Measurement of simulation speed: its relation to simulation accuracy

Robert G. Smith

Dept. of Neuroscience Univ. of PA Phila., PA, 19104-6058

Published in "Computation in Neurons and Neural Systems", 1994, Ed. by Frank H. Eeckman, Kluwer Academic Publishers, Boston.

Abstract

This article presents an unbiased method for measuring simulation speed for compartmental simulators. The method measures how long it takes to simulate a neural circuit component at a given overall accuracy. Because both spatial and temporal discretizations influence overall accuracy, it is possible to optimize both spatial and temporal accuracy to maximize overall speed.

Introduction

Recently, the Bower group1 proposed a set of benchmarks, named "Rallpacks", designed for testing compartmental neural circuit simulators. These benchmarks are useful for predicting a simulation's run time. However a problem exists with the Rallpack benchmarks as currently defined. The benchmarks lack generality because the speed measures they report are biased towards a specific type of simulator.

Several numerical integration methods have been used for solving the difference equations generated from compartmental models. The implicit method of numerical integration is widely used because of its unconditional stability. This method generates a set of simultaneous equations that must be solved each time step. With his NEURON simulator, Hines 3,4,5 has pioneered the use of the "Gaussian elimination" solution method for these difference equations. The advantage of this method, now used widely in other simulators such as GENESIS, is that it takes the same amount of computation for any time step.

Some compartmental simulators, such as NeuronC6, use the implicit integration method but employ a different method of solution, called "relaxation"2,3,6. This method is useful when the neural circuit contains resistive loops3. Other simulators employ explicit integration methods3,4. There are several differences in the way relaxation and explicit methods perform in the Rallpack benchmarks compared with the NEURON and GENESIS simulators 1,3.

The major difference is that is that the relaxation method is fast when compartments are coupled loosely, but slow when compartments are coupled tightly. In general, there are two ways to

couple compartments more loosely: 1) use small time steps, or 2) use larger space steps, i.e. break a cable into larger compartments and coupling resistances. Each of these has a disadvantage. Smaller time steps require more steps to cover the same time, which may require more computation. Larger space steps reduce accuracy.

Since NEURON takes the same amount of computation to solve a network with any degree of coupling (i.e. "stiff" or "loose" difference equations), it is limited in accuracy mainly by the time step. For non-spiking neurons, this means not larger than 100 usec for a reasonable degree of accuracy. NEURON can solve much longer (~infinite) time steps for "static" simulations with the same amount of computation as any other time step. Hines3 argues convincingly that since for most physiological simulations we don't need better accuracy than 2-5%, time steps on the order of 100 usec are useful, and therefore the Gaussian elimination method he has developed is fast.

This reasoning seems to be reflected in the Rallpacks defined by Bhalla et al1, where the spatial discretization (the "step size") is set very fine (ostensibly to remove spatial discretization as an issue in measuring accuracy). This seems reasonable because (at least for large models) the time it takes to run a simulation is proportional to the number of compartments with Hines' method.

In contrast, the relaxation method used in NeuronC is faster than the Gaussian elimination method used by the NEURON and GENESIS simulators 1,5 for small time steps, but slower when large time steps are used, depending on the compartment size. With cables broken into compartments of 0.01 lambda a simulation using relaxation runs slowly, but with compartments of 0.1 lambda it is competitive with Gaussian elimination.

Simulation Speed and Compartments

Bhalla et al have suggested the "dimensional units" for speed in neural simulators with their Rallpack standards 1. Their definition of simulation speed is:

Simulation speed = number of compartments * simulated time / run time (1)

In effect, this defines "speed" as proportional to the number of compartments solved per run-time second. This seems reasonable, because for a simulator based on compartments, how long it takes to solve a compartment might seem at first glance to be a basic and very useful measure.

However, there is a problem with this definition of speed. The problem is that it includes "compartments" in the numerator, which implies that all compartments are equal, i.e. all compartments have the same effect on the outcome of the simulation. This is misleading because not all compartments are equal in their effect on the simulation. Compartment size is a major factor in the quality of simulation. Smaller compartments give more spatial accuracy, and larger ones allow the simulation to run faster. The number of compartments required to simulate a neural circuit varies inversely with the compartment size. Therefore the speed of a simulator at running the simulation is not always directly proportional to its speed in solving compartments.

This is not to say that "compartment speed" is unimportant. On the contrary, it is imperative that a compartmental simulator be efficient at solving the difference equations that represent compartments. However, the time it takes to solve a compartment is not really at issue. What really counts is how long a simulation takes to achieve a desired degree of accuracy.

Occasionally, one may want to fix the number of compartments constant for some reason, but ordinarily this is not a major concern. If the compartment size is small enough to maintain accuracy, there seems little reason to specify exactly the number of compartments. And as Hines4 suggests, the task of specifying compartments in a branched tree is tedious and prone to error. To alleviate this problem, for some compartmental simulators (e.g. NeuronC) cables and branching trees are defined by the user as such, i.e. by length, diameter, Rm, Ri, and branching pattern, etc. The number of compartments for a particular realization of a simulation is not determined directly by the user. Instead, the number of compartments is controlled by setting a variable that determines the fineness of the spatial discretization.

Objective Measure of Speed

Speed is defined here only in terms of the computation time for a given neural circuit, computed to a certain accuracy. Such a speed vs. accuracy definition places a premium on running a particular simulation the most efficient way possible, regardless of the exact hardware, method of describing the neural circuit, method of solving the difference equations, or number of compartments. Speed can be reported as the rate of the simulation in terms of its relation to "real time". I suggest the term "slowness" here because it seems useful to define an "inverse speed".

```
Slowness   =    simulation   run  time   /   simulated time        (2)

Slowness / neuron     =         slowness1 for component1 +
                                   slowness2 for component2 +   . . . (3)

Total simulation time  =        Overhead  +  slowness  *  simulated time  (4)

Simulation speed    = 1 / simulation time                          (5)

Where:  Overhead=   time to set up "any" simulation.
       component=   cable (per lambda), branch, soma, synapse, channel, etc.
     slowness1,2=   function of size, accuracy.
```

A useful measure of a simulator might report the system resources required by a simulation. Resources could include "slowness" and "memory", but might also include graphics and other I/O facilities defined in a familiar manner. The "slowness" resource is dependent only on 1) the hardware, 2) the software environment, 3) an "overhead" term dependent on the setup time required by the simulator, and 4) an efficiency function that scales the "slowness" based on the total number and size of the neural circuit components. These reflect efficiency of scale, i.e. the fact that it sometimes requires less computation to solve one neural component (e.g. a cable) if there are many similar components in the simulation. The efficiency functions also reflect the fact that it takes less computation to solve a neural component at low accuracy than at high accuracy (Figure 1).

Determining Accuracy

Crucial to defining the "objective" benchmark is a good overall definition of "accuracy". There are many possible definitions for accuracy (assumed here to be "root-mean-square error"), but all of them will be affected by both spatial and temporal discretization errors, because the voltage vs. time waveshape recorded from the dendritic tree of a neuron reflects both spatial and temporal issues:

```
   Total error   =   function of (spatial error, temporal error)          (6)

A first approximation of the function:

   Total error   ~=   spatial error    +    temporal error                (7)

   Where:  Total error =   error measured from simulation by comparison.
         spatial error =   ideal error measured with fine temporal steps.
        temporal error =   ideal error measured with fine spatial steps.
```

To assess the error originating in one discretization, we partition the other so much finer that practically it does not influence error (i.e. coarse time, fine space). Then we reverse the discretization (i.e. fine time, coarse space). The two "partial errors" then presumably can be related to the total error with both discretizations coarse (i.e. coarse time, coarse space). At a first glance it is not obvious exactly what type of function relates the "spatial" and "temporal" errors to the total error, but in a simple, well-defined model is possible to derive the function empirically. Generally, spatial and temporal error sum additively with first-order integration (i.e. forward or backward Euler methods). Error with second-order integration (e.g. Crank-Nicolson3) in some cases is subtractive, i.e. when space and time errors are nearly commensurate, a coarser space discretization in some cases produces better total accuracy than a finer one.

Concluding Remarks

Whichever qualities of a simulation are chosen to measure "error", they are generally affected by both spatial and temporal discretization errors. These errors can be determined independently and can be used to estimate what compromise of accuracy vs. speed is desirable in a given neural circuit simulation. By calculating how long it takes to simulate the components of a neural circuit at a given accuracy, a neuroscientist can 1) compare different simulators, 2) determine how long a simulation of a different neural circuit will take, and 3) determine how to run a simulation in the most efficient manner.

This work was supported by NIMH Grant MH48168.

References

[1] Bhalla, U.S, Bilitch, D.H. and Bower, J.M. (1992) Rallpacks: a set of benchmarks for neuronal simulators. Trends Neurosci. 15: 453-458.

[2] Cooley, J.W., and Dodge, F.A. (1966) Digital computer solutions for excitation and propagation of the nerve impulse. Biophys. J. 6:583-599.

[3] Hines, M. (1984) Efficient computation of branched nerve equations. Int. J. Biomed. Comput. 15: 69-76.

[4] Hines, M. (1989) A program for simulation of nerve equations with branching geometries. Int J. Biomed. Comput. 24: 55-68.

[5] Hines, M. (1991) NEURON: an interactive neural simulation program. In: Analysis and Modeling of Neural Systems, II, (F. Eeckman, ed). Kluwer.

[6] Smith, R.G. (1992) NeuronC: a computational language for investigating functional architecture of neural circuits. J. Neurosci. Meth. 43: 83-108.