

Reaction-Diffusion in the NEURON Simulator

Robert A. McDougal

Yale School of Medicine

16 October 2015

Getting Started

What is a reaction-diffusion system?

“Reaction–diffusion systems are mathematical models which explain how the **concentration** of one or more substances distributed in space changes under the influence of two processes: **local chemical reactions** in which the substances are transformed into each other, and **diffusion** which causes the substances to spread out over a surface in space.”¹

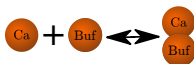
¹http://en.wikipedia.org/wiki/Reaction%E2%80%93diffusion_system

Examples

- Pure Diffusion

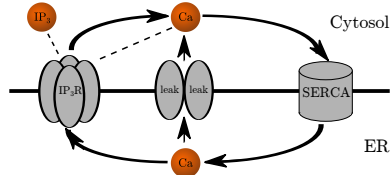
- Protein Degradation

- Buffering



- Circadian Oscillator

- Ca^{2+} -induced Ca^{2+} release



What does the rxd module do?

Reduces typing

- **In 2 lines:** declare a domain, then declare a molecule, allowing it to diffuse and respond to flux from ion channels.

```
all = rxd.Region(h.allsec(), nrn_region='i')
```

```
ca = rxd.Species(all, name='ca', d=1, charge=2)
```

- **Reduces** the risk for **errors** from typos or misunderstandings.

Allows arbitrary domains

NEURON traditionally only identified concentrations just inside and just outside the plasma membrane. The rxd module allows you to **declare your own regions** of interest (e.g. ER, mitochondria, etc).

The three questions

- **Where** do the dynamics occur?
 - Cytosol
 - Endoplasmic Reticulum
 - Mitochondria
 - Extracellular Space
- **Who** are the actors?
 - Ions
 - Proteins
- **What** are the reactions?
 - Buffering
 - Degradation
 - Phosphorylation

Declare a region with rxd.Region

Basic Usage

```
cyt = rxd.Region(seclist)
```

seclist may be any iterable of sections; e.g. a SectionList or a Python list.

Identify with a standard region

```
cyt = rxd.Region(seclist, nrn_region='i')
```

nrn_region may be i or o, corresponding to the locations of e.g. nai vs nao.

Specify the cross-sectional shape

```
cyt = rxd.Region(seclist,  
                 geometry=rxd.Shell(0.5, 1))
```

The default geometry is rxd.inside.

The geometry and nrn_region arguments may both be specified.

geometry:



Adapted from:
McDougal et al 2013.

rxd.Region tips

Specify `nrn_region` if concentrations interact with NMODL

If NMODL mechanisms (ion channels, point processes, etc) depend on or affect the concentration of a species living in a given region, that region must declare a `nrn_region` (typically 'i').

To declare a region that exists on all sections

```
r = rxd.Region(h.allsec())
```

Use list comprehensions to select sections

```
r = rxd.Region([sec for sec in h.allsec() if 'apical' in sec.name()])
```


Declare proteins and ions with rxd.Species

Basic usage

```
protein = rxd.Species(region, d=16)
```

d is the **diffusion constant** in $\mu\text{m}^2/\text{ms}$. region is an rxd.Region or an iterable of rxd.Region objects.

Initial conditions

```
protein = rxd.Species(region, initial=value)
```

value is in mM. It may be a constant or a function of the node.

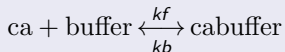
Connecting with HOC

```
ca = rxd.Species(region, name='ca', charge=2)
```

If the nrn_region of region is "i", the concentrations of this species will be stored in cai, and its concentrations will be affected by ica.

Specifying dynamics: rxd.Reaction

Mass-action kinetics

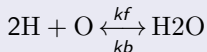


```
buffering = rxd.Reaction(ca + buffer, cabuffer, kf, kb)
```

kf is the forward reaction rate, kb is the backward reaction rate. kb may be omitted if the reaction is unidirectional.

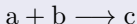
In a mass-action reaction, the reaction rate is proportional to the product of the concentrations of the reactants.

Repeated reactants



```
water_reaction = rxd.Reaction(2 * H + O, H2O, kf, kb)
```

Arbitrary reaction formula, e.g. Hill dynamics



```
hill_reaction = rxd.Reaction(a + b, c, a ^ 2 / (a ^ 2 + k ^ 2),  
                             mass_action=False)
```

Hill dynamics are often used to model cooperative reactions.

rxd.Rate and rxd.MultiCompartmentReaction

rxd.Rate

Use `rxd.Rate` to specify an explicit contribution to the rate of change of some concentration or state variable.

```
ip3degradation = rxd.Rate(ip3, -k * ip3)
```

rxd.MultiCompartmentReaction

Use `rxd.MultiCompartmentReaction` when the dynamics span multiple regions; e.g. a pump or channel.

```
ip3r = rxd.MultiCompartmentReaction(ca[er], ca[cyt], kf, kb,  
                                     membrane=cyt_er_membrane)
```

The rate of these dynamics is proportional to the membrane area.

Manipulating nodes

Getting a list of nodes

- `odelist = protein.nodes`

Filtering a list of nodes

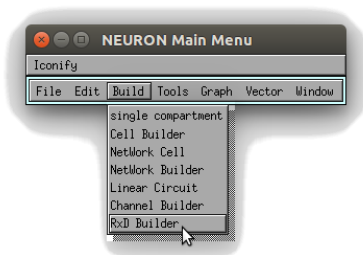
- `odelist2 = oodelist(region)`
- `odelist2 = oodelist(0.5)`
- `odelist2 = oodelist(section)(region)(0.5)`

Other operations

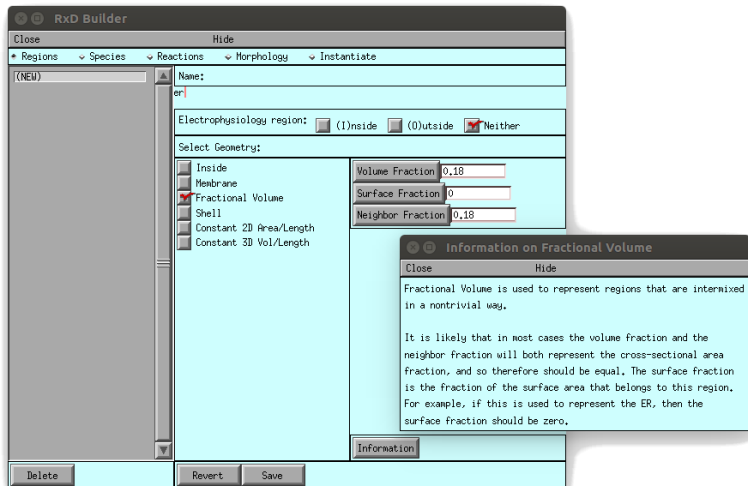
- `odelist.concentration = value`
- `values = oodelist.concentration`
- `surface_areas = oodelist.surface_area`
- `volumes = oodelist.volume`
- `node = oodelist[0]`

GUI

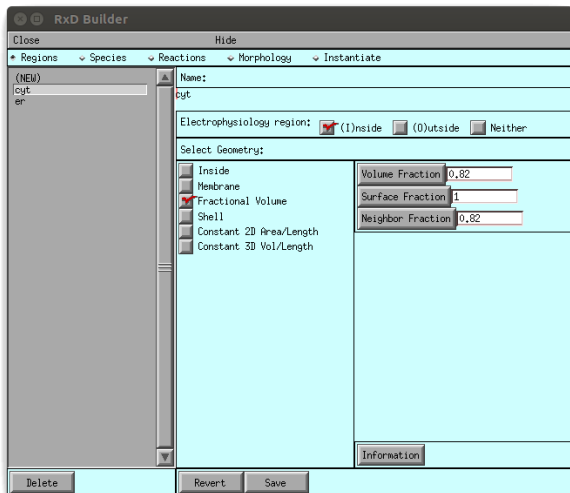
Reaction-diffusion dynamics can also be specified via the GUI. This option appears only when rxd is supported in your install (Python and scipy must be available).



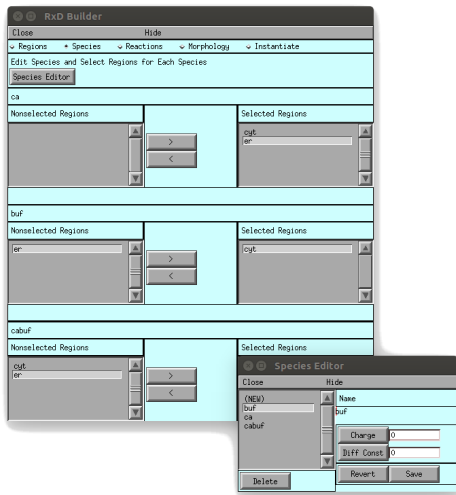
GUI



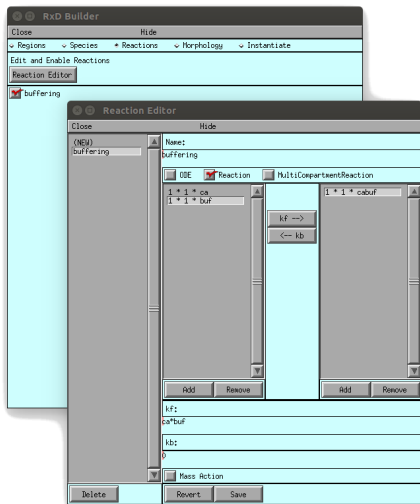
GUI



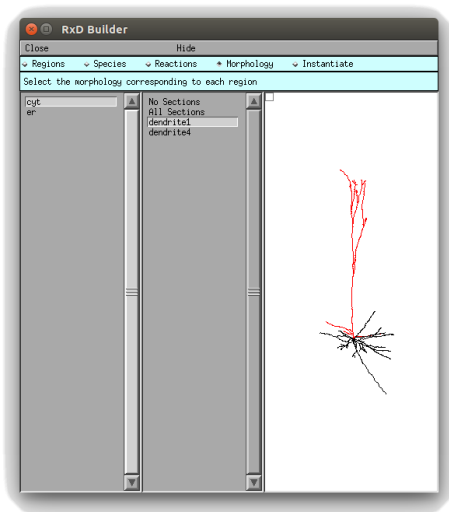
GUI



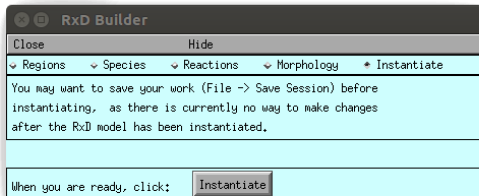
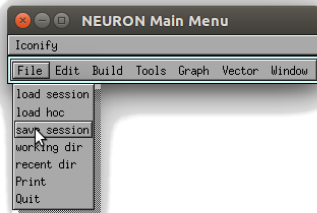
GUI



GUI



GUI



Example: Calcium buffering

```
from neuron import h, rxd, gui

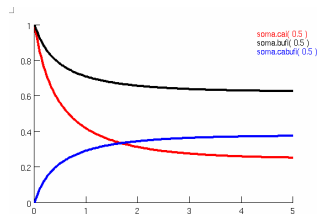
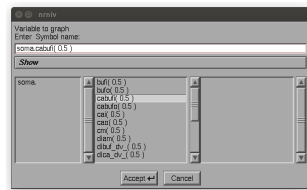
h('create soma')
soma_region = rxd.Region([h.soma], nrn_region='i')

ca = rxd.Species(soma_region, initial=1,
                 name='ca', charge=2)
buf = rxd.Species(soma_region, initial=1,
                 name='buf')
cabuf = rxd.Species(soma_region, initial=0,
                   name='cabuf')

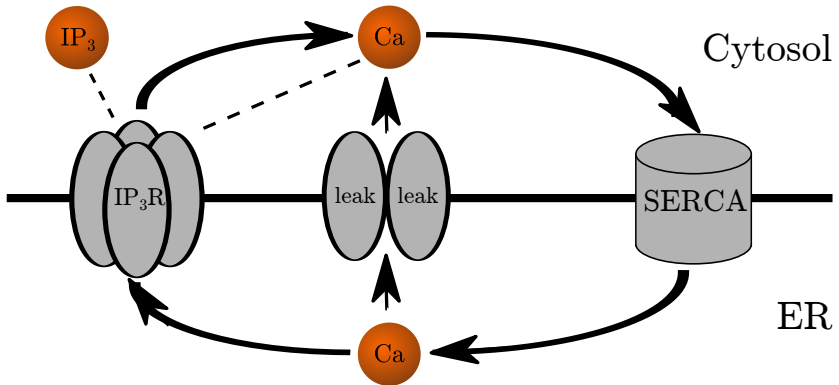
buffering = rxd.Reaction(2 * ca + buf, cabuf, 1, 0.1)
```

In this example, we suppose each buffer molecule binds two molecules of calcium. Other buffers have different properties.

Use the GUI to create a graph and run the simulation.



Example: Calcium wave dynamics



Wagner et al. 2004. *Cell Calcium*. DOI: 10.1016/j.ceca.2003.10.009

Neymotin et al. 2015. *Neural Computation*. DOI: 10.1162/NECO_a_00712

ModelDB: 168874

There are two species:

Calcium:

```
def ca_init(node):  
    return ca_cyt0 if node.region == cyt else ca_er0  
  
ca = rxd.Species([cyt, er],  
                 d=caDiff,  
                 name='ca',  
                 charge=2,  
                 initial=ca_init)
```

Inositol trisphosphate (IP3):

```
ip3 = rxd.Species(cyt,  
                  d=ip3Diff,  
                  initial=ip3_init)
```

Each pump and channel corresponds to its own “reaction”:

Leak:

```
leak = rxd.MultiCompartmentReaction(  
    ca[er] <> ca[cyt],  
    gleak,  
    gleak,  
    membrane=er_membrane)
```

SERCA:

```
serca = rxd.MultiCompartmentReaction(  
    ca[cyt] > ca[er],  
    gserca * (ca[cyt])**2 / (Kserca**2 + (ca[cyt])**2),  
    membrane=er_membrane,  
    mass_action=False)
```


The IP₃R is more complicated because it is essentially a channel that opens and closes slowly as a function of calcium concentration.

IP₃R:

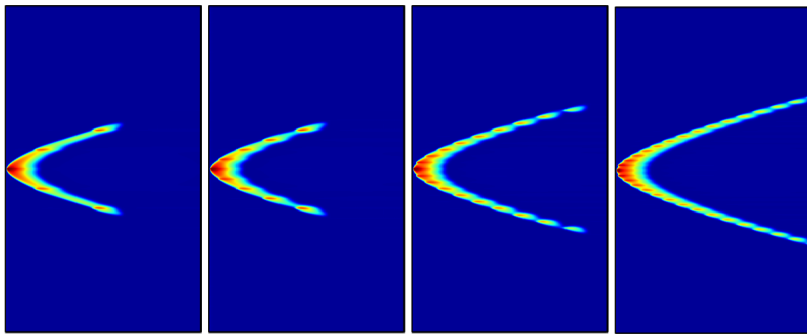
```
# slow gating due to calcium
ip3r_gate_state = rxd.State(er_membrane, initial=0.8)
h_gate = ip3r_gate_state[er_membrane]
ip3rg = rxd.Rate(h_gate,
                  (1. / (1 + ca[cyt] / scale) - h_gate) / tau)

# fast gating due to calcium and IP3
minf = ip3[cyt] * ca[cyt] / (ip3[cyt] + Kip3) / (ca[cyt] + Kact)

# same permeability for either direction of flow
k = gip3r[cyt] * (minf * h_gate) ** 3

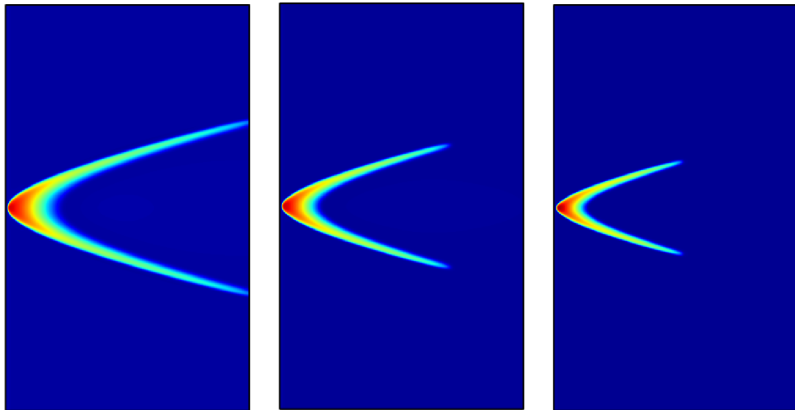
# the actual channel
ip3r = rxd.MultiCompartmentReaction(
    ca[er] <> ca[cyt], k, k, membrane=er_membrane)
```

Reducing IP_3R spacing facilitates wave propagation



More closely spaced IP_3R $\longrightarrow$

Increasing SERCA activity weakens wave propagation



Increasing SERCA activity →

Interacting with the rest of NEURON

node._ref_concentration or node._ref_value returns a pointer.

Recording traces

```
v = h.Vector()
v.record(ca.nodes[0]._ref_concentration)
```

Plotting

```
g = h.Graph()
g.addvar('ca[er][dend](0.5)',
        ca.nodes(er)(dend)(0.5)[0]._ref_concentration)
h.graphList[0].append(g)
```

Tips

`dir(·)`

To find out what methods and properties are available, use `dir`:

```
dir(ca.nodes)
```

`CVode` and `atol`

NEURON's variable step solver has a default absolute tolerance of 0.001.

Since NEURON measures concentration in mM and many cell biology concentrations are in μM , this tolerance may be too high. Try lowering it:

```
h.CVode().atol(1e-8)
```

3D Simulations

The Third Dimension

Specifying 3D Simulations

Just add one line of code²:

```
rxd.set_solve_type(dimension=3)
```

```
all = rxd.Region(h.allsec())
```

```
ca = rxd.Species(all, d=1)
```

```
ca.initial = lambda node: 1 if node.x3d < 50 else 0
```

Plotting

Get the concentration values expressed on a regular 3D grid via `odelist.value_to_grid()`

```
values = ca.nodes.value_to_grid()
```

Pass the result to a 3d volume plotter, such as Mayavi's VolumeSlicer:

```
graph = VolumeSlicer(data=ca.nodes.value_to_grid())  
graph.configure_traits()
```

²`rxd.set_solve_type` can optionally take a list of sections as its first argument; in that case only the specified sections will be simulated in three dimensions.

Example: wave curvature

```

from neuron import h, gui, rxd
import volume_slicer

sec1, sec2 = h.Section(), h.Section()
h.pt3dadd(2, 0, 0, 2, sec=sec1)
h.pt3dadd(9.9, 0, 0, 2, sec=sec1)
h.pt3dadd(10, 0, 0, 2, sec=sec1)
h.pt3dadd(10, 0, 0, 10, sec=sec2)
h.pt3dadd(18, 0, 0, 10, sec=sec2)

def do_init(node):
    return 1 if node.x3d < 8 else 0

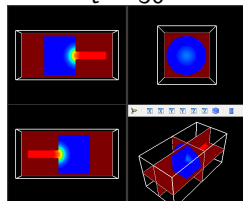
all3d = rxd.Region(h.allsec(), dimension=3)
ca = rxd.Species(all3d, initial=do_init, d=0.05)
r = rxd.Rate(ca, -ca * (1 - ca) * (0.1 - ca))

def plot_it():
    graph = volume_slicer.VolumeSlicer(
        data=ca.nodes.value_to_grid(),
        vmin=0, vmax=1)
    graph.configure_traits()

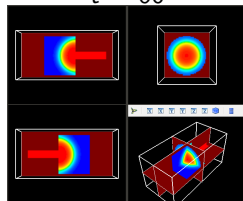
h.finitialize()
for t in [30, 60]:
    h.continuerun(t)
    plot_it()

```

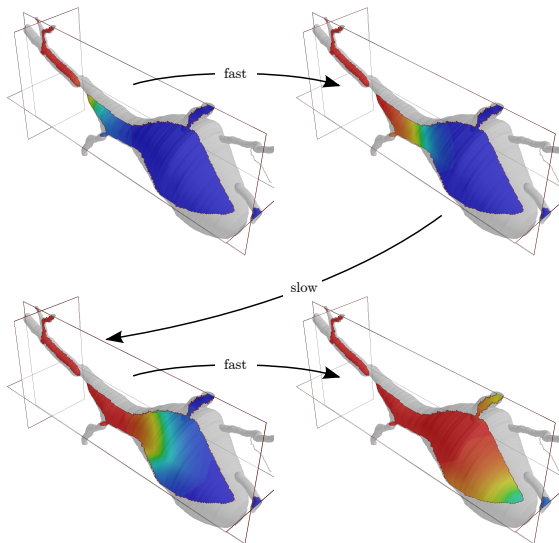
t = 30



t = 60



Example: wave curvature at soma entry



References

For more information, see:

Journal Articles on Reaction-Diffusion in NEURON

- McDougal, R. A., Hines, M. L., Lytton, W. W. (2013). Reaction-diffusion in the NEURON simulator. *Frontiers in Neuroinformatics*, 7.
- McDougal, R. A., Hines, M. L., Lytton, W. W. (2013). Water-tight membranes from neuronal morphology files. *Journal of Neuroscience Methods*, 220(2), 167-178.

Online Resources

- NEURON Forum
- Programmer's Reference
- NEURON Reaction-Diffusion Tutorials