Modeling Spiking Neurons with Python

Byron Galbraith
Boston Python Meetup Group
May 18, 2011

About Me

- Byron Galbraith
  byron.galbraith@gmail.com

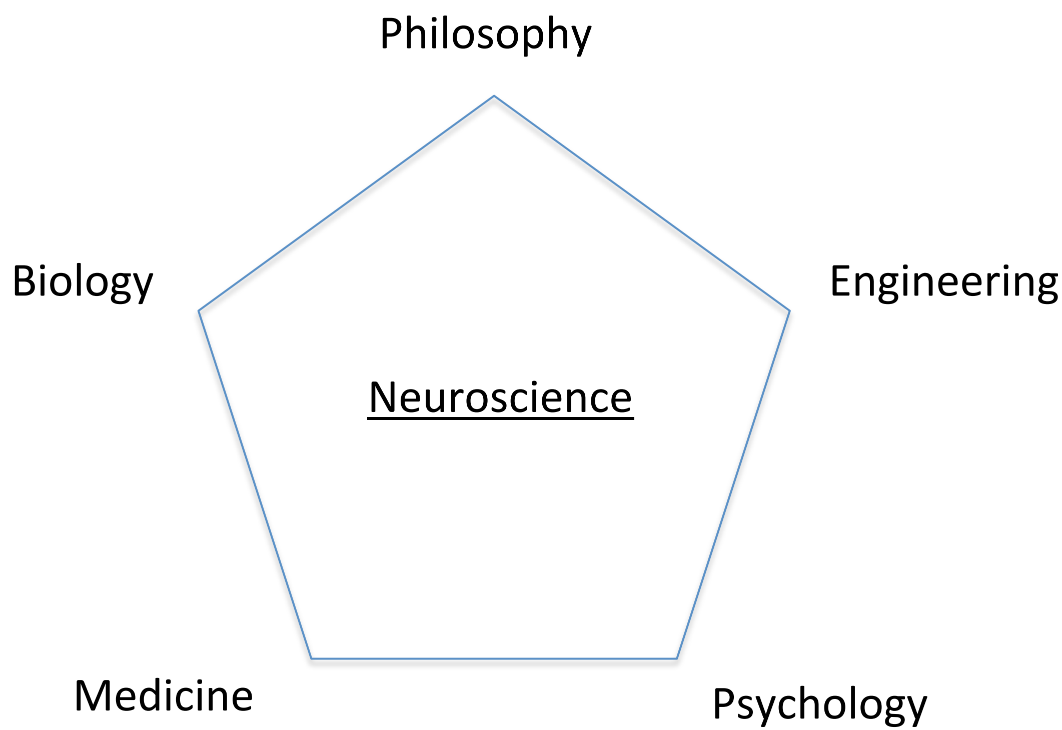
- PhD Student at Boston University
  Cognitive and Neural Systems
  Neurdon (http://www.neurdon.com/author/byron/)

- MS in Bioinformatics / BS in Bioengineering
- Full stack web developer

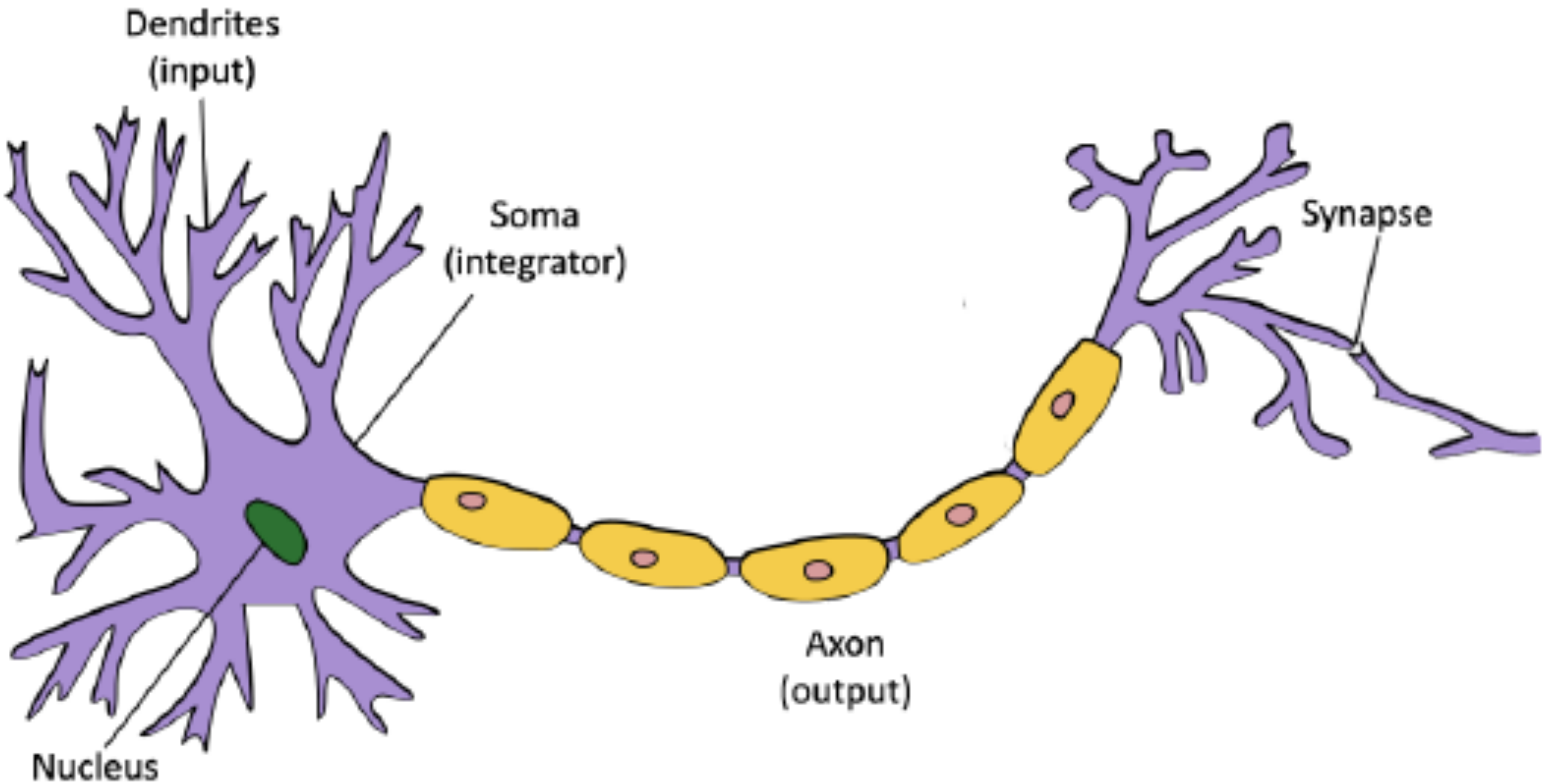
Overview

- Computational Neuroscience
- Neurons and neural activity
- Simulating neurons with Python
  Models + code snippets
- Python and other areas of Neuroscience

Computational Neuroscience

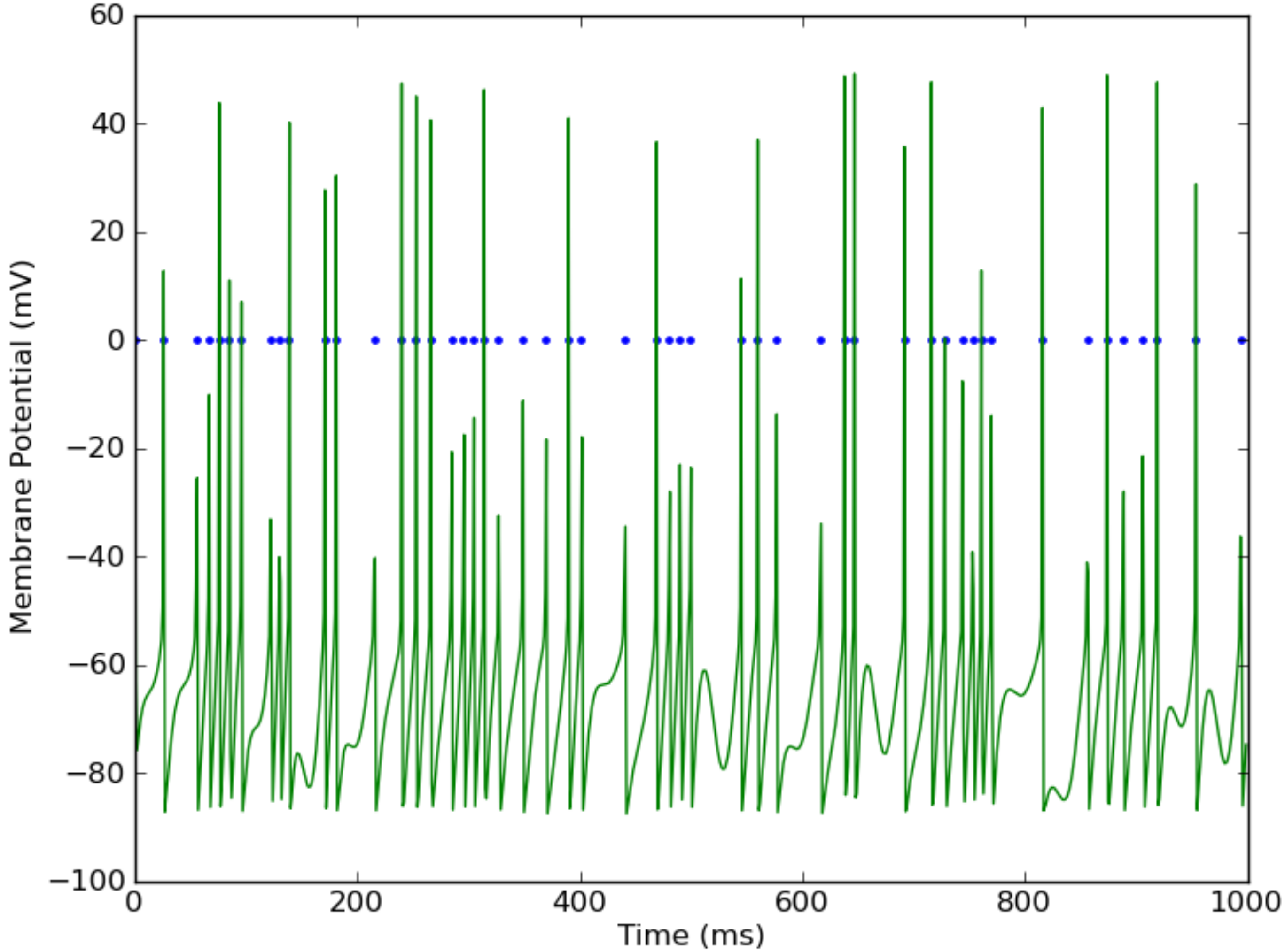


The Neuron

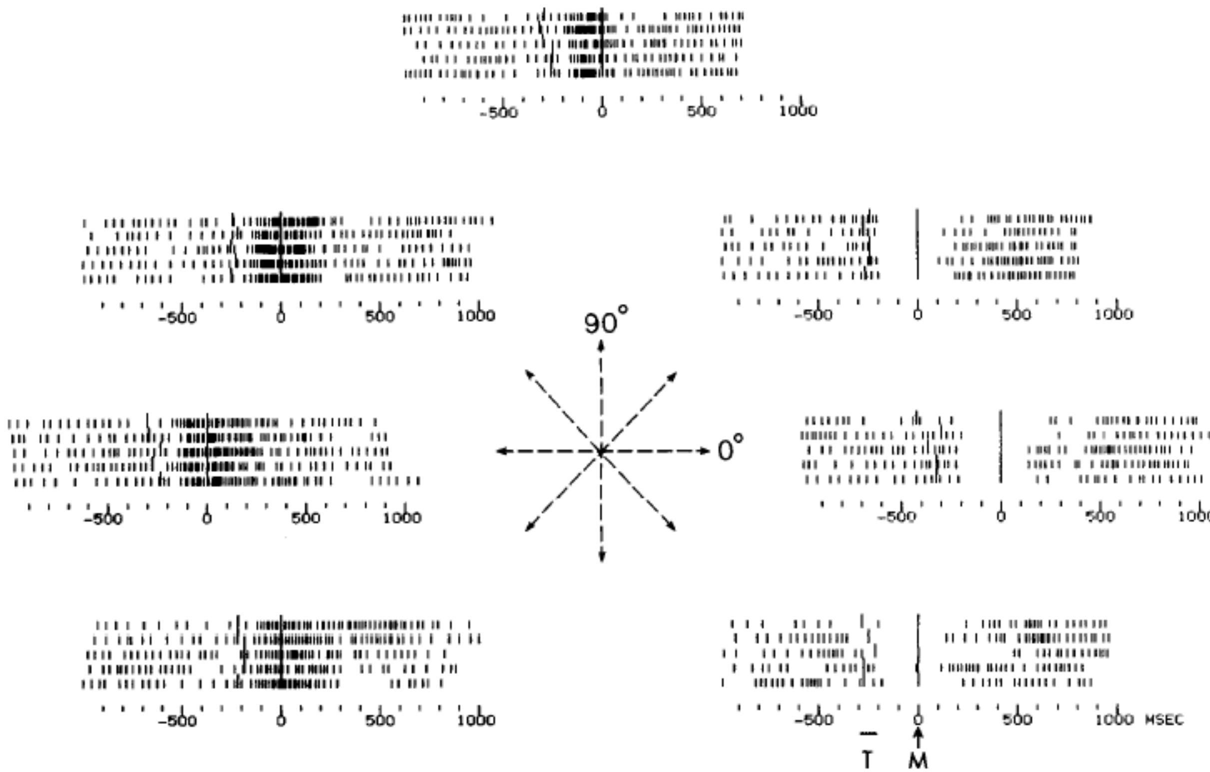


Adapted from http://upload.wikimedia.org/wikipedia/commons/b/bc/Neuron_Hand-tuned.svg

Neural Activity



Neural Activity



Georgopoulos et al. 1982

Simulating Neurons with Python

From Scratch

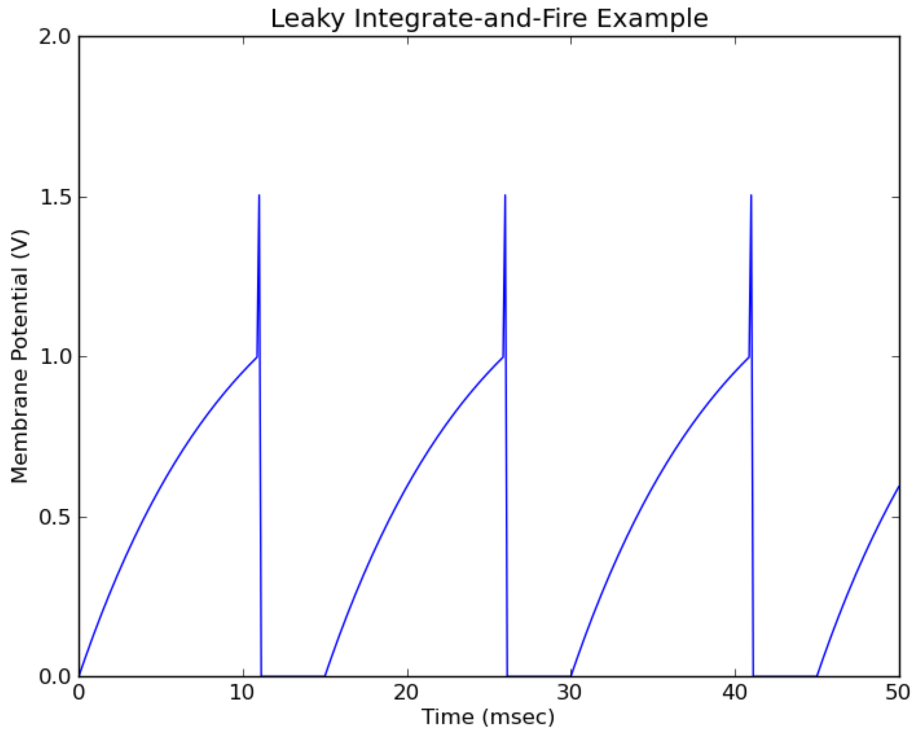
- Python (2.7)
- NumPy
- SciPy
- Matplotlib

Simulators

- Brian
- PyNEST
- PyMOOSE
- PCSIM
- NEURON / GENESIS
- PyNN

Leaky Integrate-and-Fire

$$\frac{dV}{dt} = \begin{cases} \frac{1}{\tau_m}(-V + IR_m) & t > t_{rest} \\ 0 & \text{otherwise} \end{cases}$$

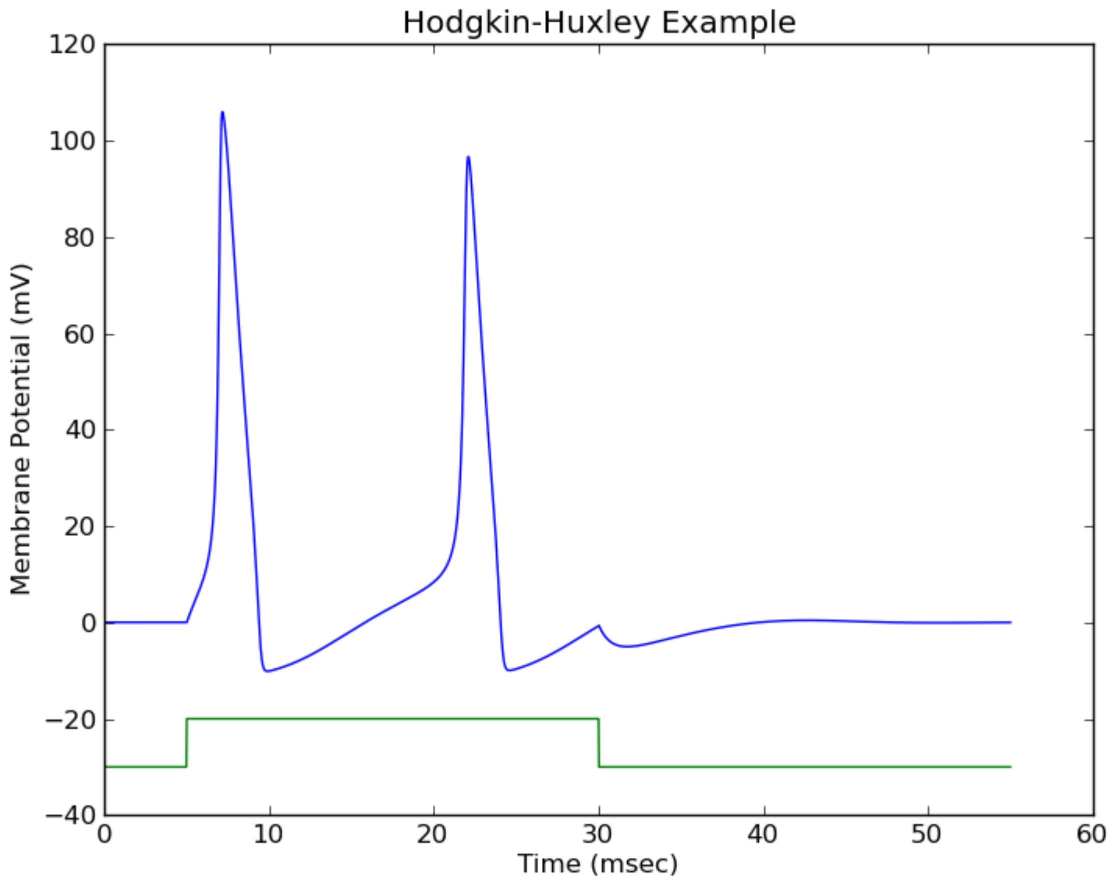


Leaky Integrate-and-Fire

```
from numpy import *
from pylab import *

## setup parameters and state variables
T       = 50                  # total time to simulate (msec)
dt      = 0.125               # simulation time step (msec)
time    = arange(0, T+dt, dt) # time array
t_rest  = 0                   # initial refractory time

## LIF properties
Vm      = zeros(len(time))    # potential (V) trace over time
Rm      = 1                   # resistance (kOhm)
Cm      = 10                  # capacitance (uF)
tau_m   = Rm*Cm               # time constant (msec)
tau_ref = 4                   # refractory period (msec)
Vth     = 1                   # spike threshold (V)
V_spike = 0.5                 # spike delta (V)

## Input stimulus
I       = 1.5                 # input current (A)
## iterate over each time step
for i, t in enumerate(time):
  if t > t_rest:
    Vm[i] = Vm[i-1] + (-Vm[i-1] + I*Rm) / tau_m * dt
    if Vm[i] >= Vth:
      Vm[i] += V_spike
      t_rest = t + tau_ref

## plot membrane potential trace
plot(time, Vm)
title('Leaky Integrate-and-Fire Example')
ylabel('Membrane Potential (V)')
xlabel('Time (msec)')
ylim([0,2])
show()
```

Hodgkin-Huxley

$$\frac{dV}{dt} = \frac{1}{C_m}(I_m - g_{Na}m^3h(V - E_{Na}) - g_K n^4(V - E_K) - g_l(V - E_l))$$



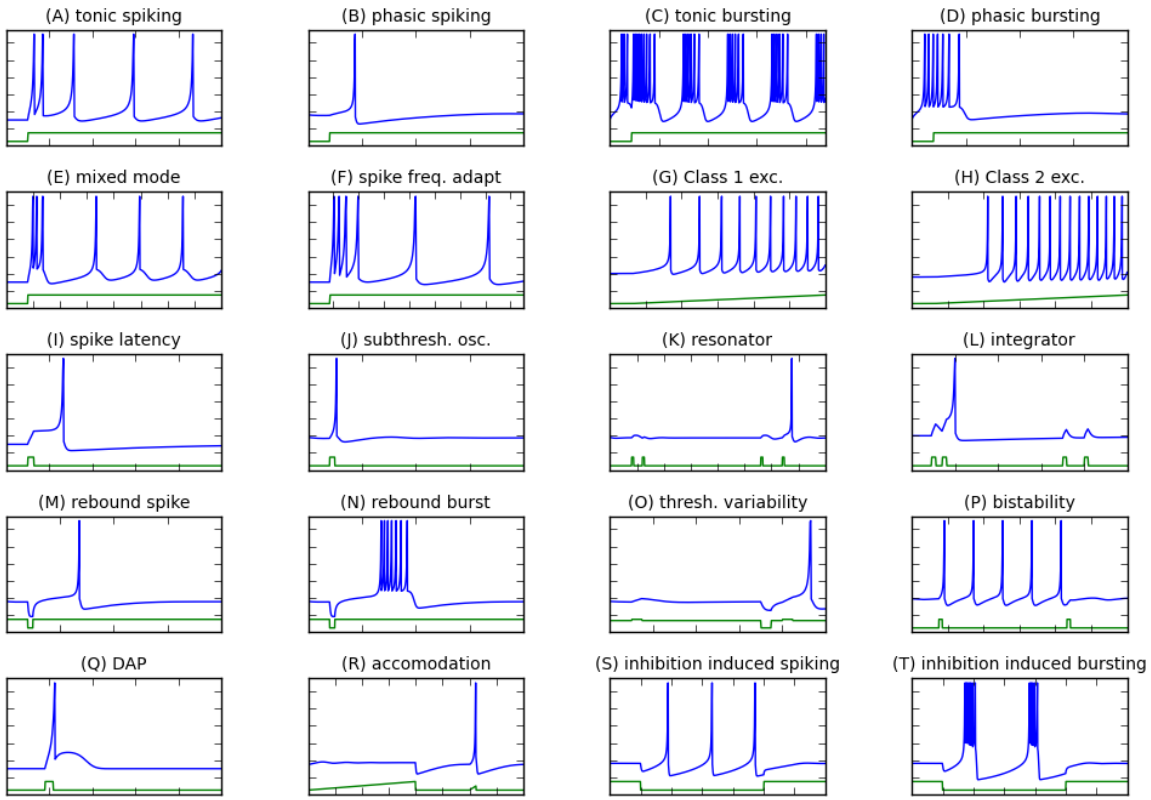
Hodgkin-Huxley

```
# Na channel (activating)
alpha_m = vectorize(lambda v: 0.1*(-v + 25)/(exp((-v + 25)/10) - 1) if v != 25 else 1)
beta_m  = lambda v: 4*exp(-v/18)

# Na channel (inactivating)
alpha_h = lambda v: 0.07*exp(-v/20)
beta_h  = lambda v: 1/(exp((-v + 30)/10) + 1)

...

## Stimulus
I = zeros(len(time))
for i, t in enumerate(time):
  if 5 <= t <= 30: I[i] = 10 # uA/cm2

## Simulate Model
for i in range(1,len(time)):
  g_Na = gbar_Na*(m**3)*h
  g_K  = gbar_K*(n**4)
  g_l  = gbar_l

  m += dt*(alpha_m(Vm[i-1])*(1 - m) - beta_m(Vm[i-1])*m)
  h += dt*(alpha_h(Vm[i-1])*(1 - h) - beta_h(Vm[i-1])*h)
  n += dt*(alpha_n(Vm[i-1])*(1 - n) - beta_n(Vm[i-1])*n)

  Vm[i] = (Vm[i-1] + (I[i-1] - g_Na*(Vm[i-1] - E_Na) -
           g_K*(Vm[i-1] - E_K) - g_l*(Vm[i-1] - E_l)) /
           Cm * dt)
```

Izhikevich

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(bv - u)$$



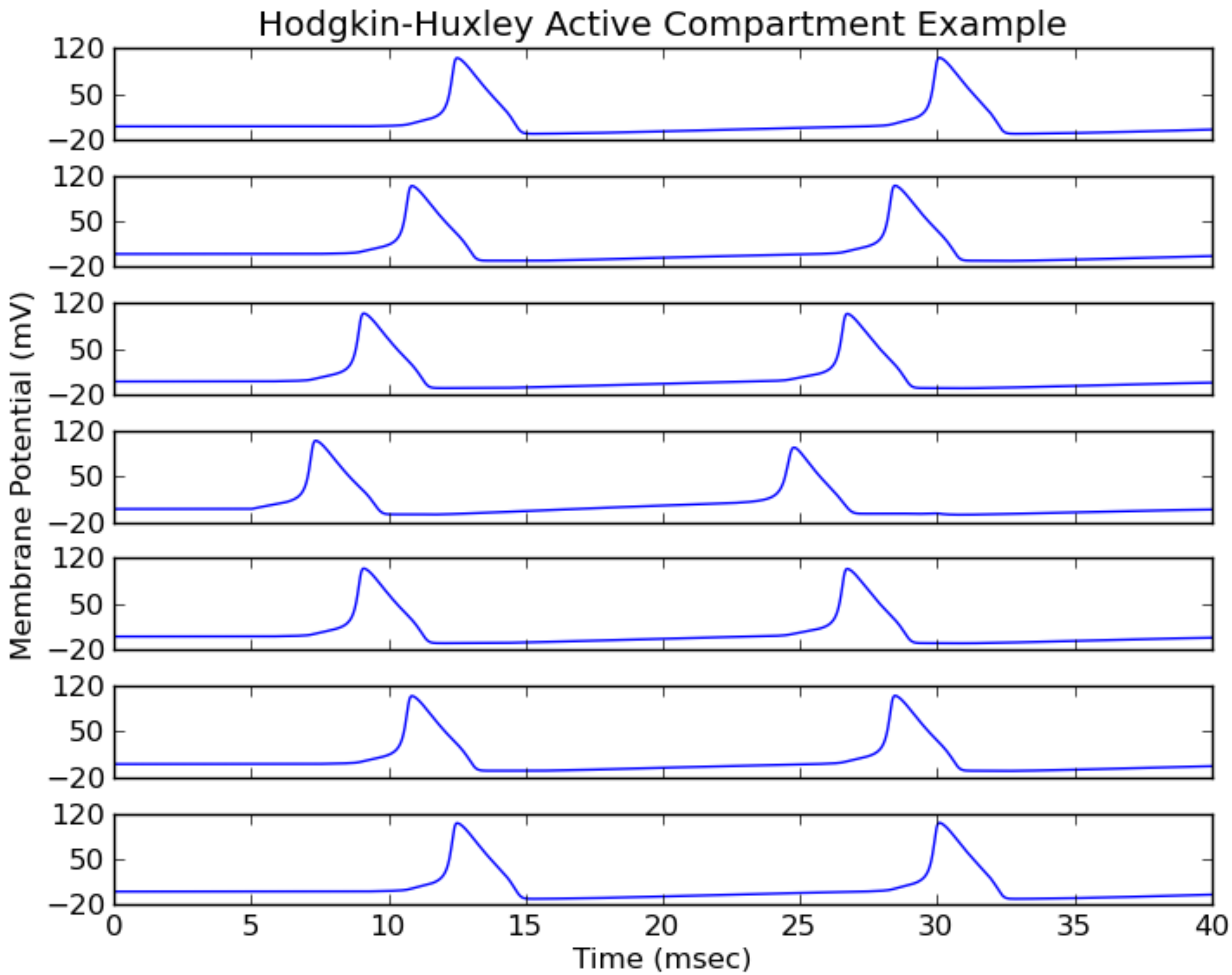
Izhikevich

```
class IzhNeuron:
  def __init__(self, label, a, b, c, d, v0, u0=None):
    self.label = label

    self.a = a
    self.b = b
    self.c = c
    self.d = d

    self.v = v0
    self.u = u0 if u0 is not None else b*v0


class IzhSim:
  def __init__(self, n, T, dt=0.25):
    self.neuron = n
    self.dt     = dt
    self.t      = t = arange(0, T+dt, dt)
    self.stim   = zeros(len(t))
    self.x      = 5
    self.y      = 140
    self.du     = lambda a, b, v, u: a*(b*v - u)

  def integrate(self, n=None):
    if n is None: n = self.neuron
    trace = zeros((2,len(self.t)))
    for i, j in enumerate(self.stim):
      n.v += self.dt * (0.04*n.v**2 + self.x*n.v + self.y - n.u + self.stim[i])
      n.u += self.dt * self.du(n.a,n.b,n.v,n.u)
      if n.v > 30:
        trace[0,i] = 30
        n.v        = n.c
        n.u       += n.d
      else:
        trace[0,i] = n.v
        trace[1,i] = n.u
    return trace
```

Izhikevich

```
## (A) tonic spiking
n = IzhNeuron("(A) tonic spiking", a=0.02, b=0.2, c=-65, d=6, v0=-70)
s = IzhSim(n, T=100)
for i, t in enumerate(s.t):
  s.stim[i] = 14 if t > 10 else 0
sims.append(s)

## (B) phasic spiking
n = IzhNeuron("(B) phasic spiking", a=0.02, b=0.25, c=-65, d=6, v0=-64)
s = IzhSim(n, T=200)
for i, t in enumerate(s.t):
  s.stim[i] = 0.5 if t > 20 else 0
sims.append(s)

# Simulate
fig = figure()
fig.set_title('Izhikevich Examples')
for i,s in enumerate(sims):
  res = s.integrate()
  ax  = subplot(5,4,i+1)

  ax.plot(s.t, res[0], s.t, -95 + ((s.stim - min(s.stim))/(max(s.stim) - min(s.stim)))*10)

  ax.set_xlim([0,s.t[-1]])
  ax.set_ylim([-100, 35])
  ax.set_title(s.neuron.label, size="small")
  ax.set_xticklabels([])
  ax.set_yticklabels([])
show()
```

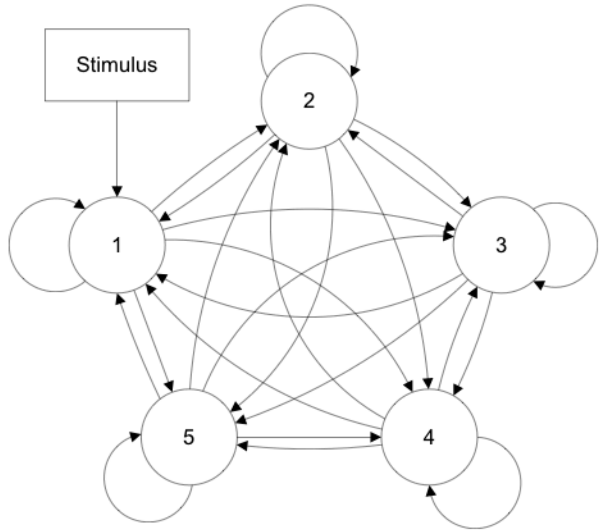
Compartmental (Cable)

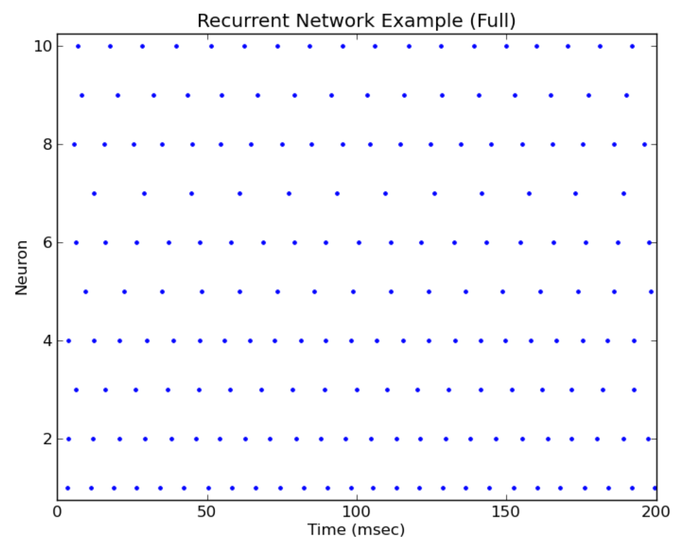


Compartmental (Cable)

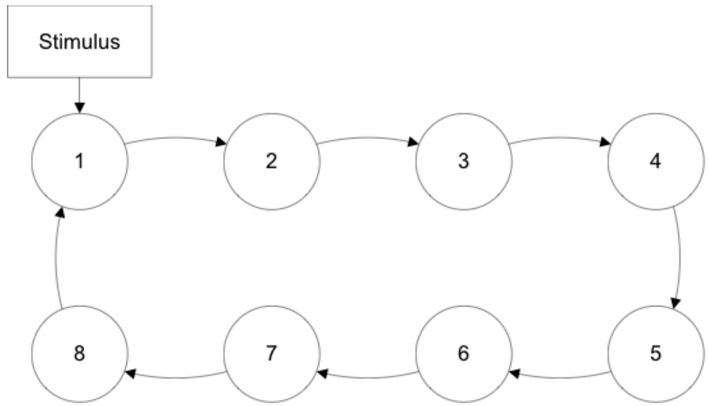
```
## connection matrix
Sc = zeros([S,S])
for i in range(S):
  if i == 0:
     Sc[i,0:2]     = [1,-1]
  elif i == S-1:
     Sc[i,i-1:S]   = [-1,1]
  else:
     Sc[i,i-1:i+2] = [-1,2,-1]

## simulate model
for i in range(1,len(time)):
  g_Na = gbar_Na*(m**3)*h
  g_K  = gbar_K*(n**4)
  g_l  = gbar_l

  m += dt*(alpha_m(Vm[:,i-1])*(1 - m) - beta_m(Vm[:,i-1])*m)
  h += dt*(alpha_h(Vm[:,i-1])*(1 - h) - beta_h(Vm[:,i-1])*h)
  n += dt*(alpha_n(Vm[:,i-1])*(1 - n) - beta_n(Vm[:,i-1])*n)

  dV = -hh(Vm[:,i-1], g_Na, g_K, g_l) - Sc.dot(Vm[:,i-1]) / Ra
  dV[elec] += I[i-1]

  Vm[:,i] = Vm[:,i-1] + dt * dV / Cm
```

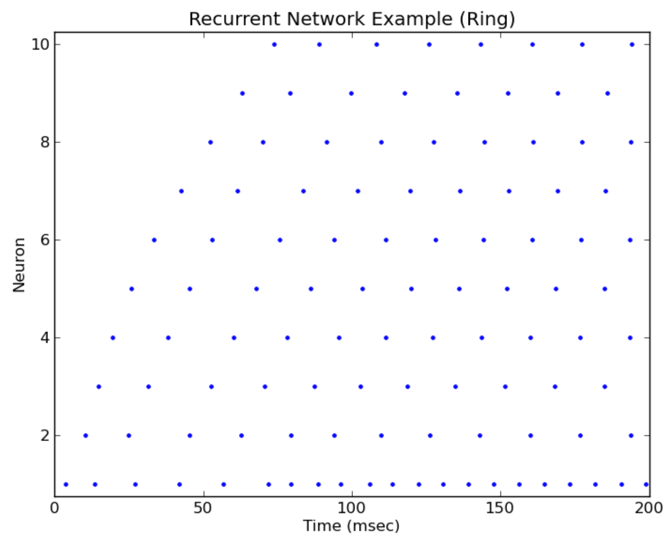
Recurrent Spiking Networks





Recurrent Spiking Networks





Recurrent Spiking Networks

```
## Synapse weight matrix
# equally weighted ring connectivity
synapses = np.eye(N)
synapses = np.roll(synapses, -1, 1)

# randomly weighted full connectivity
#synapses = np.random.rand(N,N)*0.3

## Synapse current model
def Isyn(t):
    '''t is an array of times since each neuron's last spike event'''
    t[np.nonzero(t < 0)] = 0
    return t*np.exp(-t/tau_psc)
last_spike = np.zeros(N) - tau_ref

## Simulate network
raster = np.zeros([N,len(time)])*np.nan
for i, t in enumerate(time[1:],1):
    active = np.nonzero(t > last_spike + tau_ref)
    Vm[active,i] = Vm[active,i-1] + (-Vm[active,i-1] + I[active,i-1]) / tau_m * dt

    spiked = np.nonzero(Vm[:,i] > Vth)
    last_spike[spiked] = t
    raster[spiked,i] = spiked[0]+1
    I[:,i] = Iext + synapses.dot(Isyn(t - last_spike))
```

Python and…

- Computer vision

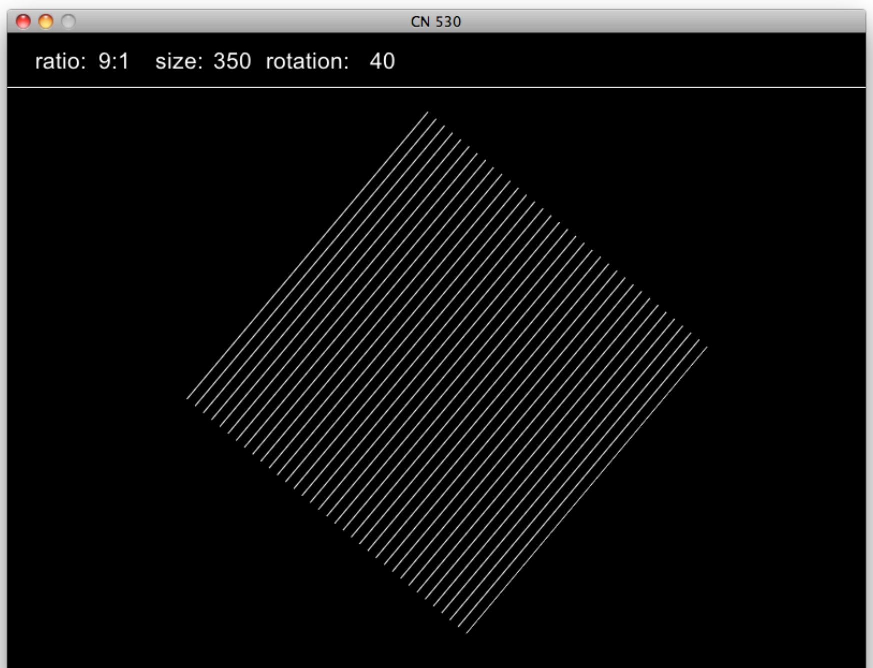
  Space variant image processing (PyCUDA)

  Modeling how the retina and visual cortex map what we see

Python and...

- Psychophysics

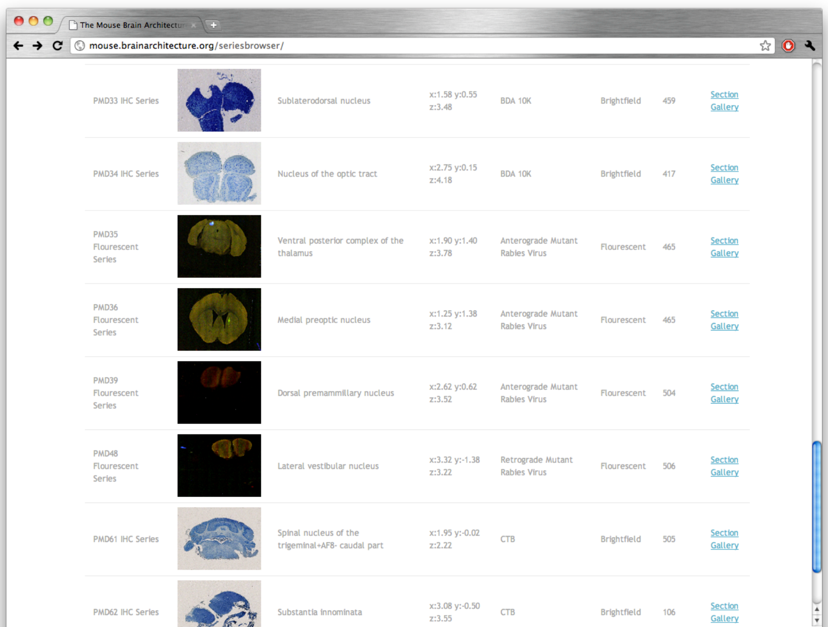
  Exploring perception of color in monochrome gratings (pyglet)



Python and...

- Brain Architecture

  Web site devoted to searching and viewing very high resolution images of histological preparations (Django)

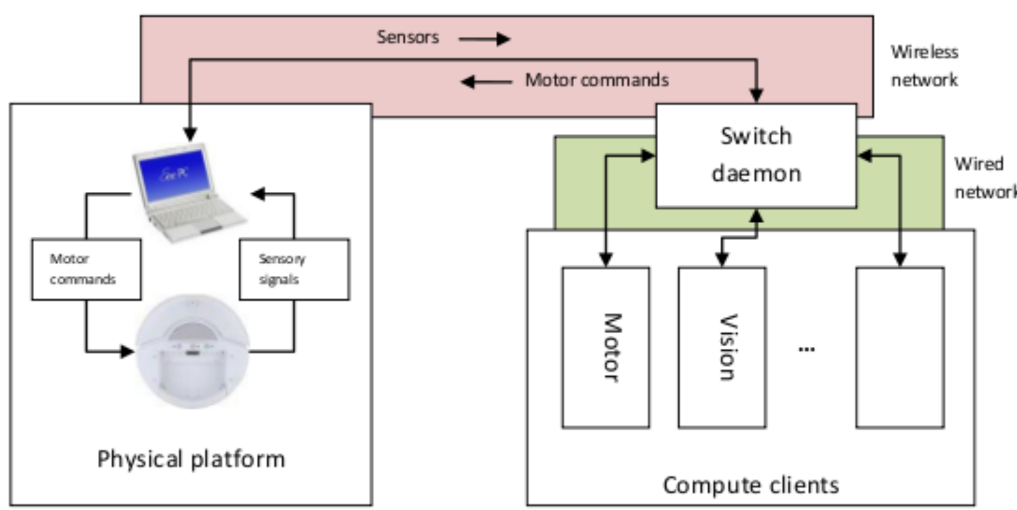


Python and…

- Neuromorphic Robotics

  Asimov: Middleware for Modeling the Brain on the iRobot Create

  Brain-Machine Interfaces (BCI)



Thank You