

# The What and the Why of Neural Modeling

The moment-to-moment operation of the nervous system involves the generation, propagation and interaction of electrical and chemical signals that are distributed in space and time.

Empirically-based modeling is needed to test hypotheses about the mechanisms that govern these signals and how nervous system function emerges from these phenomena.

# Topics

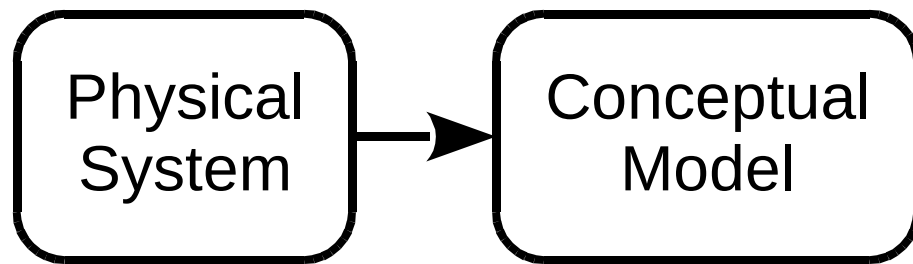
1. How to create and use models of neurons and networks of neurons
  - Specifying anatomical and biophysical properties
  - Controlling, displaying, and analyzing models and simulation results
2. How NEURON works
3. How to add user-defined mechanisms  
Ion channels, synaptic mechanisms,  
chemical signals, artificial spiking neurons . . .

# From Physical System to Computational Model



Physical  
System

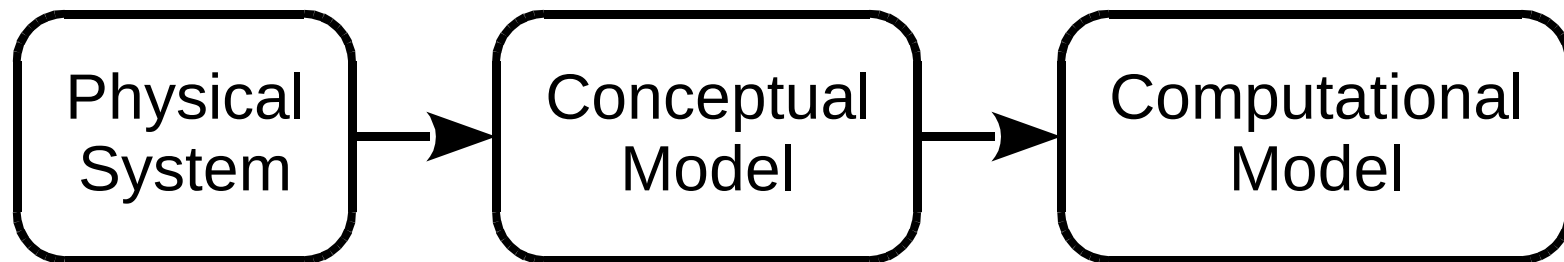
# From Physical System to Computational Model



Conceptual model

a simplified representation of the physical system

# From Physical System to Computational Model



## Conceptual model

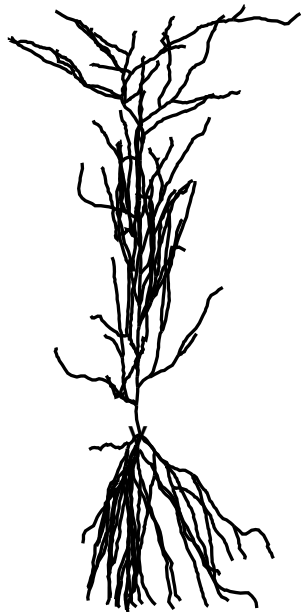
a simplified representation of the physical system

## Computational model

an accurate representation of the conceptual model

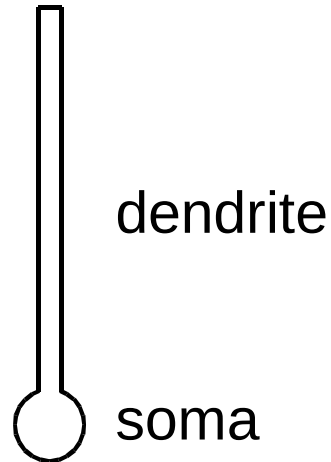
# From Physical System to Computational Model

Physical  
system



Ca1  
pyramidal  
cell

Conceptual  
model



ball  
and  
stick

Computational  
model

```
# python  
soma = h.Section(name='soma')  
dendrite = h.Section(name='dendrite')  
dendrite.connect(soma(1))
```

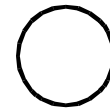
```
// hoc  
create soma, dendrite  
connect dendrite(0), soma(1)
```

source  
code

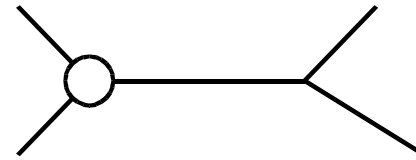
# Hierarchies of Complexity

## Structure

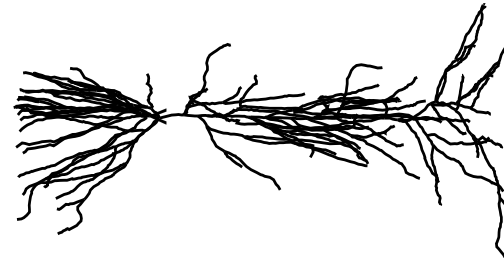
Single compartment



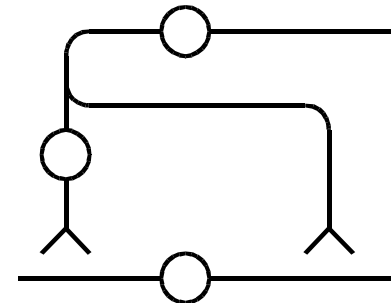
Stylized



Anatomically detailed



Network



# Hierarchies of Complexity

## Mechanism

Passive and Active currents

- HH-style

- kinetic scheme

Synaptic transmission

- continuous

- spike-triggered

Gap junctions

Extracellular fields, Linear circuits

Diffusion, buffers, transport & exchange

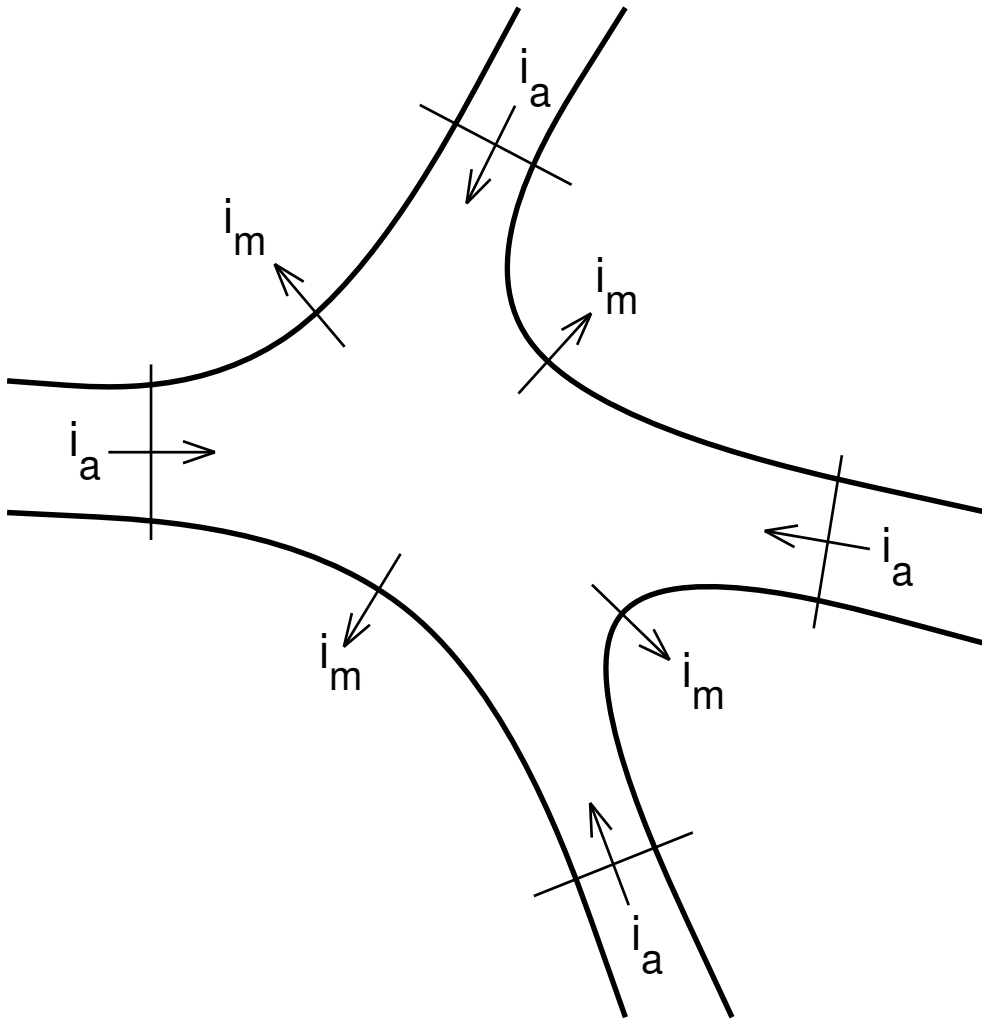
Artificial spiking cells ("integrate & fire")



# Fundamental Concepts

<b>Signals</b>	<b>What moves</b>	<b>Driving force</b>	<b>What is conserved</b>
Electrical	charge carriers	voltage gradient	charge
Chemical	solute	concentration gradient	mass

# Conservation of Charge



$$C_m \frac{dV_m}{dt} + i_{\text{ion}} = \sum i_a$$

# The Model Equations

$$c_j \frac{dv_j}{dt} + i_{ion_j} = \sum_k \frac{v_k - v_j}{r_{jk}}$$

$v_j$       membrane potential in compartment j

$i_{ion_j}$       net transmembrane ionic current in compartment j

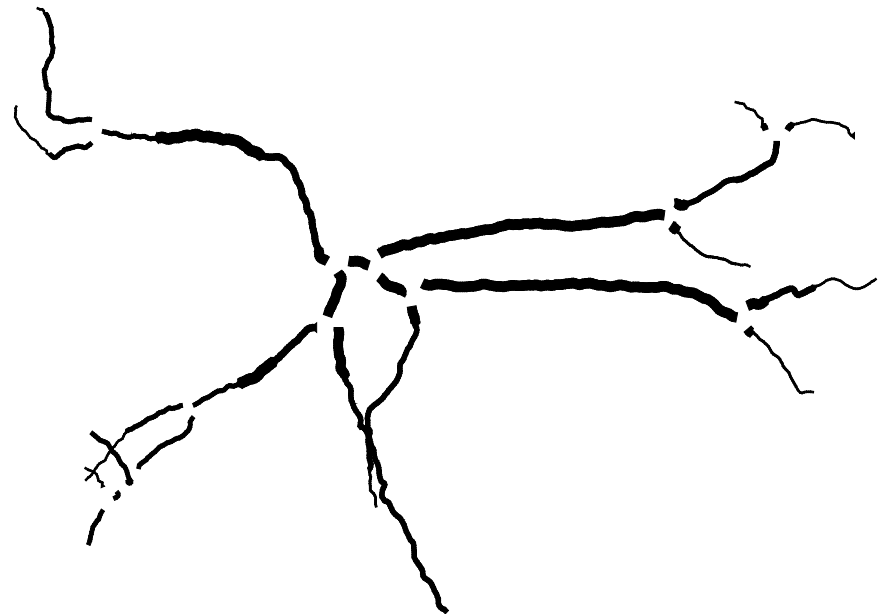
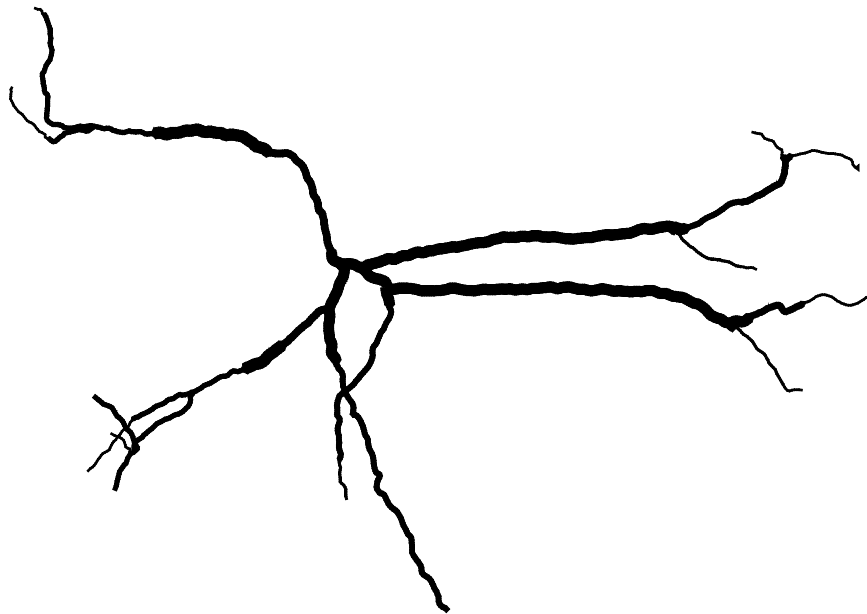
$c_j$       membrane capacitance of compartment j

$r_{jk}$       axial resistance between the centers of  
                 compartment j  
                 and  
                 adjacent compartment k

# Separating Anatomy and Biophysics from Purely Numerical Issues

section

a continuous length of unbranched cable



Anatomical data from A.I. Gulyás

# Mathematical description of a section

What we want:

$$c_j \frac{dv_j}{dt} + i_{ion_j} = \sum_k \frac{v_k - v_j}{r_{jk}}$$

What a new section gives us:

$$c_j \frac{dv_j}{dt} = \sum_k \frac{v_k - v_j}{r_{jk}}$$

i.e. membrane capacitance and axial resistance,  
but no ionic current.

How can we add ion channels, pumps, reactions . . . ?

# Adding mechanisms to sections

- Density mechanisms
  - distributed channels
  - ion accumulation

- Point processes
  - electrodes, synapses

- Described by
  - differential equations
  - kinetic schemes
  - algebraic equations

- Constructed with
  - NMODL
  - Channel Builder
  - (rxn discussed later)*

## hoc

```
create soma, dend

connect dend(0), soma(1)

soma {
    L = 50 // [um] length
    diam = 50 // [um] diameter
    nseg = 1
    insert hh // HH mechanism
}

dend {
    L = 200
    diam = 2
    nseg = 3
    insert pas // passive channels
    e_pas = -65
}
```

## Python

```
from neuron import h

soma = h.Section(name='soma')
dend = h.Section(name='dend')

dend.connect(soma(1))

soma.L = 50 # [um] length
soma.diam = 50
soma.nseg = 1
soma.insert('hh')

dend.L = 200
dend.diam = 2
dend.nseg = 3
dend.insert('pas')
dend.e_pas = -65
```

# Range Variables

Name	Meaning	Units
diam	diameter	[ $\mu\text{m}$ ]
cm	specific membrane capacitance	[ $\mu\text{f}/\text{cm}^2$ ]
g_pas (hoc) pas.g (Py)	specific conductance of the pas mechanism	[siemens/ $\text{cm}^2$ ]
v	membrane potential	[mV]



# range

normalized position along the length of a section

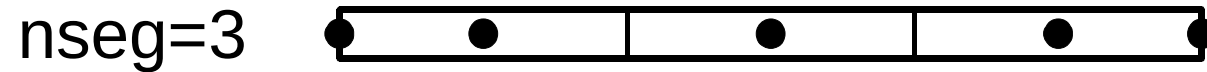
$$0 \leq \text{range} \leq 1$$

any variable name can be used for range, e.g. x



nseg

the number of points in a section at which  $v$  is calculated  
by integrating the discretized cable equation




nseg

the number of points in a section at which  $v$  is calculated  
by integrating the discretized cable equation

nseg=1 

nseg=2 

nseg=3 


Example: `axon.nseg = 3`

nseg

the number of points in a section at which v is calculated  
by integrating the discretized cable equation

nseg=1 

nseg=2 

nseg=3 

Example: `axon.nseg = 3`

To test spatial resolution

```
for sec in h.allsec():
```

```
    sec.nseg *= 3
```

and repeat the simulation

```
hoc: forall nseg *= 3
```

Syntax:

*secname(range).rangevar*

Translation: "in *secname*

at the location specified by *range*

access the value of *rangevar*"

Examples:

```
dend(0.5).v # v at middle of dend
            # hoc: dend.v(0.5)
            # shortcut: dend.v
```

```
# at each point in dend where v is calculated
#   print range, distance from 0 end, and v
for seg in dend.allseg():
    print seg.x, seg.x*dend.L, dend(seg.x).v
```

Category	Variable	Units
Time	t	[ms]
Distance	diam, L	[ $\mu\text{m}$ ]
Voltage	v	[mV]
Current		
specific	i	[mA/cm <sup>2</sup> ] (density)
absolute		[nA] (point process)
Capacitance		
specific	cm	[ $\mu\text{f/cm}^2$ ]
absolute		[nf] (point process)
Conductance		
specific	g	[S/cm <sup>2</sup> ] (density)
absolute		[ $\mu\text{S}$ ] (point process)
Cytoplasmic resistivity	Ra	[ $\Omega\text{ cm}$ ]
Resistance	SEClamp . rs	[10 <sup>6</sup> $\Omega$ ]
Concentration	cai, nao, etc.	[mM]