# SANDIA REPORT

2012-10628
Unlimited Release
Printed December, 2012

# Simulating neural systems with Xyce

Richard L. Schiek, Christy E. Warrender, Corinne Teeter,
James Bradley Aimone, Heidi Thornquist, Ting Mei, Alex Duda

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation,
a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's
National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

2012-10628
Unlimited Release
Printed December, 2012

# Simulating neural systems with Xyce

Richard L. Schiek, Heidi Thornquist, Ting Mei,
Electrical Systems Modeling,

Christy E. Warrender, James Bradley Aimone,
Cognitive Modeling,

Corinne Teeter,
Cognitive Systems,

Alex M. Duda, Department of Electrical and Computer Engineering
University of Illinois, Urbana-Champaign

**Abstract**

Sandia's parallel circuit simulator, **Xyce**, can address large scale neuron simulations in a new way extending the range within which one can perform high-fidelity, multi-compartment neuron simulations. This report documents the implementation of neuron devices in **Xyce**, their use in simulation and analysis of neuron systems.

# Acknowledgment

I would like to thank the Laboratory Directed Research Foundation at Sandia National Laboratories for supporting this work under Proposal 12-1058.

# Contents

**Summary** **11**

**1 Introduction** **13**

**2 Formulation** **15**

Hodgkin-Huxley Model Equations . . . . . 15

Connor-Stevens Model Equations . . . . . 17

Cable Model Equations . . . . . 19

Implementation in **Xyce** . . . . . 19

Synapse Modeling . . . . . 22

Basic Synapse Model without plasticity . . . . . 22

Spike-Timing Dependent Plasticity . . . . . 23

Stochastic Transmission Reliability . . . . . 25

w updates . . . . . 26

nonzero $I_{\texttt{post}}$ . . . . . 28

Example Netlists . . . . . 29

Basic Synapse Model without plasticity . . . . . 29

Clopath-Gerstner Plasticity Modulator . . . . . 30

Transmission Probability . . . . . 32

**3 Network Simulations** **35**

Model Description . . . . . 35

Neuron properties . . . . . 35

Synapse properties . . . . . 36
Network connectivity and external input . . . . . 37
Simulation run times . . . . . 37
Simulation Accuracy . . . . . 39
Scaling . . . . . 40

**4 Analysis of large scale neural network simulations 43**

Introduction . . . . . 43
Theoretical Approach . . . . . 44
Step 1: Pre-processing: temporal filtering of raw spiking data . . . . . 44
Step 2: Principle Component Analysis (PCA) . . . . . 45
Step 3: Post-processing – projecting spiking data onto principal components . 46
Results . . . . . 46
Different simulation runs share principal components . . . . . 46
Projection of raw data into PCs allows clear identification of divergence point 49
Discussion . . . . . 56

**5 Reduced-Order Modeling of Neuron 59**

Reduced-order modeling . . . . . 59
The method . . . . . 60
Example . . . . . 61

**6 Fitting the parameters of neural models to biological data 63**

Introduction . . . . . 63
Biological Data . . . . . 64
Choice of Voltage Train Characteristics . . . . . 64
Fitness Function . . . . . 65
Parameter Fitting Methods . . . . . 66

Basic Strategy of Testing Methods . . . . . 66
Evolutionary Algorithms . . . . . 66
Latin Hypercube Sampling . . . . . 68
Izhikevich Model Neuron . . . . . 71

**References** **73**

# List of Figures


1.1 Hierarchy of neural structure. . . . . 14

2.1 With presynaptic neuron fixed in time, the postsynaptic neuron varied its relative timing from -80[ms] (post-before-pre) to 80[ms] (pre-before-post). 16 spike pairs at a given timing would be stimulated; the synaptic weight difference $\Delta w = w_{16} - w_0$ would be computed (where $w_n$ represents the weight after the $\mathtt{n}^{\mathtt{th}}$ spike). The horizontal axis shows $\Delta t$; the vertical axis shows $\Delta w$. 23

2.2 16 spike pairs were stimulated at a given timing; this plot shows when it is post-before-pre 80[ms] (which accounts for the observed long-term depression of the synaptic weight,w) while the synaptic transmission reliability, $P$, is 100% (which accounts for the spiking of the postsynaptic neuron and the updating of the synaptic weight after each presynaptic neural spike). The plot shows various variables of interest that exhibit how the Clopath-Gerstner STDP model works. . . . . 26

2.3 20 presynaptic spikes were stimulated. The postsynaptic neuron, if it spiked, would do so after the presynaptic neuron as its only source of current was in response to the presynaptic neural spike and delivered via the synapse device (which accounts for the observed long-term potentiation of the synaptic weight, w) while the synaptic transmission reliability, $P$, is 50% (which accounts for the spiking of the postsynaptic neuron and the updating of the synaptic weight in response to about 50% of the presynaptic neural spikes). . . 27

3.1 Xyce Runs: Comparison of time integration methods and tolerances. The x-axis is number of CPU's and the y-axis is run time in seconds. . . . . 38

3.2 Neuron Runs: Comparison of time integration options. The x-axis is number of CPU's and the y-axis is run time in seconds. . . . . 38

3.3 Comparison of spike traces. . . . . 39

3.4 Comparison of spike trances over longer times. . . . . 40

3.5 Overall scaling for a 10,000 neuron, 2 million synapse simulation. . . . . 41

3.6 Breakdown of load, solve and setup time for a 10,000 neuron, 2 million synapse simulation. . . . . 42

4.1 A simple filtering function. . . . . 44

4.2 PC1 - 1,000 ms; 4 runs . . . . . 47

4.3 Normalized Dot Product of PC1 between runs, 1,000ms . . . . . 48

4.4 Normalized Dot Product of PC2 between runs, 1,000ms . . . . . 48

4.5 Normalized Dot Product of PC3 between runs, 1,000ms . . . . . 49

4.6 PC1 - 1,000 ms . . . . . 50

4.7 Normalized Dot Product of PC1 between runs, 10,000ms . . . . . 50

4.8 Normalized Dot Product of PC2 between runs, 10,000ms . . . . . 51

4.9 Normalized Dot Product of PC3 between runs, 10,000ms . . . . . 51

4.10 Average normalized dot product between simulations . . . . . 52

4.11 Average normalized dot product between simulations . . . . . 52

4.12 Projection of the first 500 ms of a single simulation run into the combined PC1 54

4.13 Projection of the first 500 ms of a single simulation run into the combined PC1 54

4.14 Combined projection of single simulation runs of PC1 . . . . . 55

4.15 Combined projection form all Xyce simulations using method=6 for PC1 . . . . 55

4.16 Combined projection form all Xyce simulations using method=7 for PC1 . . . . 56

4.17 How PC dynamics change due to bias. . . . . 57

5.1 Transient comparison: the full model vs. the reduced models . . . . . 62

6.1 Two examples of intracellular voltage recordings with current injection from dentate gyrus hippocampus neurons. Each voltage 'wave' (of which their are 15) corresponds to a current injection. The first wave is the increase in voltage due to a current injection of 10 pA. The subsequent waves correspond to current injections increasing in size of 10 pA. Notice that both of the neurons start firing action potentials when 30 pA are injected. . . . . 64

6.2 Normalized parameter values that yield a fit greater than 0.9 obtained by running 1 million LHS samples. Solid lines connect one set of parameter values. 70

6.3 Normalized parameter values that yield a fit greater than 0.90 obtained by running 1 million LHS samples. Solid lines connect one set of parameter values. 71

# List of Tables

2.1 Neuron Model Level Numbers . . . . . 19

# Summary

Sandia's parallel circuit simulator, **Xyce**, can address large scale neuron simulations in a new way by greatly extending the range within which one can perform high-fidelity, multi-compartment neuron simulations. This report documents the implementation of neuron devices in **Xyce**, their use in simulation and analysis of neuron systems.

# Chapter 1

# Introduction

Neurons and the interconnected networks they form are seen as a fundamental unit of cognitive systems that can sense, decide and react to environmental stimuli. In even the simplest neural systems, there is a great hierarchy of structure required to generate behavior (see figure 1). At the sub-cellular level, genes control the activity ion-channels in the neurons cellular membrane. Ion channel activity influences the membrane potential which intern influences a neurons polarization state and ability to carry a signal. Neurons are connected through synapses allowing them to convey signals from one neuron to another (or groups of neurons). Likewise, very large groups of neurons form tissue types (i.e. striatum and columns) believed to be integral in brain function [31, 13, 15, 7]

Advances in imaging and reconstruction technologies are driving many research projects to map all of the neurons in some small animal brains (see Bohland et. al. [3]). While these projects will produce detailed topology of their targeted subjects, the ability to then simulate in detail the neural-physiology is limited to high-fidelity simulations of a few neurons, or lower-fidelity simulations of thousands of neurons (i.e. multi-compartment neuron models versus behavioral, integrate-and-fire neuron models).

Sandia's parallel circuit simulator, Xyce, can address large scale neuron simulations in a new way by greatly extending the range within which one can perform high-fidelity, multi-compartment neuron simulations. With Xyce, we are creating a neuron-to-cognitive simulator that can model detailed dynamics at the level of ion-channels of millions of neurons. This is new science in that it greatly extends the range within which one can use detailed neuron models for tissue and structural level simulations of neuron tissue (i.e. modeling the top half of figure 1).

At the same time, we are enabling simulations where the uncertainty of modeling parameters can be quantified and inherent error can be bounded to better understand the stability of these dynamic systems. Specifically, how does uncertainty in size, shape, ion-channel density at the lowest level of figure 1 effect long term potentiation in coronal striatal brain slices [2]. This enables researchers to better understand the dynamics of conditioning and learning at a cellular level.

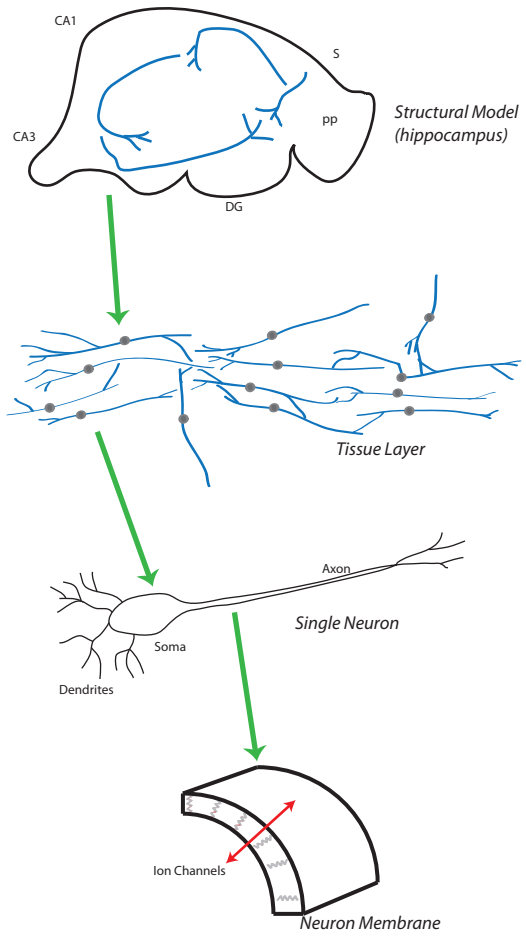


**Figure 1.1.** Hierarchy of neural structure.

# Chapter 2

# Formulation

To simulate the transient behavior of neurons, model equations are used to describe the voltage potential of a neuron's membrane as a function of the neurons state. Within **Xyce** the equations that it solves are in the format of index-one differential algebraic equations (i.e. index-1 DAE's) [19]. The general form of an index-1 DAE is:

$$f(x) + B(t) + \frac{d}{dt}q(x,t) = 0 \tag{2.1}$$

The function $f(x)$ represents that part of the system that depends only on the state of the system (i.e. $x$ where $x$ is the solution vector). The term $B(t)$ is any purely transient elements and generally can be ignored except in cases of external inputs into a system. Finally $q(x,t)$ represents the components whose time derivative contributes to the system.

Given a section of neuron membrane, the voltage potential across the membrane (i.e. from inside to outside) is described by Koch [21] and Dayan [9].

$$C_m \frac{dV}{dt} = i_m + \frac{I_e}{A} \tag{2.2}$$

where $C_m$ is the membrane capacitance, $V$ is the voltage difference, $t$ is time, $i_m$ is the current through the membrane while $I_e$ represents any externally applied current into the cell and $A$ is the surface area of the membrane.

A neuron membrane's voltage potential is a function of the type and distribution of the ion-channels active within the neuron. While there are many different models for ion-channels and membrane dynamics, a good exemplar is the Hodgkin-Huxley formulation.

## Hodgkin-Huxley Model Equations

In the Hodgkin-Huxley model of membrane current, the current is described by:

$$i_m = i_{leak} + i_{Na} + i_K \tag{2.3}$$

where $i_{leak}$ is the current that naturally leaks through the membrane, $i_{Na}$ is the current associated with sodium ion channels and $i_K$ is the current associated with potassium ion channels. This descriptive equation can be refined with algebraic expressions for the individual currents as:

$$i_m = \bar{g}_L (V - E_L) + \bar{g}_{Na} m^3 h (V - E_{Na}) + \bar{g}_K n^4 (V - E_K) \tag{2.4}$$

Parameters in this equation are: $\bar{g}_L$ is the maximal membrane conductance, $E_L$ is the membrane reversing potential, $\bar{g}_{Na}$ is the sodium ion channel conductance, $E_{Na}$ is the sodium channel reversing potential while $\bar{g}_K$ and are the potassium channel maximal conductance and reversing potential respectively. The variables, $m$, $n$ and $h$ are voltage dependent gating variables that model the relative availability of the sodium and potassium channels. Gating variables, $m$, $n$ and $h$ are described by the ordinary differential equations:

$$\frac{dm}{dt} = \alpha_m(V) \cdot (1 - m) - \beta_m(V) \cdot m \tag{2.5}$$

$$\frac{dm}{dt} = \alpha_h(V) \cdot (1 - h) - \beta_h(V) \cdot h \tag{2.6}$$

$$\frac{dm}{dt} = \alpha_n(V) \cdot (1 - n) - \beta_n(V) \cdot n \tag{2.7}$$

The voltage dependent coefficients, $\alpha(V)$ and $\beta(V)$ are described by:

$$\alpha_m(V) = \frac{0.1\,(V + 40)}{1 - e^{-0.1(V+40)}} \tag{2.8}$$

$$\beta_m(V) = 4e^{-0.0556(V+65)} \tag{2.9}$$

$$\alpha_h(V) = 0.07e^{-0.05(V+65)} \tag{2.10}$$

$$\beta_h(V) = \frac{1}{1 + e^{-0.1(V+35)}} \tag{2.11}$$

$$\alpha_n(V) = \frac{0.1\,(V + 55)}{1 - e^{-0.1(V+55)}} \tag{2.12}$$

$$\beta_n(V) = 4e^{-0.0125(V+65)} \tag{2.13}$$

Note, in the equations for $m$, $h$ and $n$, the voltage is given in units of milli-volts and time in milli-seconds.

# Connor-Stevens Model Equations

To more accurately simulate the diverse set of currents that work in typical neurons, the Connor-Stevens model adds two additional current terms to the membrane current equation:

$$i_m = \bar{g}_L (V - E_L) + \bar{g}_{Na} m^3 h (V - E_{Na}) + \bar{g}_K n^4 (V - E_K) + \bar{g}_A a^3 b (V - E_A) \\ + \bar{g}_{CaT} M^2 H (V - E_{Ca}) + \bar{g}_{KCa} c^4 (V - E_K) \tag{2.14}$$

where the maximal conductance and reversal potentials, $\bar{g}_A$, $E_A$, $\bar{g}_{CaT}$, $E_{Ca}$, $\bar{g}_{KCa}$ and $E_K$ are for the A-current, transient calcium current and calcium dependent potassium current respectively.

Gating variables for the additional current terms are described by the following.

Gating variable $a$:

$$\frac{da}{dt} = \frac{a_\infty(V) - a}{\tau_a(V)} \tag{2.15}$$

and

$$a_\infty(V) = \left( \frac{0.0761 e^{0.0314(V+94.22)}}{1 + e^{0.0346(V+1.17)}} \right)^{\frac{1}{3}} \tag{2.16}$$

$$\tau_a(V) = 0.3632 + \frac{1.158}{1 + e^{0.0497(V+55.96)}} \tag{2.17}$$

Gating variable $b$:

$$\frac{db}{dt} = \frac{b_\infty(V) - b}{\tau_b(V)} \tag{2.18}$$

and

$$b_\infty(V) = \left( \frac{1}{1 + e^{0.0688(V+53.3)}} \right)^4 \tag{2.19}$$

$$\tau_b(V) = 1.24 + \frac{2.678}{1 + e^{0.0624(V+50)}} \tag{2.20}$$

Gating variable $M$:

$$\frac{dM}{dt} = \frac{M_\infty(V) - a}{\tau_M(V)} \tag{2.21}$$

and

$$M_\infty(V) = \frac{1}{1+e^{\frac{-(V+57)}{6.2}}} \tag{2.22}$$

$$\tau_M(V) = 0.612 + \frac{1}{e^{\frac{-(V+132)}{16.7}} + e^{\frac{(V+16.8)}{18.2}}} \tag{2.23}$$

Gating variable $H$:

$$\frac{dH}{dt} = \frac{H_\infty(V) - a}{\tau_H(V)} \tag{2.24}$$

and

$$M_\infty(V) = \frac{1}{1+e^{\frac{(V+81)}{4}}} \tag{2.25}$$

$$\tau_M(V) = \begin{cases} e^{\frac{V+467}{66.6}} & V < -80\text{mV} \\ 28 + e^{\frac{-(V+22)}{10.5}} & V \geq -80\text{mV} \end{cases} \tag{2.26}$$

Finally, gating variable $c$ is described by:

$$\frac{dc}{dt} = \frac{c_\infty(V) - b}{\tau_c(V)} \tag{2.27}$$

and

$$c_\infty(V) = \left(\frac{[Ca^{2+}]}{[Ca^{2+}] + 3\mu M}\right) \frac{1}{1+e^{\frac{-(V+28.3)}{12.6}}} \tag{2.28}$$

$$\tau_b(V) = 90.3 - \frac{75.1}{1+e^{\frac{-(V+46)}{22.7}}} \tag{2.29}$$

As in the case of the Hodgkin-Huxley models gating variables, the gating variable equations stated above use voltage in milli-volts and time in milli-seconds and calcium ion concentration in micro-moles for unit consistency.

| model level | Equation set |
|---|---|
| 1 | Hodgkin-Huxley on a patch of membrane |
| 2 | Connor-Stevens on a patch of membrane |
| 3 | Hodgkin-Huxley, cable-equation formulation |
| 4 | Connor-Stevens, cable-equation formulation |
| 5 | not used |
| 6 | Cable equation with dynamic membrane models |
| 7 | Izhikevich, point neuron model |

**Table 2.1.** Neuron Model Level Numbers

# Cable Model Equations

To model a section of a neuron process, such as an axon, a cable-equation formulation is used. Here, the cable equation is specified as:

$$C_m \frac{dV_i}{dt} = -i_i^m + \frac{I_i^E}{A_i} + g_{i,i+1}\left(V_{i+1} - V_i\right) + g_{i,i-1}\left(V_{i-1} - V_i\right) \tag{2.30}$$

where $C_m$ is the membrane capacitance, $V_i$ is the voltage in compartment $i$ relative to an external ground, $t$ is time, $i_i^m$ is the current through the membrane in compartment $i$, $I_i^E$ represents any externally applied current into the cell and $A_i$ is the surface area of the membrane in compartment $i$. The final two terms represent current flow into the adjoining compartments, $i-1$, for the previous compartment and $i+1$ for the next compartment. Conductance between the compartments, $g_{i,i+1}$ and $g_{i,i-1}$ can be calculated by:

$$g_{i,j} = \frac{a_i a_j^2}{r_{Long} L_i \left(L_i a_j^2 + L_j a_i^2\right)} \tag{2.31}$$

where $a_i$ is the radius of compartment $i$, $L_i$ is the length of compartment $i$ and $r_{Long}$ is the longitudinal intracellular resistance.

# Implementation in Xyce

Given that the previous equations are all first order, ordinary differential equations, it is straightforward to cast them in DAE format. As DAE's the equations were implemented in Xyce as neuron devices with the following device-levels corresponding to specific equations sets.

Listed in table 2.1 are the model numbers used to connect specific neuron models in **Xyce** to implemented equations.

For example, the following netlist models a patch of neuron using the Hodgkin-Huxley equations. Details on the syntax of the netlist can be found in the **Xyce** Users' Guide [20]

```
A single Neuron example


* This is a standard current pulse to start an activation

.tran 0 2.0e-2

.print tran V(a) i(iin) v(b) n(y%neuron%neuron1_m) n(y%neuron%neuron1_h)
+  n(y%neuron%neuron1_n)

Iin 0 a PULSE( 0 0.40e-9 1.0e-3 1.0e-4 1.0e-4 1.0e-3 1.0e10)

* standard area is 30x30xpi um2 = 30e-6 * 30e-6 * pi m^2
*                               = 3e-5 * 3e-5 * pi m^2 = 3e-3 * 3e-3 * pi cm^2
* scale GK, Gna, manebrane capacitance, membrane conductivity
.param area = {3.0e-3 * 3.0e-3 * 3.141529}  ; [cm^2]
.param gks  = {0.036 * area }               ; [0.036 S/cm^2  * [area cm^2] ]
.param gnas = {0.120 * area }               ; [0.120 S/cm^2  * [area cm^2] ]
.param memC = {1.0e-6 * area }              ; [1.0uF/cm^2    * [area cm^2] ]
.param memG = {0.0003 * area }              ; [0.0003 S/cm^2 * [area cm^2] ]

*
* Using the above neuron model
*
.model hhParams neuron level=1 cMem={memC}  gMem={memG}
+ vRest=0.010613  eNa=0.115 gNa={gnas}  eK=-0.012  gK={gks}

yneuron neuron1  a b hhParams

* this is here to provide a non-zero potential
* outside of the membrane for this demo.  Typically
* one doesn't need it.
rloader b 0 100
.end
```

and here is an example using the Hodgkin-Huxley equations in a cable-equation representation of a neuron.

```
A neuron cable example
*
* This is a simple test of simulating a neuron cable
* via the cable equation (6.29 in Theorteical Neuroscience,
* by P. Dayan and L. Abbot)
*

* units for model parameters
* cMem = F/cm^2
* gMem = S/cm^2
* vRest = V
* eNa = V
* gNa = S/cm^2
* eK = V
* gK = S/cm^2
* r  = ohms cm
*
* NOTE: length scale is immaterial as long as it's always a consistent unit
* parameters from Dayan's book pg 173, 157, 155
*
* gMem  = 0.003 mS/mm^2 =>  0.000003 S/mm^2  => 0.0003 S/cm^2
* gK    = 0.36  mS/mm^2 =>  0.00036 S/mm^2   => 0.036  S/cm^2
* gNa   = 1.2 mS/mm^2   =>  0.0012 S/mm^2    => 0.12   S/cm^2
* vRest = -54.387 mV    => -0.054387 V
* eK    = -77 mV        => -0.077 V
* eNa   =  50 mV        =>  0.050 V
* cMem  = 10 nF/mm^2    =>  1.0e-8 F/mm^2    => 1.0e-6 F/cm^2
* r     = 1 kOhm mm     =>  1000 Ohm mm      => 100 Ohm cm

*
* Using the above neuron model
.model hhParams neuron level=3 cMem=1.0e-6 gMem=0.0003 vRest=-0.054387
+ eNa=0.050 gNa=0.12  eK=-0.077  gK=0.036

* This is a standard current pulse to start an activation
*  pulse( initial_value pulse_value delay_time rise_time fall_time pulse_width period)
Iin a 0 PULSE( 0 0.40e-7 1.0e-3 1.0e-6 1.0e-6 1.0e-3 1.0e10)

* the parameters R (intra-cellular resistivity Ohm/cm), A= radius (cm), L = length (cm)
* can be specified in the .model statement or as part of the instance.  Instance level
* parameters override model level ones.  N = number of segments.

* intra cellular resistivity, rl,  is typically 1-3 kOhm mm.
*  Resistance along the long axis (longitudinal resistance Rl = rl L / (pi a^2)
```

```
* 1 kOhm mm = 1000 Ohm mm = 100 Ohm cm

yneuron neuron1  a b hhParams R=1.0e1 A=1.0e-4 L=0.4 N=100
+ RPS=1.0e2 APS=1.0e-4 LPS=4.0e-3 RNS=1.0e2 ANS=1.0e-4 LNS=4.0e-3



.tran 0 2.0e-2
.options timeint method=7 newlte=1 newbpstepping=1 reltol=1e-3

.print tran i(iin) V(a) n(y%neuron%neuron1_V1) n(y%neuron%neuron1_n1)
+  n(y%neuron%neuron1_m1) n(y%neuron%neuron1_h1) n(y%neuron%neuron1_V10)
+  n(y%neuron%neuron1_V50) n(y%neuron%neuron1_V90)

.end
```

# Synapse Modeling

For state-of-the-art research into spiking neuron population dynamics and learning, it is critical to have a synapse device that captures some of the primary experimentally observed features. Thus, we implemented a synapse device with conductance-based dynamics, spike-timing dependent plasticity (with long-term potentiation (LTP) and long-term depression (LTD)), and a stochastic synaptic transmission reliability modulator. The details of each are discussed in the sections that follow.

# Basic Synapse Model without plasticity

The model used is based on NEURON simulators Exp2Syn mechanism [24, 16]. With $w$ representing the Clopath-Gerstner plasticity scheme outlined in the next section (set to 1 in the case where no learning occurs), $B.V$ the momentary postsynaptic voltage, and $E_{\texttt{rev}}$ the reversal potential (set to $-85 \times 10^{-3}\,[V]$), the postsynaptic current is the following:

$$I_{\texttt{post}} \;=\; w g_{\texttt{MAX}}(B.V - E_{\texttt{rev}})$$

where $g_{\texttt{MAX}}$, the maximal conductance, is defined as follows:

$$g_{\texttt{MAX}} \;=\; f_{\texttt{norm}}\left(\exp\left(\frac{-t}{\tau_{\texttt{decay}}}\right) - \exp\left(\frac{-t}{\tau_{\texttt{rise}}}\right)\right)$$

where $f_{\texttt{norm}}$ is a normalizing factor that ensures the peak is 1, $\tau_{\texttt{rise}}$ is the rise time set at $2 \times 10^{-4}\,[s]$, and $\tau_{\texttt{decay}}$ is the decay time set to $1 \times 10^{-2}\,[s]$ (making sure that $\tau_{\texttt{decay}} > \tau_{\texttt{rise}}$). To run a simulation testing the basic synapse model without plasticity, see Netlist 2.

# Spike-Timing Dependent Plasticity

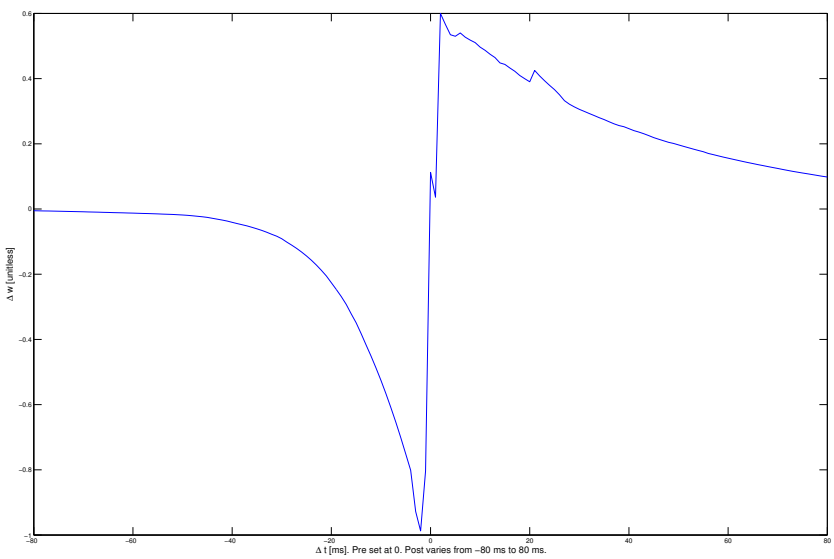


**Figure 2.1.** With presynaptic neuron fixed in time, the postsynaptic neuron varied its relative timing from -80[ms] (post-before-pre) to 80[ms] (pre-before-post). 16 spike pairs at a given timing would be stimulated; the synaptic weight difference $\Delta w = w_{16} - w_0$ would be computed (where $w_n$ represents the weight after the $\texttt{n}^{\texttt{th}}$ spike). The horizontal axis shows $\Delta t$; the vertical axis shows $\Delta w$.

We have adapted the Clopath-Gerstner model [5, 6] to be used in a real-time fashion within a Xyce circuit device that interacts with Hodgkin-Huxley spiking neuron devices (the particular model is *the standard Hodgkin-Huxley membrane patch model*[21]). This well-known phenomenological model captures a number of the important experimentally observed behaviors of plasticity in synapses. Additionally, it is easily tunable to exhibit a variety of STDP curves. The curve we want to generate is shown in Figure 2.1. With the following variables

$$S = \texttt{voltagethresholdforaspikeevent}$$

$$
\begin{aligned}
R &= \texttt{voltagevalueforarestingevent} \\
w &= \texttt{weight/strengthofsynapse} \\
A.V &= \texttt{momentarypresynapticmembranevoltage} \\
V_{L3} &= \texttt{aLPFversionofA.Vwithrate}\tau_3 \\
B.V &= \texttt{momentarypostsynapticmembranevoltage} \\
V_{L1} &= \texttt{aLPFversionofB.Vwithrate}\tau_1 \\
V_{L2} &= \texttt{aLPFversionofB.Vwithrate}\tau_2
\end{aligned}
$$

and the Boolean operator on variables $x_1$ and $x_2$ defined as follows:

$$
\begin{aligned}
x_1 > x_2 &= 1 \\
x_1 < x_2 &= 0
\end{aligned}
$$

the modified Clopath/Gerstner equation that updates the synaptic weight is as follows:

$$
\frac{dw}{dt} = \left( \frac{dw_{\texttt{LTD}}}{dt} + \frac{dw_{\texttt{LTP}}}{dt} \right) (w > w_{\texttt{min}})(w < w_{\texttt{max}})
$$

where the changes in $w$ due to LTD and LTP are:

$$
\begin{aligned}
\frac{dw_{\texttt{LTD}}}{dt} &= -A_{\texttt{LTD}}(A.V > S)(V_{\texttt{L1}} > R)(V_{L1} - R) \\
\frac{dw_{\texttt{LTP}}}{dt} &= A_{\texttt{LTP}} V_{\texttt{L3}}(B.V > S)(B.V - S)(V_{\texttt{L2}} > R)(V_{L2} - R)
\end{aligned}
$$

while the changes in the LPF voltages are:

$$
\begin{aligned}
\frac{dV_{\texttt{L1}}}{dt} &= \frac{B.V - V_{\texttt{L1}}}{\tau_1} \\
\frac{dV_{\texttt{L2}}}{dt} &= \frac{B.V - V_{\texttt{L2}}}{\tau_2} \\
\frac{dV_{\texttt{L3}}}{dt} &= \frac{(A.V > S) - V_{\texttt{L3}}}{\tau_3}
\end{aligned}
$$

The parameters are set as follows (note that some parameter values are different from the Clopath-Gerstner papers; this was necessary to obtain the desired behavior):

$$
\begin{aligned}
S &= -45.3 \times 10^{-3}\,[V] \\
R &= -72.655 \times 10^{-3}\,[V] \\
w_{\texttt{min}} &= 0.0 \\
w_{\texttt{max}} &= 1.6 \\
w &= 1(\texttt{initialvalue}) \\
A_{\texttt{LTD}} &= 5 \times 10^{-2}\left[V^{-1}\right] \\
A_{\texttt{LTP}} &= 8.5\left[V^{-2}\right] \\
\tau_1 &= 23 \times 10^{-3}\,[s] \\
\tau_2 &= 7 \times 10^{-3}\,[s] \\
\tau_3 &= 46 \times 10^{-3}\,[s]
\end{aligned}
$$

For an example of the pair of spiking neurons, as well as the different LPF voltages, and the synaptic dynamics over time, see Figure 2.2. Additionally, to run the simulation, see Netlist 2.

## Stochastic Transmission Reliability

In experimental studies, action potentials generated in presynaptic neuron, only released neurotransmitter to postsynaptic neurons about 10% of the time [36]. This value can vary depending on the species of neurons, synapses, etc. However, it is evident that being able to adjust the synaptic transmission reliability is imperative. In fact, we conjecture that stochastic transmission failure at the single synapse level plays a critical role in enabling the generation of population-level attractor dynamics that could serve as the basis for a multi-modal associative memory [11, 12].

Therefore, having such functionality in a Xyce device is important. Specifically, this behavior functions as follows (when a given presynaptic neural spike event occurs):

- with probability $P$, the synapse will work as usual
- with probability $(1 - P)$, $w$ will not be updated and no synaptic current will be generated

In order to confirm functionality, two experiments were conducted to test that there were the expected number of: (1) synaptic weight updates $N\,(\Delta w)$ and (2) nonzero postsynaptic

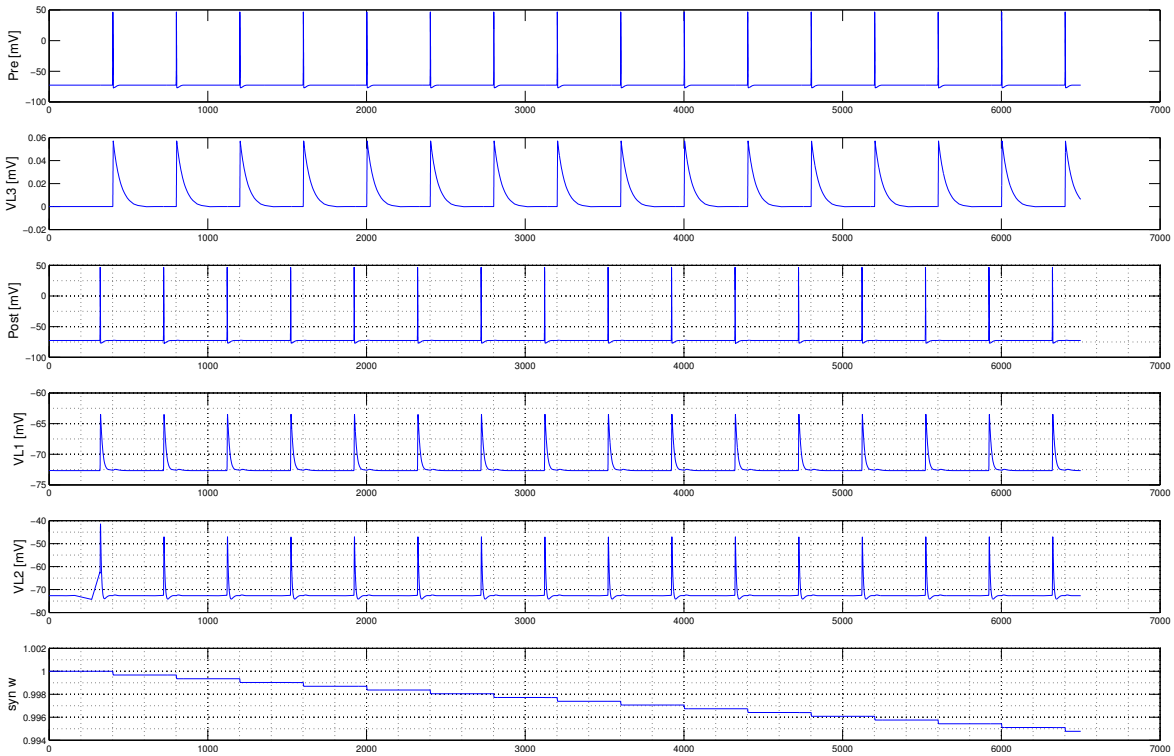


**Figure 2.2.** 16 spike pairs were stimulated at a given timing; this plot shows when it is post-before-pre 80[ms] (which accounts for the observed long-term depression of the synaptic weight,w) while the synaptic transmission reliability, $P$, is 100% (which accounts for the spiking of the postsynaptic neuron and the updating of the synaptic weight after each presynaptic neural spike). The plot shows various variables of interest that exhibit how the Clopath-Gerstner STDP model works.

currents, $N\left(I_{\texttt{post}} \neq 0\right)$. In both experiments, $P$ was fixed at a given value and the simulation was run 100 times; this was carried out for $P \in \{0, 0.1, 0.2, \ldots, 1\}$. For the purposes of the experiments, it sufficed to fix the timing between presynaptic and postsynaptic neurons. In particular, it was set to $10\,[\texttt{ms}]$ difference, pre-before-post. The results of the experiments are summarized below.

## w updates

For each $P$ value, the actual average $\mathbb{E}\left[N\left(\Delta w\right)\right]$ was computed and compared to the theoretically expected number $\hat{\mathbb{E}}\left[N\left(\Delta w\right)\right]$. The results are as follows:

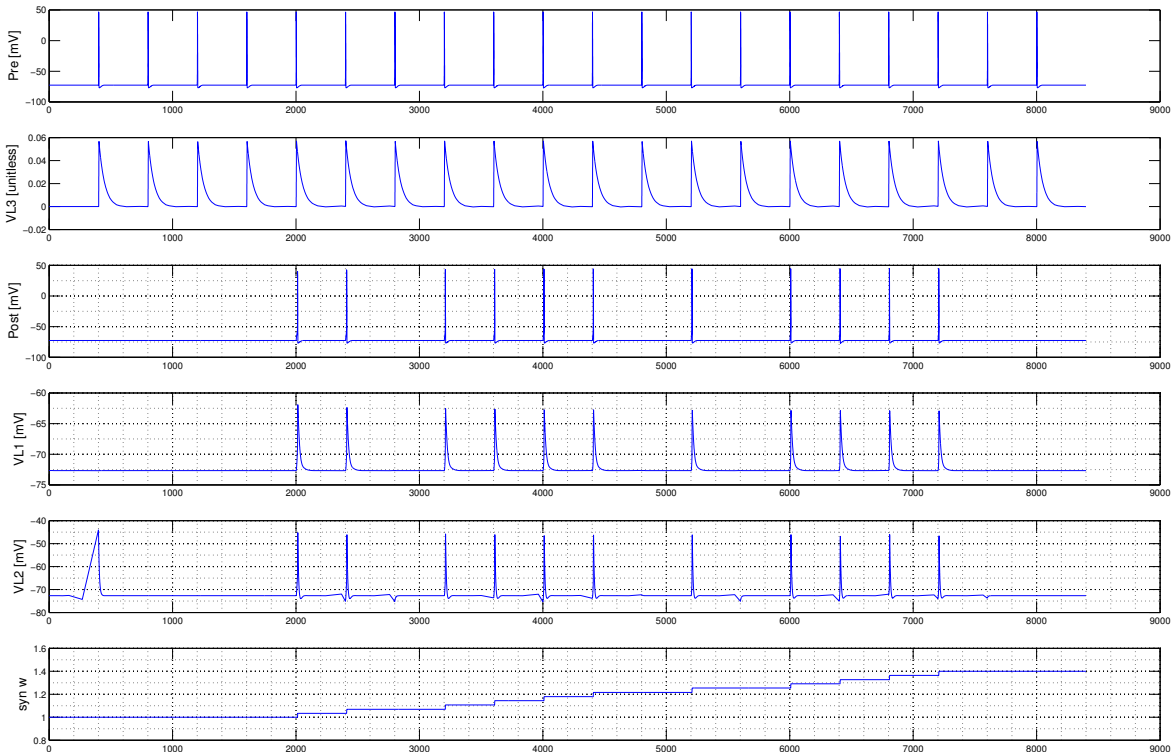


**Figure 2.3.** 20 presynaptic spikes were stimulated. The postsynaptic neuron, if it spiked, would do so after the presynaptic neuron as its only source of current was in response to the presynaptic neural spike and delivered via the synapse device (which accounts for the observed long-term potentiation of the synaptic weight, w) while the synaptic transmission reliability, $P$, is 50% (which accounts for the spiking of the postsynaptic neuron and the updating of the synaptic weight in response to about 50% of the presynaptic neural spikes).

| P | $\hat{\mathbb{E}}\left[N\left(\Delta w\right)\right]$ | $\mathbb{E}\left[N\left(\Delta w\right)\right]$ |
|---|---|---|
| 0% | 0 | 0 |
| 10% | 2 | 1.8416 |
| 20% | 4 | 4.0594 |
| 30% | 6 | 6.2178 |
| 40% | 8 | 8.1584 |
| 50% | 10 | 10.1188 |
| 60% | 12 | 11.9901 |
| 70% | 14 | 14.0594 |
| 80% | 16 | 16.2574 |
| 90% | 18 | 18.2475 |
| 100% | 20 | 20 |

The actual agree well with the expected, confirming the proper function.

### nonzero $I_{\texttt{post}}$

For each $P$ value, the actual average $\mathbb{E}\left[N\left(I_{\texttt{post}} \neq 0\right)\right]$ was computed and compared to the theoretically expected number $\hat{\mathbb{E}}\left[N\left(I_{\texttt{post}} \neq 0\right)\right]$. The results are as follows:

| P | $\hat{\mathbb{E}}\left[N\left(I_{\texttt{post}} \neq 0\right)\right]$ | $\mathbb{E}\left[N\left(I_{\texttt{post}} \neq 0\right)\right]$ |
|---|---|---|
| 0% | 0 | 0 |
| 10% | 2 | 1.8416 |
| 20% | 4 | 4.0594 |
| 30% | 6 | 6.2178 |
| 40% | 8 | 8.1584 |
| 50% | 10 | 10.1188 |
| 60% | 12 | 11.9901 |
| 70% | 14 | 14.0594 |
| 80% | 16 | 16.2574 |
| 90% | 18 | 18.2475 |
| 100% | 20 | 20 |

The actual agree well with the theoretical, confirming the proper function. The astute observer will notice that the actual values are the same in the second experiment as in the first experiment; clearly, this is what one would expect. For an example of the pair of spiking neurons, as well as the different LPF voltages, and the synaptic dynamics over time, when $P = 0.50$ for a single simulation, see Figure 2.3. Additionally, to run the simulation, see Netlist 2.

The device appears to work as desired. It will be used in the generation of a model intended to investigate the extent to which the dynamics of a population of spiking neurons can be used as the basis for an associative memory. Of particular interest is the role that synaptic transmission failure plays in the generation of stable states far-from-equilibrium, as they appear to evolve in actual neocortical regions. We conjecture that such transmission failure is critical to encoding multi-sensory memories that are robust to noise and exhibit a high degree of associativity. In fact, we posit that dynamical distributed information encoding within a population of spiking neurons depends on modulating the synaptic transmission reliability. A number of large-scale real-time experiments will be designed to investigate these theories.

# Example Netlists

## Basic Synapse Model without plasticity

```
Test Synapse Level 3 device by Alex Duda
*No learning, perfectly reliable.
*13 August 2012

.options timeint method=7 newlte=1 newbpstepping=1 reltol=1e-4

.GLOBAL_PARAM TIMING=410e-3
.param AMP={3.14159*.1825e-12}
.param WIDTH={1e-3}
.param PERIOD = {400e-3}

*FIRST WE NEED TO SEND A CURRENT INPUT to HH1 and see its effect on HH2.

In11 0 a1 PULSE( 0 {AMP} {400e-3} 1.0e-6 1.0e-6 {WIDTH} {PERIOD} )

*SECOND WE NEED TO ADD HH1 TO RECEIVE INPUT.

.param segLength = 1e-4     ; [cm]
.param segDiameter = 1e-4   ; [cm]
.param segSurfaceArea = { 3.14159 * segDiameter * segLength }

* specific membrane capacitance 1uF/cm^2
.param memC = { 1.0e-6 * segSurfaceArea } ; [F]

* leak current has membrane resistivity of 40,000 ohm cm^2,
* with reversal potential of -65mV
.param rm = { 4.0e4 / segSurfaceArea }    ; [ohm]
.param memG = { 1 / rm }                  ; [1/ohm]
.param revE = -0.065                      ; [V]

* active conductances
* Na specific conductance is 1200 S/m^2 = 1.2e-1 S/cm^2
.param gnas = { 0.12 * segSurfaceArea }   ; [S]
.param ErevNa = 0.05                      ; [V]
* K specific conductance is 360 S/m^2 = 3.6e-2 S/cm^2
.param gks  = { 0.036 * segSurfaceArea }  ; [S]
.param ErevK = -0.077                     ; [V]

* neuron model
.model HH_Params neuron level=1 cMem={memC}  gMem={memG}
```

```
+ eLeak={revE}  gNa={gnas}  gK={gks}
+ eNa={ErevNa} eK={ErevK} vRest={revE}

*CREATE THREE NEURON INSTANCES

yneuron HH1 a1 0 HH_Params
yneuron HH2 a2 0 HH_Params

.ic v(a1)=-72.655e-3
.ic v(a2)=-72.655e-3

*THIRD WE NEED TO ADD A SYNAPSE TO GO BETWEEN HH1 AND HH2.

*(express all params in [A], [V], [s], etc.)
*Tune maximal conductance, gMax, properly!
*Let gRheo be roughly the least amount of conductance
*that allows a single presyn neuronal spike to cause a postsyn neuronal spike.
.param gRheo=1.318e-12

*Tune N_Neu parameter such that it takes the desired number of presynaptic
*spiking neurons to make a postsynaptic neuron spike.
.param N_Neu=1

.model synParams synapse level=3 vThresh={-45.3e-3} delay={1e-4}
+ gMax={gRheo/N_Neu} eRev={0} tau1={1e-4} tau2={5e-3}
+ ALTD={5e-2} ALTP={8.5} L1TAU=23e-3 L2TAU=7e-3 L3TAU= 46e-3
+ R=-72.655e-3 S=-45.3e-3 WINIT=1 WMAX=1 WMIN=1

*The P parameter represents the synapse success probability.
*With probability P it will work as usual,
*With probability (1-P) it will fail to generate a synaptic current
*and the w will fail to update.

ysynapse syn12 a1 a2 synParams P={1}

.tran 0 8.4
.print tran i(In11) v(a1) v(a2) n(y%synapse%syn12_w) n(y%synapse%syn12_vl1)
+ n(y%synapse%syn12_vl2) n(y%synapse%syn12_vl3)
.end
```

## Clopath-Gerstner Plasticity Modulator

```
Test Synapse Level 3 device by Alex Duda
```

```
*Learning turned on but perfectly reliable.
*Confirming STDP learning curve.
*13 August 2012

.options timeint method=7 newlte=1 newbpstepping=1 reltol=1e-4

.GLOBAL_PARAM TIMING=320e-3
.STEP TIMING 320e-3 480e-3 1e-3

.param AMP={3.14159*.1825e-12}
.param WIDTH={1e-3}
.param PERIOD = {400e-3}


*FIRST WE NEED TO SEND A CURRENT INPUT TO HH1 AND HH2.

In11 0 a1 PULSE( 0 {AMP} {400e-3} 1.0e-6 1.0e-6 {WIDTH} {PERIOD} )
In22 0 a2 PULSE( 0 {AMP} {TIMING} 1.0e-6 1.0e-6 {WIDTH} {PERIOD} )

*SECOND WE NEED TO ADD HHs TO RECEIVE INPUT.

.param segLength = 1e-4     ; [cm]
.param segDiameter = 1e-4   ; [cm]
.param segSurfaceArea = { 3.14159 * segDiameter * segLength }

* specific membrane capacitance 1uF/cm^2
.param memC = { 1.0e-6 * segSurfaceArea } ; [F]

* leak current has membrane resistivity of 40,000 ohm cm^2,
*  with reversal potential of -65mV
.param rm = { 4.0e4 / segSurfaceArea }    ; [ohm]
.param memG = { 1 / rm }                  ; [1/ohm]
.param revE = -0.065                      ; [V]

* active conductances
* Na specific conductance is 1200 S/m^2 = 1.2e-1 S/cm^2
.param gnas = { 0.12 * segSurfaceArea }   ; [S]
.param ErevNa = 0.05                      ; [V]
* K specific conductance is 360 S/m^2 = 3.6e-2 S/cm^2
.param gks  = { 0.036 * segSurfaceArea }  ; [S]
.param ErevK = -0.077                     ; [V]

* neuron model
.model HH_Params neuron level=1 cMem={memC}  gMem={memG} eLeak={revE}
+ gNa={gnas}  gK={gks} eNa={ErevNa} eK={ErevK} vRest={revE}
```

```
*CREATE TWO NEURON INSTANCES

yneuron HH1 a1 0 HH_Params
yneuron HH2 a2 0 HH_Params

.ic v(a1)=-72.655e-3
.ic v(a2)=-72.655e-3

*THIRD WE NEED TO ADD A SYNAPSE TO GO BETWEEN HH1 AND HH2.
*(express all params in [A], [V], [s], etc.)
*Tune maximal conductance, gMax, properly!
*Let gRheo be roughly the least amount of conductance
*that allows a single presyn neuronal spike to cause a postsyn neuronal spike.
.param gRheo=1.318e-12

*Tune N_Neu parameter such that it takes the desired number of presynaptic
*spiking neurons to make a postsynaptic neuron spike.
.param N_Neu=20

*In order to have a smooth curve, we need to decrease ALTD, ALTP.
.model synParams synapse level=3 vThresh={-45.3e-3} delay={1e-4}
+ gMax={gRheo/N_Neu} eRev={0} tau1={1e-4} tau2={5e-3}
+ ALTD={5e-2} ALTP={8.5} L1TAU=23e-3 L2TAU=7e-3 L3TAU= 46e-3
+ R=-72.655e-3 S=-45.3e-3 WINIT=1 WMAX=1.6 WMIN=0

*The P parameter represents the synapse success probability.
*With probability P it will work as usual.
*With probability (1-P) it will fail to generate a synaptic current and
*the w will fail to update.

ysynapse syn12 a1 a2 synParams P={1}

*.tran 0 31.6
.tran 0 6.5
.print tran i(In11) v(a1) i(In22) v(a2) n(y%synapse%syn12_w) n(y%synapse%syn12_vl1)
+ n(y\%synapse\%syn12_vl2) n(y\%synapse\%syn12_vl3)
.end
```

## Transmission Probability

```
Test Synapse Level 3 device by Alex Duda
*(adjusted parameters and plasticity with access to internal states/variables)
```

```
*Configured so one spiking presynaptic neuron will make one postsynaptic neuron spike
*ALTP and ALTD tuned for somewhat smooth stdp curve
*13 August 2012

.options timeint method=7 newlte=1 newbpstepping=1 reltol=1e-4

.GLOBAL_PARAM TIMING=410e-3
*.STEP TIMING 390e-3 410e-3 1e-3

.GLOBAL_PARAM TEST=0
.STEP TEST 0 1 0.01

.param S=0.50
.param AMP={3.14159*.1825e-12}
.param WIDTH={1e-3}
.param PERIOD = {400e-3}


*FIRST WE NEED TO SEND A CURRENT PAIR of INPUTS to HH1 and HH2.

In11 0 a1 PULSE( 0 {AMP} {400e-3} 1.0e-6 1.0e-6 {WIDTH} {PERIOD} )

*SECOND WE NEED TO ADD HH1 TO RECEIVE INPUT.

.param segLength = 1e-4     ; [cm]
.param segDiameter = 1e-4   ; [cm]
.param segSurfaceArea = { 3.14159 * segDiameter * segLength }

* specific membrane capacitance 1uF/cm^2
.param memC = { 1.0e-6 * segSurfaceArea } ; [F]

* leak current has membrane resistivity of 40,000 ohm cm^2,
* with reversal potential of -65mV
.param rm = { 4.0e4 / segSurfaceArea }    ; [ohm]
.param memG = { 1 / rm }                  ; [1/ohm]
.param revE = -0.065                      ; [V]

* active conductances
* Na specific conductance is 1200 S/m^2 = 1.2e-1 S/cm^2
.param gnas = { 0.12 * segSurfaceArea }   ; [S]
.param ErevNa = 0.05                      ; [V]
* K specific conductance is 360 S/m^2 = 3.6e-2 S/cm^2
.param gks  = { 0.036 * segSurfaceArea }  ; [S]
.param ErevK = -0.077                     ; [V]
```

```
* neuron model
.model HH_Params neuron level=1 cMem={memC}  gMem={memG}
+ eLeak={revE}  gNa={gnas}  gK={gks}
+ eNa={ErevNa} eK={ErevK} vRest={revE}

*CREATE THREE NEURON INSTANCES

yneuron HH1 a1 0 HH_Params
yneuron HH2 a2 0 HH_Params

.ic v(a1)=-72.655e-3
.ic v(a2)=-72.655e-3

*THIRD WE NEED TO ADD A SYNAPSE TO GO BETWEEN HH1 AND HH2.
*(express all params in [A], [V], [s], etc.)
*Tune maximal conductance, gMax, properly!
*Let gRheo be roughly the least amount of conductance
*that allows a single presyn neuronal spike to cause a postsyn neuronal spike.
.param gRheo=1.318e-12

*Tune N_Neu parameter such that it takes the desired number of presynaptic
*spiking neurons to make a postsynaptic neuron spike.
.param N_Neu=1

.model synParams synapse level=3 vThresh={-45.3e-3} delay={1e-4}
+ gMax={gRheo/N_Neu} eRev={0} tau1={1e-4} tau2={5e-3}
+ ALTD={5e-2} ALTP={8.5} L1TAU=23e-3 L2TAU=7e-3 L3TAU= 46e-3
+ R=-72.655e-3 S=-45.3e-3 WINIT=1 WMAX=1.6 WMIN=0

*The P parameter represents the synapse success probability.
*With probability P it will work as usual.
*With probability (1-P) it will fail to generate a synaptic current and
*the w will fail to update.

ysynapse syn12 a1 a2 synParams P={S}

*.tran 0 31.6
.tran 0 8.4
.print tran i(In11) v(a1) v(a2) n(y%synapse%syn12_w) n(y%synapse%syn12_vl1)
+ n(y%synapse%syn12_vl2) n(y%synapse%syn12_vl3)
.end
```

# Chapter 3

# Network Simulations

## Model Description

Our network simulations were based on a benchmark published in [27], specifically, benchmark 2, which uses a variant of Hodgkin-Huxley neurons. We varied network sizes, but in all cases the network consisted of 80% excitatory neurons and 20% inhibitory neurons connected randomly with a probability of 2%.

A primary characteristic of these networks is that when random external spike inputs are provided to 2% of the excitatory neurons for the first 100 ms of the simulation, spiking activity continues to propagate through the network after the external stimulation stops.

## Neuron properties

The model neurons in these simulations were based on a neuron model originally published in [34]. This form is somewhat different than the form we used in the level 1 Xyce neuron device, described in chapter 2. The differences are important, as they are necessary to get the sustained activity mentioned above. So we implemented these equations in the level 9 neuron device. We tested the response of this device to current injection as in the Neuron implementation's test of intrinsic properties and verified that we got the same results.

The form used by the Brette benchmark is:

$$C_m \frac{dV}{dt} = -g_L(V - E_L) - g_{Na} m^3 h(V - E_{Na}) - g_{Kd} n^4 (V - E_K) + G(t) \tag{3.1}$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \tag{3.2}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h \tag{3.3}$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n \tag{3.4}$$

where $g_{Na} = 100\ mS/cm^2$ and $g_{Kd} = 30\ mS/cm^2$ are the maximal conductances of the sodium current and delayed rectifier with reversal potentials of $E_{Na} = 50\ mV$ and $E_K = -90\ mV$. $m$, $h$, and $n$ are the activation variables which time evolution depends on the voltage-dependent rate constants $\alpha_m$, $\beta_m$, $\alpha_h$, $\beta_h$, $\alpha_n$ and $\beta_n$. $G(t)$ represents synaptic input, described below. The voltage-dependent expressions of the rate constants were modified from the model described by [34]:

$$\alpha_m = 0.32 - \frac{(13 - V + VT)}{e^{\frac{13-V+VT}{4}} - 1} \tag{3.5}$$

$$\beta_m = 0.28 - \frac{(V - VT - 40)}{e^{\frac{V-VT-40}{5}} - 1} \tag{3.6}$$

$$\alpha_h = 0.128 - e^{\frac{17-V+VT}{18}} \tag{3.7}$$

$$\beta_h = \frac{4}{1 + e^{\frac{40-V+VT}{5}}} \tag{3.8}$$

$$\alpha_n = 0.032 - \frac{(15 - V + VT)}{e^{\frac{15-V+VT}{5}} - 1} \tag{3.9}$$

$$\beta_n = 0.5 - e^{\frac{10-V+VT}{40}} \tag{3.10}$$

where VT = -63 mV adjusts the threshold (which was around -50 mV for the above parameters).

# Synapse properties

A presynaptic spike generates a current in the postsynaptic neuron after a delay D. The conductance $G$ is of the form

$$G(t) = g(t)(V - E) \tag{3.11}$$

where V is the postsynaptic voltage, E is the reversal potential: 0 mV for excitatory synapses and -80 mV for inhibitory synapses. The benchmark description calls for an instantaneous

change in synaptic conductance $g$ at the start of the postsynaptic response, followed by an exponential decay. The Xyce synapse device (level 4, or level 3 with plasticity disabled) uses a double exponential implementation of synaptic dynamics rather than the discontinuous change, so we used a very small time constant, 1e-7 s, for the rise time to approximate the instantaneous change. Maximum conductance was 6 nS for excitatory synapses and 67 nS for inhibitory synapses; decay time constant was 5 ms for excitatory synapses and 10 ms for inhibitory synapses. The original benchmark actually specified no delay D between presynaptic spike and postsynaptic response; we followed the Neuron implementation in introducing a delay of 0.1 ms between neurons in the network and 1 ms after external input spikes.

## Network connectivity and external input

Network connectivity and external input are both random in this model. Since one of our purposes was to compare Xyce performance to that of the Neuron simulator, we wanted to use the same connectivity and external stimuli in both simulators. We ran the simulations initially in Neuron, saving the connectivity and stimulus info, and then used those files to generate the corresponding Xyce netlist.

## Simulation run times

The Xyce simulation tool uses implicit methods to solve the set of DAEs generated as a result of the network models and connectivity. This creates a solver loop that has a time integration method as the outer loop, Newton's method is used as a nonlinear solver, and the inner-most loop is a linear solver. The computational cost of simulation is dominated by either the device update and Jacobian matrix / residual vector load for Newton's method or the solution method for the linearized system (linear solver). For small networks, the device update is the more dominant computational cost, while the linear solver becomes the more dominant as the networks size increases.

The benchmark calls for a network with 4000 neurons. Run times for Xyce and Neuron using a variety of integration and solver options are shown in the figures below. Note that the 4000-neuron network has 16,248 unknowns, which is smaller than the size for which Xyce parallelization is expected to pay off.

For the Xyce runs (c.f. figure 3.1), m7 refers to time integration method 7, variable order Trapezoid; rtol refers to the relative tolerance used for time integration; klu indicates the use of the serial [linear] solver; btf indicates the use of the parallel [iterative linear] solver with these options:

```
.options linsol type=aztecoo tr_partition=0 tr_amd=0 tr_global_btf=2
```

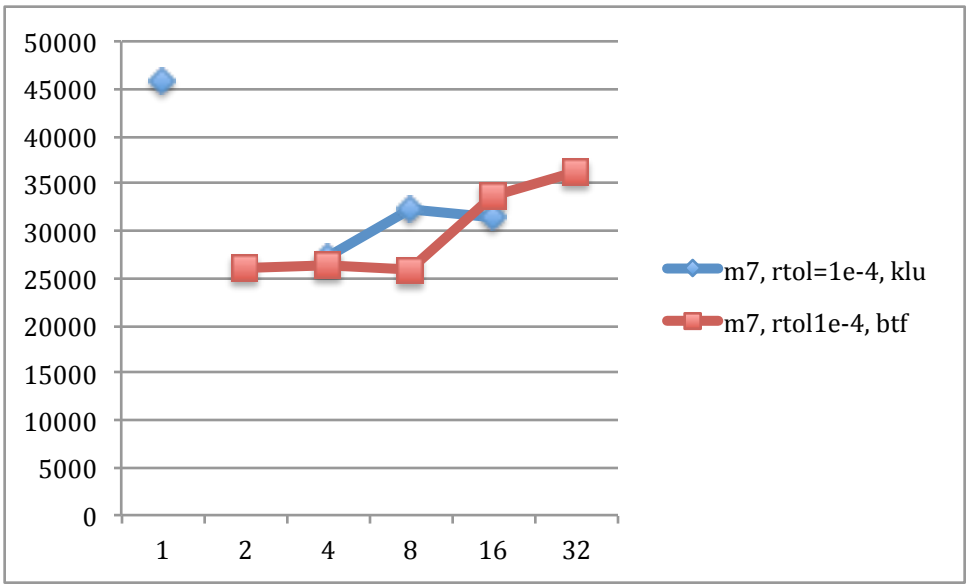


**Figure 3.1.** Xyce Runs: Comparison of time integration methods and tolerances. The x-axis is number of CPU's and the y-axis is run time in seconds.

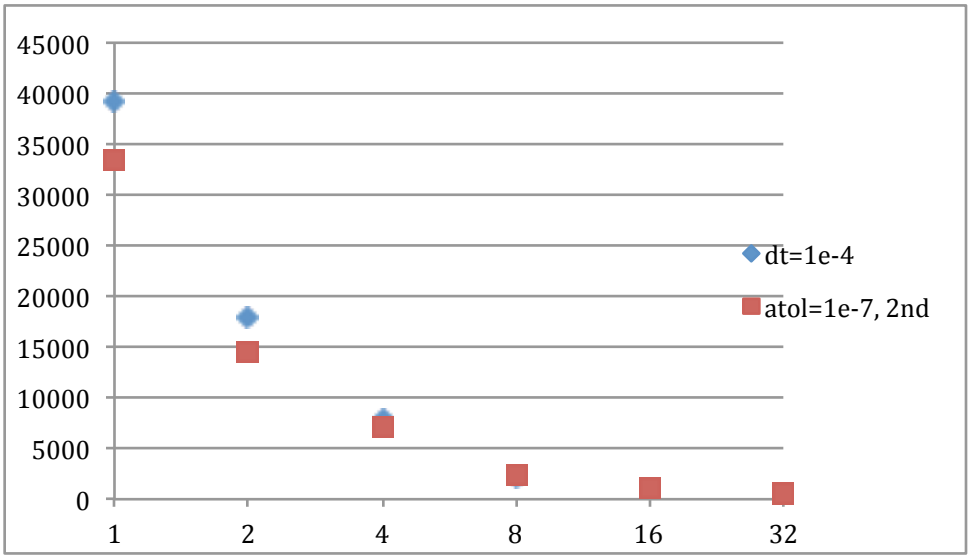


**Figure 3.2.** Neuron Runs: Comparison of time integration options. The x-axis is number of CPU's and the y-axis is run time in seconds.

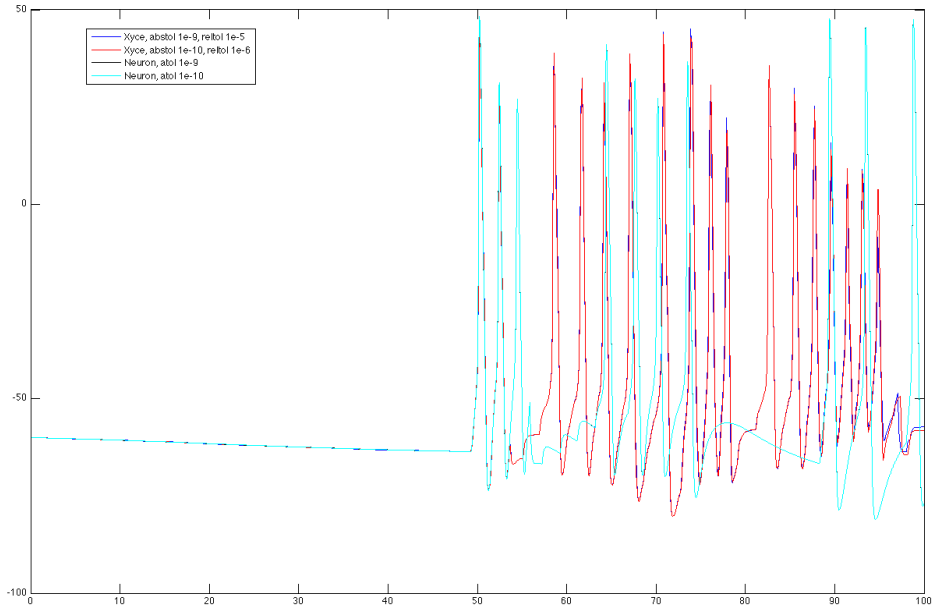


**Figure 3.3.** Comparison of spike traces.

+ tr_global_btf_verbose=1 use_ifpack_factory=0

For the Neuron runs (c.f. figure 3.2), dt=1e-4 indicates a fixed-time step integration using a time step of .1 microseconds (1e-4 ms); atol=1e-7, 2nd indicates that the variable time step integration scheme was used with an absolute tolerance of 1e-7; the 2nd refers to condition order, and means that spike times were interpolated within the time step interval.

# Simulation Accuracy

We found that different simulations of the same network between simulators or even with the same simulator using different integration options did not produce the same voltage traces for each neuron, or even the same number of spikes in the network. For this model, with its very sensitive neurons and large synaptic inputs, the time course and stability of the simulation is very sensitive to tiny differences (see figures 3.3 and 3.4). For both simulators, tightening the time integration tolerances increased the time over which the voltage traces remained consistent, but we did not find complete convergence. This is illustrated by the next two figures, showing sample voltage traces for the first neuron in the network for 100 ms and then 1000 ms.

This lack of ground truth simulation results led us to explore other means of analyzing the overall network behavior, described in chapter 4.

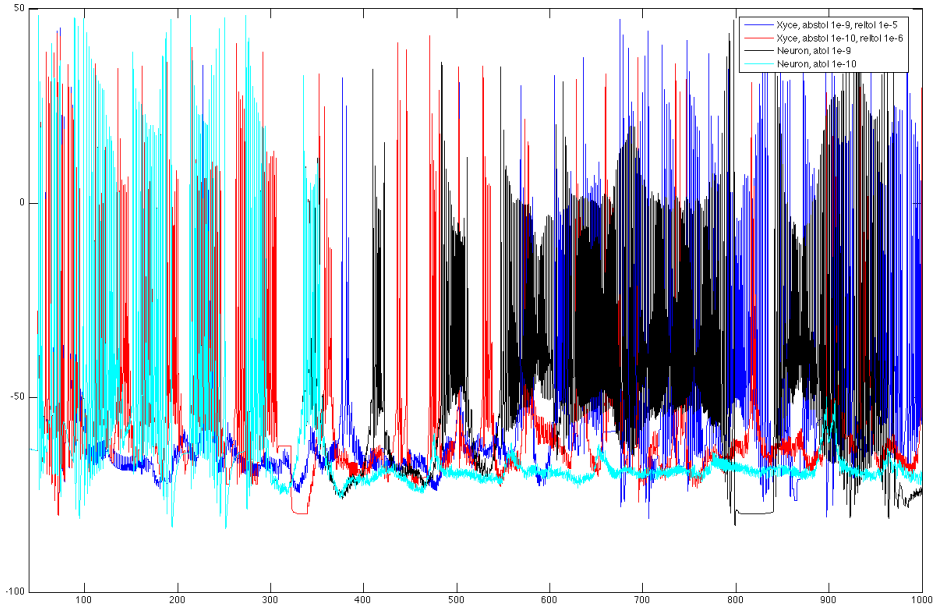


**Figure 3.4.** Comparison of spike trances over longer times.

# Scaling

To better understand how **Xyce** scales in parallel for neuron inspired problems, the same style of network described by Brette [27] was simulated on a cluster using 1 to 32 processors. As shown in figure 3.5 and 3.6, a two million neuron device simulation scales well to 16 or 32 processors. Significantly, the load solve time decreases as more processors are added indicating that the fundamental partitioning of the problem is efficient.

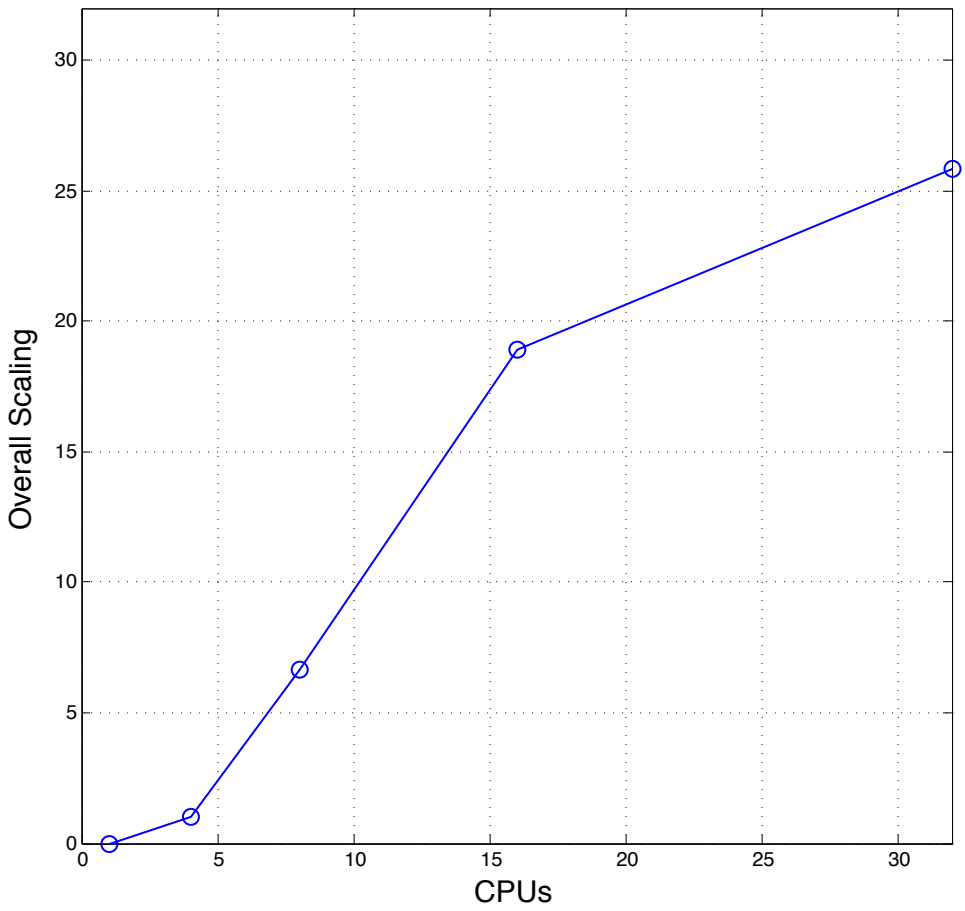


**Figure 3.5.** Overall scaling for a 10,000 neuron, 2 million synapse simulation.

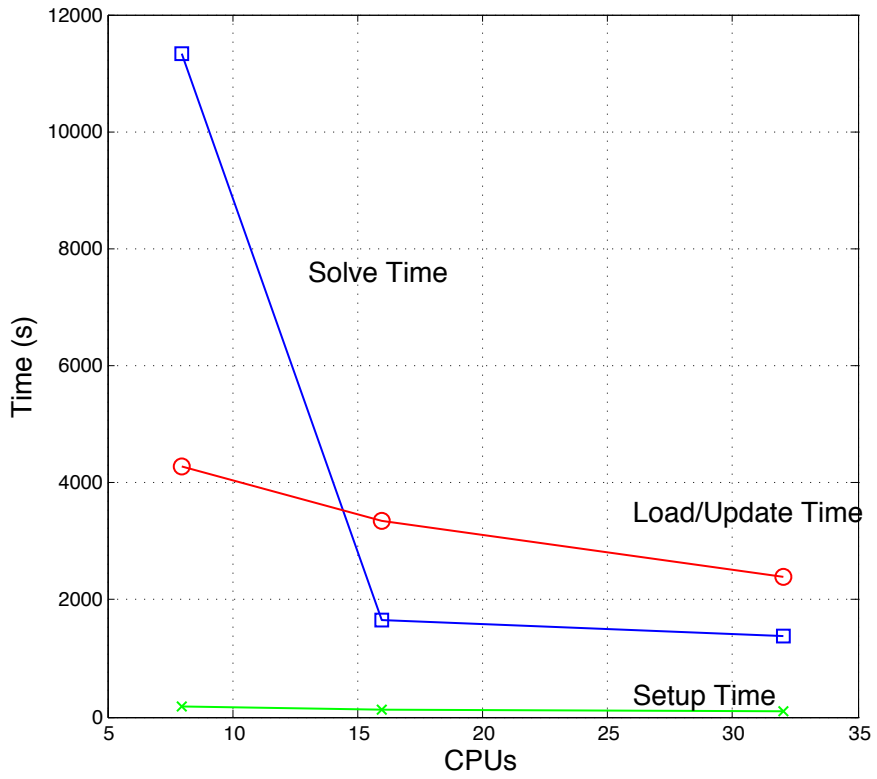


**Figure 3.6.** Breakdown of load, solve and setup time for a 10,000 neuron, 2 million synapse simulation.

# Chapter 4

# Analysis of large scale neural network simulations

## Introduction

The ability to simulate large scale neural networks provides a new problem. Even if simulations can be performed with a desired number of neurons for a desired length of time, the next consideration is how to analyze these results [14]. The foremost purpose for developing a sophisticated analysis approach is extracting insight about the modeled system. However, as shown in the previous chapter, since even identical networks can show considerable differences in behavior when different integration options are used, more sophisticated analysis approaches are necessary in order to quantify the effects of simulation conditions themselves.

Spiking neural output is a challenge to interpret for several reasons. The primary challenge is the large dimensionality; each individual neuron is an independent - but not necessarily orthogonal - measure of whatever the network is encoding or computing. An additional challenge is the temporal nature. A given neuron's code is typically sparse over time - the majority of the time the neuron is silent, but periodically it produces spikes if the inputs are sufficient to generate one. Spikes can therefore be considered highly informative about the neurons inputs, but typically only if time is taken into account. Simple analysis approaches, such as averaging over time using firing rate approaches, eliminate much of this information.

Here, we propose that a transformation of the temporal spiking data into a new reference basis to facilitate interpretation of network activity. The method of choice here is principal components analysis (PCA) [29], more generally known as singular value decomposition (SVD). PCA and SVD are widely-used dimensionality reduction methods, typically used to find an orthogonal basis set that compresses the most of a given parameter (variability in PCA's case) into as few of dimensions as possible. It is important to note that PCA is a linear technique, essentially rotating and stretching the coordinate system from "neuron-space" to "PCA-space". Although there are reasons to think that a simple linear reduction is not ideal (neurons themselves are highly non-linear), the use of PCA here is informative and may lead to further exploration on non-linear alternatives.

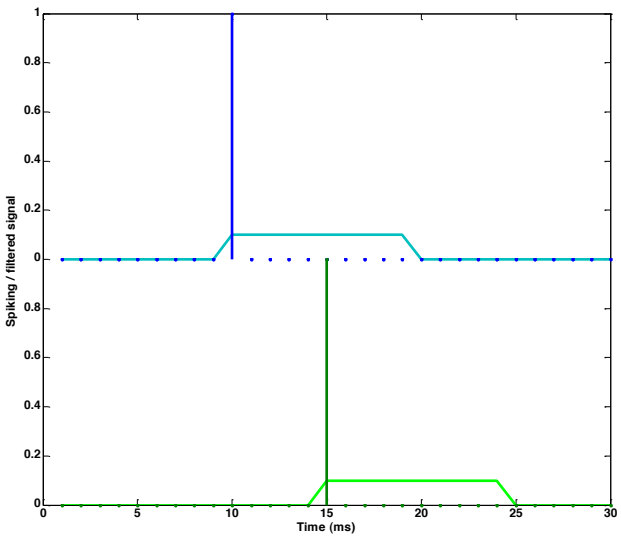


**Figure 4.1.** A simple filtering function.

# Theoretical Approach

Assume X is the collection of spiking events from a population of neurons. X can be constructed as an N x D matrix, where N is the number of observations over time (time dimension) and D is the number of neurons. Our approach is fairly simple: a pre-processing step, implementing PCA or SVD, and a post-processing step.

## Step 1: Pre-processing: temporal filtering of raw spiking data

As mentioned previously, the temporal structure of neuronal responses is critical to understanding their function. Because PCA treats all observations as independent and because spikes are essentially digital delta functions, it is thereby necessary to temporally smooth the spiking events in order to preserve the temporal relationship between observations. This temporal relationship is important to preserve since the subsequent analysis is based on the temporal correlations between neurons.

A simple linear filter, as illustrated in figure 4.1, is sufficient for our purposes. The linear filtering can be described as:

$$\chi_{filt}(n,d) = \frac{1}{\tau}\Sigma_{\epsilon=\eta-\tau}^{n}\chi(\epsilon,d) \tag{4.1}$$

Where $d$ is the neuron, $n$ is the current observation (i.e., time), $\tau$ is the length of the filter,

$\chi$ is the original spiking matrix, and $\chi_{filt}$ is the filtered matrix.

In the toy example shown, the top neuron fires 5 ms before the bottom neuron. From a neuroscience perspective, a temporal offset of this duration is interesting; it may be that the top neuron's spike caused the bottom neuron to spike; or it may be that the two neurons receive similar, but not identical, inputs. Regardless of the underlying cause, it is this temporal relationship we want to retain in our analysis. However, the correlation of the non-filtered signals (the single spikes) is very low; -0.0345 in the illustrated case. Providing each signal with a 10ms uniform filter, essentially spreading the spike equivalently over that window, yields a correlation of 0.25. Notably, the filter allows relative spike times to influence the correlational structure directly; neurons that reliably fire within a few milliseconds of one another will obtain stronger correlations than pairs of neurons whose activity is typically further apart in time.

While in our example we use a filter of 10ms, we typically used 25ms filters in our analysis.

## Step 2: Principle Component Analysis (PCA)

PCA was then performed on the filtered spiking matrix. The PCA process used was the standard approach of zero-meaning the filtered matrix ($\chi_{filt}$) and normalizing to the standard deviation of each dimension.

$$\chi_{filt,norm}(n,d) = \frac{\chi_{filt}(n,d) - \mu_d\left(\chi_{filt}(n,d)\right)}{\sigma_d\left(\chi_{filt}(n,d)\right)} \tag{4.2}$$

Next, the covariance matrix of the normalized filtered firing array is taken

$$C = \chi_{f,n}\chi_{f,n}^T \tag{4.3}$$

Next, the $dxd$ matrices of eigenvectors ($V$) and eigenvalues ($D$) of the covariance matrix $C$ are determined, such that

$$V^{-1}CV = D \tag{4.4}$$

The columns of the eigenvector and eigenvalue matrix are then re-sorted such that the eigenvalues (which exist along the diagonals in $D$) are in decreasing order. At this point, the orthogonal eigenvectors represented in $V$ represent the new basis set for the data in $\chi$, and the dimensions are ordered by the extent of variability of $\chi$ that they explain. $V$ contains the set of principal component vectors (PCx's), and can be considered as

$$V = \{PC_1, PC_2, PC_3, \ldots, PC_k\} \tag{4.5}$$

These PCs are the first result that we will use to examine the structure of dynamics. A reduced matrix $W$ can be constructed of only the top $k$ PCs that are desired for further analysis.

$$W = \{PC_1, PC_2, PC_3, \ldots, PC_k\} \tag{4.6}$$

### Step 3: Post-processing – projecting spiking data onto principal components

The eigenvectors obtained by PCA represent a basis set that is simply a transformation (through linear rotating and stretching) of our original space (neuron space). While the original data was used to determine which space is best suited to represent the original spiking data, the data itself is not represented in the basis description. To obtain this final piece, the original data is then projected into the new PCs. For the nxd matrix $\chi$ and a PC basis given by the $kxd$ matrix $W$ (which is the reduced matrix (from $V$) of the top $k$ PCs), the $nxk$ projection of the data $Y$ is given by

$$Y = XW^T \tag{4.7}$$

For the purposes of our analysis, we used the unfiltered spiking matrix $\chi$ here, and then we passed $Y$ through a comparable uniform filter to smooth the projections for visualization.

$$Y_{filt}(n, k) = \frac{1}{\tau}\Sigma_{\epsilon=n-\tau}^{n} Y(\epsilon, k) \tag{4.8}$$

The filtered $Y_{filt}$ matrix can be considered the path through the PCs that the simulation takes over time, and is our second result.

## Results

### Different simulation runs share principal components

The observation that the same model simulated with different tools or even with the same tool under different integration conditions yield markedly different behaviors (Chapter 3) is both disconcerting regarding our confidence in numerical simulations as well as problematic from an analytics point of view. Put simply, if an identical network shows qualitatively different behavior under different simulation conditions, how can we derive insight from the differences in behavior between two different networks?

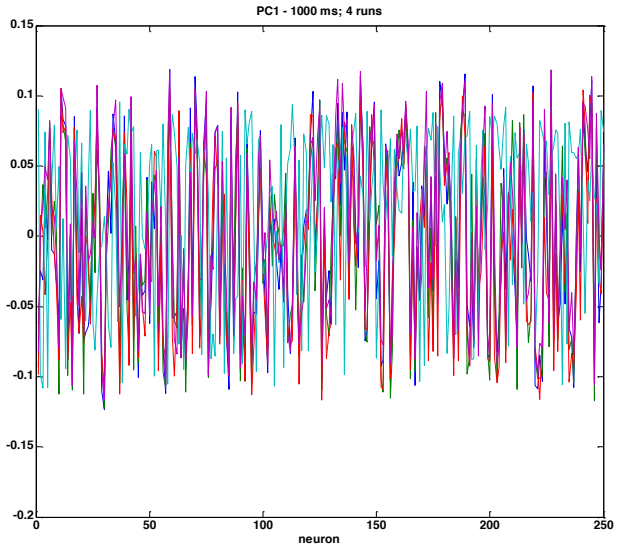


**Figure 4.2.** PC1 - 1,000 ms; 4 runs

We hypothesized that even though the raw spiking output of the networks differs extensively that there may be a commonality between the PCs of the spiking output. This would be the case if the underlying correlational structure between the neurons is unchanged even if noise due to simulation caused diverging network behavior. To assess this, we ran an identical simulation through Xyce (same netlist) for 10 seconds using different integration tolerances (1e-3 through 1e-9) and methods (m6 and m7). Over the first 1000 ms, there is some similarity between the primary PC across simulation methods, however there remain some differences.

Across all the runs, there was an average normalized dot product (NDP) of 0.72; while several of the runs shared much of the first PC, a couple of the runs did not.

This overlap of PC  strong between many of the simulations, and weak between a couple, is still striking, as the first 1000ms show drastically different dynamics between runs. After the top PC, the similarity breaks down somewhat. Shown below are the NDP scatter plots of the next two principal components (PC2, 0.5622; PC3, 0.45).

The indication that the first PC is conserved and the subsequent PCs are still loosely related between simulation runs was promising and indicates that the PC space is more robust to simulation conditions than the raw output. However, the lower similarities between higher PCs suggest that the approach is limited at the amount of time examined. Notably, 1000 ms is not very much time for understanding the correlation structure of 250 dimensions (neurons). We hoped that looking at longer simulations would provide a stronger relationship between the PC bases.

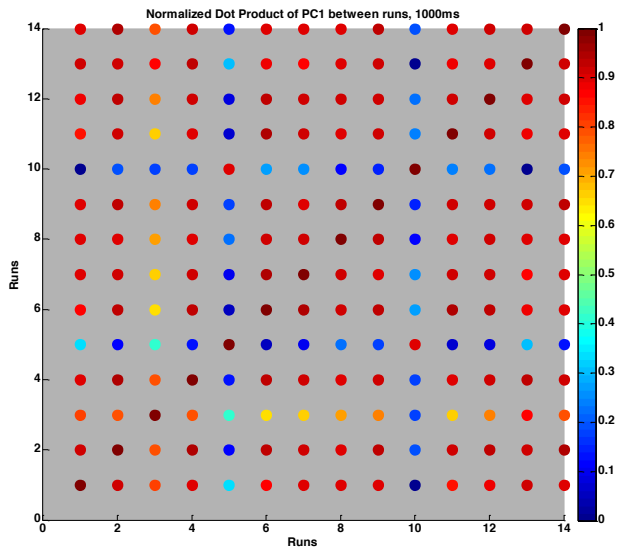


**Figure 4.3.** Normalized Dot Product of PC1 between runs, 1,000ms

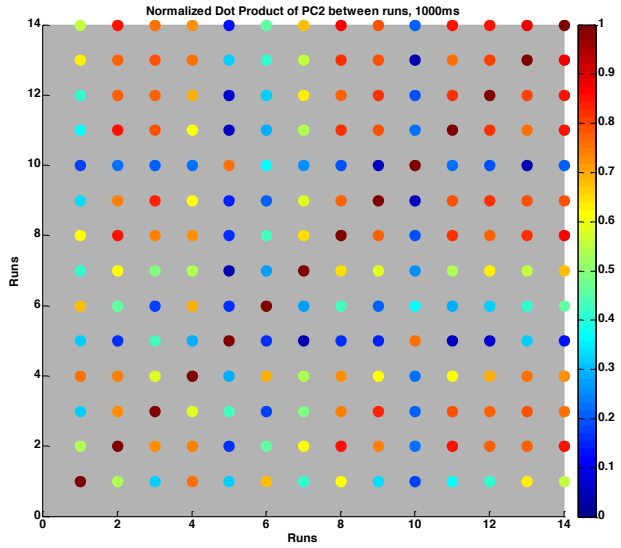


**Figure 4.4.** Normalized Dot Product of PC2 between runs, 1,000ms

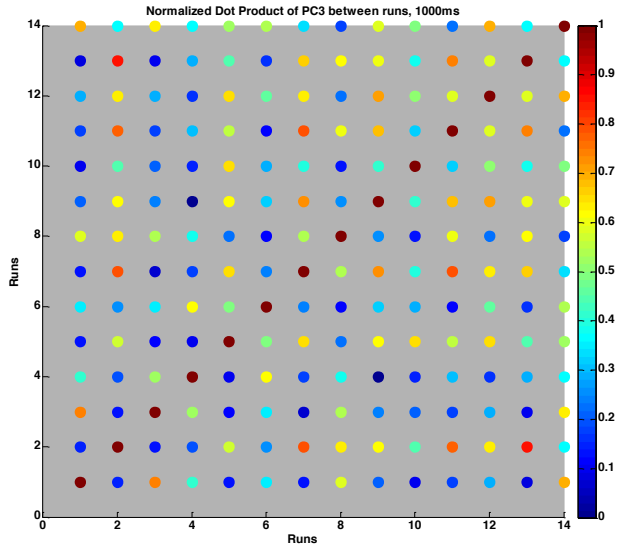


**Figure 4.5.** Normalized Dot Product of PC3 between runs, 1,000ms

Indeed, investigating 10000 ms (10x longer) yielded an incredible similarity (PC1 NDP = 0.99) between the primary principal components of runs. This is notable, because it suggests that even though the networks had longer to diverge from their initial similar starting points, the correlation structure of the dynamics converges.

This similarity holds up across all simulation runs and through the first several PCs (PC2, 0.98, PC3, 0.96; PC4, 0.90).

Notably, the relationships between higher PCs (which explain less of the overall variance in the dynamics and are thus more susceptible to noise) do eventually break down even in the 10 second runs. However, overall the longer the simulation runs, the more stable the PCs.

## Projection of raw data into PCs allows clear identification of divergence point

Although it is clearly evident that network behavior diverges in simulations using different integration options, the identification of a precise time when this divergence occurs is challenging. Quantifying a divergence time is useful because this can help better identify how different integration schemes relate to one another. Observing single neurons allows this to some extent (see Chapter 3), but with progressively larger systems it is not clear that a single neuron or a subset of neurons will be representative of when the population begins to

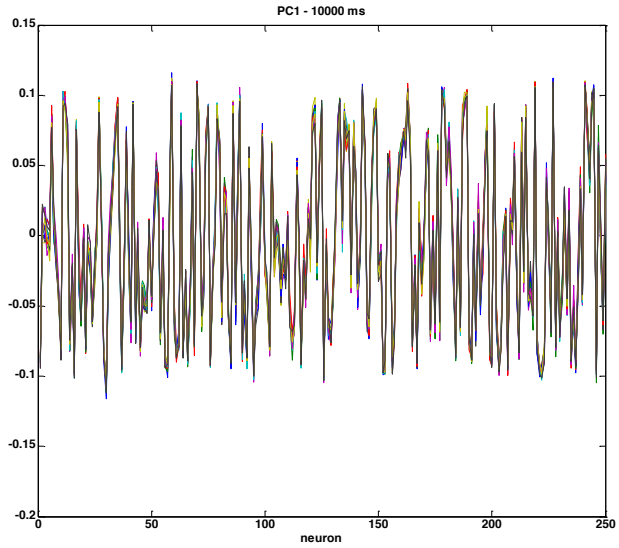


**Figure 4.6.** PC1 - 1,000 ms

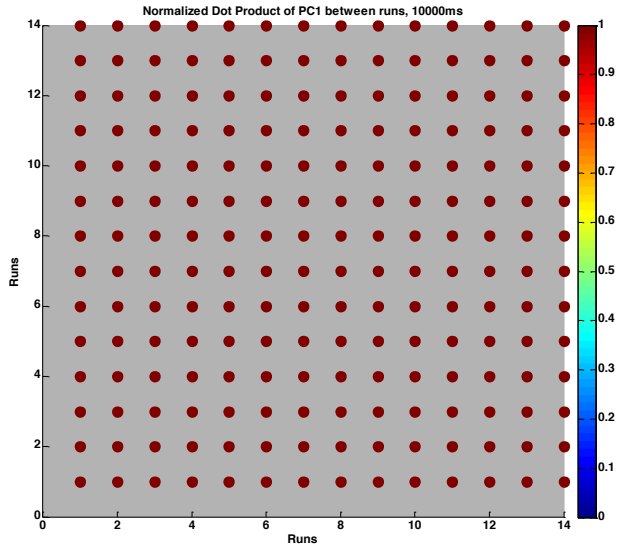


**Figure 4.7.** Normalized Dot Product of PC1 between runs, 10,000ms

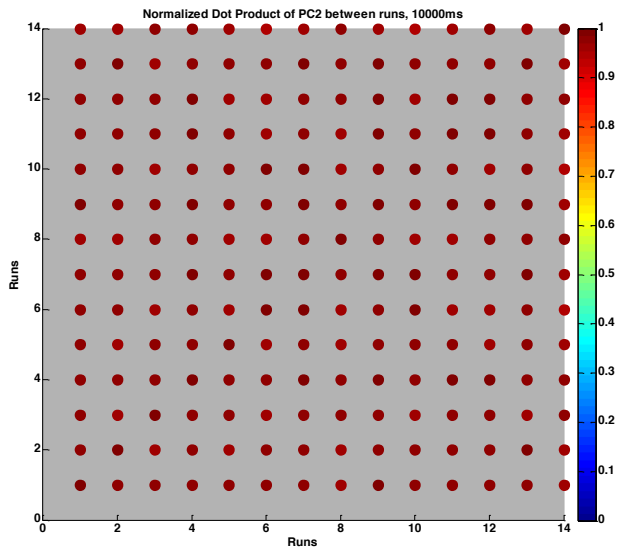


**Figure 4.8.** Normalized Dot Product of PC2 between runs, 10,000ms

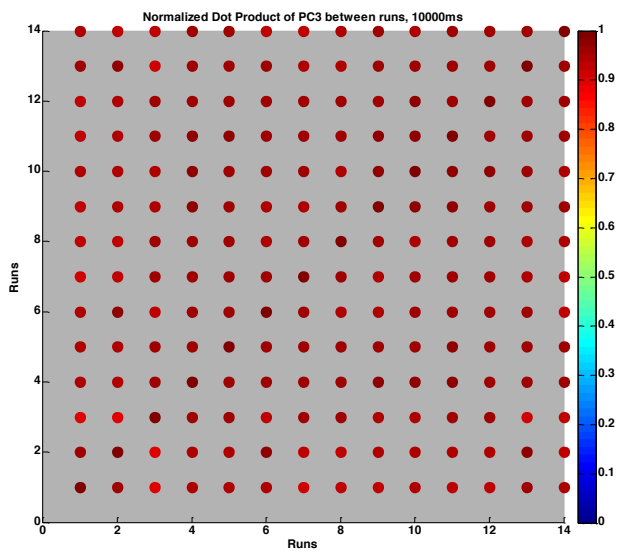


**Figure 4.9.** Normalized Dot Product of PC3 between runs, 10,000ms

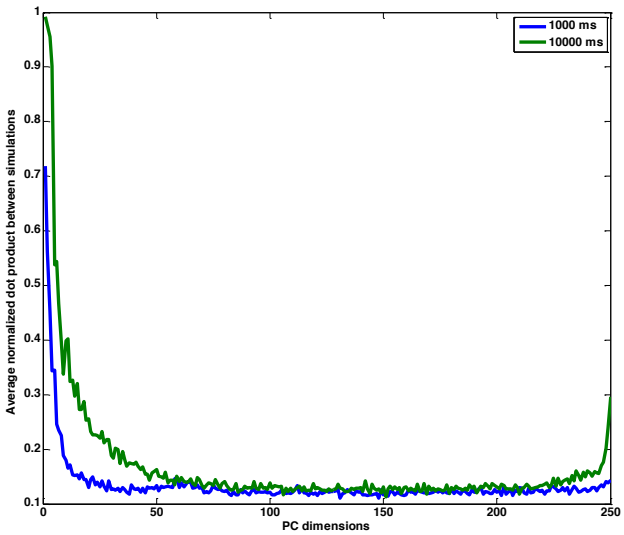


**Figure 4.10.** Average normalized dot product between simulations

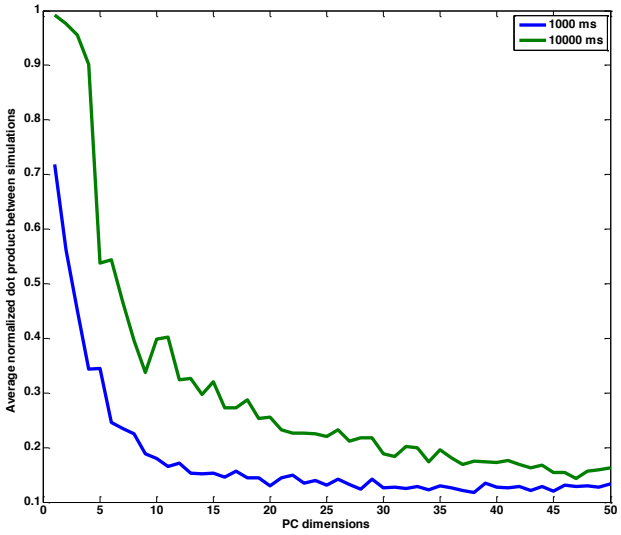


**Figure 4.11.** Average normalized dot product between simulations

behave differently.

Our observation that the different simulation runs share a common principal component basis set is useful in this regard, as it reduces a large fraction of the network behavior into a handful of dimensions. The behavior of the simulation as viewed through the top PCs enables us to observe when the populations begin to diverge and how they relate to each other in general.

As described in the methods, the principal components of the total spiking data for that network were determined. To do this, we simply combined all the different simulation runs into the equivalent of a really long simulation. Notably, as described above, the top PCs of the individual runs were shared across simulations, so it was not surprising that the top PCs of combined data set were highly similar to the PCs of the individual runs (Combined PC1 had approximately an NDP of 0.995 to each single simulation PC1). Projecting the raw spike output from each individual simulation provides the path in PC-space that that simulation took.

Figures 4.12 and 4.13 show the projection of the first 500 ms of a single simulation run into the combined PC1. Both the unfiltered (raw spike train projected into PCs) and a filtered version are shown. Subsequent plots will show only the filtered version of the projection. In this particular simulation, after an initial quiescent period, the model experienced a small movement in the negative PC1 direction, and then experienced a prolonged displacement in the positive PC1 direction. In contrast, the right figure shows a second simulation run (identical network, different integration tolerance (1e-4 vs 1e-3)). The networks' behavior initially appears similar, however after about 100 ms, the second simulation returns towards 0 in the PC1 dimension, while the first simulation moves further in in the positive direction in PC1.

Combining the figures (see 4.14), it is clear that the first evidence of divergence occurs around only 84ms.

Combining all of the simulation runs in which the m6 Xyce solver was used, it appears that while some simulations held together for roughly 100 ms (a pair even stayed together until 150ms), the runs diverged as early as 70ms. Notably, the propensity for runs to stay consistent with one another did not appear to be related to their relative tolerances; while one would expect that higher tolerances should approach a ground truth with runs staying together for longer, that does not appear to be the case here.

Looking at the m7 Xyce runs, there is a similar trend. The jobs begin to diverge at different times, some as early as 75ms. Unlike the m6 runs, it does appear that there is a progression to when they diverge, it appears that the looser tolerance jobs progressively diverge earlier than the tighter tolerances. For instance, the 1e-9 and 1e-8 simulations, which were the two tightest, are aligned in PC1 for almost 175ms, whereas 1e-3 and 1e-4 diverge very early in the simulation run.

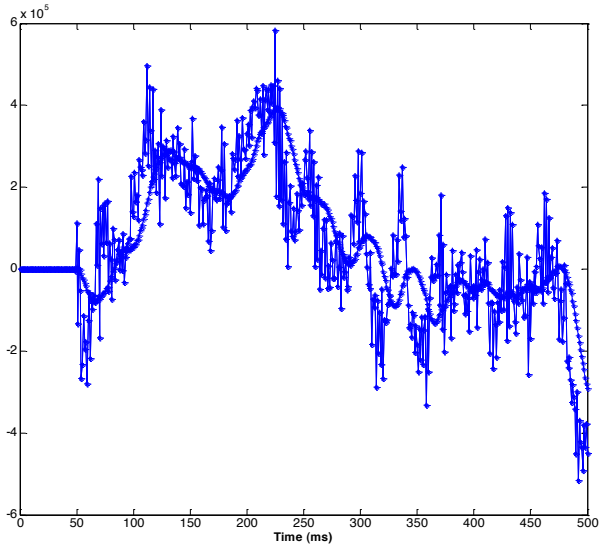


**Figure 4.12.** Projection of the first 500 ms of a single simulation run into the combined PC1

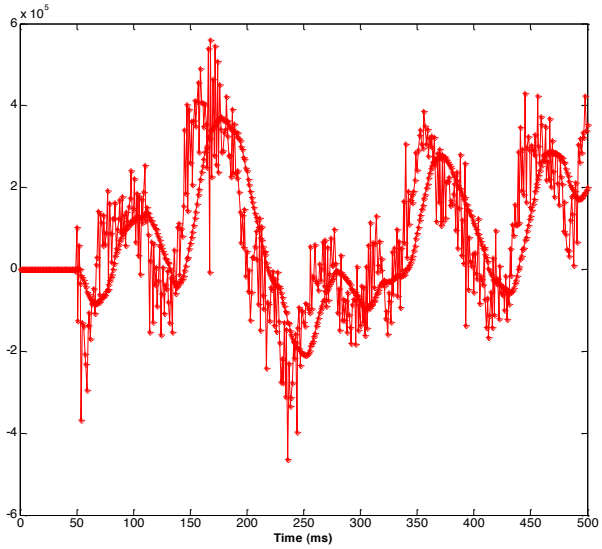


**Figure 4.13.** Projection of the first 500 ms of a single simulation run into the combined PC1

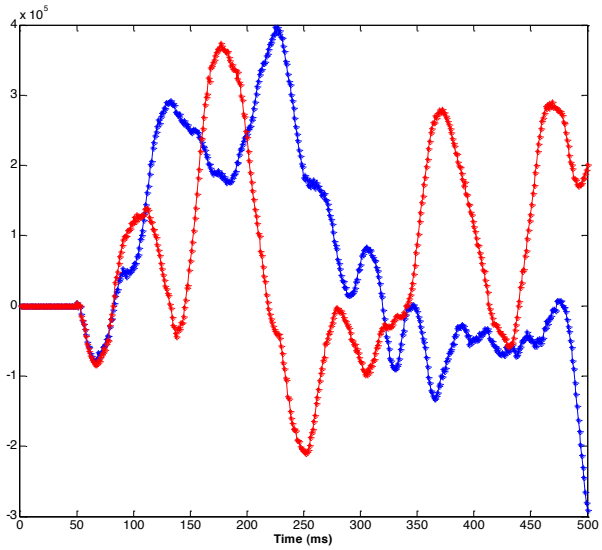


**Figure 4.14.** Combined projection of single simulation runs of PC1

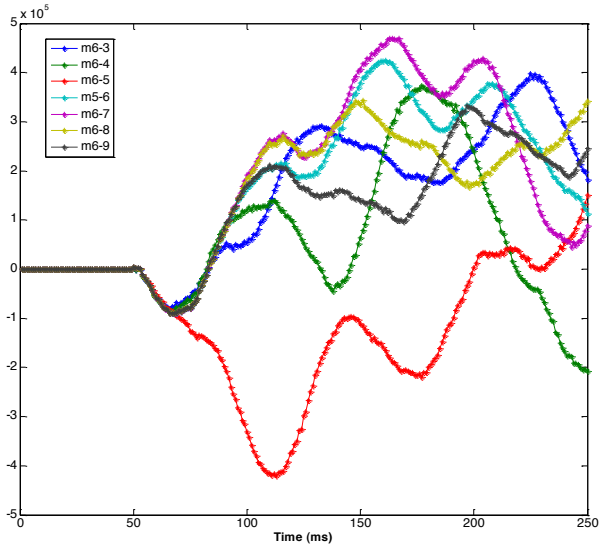


**Figure 4.15.** Combined projection form all Xyce simulations using method=6 for PC1

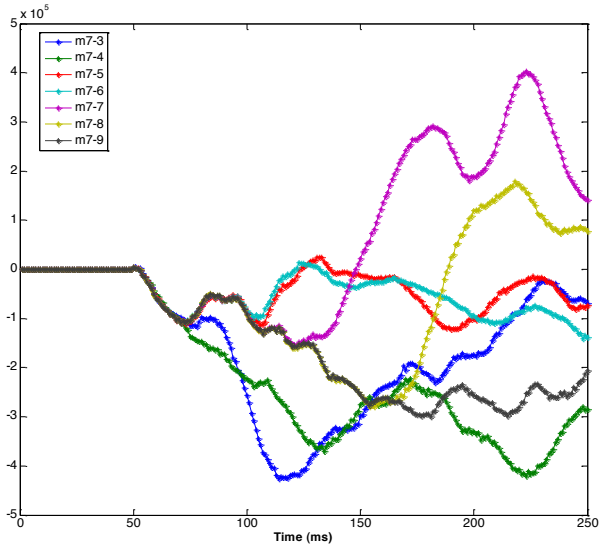


**Figure 4.16.** Combined projection form all Xyce simulations using method=7 for PC1

# Discussion

As described in the above results, the observation that simulation runs with rapidly divergent behavior share PCs has a direct value in quantitatively assessing when comparable simulations diverge. However, it is interesting to consider the broader ramifications of this observation. At one level it is not entirely surprising. Shared principal components imply a shared covariance matrix and correlational structure of the entire data set. The fact that correlations between neurons appear to be time independent is likely an indicator that the correlational structure of the data has more to do with the network architecture than the state of the network at any given time. Even if neurons happen to be co-activated at a given time by chance, one would expect that over long simulations random co-activations will cancel one another out. Whereas when neurons are co-activated due to common inputs or mutual connectivity, one would expect that their correlations will persist regardless of the simulation duration. Notably, neurons downstream of such networks in the brain typically will sample a large number of neurons and are likely tuned through learning mechanisms over long periods of time. Thus, we would then expect that downstream neurons over time become sensitive to higher order correlational structures in the source networks dynamics, not the more unpredictable patterns of behavior observed acutely in the raw simulations. For instance, in the networks described above, one could imagine a downstream neuron sampling many of the neurons that are positively represented in PC1, and another preferentially sampling those neurons that are negatively represented in PC1. Now, these neurons would be activated at different times in each simulation (whenever PC1 is strongly positive or

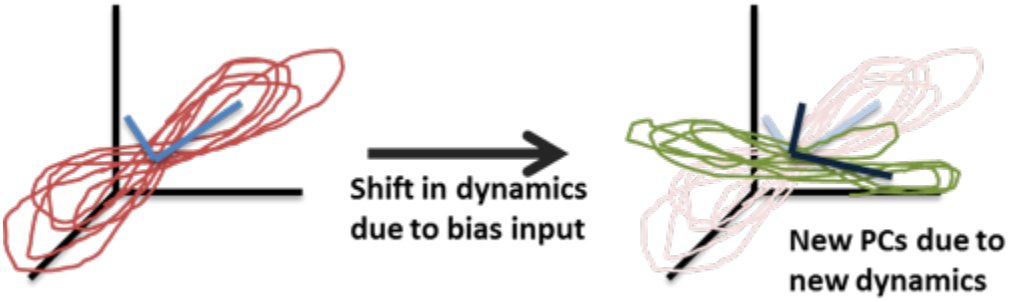


**Figure 4.17.** How PC dynamics change due to bias.

negative), but over a given amount of time the expectation would be that these neurons would be activated fairly often. Such a mechanism becomes particularly interesting if, for whatever reason, the relevant PCs of the network were to change. As mentioned above, the PCs are likely due to the network architecture and intrinsic rules governing the dynamics of the components and connections. If the network were to shift in a fundamental way, either through a bias provided by an upstream region, or through learning or neuromodulation, then it would be expected that the PCs would shift accordingly. Take the example of a bias to the network, which could just be tonic input from a set of upstream neurons. This bias would change the correlational structure of the dynamics; for example, two otherwise uncorrelated neurons now may receive a similar bias input. (Note: the extent of this effect would depend on the strength of the bias (external input) relative to the influence of the internal connectivity.) Now, because the correlational structure of the network is different, the proper PC basis would be rotated to become somewhat different (see Figure).

Now consider our hypothetical downstream neurons. The neurons that responded to the original PC1 would not typically be activated under the new situation. Whereas if there are any neurons tuned to the biased network's PCs, they could now be preferentially activated. As a result, different inputs can activate a different correlational structure in the recurrent network, a change that can be detected by output neurons that are tuned (through learning at synapses) to the relevant high-dimensional components. It is important to note that while we used PCA, there is nothing special about this approach beyond its practical use as an analysis approach. Indeed, it is likely that the most relevant combinations of neurons in neural coding (the equivalent of PCs in this case) are not actually orthogonal.

# Chapter 5

# Reduced-Order Modeling of Neuron

The reduced-order modeling (ROM) techniques have been used to accelerate simulations of dynamical systems. In this chapter, a Krylov subspace based ROM technique is presented to effectively reduce complexity of large dentrites in multi-compartment neuron models. The experimental results show that simulation with ROMs can achieve large speedup over the full model with the same accuracy.

## Reduced-order modeling

The reduced-order modeling techniques for linear time invariant systems (LTI) have been successfully applied to circuit simulations. There are a variety of well established LTI ROM techniques. These techniques are often based on projection of a LTI system into lower dimension subspaces.

A linear circuit can be described by differential algebraic equations:

$$\begin{aligned} \mathbf{C}\tfrac{d\mathbf{x}}{dt} &= -\mathbf{G}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{L}^T\mathbf{x}(t), \end{aligned} \tag{5.1}$$

where $\mathbf{C} \in \mathbb{R}^{n\times n}, \mathbf{G} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$ and $\mathbf{L} \in \mathbb{R}^{n\times p}$ and $\mathbf{x}(t)$ is the state, $\mathbf{u}(t)$ the input and $\mathbf{y}(t)$ the output of the system. Also, $n$ is the size of the original system and $p$ the number of inputs (outputs). If $p = 1$, then (5.1) is referred to as a single-input-single-output (SISO) system and, if $p > 1$, it is a multiple-input-multiple-output (MIMO) system.

For model order reduction, one constructs two projection matrices $\mathbf{W} \in \mathbb{R}^{n\times k}$ and $\mathbf{V} \in \mathbb{R}^{n\times k}$ such that $\mathbf{W}^T\mathbf{V} = \mathbf{I}_k$, where $k$ is the desired size of the reduced system ($k \ll n$). The reduced system is now $\hat{\mathbf{\Sigma}} \equiv (\hat{\mathbf{C}}, \hat{\mathbf{G}}, \hat{\mathbf{B}}, \hat{\mathbf{L}})$ governed by the following set of first-order LTI differential equations

$$\begin{aligned} \hat{\mathbf{C}}\tfrac{d\hat{\mathbf{x}}}{dt} &= -\hat{\mathbf{G}}\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}\mathbf{u}(t) \\ \hat{\mathbf{y}}(t) &= \hat{\mathbf{L}}^T\hat{\mathbf{x}}(t), \end{aligned} \tag{5.2}$$

where $\hat{\mathbf{C}} = \mathbf{W}^T\mathbf{C}\mathbf{V}$, $\hat{\mathbf{G}} = \mathbf{W}^T\mathbf{G}\mathbf{V}$, $\hat{\mathbf{B}} = \mathbf{W}^T\mathbf{B}$, $\hat{\mathbf{L}}^T = \mathbf{L}^T\mathbf{V}$.

The frequency input-output relationships of the original (5.1) and reduced (5.2) systems are determined by their corresponding transfer functions:

$$\begin{aligned} \mathbf{H}(s) &= \mathbf{L}^T(s\mathbf{C} + \mathbf{G})^{-1}\mathbf{B}, \\ \hat{\mathbf{H}}(s) &= \hat{\mathbf{L}}^T(s\hat{\mathbf{C}} + \hat{\mathbf{G}})^{-1}\hat{\mathbf{B}}. \end{aligned} \tag{5.3}$$

In Xyce, a Krylov subspace based ROM technique is developed. The algorithm, PRIMA (passive reduced-order interconnect macromodeling algorithm) is proposed by Odabasioglu et al. [26] in 1998. The algorithm utilizes the block Arnoldi procedure. Note that for PRIMA, $\mathbf{W} = \mathbf{V}$. The resulting reduced system $\hat{\mathbf{\Sigma}}$ is proven to be passive and hence, stable. The number of matched moments is equal to the desired size of the reduced system $\hat{\mathbf{\Sigma}}$ divided by the number of inputs, i.e., $l = k/p$.

However, the experiments show that transient simulation with the reduced models can be slower than the simulation of full models. The transient simulation with ROMs can take more time steps and linear solve for each step can be more expensive. To accelerate the transient simulation with ROMs, we developed scaling and sparsification techniques. These techniques make ROM algorithm much more efficient and robust.

# The method

In this section, the neuron simulation using ROM technique is described. A neuron can be accurately modeled as multiple compartments. A compartment can be modeled as equivalent circuit models. This makes it possible to apply circuit simulation techniques for neuron simulation.

The main idea of ROM for multi-compartment neuron model is to separate the nonlinear parts from the neuron cell and then apply ROM to the pssive dendrites. The nonlinear parts are the compartments that have active ion channels. For example, the soma is modeled by the Hodgkin-Huxley model described in the previous Chapters and is highly nonlinear. The synaptic input is modeled as a synaptically activited ion channel described previously and is also nonlinear. The linear parts are the passive dendrites. They are modeled as an equivalent multiport linear RC network and can be reduced by an efficient Krylov subspace based MOR technique described in the previous section. The nonlinear parts and linear parts are connected by ports. The ROM technique in Xyce not only generates a smaller model which matches both the frequency domain and time domain responses of the full model, it also preserves the passivity of the full model.

The proposed method is very efficient for neuron model reduction. First, the dendrites in neuron is modeled by RC network which is more suitable for reduction than RLC network in circuits. Second, the frequency range of neuron activities is band limited and it has a

significant low frequency component. This makes the size of the reduced systems smaller. The proposed method is very efficient with a small number of ports and can be easily extend to reduce the quasi-active systems.

# Example

In this section, we apply the proposed method to neuron simulation and show large speedup over the full model simulation. The neuron in our test has a nonlinear soma and the passive branched structure for dendrites that are similar to rallpack2. A positive current is injected to the soma and spiking trains of action potentials are generated due to the somatic current injection.

The original neuron model has about 10000 unknowns. The reduced model has 100 unknowns. The transient simulation with reduced model takes longer time than the simulation of the full model. With sparsification technique, the simulation of ROM has more than 30 times speedup over the simulation of the full model. The transient simulation performance of the reduced model is further improved by using both the scaling and sparsification techniques. We obtain a speedup of 50 times over the full model. Figure 5 compares the transient simulation among the full model and reduced models. As can be seen in the figure, the results from reduced models match that of the full model well.

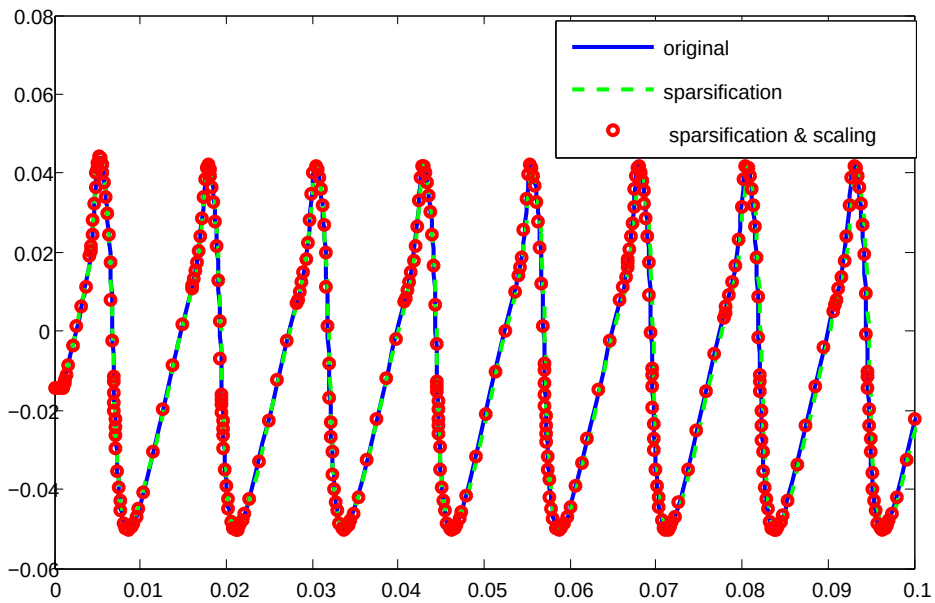


**Figure 5.1.** Transient comparison: the full model vs. the reduced models

# Chapter 6

# Fitting the parameters of neural models to biological data

## Introduction

There are a variety of mathematical models that have been developed to simulate the activity of spiking neurons. These models range from very simple phenominalogical models such as the integrate and fire model, to complex models such as Hodgkin-Huxley models which aim to recreate the actual biological mechanisms involved in creating the voltage spikes in neurons. Different models vary in their ability to recreate biologically realistic spiking behavior. Simple models are computationally much faster but cannot recreate many aspects of biological behavior. More complicated, biologically realistic models can recreate biological firing patterns but are computationally very expensive.

Due to the trade off between computational load and realistic behavior, the model a researcher chooses to implement will depend on the specific phenomena that is under investigation and the computational resources available. However, what is common to the modeling process is that all models have parameters. Regardless of the specific model, these parameters will have to be tuned in order to produce the desired behavior.

Because of the non-linear nature of neural models, parameter fitting of neural data has been a notoriously arduous process [35]. It has been shown that tiny fluxuations in parameters can drastically affect the spiking behavior of neurons [25]. Many researchers attempt to hand tune the model parameters. Occasionally hand tuning works for simple models, but often becomes impossible with biologically realistic models. As a result researchers have turned to established parameter fitting techniques. Many techniques are available such as simulated annealing, genetic algorithms, particle swarm methods, and complete sampling regimes [35, 28]. In order to use any of these methods, first and fitness function (often referred to as and error function or a cost function) must be chosen. The fitness function quantifies how similar (or dissimilar) the model behavior is to the desired, or what we will refer to as the "target" behavior.

In the following text, we will describe the biological data we are aiming to fit, define our choice of fitness function, and demonstrate the parameter fitting methods we have pursued.

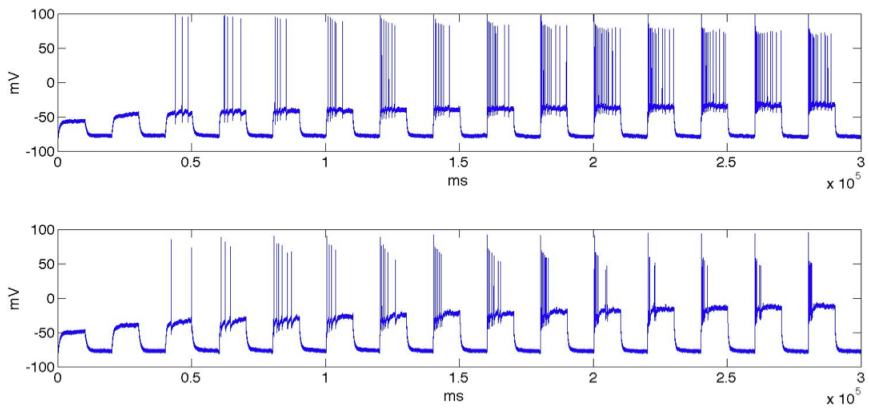


**Figure 6.1.** Two examples of intracellular voltage recordings with current injection from dentate gyrus hippocampus neurons. Each voltage 'wave' (of which their are 15) corresponds to a current injection. The first wave is the increase in voltage due to a current injection of 10 pA. The subsequent waves correspond to current injections increasing in size of 10 pA. Notice that both of the neurons start firing action potentials when 30 pA are injected.

# Biological Data

In order to collect intracellular current clamp electrophysiology data, an electrode is inserted into a live neuron. Then, current is injected into the neuron in pulses (often causing the neuron to spike) and the resulting voltage within the neuron is recorded. An example of this data is shown in Figure 6

# Choice of Voltage Train Characteristics

Before a fitness function by which the quantify the similarity between the model produced spike train and target spike train can be chosen, one needs to define the aspects of the spike train that are important. For example, often researchers can characterize a neuron in terms of its frequency of firing. Other general measures include how 'bursty' versus tonically a neuron fires. For example, in the lower voltage plot of Figure 6, in the 11th voltage wave (starting at approximately 200 sec) the neuron fires a burst of spikes, followed by a period of quiescence followed by another burst of spikes. Yet another measure involves the amount of adaptation a neuron shows. Adaptation refers to the behavior where a neuron initially fires quickly and then slows down over time as it 'adapts' to the stimulus. Evidence of this phenomena can be seen in both of the voltage traces in Figure 6.

All of the above mentioned characteristics are commonly used to characterize the spiking behavior of neurons. These are general descriptions, however, their are other aspect of firing behavior that these do not describe: mainly the specific timing of the spikes and the shape of the spikes. It is hotly debated in the neuroscience field whether the exact timing of spikes is important for computation. However, there is substantial evidence that the exact timing of spikes may indeed have an impact [8, 32, 33, 4]. The shape of a spike refers to the height and width of a spike as well as the subthreshold dynamics (shape of the voltage trace before the neuron reaches the voltage after which it is guaranteed a spike will be fired). While it is possible that attributes of shape may be important for computation, it is likely that these attributes are a by-product of the ion channels needed to produce the correct, obvious binary spiking behavior that indicates whether a neuron has been adequately stimulated to reach voltage threshold. Therefore, we focused our efforts on the patterns and timing of the spiking behavior and neglected to consider the spike shape. We chose to use a method that measures the similarity between the timing of the spike trains. The exact timing of the spikes is a more precise measure of spiking behavior than the general descriptors above. In addition, they are essentially included 'for free' in spike timing measures. I. e. if the behavior of a neuron is bursty, by fitting the times of the spikes, the obtained model behavior will also have to be bursty to achieve an adequate similarity.

There are various quantitative methods that can be used to evaluate the timing similarity of spike trains [22]. We chose to use a correlation based method developed by Schreiber and colleagues [30] described in the next section.

# Fitness Function

We used a correlation-based method developed by Schreiber and colleagues [30]. The correlation measure is defined as

$$R_{corr} = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{\vec{s_i} \cdot \vec{s_j}}{|\vec{s_i}||\vec{s_j}|} \tag{6.1}$$

Where $R_{corr}$ varies between 0 and 1; 1 would correspond to two identical spike trains, 0 is equivalent to a poor fit. $\vec{s_i}$ and $\vec{s_j}$ are the filtered spike trains. $\vec{s_i}$ and $\vec{s_j}$ are obtained by convolving the binary spike trains with a Gaussian function. The width of the Gaussian will define how much jitter versus missing and additional spikes are allowed. I.e. if a sharp Gaussian is used, the timing of the spikes will need to be precise in order to obtain a value of $R_{corr}$ close to 1. In general, we used a $\sigma = 5$ ms.

# Parameter Fitting Methods

The objective, is to tune model parameters in order to recreate the spiking behavior of the target, experimental voltage trace. With this goal in mind, we chose to explore one optimization method and one sampling method available in the Sandia DAKOTA software package: evolutionary algorithms and latin hyper square sampling. These methods and the results are described below:

## Basic Strategy of Testing Methods

To test the efficacy of the fitting methods we first made a *test target* trace with known parameter values from the model itself. We then attempted to fit the *test target* trace.

## Evolutionary Algorithms

Evolutionary or genetic algorithms are are a set of generic population-based metahuristic optimization algorithms. The algorithms use mechanisms inspired by biological evolution to find a solution. Evolutionary algorithms are global optimization algorithms that are less likely to 'get stuck' in local minima. There are a many evolutionary algorithms available with many different parameters to be set. Two genetic algorithms exist in Dakota: colony_ea and JEGA. Here we used colony_ea as the algorithm is a bit simpler and we did not yet have a need for multiobjective optimization as is offered by the JEGA method. Genetic algorithms have proven to be an effective method to fit neural data in the past [10]. In order to test the effectiveness of the colony_ea method on Hodgkin-Huxley like equations we tried to fit the conductance based models of Mainen and Sejnowski [23]. It had been shown previously that the Matlab genetic algorithm could find parameters that would yield a fitness better than 0.9 (Teeter and Chan, in preparation).

The model equations can be found in the original article [23]. An example DAKOTA input file using colony_ea is below:

```
strategy,
single_method
tabular_graphics_data

method,
coliny_ea
  max_iterations = 500
  max_function_evaluations = 1000000
  population_size = 50
  initialization_type unique_random
```

```
  fitness_type merit_function
  mutation_type offset_normal
  mutation_scale=.5
  mutation_rate = 1
  crossover_type uniform
  crossover_rate 0.8
  replacement_type elitist = 2
  solution_target=.1  #target below which target will stop
  non_adaptive

model,
single

variables,
continuous_design = 10
  lower_bounds    10*0.0
  upper_bounds    200 2000 3 1 30 200 1 30 3 200
  descriptors
      'gna_soma'
      'gkv_soma'
      'gca_soma'
      'gkm_soma'
      'gkca_soma'
      'gna_dend'
      'gkm_dend'
      'gkca_dend'
      'gca_dend'
      'gkv_dend'

interface,
fork #like a matlab system call
  analysis_driver = 'runneuron.sh'
  parameters_file 'params.in'
  results_file 'results.out'
  work_directory named 'workdir'
  directory_tag
    template_files
'MainenGaL4Stellate.hoc'
'conductances.dat.template
'x86_64'
'MainenL4StellateDefaultConddt0p05T.dat'
'MainenL4StellateDefaultConddt0p05V.dat'
'cells'
'peaks.py'
'calcfitness.py'
```

```
responses,
objective_functions = 1
no_gradients
no_hessians
```

Here we had variable levels of success fitting our *test target* trace. There were four example *target traces* in the Mainen article. The above DAKOTA input file could do a reasonable job of finding a set of parameters that would yield a fitness greater than 0.9 for the small Aspiny neuron (colony_ea could find a solution about 1 of 5 times the algorithm was run). However the success dropped of to approximately 1 in 50 for the slightly larger L4 Stellate cell and could not find appropriate parameters for the larger neurons. There are several differences between the Matlab genetic algorithm and the algorithms available in DAKOTA. Due to the difficulty in altering the original DAKOTA code to implement additional feature we did not pursue this method. However if DAKOTA genetic algorithms are used to pursue neural modeling in the future, Matlab features could be added.

## Latin Hypercube Sampling

Latin hypercube sampling (LHS) is a statistical method for generating a distribution of plausible collections of parameter values from a multidimensional distribution [1, 17]. Because of the highly non linear nature of neural models, a very large number of samples are required to find a set of parameters that adequately represent the space. In this case, it is unlikely that LHS gives an advantage over Monte Carlo methods.

Initially we were curious how many samples would yield a fit greater than 0.9 to a test target spike train. We took 1 million samples (the memory on the Redsky super computer will not hold more than a couple million samples). Below is example DAKOTA input file using LHS to sample the Izhikevich model described at the end of this chapter.

```
strategy,
single_method
tabular_graphics_data
model,
single

method,
sampling,
samples = 1000000
sample_type = lhs

variables,
uniform_uncertain = 11
```

```
#               C   Vt    Vr    k    a1  b1    c1    d1      a2   b2   d2
lower_bounds    0   -75   -90   0    0   -4    -75    0      0   -4    0
upper_bounds    500 -20  -20    1    1     4      0    1000   1     4    1000
descriptors
'C'
'Vtin'
'Vrin'
'k'
'a1'
'b1'
'c1'
'd1'
'a2'
'b2'
'd2'

interface,
fork #like a matlab system call
  analysis_driver = 'runC.sh'
  parameters_file 'params.in'
  results_file 'results.out'
  work_directory named 'workdir'
  directory_tag
    template_files
'testTraceMakeWithC100Vt-45Vr-75k0p1a0p1b-0p1c-60d100a0p005b0d50a0b0d0.dat'
'single_Izy_proto_multiu.o'
'parameters.dat.template'
'peaks.py'
'calcfitness.py'

responses,
objective_functions = 1
no_gradients
no_hessians
```

The test target trace was made with $C = 100, V_t = -45, V_r = -75, k = 0.1, a_1 = 0.1, b_1 = -0.1, c_1 = -60, d_1 = 100, a_2 = 0.005, b_2 = 0$, and $d_2 = 50$. Figure 6 shows the normalized parameters that yielded a fit better than 0.9 using a Gaussian filter value of $\sigma$=5 ms. The variables were allowed to vary between the values listed in the example DAKOTA input file above and on the figure heading. 198 parameter sets yielded a better than 0.9 fit. Notice how there doesn't seem to be structure to the parameter values that yield a sufficient fit. This suggests that there is not a region of the parameter space that fits the target behavior, instead there seems to be a diverse set of solutions to the problem. This speaks to the highly non-linear nature of the space.

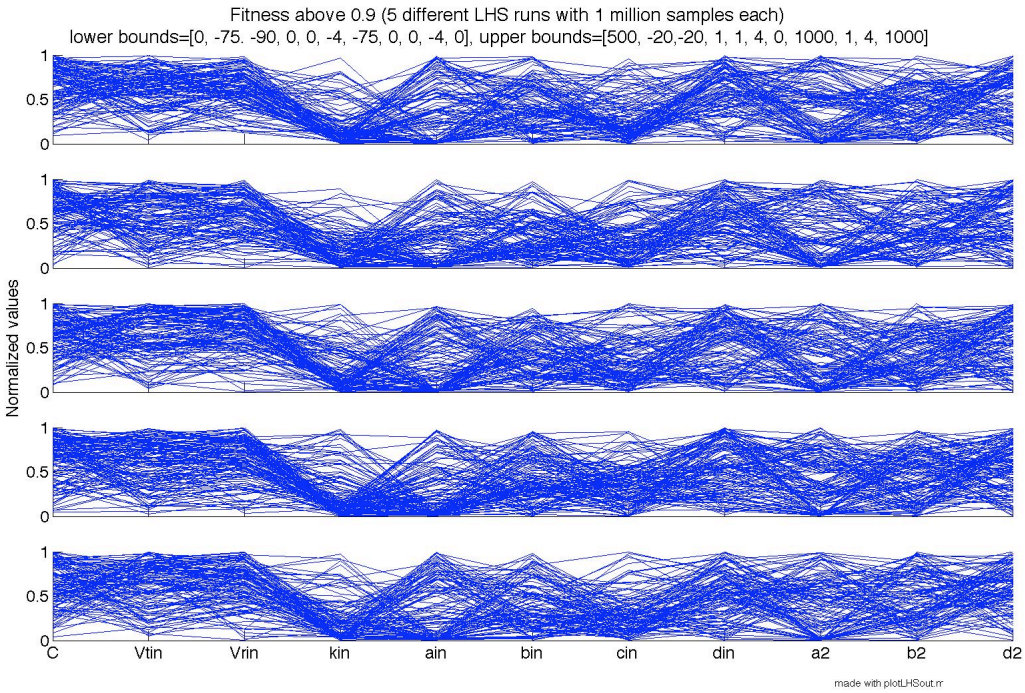


**Figure 6.2.** Normalized parameter values that yield a fit greater than 0.9 obtained by running 1 million LHS samples. Solid lines connect one set of parameter values.

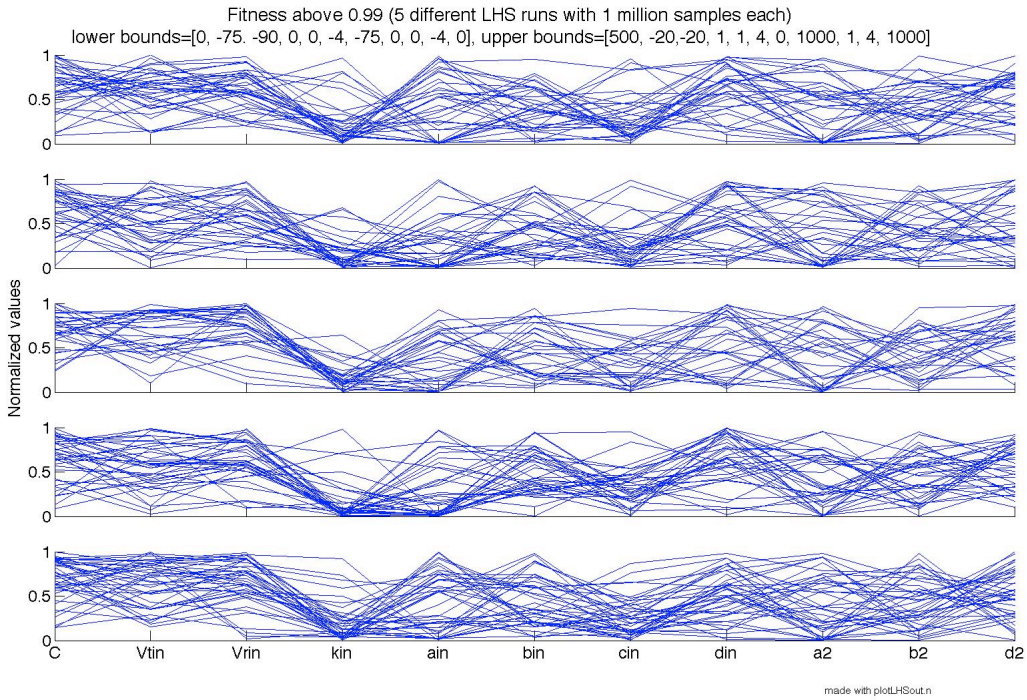


**Figure 6.3.** Normalized parameter values that yield a fit greater than 0.90 obtained by running 1 million LHS samples. Solid lines connect one set of parameter values.

To further explore this issue, we looked at how many of the parameters would yield a fitness larger than 0.99 (Figure 6. Again there are not any obvious patterns between the sets of parameters that yield a fit of 0.99.

Given the number of parameter sets we found yielding a sufficient fit for the test target trace, we moved onto a target trace consisting of real current clamp data. Unfortunately using the same method, we were unable to find any fits that yielded a fit greater than 0.9. There could be several reasons for this result. 1. real neural data is somewhat probabilistic and our fitness function does not take this into account. 2. The model is not sufficient to fit real biological data. To get to the heart of this matter, it would be ideal to compare several different current clamp voltage traces from the same neuron in order to access the statistical deviations firing patterns within neurons.

## Izhikevich Model Neuron

A computationally simple model that recreates many biological spiking patterns was created by Eugene Izhikevich [18].

The simplest equations for the model can be found in [18]. The equations we use here are an alteration on the original Izhikevich model which allow for more biologically realistic behavior. The equations we use follow:

$$v' = (kv_r v_t(k_1 + k_2 tanh(v_t)) - u_1 - u_2 + I_{syn}) + I_{comp})/C \tag{6.2}$$

$$u_1' = a_1(b_1 v_r - u_1) \tag{6.3}$$

$$u_2' = a_2(b_2 v_r - u_2) \tag{6.4}$$

Where

$$v_r = v - v_r \tag{6.5}$$

$$v_t = v - v_t \tag{6.6}$$

With after-spike resetting

$$if v \geq 30mV, then \begin{cases} v \leftarrow c \\ u_1 \leftarrow u_1 + d_1 u_2 \leftarrow u_2 + d_2 \end{cases} \tag{6.7}$$

# References


[1] B.M. Adams, W.J. Bohnhoff, K.R. Dalbey, J.P. Eddy, M.S. Eldred, D.M. Gay, K. Haskell, P.D. Hough, and L.P. Swiler. DAKOTA, a multilevel parallel Object-Oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual, November 2011.

[2] M. Baca, A. Schiess, D. Jelnek, C. James, and D. Partridge (UNM). 2009.

[3] Jason W. Bohland, Caizhi Wu, Helen Barbas, Hemant Bokil, Mihail Bota, Hans C. Breiter, Hollis T. Cline, John C. Doyle, Peter J. Freed, Ralph J. Greenspan, Suzanne N. Haber, Michael Hawrylycz, Daniel G. Herrera, Claus C. Hilgetag, Z. Josh Huang, Allan Jones, Edward G. Jones, Harvey J. Karten, David Kleinfeld, Rolf Kötter, Henry A. Lester, John M. Lin, Brett D. Mensh, Shawn Mikula, Jaak Panksepp, Joseph L. Price, Joseph Safdieh, Clifford B. Saper, Nicholas D. Schiff, Jeremy D. Schmahmann, Bruce W. Stillman1, Karel Svoboda, Larry W. Swanson, Arthur W. Toga, David C. Van Essen, James D. Watson, and Partha P. Mitra. A proposal for a coordinated effort for the determination of brainwide neuroanatomical connectivity in model organisms at a mesoscopic scale. *PLoS Computational Biology*, 5(3):e1000334.

[4] D.A. Butts, C. Weng, J. Jin, C.I. Yeh, N.A. Lesica, J.M. Alonso, and G.B. Stanley. Temporal precision in the neural code and the timescales of natural vision. *Nature*, 449(7158):92–95, 2007.

[5] Claudia Clopath, Lars Busing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nat Neurosci*, 13(3):344–352, 03 2010.

[6] Claudia Clopath and Wulfram Gerstner. Voltage and spike timing interact in stdp - a unified model. *Frontiers in Synaptic Neuroscience*, 2(00025), 2010.

[7] ed. Cowan M. *Synapses*. Johns-Hopkins, Baltimore, Maryland, U.S.A., 2001.

[8] P. Dayan and LF Abbott. Theorethical neuroscience, 2001.

[9] Peter Dayan and Abbott L. F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, Cambridge Massachusetts, 2001.

[10] Shaul Druckmann, Yoav Banitt, Albert Gidon, Felix Schrmann, Henry Markram, and Idan Segev. A novel multiple objective optimization framework for constraining Conductance-Based neuron models by experimental data. *Frontiers in Neuroscience*, 1(1):7–18, 2007. PMID: 18982116 PMCID: 2570085.

[11] Alexander M. Duda and Stephen E. Levinson. Nonlinear dynamical multi-scale model of associative memory. In *Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications*, ICMLA '10, pages 867–872, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Alexander M. Duda and Stephen E. Levinson. Complex networks of spiking neurons: Collective behavior characterization. *Proceedings of the Eighth International Conference on Complex Systems*, pages 1627–1629, 2011.

[13] Howard Eichenbaum and Neal Cohen. *From Conditioning to Conscious Recollection: Memory Systems of the Brain*. Oxford University Press, New York, 2002.

[14] Brown E.N., Kass R.E, and P.P. Mitra. Multiple neural spike train data anaylsis: state-of-the-art and future challenges. *Nature Neuroscience*, doi: 10.1038/nn.1228, 2004.

[15] B. Hille. *Ion Channels and Excitable Membranes*. Sinauer Associates, Inc., Sunderland, Massachusetts, U.S.A., 2001.

[16] Michael Hines. Exp2Syn Comments, 2008. [Online]. http://www.neuron.yale.edu/hg/neuron/ nrn/file/1501e76f9aa2/src/nrnoc/exp2syn.mod.

[17] R. L Iman, J. M Davenport, and D. K Zeigler. *Latin hypercube sampling (program user's guide). [LHC, in FORTRAN]*. 1980.

[18] E. M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):15691572, 2003.

[19] Eric R. Keiter, Scott A. Hutchinson, Robert J. Hoekstra, Lon J. Waters, and Thomas V. Russo. Xyce parallel electronic simulator design: Mathematical formulation, version 2.0. Technical Report SAND2004-2283, Sandia National Laboratories, Albuquerque, NM, June 2004.

[20] Eric R. Keiter, Ting Mei, Thomas V. Russo, Eric L. Rankin, Roger P. Pawlowski, Richard L. Schiek, Keith R. Santarelli, Todd S. Coffey, Heidi K. Thornquist, Jason C. Verley, Christina E. Warrender, and Deborah A. Fixel. Xyce parallel electronic simulator: User's guide, version 5.3. Technical Report SAND2012-4805, Sandia National Laboratories, Albuquerque, NM, 2012.

[21] Christof Koch. *Biophysics of Computation: Information Processing in Single Neurons*. 1998.

[22] Thomas Kreuz. Measures of spike train synchrony. *Scholarpedia*, 6(10):11934, 2011.

[23] Z. F Mainen and T. J Sejnowski. Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature*, 382(6589):363366, 1996.

[24] NEURON. Exp2Syn Description. [Online]. http://www.neuron.yale.edu/neuron/static/docs/help/neuron/neuron/mech.html#Exp2Syn.

[25] Thomas Nowotny, Attila Szcs, Rafael Levi, and Allen I Selverston. Models wagging the dog: are circuits constructed with disparate parameters? *Neural Computation*, 19(8):1985–2003, August 2007. PMID: 17571936.

[26] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:645–654, 1998.

[27] Brette R., Rudolph M., Carnevale T., Hines M., Beeman D., Bower J.M., Diesmann M., Goodman P.H., Harris F.C.J., Zirpe M., Natschläger T., Pecevski D., Ermentrout B., Djurfeldt M., Lansner A., Rochel O., Vieville T., Muller E., Davison A., El Boustani S., and Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *J. Computational Neurosci.*, 23(3), 2007.

[28] Cyrille Rossant, Dan F M Goodman, Jonathan Platkiewicz, and Romain Brette. Automatic fitting of spiking neuron models to electrophysiological recordings. *Frontiers in Neuroinformatics*, 4:2, 2010. PMID: 20224819.

[29] Jonathon Schlens. A tutorial on principal components analysis. 2005.

[30] S. Schreiber, J. M. Fellous, D. Whitmer, P. Tiesinga, and T. J. Sejnowski. A new correlation-based measure of spike timing reliability. *Neurocomputing*, 52:925931, 2003.

[31] Gordon M. Shepherd. *The Synaptic Organization of the Brain, 5th edition.* Oxford University Press, New York, 1998.

[32] R.B. Stein, E.R. Gossen, and K.E. Jones. Neuronal variability: noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397, 2005.

[33] S.J. Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel processing in neural systems*, pages 91–94, 1990.

[34] Traub and Miles. *Neuronal Networks of the Hippocampus.* Cambridge University Press, Cambridge, UK, 1991.

[35] W Van Geit, E De Schutter, and P Achard. Automated neuron model optimization techniques: a review. *Biological Cybernetics*, 99(4-5):241–251, November 2008. PMID: 19011918.

[36] Maxim Volgushev, Igor Kudryashov, Marina Chistiakova, Mikhail Mukovski, Johannes Niesmann, and Ulf T. Eysel. Probability of transmitter release at neocortical synapses at different temperatures. *Journal of Neurophysiology*, 92(1):212–220, 2004.

DISTRIBUTION:

| | | |
|---|---|---|
| | MS | , |
| | | , |
| 1 | MS 0899 | Technical Library, 9536 (electronic copy) |

Sandia National Laboratories