Reaction-Diffusion in the NEURON Simulator

Robert A. McDougal
Anna Bulanova

Yale School of Medicine

26 July 2014
Getting Started

When should I use the reaction-diffusion module?
How do I use the rxd module?
Interacting with the rest of NEURON
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.”

\[1\] http://en.wikipedia.org/wiki/Reaction%E2%80%93diffusion_system
Examples

- Pure Diffusion

- Protein Degradation

- Buffering

Circadian Oscillator

Ca\(^{2+}\)-induced Ca\(^{2+}\) release

IP\(_3\) R leak leak SERCA

Cytosol

ER

Robert A. McDougal & Anna Bulanova

Reaction-Diffusion in the NEURON Simulator
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**

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

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

**Identify with a standard region**

```python
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**

```python
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: 

- `rxd.inside`
- `rxd.membrane`
- `rxd.FractionalVolume(volume_fraction=f_1, surface_fraction=f_2)`
- `rxd.Shell(r_1/R, r_2/R)`

Adapted from: McDougal et al 2013.
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 \texttt{\'i\'}).

To declare a region that exists on all sections

\begin{verbatim}
r = rxd.Region(h.allsec())
\end{verbatim}

Use list comprehensions to select sections

\begin{verbatim}
r = rxd.Region([sec for sec in h.allsec() if 'apical' in sec.name()])
\end{verbatim}
Declare proteins and ions with rxd.Species

**Basic usage**

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

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

**Initial conditions**

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

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

**Connecting with HOC**

```python
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**

\[ ca + \text{buffer} \xrightleftharpoons[kf]{kb} \text{cabuffer} \]

\[ \text{buffering} = \text{rxd.Reaction}(ca + \text{buffer}, \text{cabuffer}, kf, kb) \]

$k_f$ is the forward reaction rate, $k_b$ is the backward reaction rate. $k_b$ 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**

\[ 2\text{H} + \text{O} \xrightleftharpoons[kf]{kb} \text{H2O} \]

\[ \text{water\_reaction} = \text{rxd.Reaction}(2 * \text{H} + \text{O}, \text{H2O}, kf, kb) \]

**Arbitrary reaction formula, e.g. Hill dynamics**

\[ a + b \rightarrow c \]

\[ \text{hill\_reaction} = \text{rxd.Reaction}(a + b, c, a^2 / (a^2 + k^2), \text{mass\_action=False}) \]

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.

\[
ip3\text{degradation} = \text{rxd.Rate}(ip3, -k * ip3)
\]

**rxd.MultiCompartmentReaction**

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

\[
ip3r = \text{rxd.MultiCompartmentReaction}(ca[er], ca[cyt], kf, kb,
\text{membrane=cyt\_er\_membrane})
\]

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

Getting a list of nodes

- nodelist = protein.nodes

Filtering a list of nodes

- nodelist2 = nodelist(region)
- nodelist2 = nodelist(0.5)
- nodelist2 = nodelist(section)(region)(0.5)

Other operations

- nodelist.concentration = value
- values = nodelist.concentration
- surface_areas = nodelist.surface_area
- volumes = nodelist.volume
- node = nodelist[0]
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).
When should I use the reaction-diffusion module?
How do I use the rxd module?
Interacting with the rest of NEURON

GUI

Robert A. McDougal & Anna Bulanova
Reaction-Diffusion in the NEURON Simulator
When should I use the reaction-diffusion module?

How do I use the rxd module?

Interacting with the rest of NEURON

GUI

Robert A. McDougal & Anna Bulanova

Reaction-Diffusion in the NEURON Simulator
GUI

Robert A. McDougal & Anna Bulanova

Reaction-Diffusion in the NEURON Simulator
GETTING STARTED

New and in Development

References

When should I use the reaction-diffusion module?
How do I use the rxd module?
Interacting with the rest of NEURON

GUI

Robert A. McDougal & Anna Bulanova

Reaction-Diffusion in the NEURON Simulator
Getting Started
New and in Development
References

When should I use the reaction-diffusion module?
How do I use the rxd module?
Interacting with the rest of NEURON

GUI

Robert A. McDougal & Anna Bulanova
Reaction-Diffusion in the NEURON Simulator
Example: calcium buffering

```python
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.
node._ref_concentration or node._ref_value returns a pointer.

**Recording traces**

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

**Plotting**

```python
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`:

```python
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 \( \mu \text{M} \), this tolerance may be too high. Try lowering it:

```python
h.CVode().atol(1e-8)
```
New and in Development
### Specifying 3D Simulations

Just add "dimension=3" to your rxd.Region:

```python
all = rxd.Region(h.allsec(), dimension=3)
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 `nodelist.value_to_grid()`:

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

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

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

---

2 This is an experimental feature in the development version.
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()
Subdiscretization

Specifying number or subsegments

\[ \text{rxd.options.nsubseg} = 5 \]
SBML is an XML-based representation format used for specifying computational models of biological processes.

BioModels online database: http://www.ebi.ac.uk/biomodels-main/

Usage:

1. User loads or constructs an electrophysiology model
2. SBML data is loaded.
3. User matches state variables and parameters across the two models, and sets missing parameters.
4. RxD objects are instantiated
5. User can adjust the parameters and make simulation runs of the model.
References
For more information, see:

**Journal Articles**

- 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