Journal of Neuroscience Methods 171 (2008) 309–315

Contents lists available at ScienceDirect

Journal of Neuroscience Methods

journal homepage: www.elsevier.com/locate/jneumeth

The virtual slice setup

William W. Lytton[a,b,*], Samuel A. Neymotin[b], Michael L. Hines[c]

[a] Departments of Physiology & Pharmacology, and Neurology, SUNY Downstate, Brooklyn, NY, USA
[b] Department of Biomedical Engineering, SUNY Downstate, Brooklyn, NY, USA
[c] Department of Computer Science, Yale University, New Haven, CT, USA

ARTICLE INFO

Article history:
Received 26 December 2007
Received in revised form 12 March 2008
Accepted 13 March 2008

Keywords:
Computer simulation
Computer modeling
Neuronal networks
Electrophysiology

ABSTRACT

In an effort to design a simulation environment that is more similar to that of neurophysiology, we introduce a virtual slice setup in the NEURON simulator. The virtual slice setup runs continuously and permits parameter changes, including changes to synaptic weights and time course and to intrinsic cell properties. The virtual slice setup permits shocks to be applied at chosen locations and activity to be sampled intra- or extracellularly from chosen locations. By default, a summed population display is shown during a run to indicate the level of activity and no states are saved. Simulations can run for hours of model time, therefore it is not practical to save all of the state variables. These, in any case, are primarily of interest at discrete times when experiments are being run: the simulation can be stopped momentarily at such times to save activity patterns. The virtual slice setup maintains an automated notebook showing shocks and parameter changes as well as user comments. We demonstrate how interaction with a continuously running simulation encourages experimental prototyping and can suggest additional dynamical features such as ligand wash-in and wash-out—alternatives to typical instantaneous parameter change. The virtual slice setup currently uses event-driven cells and runs at approximately 2 min/h on a laptop.

1. Introduction

Neural simulation has traditionally been practiced more like American rather than like international football: discrete simulations followed by regrouping and reorganization to prepare for the next attempt. This method necessarily distances the practice of simulation from *in vivo* neurophysiology, where experiments are performed on an active dynamical system which is never truly statistically stationary. It is more similar to performing experiments on a quiescent brain slice that requires repeated shocks to produce transient activity but again dissimilar to slice experiments on an active, firing network—an "epileptic" slice.

An alternative to the traditional simulation method has been called *reactive animation* (RA) by Efroni et al. (2005, 2007). The "reactive" refers to *reactive systems*, a term originating in engineering and now being introduced in biology. In engineering, reactive systems can be distinguished from transformational systems, which are designed to terminate in a distinct output. Reactive systems, by contrast, operate in real-time (*e.g.*, cruise controls and autopilots) and produce outputs that are state dependent. A particular output is only correct when it is produced at the correct time: the reactive system is in a continuous dynamical interplay with its environment. Seen in these terms, all biological systems are reactive systems. A biological system is continuously evolving, reacting to inputs that may also alter the system itself (plasticity). As with aircraft or process engineering, a reactive, real-time, ongoing biological system may be best served by use of reactive simulation.

The "animation" of reactive animation is obligatory rather than cosmetic: it provides the means for interaction with the running simulation, providing continuous or statistical evaluation of state variables and allowing control of system parameters. Like a video game, the quality of the simulation experience depends largely on the adherence to both the pragmatics and the dynamics of the system. As we will show, the experience of immediate interaction with the simulation can lead one to make improvements to this realism. However, neurophysiological simulation still suffers from a severe lack of detail compared to engineering systems or to simulation of other organ systems. In particular, there is a lack of detailed wiring information for brain areas, contrasting markedly with the relatively sophisticated knowledge of the single neuron.

Compared to experiment, simulation offers advantages of detailed observability and control. One has the ability to see all voltages and concentrations and to manipulate any neurotransmitter or ion channel at will. Indeed, one of the difficult problems in designing an RA simulator is adapting the graphical environment to the user, showing the user necessary information for a particular

* Corresponding author at: 450 Clarkson Avenue, Box 31, Brooklyn, NY 11203-2098, USA. Tel.: +1 718 270 6789; fax: +1 815 642 4019.
E-mail address: billl@neurosim.downstate.edu (W.W. Lytton).

0165-0270/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2008.03.005

experiment without overwhelming him with extraneous data or multiple control panels.

Although the formalized notions of RA are relatively new to biology, the idea of interactive simulation in neurophysiology dates back at least to P. Rowat's "Preparation" simulator. This lobster stomatogastric ganglion simulator was developed in the late 1980s, only about 5 years after the development of standalone graphical workstations made sophisticated graphics readily available (Rowat and Selverston, 1993). More recently, M. Hereld and collaborators have been advancing the idea of interactive simulations running on large parallel supercomputers in continuous communication with a front-end graphical workstation (Hereld et al., 2007). The virtual slice (VS) setup developed here has the advantage of being fairly large (expandable to about 1×10^5 neurons on a standard workstation) without requiring a supercomputer. Here we illustrate a 2700 cell simulation which runs at approximately 2 model minutes/hour on a laptop. This simulation rate makes it easy to run ion channel and synaptic blockade experiments over periods of several seconds of simulated time.

2. Materials and methods

The techniques and simulations described here are implemented in the NEURON simulator (web site, 2007; Carnevale and Hines, 2006) using a rule-based artificial cell mechanism (Lytton and Hines, 2004; Lytton and Stewart, 2005, 2006). This neuron model is a fast event-driven unit that was designed with several of the attributes of biological neurons, including adaptation, bursting, depolarization blockade, Mg^{++}-sensitive NMDA conductance, anode-break depolarization, and others.

The unit has 5 state variables: 4 for inputs— V_i for AMPA, NMDA, GABAA, GABAB (the acronyms refer here to the dynamics of the associated receptors and not to the chemicals), and 1 intrinsic state variable— V_{AHP} (after hyperpolarization following a spike). State variables are only updated when an event, external or internal, is received. External events arrive from other neurons. Internal events indicate an internal state update: *e.g.*, the end of the refractory period. External events produce a step increment in associated synaptic state variables. A spike produces an increment in V_{AHP}. State variables are updated according to a weight parameter W_i. This weight is multiplied by a driving force to produce a step voltage increment $S : S_i = W_i \times ((E_i - V_i)/E_i)$. W_{NMDA} is scaled by a function of postsynaptic voltage and Mg^{++} concentration. W_i is a unitless weight, not a conductance. Steps are unidirectional—*i.e.*, reversal of synaptic current is not simulated. S_i instantaneously updates the associated state variable: $V_i += S_i$. Except at a step, V_i decays with time constant τ_i. Because state variables decay exponentially, they can be updated analytically at event times. State variables are summed with resting membrane potential (RMP) to arrive at a final membrane voltage $V_m = \sum_i V_i$ for $i =$ {AMPA, NMDA, GABAA, GABAB, AHP}. V_m is compared to threshold to determine if firing takes place (comparable to integrate and fire).

Additional parameters and check points are used to provide varying refractory period, bursting, depolarization blockade and other features. Speed-up is obtained through use of table look-up to avoid run-time calculation of exponential decays and other response wave forms.

The network used here is an epileptic network using the event-driven cell type described with three different parameterizations reflecting three cell types: (1) a small but more excitable excitatory D (driver) population of 200 cells, that played the role of a driver of activity; (2) a larger excitatory population E (expressor) of 2000 cells, and (3) an inhibitory population I, consisting of 500 cells (Lytton and Omurtag, 2007; Lytton et al., in press). Connectivity was based on a random (Renyi-Erdös) directed graph with low connection densities:

FROM ⇒	D	E	I
TO ⇓ D	1%	0.5%	1%
E	1%	0.5%	10%
I	1%	1%	1%

Synaptic weights were assigned randomly with a normal distribution around the parameter central point within a narrow range. Synaptic delays were chosen randomly from a uniform distribution within a range of 2-3 ms. Individual cell parameters were chosen from distributions based around central values for intrinsic parameters, each set differently for the 3 different cell populations. Approximate maximal firing rates were 150, 100, and 330 Hz (for D, E and I cell types respectively), determined dynamically by refractory periods, afterhyperpolarization weights and afterhyperpolarization decay constants.

The full simulation described in this paper is available in runnable open-source form at the ModelDB web site (Hines and Carnevale, 2004; web site, 2007). The network is self-sustaining, running continuously without external input and with no intrinsic activity in any cells. Though the sparse random network used is nominally based on values derived from area CA3 of hippocampus, we are not in the present instance attempting to model specific slice phenomenology.

Simulations were run on an IBM X60 laptop with dual core 1.83 GHz processors and 2 GB memory, demonstrating simulation feasibility with a small-sized machine. Substantially larger simulations can be run on compute-serve workstations as will be noted below. This is in part due to savings in computer memory space using the recently developed JitCon (just-in-time connectivity) mechanism (Lytton et al., in press).

3. Results

3.1. Using the virtual slice setup

The default display for the VS consists of only two windows, though with multiple recording sites and parameter panels the display can quickly grow to encompass many tens of panels. Because the program is profligate in its use of windows, they are grouped so as to allow sets of them to be closed or reopened in concert. The two default windows are population activity display and the control panel (Fig. 1). The activity display (left) shows the total number of spikes occurring over a customizable integration period (default 10 ms). In the present instance, the graphic suggests that approximately 1700–1900 cells are active out of the 2700 cells in the simulation. This is therefore a highly active simulation with an average firing frequency of about 70 Hz. Since the integration period here is about twice the length of the average refractory period for the individual cell, this spike measure may slightly overestimate the number of active cells due to the rare occurrence of doublet firing during the period. We have used the simple spike count as the standard view, rather than providing a more sophisticated calculation of field potential that would take longer to calculate. We refer to the integrated spike count as a "field" below as a shorthand. As with other interactive computer systems, it's often necessary to opt for speed over detail in order to allow the user to work with the system in real time.

By default, 1 s of simulation time is shown as a continuous sweep (customizable parameter: field duration). We have run simulations for up to several hours of model time. Saving all the data from one of

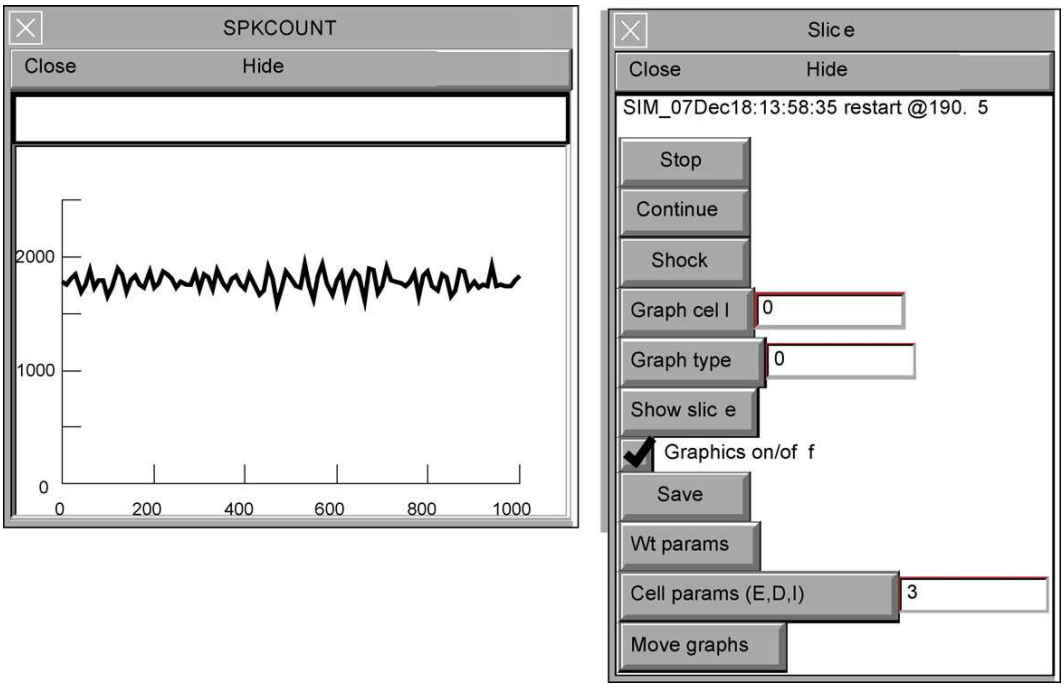

Fig. 1. Main display and control panels for VS (*x*-axis: ms, *y*-axis: spikes/10 ms).

these simulations would require ~290 MB/model hour/trace at our default 10 kHz sub-sampling. This would also entail frequent writing to disk which would again slow the simulation process. Saving all of the voltage traces from a simulation of this size would require ~780 GB/model hour and saving all state variables ~3.9 TB/model hour. Instead of continuous saving, the user has the option of saving desired fields, voltages or state variables on request when a phenomenon of interest is noted or a particular manipulation has been made.

The control panel (Fig. 1 right) has buttons that are largely self-explanatory. The top line is a system message that generally reflects simulation status, in the present case giving the simulation name (a customizable string based by default on local time when the simulation was started) and indicating that the simulation was restarted at 190.5 s in simulation time. The simulation can be stopped at any time to permit assessment and to allow further analysis from the command line. As with the 10 ms graphical integration time, the buttons are only queried every 10 ms by default. This could be readily decreased to a 1 ms interval when using a fast CPU. The *Continue* button then restarts the simulation. The third button on the control panel, *Shock*, provides instantaneous activation of a percent of the population in the current simulation. Stimulation parameters are customized in a separate panel (not shown).

As described in Section 2, the simulation used here has three cell types: a primary excitatory cell considered an *Expressor* of activity (E cells), a smaller population of higher-excitability cells considered activity *Drivers* (D cells) and a population of inhibitory cells (I cells) (Lytton and Omurtag, 2007). A single cell from any of these populations can be displayed by entering the cell number and pressing *Graph cell* (Fig. 2). Similarly, activity in the population of a given cell type can be displayed with *Graph* type. Note that the cell type is represented by a symbolic parameter which resolves to an index number: in the present case the E cells are represented by *EX* (E in Neuron is reserved for the base of the natural logarithm). *EX* resolves to 3, so that either "EX" or "3" could be entered next to the button to display the E-cell field. All graphics can be toggled off (*Graphics on/off*) to allow the simulation to move forward more rapidly, which might be desirable when waiting for a steady state to be reached.

The *Show slice* button provides clickable views of the slice that permit individual cells or groups of cells to be interactively chosen for display. It launches multiple views (currently 3) with cells located according to their defined *x,y* coordinate positions (Fig. 3). In the present instance, the cells have been randomly placed at locations within an ellipse; each cell is represented by a black dot. The multiple views are available to avoid the crowding and confusion of superimposed symbols. Each view has different mouse functionality: *e.g.*, the ability to position a shock, to launch single cell or field graphics or to launch various kinds of activity animations. In Fig. 3, the top view launches graphics. A mouse click on the top view places a colored cross over a cell symbol, *e.g.*, the red cross at the right, and launches a separate line graph with the voltage trace of that cell (here the cell trace in Fig. 2B). Similarly, a mouse drag-event colors in a set of cells within the defined circle and launches an identically-colored field trace comparable to the field traces in Fig. 2 A. The second view provides local field potential animation: it defines a circle of cells as in the first view but instead of launching a trace produces an animated red ellipse which grows and shrinks vertically with activity in the defined area. The third view provides a cell-based animation with individual squares showing the potential in each individual neuron.

When something interesting is seen on the screen, the *Save* button will launch a *Save dialog* (Fig. 4). This screen allows 1. entering a free-form comment; 2. changing the output file name stem (default is based on the simulation name and simulation time); 3. selection of one of several formats for saving. All currently displayed fields or cell traces will be saved. These can be saved either in a graphical or vector formats. Only the currently displayed traces can be saved to disk since other state variables are not being retained in memory. Fig. 2 was saved by selecting the checkbox for *Save to idraw* which saved directly to the editable postscript format used by Neuron. The final version of

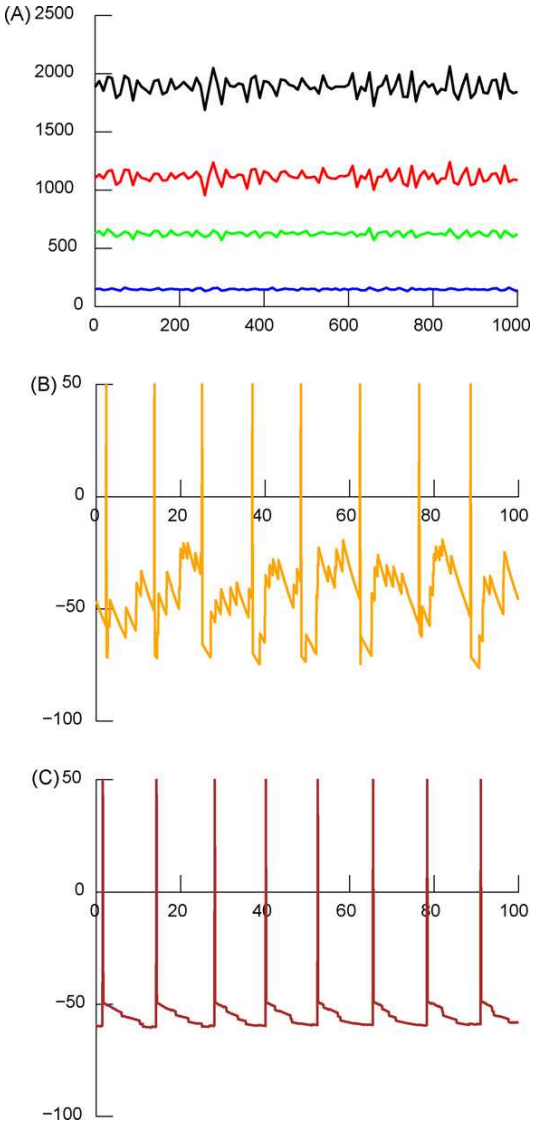

Fig. 2. "Field" (integrated spike count) and single cell voltage displays. All traces are color-coded in the original. (A) 4 simulated fields (spike counts) (from top to bottom: all cells E, I, D). We can observe that most of the activity variability comes from the E cells. (B) Voltage trace from an E cell. (C) Voltage trace from an I cell. Voltage in mV; time in ms.

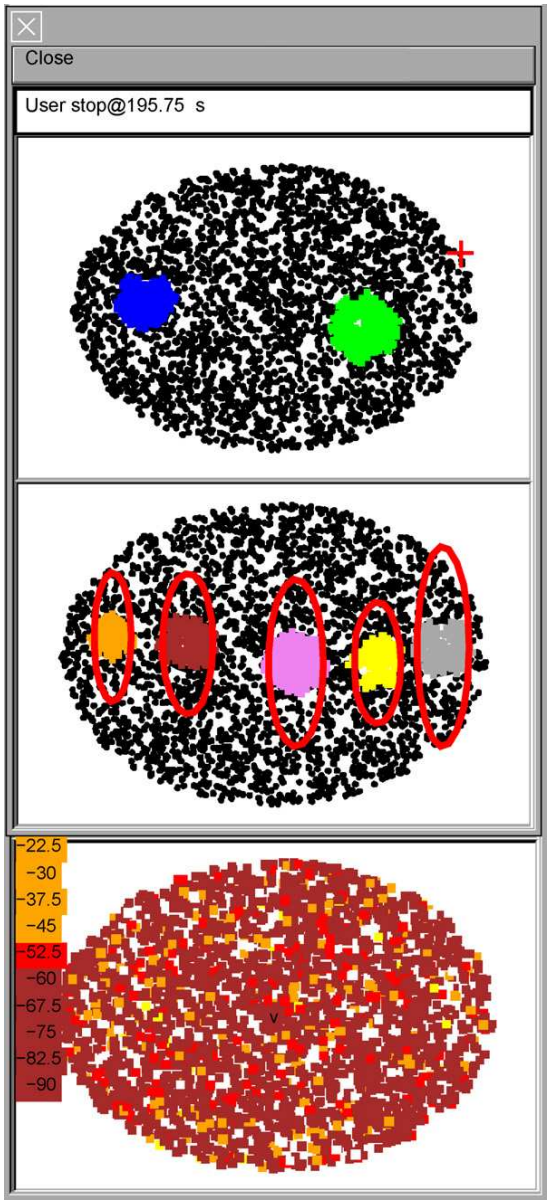

Fig. 3. Redundant views of the cells in slice permit placement of stimulating and recording electrodes. Top view launches traces such as those of Fig. 2; middle view is an animation where ellipse height shows activity in the colored population; bottom view is a cell-level animation of voltages.

Fig. 2 then required only alignment and labeling of the 3 panels.

When the VS is run a notebook, "notes.txt" by default, is maintained which gives the simulation name and any information regarding parameter changes, shocks or file saves. In the case of Fig. 2, the section of notebook reads:

```
* Sim start at 07Dec18:13:58:35EST

User stop@53.37

USER NOTE@t=96.86:For paper

Data written to data/SIM_07Dec18:13:58@96.86.id

User stop@109.57
```

Here the comment we entered was "for paper" and we have left the unwieldy full file name. The notes.txt file is user-editable text: additional notes can be added from an editor.

3.2. Parameter changes

For the present simulation, weight and intrinsic cell parameters have been made available for direct change via panels. Although additional parameters can be changed by stopping the simulation, changing a parameter in the interpreter and restarting, this interferes with one's dynamical sense of activity alteration. In any case, a newly designed parameter panel can be launched from the interpreter with a single function call. As with the *Graph type* button, the *Cell* params button requires entry of a cell type in either symbolic or numerical form (*e.g.*, typing either "EX" or "3" for the E-cells).

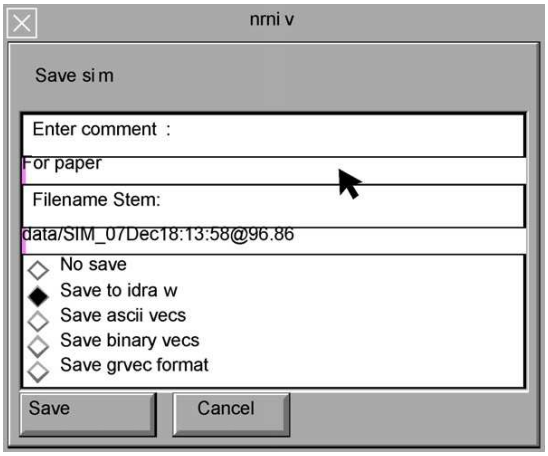

Fig. 4. Panel permits saving to a variety of file formats.

One can of course launch all 3 cell-parameter panels. Uniform or Gaussian variability of individual cell parameters around a central value is provided; this variability is itself parameterized and could be made interactively available in a panel. We do not currently envision a need to alter the parameters of individual cells directly rather than as part of a population.

Fig. 5 shows the current parameter panels for weights and cell parameters (here for the E cells). The weight parameters are separated by pre- and postsynaptic cell type and by synapse type (*i.e.*, AMPA, NMDA, $GABA_A$; $GABA_B$ is not present in this simulation). Some of the cell parameters are actually synaptic – the time constant of AMPA, NMDA or $GABA_A$ decay are attributes of the postsynaptic cell. Several points merit further explanation. First, with 3 cell types the weight panel is barely manageable in length. With more cell types, the scrollbar would need to be used. In practice, we anticipate that the panel would instead be modified to only display the weights of interest for a particular set of simulations. As noted above, panel contents are readily altered using function calls from the Neuron simulator interpreter. Second, we have here altered standard Neuron panel usage in one respect: changing a value and clicking the associated button *does not* immediately alter that parameter in the simulation. Instead a change to a weight parameter produces the following message at the top of the panel: "Press 'Change weights' button for effect". This additional button press is required in order to permit multiple parameter changes to be effected simultaneously in order to simulate the "dirty" (multiple binding site) effects of many drugs and endogenous ligands. For this reason, the various parameter panels can also be linked so that a single button press simultaneously produces changes in both weights and in intrinsic cell parameters. In practice, a particular simulation might use, *e.g.*, a single "Acetylcholine" button rather than making several alterations individually. Third, pressing the top button not only makes the weight changes but also documents the change in the notes.txt notebook, *e.g.*,

```
Weight changes @t=29080:  E->I:AMPA:0.01->0.05; E->I:NMDA:0.2->0.1;
D->E:AMPA:0.1->0.2.
```

Fourth, the values shown in the Weight Panel are unitless scaling factors. As with the cell parameters, the weights are randomized around a central value; rescaling shifts all of the associated weights without affecting their relative strengths. Fifth, the values shown in the cell parameter panel are in ms (*e.g.*, *tauAM* —the AMPA time constant and *refrac* —the refractory period) or mV (*e.g.*, RMP— resting membrane potential and VTH—the firing threshold). These parameters are specific to the neuron model parameterization being used here (see Section 2). The parameters would of course be quite different, and substantially more complicated, for a compartmental model. Currently, use of full multi-compartmental model at this scale is not practical on a single CPU but could be performed in the future using multi-threading or via linkage to a parallel supercomputer (see Section 5).

4. Advantages of continuous simulation

An advantage of the VS is that usage of this tool tends to suggest further improvements to allow the VS to conform still more closely to experimental physiology practice. This is best illustrated by the *Wash in* button near the top of the weight parameters panel (Fig. 5A) which was not originally included. The discrepancy with experimental practice was noticeable when clicking the *Change weights* button which effected instantaneous alterations in synaptic strengths. Such instantaneous effects are of course not possible in physiology where an exogenous ligand requires time to flow into the chamber, to diffuse into the tissue, to equilibrate at concentration, and to bind to receptors. We therefore added a *Wash in/out* button which provides linear increase or decrease in the changed parameters over the time period given on the panel (300 ms in Fig. 5A). This adds the simulation of experimental dynamics to that of physiological dynamics.

We found that this alteration in parameter-change timing could produce substantial and reproducible effects. Fig. 6 shows a 35% synaptic weight reduction from E to D cells. This synaptic strength reduction generally terminated ongoing activity when done instantaneously (9 out of 10 trials of different random networks). The same synaptic reduction produced no termination during parameter wash-in over 300 ms (10 trials each). An even greater weight reduction was possible without activity termination when using a longer wash-in time. Note that, as with physiological experiment, it was important to repeat the experiment several times with several different VSs (different randomizations of connectivity and intrinsic parameters) to validate the findings. This need to replicate experiments, an apparent cost of VS verisimilitude, is in fact an advantage. In traditional simulation, it is tempting to accept a single-run result as "correct" since it is absolutely and endlessly replicable. This is reasonable for an intervention that takes place from a dynamical fixed point, *e.g.*, the stabilized resting membrane potential in a single cell simulation. However, this assumption is not adequate when dealing with an evolving, reactive, dynamical system such as a firing single cell or a network.

5. Discussion

We have designed, developed and demonstrated an interactive neuronal network simulation with several attributes reminiscent of experimentation with an electrophysiological slice preparation: the ability to place electrodes to record either intracellularly or by selected population; a facility for placing stimulation electrodes at particular locations and stimulating with variable strength; the ability to wash-in and wash-out neuroactive ligands with the potential for multiple effects on synaptic strengths, synaptic dynamics and cellular firing patterns.

This VS represents a different way of interacting with the simulation, rather than a change in simulation design, biological fidelity, or underlying technology. However, the process of simulation interaction encourages rapid prototyping of novel experiments that would be laborious to do in a traditional simulation environment (and

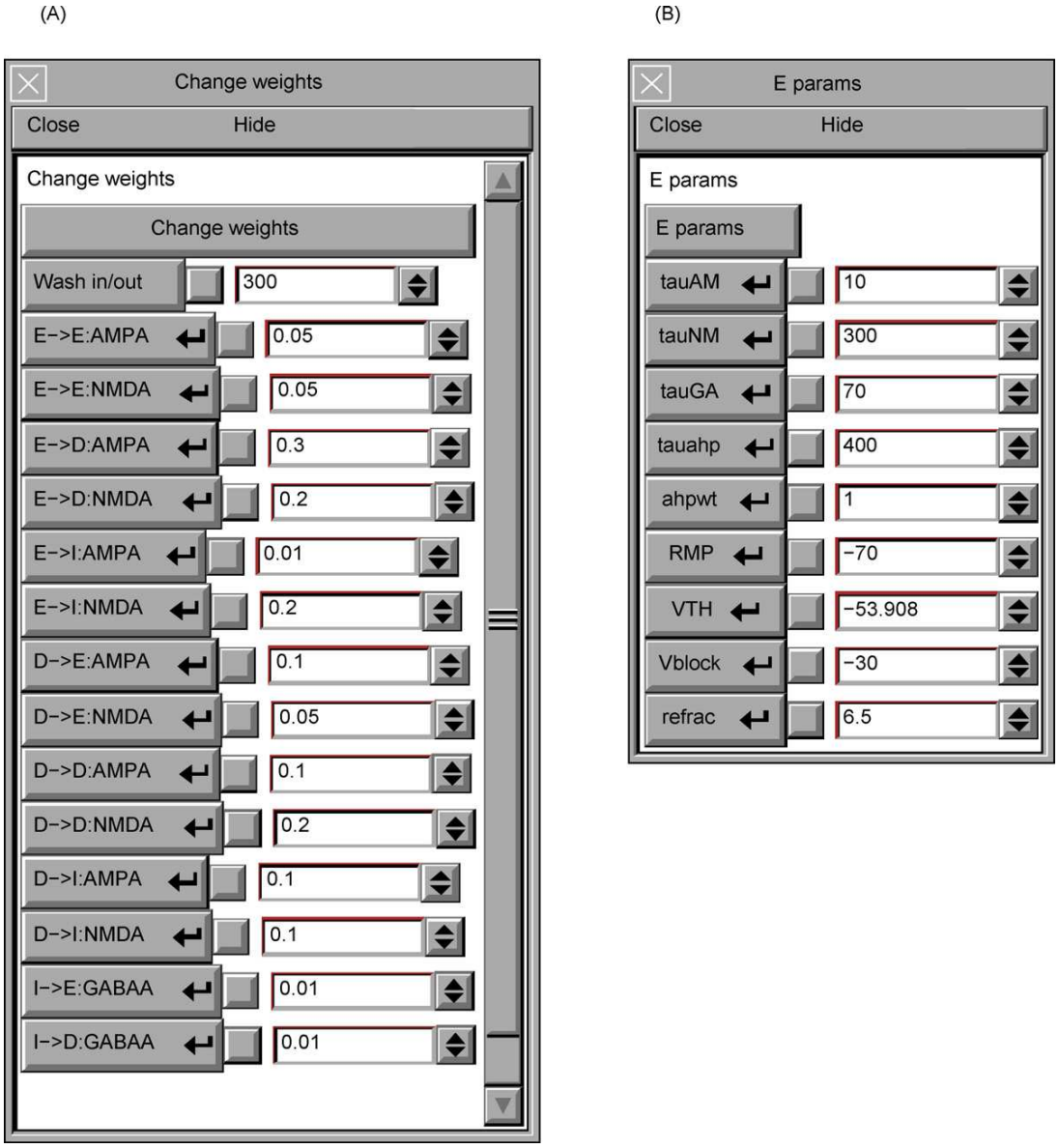

Fig. 5. Panel for changing (A) weight parameters; (B) intrinsic cell parameters for the Expressor cells.

far more laborious to do in a real slice). Additionally, this process directly suggested simulation enhancements to further establish concordance with a real slice. Above, we discussed wash-in as an example of such an enhancement. This example could be readily expanded by using a spatial as well as a temporal gradient of parameter change: drug in a slice is typically washed in from one end of the bath (acute slice penetration is likely far slower than flow rate; however, this effect might be relevant in cultured slices).

We have here illustrated VS basic usage with only the simplest evaluation and manipulation tools. In the future, we anticipate utilizing more sophisticated methods for both simulation evaluation and display. For example, we could calculate complex activity measures continuously in the background and provide status indicators or signal when certain dynamical conditions are met. An example would be responses to repeated single-shock test stimuli in a long-term potentiation/depression paradigm (or similarly, responses to complex spatially patterned tests in a slice "learning" paradigm). These stimuli would be given periodically and a simple histogram could display response strength. The responses would also be saved to the automated notebook. Still more complex data summaries would result from using continuous data-mining of the running simulation (Lytton and Stewart, 2007; Lytton, 2006) On the graphical level, high-dimensional displays could summarize several aspects of network activity – for example level of activity, spatial and temporal frequencies and phase shifts, level of depolarization blockade – simultaneously by utilizing dimensions such as color, symbol size, radial direction in addition to Cartesian coordinates and the ongoing dimension of time. Such displays would require substantial training to use effectively.

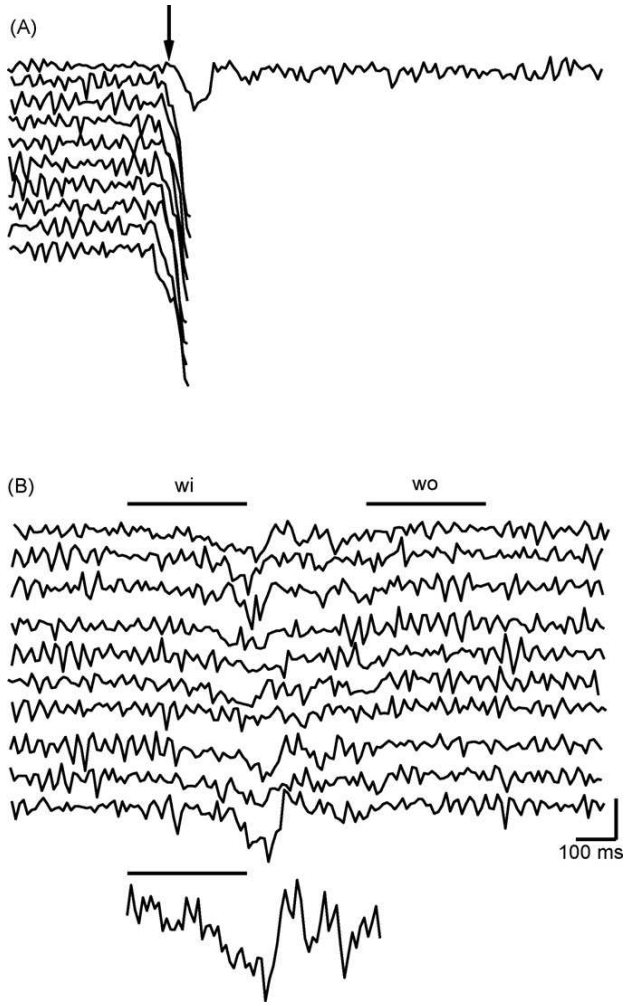

Fig. 6. (A) Instantaneous 35% reduction in E → D cell synaptic strength terminates most random networks (9 out of 10 networks with identical parameters but different randomized connectivity and individual cell properties). Note that activity in topmost trace does not terminate after synaptic strength reduction. (B) Wash in (*wi*) of same parameter change over 300 ms results in no termination over 10 other random networks, though activity drops transiently. Detail at bottom shows top trace at 2 ×. *wo* is washout period. Baseline for simulation ∼ 1880 spikes/10 ms; vertical scalebar 500 spikes/10 ms.

As suggested by several groups, the future of neurocomputing lies in supercomputing (Migliore et al., 2006; Hereld et al., 2007; Markram, 2006). The difficulty of providing VS graphical interface control of a multiprocessor supercomputer would lie in channeling the large volume of messages that flow among the many processors into a single pipe to allow access by a graphical workstation. This would be done by an extension of our current approach of using relatively low spatial and temporal sampling for data output. Another approach that would provide substantial speed-ups would be to make concurrent use of the multiple processors in mid-size shared-memory workstations through multi-threading. This form of parallelism is currently being added to Neuron. With multi-threading, the parallel computer is running both the graphics and the simulation and there will be no impediment to simulator communication. Using multi-threading, a 4 or 8 CPU workstation would be able to handle interactive network simulations of several thousand small (3–10 compartment) compartmental models or several million artificial cells or hybrid networks (Lytton and Hines, 2004).

The VS is freely available and will run "out of the box" for the network shown here. It is also readily customizable to utilize different networks. We introduce it in the hope that it will be utilized to look at different brain systems and neurobiological issues. This can be readily effected by those familiar with Neuron's languages (hoc and mod) and the Neuron environment. For others, the process of developing and wiring a different network design has not yet been automated or made user-friendly. This remains a task for the future. In the meantime, the authors would be happy to collaborate to create additional networks.

Acknowledgment

This research is supported by NIH (NS045612 and NS11613).

References

Carnevale NT, Hines ML. The NEURON book. New York: Cambridge; 2006.
Efroni S, Harel D, Cohen IR. Reactive animation: realistic modeling of complex dynamic systems. Computer 2005;38:38–47.
Efroni S, Harel D, Cohen IR. Emergent dynamics of thymocyte development and lineage determination. Plos Comput Biol 2007;3:e13.
Hereld M, Stevens RL, Lee HC, van Drongelen W. Framework for interactive million-neuron simulation. J Clin Neurophysiol 2007;24:189–96.
Hines ML, Carnevale NT. Discrete event simulation in the neuron environment. Neurocomputing 2004;58:1117–22.
Lytton WW, Hines M. Hybrid neural networks–combining abstract and realistic neural units. IEEE Eng Med Biol Soc Proc 2004;6:3996–8.
Lytton WW, Stewart M. A rule-based firing model for neural networks. Int J Bioelectromagn 2005;7:47–50.
Lytton WW. Neural query system: data-mining from within the neuron simulator. Neuroinformatics 2006;4:163–76.
Lytton WW, Stewart M. Rule-based firing for network simulations. Neurocomputing 2006;69:1160–4.
Lytton WW, Stewart M. Data-mining through simulation: introduction to the neural query system. In: Crasto C, editor. Neuroinformatics. New York: Humana Press; 2007.
Lytton WW, Omurtag A. Tonic–clonic transitions in computer simulation. J Clin Neurophys 2007;24:175–81.
Lytton WW, Omurtag A, Neymotin SA, Hines ML. Just in time connectivity for large spiking networks. Neural Comput, in press.
Lytton WW, Stewart M, Hines ML. Simulation of large networks: technique and progress. In: Soltesz I, Staley K, editors. Computational neuroscience in epilepsy, 1. Amsterdam: Elsevier; 2008. p. 3–17.
Markram H. The blue brain project. Nat Rev Neurosci 2006;7:153–60.
Migliore M, Cannia C, Lytton WW, Hines ML. Parallel network simulations with neuron. J Comput Neurosci 2006;6:119–29.
Rowat PF, Selverston AI. Modeling the gastric mill central pattern generator of the lobster with a relaxation-oscillator network. J Neurophysiol 1993;70:1030–53.
web site ModelDB. *ModelDB*. http://senselab.med.yale.edu/senselab/ModelDB, retrieved Apr 26, 2007.
web site Neuron. *NEURON for empirically-based simulations of neurons and networks of neurons*. http://www.neuron.yale.edu, retrieved Apr 26, 2007.