

Integration of rule-based models and compartmental models of neurons

David C. Sterratt*, Oksana Sorokina, and J. Douglas Armstrong

School of Informatics, University of Edinburgh
10 Crichton St, Edinburgh EH8 9AB, UK
`david.c.sterratt@ed.ac.uk`, `osorokin@inf.ed.ac.uk`
`douglas.armstrong@ed.ac.uk`

Abstract. Synaptic plasticity depends on the interaction between electrical activity in neurons and the synaptic proteome, the collection of over 1000 proteins in the post-synaptic density (PSD) of synapses. To construct models of synaptic plasticity with realistic numbers of proteins, we aim to combine rule-based models of molecular interactions in the synaptic proteome with compartmental models of the electrical activity of neurons. Rule-based systems allow interactions between the combinatorially large number of protein complexes in the postsynaptic proteome to be expressed straightforwardly, and simulations of rule-based systems are stochastic and thus can deal with the small copy numbers of proteins and complexes in the PSD. Compartmental models of neurons are expressed as systems of coupled ordinary differential equations and solved deterministically. We present an algorithm which incorporates stochastic rule-based models into deterministic compartmental models and demonstrate an implementation of this hybrid system using the SpatialKappa and NEURON simulators.

Keywords: Hybrid stochastic-deterministic simulations, hybrid spatial-nonspatial simulations, multiscale simulation, rule-based models, compartmental models, computational neuroscience

1 Introduction

The experimental phenomena of long term potentiation (LTP) and long term depression (LTD) show that synapses can transduce patterns of electrical activity on a timescale of milliseconds in the neurons they connect into long-lasting changes in the expression levels of neurotransmitter receptor proteins. This synaptic plasticity plays a crucial role in the development of a functional nervous system and in encoding semantic memories (e.g. motor patterns) and episodic memories (experiences), converting stimuli lasting for seconds into memories that last a lifetime [15].

* We thank Anatoly Sorokin for his help with SpatialKappa and comments on an earlier version of the manuscript and Vincent Danos for thought-provoking discussions.

There are a number of computational models of how synaptic plasticity arises from patterns of pre- and postsynaptic electrical activity, the dynamics of α -amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid receptors (AMPA) and *N*-methyl-D-aspartic acid receptors (NMDARs), calcium influx through these receptors and intracellular signalling in the postsynaptic density (PSD), a dense, protein-rich structure attached to neurotransmitter receptors [1, 21, 14, 27]. The level of detail of the molecular component of these models ranges from deterministic simulations in one compartment [1] thorough stochastic models with coarse granularity [26] and, at the greatest level of detail, diffusion of molecules through space using MCell [25, 27].

The model with the greatest number of molecular species has 75 variables representing the concentrations of signalling molecules, complexes of signalling molecules and phosphorylation states [1]. This constitutes a small subset of the 1000 proteins identified in the mouse postsynaptic proteome, the collection of proteins in the PSD [5]. Even the subset of the postsynaptic proteome containing proteins associated with membrane-bound neurotransmitter receptors contains over 100 members [19]. As these proteins are particularly associated with synaptic plasticity, it would be desirable to increase the number of proteins it is possible to describe in simulations.

As each protein has a number of binding sites, a combinatorially large number of complexes can arise, meaning that a correspondingly large number of reactions are needed to describe the dynamics of all possible complexes. By specifying rules whose elements are fragments of complexes, rule-based languages and simulators, such as Kappa [6] or BioNetGen [8], obviate the need to specify reactions for all possible complexes. These rules are simulated using a method similar to Gillespie’s stochastic simulation method for reactions [6]. Kappa, along with curation of proteomic data, has been used to predict the sizes of clusters of proteins in the postsynaptic proteins [22].

Compartmental models of electrical activity in neurons split the neuronal morphology into a number (ranging from 1 to around 1000) of compartments, and specify the dynamics of the membrane potential in each compartment in terms of coupled ordinary differential equations (ODEs) [11, 13, 24]. Quantities beyond the membrane potential can also be modelled, e.g. intracellular calcium concentration and concentrations of a few other molecules such as buffers and pumps. Various packages can generate and solve the equations underlying compartmental models from various model description languages, for example NEURON [3], MOOSE [20] and PSICS [2].

We present an algorithm which integrates rule-based models into compartmental models of neurons. To be sure of understanding a simple, yet interesting, case, we limit ourselves to considering isolated postsynaptic proteomes in a neuron of arbitrary morphology. Although of interest, we do not consider diffusion of molecules within the neuron. We implement the algorithm by integrating the SpatialKappa rule-based simulator [23] into the NEURON simulator [3]. We validate the combined simulator against stand-alone NEURON, and demonstrate how the system allows simulation of complex models.

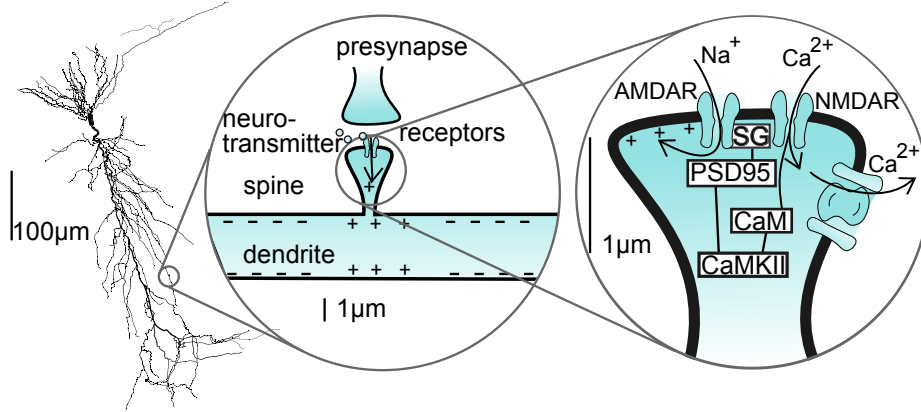


Fig. 1. The system to be simulated depicted on the scale of a whole neuron (left), a spine (middle) and the postsynapse (right). Scale bars are approximate. In the postsynapse, the rectangular boxes represent intracellular molecules: SG – stargazin; CaM – calmodulin; CaMKII – Ca^{2+} /calmodulin-dependent protein kinase II.

2 Simulation method

2.1 The system to be simulated

An example of the type of system to be simulated is shown in Figure 1. There is a neuron (left) of arbitrary complexity, upon which are located a number of synapses. The excitatory synapses are generally located on synaptic spines, small protuberances from the neuron whose narrow necks limit, to an extent, diffusion of ions and molecules between the spine head and the rest of the neuron (middle). The synapse contains a postsynaptic proteome of arbitrary complexity (right). Firing in presynaptic neurons causes release of neurotransmitter, which, after diffusing across the synaptic cleft, binds to AMPARs and NMDARs. The AMPARs open and close on a sub-millisecond timescale, allowing a net flow of positive ions into the cell. This flow charges the membrane locally and flows to other parts of the neuron, where it also charges the membrane (middle, Figure 1). The NMDARs open and close with slower dynamics, and allow Ca^{2+} ions to flow into the spine. The Ca^{2+} ions bind to various proteins such as calmodulin. Calmodulin with calcium bound may then bind to Ca^{2+} /calmodulin-dependent protein kinase II (CaMKII), initiating signalling known to be critical for the induction of LTP and LTD.

We will first describe the equations needed to simulate this system deterministically, and then describe the changes we propose to incorporate rule-based models.

2.2 Deterministic description of electrical and chemical activity in a neuron

To describe electrical activity in the neuron, the neuron is split into compartments, each of which should be small enough so as to be approximately isopotential. Apart from the root compartment, each compartment has a parent, and a compartment may have one or more children. The equation for the membrane potential V_i in compartment i derives from Kirchoff's current law:

$$C_i \frac{dV_i}{dt} = \sum_{j \in \mathcal{N}_i} \frac{d_{ij}(V_j - V_i)}{4R_a l_{ij}^2} - \sum_S (I_{S,i}^{\text{chan}} + I_{S,i}^{\text{pump}}) \quad (1)$$

The left hand side is the current per unit membrane area charging or discharging the membrane; C_i is the specific membrane capacitance in compartment i . The first term on the right hand side describes current flow into compartment i from its neighbours $\mathcal{N}_i$; R_a is the intracellular resistivity, l_{ij} is the path length between the midpoints of i and j and d_{ij} is the mean diameter of the path. The second term on the right hand side is the total transmembrane current density in compartment i carried by various species of ion S via ion channels ($I_{S,i}^{\text{chan}}$) and membrane pumps ($I_{S,i}^{\text{pump}}$), which act to maintain concentration differences between the intracellular and extracellular space. The minus sign reflects the conventions that inward current is negative and the extracellular space is regarded as ground.

The channel current density of species S is through types j of ion channel is

$$I_{S,i}^{\text{chan}} = \sum_j g_{ji} (O_{ij}, t) f_{S,i}(V_i, [S]_i, [S]_e) \quad (2)$$

where g_{ji} is the ion channel conductance, which may be a function of time or a state variable O_{ij} , and $f_{S,j}$ and is a function describing the I - V characteristic of current flow of ions of type S through channel j , which may depend on $[S]_i$, the intracellular concentration of S in compartment i , and $[S]_e$, the extracellular concentration of S , which is assumed to be constant. A normalised Goldman-Hodgkin-Katz (GHK) current equation [24] can be used for $f_{S,j}$. For channels through which calcium flows, the typically large ratio between intracellular and extracellular calcium concentrations means this function depends quite strongly on the intracellular calcium concentration $[\text{Ca}^{2+}]$, but in channels not permeable to calcium it is usual to use a linear approximation $V_i - E_j$, where E_j is the reversal potential for that channel. By removing the dependence on intracellular concentrations $[S]_i$, this approximation allows currents carried by ions other than calcium to be lumped together in a nonspecific ion category.

The state variable O_{ij} is the number of type j ion channels in compartment i which are in an open conformation. It is modelled as the occupancy of a state of a Markov process with membrane potential-dependent transition rates. For small number of channels the Markov process is simulated stochastically, but with large numbers of channels the system is practically deterministic and the master equation of the Markov process is simulated using ODEs.

The dynamics of the intracellular concentrations of ions can be modelled using further ODEs. The rate of change of $[S]_i$ depends on the total transmembrane current density of S ($I_{S,i}^{\text{chan}} + I_{S,i}^{\text{pump}}$) and consumption and release by intracellular reactions:

$$\frac{d[S]_i}{dt} = -\frac{a_i}{z_S F v_i} (I_{S,i}^{\text{chan}} + I_{S,i}^{\text{pump}}) + \sum_r J_{S,r,i} \quad (3)$$

where a_i is the surface area of the compartment, v_i is the notional volume of the compartment, z_S is the valency of ion S , and F is Faraday's constant. The volume may describe the volume of a thin submembrane shell rather than the volume of the whole compartment. The term $\sum_r J_{S,r,i}$ describes the net flux of S due to intracellular reactions r . It arises from treating the intracellular reactions in compartment i as a set of kinetic schemes:



The flux corresponding to this reaction would be:

$$J_{S,r,i} = -k_r [S]_i [T]_i + k_{-r} [S \cdot T]_i$$

Thus the whole system is defined by a system of ODEs (Equation 12, Appendix A.1).

2.3 Simulation of Deterministic Variables

Simulators of deterministic electrical and chemical activity in neurons, such as NEURON [3], solve the set of coupled ODEs described above by gathering the variables V_i , $[S]_i$ and other state variables into one state vector $\mathbf{x}$ and solving the ODE system

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{F}(\mathbf{x}) + \mathbf{b}(t) \quad (5)$$

where $\mathbf{F}(\mathbf{x})$ is the rate of change of each state variable and $\mathbf{b}(t)$ is a time dependent forcing input. In principle $\mathbf{F}(\mathbf{x})$ depends on all variables, though the structure of compartmental model means each element of $\mathbf{F}(\mathbf{x})$ depends on only a few elements of $\mathbf{x}$. The simplest stable method for solving these equations is implicit, backward Euler integration, in which the derivative is evaluated at $t + \Delta t$, the end of the time step:

$$\frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} = \mathbf{F}(\mathbf{x}(t + \Delta t)) + \mathbf{b}(t + \Delta t) \quad (6)$$

Taylor expanding the right hand side of (6) in Δt and rearranging gives the equation for one update step

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \left(I - \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \Delta t \right)^{-1} (\mathbf{F}(\mathbf{x}(t)) + \mathbf{b}(t)) \Delta t \quad (7)$$

where $\partial \mathbf{F} / \partial \mathbf{x}$ is the Jacobian matrix at time t , which can be computed numerically, or analytically for efficiency. To optimise simulation speed, the time step Δt may vary depending on the rate of change of the variables, but the essential point is that all variables are updated simultaneously.

2.4 Modifications to accommodate rule-based simulation

In order to integrate fixed-step simulation of continuous variable described by ODEs with discrete variables described by stochastic, rule-based models, we use principles already used in the hybrid simulations of systems of chemical reactions [12]. The state variables are partitioned into continuous variables, which are updated at fixed intervals of Δt by an ODE solver and discrete variables which are updated asynchronously by the rule-based solver. Some “bridge” variables are referred to by both solvers. The combined simulation algorithm must ensure that the two solvers are synchronised appropriately and that conversions between continuous and discrete quantities are made.

In the case of neural simulations the membrane potential would be a continuous variable, intracellular molecules such as calmodulin and CaMKII would be stochastic variables and the intracellular calcium concentration in spine would be a stochastic bridge variable.

Conversions The rule-based simulator manages a subset of elements of $\mathbf{x}$. For elements which are species concentrations, the conversion formula is:

$$X_k = N_A v_k x_k \quad (8)$$

where x_k is the concentration of the molecule and X_k is the equivalent number of molecules in the volume v_k and N_A is Avogadro’s constant. We assume that rate coefficients for reactions based on concentrations are supplied and these must be converted into reactions for species number as described in Appendix A.1. State variables (for example the state of channels) may also be controlled by the rule-based simulator, and here different conversions formulae apply.

Creation and destruction rules for bridging variables To simulate the transmembrane currents in the rule-based solver creation and destruction rules need to be written which are equivalent to $I_{S,i}^{\text{chan}}$ and $I_{S,i}^{\text{pump}}$ in Equation (3). These rules are:

$$\xrightarrow{-a_i I_{S,i}^{\text{chan}} N_A / z_S F} S \quad \text{and} \quad S \xrightarrow{-a_i I_{S,i}^{\text{pump}} N_A / z_S F} \quad (9)$$

Update and synchronisation The procedure for updating the time from t to $t + \Delta t$ is:

1. Pass all relevant continuous variables to the rule-based simulator
2. Run the rule-based solver from t to $t + \Delta t$
3. Compute the net change ΔS_i^{tot} in total number of each bridging species S (including in any complexes) over time step and convert this back to a current $I_{S,i}^{\text{chan}} + I_{S,i}^{\text{pump}} = -\Delta S_i^{\text{tot}} z_S F / a_i N_A$
4. Treating these currents of bridge variables as constants in $\mathbf{b}(t)$, update the continuous variables according to the update step (7).

When running the rule-based model, it will not stop precisely on the boundary of the time step since the update times are generated stochastically. To deal with this problem, the event time in the rule-based component is computed before updating the variables. As soon as the event time is after the end of the deterministic step, that event is thrown away, as justified in Appendix A.3.

3 Implementation

We have implemented an approximate version of the algorithm described in the previous section by linking the Java-based SpatialKappa implementation of the Kappa language [23] to version 7.4 of NEURON, which allows Reaction-Diffusion equations to be specified in Python [17]. Our development code is available at <http://github.com/davidcsterratt/KappaNEURON>. To do this we have extended the SpatialKappa simulator to use the py4j¹ package to run SpatialKappa Java objects in Python. The wrapper system in NEURON 7.4 allows us to override NEURON’s built-in non-voltage set-up and fixed solve callback functions with ones that call the SpatialKappa simulator at each time step.

Two parts of our implementation approximate the algorithm: (i) Pending work on a programmatic interface to rule creation in SpatialKappa, the creation rule in (9) is instead achieved by drawing the amount of species S to be added throughout the time step from a Poisson distribution with mean $I_{S,i}^{\text{chan}} N_A / z_S F \Delta t$ and adding this to the SpatialKappa simulation midway through the time step. (ii) The pump current I^{pump} is not passed back from SpatialKappa to NEURON because code to generate the query of ΔS^{tot} still needs to be written, and we already know the ion channel contribution from (i). We expect I^{pump} to be small compared to channel currents, so the effect of neglecting it will be small.

In order to specify a model the Kappa component is specified in a separate file. Figure 2 shows an example of a simple calcium pump specified in Kappa. This file is then linked into the NEURON simulation as demonstrated in Figure 3. The mechanisms specified in the Kappa file take over all of NEURON’s handling of molecules in the cytoplasm of chosen sections.

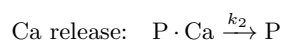
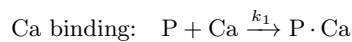
4 Results

4.1 Validation

In order to validate the implementation, we simulated a simple model using standard NEURON and the integrated SpatialKappa-NEURON simulator. There is single compartment containing a transmembrane calcium pump, described by the kinetic scheme in Figure 2A. Calcium flows into the compartment via a calcium conductance of magnitude g_{Ca} , the current I_{Ca} being determined by the GHK current equations. The volume of the compartment ranges from $0.01\text{--}1\mu\text{m}^3$, typical of volumes of spine heads in the vertebrate CNS [10]. The compartment is modelled as a cylinder with diameter ranging from $0.2\text{--}1\mu\text{m}$ and length $1\mu\text{m}$.

The kinetic scheme (Figure 2A) along with the calcium influx can be converted to a pair of ordinary differential equations. This mechanism was implemented using NEURON’s NMODL language for describing membrane and intracellular mechanisms. The model compartment was held under voltage clamp control and the calcium conductance g_{Ca} is zero apart from during the voltage pulse. The results of this simulation are shown the blue traces in Figure 4.

¹ <http://py4j.sourceforge.net/>

A**B**

```
## File caPump.ka - Simple calcium pump

## Agent declarations, showing the agent names and binding sites
%agent: ca(x)    # Calcium with binding site
%agent: P(x)     # Pump molecule with binding site

## Variable declarations
%var: 'vol' 1      # Volume in um3
%var: 'NA' 6.02205E23 # Avagadro's constant
# Concentration of one agent in the volume in mM
%var: 'agconc' 1E18/('NA' * 'vol')
# Rate constants in /mM-ms or /ms, depending on the number of
# complexes on LHS of rule
%var: 'k1' 0.001    # /mM-ms
%var: 'k2' 1        # /ms

## Rules
# Note the scaling of the rate constant of the bimolecular reaction
'ca binding' ca(x), P(x)    -> ca(x!1), P(x!1) @ 'k1' * 'agconc'
'ca release' ca(x!1), P(x!1) -> P(x) @ 'k2'

## Initialisation
%init: 1000 ca(x)
%init: 10000 P(x)

## Observations
%obs: 'ca'    ca(x)          # Free Ca
%obs: 'P-Ca'  ca(x!1), P(x!1) # Bound Ca-P
%obs: 'P'     P(x)           # Free P
```

Fig. 2. Example of a Kappa file for a simple calcium pump. **A** Rules defining the calcium pump, which in this simple case can be interpreted as kinetic equations. Ca represents intracellular calcium, P represents a pump molecule in the membrane, $P \cdot \text{Ca}$ is the pump molecule bound by calcium and k_1 and k_2 are rate coefficients. **B** The Kappa file implementation of this system, simulated in a volume of $1\mu\text{m}^3$. Note the conversion of the forward rate coefficient from units of $\text{mM}^{-1} \text{ms}^{-1}$ to ms^{-1} . For an introduction to the Kappa language, see the short description at <http://www.kappalanguage.org/syntax.html>.

```

from neuron import *
import KappaNEURON

## Create a compartment
sh = h.Section()
sh.insert("pas")          # Passive channel
sh.insert("capulse")      # Code to give Ca pulse
# This setting of parameters gives a calcium influx and pump
# activation that is scale-independent
sh.gcalbar_capulse = gcalbar*sh.diam

## Define region where the dynamics will take place
r = rxd.Region([sh], nrn_region='i')

## Define the species, the ca ion (already built-in to NEURON), and the
## pump molecule. These names must correspond to the agent names in
## the Kappa file.
ca = rxd.Species(r, name='ca', charge=2, initial=0.0)
P = rxd.Species(r, name='P', charge=0, initial=0.2)

## Create the link between the Kappa model and the species just defined
kappa = KappaNEURON.Kappa([ca, P], "caPump.ka", r)

## Transfer variable settings to the kappa model
vol = sh.L*numpy.pi*(sh.diam/2)**2
kappa.setVariable('k1', 47.3)
kappa.setVariable('k2', gamma2)
kappa.setVariable('vol', vol)

## Run
init()
run(30)

```

Fig. 3. Extract of Python code to link the Kappa file shown in Figure 2 into a compartment in NEURON.

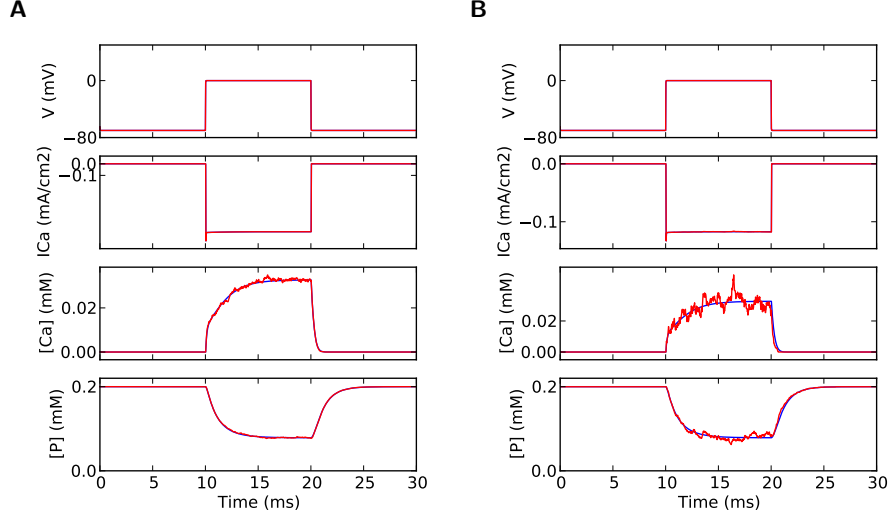


Fig. 4. Reference simulation. **A**, Traces generated by NEURON with NMODL (blue) and NEURON/SpatialKappa combination (red) when the diameter is $1\mu\text{m}$. **B**, The same simulations but with a diameter of $0.2\mu\text{m}$. There is more noise evident in the combined simulation due to the smaller number of ions involved.

When the voltage is at -70 mV there is no calcium current. When the voltage goes up to 0 mV , the conductance takes on a positive value, and the calcium current begins to flow. This causes a sharp rise in intracellular calcium concentration, which, because of the GHK current equation, reduces the calcium current slightly, accounting for the initial “blip” in the calcium current. As the calcium concentration increases, it starts binding to the pump molecule, depleting the amount of the free pump molecule. Once the voltage pulse is over, the calcium influx stops, and the remaining free calcium is taken up quickly by the pump. The pump-calcium complex dissociates at a slower rate.

The red trace in Figure 4A shows a simulation of the same system, in the NEURON-SpatialKappa implementation in a cell with a diameter of $1\mu\text{m}$. The traces appear very similar apart from some random fluctuations, particularly in the trace of calcium. The red trace in Figure 4B shows a simulation in a spine with a diameter of $0.2\mu\text{m}$ (i.e. $1/25$ of the volume of the simulation in A). Due to the smaller numbers of ions involved, the fluctuations are relatively bigger.

4.2 Demonstration simulations

To demonstrate the utility of integrated electrical and rule-based neuronal models, we constructed a model of a subset of the synaptic proteome, with a focus on the signal processing at the early stages of the CaM-CaMKII pathway. We encoded in Kappa published models of: the dynamics of NMDARs [26]; binding of calcium with calmodulin and binding of calmodulin-calcium complexes to

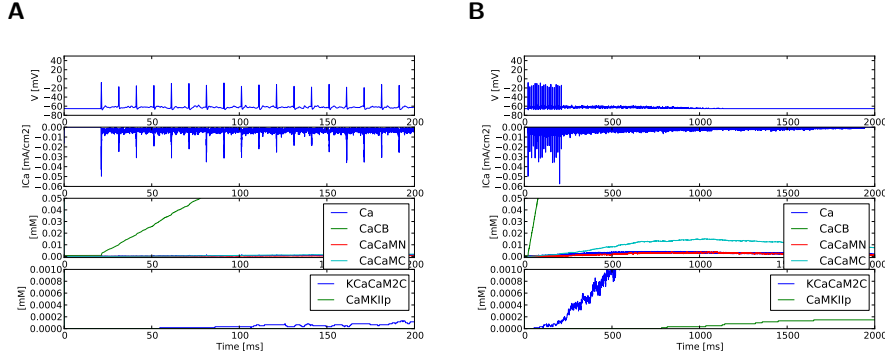


Fig. 5. Demonstration simulations. **A.** The first 200ms of the simulation. **B.** The first 2000ms of the simulation. Labels: Ca, free calcium; CaCB, calcium bound to calbindin; CaCaMN, calcium bound to the N lobe of calmodulin; CaCaMC, calcium bound to the C lobe of calmodulin; KCaCaM2C, CaMKII bound to calmodulin with two calcium ions on its C lobe; and CaMKIIp, phosphorylated CaMKII.

CaMKII [18]; and binding of calcium to a calbindin buffer [7]. We embedded these linked models into a very simple model of a synaptic spine, comprising head and neck compartments, connected to a dendrite. As well as the NMDAR, modelled in Kappa, there was an AMPAR in the spine head, and backpropagating spikes were modelled by inserting standard Hodgkin-Huxley ion-channels [11] in the dendritic membrane. There were also 50 other synaptic inputs onto the dendrite, though these did not contain the rule-based model.

Figure 5 shows results from one simulation at short (200ms) and long (2000ms) time scales. Synaptic stimulation of the detailed synapse releases glutamate, which binds to NMDARs according to the kinetics of [26]. It can be seen that the calcium signal entering through the NMDARs is taken up by the calbindin and calmodulin buffers. When calcium is bound to the N or C lobes of the calmodulin (CaM) buffer, the bound CaM can bind to CaMKII, the rate depending on the number of calcium ions bound and on which lobe they are bound to. This CaMKII-CaM- Ca^{2+} complex can then be phosphorylated.

5 Discussion

We have presented a method for integrating a stochastically simulated rule-based model of proteins in a sub-micron sized region of a neuron into a compartmental model of electrical activity in the whole neuron. This allows the biochemical interactions between binding sites on proteins to specified using a tractable number of rules, with the simulator taking on the work of tracking which complexes are present at any point during the simulation.

Recently Mattioni and Le Novère [16] have integrated the ECell simulator with NEURON. The approach we have taken is very similar to theirs, though with two differences. Firstly we have integrated a rule-based simulator. This has

the advantage that the interactions between the combinatorially large numbers of complexes present in the PSD can be specified using a tractable number of rules, though this limits the simulation method to the sequential style of Gillespie’s stochastic simulation algorithm, and does not allow for any of the approximations that increase the algorithm’s efficiency. Secondly Mattioni and Le Novère get the ODE-based solver to handle calcium, whereas we handle it in the rule-based solver. Again our approach is less efficient computationally, but it ensures that all biochemical quantities are consistent and avoids having to make any assumptions about the relative speeds of processes.

Our approach allows us to model at a considerable level of detail. For example the conformation of NMDARs may be part of the biochemical model, allowing proteins in the PSD (e.g. calmodulin bound to calcium) to modulate the state of the channel [26]. We can also use one rule-based scheme to model both the presynapse and the postsynapse, which could help to understand transynaptic signalling via molecules such as endocannabinoids [4].

At present our implementation of our algorithm in SpatialKappa and NEURON is publicly available but experimental, pending further development, validation and simplification of the syntax required in the NEURON Python code to link to the variables in the Kappa file. In the longer term we would like to make available to NEURON the capability of SpatialKappa to simulate diffusion of complexes using a voxels.

A Appendix

A.1 Scaling

The variables in deterministic simulations of biochemical reactions in neurons (e.g. [1]) are intensive quantities such as concentration and current density whereas in stochastic models the variables are extensive quantities such the number of molecules in a volume. Thus to compare deterministic and stochastic simulations, intensive and extensive quantities need to be interconverted. A conversion formula can be derived by considering this bimolecular kinetic scheme in which k is the rate coefficient:



In deterministic reactions the quantities of the reactants A and B and the product $A \cdot B$ are described by the concentrations $[A]$, $[B]$ and $[A \cdot B]$, and the kinetic scheme can be translated into an ordinary differential equation (ODE) whose variables are these concentrations:

$$\frac{d[A \cdot B]}{dt} = k[A][B] \quad (11)$$

Typical units for the rate coefficient k are $M^{-1}s^{-1}$. If this reaction occurs in a volume v , the number of molecules $|A|$, $|B|$ and $|A \cdot B|$ can be found from the concentrations by multiplying each concentration by $N_A v$, where N_A is Avogadro’s

number. Making this substitution yields the following ODE whose variables are numbers of molecules:

$$\frac{d|A \cdot B|}{dt} = \gamma |A| |B| \quad (12)$$

where the converted rate coefficient $\gamma = k/N_A v$ is dimensionless. In general for an equation with n reactants, the relation between rate coefficients for numbers of molecules and concentrations is $\gamma = (N_A v)^{-n+1} k$.

A.2 Kappa simulation method

In stochastic simulation methods such as Gillespie's direct method [9] or the rule-based Kappa simulation language [6], variables are updated asynchronously and there is no global Δt . To illustrate the principle of how Gillespie's direct method [9] works, consider the following kinetic scheme describe a calcium pump in Figure 2A. Here Ca represents intracellular calcium, P represents a pump molecule in the membrane, $P \cdot Ca$ is the pump molecule bound by calcium and k_1 and k_2 are rate coefficients, which are rescaled to the variables γ_1 and γ_2 as explained earlier. The following procedure is used to simulate this scheme using the Gillespie method:

1. Compute the *propensities* of the reactions $a_1 = \gamma_1 |Ca| |P|$ and $a_2 = \gamma_2 |Ca \cdot P|$
2. The total propensity is $A = a_1 + a_2$
3. Pick reaction R_i with probability a_i/A
4. Pick time to reaction $T = -(\ln r)/A$.
5. Goto 1

Kappa uses an analogous method, but applied to rules that are currently active. Both methods are event-based rather than time-step based.

A.3 Justification for throwing away events

To justify throwing away events occurring after a time step ending at $t + \Delta t$, we need to show that distribution of event times is the same in two cases:

1. The event time T is drawn from an exponential distribution $A \exp(-AT)$ (for $T > 0$), where A is the propensity.
2. An event time T_0 is drawn as above. If T_0 is the exceeds the end of the time step Δt , accept $T = T_0$ as the event time. If $T_0 > \Delta t$, throw away this event time and sample a new interval T_1 from an exponential distribution with a time constant of A . Set the event time to $T = \Delta t + T_1$.

To show this, consider that in case 2, the overall distribution is:

$$\begin{aligned} P(\text{event at } T \leq \Delta t) &= A \exp(-AT) \\ P(\text{event at } T > \Delta t) &= P(\text{survival to } \Delta t) P(\text{event at } T_1) \\ &= \exp(-A\Delta t) A \exp(-A(T - \Delta t)) = A \exp(-AT) \end{aligned}$$

Here we have used $T_1 = T - \Delta t$. Thus the distribution is the same as in case 1.

References

1. Bhalla, U.S., Iyengar, R.: Emergent properties of networks of biological signalling pathways. *Science* 283, 381–387 (1999)
2. Cannon, R.C., O'Donnell, C., Nolan, M.F.: Stochastic ion channel gating in dendritic neurons: Morphology dependence and probabilistic synaptic activation of dendritic spikes. *PLoS Comput Biol* 68, e1000886 (2010)
3. Carnevale, T., Hines, M.: *The NEURON Book*. Cambridge University Press, Cambridge, UK (2006)
4. Castillo, P.E., Younts, T.J., Chávez, A.E., Hashimoto, Y.: Endocannabinoid signaling and synaptic function. *Neuron* 761, 70–81 (2012)
5. Collins, M.O., Husi, H., Yu, L., Brandon, J.M., Anderson, C.N.G., Blackstock, W.P., Choudhary, J.S., Grant, S.G.N.: Molecular characterization and comparison of the components and multiprotein complexes in the postsynaptic proteome. *Journal of Neurochemistry* 97, 16–23 (2006)
6. Danos, V., Feret, J., Fontana, W., Krivine, J.: Scalable simulation of cellular signaling networks. In: Shao, Z. (ed.) *Programming Languages and Systems, Lecture Notes in Computer Science*, vol. 4807, chap. 10, pp. 139–157. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
7. Faas, G.C., Raghavachari, S., Lisman, J.E., Mody, I.: Calmodulin as a direct detector of Ca^{2+} signals. *Nature Neuroscience* 143, 301–304 (2011)
8. Faeder, J., Blinov, M., Hlavacek, W.: Rule-based modeling of biochemical systems with BioNetGen. In: Maly, I.V. (ed.) *Systems Biology, Methods in Molecular Biology*, vol. 500, pp. 113–167. Humana Press (2009)
9. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry* 81, 2340–2361 (1977)
10. Harris, K.M., Kater, S.B.: Dendritic spines: Cellular specializations imparting both stability and flexibility to synaptic function. *Annual Review of Neuroscience* 17, 341–371 (1994)
11. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 117, 500–544 (1952)
12. Kiehl, T.R., Mattheyses, R.M., Simmons, M.K.: Hybrid simulation of cellular behavior. *Bioinformatics* 203, 316–322 (2004)
13. Koch, C.: *Biophysics of Computation*. Oxford University Press, New York (1999)
14. Lisman, J.E., Zhabotinsky, A.M.: A model of synaptic memory: a CaMKII/PP1 switch that potentiates transmission by organizing an AMPA receptor anchoring assembly. *Neuron* 312, 191–201 (2001)
15. Martin, S.J., Grimwood, P.D., Morris, R.G.M.: Synaptic plasticity and memory: An evaluation of the hypothesis. *Annual Review of Neuroscience* 231, 649–711 (2000)
16. Mattioni, M., Le Novère, N.: Integration of biochemical and electrical Signaling-Multiscale model of the medium spiny neuron of the striatum. *PLoS ONE* 87, e66811+ (2013)
17. McDougal, R.A., Hines, M.L., Lytton, W.W.: Reaction-diffusion in the NEURON simulator. *Frontiers in Neuroinformatics* 728 (2013)
18. Pepke, S., Kinzer-Ursem, T., Mihalas, S., Kennedy, M.B.: A dynamic model of interactions of Ca^{2+} , calmodulin, and catalytic subunits of Ca^{2+} /calmodulin-dependent protein kinase II. *PLoS Computational Biology* 62 (2010)

19. Pocklington, A.J., Cumiskey, M., Armstrong, J.D., Grant, S.G.N.: The proteomes of neurotransmitter receptor complexes form modular networks with distributed functionality underlying plasticity and behaviour. *Molecular Systems Biology* 21 (2006)
20. Ray, S., Bhalla, U.S.: PyMOOSE: Interoperable scripting in python for MOOSE. *Frontiers in Neuroinformatics* 2 (2008)
21. Smolen, P., Baxter, D.A., Byrne, J.H.: A model of the roles of essential kinases in the induction and expression of late long-term potentiation. *Biophysical Journal* 908, 2760–2775 (2006)
22. Sorokina, O., Sorokin, A., Armstrong, J.D.: Towards a quantitative model of the post-synaptic proteome. *Molecular Biosystems* 7, 2813–2823 (2011)
23. Sorokina, O., Sorokin, A., Armstrong, J.D., Danos, V.: A simulator for spatially extended kappa models. *Bioinformatics* pp. 3105–3106 (2013)
24. Sterratt, D., Graham, B., Gillies, A., Willshaw, D.: *Principles of Computational Modelling in Neuroscience*. Cambridge University Press (2011)
25. Stiles, J.R., Bartol, T.M.: Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. In: De Schutter, E. (ed.) *Computational Neuroscience: Realistic Modeling for Experimentalists*, chap. 4, pp. 87–127. CRC Press, Boca Raton, FL (2001)
26. Urakubo, H., Honda, M., Froemke, R.C., Kuroda, S.: Requirement of an allosteric kinetics of NMDA receptors for spike timing-dependent plasticity. *Journal of Neuroscience* 2813, 3310–3323 (2008)
27. Zeng, S., Holmes, W.R.: The effect of noise on CaMKII activation in a dendritic spine during LTP induction. *Journal of Neurophysiology* 1034, 1798–1808 (2010)