Modeling extracellular fields for a three-dimensional network of cells using NEURON

Appukuttan, Shailesh; Brain, Keith L; Manchanda, Rohit

DOI:
10.1016/j.jneumeth.2017.07.005
10.1016/j.jneumeth.2017.07.005

Modeling Extracellular Fields for a Three-Dimensional Network of Cells using NEURON

Shailesh Appukuttan[1,*], Keith L. Brain[2], Rohit Manchanda[1]

[1] Department of Biosciences and Bioengineering, Indian Institute of Technology Bombay, Mumbai, Maharashtra, India

[2] School of Clinical and Experimental Medicine, College of Medical and Dental Sciences, University of Birmingham, Birmingham, UK

Supplementary Document

S1. Toy Neuron Model

We present here further details about the development of the toy neuron model (see section 3.5). NEURON codes for model development and for the linking of extracellular spaces, via the technique presented in this work, have been provided along with a flowchart that summarizes the various steps involved.

S1.1. Model Biophysics

As seen in Fig. 13, each neuron consists of a soma, axon, and two proximal dendrites, each of which divides into two distal dendrites. The soma is spherical and is attached to the axon at one end, and the two proximal dendrites at the other end. The axon has been modeled as being long and thin, with uniform diameter. The diameter of the proximal and distal dendrites is tapered with increasing distance from the soma. The morphological parameters for the neuron model are presented in table S1. The soma and the axon were endowed with Hodgkin-Huxley (HH) channels so as to enable them to produce action potentials. The dendrites were provided only passive leak channels. The reversal potential of the leak channels was set to −65 mV, similar to that for the HH

*Current Address: *Neuroinformatics group, Unité de Neurosciences, Information et Complexité, Centre National de la Recherche Scientifique, Gif sur Yvette, France*

Table S1: Morphological parameters for the toy neuron model

Section	Parameter	Units	Value
soma	length	μm	20
	diameter	μm	20
	# of segments	-	1
axon	length	μm	1000
	diameter	μm	5
	# of segments	-	1001
proximal dendrites	length	μm	200
	diameter	μm	2 → 1
	# of segments	-	201
distal dendrites	length	μm	100
	diameter	μm	1 → 0.2
	# of segments	-	101

channels, thereby resulting in a resting membrane potential of around −65 mV for the neuron.

S1.2. NEURON Code For Creating The Individual Neurons

The NEURON code presented below constructs a cell template, based on the biophysical parameters specified in the previous section. This template is then utilized to create two instances of the toy neuron.

```
// ******************* Start of Template *******************
begintemplate dummy_neuron
     public soma, axon, p_dend, d_dend
     create soma, axon, p_dend[1], d_dend[1]

     proc init() {
```

```
create soma, axon, p_dend[2], d_dend[4]
soma {
     L = 20
     nseg = 1
     diam = 20
     insert hh
}
axon {
     L = 1000
     nseg = 1001
     diam = 5
     insert hh
}
forsec "p_dend" {
     L = 200
     nseg = 201
     diam(0:1) = 2:1
     insert pas
     e_pas = -65
}
forsec "d_dend" {
     L = 100
     nseg = 101
     diam(0:1) = 1:0.2
     insert pas
     e_pas = -65
}

//Connecting all the sections together
//-- Axon to Soma
connect axon(1), soma(0)
//-- Primary Dendrites to Soma
for i = 0, 1 {
     connect p_dend[i](0), soma(1)
}
//-- Secondary Dendrites to Primary Dendrites
for i = 0, 1 {
```

```
            connect d_dend[i*2](0), p_dend[i](1)
            connect d_dend[(i*2)+1](0), p_dend[i](1)
        }
    }
endtemplate dummy_neuron
// ******************** End of Template ********************

//Creating the Neurons
objref neuron[2]
for i = 0, 1 {
    neuron[i] = new dummy_neuron()
}
```

S1.3. NEURON Code For Linking The Extracellular Fields

The NEURON code below implements the coupling of extracellular fields of the two adjacent neurons, at regions where they are in close spatial proximity.

```
objref slist1, slist2, list_temp1, list_temp2
objref gmat, cmat, bvec, e, xl, layer, sl, lm

RATIO_ra_by_re = 0.01                                   // Specify
Re = Ra/RATIO_ra_by_re                                  // Ohm.cm
xraxial_value = Re*1e-6/(PI*((diam/2)^2)*1e-8)          // MOhm/cm

Re_Ohm = Re*1e-6*L*1e-4/(PI*((diam/2)^2)*1e-8)          // MOhm
rlink = Re_Ohm/nseg                                     // MOhm
glink = 1000/rlink                                      // nS
ge_value = (glink*0.1)/area(0.5)                        // S/cm2

proc setExtra() {
    forall {
        insert extracellular
        xc[0] = 0
        xc[1] = 0
```

```
        xg[0] = 1e-9   // Infinite Resistance
        xg[1] = 1e-9   // Infinite Resistance

        xraxial[0] = xraxial_value              // MOhm/cm
        xraxial[1] = 1e9  // Infinite Resistance
    }

    print "_________________________________________"
    print "Extracellular Mechanism Inserted"
    print "_________________________________________"
}
setExtra()

proc setExtraLink() {
    // connected sections in neuron 1
    slist1 = new SectionList()
    // connected sections in neuron 2
    slist2 = new SectionList()

    // ensure that the order is the same
    neuron[0].axon slist1.append()
    neuron[1].axon slist2.append()
    neuron[0].soma slist1.append()
    neuron[1].soma slist2.append()
    neuron[0].p_dend[0] slist1.append()
    neuron[1].p_dend[0] slist2.append()
    neuron[0].d_dend[0] slist1.append()
    neuron[1].d_dend[0] slist2.append()
    neuron[0].d_dend[1] slist1.append()
    neuron[1].d_dend[1] slist2.append()

    nsegs = 0 // will contain total connected segs
    forsec slist1 {
        nsegs += nseg
    }
    print "_________________________________________"
```

```
print "Total Connected Segments = ", 2*nsegs
print "_________________________________________"

gmat = new Matrix(2*nsegs, 2*nsegs, 2)
cmat = new Matrix(2*nsegs, 2*nsegs, 2)
bvec = new Vector(2*nsegs)
xl = new Vector()
layer = new Vector(2*nsegs)
layer.fill(1)

forsec slist1 {
    for (x, 0) {
        xl.append(x)   // for neuron 1
        xl.append(x)   // for neuron 2
    }
}

e = new Vector(2*nsegs)
sl = new SectionList()

list_temp1 = new List()
forsec slist1 {
    for (x, 0) {
        list_temp1.append(new SectionRef())
    }
    slist1.remove()
}
list_temp2 = new List()
forsec slist2 {
    for (x, 0) {
        list_temp2.append(new SectionRef())
    }
    slist2.remove()
}

for i = 0, nsegs-1 {
    list_temp1.object(i).sec sl.append()
```

```
            list_temp2.object(i).sec sl.append()
        }

        ge = ge_value  // S/cm2

        for (i=0; i<(2*nsegs); i=i+2) {
            gmat.x[i][i] += ge
            gmat.x[i+1][i+1] += ge
            gmat.x[i][i+1] += -ge
            gmat.x[i+1][i] += -ge
        }

        lm = new LinearMechanism(cmat, gmat, e, bvec, sl, xl, layer)

        print "__________________________________________"
        print "Extracellular Connected Via Link"
        print "__________________________________________"
}
setExtraLink()
```

As described in section 3.5, the neurons have been oriented such that one primary dendrite of each of the two neurons, along with the distal dendrites emerging from them, are considered to be at a sufficiently large distance from the other neuron to not have their extracellular spaces affected directly by it. This constraint was imposed to demonstrate an element of heterogeneity in the coupling of the extracellular regions of the two neurons. The exact nature of coupling between the two neurons can be made as complex as necessary, with the technique being amenable to any such requirements. The toy neuron model simply being an illustrative example, we have chosen to limit its complexity here by assigning equal coupling strengths between the various regions at which the two neurons are in close spatial proximity. This assumes that the two neurons maintain a constant spatial separation throughout their morphology. If the need arises for differential coupling, potentially based on the distance of separation and/or other factors, the implementation requires only a minor modification to reflect the same. In the above implementation, this would translate into the dynamic evaluation of the parameter '*ge_value*' for each segment, with the value being a function of the distance of separation.

In the case of a network/syncytial model, variations in the boundary conditions are easily feasible within the presented framework. The extracellular boundary conditions, in NEURON, can be altered as required by modifying the values of the resistive, capacitive and voltage components (see Table 1 and Fig. 2) that define the extracellular features of each cell. Variations can be introduced, such as between peripheral and non-peripheral cells, or across the ends of a tissue/network, depending on the modeling requirements.

S1.4. Flowchart Describing Steps Followed For Toy Model

Fig. S1 shows a flowchart that summarizes the series of steps involved in developing the toy model. The steps involved in the coupling of the extracellular spaces of the two neurons, which form the crux of the present work, have been encapsulated within the dashed box. The only main prerequisite is to have the models of the individual neurons that need to be coupled extracellularly. The various stages, and also the terminology used for the various parameters (in red), are closely linked to the NEURON code presented in the previous section. Certain stages in the workflow have been represented via colored boxes, and these correspond to the stages where user input is required. The other stages are capable of continuing in their default configuration,

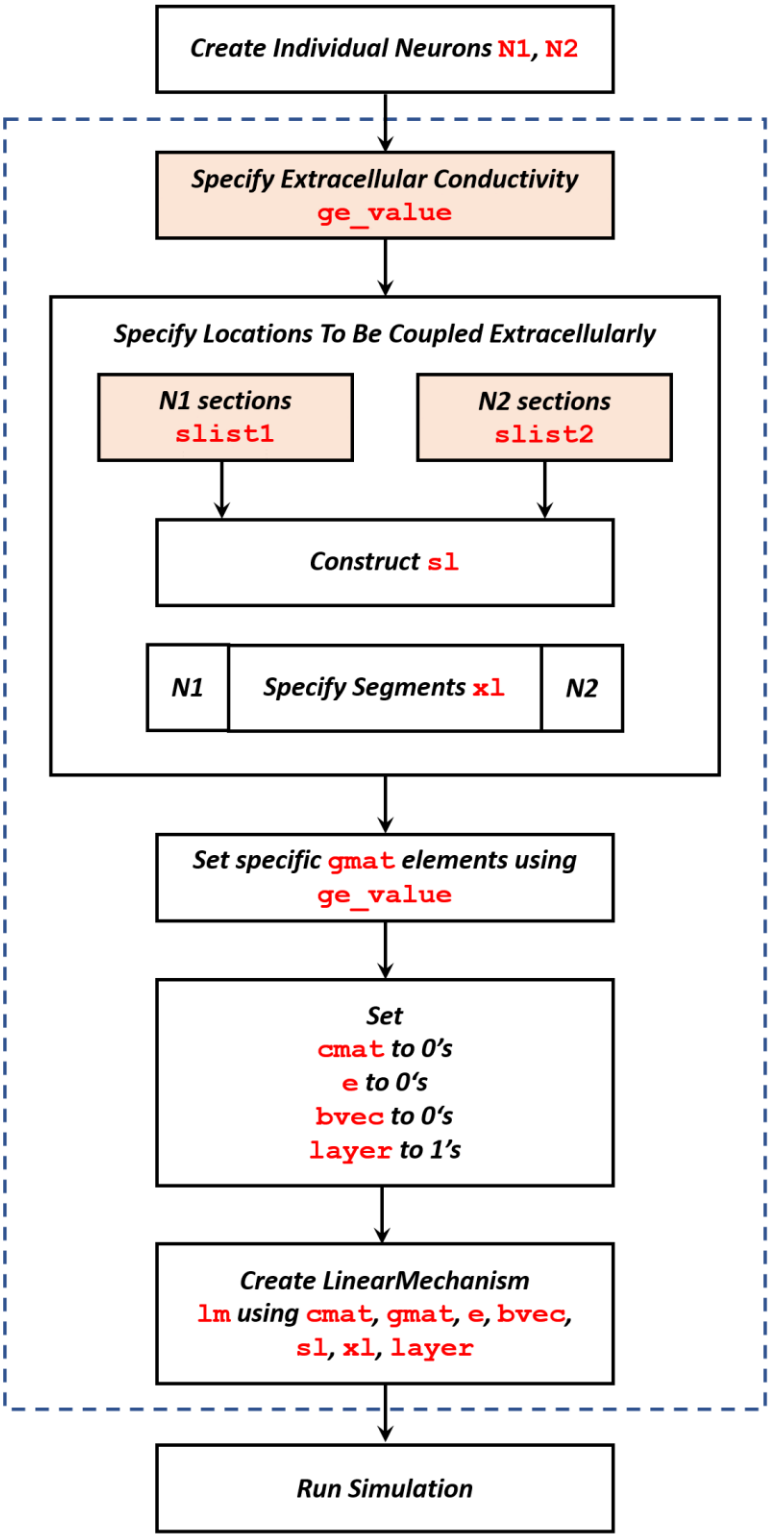

Figure S1: Flowchart summarizing the steps involved in developing the toy model. The dashed box encapsulates the steps involved in the coupling of the extracellular fields, and corresponds to the NEURON code presented in section S1.3. The colored boxes indicate the minimal stages where user input is required. The terms in red correspond to the variable names employed in the NEURON code.

coupling. This offers a degree of automation if the modeler simply wishes to reuse the existing extracellular coupling mechanism between different pairs of neurons. The process can be further streamlined by creating templates for the coupling mechanism and instantiating these wherever required. This would, in essence, be similar to creating templates for cells as employed in section S1.2, and may be advisable when handling large neuronal networks. But the wider capabilities and flexibility of the technique presented here is more evident when the modeler opts to explicitly tweak the individual parameters as dictated by the biophysical requirements of their model. The flowchart only attempts to present a very basic and easily reusable implementation of the technique.