Tutorial 3: Single Neuron Modeling

Jimin Kim (jk55@uw.edu)

University of Washington

OUTLINE

Part 1: Modeling a graded neuron with simple cubic model

- RIM inter neuron

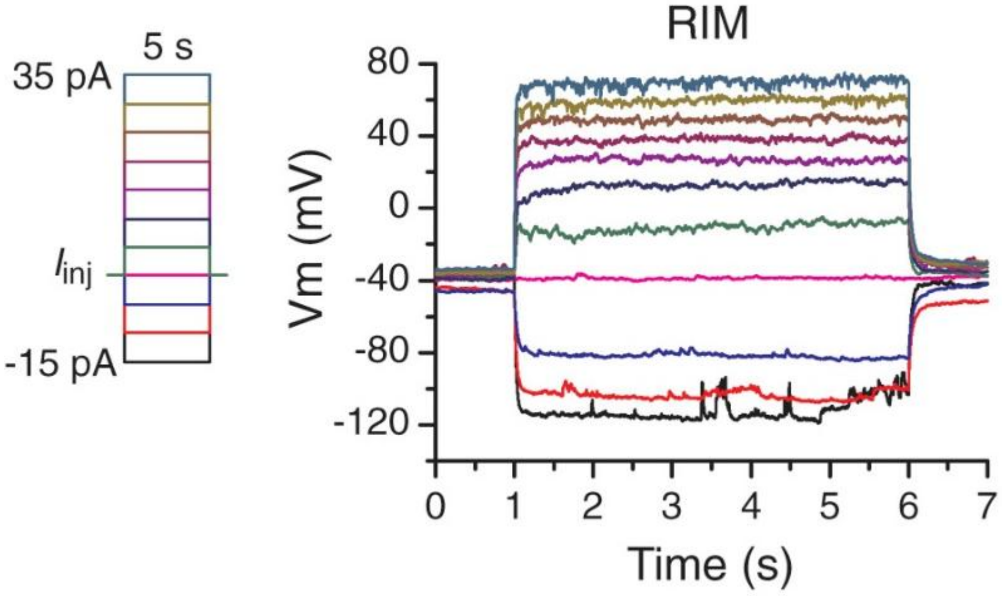
Part 2: Modeling a spiking neuron with Hodgkin-Huxley model

- AWA sensory neuron

Graded neuron example: RIM neuron

(Jupyter Notebook: 1. Single neuron modeling - Simple RIM neuron.ipynb)

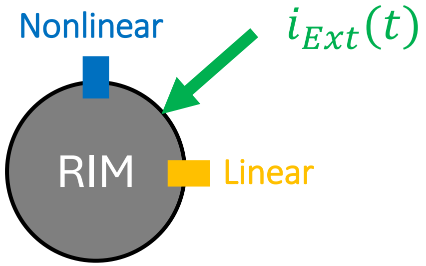
Electrophysiological recording



Liu et al, 2018,
Cell

Model

$$\frac{dV}{dt} = (-(\underbrace{aV^3 + bV^2}_{\text{Nonlinear channel}} + \underbrace{cV + d}_{\text{Linear channel}}) + \underbrace{i_{Ext}(t)}_{\text{External input}})/\tau$$



Naudin et al, 2022,
Neural Computation

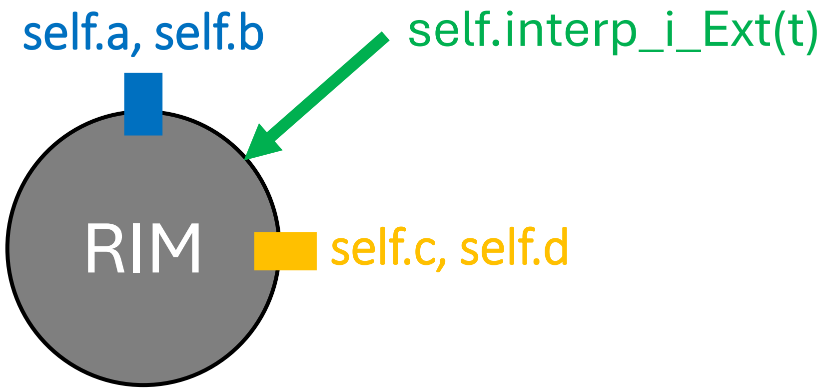
Defining the simple cubic model

Model name

```python
class Celegans_SimpleRIM:

    def __init__(self):

        self.network_Size = 1                  # of neurons

        self.a = 0.000024
        self.b = 0.0036
        self.c = 0.31                          Model parameters
        self.d = 7.22
        self.tau = 0.0042

        self.initcond = np.array([-75])        Initial condition
        self.timescale = 0.001                 Integration time step (s)

    def forward_Network(self, t, y):

        fV = -(self.a*y**3 + self.b*y**2 + self.c*y + self.d)
        i_Ext = self.interp_i_Ext(t)

        dv = (fV + i_Ext)/self.tau

        return dv
```

Model definition

Model dynamics



$$\frac{dV}{dt} = (\underbrace{-(aV^3 + bV^2 + cV + d)}_{f(V)} + i_{Ext}(t))/\tau$$

Configuring stimulus to neuron

```
rim_neuron = Celegans_SimpleRIM()
```

Initialize the model class

```
RIM_current_clamp_list = np.arange(-15, 40, 5)

simulation_time = 5
simulation_steps = int(simulation_time/rim_neuron.timescale)

input_mat_list = []

for iext in RIM_current_clamp_list:

    input_mat = np.zeros((simulation_steps, rim_neuron.network_Size))
    input_mat[:, 0] = iext
    input_mat_list.append(input_mat)
```

Define a list of input stimulus amplitudes (e.g., -15pA → 35pA with 5pA increment

Define total number of simulation timesteps (5 seconds → 5000 steps)

For each stimulus amplitude, populate a 1D time dependent stimulus array (5000, 1) and append to a list

Simulating the simple cubic model

```
v_sol_list = []

for input_mat in input_mat_list:

    solution_dict = n_sim.run_network(rim_neuron, input_mat)
    v_sol_list.append(solution_dict['v_solution'])
```

Initialize the list for storing the voltage solution

Simulate the model for each input stimulus vector and store the solution (dim = (5000,)) to the list

```
v_sol_list = np.vstack(v_sol_list)
```

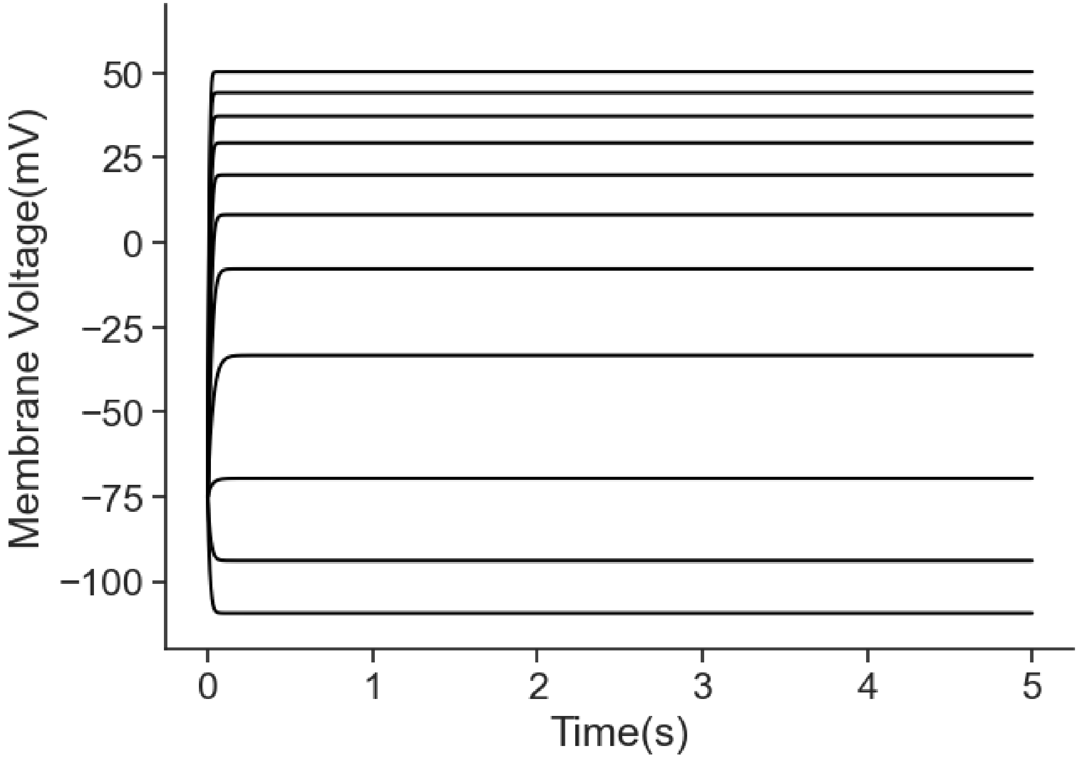
Stack the solutions into a single numpy array (dim = (11, 5000))

Plotting the simulated results

Using matplotlib.pyplot() for plotting

```
plt.figure(figsize = (7, 5))

plt.plot(np.arange(0, 5, 0.001), v_sol_list.T, color = 'black')
plt.xlabel('Time(s)')
plt.ylabel('Membrane Voltage(mV)')
plt.ylim(-120, 70)
sns.despine()
```

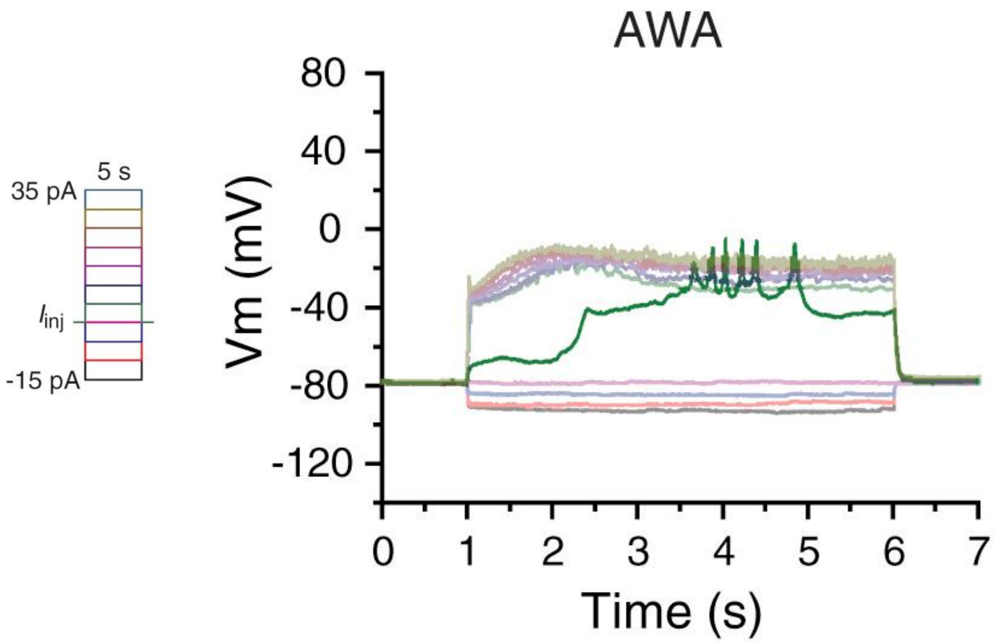
Simulated RIM neuron



Spiking neuron example: AWA neuron

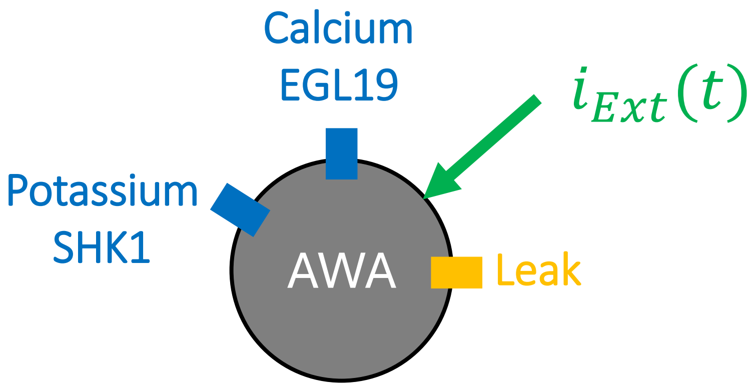
(Jupyter Notebook: 1. Single neuron modeling - HH AWA neuron.ipynb)

Electrophysiological recording



Model

$$\frac{dV}{dt} = (-\underbrace{(g_l(V - V_l)}_{\text{Leak current}} + \underbrace{g_{Ca}(c_1 + f c_2)(V - V_{Ca})}_{\text{Calcium current (EGL19)}} + \underbrace{g_k m(V - V_k)}_{\text{Potassium current (SHK1)}}) + \underbrace{i_{Ext}(t)}_{\text{External input}})/C$$



Liu et al, 2018, Cell

Neuron channel database & Units

```
"SHK1_awa": {
    "name": "SHK1_awa",
    "num_params":27,
    "param_list":["gK", "VK", "vk1", "vk2", "gKI", "gK2", "gK3", "gK4", "gK5", "gK6",
                  "gK7", "TK", "vq1", "vq2", "vs1", "vs2", "TS", "TS2", "vb1", "vb2",
                  "Tbk", "vp1", "vp2", "vtk1", "vtk2", "TKL", "TKH"],
    "num_vars": 5,
    "var_list":["w", "bk", "slo", "slo2", "kb"]

},

"EGL19_awa": {
    "name": "EGL19_awa",
    "num_params":12,
    "param_list":["gCa", "VCa", "vm1", "vm2", "vm3", "vm4", "vt1", "vt2", "mx", "TC1",
                  "TC2", "fac"],
    "num_vars": 2,
    "var_list":["c1", "c2"]

},
```

channels.json contains **parameter names** and **dynamic variables** for nonlinear neuron channels modules included in modWorm

Capacitance (C): nF

Voltage (V): mV

Conductance (g): nS

Current (i): pA

Time (t): $seconds$

Units for modeling and simulations

Defining the HH spiking neuron with in-built channel modules

```python
class Celegans_SpikingAWA:

    def __init__(self):

        self.network_Size = 1

        self.neuron_C = n_dyn.init_neuron_C(capacitance = np.array([0.0015]))
        self.neuron_Linear = n_dyn.init_neuron_Linear(conductance = np.array([0.25]),
                                                      leak_voltage = np.array([-65]))
```

Pre-built Cell capacitance, Leak channel module

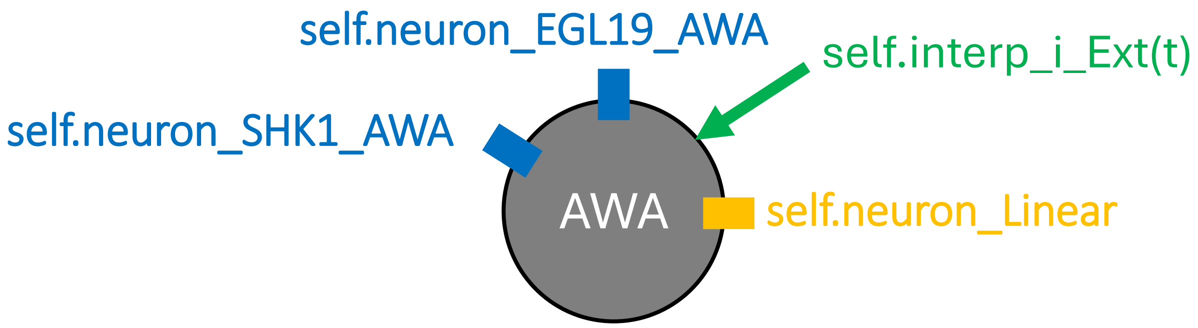
```python
        self.neuron_EGL19_AWA = n_dyn.init_neuron_Nonlinear(self, channel_type = 'EGL19_awa',
                                                            neuron_inds = [0],
                                                            params_mat = EGL19_params,
                                                            added_order = 1,
                                                            initconds_mat = EGL19_initcond,
                                                            using_julia = False)
```

Pre-built Calcium channel (EGL19) module

```python
        self.neuron_SHK1_AWA = n_dyn.init_neuron_Nonlinear(self, channel_type = 'SHK1_awa',
                                                           neuron_inds = [0],
                                                           params_mat = SHK1_params,
                                                           added_order = 2,
                                                           initconds_mat = SHK1_initcond,
                                                           using_julia = False)
```

Pre-built Potassium channel (SHK1) module

```python
        # Neural initial condition
        self.initcond = n_sim.init_Initcond(self)
        self.initcond[0] = -75.
        self.timescale = 0.001
```



$$[V, \underbrace{c1, c2}_{\substack{\text{EGL19 variables} \\ \boxed{\text{added\_order = 1}}}}, \underbrace{w, bk, slo, kb}_{\substack{\text{SHK1 variables} \\ \boxed{\text{added\_order = 2}}}}]$$

Defining the HH spiking neuron with in-built channel modules

```
def forward_Network(self, t, y):

    v, channel_vars = np.array([y[0]]), y[1:len(y)]

    i_Linear = n_dyn.fwd_i_Linear(self, v)
    i_EGL19, dEGL19 = n_dyn.fwd_i_EGL19_AWA(self, v, channel_vars)
    i_SHK1, dSHK1 = n_dyn.fwd_i_SHK1_AWA(self, v, channel_vars)
    i_Ext = self.interp_i_Ext(t)

    dv = (-(i_Linear + i_EGL19 + i_SHK1) + i_Ext)/self.neuron_C

    return np.concatenate([dv, dEGL19, dSHK1])
```

Channel variables

Split y into voltage and channel variables

Use dynamics functions fwd_i_xyz() associated with channel modules to compute currents and channel variables dynamics

Compute final derivatives for voltage and channel variables and concatenate them into a single 1D array

$$\frac{dV}{dt} = (-(\underbrace{g_l(V - V_l)}_{\text{i\_Linear}} + \underbrace{g_{Ca}(c_1 + f c_2)(V - V_{Ca})}_{\text{i\_EGL19}} + \underbrace{g_k m(V - V_k)}_{\text{i\_SHK1}}) + i_{Ext}(t))/C$$

Defining the HH spiking neuron with in-built channel modules

Pre-built channel modules	Names in __init__	Names in forward_Network
init_neuron_Linear()	self.neuron_Linear	fwd_i_Linear()
init_neuron_Nonlinear(channel_type = ‘EGL19_AWA’)	self.neuron_EGL19_AWA	fwd_i_EGL19_AWA()
init_neuron_Nonlinear(channel_type = ‘SHK1_AWA’)	self.neuron_SHK1_AWA	fwd_i_SHK1_AWA()

Configuring stimulus to neuron

```
awa_neuron = Celegans_SpikingAWA()
```

Initialize the model class

```
AWA_current_clamp_list = np.arange(-15, 40, 5)

simulation_time = 7
simulation_steps = int(simulation_time/awa_neuron.timescale)

input_mat_list = []

for iext in AWA_current_clamp_list:

    input_mat = np.zeros((simulation_steps, awa_neuron.network_Size))
    input_mat[1000:6000, 0] = iext
    input_mat_list.append(input_mat)
```

Define a list of input stimulus amplitudes

Define total number of simulation timesteps (7 seconds → 7000 steps)

For each stimulus amplitude, populate a 1D time dependent stimulus array (pulse from 1s – 6s) and append to a list

Simulating the HH spiking model

```python
v_sol_list = []

for input_mat in input_mat_list:

    solution_dict = n_sim.run_network(awa_neuron, input_mat)
    v_sol_list.append(solution_dict['v_solution'])
```

Initialize the list for storing the voltage solution

Simulate the model for each input stimulus vector and store the solution (dim = (7000,)) to the list

```python
v_sol_list = np.vstack(v_sol_list)
```

Stack the solutions into a single numpy array (dim = (11, 7000))

Plotting the simulated results

Using matplotlib.pyplot() for plotting

```
plt.figure(figsize = (10, 5))

plt.plot(np.arange(0, 7, 0.001), v_sol_list.T, color = 'black')
plt.xlabel('Time(s)')
plt.ylabel('Membrane Voltage(mV)')
#plt.ylim(-90, 0)
sns.despine()
```

Simulated AWA neuron