **Appendix** as the adjoint method for computing the gradient of a running average error function.

Data is included as a matrix of values in an ODE template that MATLAB can read as the dynamical system model and available discrete data to be fit. In order to compare with standard methods, the first experiment is to simply allow the built-in minimization function to run a point by point Euler comparison of data followed by the use of finite differences to compute the gradient and direction of steepest descent to find parameters. This crude, standard method will be compared with with running averages using the adjoint method for computing the gradient [2]. Appropriate parameters are already known from the fit model and the synthesized data taken from a conference proceedings [7].

In the first experiment, the following fit of the model (dotted line) compared with true data (solid) was compared to show the success of the method. With a step size set to $10^{-5}s$, after waiting 13 iterations and 3004 seconds for the computations to attain a fit of a the single B34 neuron's spike trains resulting from a periodic stimulus. Parameters were the conductance of the sodium channel (p1) and conductance of the potassium channel (p2). We began the parameter guess at p1=10 and p2=20 so that there would not be so far to go to recover true parameters.

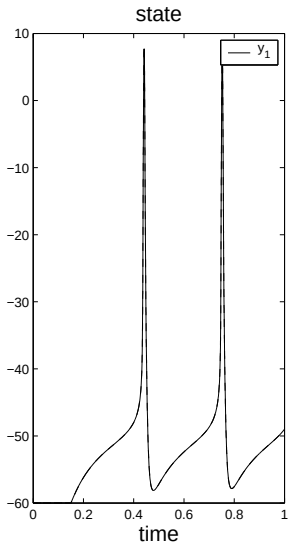

Figure 2: Recovered parameters are accurate and the fit is fine, but in order to accomplish this, the Euler step for point by point comaprison had to be reduced to $10^{-5}s$. This expensive measure cause the method to take 3000s in order to finally attain an accurate fit. Comparison

with the running average method will display whether time can be reduced with the use of more efficient tools.

With running averages as an error function will smooth out the spiky comparison error function drastically. This should have made the method more able to handle the fit because the gradient can be computed more quickly. Certainly, the inital conditions were the same and the method proved better at handling iterative computation. Look at the below graphic of the model to the true data.

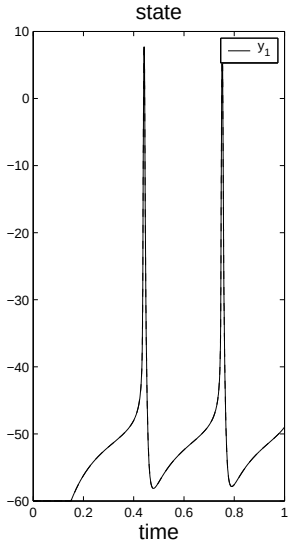

Figure 3: Although the fit is the same, this was acheived in 16 iteration that took a total of 1696 seconds. Clearly, the quicker computation of the gradient ended up with perhaps similar accuracy but in a must more accelerated fit of the data at hand.

Note that this is synthesized data, if we were to throw noise on top of this data this would prove a problem that often arises when collecting data is not so bad during the fit. The fit below is to noisy data.

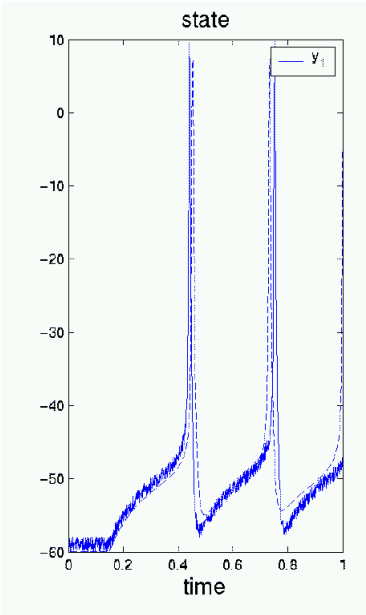

Figure 4: Recovered Parameters were approximately p1=7 and p2=18, whereas true parameters were p1=7.5 and p2=25. This undoubtably attests to the difficulty of fitting noisy data as the template to which one's model is being judged against is incorrect. However, the spikes settle in general vicinity of one another and there was a large improvement from the intial guess.

Considerations that are accounted in this small model are the multiple local minima that can be quite severe when there are several spike trains being fit. A model spike may line up with a data spike and it may not be the right spike, but to the computational error it appears as a good spot for the model to rest. Also, even when the model and data spikes are quite close together, the fact that they are not right on top of each other severely penalizes most error functions. Larger networks will illuminate this problem to a greater extent.

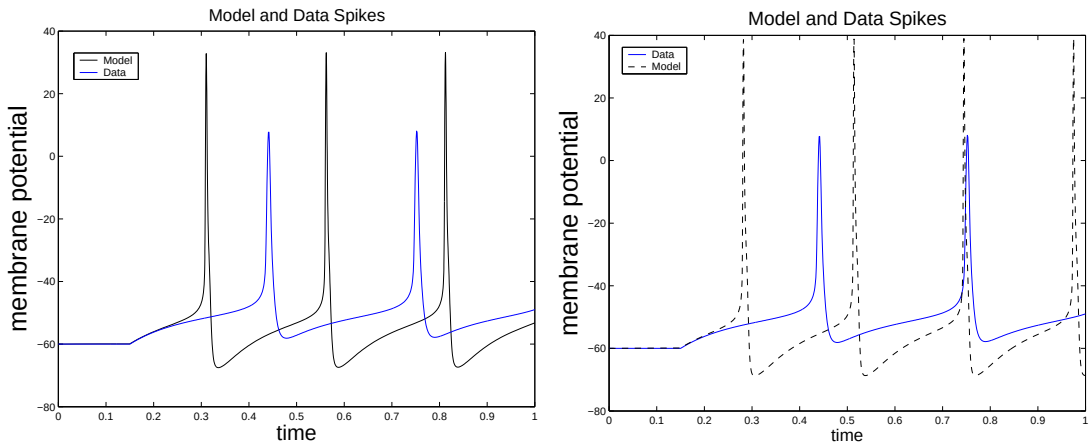

Figure 5: *Left*: Error functions computed for a point to point comparison would clearly fail here in attaining some semblance of the best direction to move as either shifting spikes to the left or right would improve the error function if the spikes settled upon one another. Despite closeness of model spikes and data spike to one another, error functions can be drastically penalized if spikes are not right on top of one another in a point by point comparsion. *Right*: Notice the third model spike has settled on top of the data spike and, likely, the model attains a local minimum here, which the error function would penalize a move from. Often point to point comparsions have this problem of being stuck on data spikes which are the incorrect spikes for the model spike to fit.

Single cells can illuminate properties of modeling and fitting those models' parameters on a small scale. B34 was much better fit when using the adaptive step, running average, adjoint gradient method as oppose to standards. Spike trains are a special cases that require attention to their standout uniqueness when fit. Error functions that account for stiff ODEs can be valuable for applications outside of spike trains however. For now we continue on with the large scale model already extensively worked upon by Randall Hayes.

0.2 Modeling the Neural Network Responsible for Aplysia Feeding

0.2.1 Neurons Responsible for Determining the Feeding Circuit

We might class these neurons into four types, ones that affect protraction of the toothed radula; those that affect retraction; those that initiate patterns in the neural circuit; and those that affect motor processes. Below is an itemization of these cells

Protaction	B20, B31/32, B34, B40, B52, B63, B65
Retraction	B4/5, B34, B51, B64
Patterns	CBI-2, CBI-3
Motor	B8

We begin by examining a brief overview of each cell provided by several biological experiments. Perhaps, first, the most interesting is the cerebral buccal interneuron 2 (CBI-2) as it is considered a command-like neuron responsible for pattern generation throughout the network [13], [14]. CBI-2 can elicit both ingestive and egestive responses for the Aplysia, and it may be activated alone or with CBI-3 which can change to motor programs. When the cell signals the B20 cell, for example, this begins a response of protraction of the toothed radula. B4/5 are being driven then at the same time, which are responsible for retraction. These mediator cells assist CBI-2 in egestive motor programs, but evidence support that ingestive programs also arise by the control of this interneuron of its motor program mediators. Therefore, the cell displays some important properties as a traffic director of the neural networks response to stimuli. If it is coactivated with CBI-3, then its elicited egestive programs may be converted into ingestive ones. Firing of B20 can convert CBI-2 elicited ingestive programs into egestive ones. Hyperpolarization of B20 switches CBI-2 -elicited egestive programs to ingestive ones. Connectivity between these neurons is fairly clear.

As mentioned, CBI-3 functions as a neuron that when coupled with CBI-2 can change, in effect, that interneurons function from egestive to ingestive programs [13], [10]. Therefore, one might consider that neuron a switch for a switch. Since the network, during its generating pattern, may alternately protract and retract the radula, these functions are meaningful in the context of the current feeding phase. CBI-3 holds importance by determining this feeding phase if it does or does not fire with CBI-2.

Taking the B20 neuron, this is a cell affected by the activation of CBI-2 [13]. Obviously there exists a linkage here between the cerebral and messenging interneurons. B20 is responsible for the protractive phase of ingestion when its cyclic patterns are initiated. This cell's alternation with B4/5 caused the

oscillatory movement of the radula. Now the example neuron taken earlier, B34 can switch roles depending on the phase of the network, but it has been shown to be a cell responsible for protraction during egestion [10]. Protraction causes rejection of material from the Aplysia gut. However, evidence support that the cell may have a role to play in ingestion as well [9]. The model should reflect this behavior.

Persistently firing cells do not have the common spike train sequence of abrupt changes to the membrane potential, there is a slow rise and plateauing of potential. During protraction, B31 undergoes this change to its voltage from a state of rest [10]. With a persistent sodium channel and leakage current as the dictating forces in opposition for its excitation, the cell drives other protractor cells with the persistenly "on" quality.

Considering B20, this is also a protractor cell, but this is active during ingestive and egestive motor programs alike [13]. CBI-2 drives the central pattern generator of this neuronal network system. Model should capture the firing of this neuron along with other protractors during the protraction phase and not during retraction, at least to not the extent of activation.

B63 has also been shown to be a protractor neuron [9]. See **Figure 1** and find that it is interconnected to other protractor neurons like B34, which upregulates it activation. Noticably, retraction interneurons also down regulate its activity so that when cells like B64 begin to fire, the potenital of the B63 cell is inhibited. Activity in the model would be spike trains during protraction and flatline likely during retraction.

B35 plays a major role in protraction and exctitation of other protracting interneurons. Experimentally, it has been shown part of the central pattern of the radula opening and closing through its activation during protraction [17]. Interstingly, the B52 neuron has been shown to play arole in terminating retraction and beginning protraction [5]. As it fires during ingestive programs it may be specific to this phase of feeding but certainly its importance would be held as a protractive neuron for changing the state of the patterned response.

As has been mentioned, B20 is quite important for protraction and its activity alternated with that of B4/5 in experiments [13]. However, it is also shown to gate between ingestion and egestion, serving a dual purpose. This implies that it has several linkages to many cells that a simple protracting cell may not have [13]. Long protraction has been shown to be articulated by the B40 cell, when depolarized extensively [14]. B65 is the last protracting interneuron to be mentioned here; it elicits a slow persistent synaptic potential in B8 and has patterned activity [15]. B4/5 can be consider somewhat of one neuron as the electrical coupling is so close. This pair's activation during retraction makes its electrical spike trains opposite those like B20 as the patterned firing dictated by CBI-2 has them shuttle in between activation and in activation as the radula protract and retract [13]. Additionally, the B51 elicits a plateau potential during ingestion [5]. During rhythmic activity is switches in between active and inactive states. It is active when the radula are retracting. Consider it to be a retractor cell then.

B64 is responsible for the switch between protractive to retractive move-

memnt f the radula [11]. Very necessary as an element of the central pattern generator, it should have plenty of connections to other neurons to inform of the switch and perhaps there will be constant spike train activation during the time of protraction for this cell.

The motor neuron, B8, should fire in phase with the interneurons responsible for the current activity of the *Aplysia* in its feeding circuit. It would fire at the same time as the protractor neurons () if the toothed radula were undergoing protraction and likewise in phase with retractor neurons () during retraction. Exceptional dynamic models of this neuronal circuit should capture the pattern observations of the many experiments run to find "switch" neurons, those that can change function depending on the state of the network [13], [14], [17]. Note how B34 is classified as both an egestion and ingestion neurons [9]. The neuron has been shown to serve as both, experimentally, depending on the state of the network patterns. From experimental data, the connections between cells has been determined by stimulating one cell and taking note of the electric potential time course in other cells. These are the type of experiments that have aided in recovering the connections and important cells. Below is a conceptual schematic of the ways that all of the most important cells of the feeding circuit network affect each other. These connections are synapses.

0.2.2 Writing and Solving Equations for Cell and Synapse Dynamics

Cells and connections have been conceived and so the task in then to model this mathematically. As synapses can be either electrical or chemical, there are two separate ways to model synapses depending on which class they are. Whereas electrical synapses occur essentially in real time as it is simply a transfer of current, this is a single term in the potential of the cell receiving the synapse (the postsynaptic cell). Chemical synapses require accounting for the neurotransmitter and so one must include a variable for synaptic neurotransmitter (APCD) concentration for each chemical synapse. Thus, in addition ot the Hodgkin-Huxley dynamics apparent in each cell, potential ODEs will have terms for the electric synapse simply of form

$$g_{syn}(V_{post} - V_{pre}). \tag{7}$$

While chemical synapses will require terms based on neurotransmitter concentration, a dynamic variable, multiplied by the current cell potential shifted by a half potential constant,

$$g_{syn} f_{syn}(V_{post} - V_{\lambda}). \tag{8}$$

Coding this for all cells and synapses of the feeding circuit, the task then remains to recover the appropriate synaptic connectivities (g_{syn}) to form a network that

can shift between two recognizable patterns. Protraction and retraction may be switched in between depending on the cell that is stimulated. This natural functional change would be reflective of a mathematical model with the ability to display plasticity in a cell or collection of cells.

For practical purposes MATLAB was selected to transfer the findings of Hayes from Simulator for Neural Networks and Action Potentials (SNNAP) [19]. Hayes had been using PEST, a parametrization tool that he had coded that worked with simply Euler's method to solve the ODE system; finite difference to find the gradient; and steepest descent then as a search direction for the parameter space. Equations have been written and parameter approximations have been made, but the results do not show perfect success in the model system's behavior. Attached are plots of voltages over a 30 second time stimulating the B63 cell to initate network firing patterns.

Figure 7: Clearly, some of the protractor cells fire in phase, but the alternation present in several experiment that shows protractor and then retractor firing is not entirely present. [See figure on last page after refs]

Therefore, in order to take advantage of the tools constructed in MATLAB (see **Appendix**), and arrive at better parameter fits, the ODE system needed conversion to MATLAB. Boolean statements as well as dynamical systems carried over, and the network was a quite large 222 ODEs when fully converted. For a stiff ODE solver that demands some low tolerance of accuracy, this is cumbersome to simulateously computed. As the system is nonparallelizable, one must search for the fastest way to solve a good old coupled nonlinear system of ODEs.

Bachir looked at methods in Fortran, a faster but more opaque language than MATLAB, for solving ODE system (`vodpk.f`) and coupling this with the `minpack` toolbox for optimization methods in Fortran. Essentially, this is the machinery that was used in MATLAB in order to fit smaller neural networks or single cells like B34. With these tools at our disposal, we should likely be able to enact the use of the adjoint method for nonlinear least squares and recover parameters that work to give the network two different phases. A robust network should have stability as well, so the certain time phase of radula movement should maintain until another signal or depression is strong enough to switch the pattern. Clean switches would accurately demonstrate properties observed experimentally [13], [14], [12], [9], [10], [15], [17].

.1 Analytical Explanation

Introduction

Take dynamical system of the form

$$y'(t) \quad = \quad F(t, y(t), p), \quad y(0) = y_0 \tag{9}$$

F maps from $\mathbf{R} \times \mathbf{R}^N \times \mathbf{R}^M$ to $\mathbf{R}^N$.

Full Measurement

If one assumes knowledge of the values at virtually all points t entire vector of functions y which is a vector of functions, which is the case for real time data collections, then we can use piecewise computational interpolation and call this true data, $y^{\#}$, on interval, $[0, T]$, and we would like to find the appropriate parameterization p such that this data is then produced by a model function $y(t; p)$. Thus, the minimization problem arises

$$\min_{p \in \mathbf{R}^M} J(p), \quad J(p) \quad \equiv \quad \int_0^T ||y(t; p) - y^{\#}(t)||^2 dt \tag{10}$$

and $y(t; p)$ is the solution to (1) for our current choice of p. Our interest is in an effective way of computing the gradient of the objective function, J, in (1).

Beginning with equation (2), we can consider its Lagrangain by including the initial value problem (1) with the objective function as an inner product with adjoint function $z(t)$ in the L^2 norm shown below in (9). That is, we ask to minimize the intergral over the entire interval of the discrepancy between $y'(t)$ and $F(t, y(t), p)$ multiplied by z. Adding this to the original minimization problem in (2) allow for IVP satisfaction.

$$L(y, z, p) \quad = \quad \int_0^T ||y(t) - y^{\#}(t)||^2 dt + \int_0^T z(t) \cdot (y'(t) - F(t, y(t), p)) dt \tag{11}$$

This turns the original constrained optimization probem into an uncontrained optimization problem that requires a few more steps of computation. Notice

within the first integral, what was $y(t;p)$ is now $y(t)$, which removes the complication of computing the derivitive in the direction p. First we compute the $\widetilde{y}$ direction derivitive

$$L_y(y,z,p)\cdot\widetilde{y} \quad = \quad 2\int_0^T (y(t)-y^{\#}(t))\cdot\widetilde{y}(t)dt + \int_0^T z(t)\cdot(\widetilde{y}-F_y(t,y(t),p)\widetilde{y}(t))dt \tag{12}$$

If we integrate this expression by parts, we get

$$L_y(y,z,p)\cdot\widetilde{y} \quad = \quad \int_0^T (y(t)-y^{\#}(t))\cdot\widetilde{y}(t)dt + z(T)\widetilde{y}(T) - \int_0^T (z'(t)+F_y^*(t,y(t),p)z(t))\cdot\widetilde{y}(t)dt \tag{13}$$

In every direction that $\widetilde{y}$ can go, this must go to zero. The only way this is possibel is if that which is multiplied by $\widetilde{y}$ within the integral is itself zero. This gives us

$$z'(t)+F_y^*(t,y(t),p)z(t) \quad = \quad 2(y(t)-y^{\#}(t)), \quad z(T)=0. \tag{14}$$

Both of these ODEs can then be solved for y and z. This allows us to then compute

$$J_p(p)\cdot\widetilde{p} \quad = \quad L_p(y,z,p)\cdot\widetilde{p} = -\int_0^T z(t)\cdot F_p(t,y(t),p)\widetilde{p}dt \tag{15}$$

Spiky Data

Expanding this idea to account for spiky data, we would have the objective function J be a difference of *integrals* so that the spikes are smoothed ina sense and ad infinitum adaptive step shrinkage is prevented. So J takes the form

$$J(p) \equiv \int_0^T ||Y(t;p)-Y^{\#}(t)||^2 dt \tag{16}$$

where Y is a running integral of y, i.e.

$$Y(t;p) \equiv \int_O^t y(s;p)ds. \tag{17}$$

We now have the leading term of the y-derivitive of the associated Lagrangian is

$$2\int_0^T (\mu(T)-\mu(t))\cdot\widetilde{y}(t)dt \tag{18}$$

where

$$\mu(t) \equiv \int_0^t (t-s)(y(s) - y^{\#}(s))ds \tag{19}$$

Pointwise Measurement

Here, we only have specific points whose data in between would be lost on interpolation. We have the states at specific times, $t_k{}_{k=1}^{K}$, our objective function will then be

$$J(p) \equiv \sum_{k=1}^{K} ||y(t_k;p) - y^{\#}(t_k)||^2 = \int_0^T \sum_{k=1}^{K} ||y(t;p) - y^{\#}||^2 \delta(t-t_k)dt \tag{20}$$

Now all that changes is the right hand side of the adjoint equation. Namely, we must solve

$$z'(t) + F_y^*(t,y(t),p)z(t) = 2(y(t) - y^{\#}(t)) \sum_{k=1}^{K} \delta(t-t_k), \qquad z(T) = 0. \tag{21}$$

Between each measurement time, only a simple problem is required. First

$$\begin{aligned} z'(t;K) + F_y^*(t,y(t),p)z(t;K) &= 0, \quad t_{K-1} < t < t_K, & (22)\\ z(t_K;K) &= 2(y^{\#}(t_K) - y(t_K)) & (23) \end{aligned}$$

next

$$\begin{aligned} z'(t;K-1) + F_y^*(t,y(t),p)z(t;K-1) &= 0, \quad t_{K-2} < t < t_{K-1}, & (24)\\ z(t_{K_1};K-1) &= z(t_{K-1};K) - 2(y(t_{K-1}) - y^{\#}(t_{K-1})) & (25) \end{aligned}$$

and in this fashion until

$$\begin{aligned} z'(t;1) + F_y^*(t,y(t),p)z(t;1) &= 0, \quad 0 < t < t_1, & (26)\\ z(t_1;1) &= z(t_1;2) - 2(y(t_1) - y^{\#}(t_1)) & (27) \end{aligned}$$

Thus are some basic analytical methods using the adjoint for gradient computation. Using this computationally will require some software and user understanding of inputting.

.2 Adjoint Method in MATLAB

With the analytical result at hand, automating this process in a choice computational tool suited to handle dynamical ODE systems and minimization will test its success in practice. MATLAB is a most accessible and sensible language to begin with, and the method was coded as an m-file function set to return `bp`, the best fit parameters for the system. Required are the seven arguments of `sanlsq(name,T,C,grad,meth,p,err)`.

An acutal model ODE system named `name` must be constructed as a symbolic function that returns (in the following order): `y`, a symbolic vector of the y variable of the system $y'(t) = F(t, y(t), p)$; `p`, the symbolic vector of parameter coefficients; `F`, the vector $F(t, y(t), p)$; `ts` times for all discrete measurements of the true data; `ys` a matrix of the data availabel for any of the y variables at times `ts`; `LB, UB`, lower and upper bounds on parameter estimates (`LB(i)<=bp(i)<=UB(i)`); and `y0` initial conditions for all y variables. Assignment of the function may look like

```
function [y,p,F,ts,ys,LB,UB,y0] = system
```

Assuming the interval of integration begins at time zero, the end of the interval is defined by the user as the second option, `T`. Now, the third option is of importance if the user has an incomplete data set. States to be observed, `C`, would be set to some vector of length less than the total number of ODEs, `y`. Flagging for whether or not the gradient of the dynamical system is know helps to save time if it is known. However, even if the gradient is not known, the function `fmincon` will compute it within using very short time-steps. Clearly, the function will run faster if this is not necessary and `grad` is set to `on`. MATLAB has seven built-in ODE solvers, any one of these can be typed as a string for the fifth option `meth`: `ode15s` and `ode23s` work quite well for stiff ODE systems as they are stiff solvers. Finally, the parameter guess is `ip`, it must be as long as `p` from the ODE system file and within the bounds `LB` and `UB`. Finally, `err` is a number which selects the error function to be used: `1` is the simple point by point difference error; `2` is the cumulative voltage integral; `3` runs a maximum and minimum comparison error. Thus explains the seven inputs to the outer shell of the computational implementation of the adjoint method for solving nonlinear least squares probem: `sanlsq(name,T,C,grad,ip)`. The inner functions `agen.m & Gmisfit.m` do most of the heavy lifting to follow.

As the user provides the function with a symbolically written dynamical system with parameters to be found, `agen` can then produce the adjoint ODE as listed in the analytical section under equation (12). First, there is a check made to make sure that the user has provided a system which contains all appropriate syntaxes, `p=[p1 p2 ...], y=[y1 y2 ...], & F(1)= ....` If this is so, then the jacobian is constructed by the `jacobian` function, which acts on appropriately written symbolic systems such as this, for the y and the p direction. These are the functions $F_y^*(t, y(t), p)$ and $F_p(t, y(t), p)$, respectively. They will be later plugged into the function that we wish to minimize in order to compute the

values for y and z so that the next best guess for p can be found. Automated construction of functions `sysname, adjname, & jacname` that can be used by the method saves the user the task of having to construct these. Functions are written as `*.m` files fully usable by MATLAB.

Now that we have the elements for an adjoint solve of the optimization (F, F_y^*, and F_p) they can be fed into a function minimization routine, particularly here `fmincon` was used because the parameters were constrained by an upper and lower boundary. All elements necessary for the evaluation of `Gmisfit` are fed into `fmincon` as well so that the function can be examined and further minimized at each iteration. Necessary inputs to be provided for `Gmisfit(p,sys,adj,jac,y0,ts,ys,T,C,af,meth)` are mostly defined by the user, but the functions have been constructed in `agen`. `p` is a vector of parameters that is essentially a guess at the best parameters to minimize `Gmisfit`. Of course the best value for `p` would produce a minimum value for `Gmisfit`, which is a scalar. `sys` is the tag for the system of equations generated from the user defined symbolically constructed dynamical system. `adj` is the F_y^* function, and `jac` is the jacobian or the gradient of F in the p-direction, F_p. Remaining specifications are provided by the user. $y0$ are initial conditions for the system and ys is a matrix of a column of times followed by the y values of the system for each element in the y vector at that specific time. Now `T & C` serve the same role as they did in `sanlsq` as the termination time and the states to be observed by the minimization routine, respectively. For use of the adjoint gradient, `af` is a flag which states whether or not the gradient of the function will be used within the misfit function in order to expedite the process of minimization. If it is listed as `off` then it will be necessary for `fmincon` to compute the gradient, computationally, on the fly, which can become quite expensive, depeding on the size of the system. The user selected ODE method has its handle passed as `meth`. Using an adjoint system, the costate function includes the gradient as information in the minimization routine to provide a more enlightened view of the function as such. Both the costate is evaluated using the specified ODE method and the gradient is evaluated using the trapezoid rule. Both the gradient of F with respect to p for each time and the costate are multiplied each iteration to attain the Lagrangian with respect to p, whose minimization yield the next best parameter guess.

Now, depending one which error function is selected for use, a different Gmisift will be called. Voltage-time series, the first, will first compute `val` as the integral of the difference of each point selected by the adaptive step ODE solver compared between the model points and true data points. Cumulative voltage integral requires the computation of μ at each time point selected by the ODE solver. Final computation involves the integration from $0 \rightarrow T$ for all μ. Trapezoid rule will work fine for this function that is smoother than voltage-time series. Finally, problems that may arise in only using voltage-time or cumulative integral as the sole error function may be that the initial guess is so far from the global minimum that `fmincon` cannot see the neccessity of moving a far distance in the phase space. Maximum and minimum comparisons produce wide sites of steepest descent not captured by the narrower specifications of previous error

functions. Attaining some rough approximation of parameters the more specific error functions may then be of use with the refined guess. Looking back, the order of events through `sanlsq` are as follows. First `agen` is called to produce the `sys, adj, & jac` functions which are F, F_y^*, and F_p respectively. These are written to their own file. Next, the functions are passed in, with `Gmisfit` into `fmincon` which is a computational tool for constrained optimization. The options provided check `Gmisfit` at each iteration and enlist the aid of the computed functions to improve the parameter guess until the function terminates on account of reaching a minimum or maximum iteration count. Returned from `Gmisfit`, `val` is the function $J(p)$ while `grad` is the gradient of $J(p)$ with respect to p. `fmincon` uses all of these values to decide which direction to move along the p vector. Optimization with this function terminates when the error ceases to change by some certain tolerance each step or when some maximum number of iterations is reached. Best parameters are printed to screen.

Bibliography

[1] R. Calbrese, A. Hill, and S. Hooser. Realistic modeling of small neuronal circuits. In Erik DeSchutter, editor, *Computational Neuroscience*, chapter 10. CRC Press, New York, 2001.

[2] Steve Cox. Gradients and nonlinear least squares. Describes adjoint method for model optimization., May 2004.

[3] P. Dayan and L.F. Abbott. *Theoretical Neuroscience*. MIT Press, Cambridge, MS, 2001.

[4] G.B. Ermentrout and N. Kopell. Fine structure of neural spiking and synchronization in the presence of conduction delays. *PNAS*, 95:1259–1264, 1998.

[5] C.G. Evans, S. Rosen, I. Kupferman, K. Weiss, and E.C. Cropper. Characterization of a radula opener neuromuscular system in aplysia. *J. Neurophysiology*, 76:1267–1281, 1996.

[6] David Golomb and G. Bard Ermentrout. Continuous and lurching traveling pulses in neuronal networks with delay and spatially decaying connectivity. *PNAS*, 96:13480–13485, 1999.

[7] Randall Hayes, Jack Byrne, Doug Baxter, and Steve Cox. Estimation of single-neuron model parameters from spike train data. Different error functions to be minimized for model fitting.

[8] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (London)*, 117:500–544, 1952.

[9] I. Hurwitz and S. Fomin. Different roles of neurons b63 and b34 that are active during the protraction phase of buccal motor programs in aplysia californica. *Journal of Neurophysiology*, 78:1305–1319, 1997.

[10] Itay Hurwitz, Irving Kupferman, and Klaudiusz Weiss. Fast synaptic connections from cbis to pattern generating neurons in aplysia: Initiation and modification of motor programs. *Journal of Neurophysiology*, 89:2120–2136, 2002.

[11] Itay Hurwitz and Abraham Susswein. B64, a newly-identified central pattern genrator element producing a phase switch from protraction to retraction in buccal motor programs. *J. Neurophysiology*, 75:1327–1344, 1996.

[12] Jian Jing, Ferdinand S. Vilim, Jin-Sheng Wu, Ji Ho Park, and Klaudiusz R. Weiss. Concerted gabaergic actions of aplysia feeding interneurons in motor program specification. *J. Neuroscience*, 23:5283–5294, 2003.

[13] Jian Jing and Klaudiusz R. Weiss. Neural mechanisms of motor program switching in aplysia. *J. Neuroscience*, 21:7349–7362, 2001.

[14] Jian Jing and Klaudiusz R. Weiss. Interneuronal basis of the generation of related but distinct motor programs in aplysia: Implications for current neuronal models of vertebrate intralimb coordination. *J. Neuroscience*, 22:6228–6238, 2002.

[15] E.A. Kabotyanski, D.A. Baxter, and J.H. Byrne. Identification and characterization of catecholammerigic neuron b65, which initiates and modifies patterned activity in the buccal ganglion. *J. Neurophysiology*, 79:605–621, 1998.

[16] Lawrence Shampine and Mark Reichelt. The matlab ode suite. *SIAM J. Scientific Computing*, 18:1–22, 1997.

[17] Abraham Susswein and Jack Byrne. Identification and characterization of neurons initiating patterned neural activity in the buccal ganglia of aplysia. *Journal of Neuroscience*, 8:2049–2061, 1988.

[18] Arnaud Tonnelier and Wulfram Gerstner. Piecewise linear differential equations and integrate-and-fire neurons: Insights from two-dimensional membrane models. *Physical Review*, 67(021908):1–16, 2003.

[19] I. Ziv, D.A. Baxter, and J.H. Byrne. Simulator for neural networks and action potentials:description and application. *J. Neurophysiology*, 71:294–308, 1994.