28 June 2022

Incorporating chemical dynamics

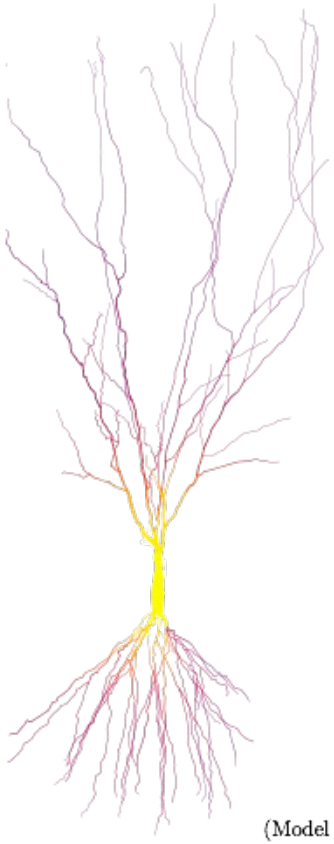
12% of the population suffers from migraines.

Risk of dementia doubles every 5 years after age 65.

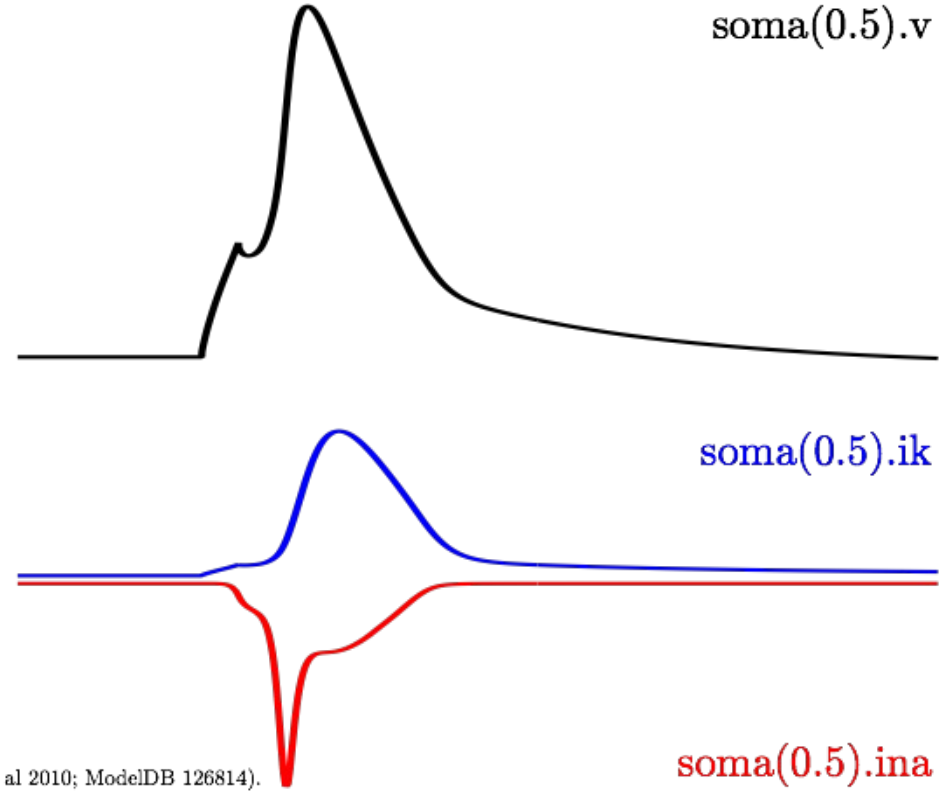
Stroke is the #2 cause of death worldwide.

https://my.clevelandclinic.org/health/diseases/5005-migraine-headaches
https://www.sciencedaily.com/releases/2019/10/191029084315.htm
http://www9.who.int/gho/mortality burden disease/en/

Action potentials arise from ions moving across the membrane



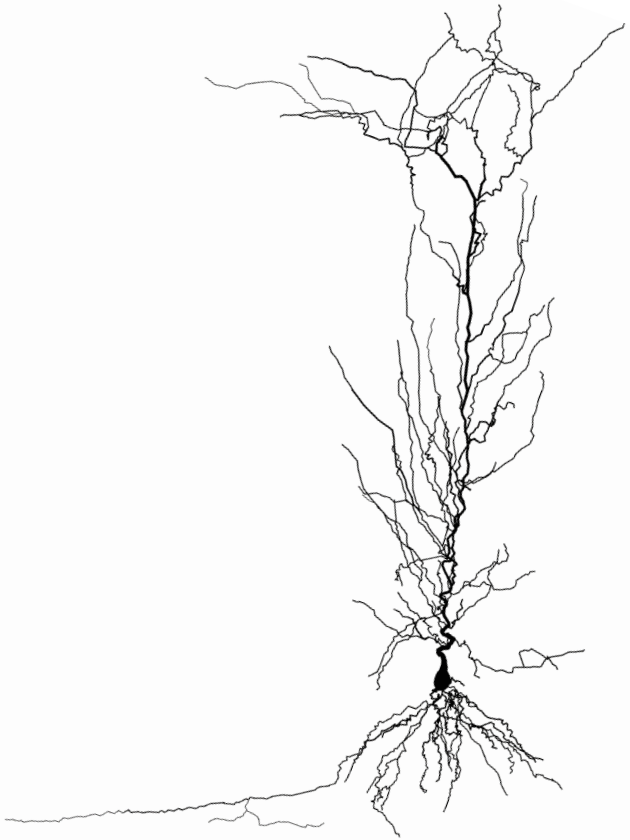
(Model from Safiulina et al 2010; ModelDB 126814).



Questions

- What's the relationship between sodium current and intracellular concentration?
  - $I = \Delta Q / \Delta t$
  - $F \approx 96485.3$ C mol^{-1}
  - $\Delta[\mathrm{Na}] = \frac{10000\, I_{Na}\, (area)}{(F)(volume)}$ ← in NEURON's units in 1 ms
- What's a typical intracellular sodium concentration for a squid giant axon?
- What about potassium?
- How are things different if it's calcium instead of sodium?

A neuron is not a transistor

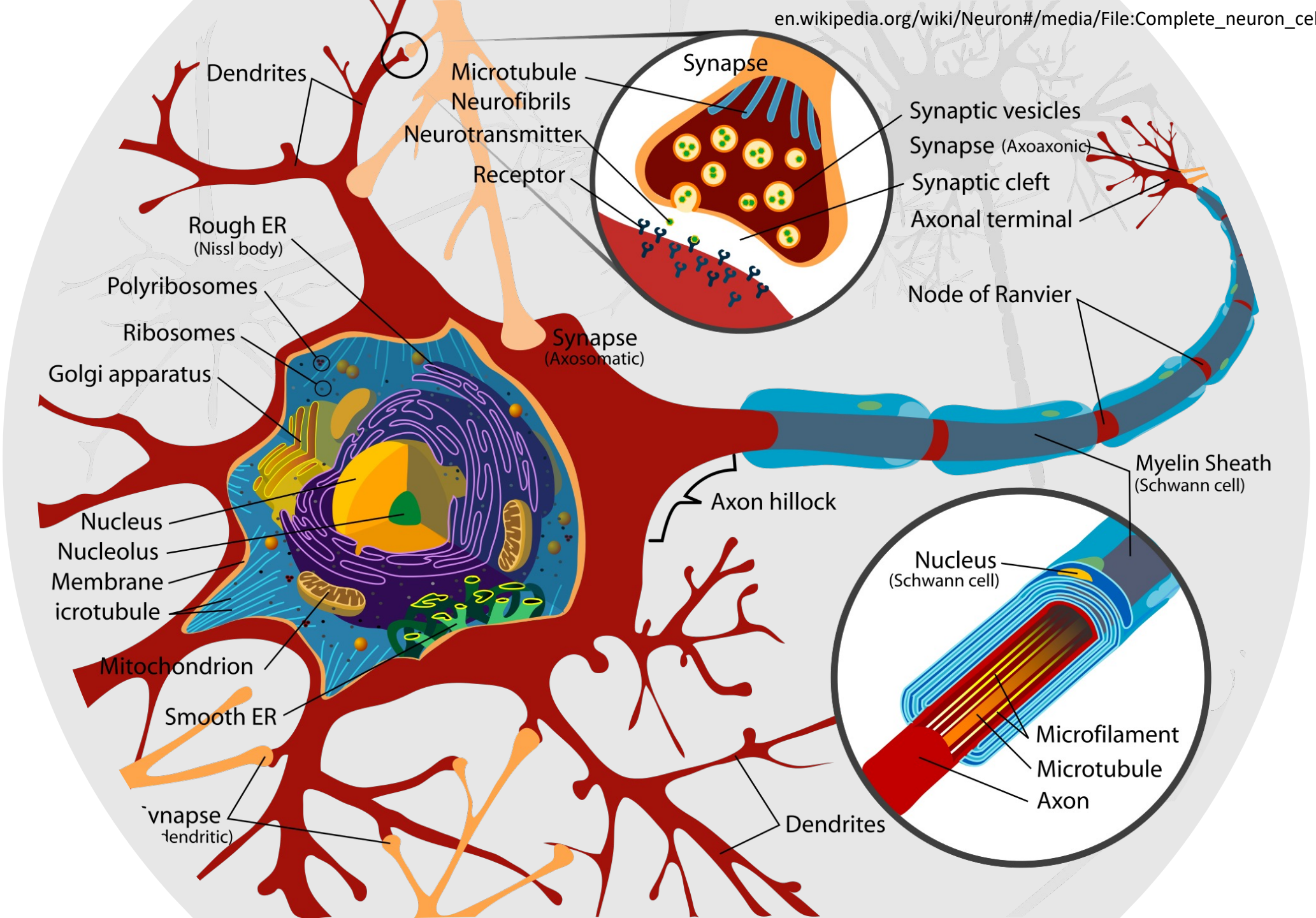


≠

https://commons.wikimedia.org/wiki/File:Bc635-transistor.png

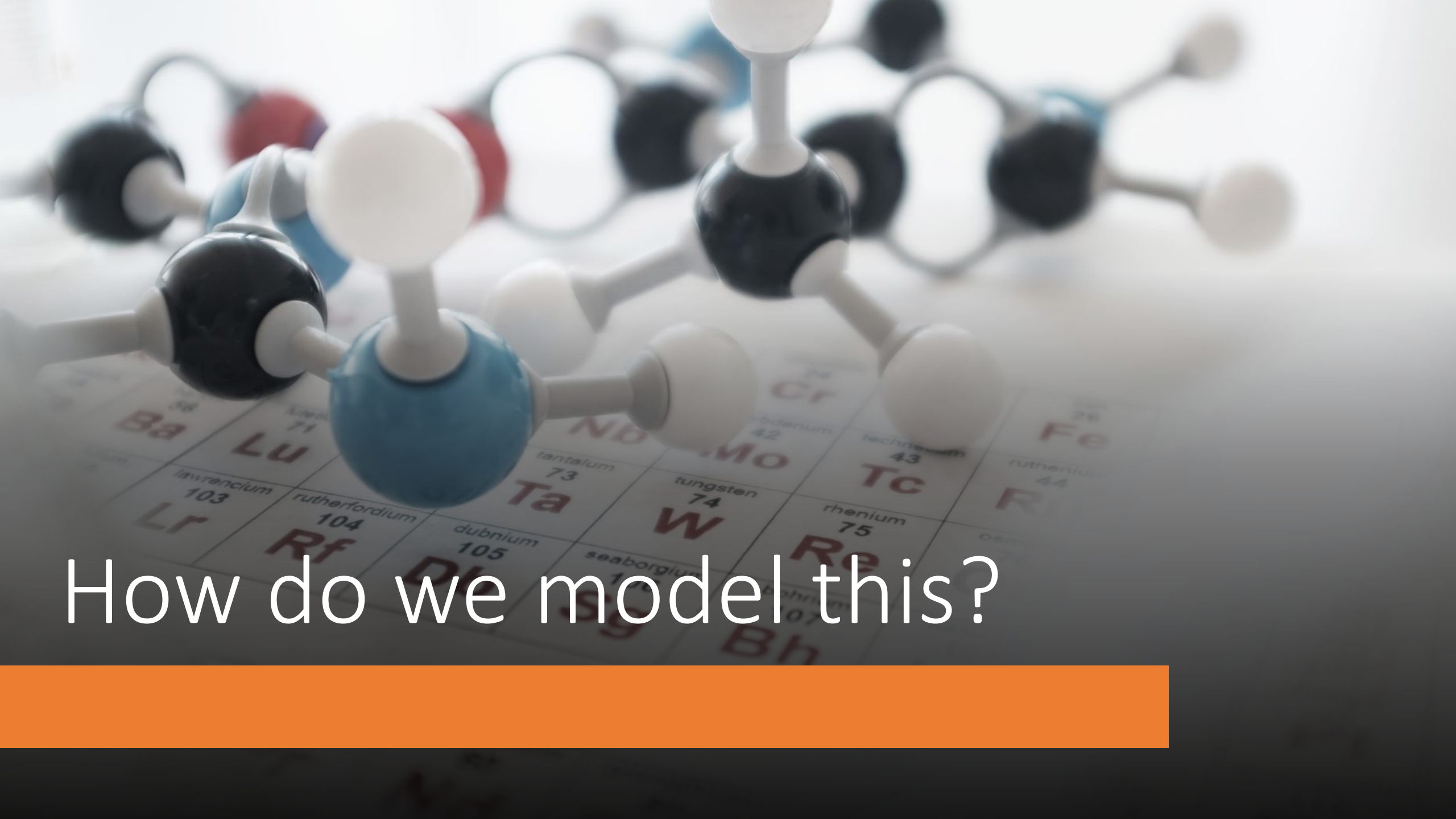
Neuron reconstruction from Pyapali et al 1998 via http://neuromorpho.org/neuron_info.jsp?neuron_name=n123; rendered with NEURON.

en.wikipedia.org/wiki/Neuron#/media/File:Complete_neuron_cell_diagram_en.svg



Neurons have state and are affected by the brain's state

- Protein oscillations in the SCN
- Hyperpolarization-activated graded persistent activity in the PFC (e.g. Winograd et al., 2008)
- Intracellular calcium dynamics
- Synaptic pathways
- Ischemia



How do we model this?

“**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 across space.”

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

Mass-Action kinetics

The model

- A reaction's product is formed at a rate proportional to the concentration of the reactants.

Example

- Consider the reaction

$$\mathrm{Na} + \mathrm{Cl} \xrightarrow{k} \mathrm{NaCl}$$

- Then:

$$[\mathrm{Na}]' = -k[\mathrm{Na}][\mathrm{Cl}]$$
$$[\mathrm{Cl}]' = -k[\mathrm{Na}][\mathrm{Cl}]$$
$$[\mathrm{NaCl}]' = k[\mathrm{Na}][\mathrm{Cl}]$$

Conservation of mass.

Matter is neither created nor destroyed by reactions.

In our equations, this means:

[Na] + [NaCl] = constant
[Cl] + [NaCl] = constant

Example

Every reaction can go both ways, so there is always some possibility for the reaction to reverse. So more realistic kinetics look like:

$$\text{Ca} + 2\ \text{Cl} \underset{k_b}{\overset{k_f}{\rightleftarrows}} \text{CaCl}_2$$

With corresponding equations:

$$[\text{Ca}]' = -k_f[\text{Ca}][\text{Cl}]^2 + k_b[\text{CaCl}_2]$$
$$[\text{Cl}]' = -2k_f[\text{Ca}][\text{Cl}]^2 + 2k_b[\text{CaCl}_2]$$
$$[\text{CaCl}_2]' = k_f[\text{Ca}][\text{Cl}]^2 - k_b[\text{CaCl}_2]$$

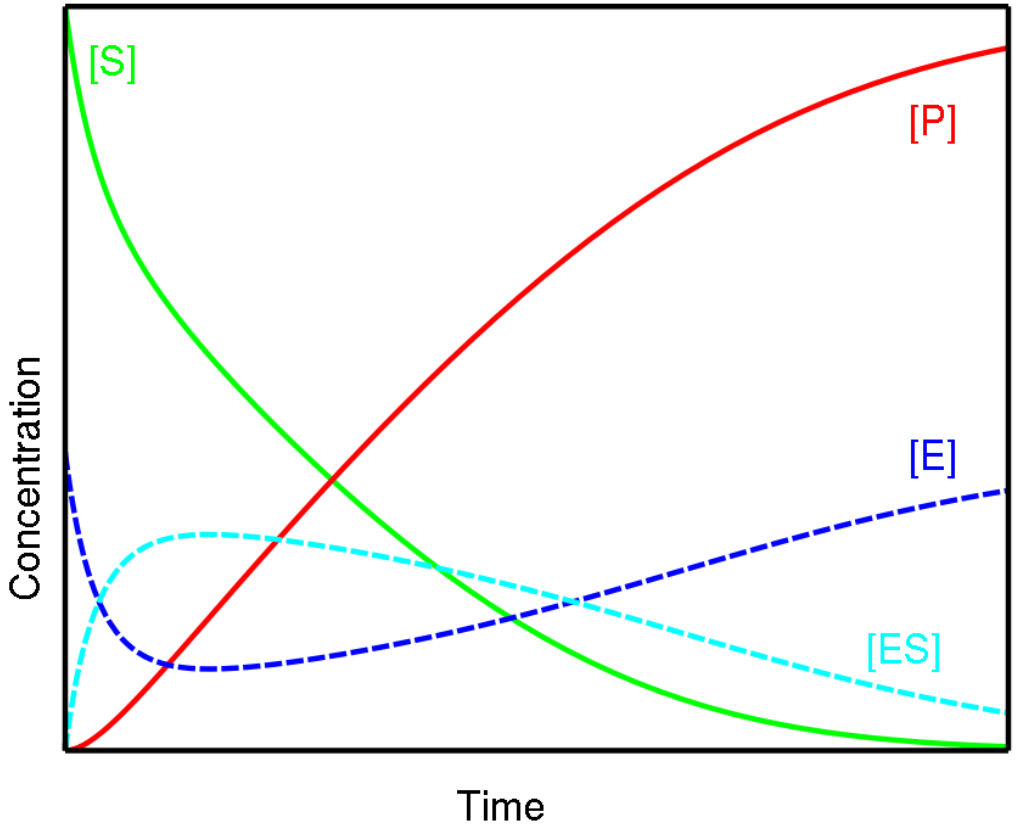
Enzyme kinetics

It is generally **not** the case that a substrate transforms directly into a product:

$$S \rightarrow P$$

Instead, an enzyme is often involved:

$$\mathrm{E} + \mathrm{S} \underset{k_b}{\overset{k_f}{\rightleftarrows}} \mathrm{ES} \xrightarrow{k_{\mathrm{cat}}} \mathrm{E} + \mathrm{P}$$



https://commons.wikimedia.org/wiki/File:Michaelis_Menten_S_P_E_ES.svg

Michaelis-Menten

If we can assume either:

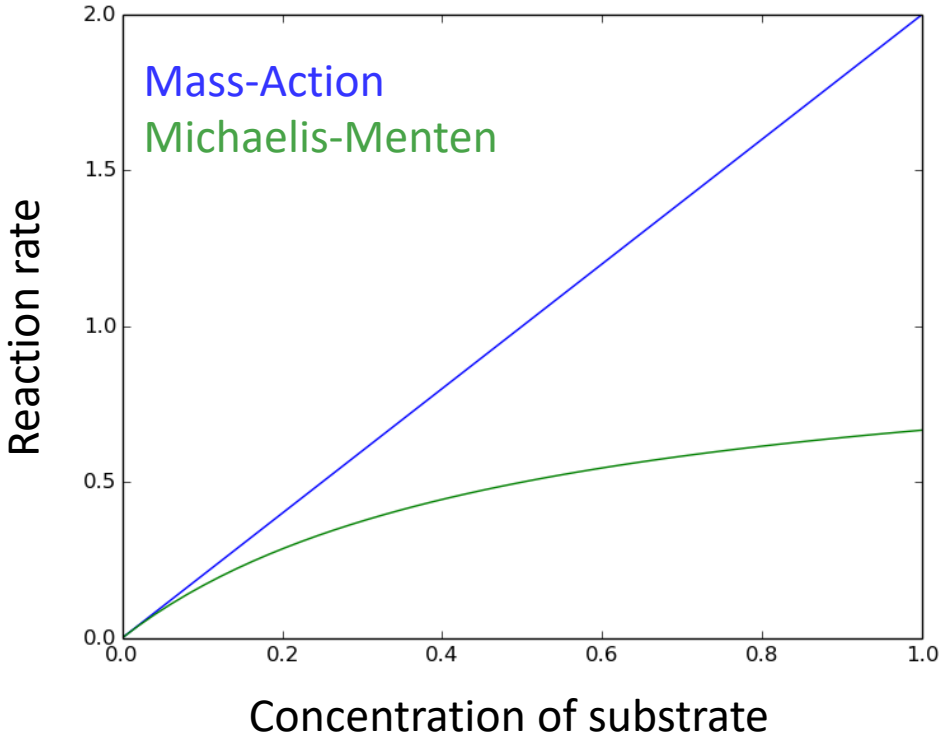
- the substrate (S) and the complex (ES) are in instantaneous equilibrium, or
- the concentration of the complex (ES) does not change on the time-scale of product formation

Then the rate of the enzymatic reaction reduces to:

$$\frac{V_{max}\,[S]}{K_M + [S]}$$

K_M is called the *Michaelis constant*. It is the concentration at which the reaction proceeds at half its maximum rate.

Michaelis-Menten vs Mass-Action

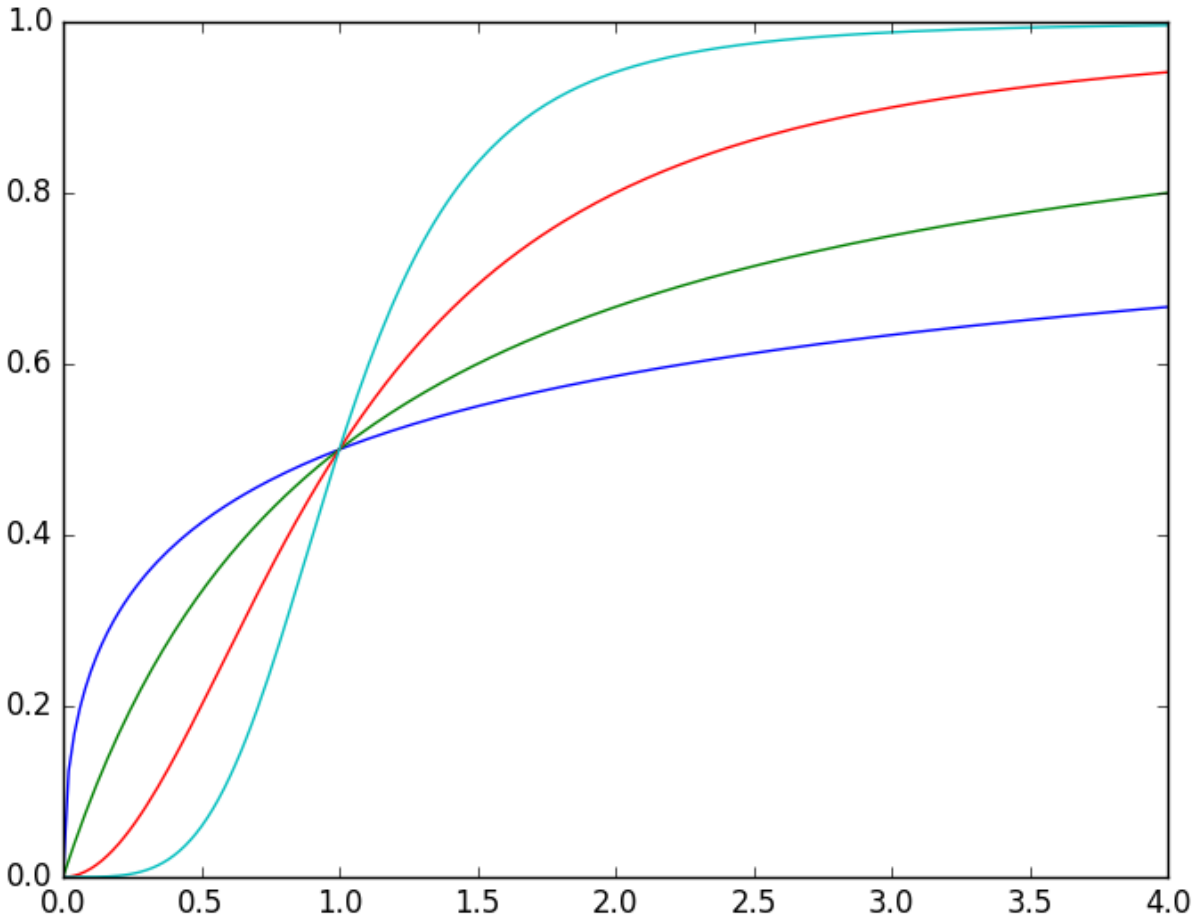


$$S \rightarrow P$$

Both curves on the left have the same rate of reaction when the substrate concentration is low, but the Michaelis-Menten rate levels off (due to limited enzyme availability) as concentrations increase.

$$y = 2x \qquad y = \frac{x}{x + 0.5}$$

Hill equation: cooperative binding



$n = 4$

$n = 2$

$n = 1$

$n = 0.5$

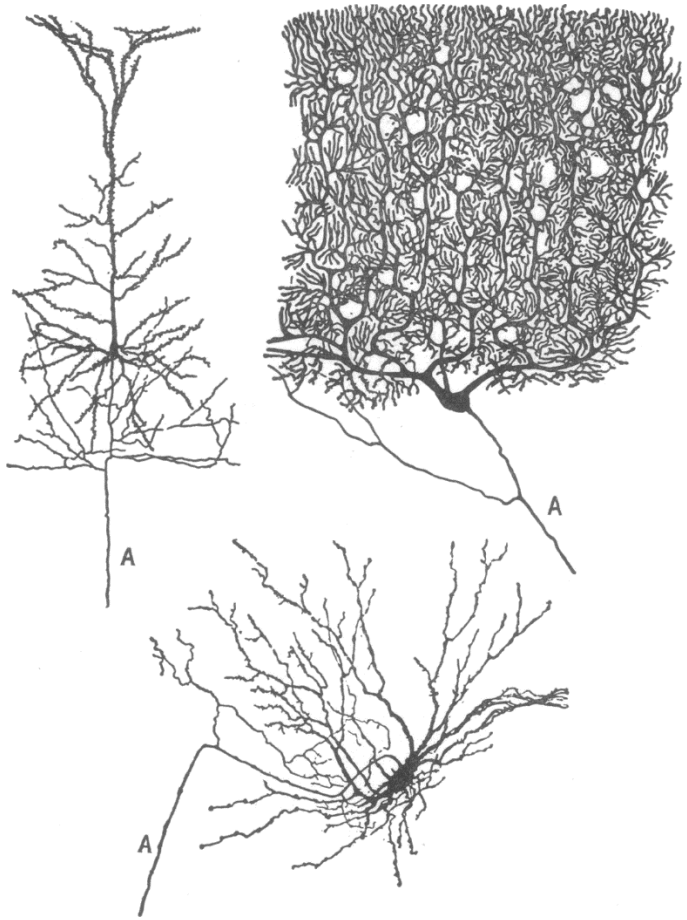
$$\frac{V_{max}\,[S]^n}{[k_A]^n + [S]^n}$$

$$\frac{[S]^n}{1^n + [S]^n}$$

If n > 1, positive cooperativity.

If n < 1, negative cooperativity.

Neurons have spatial extent



Cajal 1909 as reproduced in Rall 1962.

Effects of non-point-ness:

- Ion and protein concentrations vary with space.
- Cellular mechanisms (ER, ion channels, etc) vary with space.

Concentrations at different locations affect each other:

- Transport
- Diffusion

Fick's Laws

The mathematics of diffusion

Fick's First Law:

- Diffusive flux is proportional to the concentration gradient.

$$J = -D\nabla\varphi$$

- Here D is called the *diffusion coefficient*.

Fick's Second Law (the diffusion equation):

$$\frac{\partial\varphi}{\partial t} = \nabla \cdot (D\nabla\varphi) = D\,\nabla^2\varphi$$

where the last equality only holds if D is constant.

Where does diffusion occur?

Cytosol

- But not full cross section because of organelles

Organelles

- e.g. endoplasmic reticulum

Extracellular space

- Tortuosity
- Anisotropy
- Volume fraction

Practical limits of pure diffusion

The expected time $\mathrm{E}[t]$ for a molecule with diffusion constant D to diffuse a distance x is:

$$\mathrm{E}[t] = \frac{x^2}{2D}$$

So in particular, if

$D = 1$ μm^2/ms and

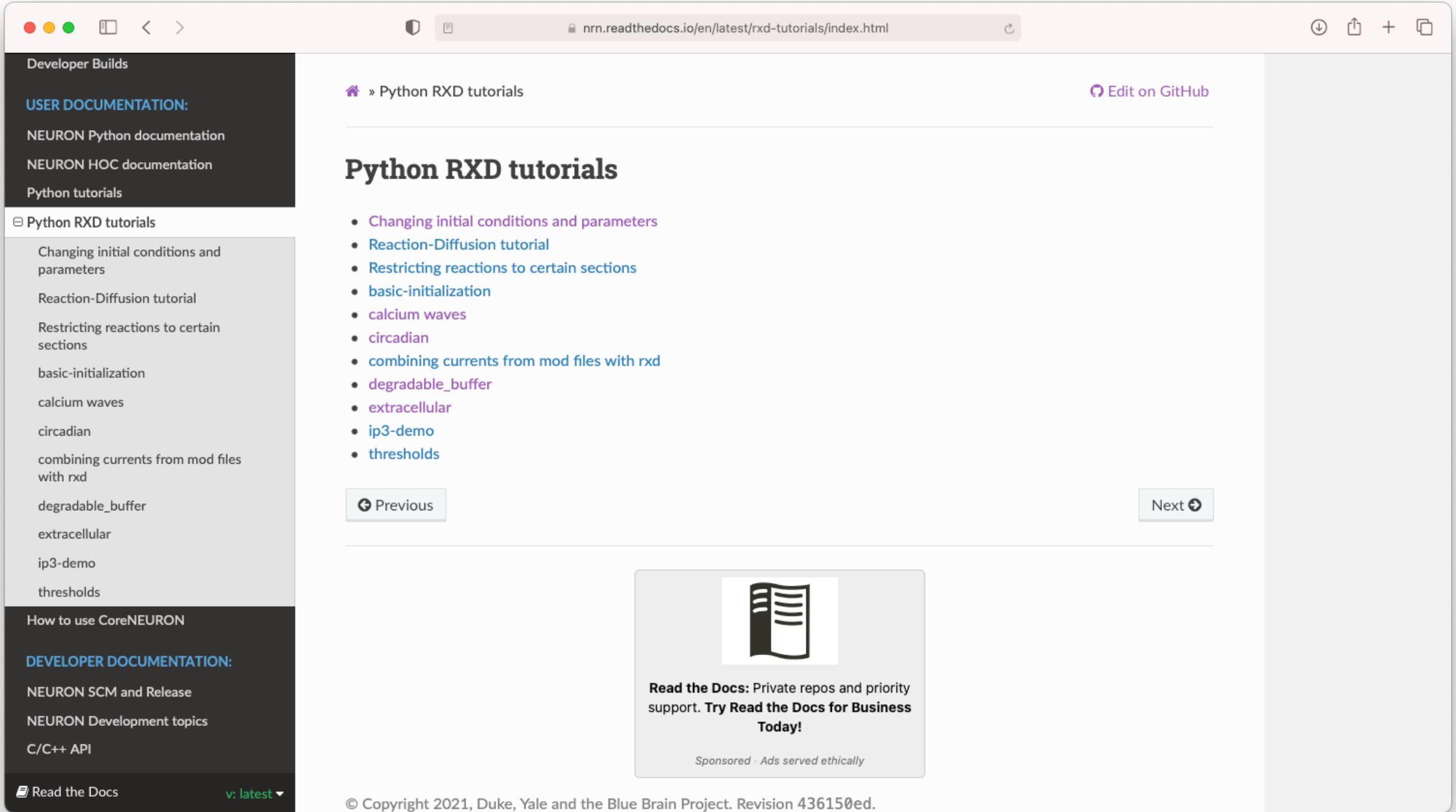
$x = 100$ μm,

Then

$E[t] = \frac{100^2}{2} = 5000$ ms.

Reaction-Diffusion in NEURON

```
from neuron import rxd
```



https://nrn.readthedocs.io/en/latest/rxd-tutorials/index.html

Why use NEURON's `rxd` module?

Reduces typing

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

```
cyt = rxd.Region(soma.wholetree(), nrn_region="i")
ca = rxd.Species(cyt, name="ca", d=1, charge=2)
```

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

Allow arbitrary domains

- By default, NEURON only has two domains for chemical 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).

`rxd` module overview

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

Interface design principle

- Reaction-diffusion model specification is independent of:
  - Deterministic or stochastic
  - 1D or 3D

Declaring a region: `rxd.Region`

Basic usage

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

seclist may be any iterable of sections; e.g. a Python list or h.allsec() or…

Identify with a standard region

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

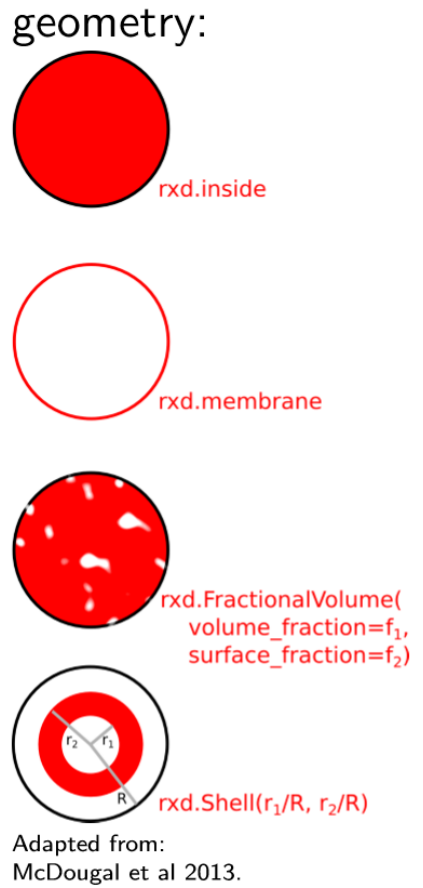
`nrn_region` may be I or o, corresponding to the locations of e.g. nai vs nao

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.



Adapted from:
McDougal et al 2013.

`rxd.Region` tips

Specify `nrn_region` if concentration interact with NMODL

- If NMODL mechanism (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 str(sec)])
```

Ions and proteins: `rxd.Species`

Basic usage

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

- d is the diffusion constant in $\mu m^2/ms$; region is either an rxd.Region or an iterable of 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 NMODL, InterViews graphics, etc…

```
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.`

`protein.initial` can be read and set, to allow exploration of the role of initial conditions

Tip:

Variable step integration tolerances

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

Since NEURON measures concentration in mM and some cell biology concentrations (e.g. calcium) are in µM, this tolerance may be too high. Compensate by using an `atolscale` in the constructor, e.g.

```
ca = rxd.Species(cyt, atolscale=1e-6)
```

Example:

Initial value as a function of distance from a point

```
def my_initial(node):
    # compute the distance
    distance = h.distance(soma(0.5), node)
    # return a certain function of the distance
    return 2 * h.tanh(distance / 1000.)

cyt = rxd.Region(soma.wholetree(), name='cyt', nrn_region='i')

ip3 = rxd.Species(cyt, name='ip3', charge=2, initial=my_initial)
```

Example:

Initial value as a function of spatial position

```
def my_initial(node):
    # return a certain function of the x-coordinate
    return 1 + h.tanh(node.x3d / 100.)

cyt = rxd.Region(h.allsec(), name='cyt', nrn_region='i')

ip3 = rxd.Species(cyt, name='ip3', charge=2, initial=my_initial)
```

Tip:

rxd.Parameter

- Use `rxd.Parameter` objects to represent things that vary either spatially or across different simulations. They use the same specification as `rxd.Species`.

- e.g.

```
α = rxd.Parameter(cyt, initial=0.3)
```

Tip:

`rxd.Parameter`

Use short-hand to avoid repeatedly writing `rxd.Parameter` boilerplate; e.g.

```
def declare_parameters(r, **kwargs):
    '''enables clean declaration of parameters in top namespace'''
    for key, value in kwargs.items():
        globals()[key] = rxd.Parameter(r, name=key, initial=value)
```

Can then, e.g.:

```
declare_parameters(r,
    vsP=1.1 * nM / hour,
    KmP=0.2 * nM,
    ksP=0.9 / hour)
```

- Units come from: `from neuron.units import nM, hour`

Specifying dynamics: `rxd.Reaction`

Mass-action kinetics

$$\mathrm{ca} + \mathrm{buffer} \underset{kb}{\overset{kf}{\longleftrightarrow}} \mathrm{cabuffer}$$

```
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

$$2\mathrm{H} + \mathrm{O} \underset{kb}{\overset{kf}{\longleftrightarrow}} \mathrm{H2O}$$

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

Arbitrary reaction formula, e.g. Hill dynamics

$$\mathrm{a} + \mathrm{b} \longrightarrow \mathrm{c}$$

```
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

- `nodelist = protein.nodes`
- `nodes_in_region = protein.nodes(region)`
- `nodes_in_sec_and_region = protein.nodes(sec)(region)`

Working with concentrations and values

- `nodelist.concentration = value`
- `values = nodelist.concentration`
- `ca_timeseries = h.Vector().record(ca(er)(soma(0.5))._ref_concentration)`
-
```
for node in state_var.nodes:
    node.value *= 2
```

Other operations

- `surface_areas = nodelist.surface_area`
- `volumes = nodelist.volume`
- `first_node = nodelist[0]`

Concentration pointers

node._ref_concentration

If there's only one node in a nodelist, you can also use:

nodelist._ref_concentration

Recording traces:

```
cai = h.Vector().record(
    ca.nodes(soma(0.5))._ref_concentration
)
```

Plotting in an h.Graph:

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

h.graphList[0].append(g)
```

Remember, you can use e.g. `dir(ca.nodes)` to find out what methods exist.

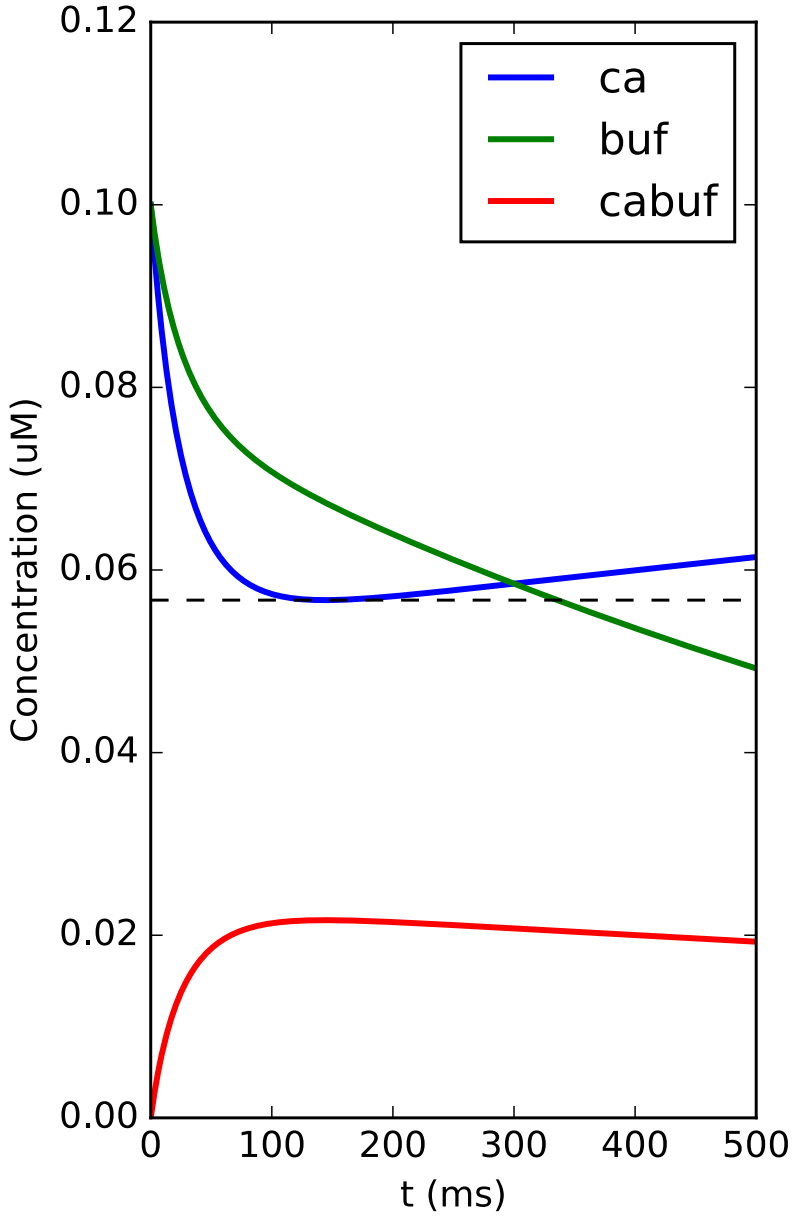
Example:

Calcium buffering*

Consider calcium buffering with a degradable buffer:

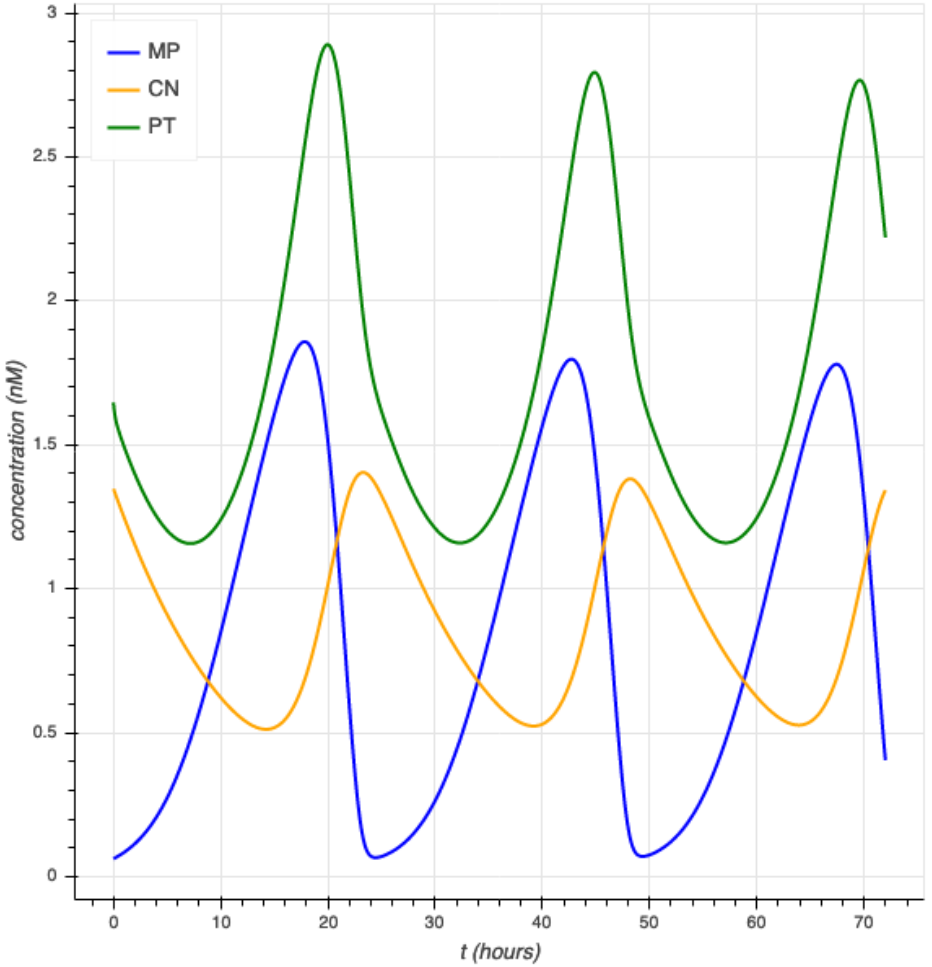
$$2Ca + Buf \leftrightarrow CaBuf, \quad Buf \rightarrow (degraded)$$

```
from neuron import h, rxd
from neuron.units import nM

# where
soma = h.Section(name="soma")
cyt = rxd.Region([soma], nrn_region="i")

# who
ca = rxd.Species(cyt, name="ca", charge=2, initial=100*nM)
buf = rxd.Species(cyt, name="buf", initial=100*nM)
cabuf = rxd.Species(cyt, name="cabuf", initial=0)

# what
buffering = rxd.Reaction(2 * ca + buf, cabuf, 1e6, 1e-2)
degradation = rxd.Rate(buf, 1e-3 * buf)
```



Example:

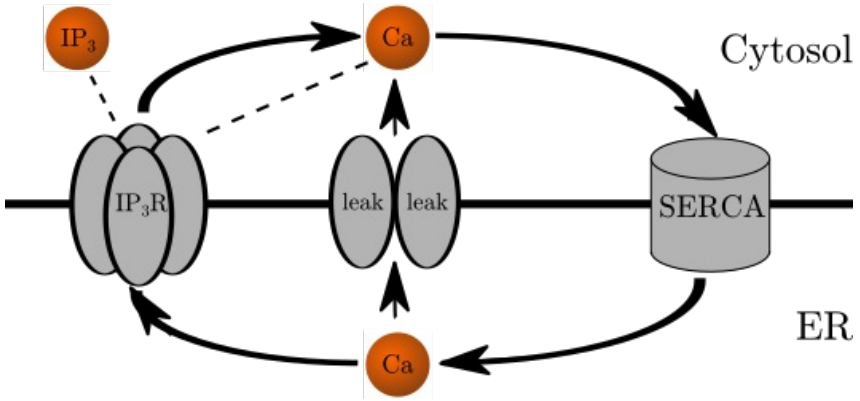
Leloup, Gonze, Goldbeter 1999



https://tinyurl.com/neuron-leloup-1999

Example:

CICR



$$\frac{\partial \mathrm{Ca}^{2+}_{\mathrm{cyt}}}{\partial t} = d_{\mathrm{Ca}^{2+}_{\mathrm{cyt}}} \cdot \Delta \mathrm{Ca}^{2+}_{\mathrm{cyt}} + \frac{J_{\mathrm{IP3R}} - J_{\mathrm{SERCA}} + J_{\mathrm{leakER}}}{f_{cyt}} + c_{\mathrm{ionic}},$$

$$\frac{\partial \mathrm{Ca}^{2+}_{\mathrm{ER}}}{\partial t} = d_{\mathrm{Ca}^{2+}_{\mathrm{ER}}} \cdot \Delta \mathrm{Ca}^{2+}_{\mathrm{ER}} - \frac{J_{\mathrm{IP3R}} - J_{\mathrm{SERCA}} + J_{\mathrm{leakER}}}{f_{ER}},$$

$$\frac{\partial \mathrm{IP}_3}{\partial t} = d_{\mathrm{IP}_3} \cdot \Delta \mathrm{IP}_3,$$

$$J_{\mathrm{IP3R}} = \bar{p}_{\mathrm{IP}_3\mathrm{R}} \cdot m^3_{\mathrm{IP}_3\mathrm{R}} \cdot n^3_{\mathrm{IP}_3\mathrm{R}} \cdot h^3_{\mathrm{IP}_3\mathrm{R}} \cdot (\mathrm{Ca}^{2+}_{\mathrm{ER}} - \mathrm{Ca}^{2+}_{\mathrm{cyt}})/\Xi,$$

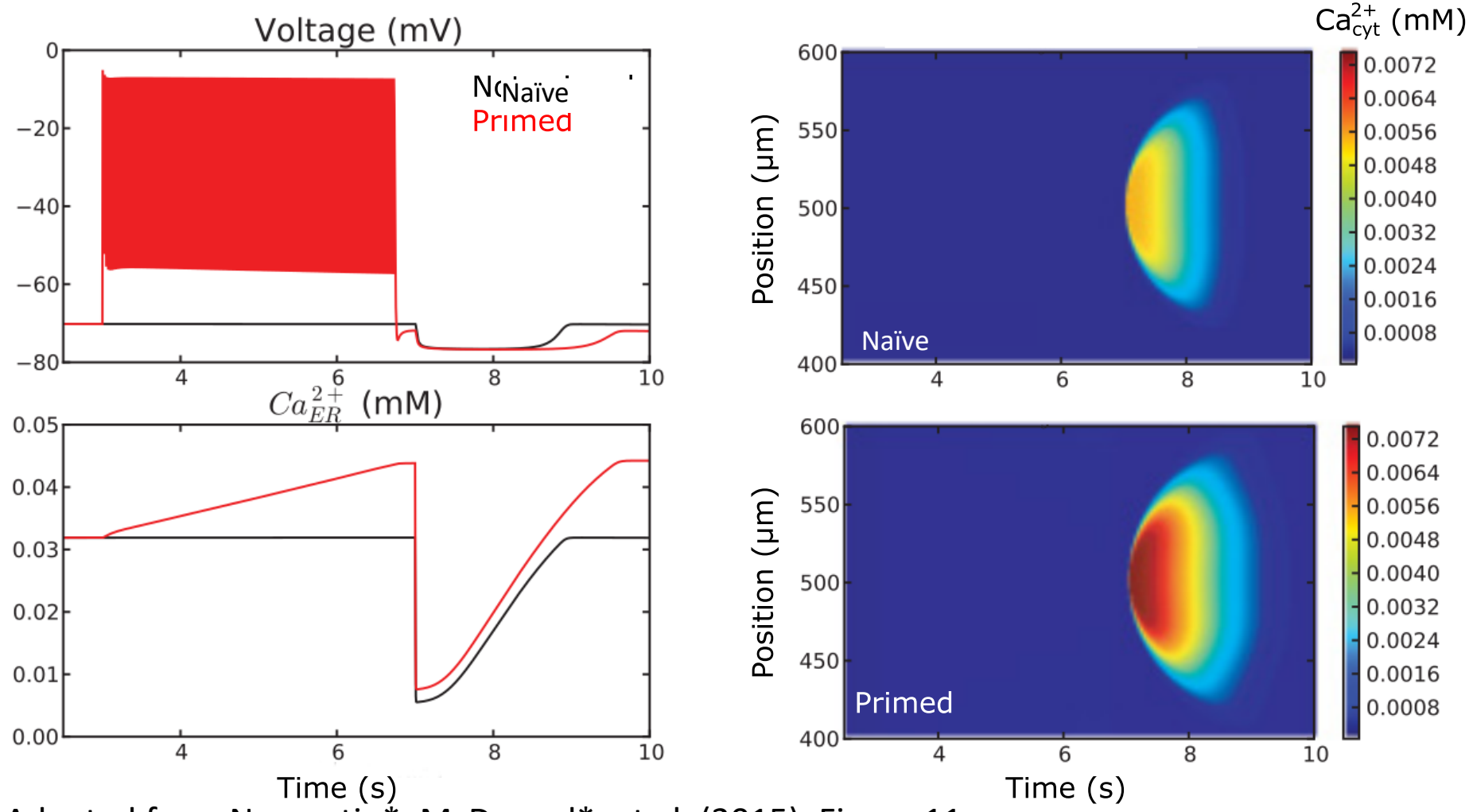
$$J_{\mathrm{SERCA}} = -\frac{\bar{p}_{serca} \cdot {\mathrm{Ca}^{2+}_{\mathrm{cyt}}}^2}{(k^2_{serca} + {\mathrm{Ca}^{2+}_{\mathrm{cyt}}}^2) \cdot \Xi},$$

$$J_{\mathrm{leakER}} = \bar{p}_{leakER} \cdot (\mathrm{Ca}^{2+}_{\mathrm{ER}} - \mathrm{Ca}^{2+}_{\mathrm{cyt}})/\Xi.$$

Neymotin*, McDougal*, et al. (2015)

Example:

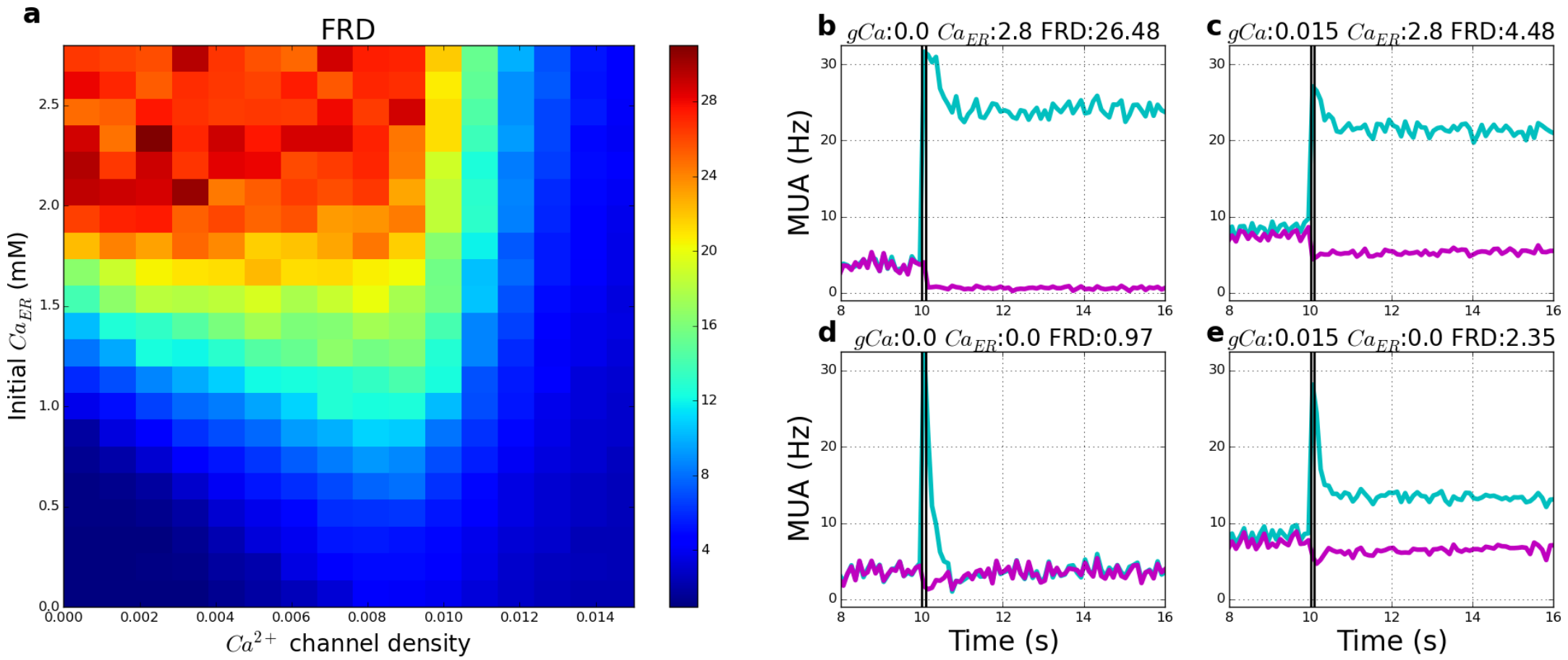
CICR



Adapted from Neymotin*, McDougal*, et al. (2015). Figure 11.

Example:

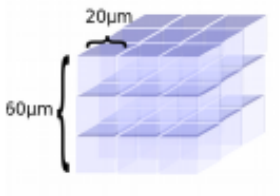
CICR



Neymotin, McDougal, et al. (2016). Figure 12: 582 Pyramidal, 97 fast spiking interneurons, 97 low threshold interneurons.

Extracellular diffusion

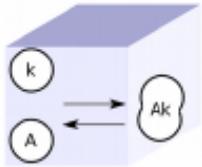
- Uses the same simple Python interface




```
ecs = rxd.Extracellular(xlo=-30, ylo=-30, zlo=-30,
                        xhi=30, yhi=30, zhi=30,
                        dx=20, tortuosity=1.6,
                        volume_fraction=0.2)
```

Tortuosity and volume_fraction can be either constants or functions of (x, y, z).

```
astrocytic_buffering = rxd.Reaction(A + k, AK, kf, kb)
```



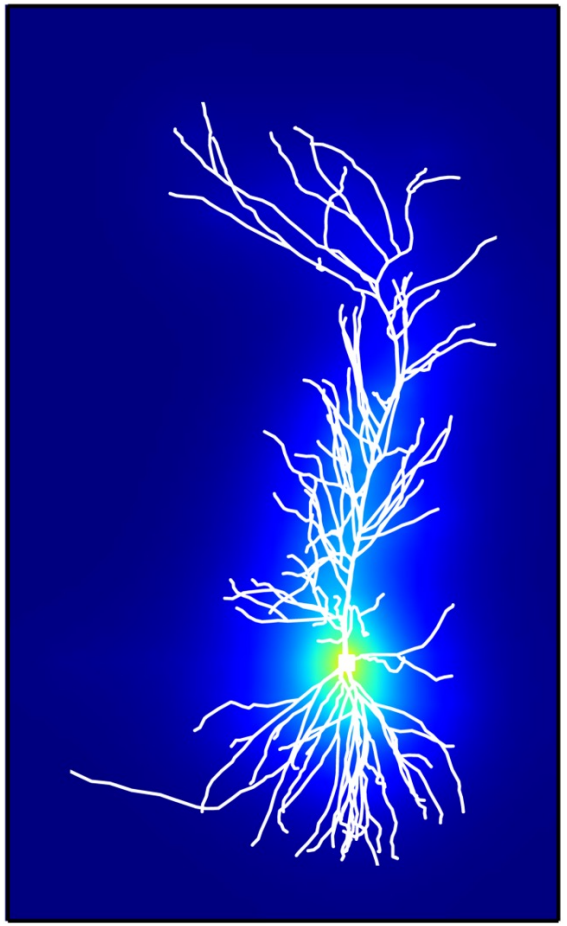
- Rectangular cuboid grid

- Supports
  - anisotropy & heterogeneous tissue characteristics

```
k = rxd.Species(ecs, name='k', d=2.62, charge=1,
      initial=lambda nd: 40 if nd.x3d**2 + nd.y3d**2 + nd.z3d**2 < 100**2 else 3.5,
      ecs_boundary_conditions=3.5
```

nrn.readthedocs.io/en/latest/rxd-tutorials/extracellular.html

Extracellular diffusion



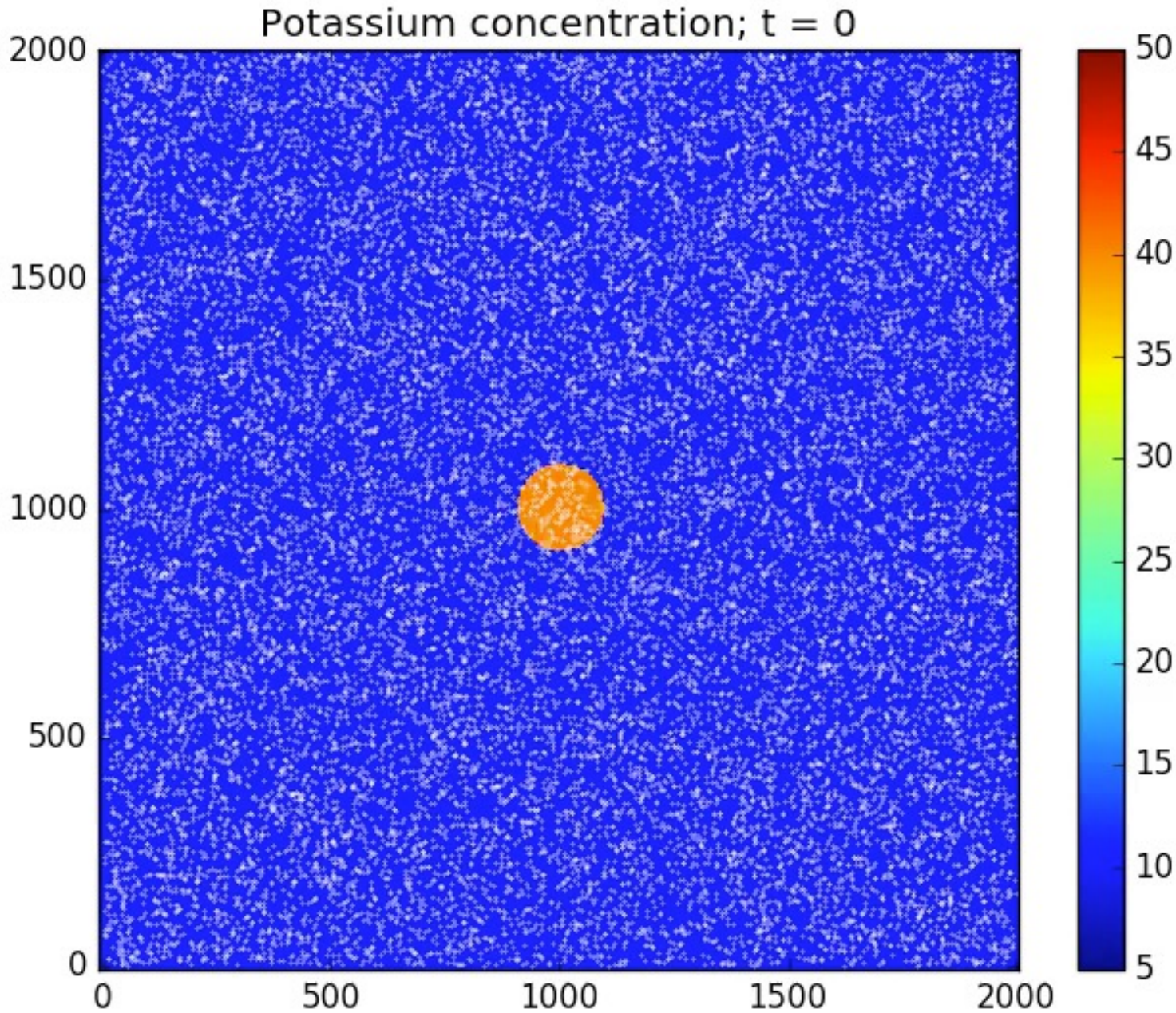
New region type:

```
ecs = rxd.Extracellular(xlo, ylo, zlo, xhi, yhi, zhi,
                        dx=dx, tortuosity=1,
                        volume_fraction=1)
```

Setting/getting extracellular concentrations:

```
ca[ecs].states3d[5:15, 5:15, :] = 1
pyplot.imshow(ca[ecs].states3d[:, :, 0],
              interpolation='nearest', vmin=0, vmax=1,
              extent=ca[ecs].extent('xy'),
              origin='lower')
```

All 3D simulation in NEURON is done using a parallelized Douglas-Gunn alternating implicit method.


Potassium concentration; t = 0
2000
1500
1000
500
0
0
500
1000
1500
2000
50
45
40
35
30
25
20
15
10
5

3D Simulations

Specifying 3D Simulations

Just add one line of code[2]:

```
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 nodelist.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.

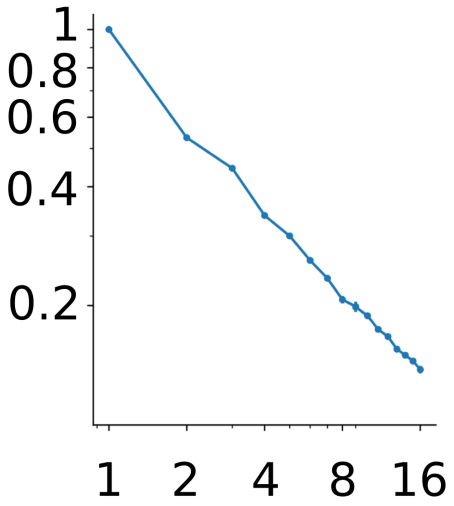
Threading

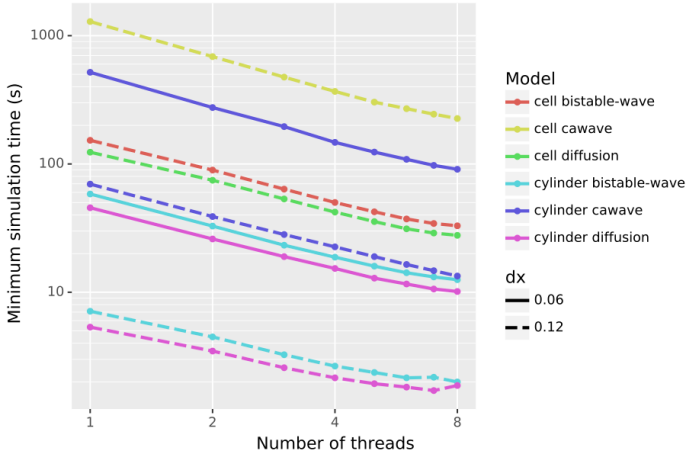
- Extracellular and 3D simulations may be threaded using, e.g.

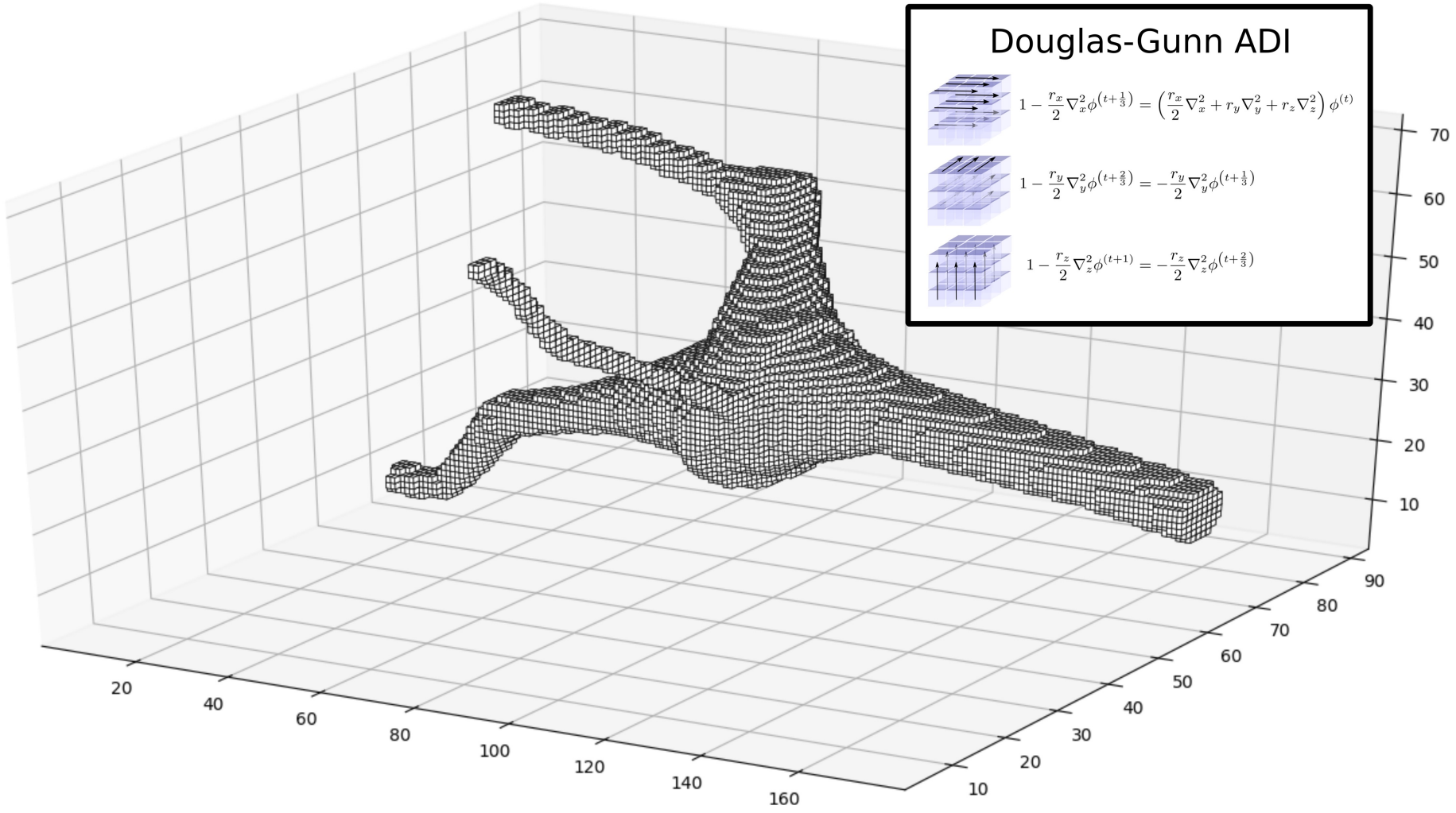
```
rxd.nthread(4) # for four threads
```

- Either electrophysiology or reaction-diffusion can be threaded, but not both.

3D extra- (left) and intracellular (right) runtime by number of threads.

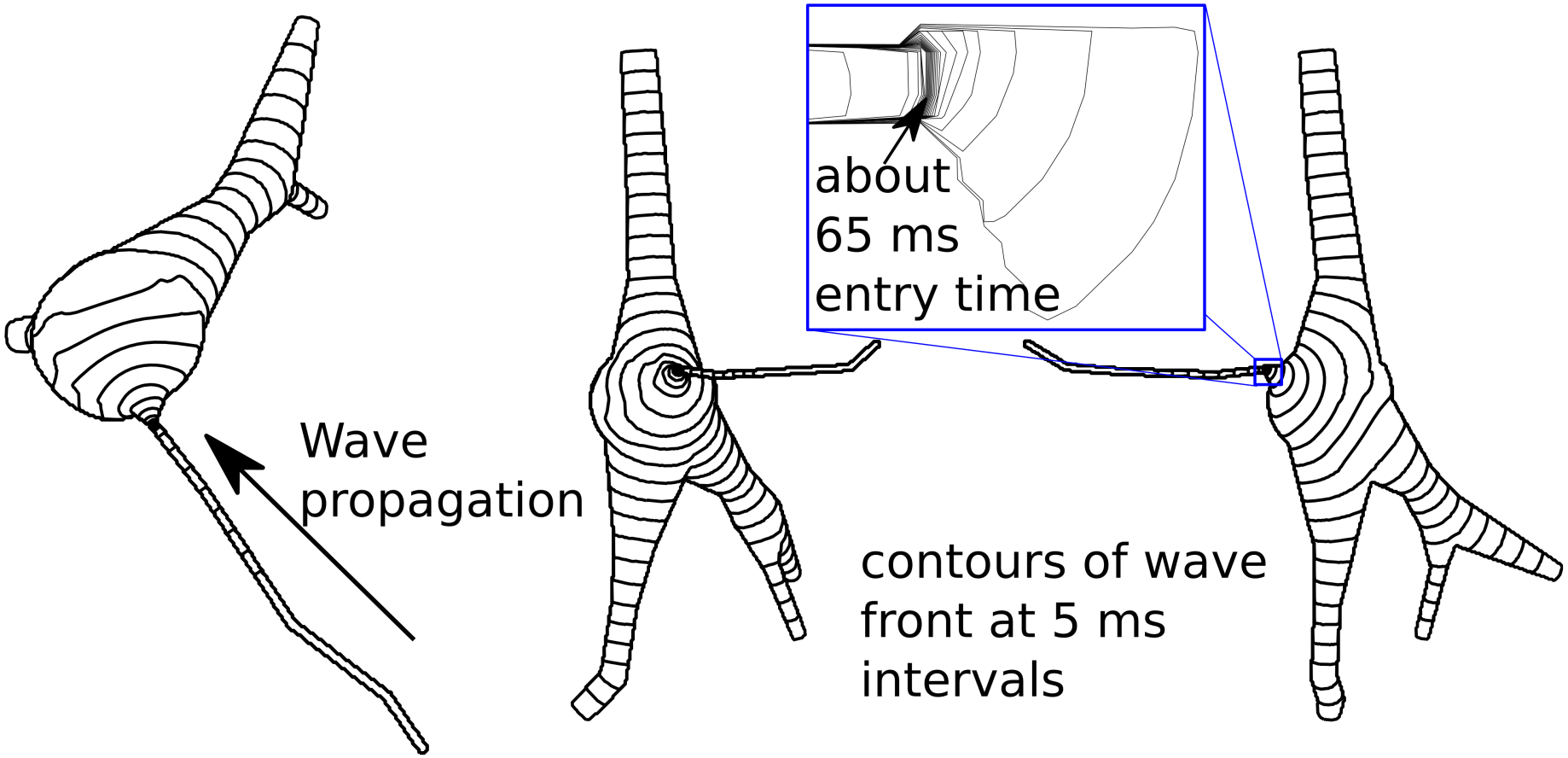






NEURON 7.7+ uses threaded DG-ADI; previous versions used bicgstab.

Wave curvature and delays at soma



$$u_t = \mathrm{D}\,\nabla^2 u - \xi u(1-u)(\alpha - u)$$

For about 750k voxels, a pre-alpha branch of NEURON 7.7 simulated 300 ms of this (dt=0.025ms) with four threads in 258 s.

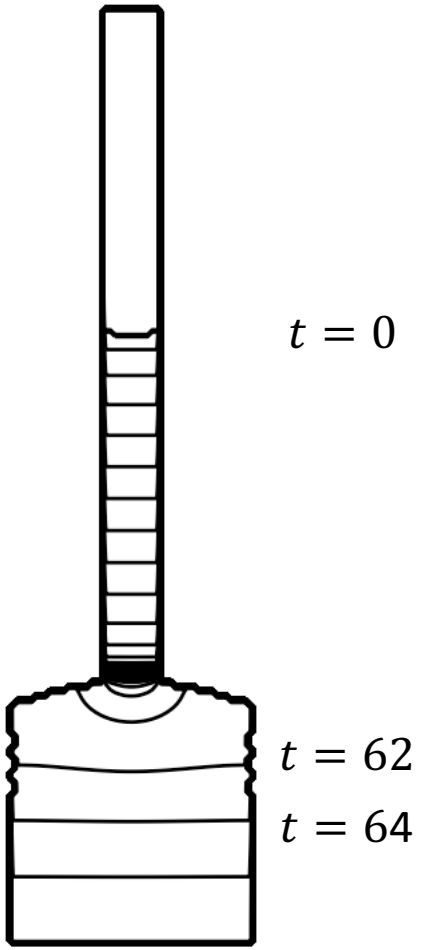
For a similar example, see:
https://neuron.yale.edu/neuron/docs/3dhybrid-intracellular-tutorial

Full 3D simulation (part 1 of 2)

```
from neuron import h, rxd
from neuron.units import um, ms, mV
import matplotlib.pyplot as plt
import numpy as np

h.load_file("stdrun.hoc")

# Create morphology
soma = h.Section(name="soma")
soma.L = soma.diam = 5 * um
soma.nseg = 5
dend = h.Section(name="dend")
dend.L = 15 * um
dend.diam = 1 * um
dend.nseg = 15
dend.connect(soma)

# tell NEURON this cell should be in 3D
rxd.set_solve_type(soma.wholetree(), dimension=3)

def ca_init(node):
  return 1 if node.sec == dend and node.segment.x > 0.5 else 0
```

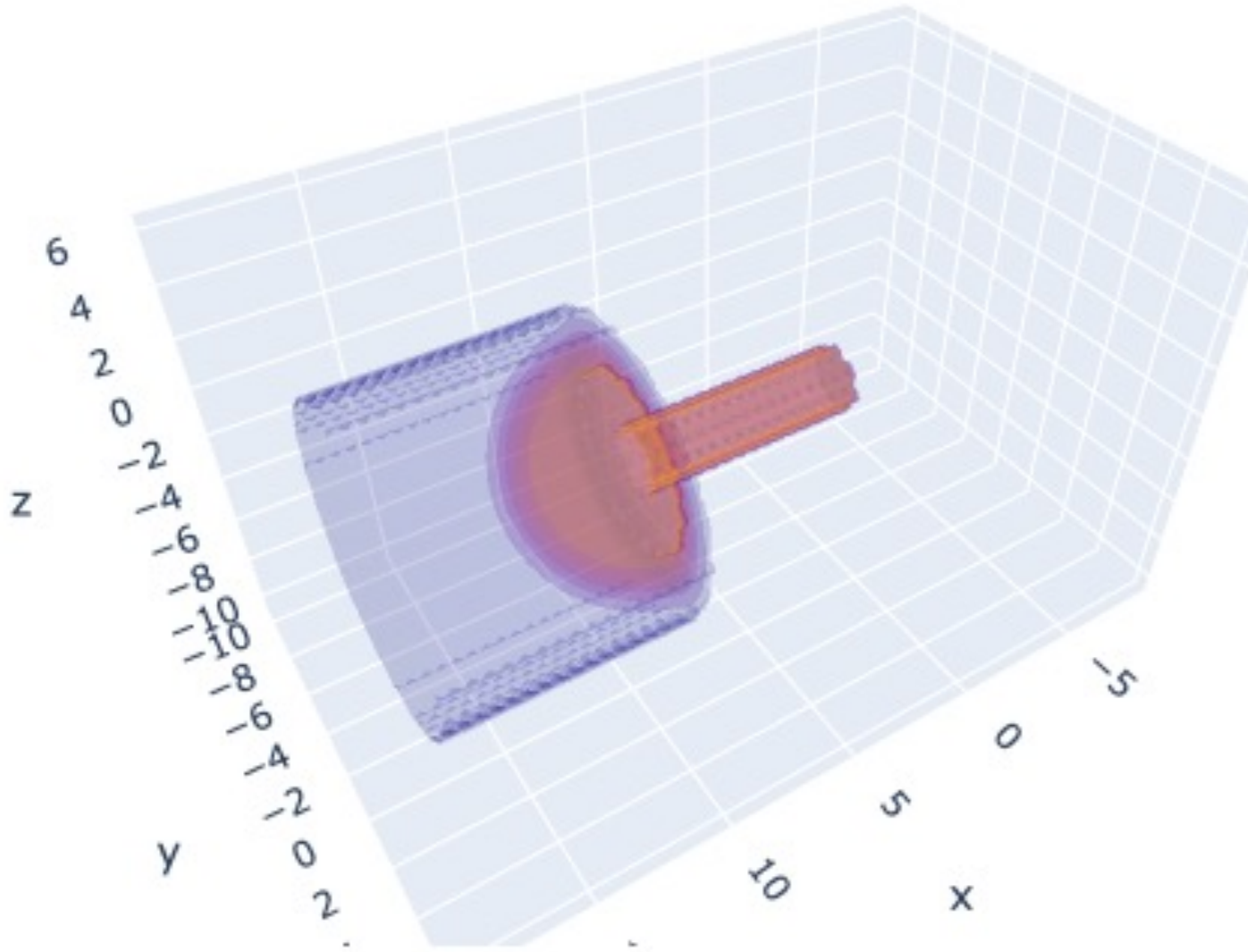
Full 3D simulation (part 2 of 2)

```
r = rxd.Region(soma.wholetree(), dx=0.1 * um)
ca = rxd.Species(r, d=0.3, name="ca", initial=ca_init, charge=2)
bistable_reaction = rxd.Rate(ca, -ca * (1 - ca) * (0.05 - ca))


def plot_contours(species):
  g = species.nodes.value_to_grid()
  collapsed = np.max(np.nan_to_num(g), axis=1)
  xs, ys = np.meshgrid(range(collapsed.shape[1]), range(collapsed.shape[0]))
  plt.contour(xs, ys, collapsed, 1, colors="k", linewidths=1)
  plt.axis("equal")
  plt.axis("off")


h.finitialize(-65 * mV)
plt.figure(figsize=(3, 6))
for tstop in range(0, 100 * ms, 2 * ms):
  h.continuerun(tstop)
  print(tstop)
  plot_contours(ca)
  plt.show()
```



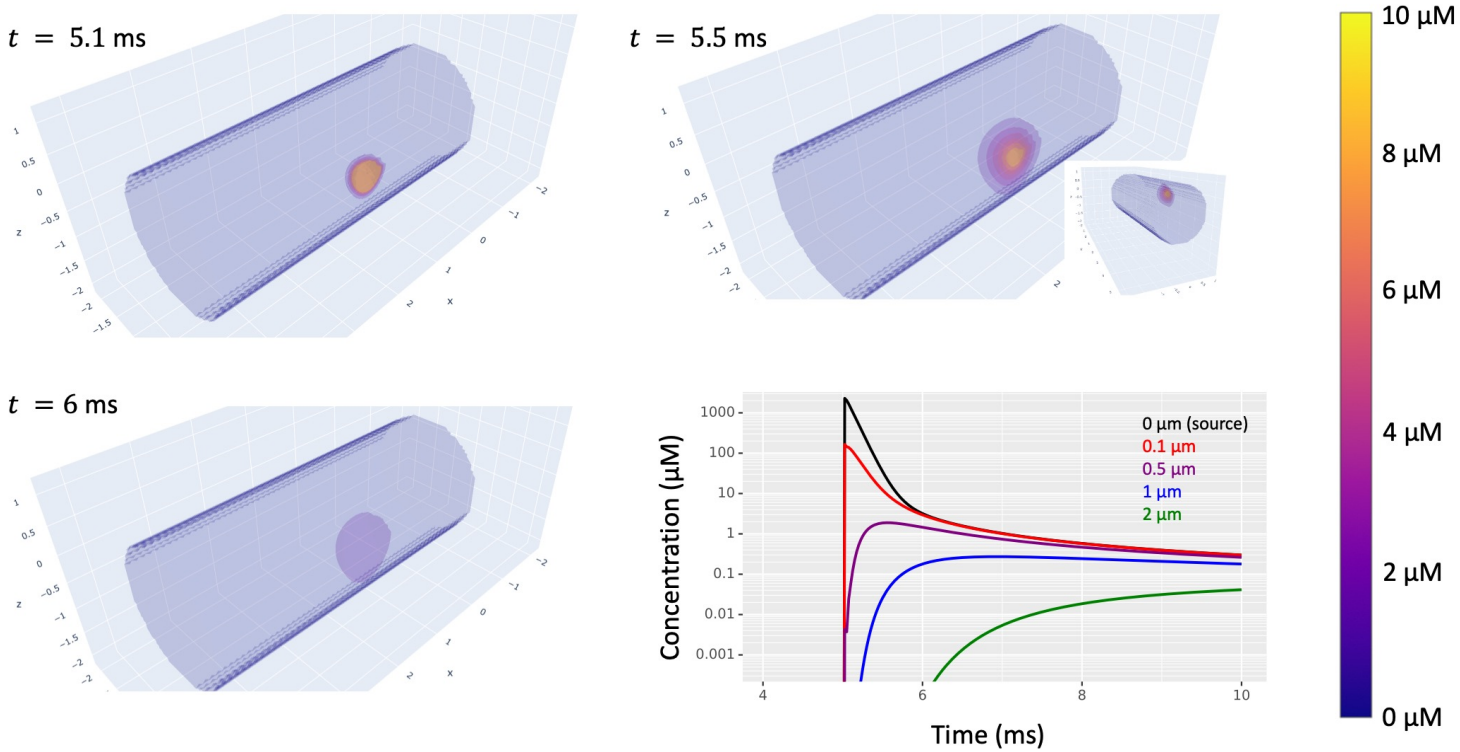
Example: Wave Curvature

- A regenerative wave front with a reflective boundary will always meet that boundary at a right angle.
- This has the effect of forcing regenerative waves to bend and slow when the geometry widens.
- Here we visualize using plotly's volume renderer.



tinyurl.com/neuron-wave-curvature

Example: 3D point source




```
NEURON {
  POINT_PROCESS RxDSyn
  RANGE tau
}

PARAMETER {
  tau = 1 (ms)
}

STATE { g }

INITIAL { g=0 }

BREAKPOINT {SOLVE state METHOD cnexp}

DERIVATIVE state { g' = -g/tau }
NET_RECEIVE(weight) {
  g = g + weight
}
```

```
node.include_flux(r._ref_g)
```

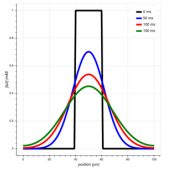
For more information

Journal articles on reaction-diffusion in NEURON	McDougal RA, Hines ML, Lytton WW. (2013). Reaction-diffusion in the NEURON simulator. *Frontiers in Neuroinformatics*, 7.
	McDougal RA, Hines ML, Lytton WW. (2013). Water-tight membranes from neuronal morphology files. *Journal of Neuroscience Methods*, 220(2), 167-178.
	Newton AJH, McDougal RA, Hines ML, Lytton WW. (2018). Using NEURON for reaction-diffusion modeling of extracellular dynamics. *Frontiers in Neuroinformatics*, 12, 41.
	McDougal RA, Conte C, Eggleston L, Newton AJH, Galijasevic H. (2022). Efficient simulations of 3D reaction-diffusion in models of neurons and networks. *Frontiers in Neuroinformatics.*
Online resources	NEURON forum https://neuron.yale.edu/forum/
	Programmer's reference http://neuronsimulator.github.io/nrn/py_doc/
	NEURON reaction-diffusion tutorials https://www.neuron.yale.edu/neuron/docs/reaction-diffusion

Exercises

tinyurl.com/neuron-diffusion-exercise

- Pure diffusion



tinyurl.com/neuron-regen-signal

- Regenerative signaling

tinyurl.com/neuron-hh-example-2

- Hodgkin-Huxley, concentrations, and homeostasis