brains, minds media
ISSN: 1861-1680

Simulating *in vivo*-like Synaptic Input Patterns in Multicompartmental Models

Jeremy R. Edgerton

Department of Biology, Emory University, Atlanta, GA, USA; email: jedgert@emory.edu

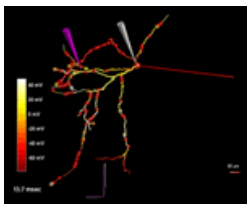
urn:nbn:de:0009-3-2256

Abstract. In the laboratory of Dr. Dieter Jaeger at Emory University, we use computer simulations to study how the biophysical properties of neurons—including their three-dimensional structure, passive membrane resistance and capacitance, and active membrane conductances generated by ion channels—affect the way that the neurons transfer synaptic inputs into the action potential streams that represent their output. Because our ultimate goal is to understand how neurons process and relay information in a living animal, we try to make our computer simulations as realistic as possible. As such, the computer models reflect the detailed morphology and all of the ion channels known to exist in the particular neuron types being simulated, and the model neurons are tested with synaptic input patterns that are intended to approximate the inputs that real neurons receive *in vivo*. The purpose of this workshop tutorial was to explain what we mean by '*in vivo*-like' synaptic input patterns, and how we introduce these input patterns into our computer simulations using the freely available GENESIS software package (http://www.genesis-sim.org/GENESIS). The presentation was divided into four sections: first, an explanation of what we are talking about when we refer to *in vivo*-like synaptic input patterns; second, a discussion of why computer simulations of realistic input patterns are useful; third, an overview of how these simulations are carried out in GENESIS; and fourth, a brief introduction to some strategies for analyzing the results of the simulations.

Keywords: GENESIS, synapse, compartmental, model, globus, pallidus, tutorial

Citation: Edgerton JR (2005). Simulating *in vivo*-like synaptic input patterns in multicompartmental models. Brains, Minds and Media, Vol. 1, bmm225 (urn:nbn:de:0009-3-2256).

Supplementary Material



Article Resources

GENESIS Resources

Datasheet

Synaptic input *in vivo*

Neurons in the mammalian central nervous system typically receive synaptic inputs from many other neurons. For example, each individual human cerebellar Purkinje neuron is estimated to receive more than 100,000 excitatory synaptic contacts from granule cells, and additional contacts from local circuit inhibitory interneurons and the powerfully excitatory climbing fiber (Ito 1984). Although Purkinje neurons are an extreme example of synaptic convergence, individual neurons that receive thousands of synaptic inputs are not unusual in the mammalian CNS. As pointed out by Pare and colleagues (1998), the average cortical pyramidal neuron receives about 10,000 synaptic inputs, and most of them are from other cortical neurons, which typically fire about 10 spikes per second in awake animals. Thus, the average cortical pyramidal neuron can be expected to receive roughly 100,000 synaptic events *every second*. Even for globus pallidus neurons, which have significantly less membrane surface area and receive most of their synaptic contacts from striatal medium sized spiny neurons (which have low average spike rates *in vivo*), the total number of synaptic events received per second is expected to number in the thousands.

The potential significance of this ongoing synaptic activity for a neuron can be illustrated with a hypothetical example. Imagine a neuron that receives 5,000 excitatory and 500 inhibitory synaptic contacts, each active at an average rate of 10 impulses per second. If each excitatory input had a unitary peak conductance of 500 pS, each inhibitory input a peak conductance of 2 nS, and physiologically realistic time constants, the average total synaptic conductance applied to the neuron could surpass 300 nS. If a conductance of this size were present in a single physical location (a 'point conductance'), every 10 millivolts of difference between the neuron's membrane potential and the net reversal potential of the combined synaptic conductance would generate 3 nA of synaptic current—which is a lot of current. Although this example exaggerates the actual effect of *in vivo* synaptic conductances by ignoring their spatial distribution, it illustrates the point that for most types of neurons, the total membrane conductance level *in vivo* (1) is quite high, and (2) fluctuates independently of the membrane potential because it is dominated by synaptic rather than intrinsic (i.e. voltage or ion gated) conductance mechanisms. The existence of high levels of ongoing, fluctuating synaptic input to neurons *in vivo* has now been clearly demonstrated using intracellular electrophysiological recordings from laboratory animals.

Why simulate *in vivo*-like synaptic activity?

There are many good reasons to include realistic patterns of synaptic input in computer simulations of neurons and neural circuits, but two of the reasons are particularly relevant to our work. The first is that we want to better understand how the morphological and biophysical properties of neurons contribute to their computational capabilities, and to address this issue we perform many electrophysiological experiments using the acute brain slice preparation, where we can manipulate biophysical properties of neurons in a controlled way. For example, we may examine the input-output properties of a neuron under 'normal' conditions, and then apply an agent that blocks a specific ion channel and assess how the loss of that channel type affects the cell's function. This is a standard experimental method, but it suffers from the problem that unlike neurons *in vivo*, most of which are under the influence of substantial, fluctuating net synaptic conductances at all times, neurons in acute brain slices and most other *in vitro* preparations

receive comparatively little synaptic input. Furthermore, what synaptic input does exist is often blocked during experiments in order to focus on the intrinsic biophysical properties of the neurons.

As an example, consider some experiments I did as a graduate student working with Dr. Peter H. Reinhart. The experiments involved blocking small conductance calcium-activated potassium channels (SK channels) in cerebellar Purkinje neurons in acute slices of rat brain. To avoid potential presynaptic effects of the SK channel blocker, fast synaptic transmission was blocked during the experiments. We observed that when SK channels were blocked, Purkinje neuron firing behavior changed dramatically (Edgerton and Reinhart 2003), and we concluded that in Purkinje neurons, SK channels are critical determinants of both the rate and regularity of spiking (Fig. 1). Our conclusions were well supported for Purkinje neurons *in vitro*; however, whether or not a blocker of SK channels would have similar effects *in vivo*, where the conductance due to SK channels would be only a small fraction of the net synaptic conductance, is a question that remains to be answered.

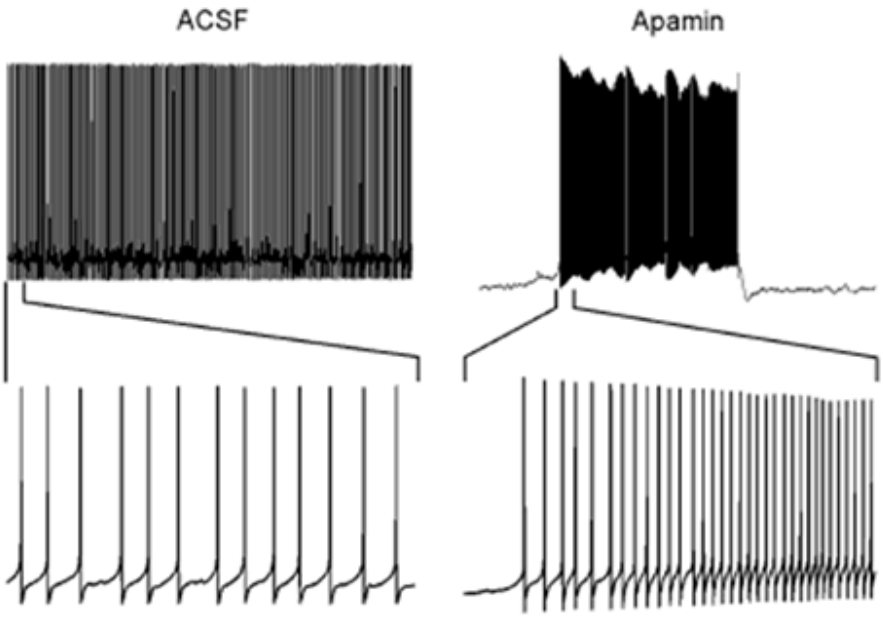


Figure 1: Small conductance K (Ca^{2+}) channels (Sk channels) regulate the firing rate of Purkinje neurons *in vitro* – but is this also true *in vivo*? Figure adapted from (Edgerton and Reinhart 2003)

Many similar experiments have been carried out in slice preparations or cultured neurons, and in most cases the extent to which the *in vitro* results would translate to the *in vivo* condition remains uncertain. However, dynamic clamp experiments investigating the role of SK channels in deep cerebellar nucleus neurons (Volker Gauck and Dieter Jaeger, unpublished data), and modeling studies on the influence of ion channels in the presence and absence of background synaptic input (Destexhe and Pare 1999), suggest that only some of the pharmacological effects observed *in vitro* will hold true in the presence of *in vivo* synaptic input, meaning that the *in vitro* data should be interpreted with some caution.

Part of the problem is that there is no simple way to generate large amounts of asynchronous, spatially distributed, ongoing synaptic input to a neuron *in vitro* without directly affecting the intrinsic properties of the neuron. For example, electrical stimulation of presynaptic axons could be used to drive synaptic inputs, but would typically result in synchronous activation of a restricted subset of inputs. Certain pharmacological manipulations, such as elevating the external potassium concentration or bath application of low concentrations of the potassium channel blocker 4-aminopyridine (4-AP) have been shown to induce high levels of spontaneous synaptic input in brain slice preparations (Avoli et al. 2002). But in most cases these manipulations will directly alter the neuron being studied, such that the 'control' condition no longer captures the normal electrophysiological properties of the cell.

This brings up an important point about modeling: in a multicompartmental model of a neuron, all of the synaptic inputs and intrinsic conductances can be controlled explicitly, allowing any pattern of synaptic input to be tested. The influence of a particular ion channel can then be assessed in simulations of both *in vitro* and *in vivo* conditions. Although the model will never be more than an approximation of the real neuron, a model that replicates the *in vitro* data accurately can make strong predictions about the *in vivo* condition because the only change is the addition of background synaptic input, which can be simulated with a reasonably high degree of precision.

The second reason why *in vivo*-like synaptic input patterns are particularly relevant to our work is that we want to understand what types of computations single neurons carry out. Since single neuron computations involve transforming complex, spatially distributed synaptic input trains into single output trains, it is obviously necessary to include such patterns of synaptic input in the models. Because every input can be controlled explicitly, modeling may offer the best way of beginning to address this difficult question.

How background synaptic input can be simulated using GENESIS

There are multiple ways to do just about anything in GENESIS, and synaptic input is no exception. But we find the method presented here to be convenient and effective. We use a combination of three standard GENESIS objects for our synapses, each of which is described in detail in the online GENESIS manual at the following link: http://www.genesis-sim.org/GENESIS/Hyperdoc/Manual.html

Steps involved in setting up the simulations:

1. **Cell morphology:** reconstruct a filled neuron, obtain a morphology file from a colleague or the web, or make a simplified morphology model.

2. **Passive parameters:** Rm, Cm, Ri

3. **Active conductances:** GENESIS `tabchannel` objects

4. **Synapse templates (AMPA, GABA, etc.):** gmax, τ_{rise}, τ_{fall}, E_{rev}

5. **Compartments:** list of those receiving input
6. **For every independent synapse** (in a loop):
    a. **Copy the synaptic conductance** from a template library to the compt
    b. **Create a timetable object** to determine when the synapse activates
    c. **Create a spikegen object** to communicate with the synapse

First, the characteristics of the individual synaptic conductances—peak size, kinetic properties, and reversal potential—are defined using the **synchan** object. Example code showing the creation of a `synchan` element called AMPA is shown below. The `synchan` object also allows the user to set a frequency of activation, but we don't use this mechanism, so the frequency field of all of our `synchan` elements is set to 0. We begin by making a single, template `synchan` element and placing it into a neutral element called library. Individual, independent `synchan` elements can then be copied from this template to different compartments of the model cell using the **copy** command, creating a population of synaptic inputs like a real neuron would receive. Since the different instances of the `synchan` are independent of one another, their properties can be set individually so that the population of synapses is heterogeneous. `Synchan` elements are conductances that must be taken into account by the simulation solver, so they are placed within compartments in the cell path just like `tabchannels` and other conductances.

GENESIS script defining AMPA-type conductances

```
//GENESIS script to define AMPA-type conductance
function make_AMPA_syn
 // make AMPA-type synapse
 if (!({exists AMPA}))
       create synchan AMPA
 end
 // assign specific synapse properties
 setfield AMPA Ek {E_AMPA}
 setfield AMPA tau1 {tauRise_AMPA}
 setfield AMPA tau2 {tauFall_AMPA}
 setfield AMPA gmax {G_AMPA}
 setfield AMPA frequency 0
end

//GENESIS script to create library template objects
//First, include my synapse and channel function files
include Syns.g
include Chans.g
```

```
//Check if library already exists
if (!{exists /library})
            create neutral /library
            disable /library
end

//Push library element, make conductance elements, pop library
pushe /library
 make_AMPA_syn
 make_G_Na
 make_G_K
pope
```

Second, the activation times of each synapse are determined using **timetable** objects. A separate `timetable` is needed for every synapse that is to be independent; if two `synchan` objects share the same `timetable`, they will be active in perfect synchrony. Each `timetable` is simply a list of times at which any connected synapses will be activated. The list of times can be a pseudorandom distribution with a specified mean and shape (exponential or gamma distribution), or it can be read from an ascii text file. `Timetables` are maintained in a separate location outside the cellpath, because they don't need to be handled by the solver. We create a separate neutral element called '`inputs`' to hold these objects.

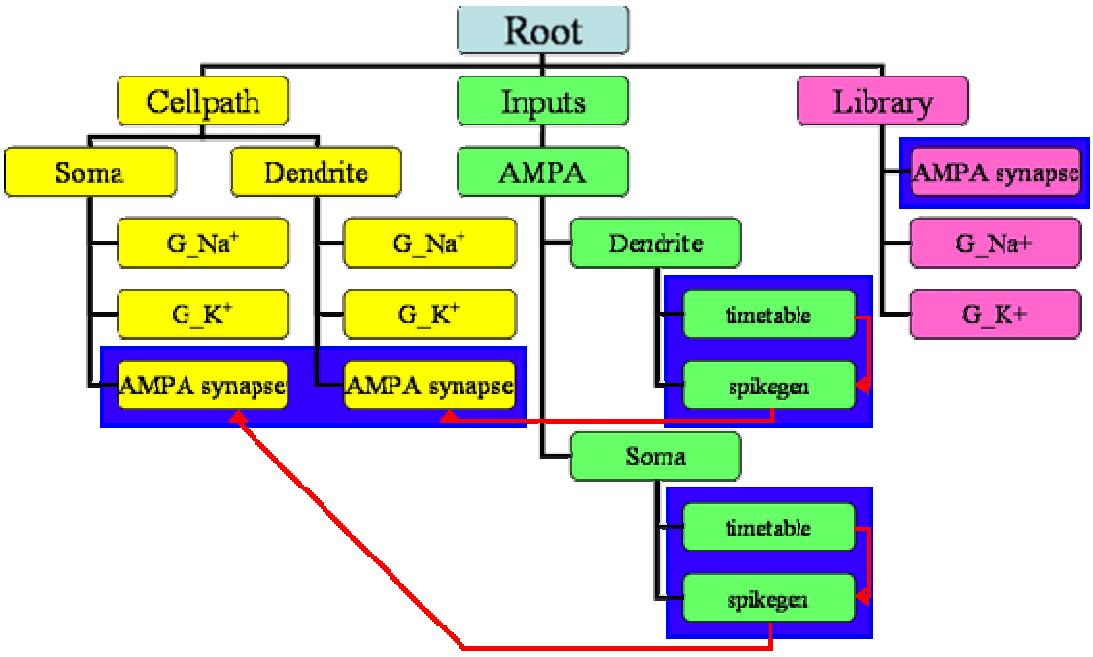


Figure 2: Element tree for a simulation of a 2-compartment model (with 'soma' and 'dendrite' compartments). Each colored box represents a GENESIS element. Yellow boxes represent the essentials of the model—its morphological and physiological features. Green boxes represent elements that control synaptic input timing. Pink boxes represent template objects that can be copied from the library and inserted into the model (at which point they would become yellow boxes).

Last, each `timetable` needs to be connected to the synapse(s) it will control, and to do this we use the **spikegen** object. A separate `spikegen` element is created for each `timetable` and placed in the same location. Whenever the simulation run time matches a time in the `timetable's` list, the timetable sends a message to the `spikegen` and the `spikegen` activates the `synchan`.

Pseudocode for all compartments receiving input:

```
//Using the same random seed means you get the same timetables next
time too.
randseed 78923456
//Loop: for each compartment that receives a synapse…
    1. copy the AMPA synapse from the library to the compartment
    2. addmsg: connect the synaptic conductance to the compartment
       with
       CHANNEL and VOLTAGE messages

    //set up the timetable
    1. create a unique timetable object for this compartment's AMPA
       synapse
    2. set timetable fields with setfield:
       method:
       1 = exponential distribution of intervals
       2 = gamma distribution of intervals
       3 = regular intervals
       4 = read times from ascii file
 meth_desc1: mean interval (= 1/rate)
 meth_desc2: refractory period (we use 0.005)
 meth_desc3: order of gamma distribution (we use 3)
    3. call /inputs/Excit/soma/timetable TABFILL

//set up spikegen
      create a unique spikegen object for this compartment's synapse
      set the spikegen fields with setfield
 output_amp: 1
 thresh 0.5
      //the spikegen tells the synapse when to activate based on the
        timetable
        addmsg from timetable to spikegen: type = INPUT, message =
            activation
        addmsg from the spikegen to the compartment's AMPA element,
            type = SPIKE

// Next loop iteration or END
```

Since separate `timetables` and `spikegens` are needed for each independent synapse, it is easiest to set these up in a **for** loop. Example code for such an operation can be found in the read_STN_syns file, listed below.

Example code to create a *while loop* in GENESIS:

```
/*
Script to add STN synaps es (AMPA synapses from subthalamic nucleus)
to GP model.
The compartments that re ceive the inputs must be listed in an ascii
text file.
*/
str STNfilename = "stn_syns.asc"
randseed 78923456
str stncompartment
//create input element t ree outside of the cell path
if (!{exists /inputs})
      create neutral /inputs
end
create neutral /inputs/STN
num_STN = 0
// Open the file with th e list of compartment names
// File MUST NOT have an y blank lines at the end, or function will
fail.
openfile {STNfilename} r
stncompartment = {readfile {STNfilename}}
//cycle through STN inpu t compartments
while (! {eof {STNfilename}})
      num_STN = {num_STN} + 1
      //Add AMPA synapse from library
      copy /library/AMPA {cellpath}/{stncompartment}/AMPA
      addmsg  {cellpath}/{stncompartment}/AMPA \
             {cellpath}/{stncompartment} CHANNEL Gk Ek
      addmsg  {cellpath}/{stncompartment} \
             {cellpath}/{stncompartment}/AMPA VOLTAGE Vm
      //set up timetable
      create neutral /inputs/STN/{stncompartment}
      create timetable /inputs/STN/{stncompartment}/timetable
      if ({STN_rate} > 0)
             setfield /inputs/STN/{stncompartment}/timetable \
                    maxtime {rundur} act_val 1.0 method 2     \
                    meth_desc1 {1/{STN_rate}} meth_desc2      \
                    0.005 meth_desc3 3
             call /inputs/STN/{stncompartment}/timetable TABFILL
      end
      //set up spikegen
      create spikegen /inputs/STN/{stncompartment}/spikegen
```

```
         setfield /inputs/STN/{stncompartment}/spikegen \
             output_amp 1 thresh 0.5
      //connect timetables to AMPA synapses
      if ({STN_rate} > 0)
             addmsg /inputs/STN/{stncompartme nt}/timetable    \
                 /inputs/STN/{s tncompartment}/spikegen INPUT  \
                 activation
             addmsg /inputs/STN/{stncompartme nt}/spikegen \
                 {cellpath}/{st ncompartment}/AMPA SPIKE   \
                 end

      // get next compartment name
      stncompartment = {readfile {STNfilename}}
end
closefile {STNfilename}
```

Analyzing the results

General points

1. Simulation movies are useful for multicompartmental model analysis. One of the principal difficulties in working with morphologically and biophysically realistic multicompartmental models is how to analyze the results of the simulations. For one thing, simply deciding which of the variables that are computed during the simulations you want to store and analyze is not always easy. If a model neuron has 500 compartments, there are already 500 different membrane potential values to consider, not to mention all of the currents, conductances, and internal ion concentrations. When starting out with a complex multicompartmental model, one way to begin getting a handle on the model's behavior is to save the membrane potential of every compartment in the model during a simulation, convert the membrane potential values to colors, and use the color coding to show how the membrane voltage moves around the model neuron's morphology over the course of a short simulation. This sort of simulation movie (Fig. 3) can be immensely helpful when explaining the model neuron's behavior to an audience, and can also demonstrate spatio-temporal characteristics of the model that are difficult or impossible to detect from the voltage of any single compartment—events such as dendritic initiation of action potentials, propagation of voltage signals through the dendritic tree, and failures of spikes to propagate past dendritic branch points, for example.

2. Examine how the conclusions depend on uncertain model assumptions. One must always bear in mind that no matter how well a model matches the available data, it is virtually certain to be missing important features of the real biological system and to contain some features that are not found in the real system. For this reason, the extent to which the conclusions of a modeling study depend upon uncertain model parameter settings must be explored. One strategy is to vary the relevant parameters of the model over their biologically plausible ranges and observe how the model's behavior changes as a function of each

parameter. A second strategy is to divide the possible solution space into distinct conceptual alternatives and make different versions of the model neuron for each alternative. As an example of the latter strategy, we have multicompartmental models of rat globus pallidus neurons and we are interested to see how dendritic action potential initiation influences how these neurons integrate synaptic inputs. We can therefore draw a conceptual distinction between model neurons that have dendritic action potentials and those that do not, and compare these two alternative model types in the way that they translate synaptic inputs into spiking output.

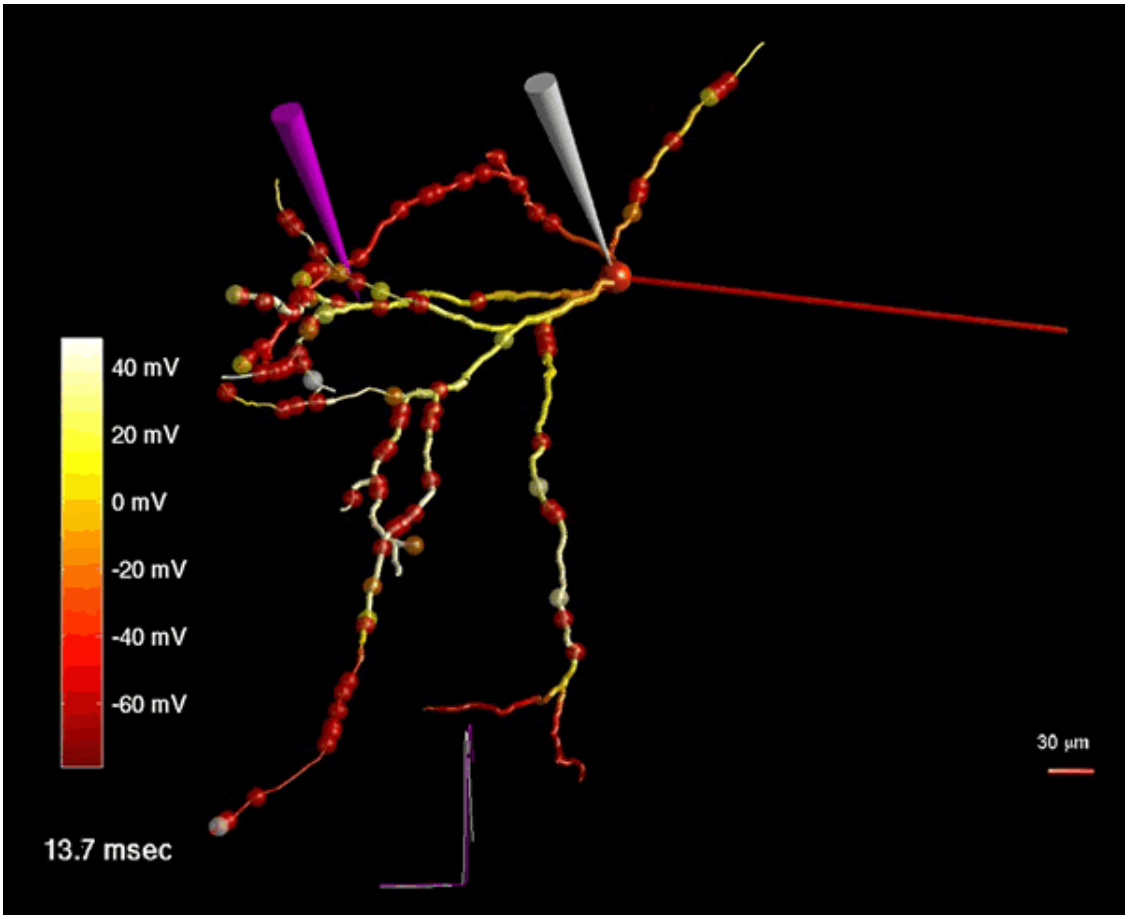


Figure 3: Screenshot of a simulation movie (view movie). The movie shows a model globus pallidus neuron being driven to spike by synaptic inputs. There are 100 excitatory inputs (represented as spheres on the dendrites) and 1024 inhibitory inputs (not shown), each activated randomly. Excitatory synapses activate when their color changes from red to white. The membrane potential of each compartment is presented in pseudocolor, showing that action potentials initiate in distal dendritic compartments and propagate to all regions of the model neuron. Near the end of the movie (approximately 94 msec on the simulation timer), a spike initiates in a thin dendrite but fails to propagate past the first branch point it reaches, suggesting that branchpoint failures may be important components of dendritic integration in the model.

Analysis of *in vivo*-like synaptic input patterns

With hundreds or thousands of independent synaptic inputs to a model neuron, analyzing synaptic integration becomes a difficult problem. We find that it is important to consider the results at two different levels.

1. A commonly used way of describing a model neuron's input-output function is the f-F curve: a plot of the model's output spike rate as a function of the mean rate of excitatory input. Separate f-F curves are then generated for different levels of average inhibitory input. If increasing inhibition simply shifts the f-F curve to the right, the model neuron is considered a linear integrator of inhibition and excitation. By contrast, a reduction in the slope of the f-F curve represents a divisive (or gain scaling) effect of inhibition. Gain changing operations are believed to be particularly important for the function of the nervous system (Koch 1999; Salinas and Thier 2000).

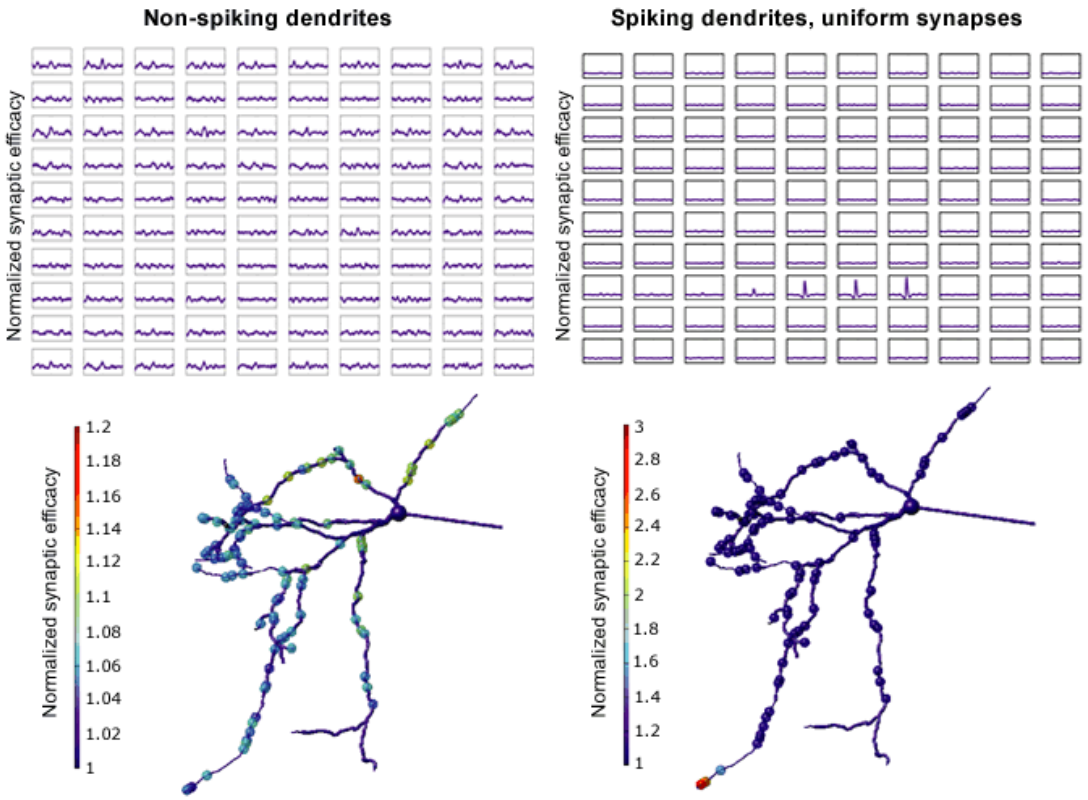


Figure 4: Quatifying synaptic efficacy using spike-triggered averages of synaptic conductances. At the top are shown normalized spike-triggered averages for the conductance of each of the 100 excitatory synapses in two versions of the globus pallidus neuron model: one with no dendritic sodium channels (left), and the other with uniform sodium conductance density across the soma and dendrites (right). The efficacy of each synapse was quantified as the peak amplitude of the synapse's spike-triggered average. The efficacy values for the synapses with respect to their location in the model are shown in the morphology plots. While the model with no dendritic sodium conductance showed little relationship between location and efficacy, the model with spiking dendrites was driven almost exclusively by a few synapses located near one particularly excitable region of the dendritic tree.

2. In addition to looking at the overall input-output properties of a model neuron, we have found that a careful analysis of how individual synapses control spiking is important for interpreting our results. We quantified synaptic efficacy in two different ways: first, for each synapse we calculated the probability that there would be an action potential at the soma of the model neuron within a specified time window after the activation of the synapse:

Efficacy = P (output spike | synaptic activation) / P (output spike)

Second, we made spike-triggered averages of the conductances of all synapses, using somatic action potentials as the trigger spikes, and took the peak of the conductance average for the synapse (figure 4).

Efficacy = peak of synapse's spike-triggered average conductance

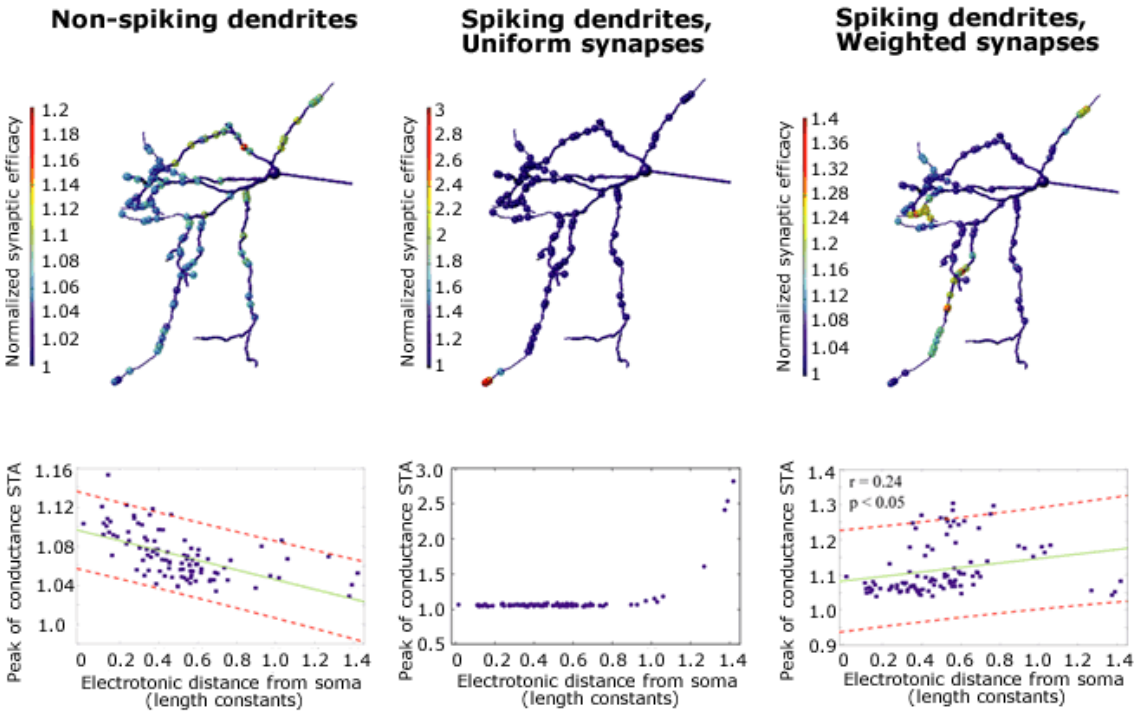


Figure. 5: When all excitatory synapses had the same unitary conductance, the model with no dendritic sodium conductance (left) was driven most effectively by synapses that were electrotonically near to the soma, as expected from passive decay of synaptic potentials over space. By contrast, the model with high dendritic sodium conductance (middle) showed the opposite relationship: those synapses that were electrotonically most distant from the soma were the most effective at driving spiking. Weight-scaling of synaptic amplitudes inversely to their measured efficacies produced a version of this model that responded to synapses in a more location-independent manner (right).

Both methods provide estimates of the likelihood that the model neuron will spike following a given synapse's activation, but the first method allows more jitter in the latency of the spike than the second method. The results of the two methods were quantitatively different but qualitatively similar, and showed

that variations in the biophysical properties of the model neuron could dramatically alter synaptic efficacy in a location-dependent way (figure 5).

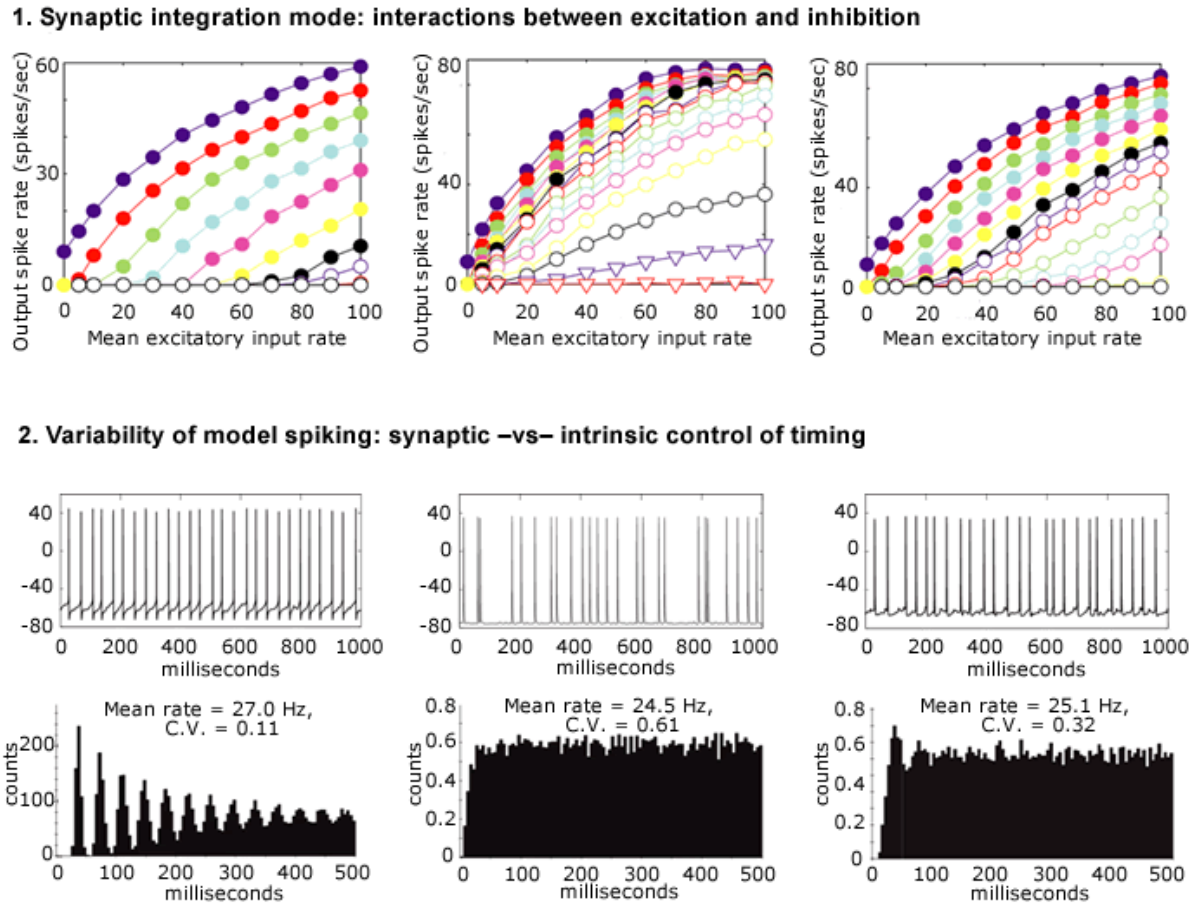


Figure 6: the three different versions of the model described in figure 5 have different input-output characteristics. The top plots show model output frequency as a function of excitatory input frequency at several different levels of inhibition (different colors) for each model, while the lower plots show example traces of the somatic voltage and autocorrelations for each model. The model with no dendritic sodium conductance (left) showed a primarily offset-shifting effect of inhibition and had a regular firing pattern even in the presence of irregular synaptic input. The model with the dendritic hotspot (middle) showed a substantial gain-scaling effect of inhibition, and highly irregular spike timing in the autocorrelation. The model with spiking dendrites but balanced synaptic efficacy (right) had properties that fell in between the other two models.

Furthermore, the way that the model integrated excitation and inhibition was more strongly influenced by the synaptic efficacies of the individual inputs than by the intrinsic biophysical properties of the neurons. When individual synapses were weighted to achieve approximately equal efficacies across the population in both the spiking dendrite model and the non-spiking dendrite model, the influence of spiking dendrites on synaptic integration became much less apparent (figure 6). If we had relied only on the overall f-F

curves of the models and not examined the contributions of each individual synapse, we would have failed to recognize this important point.

Conclusions

In summary, it is important to keep in mind that in the mammalian central nervous system, neurons typically receive large amounts of ongoing, fluctuating synaptic conductance input, and that this conductance can significantly alter the integrative properties of the neurons. As such, experiments done using *in vitro* preparations lacking synaptic input should be interpreted with some caution and supplemented with modeling where possible. Complex, *in vivo*-like patterns of synaptic input can easily be implemented in multicompartmental computer simulations with the GENESIS software package. The main points of this tutorial can be summarized as follows:

- Many independent synapses are needed to approximate synaptic input patterns *in vivo*, and can easily be added to a multicompartmental model using the synchan, timetable and spikegen objects in GENESIS as described above.

- This method is useful for making inferences about how *in vitro* results will apply to the *in vivo* system and for studying single neuron input-output functions.

- With complex multicompartmental models, analyzing the simulation data becomes one of the more difficult problems faced by the experimenter. Matlab software (The MathWorks, Natick, Massachusetts, USA), provides a convenient platform for customizing and automating the analysis of the data.

References

Avoli M, D'Antuono M, Louvel J, Kohling R, Biagini G, Pumain R, D'Arcangelo G, Tancredi V (2002). Network and pharmacological mechanisms leading to epileptiform synchronization in the limbic system in vitro. *Prog Neurobiol* **68**: 167-207.

Destexhe A, Pare D (1999). Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. *J Neurophysiol* **81**: 1531-1547.

Edgerton JR, Reinhart PH (2003). Distinct contributions of small and large conductance Ca2+-activated K+ channels to rat Purkinje neuron function. *J Physiol* **548**: 53-69.

Ito M (1984). *The Cerebellum and Neural Control*. Raven Press, New York.

Koch C (1999). *Biophysics of Computation*. Oxford University Press, New York.

Pare D, Shink E, Gaudreau H, Destexhe A, Lang EJ (1998). Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo. J Neurophysiol 79: 1460-70.

Salinas E, Thier P (2000). Gain modulation: a major computational principle of the central nervous system. Neuron 27: 15-21.