NOTE Communicated by Eugene Izhikevich

Solution Methods for a New Class of Simple Model Neurons

Mark D. Humphries
m.d.humphries@sheffield.ac.uk
Kevin Gurney
K.Gurney@shef.ac.uk
Adaptive Behaviour Research Group, Department of Psychology, University of Sheffield, Sheffield S10 2TP, U.K.

Izhikevich (2003) proposed a new canonical neuron model of spike generation. The model was surprisingly simple yet able to accurately replicate the firing patterns of different types of cortical cell. Here, we derive a solution method that allows efficient simulation of the model.

1 Introduction

Two constraining criteria govern the choice of neuron models: accuracy and speed. If a model is highly accurate in its behavior, it probably demands too much computation time to be used in large networks; if a model has a low computation time, it is probably not accurate enough to capture all key dynamic properties of the modeled cell. As modelers, we seek efficient models that maximize the trade-off between accuracy and speed and which thus allow networks of sufficiently accurate neurons to be simulated comfortably on modest computing resources: this search has led to the proposal of numerous models.

Izhikevich (2003) proposed a simple neuron model that was far more accurate in its replication of membrane potential behavior than the common leaky integrate-and-fire model, but that was not much more complex in its mathematical form. An equivalent, biophysical form of the model is given in Izhikevich (2006), which we use here. If v is the membrane potential and u is the dominant slow current of the neuron class, then

$$C\dot{v} = k(v - v_r)(v - v_t) - u + I, \tag{1.1}$$

$$\dot{u} = a\,[b(v - v_r) - u]\,, \tag{1.2}$$

with reset condition

$$\text{if } v \geq v_{peak} \text{ then } v \leftarrow c,\ u \leftarrow u + d,$$

where C is capacitance, v_r and v_t are the resting and threshold potentials, a is the time constant of the slow current, and c is the reset potential (i.e.,

the value of the membrane potential immediately after an action potential is fired). Parameters k and b are derived from the current-voltage relation of the neuron, and d is tuned to achieve the desired spiking behavior. All of these parameters are given constant values for a specific neuron class, and the range of spike train phenomena captured by this two-dimensional model is impressive (Izhikevich, 2004).

The rationale behind Izhikevich's original work was to provide a better accuracy-speed trade-off than was available in existing models (Izhikevich, 2004). Notwithstanding the success of this program, our aim was to improve this trade-off even further by finding better solution methods for the model than those used in its original deployments (Izhikevich, 2004; Izhikevich, Gally, & Edelman, 2004).

2 A Zero-Order Hold Solution

In all example simulations and code that Izhikevich (2006) provides and in the large-scale model of cortex (Izhikevich et al., 2004), a forward Euler update is used as the numerical method for solving equations 1.1 and 1.2: given a time step of Δt, a differential $\dot{y}$ is approximated at time $t + \Delta t$ by $y(t + \Delta t) = y(t) + \Delta t\, \dot{y}(t)$. Forward Euler is the simplest form of numerical integration but has considerable problems of stability and accuracy under certain conditions unless a small time step is used (Hansel, Mato, Meunier, & Neltner, 1998). Its main advantage is that each equation need be computed only once per time step. However, there are other once-per-time-step numerical integration methods available that achieve better levels of accuracy, as usually required in simulating neural models.

One method, used with coupled ordinary differential equations (ODEs) of the form $\dot{y}_k = f\left(y_k, y_i, y_j, \ldots\right)$, is to treat the other time-dependent variables $(y_i, y_j, \ldots)$ in each equation as constant over some small time interval Δt (equivalent to approximating these variables by the zeroth-order terms in their Taylor series expansions). If there is an exact solution for each of the resulting equations $\dot{y}_k = f(y_k, D)$, where D is constant, then these may be used to determine $y_k(t + \Delta t)$ by holding suitable sets of variables constant at any one time. With two equations (as in equations 1.1 and 1.2), we simply solve exactly each equation by holding the other variable constant. This scheme is referred to as the *zero-order hold approximation* (ZOH) and has been used, for example, to solve Hodgkin-Huxley type models (Dayan & Abbot, 2001). Solving the typical form of the equations for neural models in this way produces solutions that are grouped as a numerical method called *exponential Euler*: the neural simulator GENESIS uses this a solution method (Bower & Beeman, 1998). We now show that the zero-order hold technique may be applied to the Izhikevich model equations.

2.1 Slow Current. We begin with the slow current u, as this has a straightforward solution. We take equation 1.2 and assume v to be constant

between an initial t_0 and current time t, and solve. Noting that the right-hand-side of equation 1.2 has no explicit dependence on time, it is trivially linearly separable in u, t and can be solved exactly using an integrating factor $e^{\int a\,dt} = e^{at}$:

$$u(t) = u(t_0)e^{-a(t-t_0)} + \int_{t_0}^{t} ab(v - v_r)e^{-a(t-\tau)}d\tau.$$

Assuming constant v, the solution is therefore

$$u(t) = b(v - v_r)[1 - e^{-a(t-t_0)}] + u(t_0)e^{-a(t-t_0)}. \tag{2.1}$$

We use this as a discrete-time approximation of u after some small time step Δt with the replacements $t \leftarrow t_0 + \Delta t$.

Izhikevich (2006) gives some models of specific neuron species that derive u from a slightly different approximation to the dynamics of the slow current. However, the resulting form of u is similar to equation 1.2, and so rederiving the ZOH solution to replace equation 2.1 is straightforward. We give an example for the model of cortical fast-spiking interneurons, as we use this model later: the form of u for this model is

$$\dot{u} = \begin{cases} -a\,u, & \text{if } v < v_b, \\ a\left[b(v - v_b)^3 - u\right], & \text{if } v \geq v_b, \end{cases}$$

and there is no reset of u following a spike (i.e., $d = 0$). Given bounds t_0 and t as before, the solutions are

$$u(t) = \begin{cases} u(t_0)e^{-a(t-t_0)}, & \text{if } v(t_0) < v_b \\ b(v - v_b)^3\left[1 - e^{-a(t-t_0)}\right] + u(t_0)e^{-a(t-t_0)}, & \text{if } v(t_0) \geq v_b. \end{cases} \tag{2.2}$$

2.2 The Membrane Potential. We begin the same way as above: we take equation 1.1 and assume u to be constant over some small time interval between the start of integration t_0 and current time t, and solve (we assume I to be a constant current injection, so the solution is exact in I as well). In general, nonlinear ODEs do not have a solution; fortunately, equation 1.1 is a special case and can be expanded out into

$$\dot{v} = \frac{kv_r v_t - u + I}{C} - \frac{k(v_r + v_t)}{C}v + \frac{k}{C}v^2, \tag{2.3}$$

which is a Riccati differential equation, of the form $\dot{v} = P(t) + Q(t)v + R(t)v^2$. These can be solved by substituting $z = 1/(v - v_1)$, where v_1 is a particular solution to $\dot{v}$. Since P, Q, R are in fact independent of time (because of our assumptions of constancy), the equation for $\dot{z}$ is (trivially) linearly

separable: $\dot{z} = -[Q(t) + 2v_1 R(t)]z - R(t)$. Having solved this in terms of z, we can solve for v with the relation $v = v_1 + 1/z$.

Finding a particular solution to equation 2.3 by inspection is difficult, so we propose here a simple method that generates a particular solution, and thus allows the solution of any Ricatti differential (we are not aware of this method previously appearing in the literature). We seek a particular solution by putting $\dot{v} = 0$ since, with I and u constant, we then obtain a simple quadratic in v

$$kv^2 - k(v_r + v_t)v - u + I + kv_r v_t = 0,$$

which has solutions

$$v_1 = \frac{k(v_r + v_t) \pm A}{2k}, \; A = \sqrt{[-k(v_r + v_t)]^2 - 4k(-u + I + kv_r v_t)}. \quad (2.4)$$

Since we need only one particular solution, we choose the root with $+A$. Now that we have an expression for the particular solution v_1, we can complete the substitution to express equation 2.3 in terms of z,

$$\dot{z} = -\left\{-\frac{k(v_r + v_t)}{C} + 2\left[\frac{k(v_r + v_t) + A}{2k}\right]\frac{k}{C}\right\}z - \frac{k}{C},$$

which simplifies to the linearly separable form

$$\dot{z} + \frac{A}{C}z = -\frac{k}{C}. \quad (2.5)$$

This can be solved with an integrating factor $e^{\int A/C dt} = e^{tA/C}$, giving the solution between t_0 and t,

$$z(t) = \int_{t_0}^{t} -\frac{k}{C}e^{-(t-\tau)A/C}d\tau + C_0 e^{-tA/C}, \quad (2.6)$$

where C_0 is a constant of integration (which we find below). Integrating equation 2.6 gives

$$z(t) = -\frac{k}{A}(1 - e^{-(t-t_0)A/C}) + C_0 e^{-tA/C},$$

and we go back to v with the relation given above to get

$$v(t) = \frac{k(v_r + v_t) + A}{2k} + \left[-\frac{k}{A}(1 - e^{-(t-t_0)A/C}) + C_0 e^{-tA/C}\right]^{-1}. \quad (2.7)$$

We now find C_0 in terms of the initial value $v(t_0)$. Thus,

$$v(t_0) = \frac{k(v_r + v_t) + A}{2k} + \frac{e^{t_0 A/C}}{C_0},$$

and solving for C_0 gives

$$C_0 = e^{t_0 A/C}\left[v(t_0) - \frac{k(v_r + v_t) + A}{2k}\right]^{-1}. \tag{2.8}$$

We substitute equation 2.8 into 2.7 to get the full ZOH solution for v:

$$v(t) = \frac{k(v_r + v_t) + A}{2k} + \left\{ -\frac{k}{A}(1 - e^{-(t-t_0)A/C}) + e^{-(t-t_0)A/C}\left[v(t_0) - \frac{k(v_r + v_t) + A}{2k}\right]^{-1} \right\}^{-1}, \tag{2.9}$$

where A is given in equation 2.4. Again this becomes a discrete-time solution with the replacement $t \leftarrow t_0 + \Delta t$. Spike detection and reset in discrete time is handled by: if $v(t_0 + \Delta t) \geq v_{\text{peak}}$ then $v(t_0 + \Delta t) \leftarrow c$, $u(t_0 + \Delta t) \leftarrow u(t_0 + \Delta t) + d$. Following Izhikevich (2006), we also set $v(t_0) = v_{\text{peak}}$.

2.3 Potential Limitations of the Solution. For some values of u and I, the particular solution v_1 found by setting $\dot{v} = 0$ may have complex roots (with A imaginary). This implies that $\dot{v}$ has no fixed point and will escape to infinity in a finite time. However, even if v_1 is complex, it can be shown that the resulting value for v is always real as required. Let us assume that A is imaginary, that is, $A = iB$. Substituting this in equation 2.9 and using the Euler identity ($e^{i\theta} = \cos\theta + i\sin\theta$) gives

$$v(t) = \frac{(v_r + v_t)}{2} + \frac{Y\cos X + (kY^2/B)\sin X - (B/4k)\sin X}{\frac{1}{2}(1 + \cos X) + (2kY/B)\sin X - 2(kY/B)^2(-1 + \cos X)} \tag{2.10}$$

where $X = -(t - t_0)B/C$ and $Y = v(t_0) - (v_r + v_t)/2$. Notice that v is always real and that our method for finding a particular solution thus produces sensible results. In addition, equation 2.10 gives a form for the calculation of v that avoids having to represent and compute complex numbers in simulation.

As with any other numerical method, it is useful to find an upper bound on Δt. Our limit here is the value beyond which the assumption of constant u in equation 1.1 and constant v in equation 1.2 breaks down.

Ideally, for equation 1.1, $u(t_0) \gg \Delta t\dot{u}(t_0)$, so that the deviation from a constant value of u over Δt is very small (based on the Taylor series expansion, and assuming contribution from higher-order terms will be negligible). Thus, a reasonable upper bound is $\Delta t < u(t_0)/\dot{u}(t_0)$. Similarly, for equation 1.2, the upper bound is $\Delta t < v(t_0)/\dot{v}(t_0)$. These bounds should hold for most but not all t (if v does escape to infinity in a finite time, as when a spike is produced, then the ratios would also go to infinity and should be discounted).

3 Comparison of Numerical Solution Methods

For a given time step, the forward Euler method is always the fastest fixed-step solution method (in terms of computation time), as it requires one multiplication and one addition operation per ODE above the floating-point operation requirements of the ODE system itself. This, combined with its ease of implementation, explains its widespread use in neural modeling (including the solution of Izhikevich's model neurons). Conversely, one would expect that the forward Euler method is always the least accurate fixed-step solution for a given time step (Hansel et al., 1998). We thus compare our zero-order hold (ZOH) solutions and forward Euler as numerical methods for solving the simple model neurons to determine which method is the more efficient (achieving the best accuracy-to-speed trade-off).

We use four classes of cortical neurons modeled as Izhikevich simple model neurons: the regular-spiking (RS) neurons, intrinsically bursting (IB) neurons, chattering (CH) neurons, and fast-spiking (FS) interneurons. Their parameter values, taken from Izhikevich (2006), are given in the legend of Figure 1. All model neurons were coded in both Matlab 7 (Mathworks) and C versions to test differences due to language implementations (all code available at www.abrg.group.shef.ac.uk). The C version used equation 2.10 if A, given by equation 2.4, was complex, and equation 2.9 otherwise. One Matlab version ("control") also used this system, and hence acted as a control for other differences between the C and Matlab implementations; the other Matlab version ("native") made use of Matlab's built-in handling of complex numbers and used equation 2.9 throughout.

As a benchmark for accuracy, we solved each model neuron with a high-order variable-step solver, **ode45** from the Matlab ODE suite: a Runge-Kutta 4th-5th order formula, the Dormand-Prince pair (Ashino, Nagase, & Vaillancourt, 2000). Each benchmark simulation for a model neuron class was run for 1000 ms, with a current step I onset at 100 ms. Using the same input, the fixed-step solutions (forward Euler and ZOH) were then assessed for each time step Δt taken from the interval $[0.01, 0.5]$ms in increments of 0.01 ms, giving 50 tested time steps in total. (The upper limit of $\Delta t = 0.5$ ms was determined by computing the rough bounds on step size for the ZOH method for each neuron class, as described above.) Each simulation using a fixed-step solution terminated when the same number of spikes as the

benchmark simulation (using the variable-step solver) had been produced. Errors in solution were expressed as the mean error per spike for each Δt, where each error per spike was given as the time difference between the variable-step solution crossing v_{peak} and the fixed-step solution's crossing of v_{peak}. The minimum possible error would thus be in the interval $[0,\ \Delta t/2]$, due to the quantization imposed by the fixed-step methods.

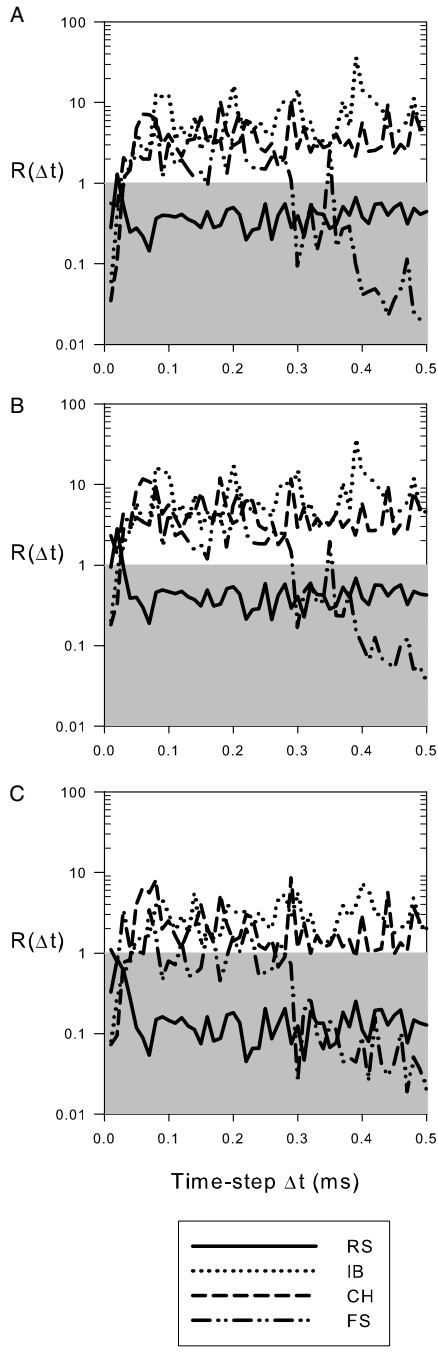
Computation times for the fixed-step solutions for each model neuron class were also computed. For the Matlab versions, a batch of 30 simulations was run for each Δt, and the time taken for each simulation was recorded. The mean computation time was then calculated by averaging over that batch. For the C versions, 30 batches of 1000 simulations were run for each Δt, and the time taken for each batch was recorded. The mean computation time was then calculated by averaging over all 30 batches.

For each language-implementation and neuron-class combination, we took the mean computation times $T^{\text{zoh}}(\Delta t)$, $T^{\text{euler}}(\Delta t)$ and mean errors per spike $E^{\text{zoh}}(\Delta t)$, $E^{\text{euler}}(\Delta t)$ and calculated the following ratios: $T^{\text{r}}(\Delta t) = T^{\text{euler}}(\Delta t)/T^{\text{zoh}}(\Delta t)$ and $E^{\text{r}}(\Delta t) = E^{\text{euler}}(\Delta t)/E^{\text{zoh}}(\Delta t)$. We expected (and indeed found) that $T^{\text{r}}(\Delta t) > 1$ always, because the forward Euler method is always faster for a given time step; we expected that $E^{\text{r}}(\Delta t) < 1$ for most time steps and neuron types, because the ZOH method is expected to be more accurate than forward Euler. To express the accuracy-to-speed trade-off, we calculate $R(\Delta t) = T^{\text{r}}(\Delta t) \times E^{\text{r}}(\Delta t)$. Thus, if $R(\Delta t) > 1$, then the magnitude gain in accuracy by using the ZOH method is greater than the magnitude loss in speed for that time step; in other words, the ZOH method is more efficient.

We plot $R(\Delta t)$ for each neuron class and language implementation combination in Figure 1 (the log scale is used because if $R(\Delta t) = 2$ means that the ZOH method was twice as efficient, then $R(\Delta t) = 0.5$ means it was

---

Figure 1: Accuracy-to-speed trade-off for zero-order hold (ZOH) and forward Euler numerical solutions to four classes of simple model neurons. A value of $R(\Delta t) > 1$ (white area) indicates that the ZOH method had a greater magnitude gain in accuracy than loss of speed for that time step, and thus achieved a better accuracy-to-speed trade-off than forward Euler. (A) Native Matlab version, making use of its built-in handling of complex numbers. (B) Control Matlab version. (C) The C version, using an equivalent scheme to the control Matlab version. The parameter values used for each model were as follows. Regular spiking (RS) neuron: $C = 100$, $v_{\text{r}} = -60$, $v_{\text{t}} = -40$, $I = 70$, $k = 0.7$, $a = 0.03$, $b = -2, c = -50, d = 100$, $v_{\text{peak}} = 35$; Intrinsically bursting (IB) neuron: $C = 150$, $v_{\text{r}} = -75$, $v_{\text{t}} = -45$, $I = 500$, $k = 1.2$, $a = 0.01$, $b = 5$, $c = -56$, $d = 130$, $v_{\text{peak}} = 50$. Chattering (CH) neuron: $C = 50$, $v_{\text{r}} = -60$, $v_{\text{t}} = -40$, $I = 200$, $k = 1.5$, $a = 0.03$, $b = 1$, $c = -40$, $d = 150$, $v_{\text{peak}} = 20$. Fast-spiking (FS) interneuron: $C = 20$, $v_{\text{r}} = -55$, $v_{\text{t}} = -40$, $I = 100$, $k = 1$, $a = 0.2$, $b = 0.025$, $c = -45$, $d = 0$, $v_{\text{peak}} = 25$, and $v_{\text{b}} = -55$ (see text).


A
B
C
R(Δt)
100
10
1
0.1
0.01
0.0
0.1
0.2
0.3
0.4
0.5
Time-step Δt (ms)
RS
IB
CH
FS

equivalently half as efficient). We find consistent patterns of dependence on Δt for each neuron class across all language implementations. Solving the RS class was always more efficient using forward Euler, an unsurprising result given the simple dynamics of this model. Solving the CH and IB classes was always more efficient using the ZOH method, except at the smallest Δt. In either Matlab version, solving the FS class was more efficient using the ZOH method over the approximate interval $\Delta t \in [0.02, 0.3]$ ms; over the same interval, the C version showed that neither method was clearly more efficient: mean $R(\Delta t)$ was $\sim$1.17 over that interval.

Over all simulations we found (1) that the errors were identical to three significant places between the two Matlab versions, verifying that equation 2.10 gives correct results in simulation; (2) that computation using equation 2.10 was faster in Matlab than using built-in complex numbers, hence the higher $R(\Delta t)$ for the "control" Matlab version; and (3) that there was a considerable difference in relative computation times across language implementations (which was not dependent on the form of ZOH solution). Across all neuron classes and time steps, the "control" Matlab version had $T^{\mathrm{r}}(\Delta t) \sim 0.7$ but the C version had $T^{\mathrm{r}}(\Delta t) \sim 0.2$.

4 Conclusion

The new simple neuron model of Izhikevich (2003) may provide the best trade-off yet between computation time and accuracy for a model of spiking activity. We have derived a new numerical solution to this model, which may allow for significantly improved efficiency in simulation. Our numerical simulations showed how this efficiency was dependent on the choice of time step for most neuron classes. In general, we expect the choice of time step to be away from the extremes of the range that we tested, around $\Delta t \sim 0.1$. Too small ($\Delta t \sim 0.01$), and there is no gain in efficiency over more accurate methods, including other fixed-step methods such as midpoint (in simulation, we found that the fixed-step simulations in Matlab took at least as long as those using the variable-step solver for $\Delta t \in [0.01, 0.05]$). Too large ($\Delta t \sim 0.5$), and enough error is introduced to the simulation to markedly affect the results (in simulation, we found that the mean error for the CH, IB, and FS neuron classes was at least 10 ms per spike). Thus, from our results, we believe that ZOH is a more efficient solution method for most neuron classes over the likely range of usable time steps.

Future work will determine how these results from a single-neuron case extrapolate to a network. One might naively expect that errors in individual neuron solutions would propagate through a network, leading to considerable accumulation of error over time; on the other hand, the firing of a spike is dependent on multiple presynaptic firings, and thus the errors may be absorbed in the naturally noisy process of spike accumulation. The choice of network time step is more critical than for the single neuron: in addition to reducing error (whether it propagates or not), the time step ideally should

be smaller than any characteristic scale of the dynamics of interest, to avoid artifactual oscillations and synchronizations caused by quantizing spikes to the time-step.

The solution also forms the basis of further analytic studies of this neuron class. For example, solutions 2.1 and 2.9 give exact values if equilibrium solutions to the coupled system 1.1–1.2 exist for given parameter values and injection currents.

Acknowledgments

We thank Nathan Lepora for discussions on the mathematics and Ric Wood for helpful comments on a draft of this manuscript. This work was supported by the EU Framework 6 ICEA project.

References

Ashino, R., Nagase, M., & Vaillancourt, R. (2000). Behind and beyond the MATLAB ODE suite. *Computers and Mathematics with Applications*, *40*, 491–512.

Bower, J. M., & Beeman, D. (1998). *The book of GENESIS: Exploring realistic neural models with the general neural simulation system* (2nd ed.). New York: Springer-Verlag.

Dayan, P., & Abbot, L. F. (2001). *Theoretical neuroscience*. Cambridge, MA: MIT Press.

Hansel, D., Mato, G., Meunier, C., & Neltner, L. (1998). On numerical simulations of integrate-and-fire neural networks. *Neural Comput.*, *10*(2), 467–483.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.*, *14*, 1569–1572.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.*, *15*(5), 1063–1070.

Izhikevich, E. M. (2006). *Dynamical systems in neuroscience*. Cambridge, MA: MIT Press.

Izhikevich, E. M., Gally, J. A., & Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cereb. Cortex*, *14*(8), 933–944.

Received June 21, 2006; accepted October 7, 2006.