# No event left behind: adapting variable timestep integration to networks


## William Lytton and Michael Hines



**Abstract**

Realistic neural networks involve the co-existence of stiff, coupled, continuous differential equations arising from the integrations of individual neurons, with the discrete events with delays used for modeling synaptic connections. We present here an integration method, the local variable time step method (*lvardt*) that utilizes separate variable step integrators for individual neurons in the network. Cells which are undergoing excitation tend to have small time steps and cells which are at rest with little synaptic input tend to have large time steps. A synaptic input to a cell causes re-initialization of only that cell's integrator without affecting the integration of other cells.


**Summary**

We have previously demonstrated some advantages of the global variable time step integrators CVODE and CVODES (Cohen 1994) over traditional fixed step methods such as Euler, Runge-Kutta or Crank-Nicholson for simulating single cells.(Hines & Carnevale 2001) The major advantage was due to the fact that neuron activity features spikes, requiring short times steps, followed by interspike intervals, which allow long time steps. The associated speed-up in the single-cell integration realm do not extend to simulation of networks however. A major reason is that the global timestep is governed by the fastest changing state-variable. In an active network, some cell is usually firing, requiring a small timestep for the entire network. Another, related reason, is that synaptic events generally cause a discontinuity in a parameter or state-variable. This requires a re-initialization as the integrator must start again with a new initial-condition problem. In a network simulation, this means re-initialization of the entire network due to a single state variable change in one cell. With re-initialization, the integrator is working without any past history. Hence the first step can only be first-order accurate and must be very short.

We demonstrate here that the poor network performance of the global variable timestep method can be overcome by giving each neuron in the system an independent variable time-step integrator. Thus, a single cell's individual integrator uses a large $dt$ when a neuron is quiescent or changing slowly, even though activity in other neurons in the network may cause those other integrators to proceed forward with many steps (small $dt$). When a cell receives a synaptic event, only that cell's integrator has to be re-initialized.

The critical problem in the implementation of the local time-step method (*lvardt*) is to ensure that when an event arrives at a cell at time $t_e$ that all the state-variables for the receiving cell are also at time $t_e$. This requires coordinating individual integrators so that one cell does not get so far ahead that it cannot receive a synaptic signal from another cell.

The techniques and simulations described here are implemented in the NEURON simulator (www.neuron.yale.edu). The simulator provides several global time step

integration schemes. For global fixed step methods (Hines, 1984; Hines and Carnevale, 1997) one can select either a first order backward euler integration scheme that is numerically stable for all reasonable neural models or the second order Crank-Nicholson method. There are two global variable step methods, both part of the Livermore SUNDIALS package http://www.llnl.gov/CASC/sundials/, CVODES and IDA. Simulations were run on 2.40 and 2.80 GHz Intel CPUs under Linux and Solaris operating systems.

The global variable time-step method has advantages in any simulation where periods of intense simulation activity alternate with periods where state-variables remain relatively constant for a period of time. In general this situation is more likely to occur during simulation of a single cell rather than a network. The larger the network simulation, the greater the likelihood that a neuron somewhere in the network is showing spike activity. Using a global variable time-step method, this activity slows the entire simulation to tend to that one neuron's integration needs. Neurons that are not active will be integrated with an unnecessarily short time-step.

These considerations suggested the development of the local variable time-step method (*lvardt*) to integrate a network piece-meal, providing short time-step integration for active neurons and long time-step integration for inactive neurons, while maintaining consistent integration accuracy throughout. Neurons that fire at different times get their state-variables calculated at different times and, more importantly, different intervals (Fig. 1). Using the global method (top graph), the two cells have their trajectories calculated at the same times. With *lvardt* (bottom graph), integration points are independent. This is most obvious at the beginning of the simulation, where the cell that fires first (at right on schematic; vertical lines on graph) has only 2 integration points while the other cell (bottom; crosses) has 12 integration points. Where the trajectories cross, the first-spike cell integrator is getting busy while the second-spike cell integrator is using longer *dt*s. At the peak of the first spike, both cells are being updated frequently since the second-spike cell is coincidentally approaching threshold. At the peak of the second spike, however, the first-spike cell is on the falling phase of its spike and has far fewer integration points.

State-variables change quickly near threshold and at the peak of the action potential. At these times, the integrators use short timesteps to accurately follow the trajectory through state space. At other times, much larger *dt*'s can be used to achieve the same accuracy for the more slowly varying state-variables. Using *lvardt*, a neuron that is inactive does not waste CPU time. Overall performance evaluation for this simple simulation demonstrates that the global method integrates its 8 state-variables (the 4 Hodgkin-Huxley variables $m, h, n, V$ for each cell) 177 times, while the integrators in the *lvardt* example integrate 4 state-variables 138 times (first-spike cell) and 115 times (second-spike cell) for a total of $4 \cdot (138 + 115) = 1012$ state-variable integrations. Comparison with the total integrations for the global method suggested a 40% speed-up which was confirmed by simulation.

The *lvardt* method creates a separate CVODES integrator for each cell in the network.

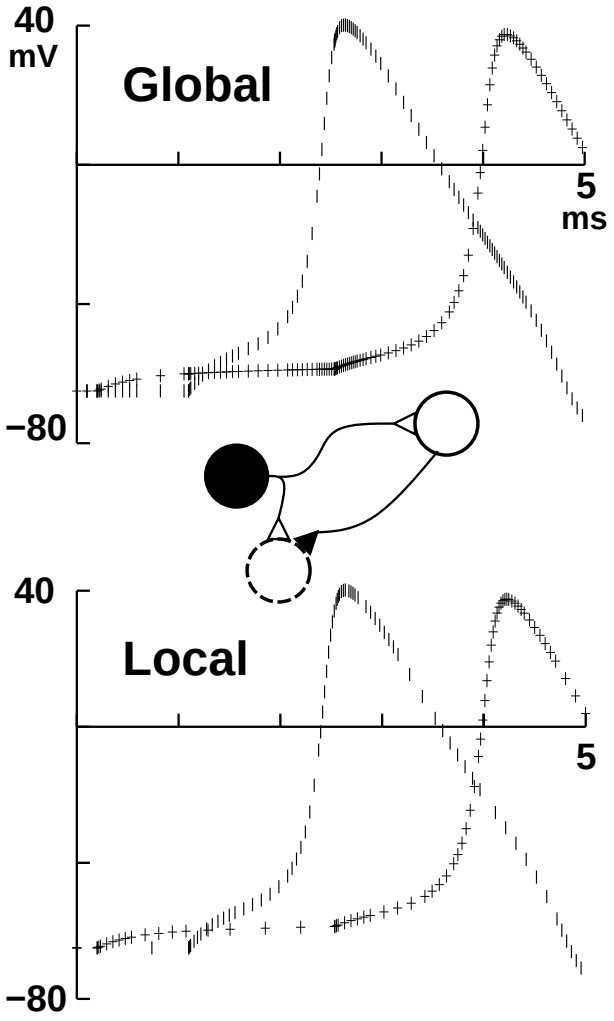


**Fig. 1: Voltage trajectories for two cells are shown for the global variable step method (top) and *lvardt*(bottom). $dt$ (interval between marks) varies together at top and separately at bottom. Network is shown in schematic: stimulator cell (filled circle) fires at time 0 and drives bottom cell weakly with a delay of 0.1 ms (+'s on graphs) and the right cell strongly with a delay of 1 ms (vertical lines on graphs). The two cells have Hodgkin-Huxley dynamics. Connection delay between right and bottom cell is 0.1 ms.**

Although there are many more integrators ($n$ for $n$ cells instead of 1), each integrator is more compact since it only has to handle the state-variables belonging to its particular neuron. Whether using one or $n$ integrators, the total number of state-variables remains the same. Although the expected relative performance gain with *lvardt* by function-call statistics in Fig. 1 is 40%, there is fixed overhead associated with each integrator and a per-timestep overhead required to determine which cell is to be integrated next. These reduce performance only slightly.

Because the system now is being calculated forward in time by multiple, independent integrators, an integration-coordinator is used to maintain the overall coherence of the integration. If the various neurons in the network are not connected, as in the case of testing parameter variation over a set of neurons, such coordination is not needed. However, in a network, the integration-coordinator is vital to permit synaptic signals to be communicated at appropriate times.

## Handling events

Handling events with *lvardt* requires that when an event arrives at a cell all of the state-variables for that cell are at their appropriate values for that time. This is accomplished with 3 standard variable-step-integrator operations: single step integration, interpolation, and re-initialization. Using these operations, we ensure that a) incoming events are always within reach of the receiving cell's integrator. b) individual integrators do not move too far beyond the network as a whole.

The individual integrators maintain state and derivative information on the interval of the most recent timestep. That is, each neuron's integrator can access states over an interval between the beginning $t_a$ and the end $t_b$ of a time-step: $t_b^i - t_a^i = dt^i$ for the i$^{th}$ neuron. This gives each individual integrator the ability to provide fast, high-order interpolation to a state at any time within the interpolation-range defined by the two bounding times. The integration-coordinator ensures that there is always overlap in these interpolation-ranges: $t_a^i \leq t_b^j \; \forall \; i, j$. We define $t_{b/e}min$ as the time of the earliest event $t_e$ or least-advanced integration bound $t_b$. To guarantee this, the integration-coordinator either handles the least-time event or single-steps the least advanced integrator, whichever is earlier. In this way, no integrator's $t_b$ and no event ever falls behind any $t_a$.

Fig. 2A is a schematic showing the integrator situation for 6 cells. The interpolation-range for each integrator is shown by a black rectangle. The integration-coordinator examines $t_{b/e}min$. In this case $t_{b/e}min = t_b^0$ and cell 0 gets advanced by a single step (striped rectangle). The state information for the prior $t_a^0$ is discarded as the bounds of the striped rectangle define a new $t_a^0$ and $t_b^0$. The requirement for continued overlap for all interpolation-ranges is guaranteed: the new $t_a^0$ equals the old $t_{b/e}min$, therefore $t_a^0$ remains less than all $t_b^i$.

Note that each individual variable time-step integrator will select a suitable, and typically different, $dt$ for its cell. The size of this $dt$ will depend on how rapidly the state-variables in that cell are changing. Therefore, the cells will each jump forward by

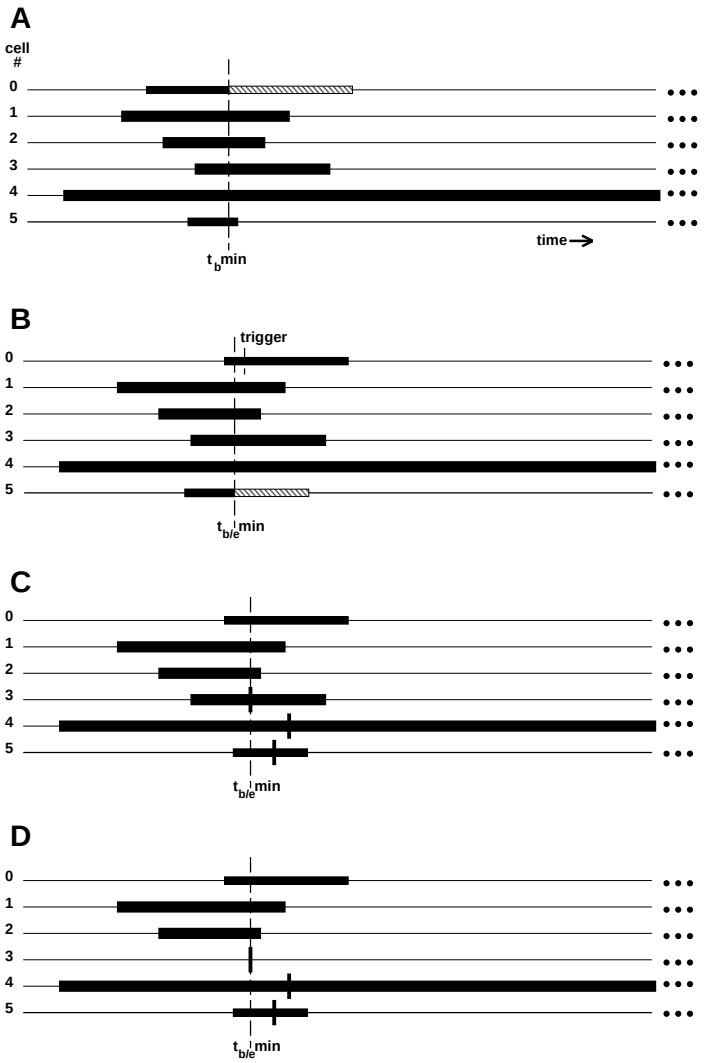


**Fig. 2: Local variable step method advances integration by interpolation-ranges (black rectangles) for 6 cells. A. Integration-coordinator requests integration for lagging cell (# 0 with minimum $t_b$). Integrator advances by $dt$ (length of hashed rectangle). B. Cell 0 triggers (crosses threshold). This is an event whose time is tentative since unprocessed synaptic events could still influence this cell. C. Cell 5 integrates forward. Trigger in cell 0 can now be handled since it is the earliest event. The handling of this trigger creates 3 events to be delivered to cells 3,4,5 at varying delays (vertical lines). Event in cell 3 is now $t_{b/e}min$. D. Cell 3 back-interpolates and re-initializes, giving $t_a^3 = t_b^3 = t_e^3$.**

different $dt$'s as the integrator for each is called.

Notice that cell 4 in Fig. 2A has advanced far ahead of the other integrators. This represents an extreme, yet common, situation that occurs when a quiescent cell quickly integrates forward to the end of simulation time. Until this cell receives an event, it has nothing to do and its integrator will never be called: $t_b^4 \geq t_b^i$ for all $i$. This cell will use no CPU time until it is activated by a synaptic event from another cell or a stimulus event from outside the network (e.g., a current injection).

The requirement for overlapping interpolation-range is critical for the handling of events. Overlapping interpolation-range guarantees that any event generated by the forward integration of a cell will be received by any other cell (or self) either in its interpolation-range or in its future: no event left behind. Fig. 2B illustrates the triggering of an event during integration of cell 0 (thin vertical line). A typical trigger would be the detection of a spike in cell 0 ($V_0 > V_{threshold}$). Notice that this event-triggering has no effect on the integration of cell 0 itself. Also note that the assignment of the trigger time remains tentative until $t_{b/e}min$ gets to this time. This is because a event could be received by cell 0 which would alter its voltage trajectory and change or remove the trigger time. The fact that event generation times do not have to lie on natural integration step boundaries allows greater than first order correctness in the calculation of event times. Cell 5 is integrated next: $t_{b/e}min = t_b^5$ (Fig. 2B).

The cell 0 event trigger is now $t_{b/e}min$ so that this event can be handled. Let's say that cell 0 is synaptically coupled to cells 3,4 and 5. The event-trigger will place 3 events on the queue for processing at times $t_e^j = t_{trigger} + delay_{0j}$ $(j = 3, 4, 5)$ (Fig. 2C). (We illustrate the general case of different positive synaptic delays. Delays could all be the same or could all be zero.) The next $t_{b/e}min$ is the event for cell 3. The integrator for cell 3 is called with 1) the event time and 2) information regarding the state change associated with this event. The cell 3 integrator must then back-interpolate to the event time and perform a re-initialization (Fig. 2D). The re-initialization abandons the current integration step for this cell and effectively creates a zero-length interpolation-range with $t_a = t_b$ at $t_e$. Re-initialization must be a separate step because zero-delay events are permissible. Therefore, any event received could trigger instantaneous events in other cells, causing them to re-initialize as well. In the absence of such a zero-delay event, the $t_b^3$ will now be $t_{b/e}min$ and the cell 3 integrator will be called to take the next forward step, incorporating the state-variable change associated with the received event (typically a change in a synaptic conductance or current). This cell 3 forward-step follows, and is separate from, the cell 3 re-initialization.

Fig. 2 illustrates several characteristics of the local variable time-step integration. First, unlike a global time-step method, there is no single simulation time. If you stop the simulation through an interrupt and start examining state-variables in the network, these will not generally represent correct values for the time returned by evaluating $t$. Instead, the simulation must record times and state-variables for different cells as each new $t_a$ is established. Because $t_a$ is a backward interpolation boundary, we know that this point will remain valid. To save simulation values at a particular time (such as the end of the

simulation), this time has to be declared as an recording event for all of the integrators and then handled in turn like any other event. At the end of the simulation, all integrators must move past the end-time and interpolate back.

A second point is that events that modify state-variables require re-initialization of individual integrators. This is the largest overhead occurrence in the integration process. However, it does not become a problem as long as events are relatively rare compared to forward integration steps. Note that recording events do not modify state-variables and require only interpolation without re-initialization.

A third point is illustrated in Fig. 2B. Events can be triggered at time points that do not correspond to time-step boundaries. This condition-case interpolation process differs from the time interpolation discussed above. Condition-case interpolation is handled by linear interpolation between state-variable values at the time-interpolation boundaries. A more sophisticated, but slower, method would use a Newton method to converge onto the threshold-passing time.

In order to demonstrate the usefulness of the method, we explored its performance in several examples. To maximally challenge *lvardt*, we used a mutual-inhibition model, a model which fully synchronizes through recurrent inhibition with full connectivity. In this case, the expected superiority of multiple integrators is expected to be negated by the fact that all integrators are doing the same thing at the same time. At the other extreme we show that synfire chains enjoy a dramatic performance improvement when using *lvardt*. Finally, we demonstrate the performance improvement obtained using *lvardt* on a more biologically detailed thalamocortical network.

**References**

SD Cohen and AC Hindmarsh. Cvode user guide. Technical report, Lawrence Livermore National Laboratory, Livermore, CA, 1994.

ML Hines and NT Carnevale. Neuron: a tool for neuroscientists. *The Neuroscientist*, 7:123–135, 2001.