Tutorial 2:
Getting started with modWorm

Jimin Kim (jk55@uw.edu)

University of Washington

OUTLINE

Part 1: Quick start modWorm

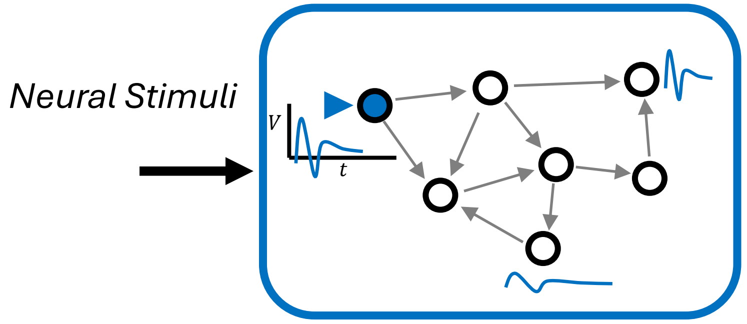
- Nervous system simulation
- Biomechanics simulation
- Linking Nervous system and Biomechanics with proprioceptive feedback

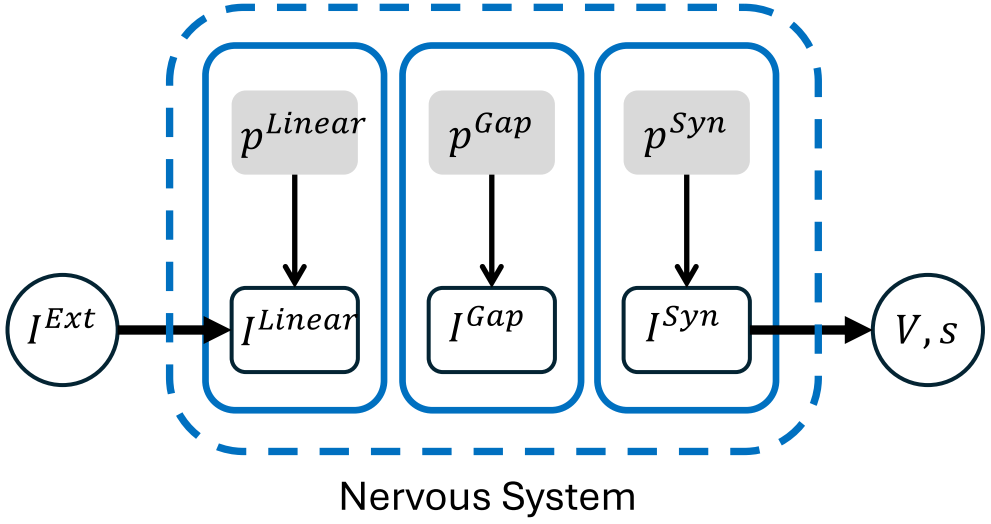
Part 2: Custom start modWorm

- Connectome variations
- Body-Environmental variations

Nervous system simulation

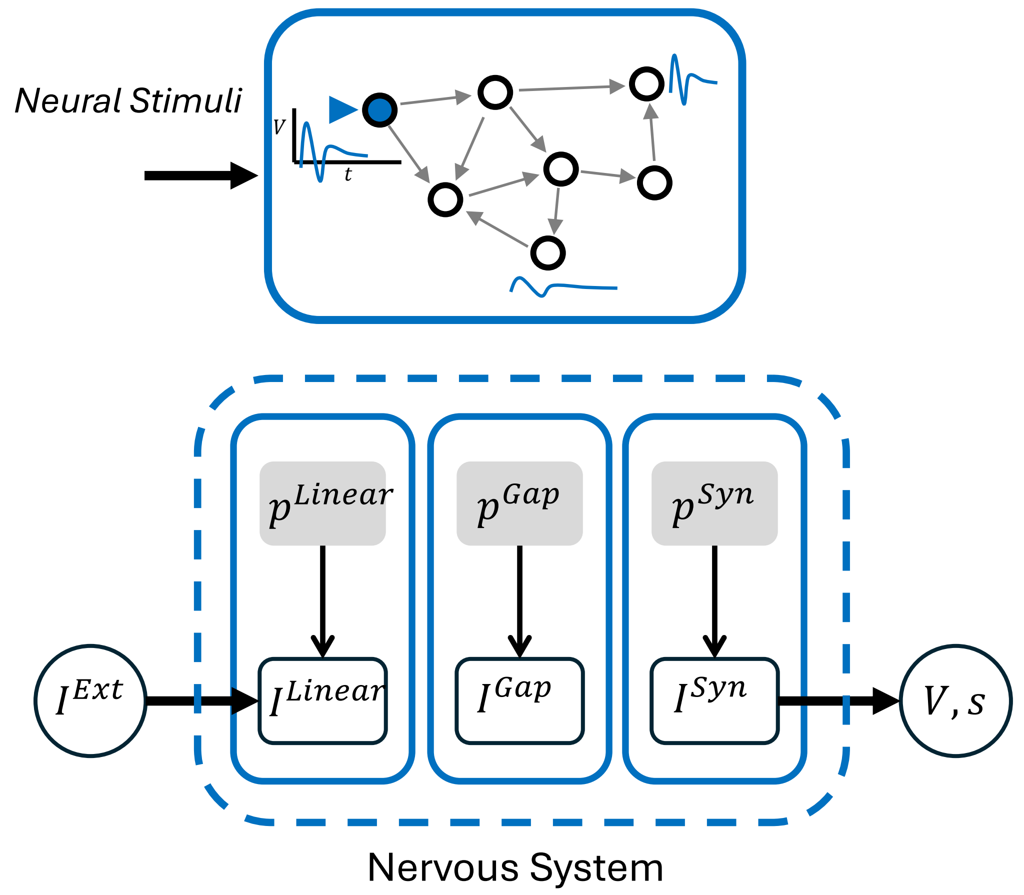
(Jupyter Notebook: 1.1 Quick start (Nervous System).ipynb)





Nervous system simulation

(Jupyter Notebook: 1.1 Quick start (Nervous System).ipynb)

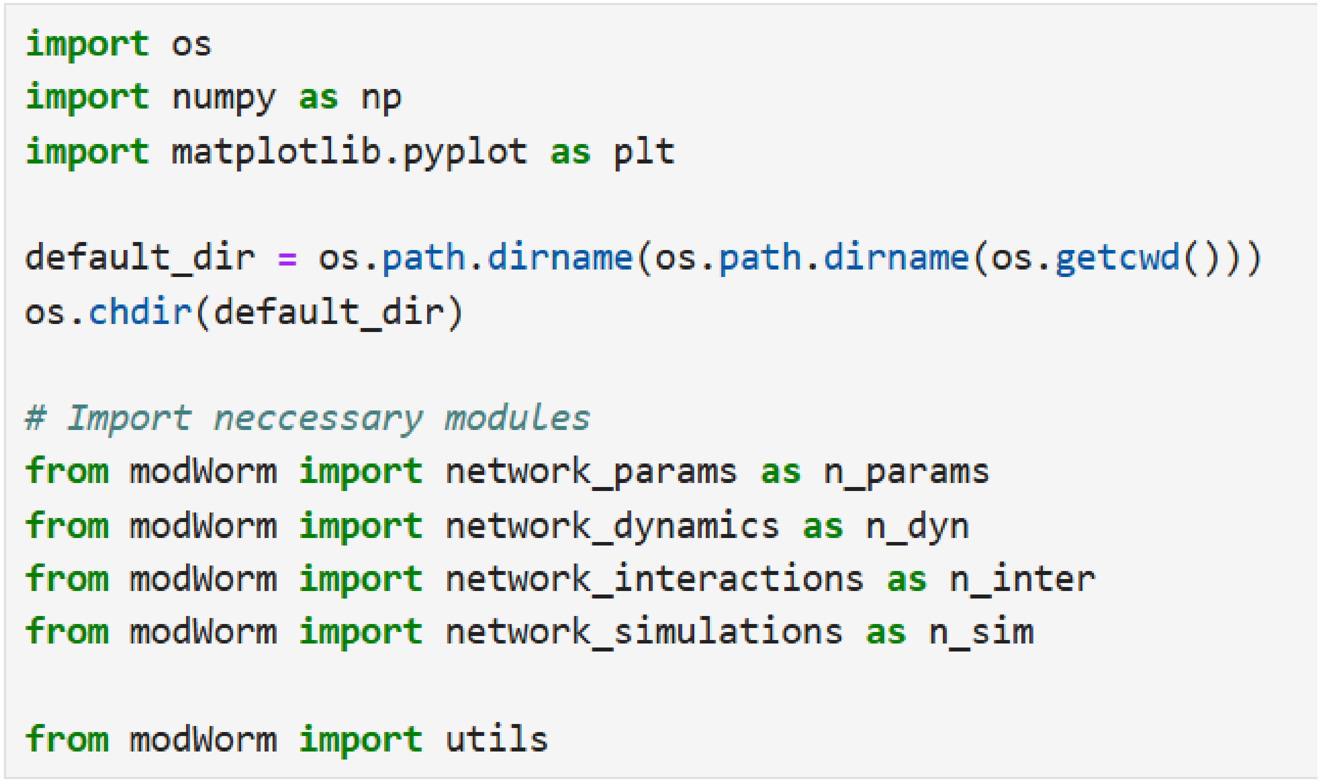


modWorm base nervous system model

- Complete somatic nervous system (279 neurons)
- Linear individual neuron dynamics (I^{Linear})
- Gap and synaptic neural interactions (I^{Gap}, I^{Syn})

More details can be found in Kim et al, 2025 (modWorm paper)

1. Import necessary modules



```
import os
import numpy as np
import matplotlib.pyplot as plt

default_dir = os.path.dirname(os.path.dirname(os.getcwd()))
os.chdir(default_dir)

# Import neccessary modules
from modWorm import network_params as n_params
from modWorm import network_dynamics as n_dyn
from modWorm import network_interactions as n_inter
from modWorm import network_simulations as n_sim

from modWorm import utils
```

Load NumPy and plotting tools

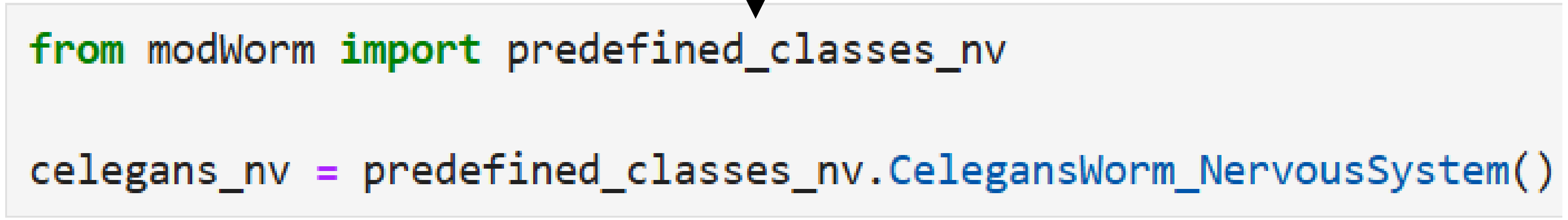
Set directory to library parent folder

Import library modules for nervous system simulations

Import utility module

2. Load pre-defined nervous system model

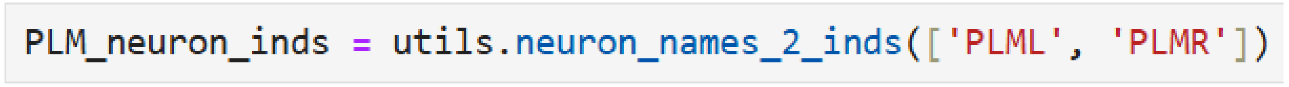
Predefined classes module for nervous system models



```python
from modWorm import predefined_classes_nv

celegans_nv = predefined_classes_nv.CelegansWorm_NervousSystem()
```

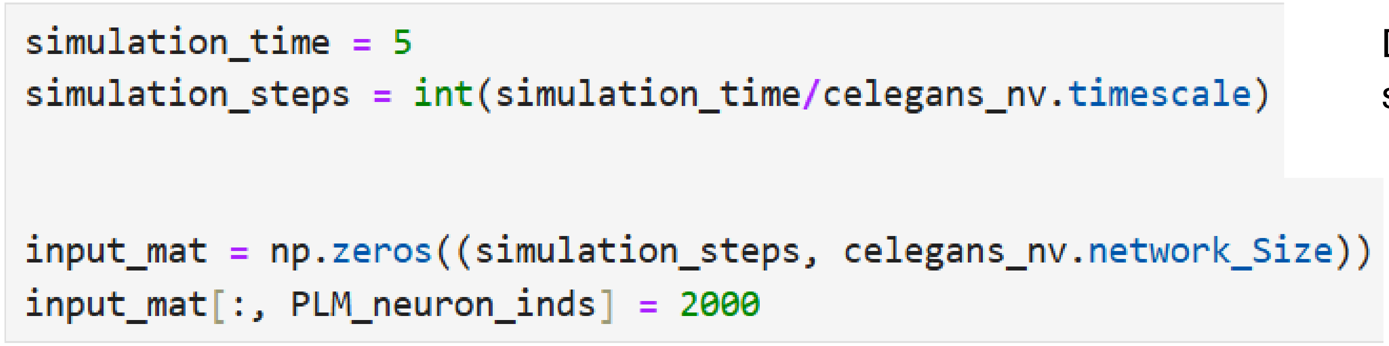
Load the base nervous system model class – **CelegansWorm_NervousSystem()**

3. Define input current stimuli



```python
PLM_neuron_inds = utils.neuron_names_2_inds(['PLML', 'PLMR'])
```

Convert neuron names to indices used by the model



```python
simulation_time = 5
simulation_steps = int(simulation_time/celegans_nv.timescale)
```

Define simulation duration (5s) and number of simulation steps (500 steps)

```python
input_mat = np.zeros((simulation_steps, celegans_nv.network_Size))
input_mat[:, PLM_neuron_inds] = 2000
```

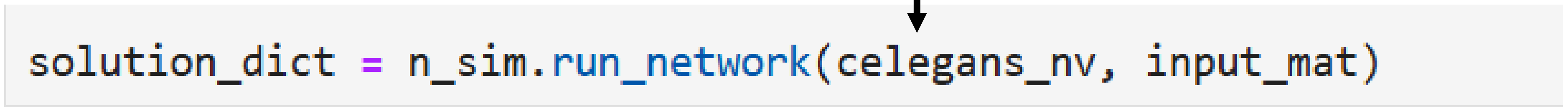
Create 2D array (timesteps, # of neurons) to define stimuli at each timestep

default recommended simulation timestep (timescale) = 0.01s

Model variable units can be found under /data/documents/Celegans_model_units.pdf

4. Simulate nervous system

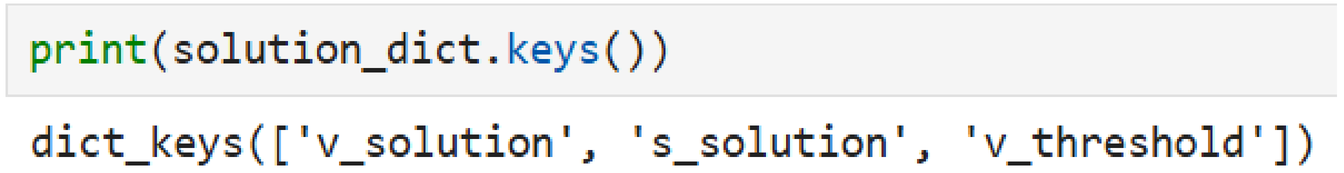
Nervous system model class



```
solution_dict = n_sim.run_network(celegans_nv, input_mat)
```

2D input stimuli array

5. Analyze simulation results



```
print(solution_dict.keys())
```

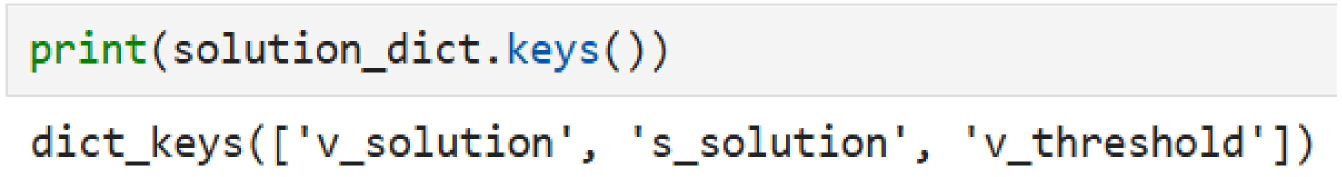
```
dict_keys(['v_solution', 's_solution', 'v_threshold'])
```

membrane potential dynamics
(timesteps * 279)

synaptic activity dynamics
(timesteps * 279)

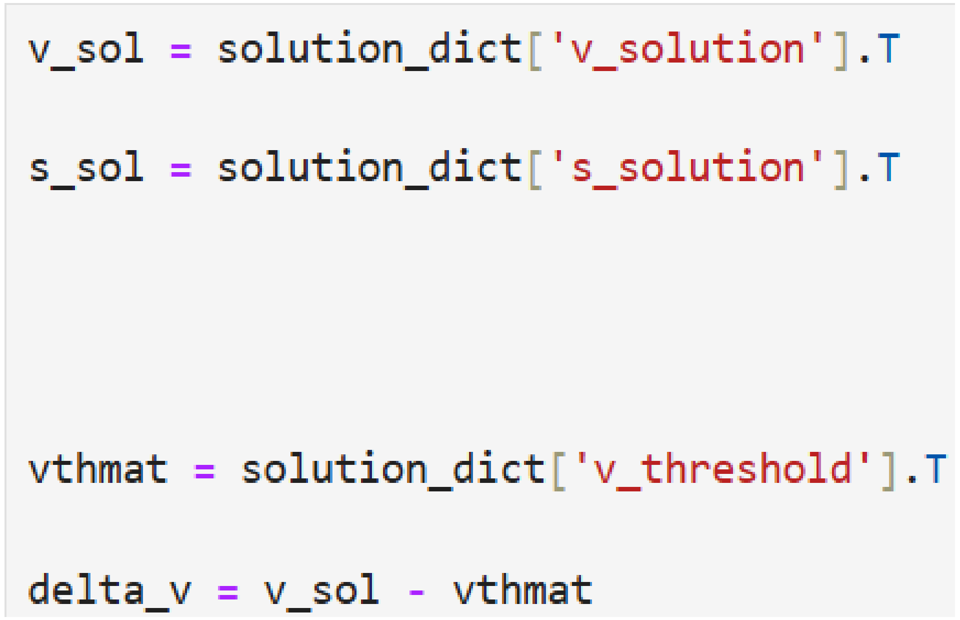
membrane potential threshold
(timesteps * 279)

5. Analyze simulation results



```
print(solution_dict.keys())
```

```
dict_keys(['v_solution', 's_solution', 'v_threshold'])
```



```
v_sol = solution_dict['v_solution'].T

s_sol = solution_dict['s_solution'].T




vthmat = solution_dict['v_threshold'].T

delta_v = v_sol - vthmat
```

Extract membrane potential dynamics

Extract synaptic activity variable

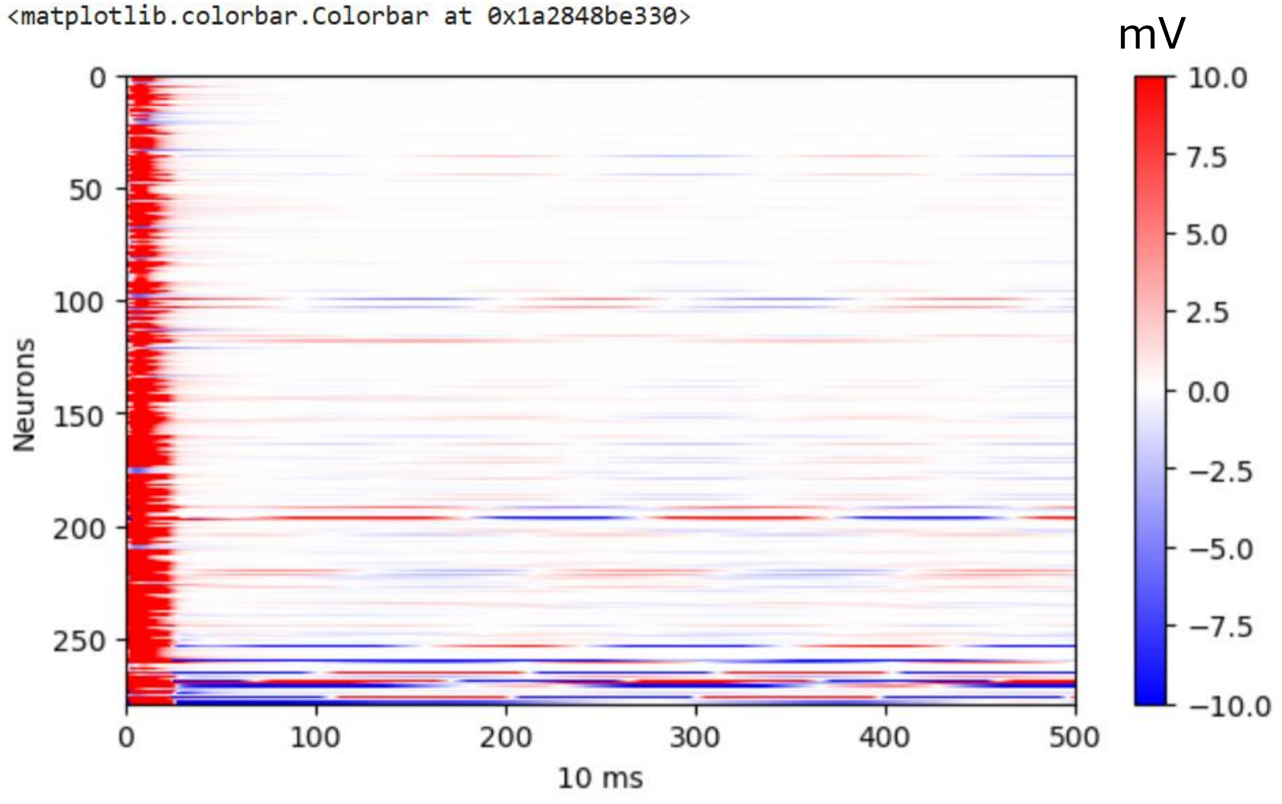
Extract membrane potential threshold (equilibrium) at each timestep

Compute membrane potential with respect to equilibrium

5. Analyze simulation results

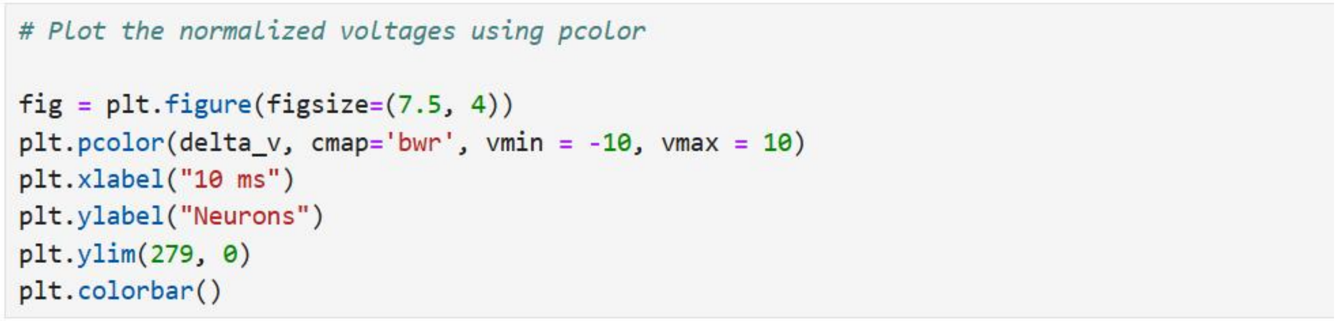
```
# Plot the normalized voltages using pcolor

fig = plt.figure(figsize=(7.5, 4))
plt.pcolor(delta_v, cmap='bwr', vmin = -10, vmax = 10)
plt.xlabel("10 ms")
plt.ylabel("Neurons")
plt.ylim(279, 0)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x1a2848be330>
```



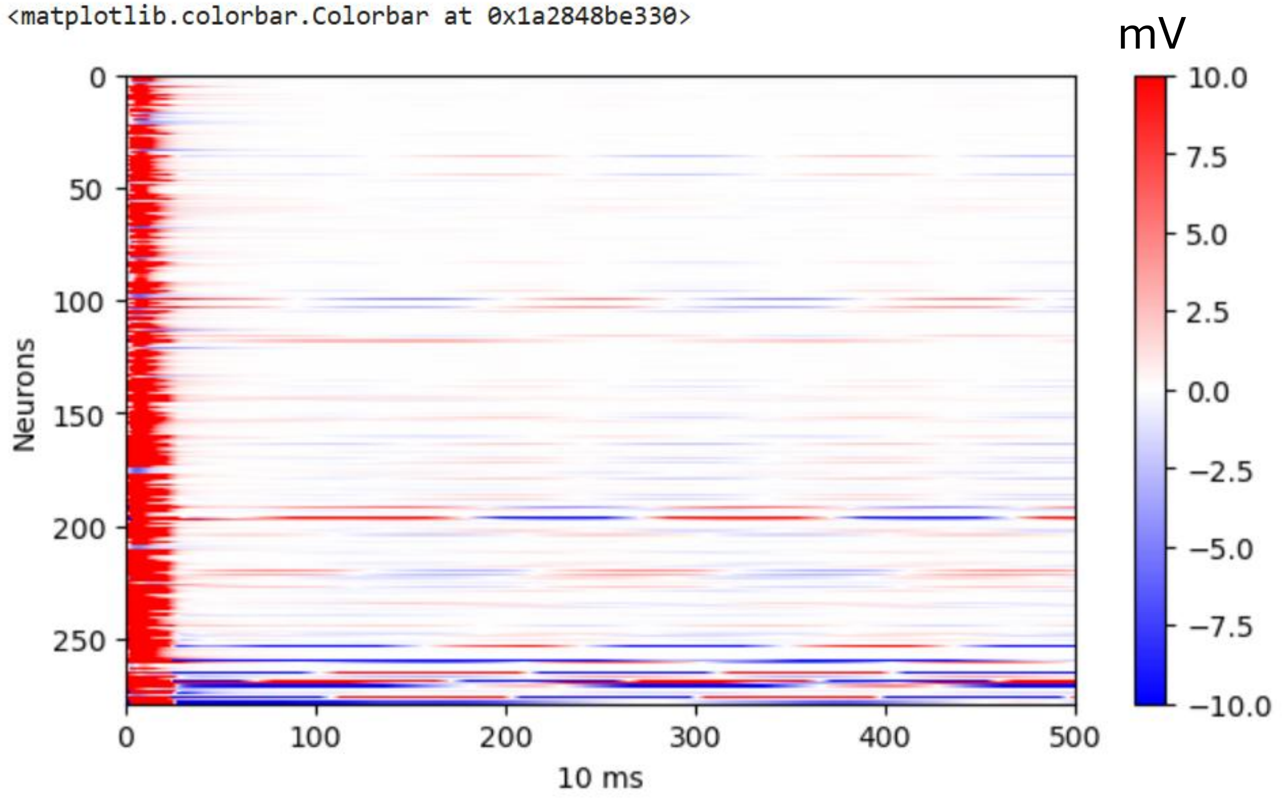
Plot neural dynamics using pcolor() function given by matplotlib library

5. Analyze simulation results



```
# Plot the normalized voltages using pcolor

fig = plt.figure(figsize=(7.5, 4))
plt.pcolor(delta_v, cmap='bwr', vmin = -10, vmax = 10)
plt.xlabel("10 ms")
plt.ylabel("Neurons")
plt.ylim(279, 0)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x1a2848be330>
```

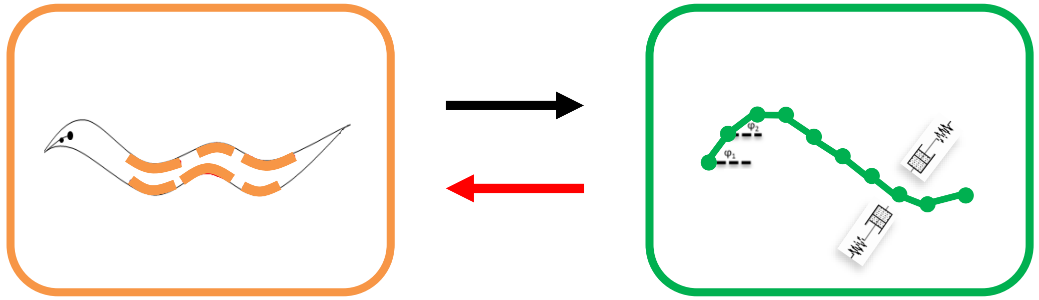


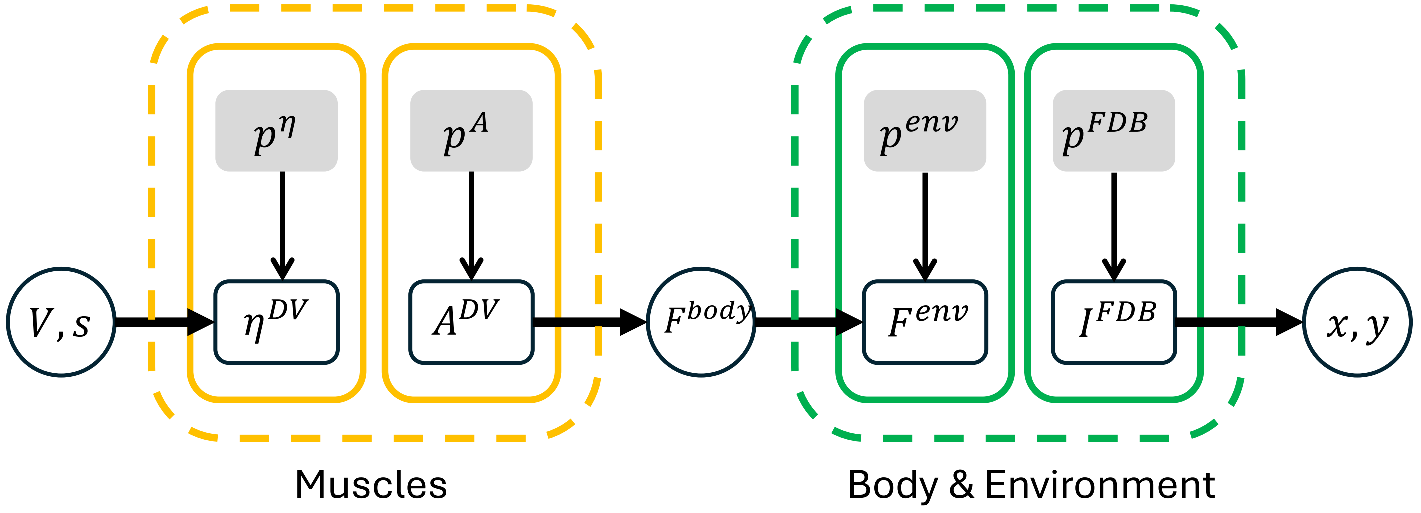
Plot neural dynamics using pcolor() function given by matplotlib library

List of all neurons and their indices can be found in **neurons.json** under /modWorm

Biomechanics Simulation

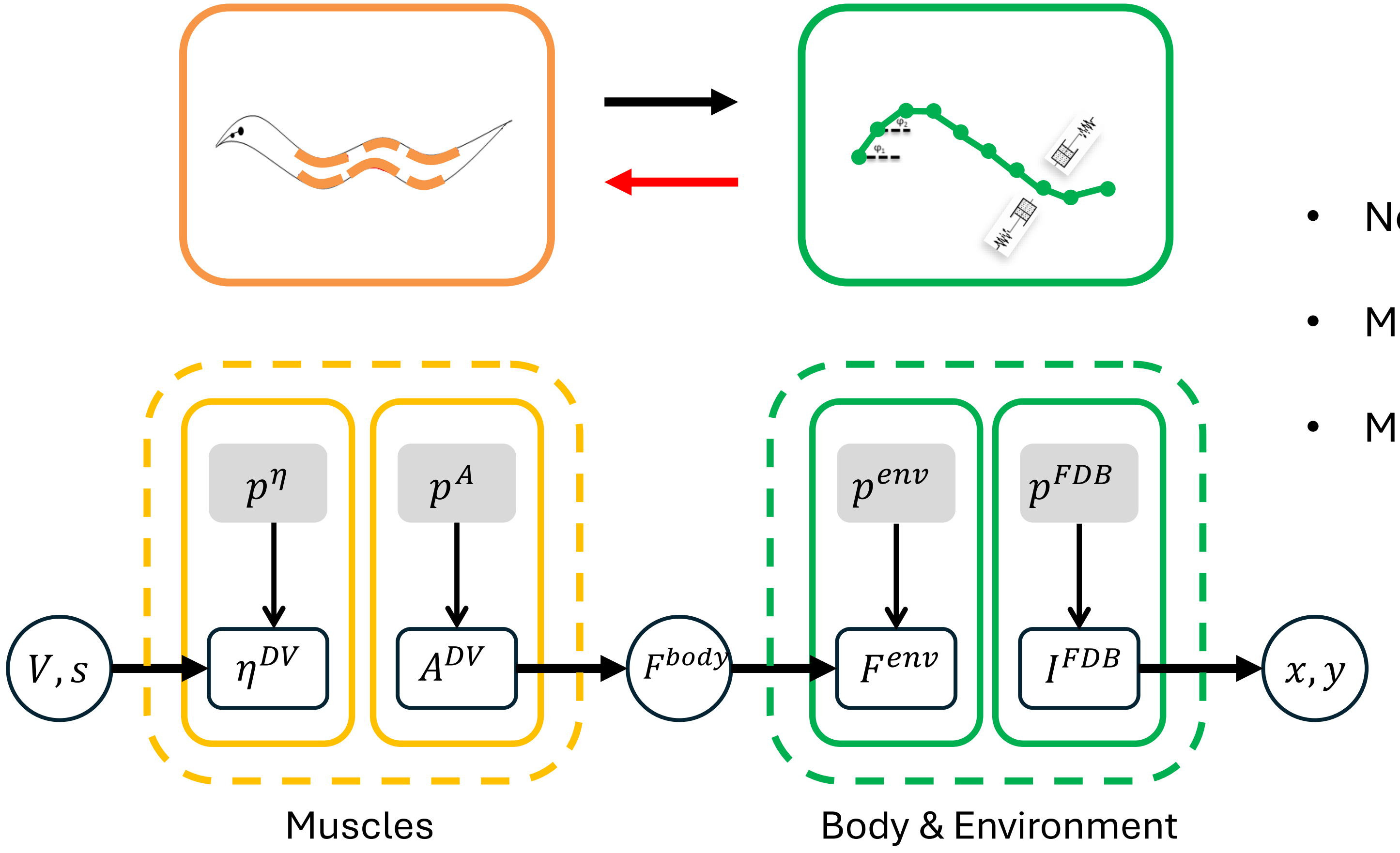
(Jupyter Notebook: 1.2 Quick start (Biomechanics).ipynb)





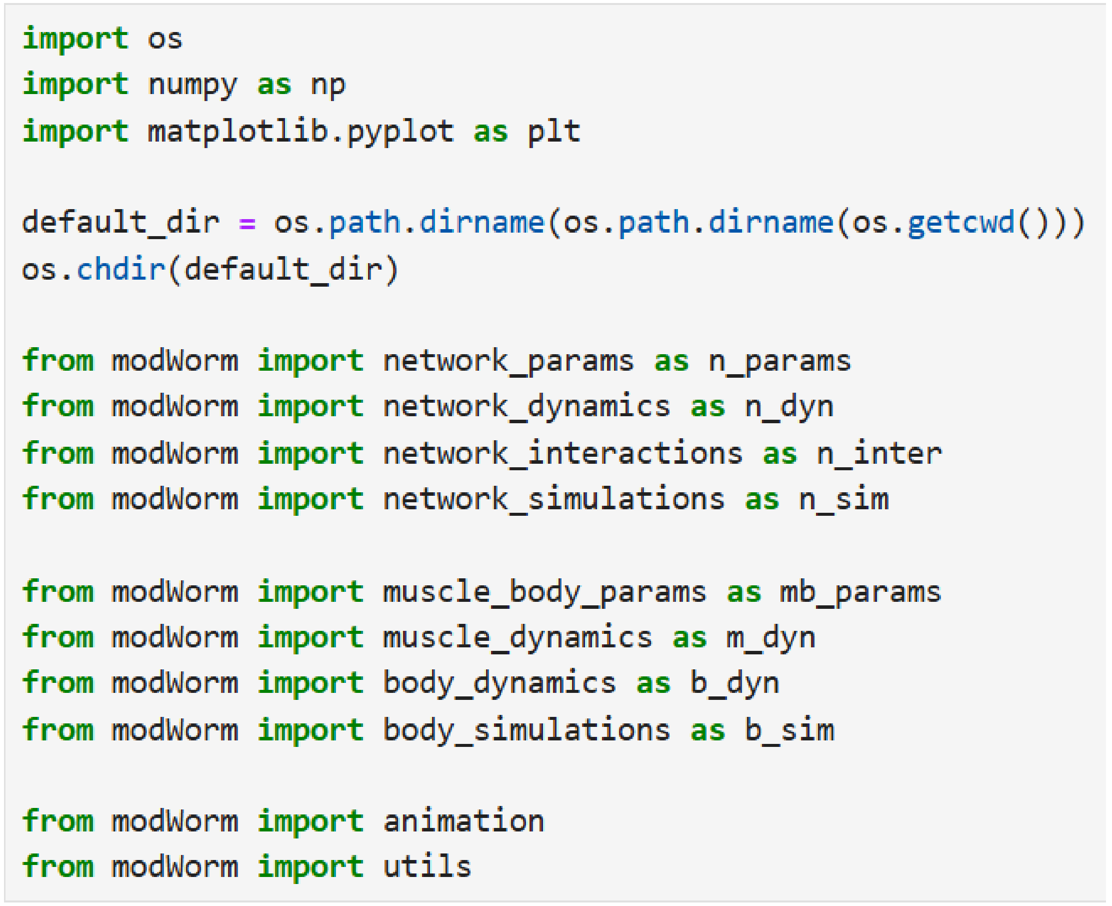
Biomechanics Simulation

(Jupyter Notebook: 1.2 Quick start (Biomechanics).ipynb)



- Neural dynamics to muscle calcium dynamics η^{DV}
- Muscle calcium dynamics to muscle activations A^{DV}
- Muscle activations to body postures (2D)

1. Import necessary modules



```python
import os
import numpy as np
import matplotlib.pyplot as plt

default_dir = os.path.dirname(os.path.dirname(os.getcwd()))
os.chdir(default_dir)

from modWorm import network_params as n_params
from modWorm import network_dynamics as n_dyn
from modWorm import network_interactions as n_inter
from modWorm import network_simulations as n_sim

from modWorm import muscle_body_params as mb_params
from modWorm import muscle_dynamics as m_dyn
from modWorm import body_dynamics as b_dyn
from modWorm import body_simulations as b_sim

from modWorm import animation
from modWorm import utils
```

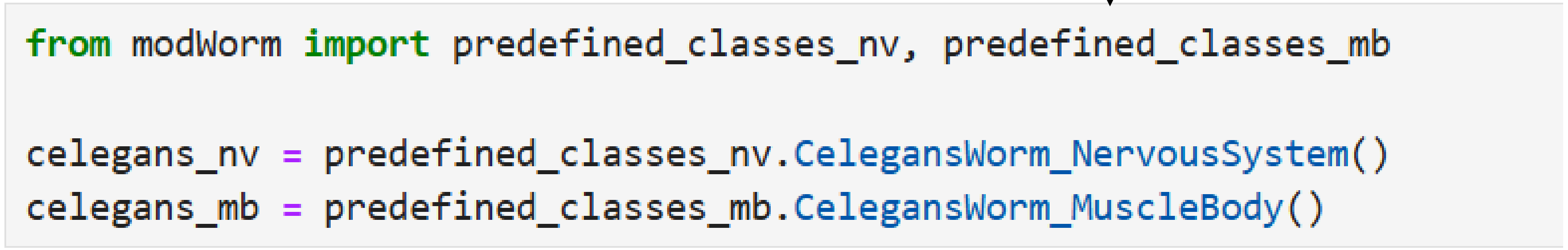
Nervous system modules

Import library modules for biomechanics simulations

Import animation module for body animation

2. Load pre-defined biomechanics model

Predefined classes module for biomechanics models

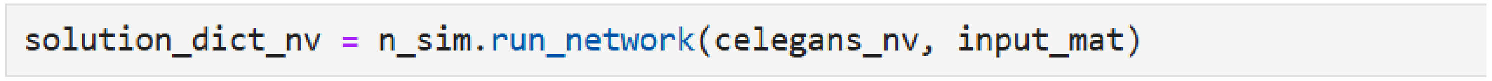


```python
from modWorm import predefined_classes_nv, predefined_classes_mb

celegans_nv = predefined_classes_nv.CelegansWorm_NervousSystem()
celegans_mb = predefined_classes_mb.CelegansWorm_MuscleBody()
```

Load the base biomechanics model – “CelegansWorm_MuscleBody()”

3. Simulate biomechanics

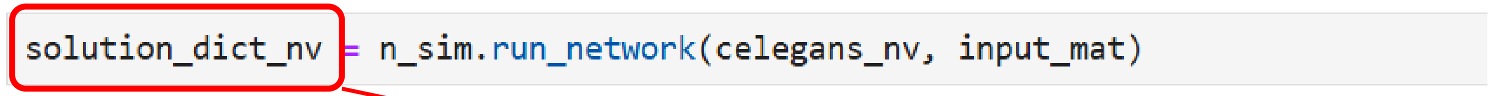
Simulate nervous system



```
solution_dict_nv = n_sim.run_network(celegans_nv, input_mat)
```

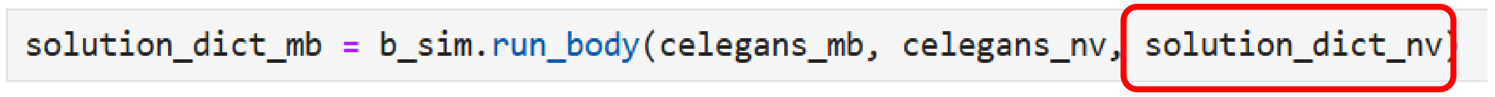
4. Simulate biomechanics

Simulate nervous system



```
solution_dict_nv = n_sim.run_network(celegans_nv, input_mat)
```

Simulate biomechanics



```
solution_dict_mb = b_sim.run_body(celegans_mb, celegans_nv, solution_dict_nv)
```

Biomechanics model class

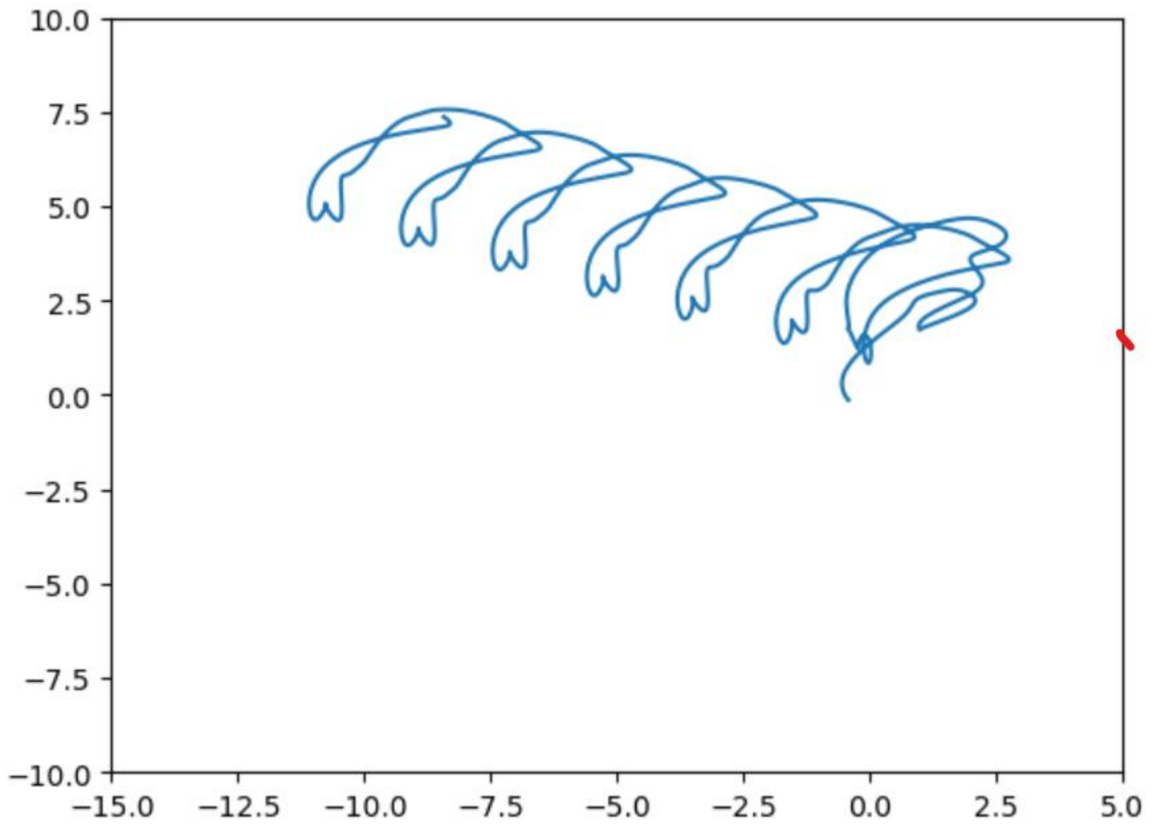
Nervous system model class

Nervous system solution dictionary

5. Analyze simulation results

```
plt.plot(solution_dict_mb['x_solution'][:, 0], solution_dict_mb['y_solution'][:, 0])
plt.ylim(-10, 10)
plt.xlim(-15, 5)
```

```
(-15.0, 5.0)
```

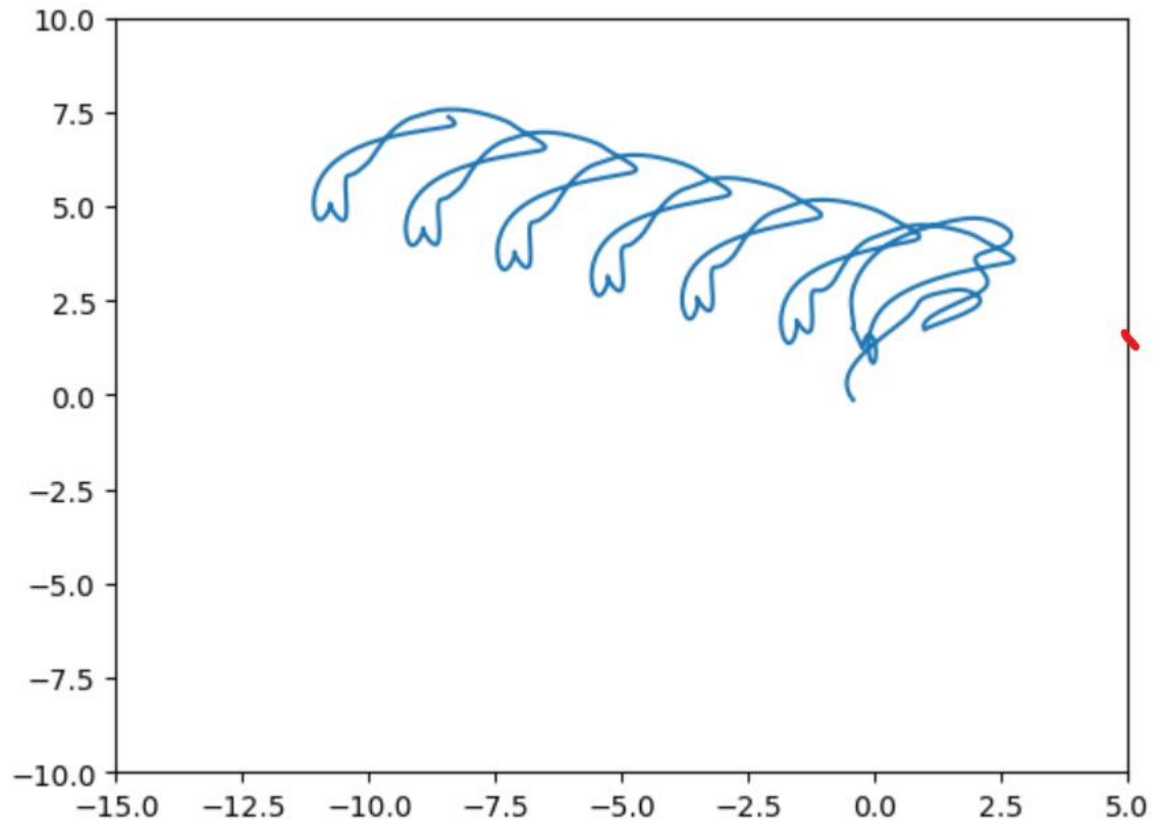


x_solution and **y_solution** contain x and y coordinates respectively for **192** body segments

5. Analyze simulation results

```
plt.plot(solution_dict_mb['x_solution'][:, 0], solution_dict_mb['y_solution'][:, 0])
plt.ylim(-10, 10)
plt.xlim(-15, 5)
```

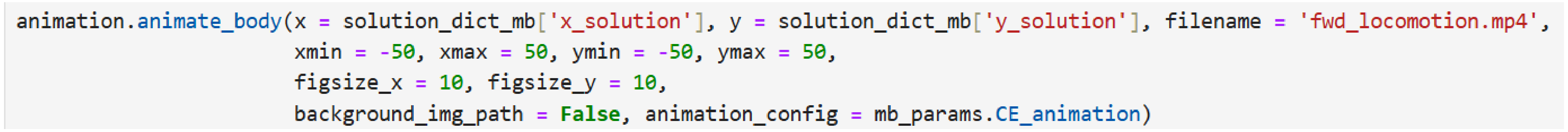
```
(-15.0, 5.0)
```



x_solution and **y_solution** contain x and y coordinates respectively for **192** body segments

x_solution[:, 0] and **y_solution[:, 0]** represent **head trajectory** throughout the simulation

5. Analyze simulation results



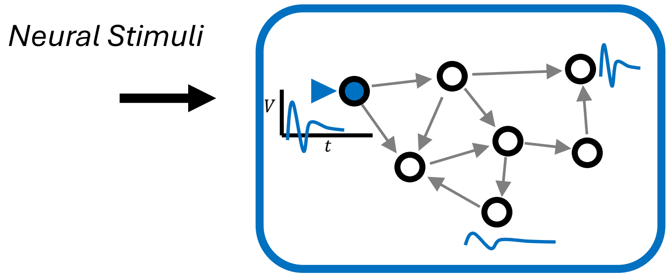
```
animation.animate_body(x = solution_dict_mb['x_solution'], y = solution_dict_mb['y_solution'], filename = 'fwd_locomotion.mp4',
                       xmin = -50, xmax = 50, ymin = -50, ymax = 50,
                       figsize_x = 10, figsize_y = 10,
                       background_img_path = False, animation_config = mb_params.CE_animation)
```

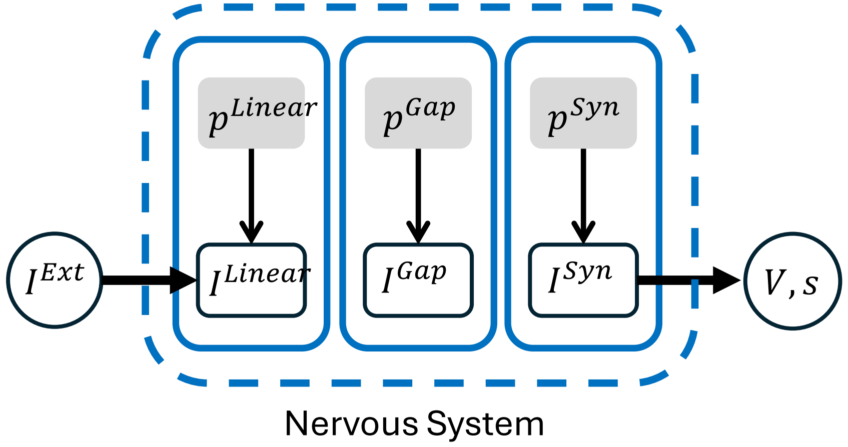
Use **animation.animate_body()** to create video for the body simulations.

Videos are saved under **/created_vids** located at the modWorm parent folder.

Linking Nervous System and Biomechanics

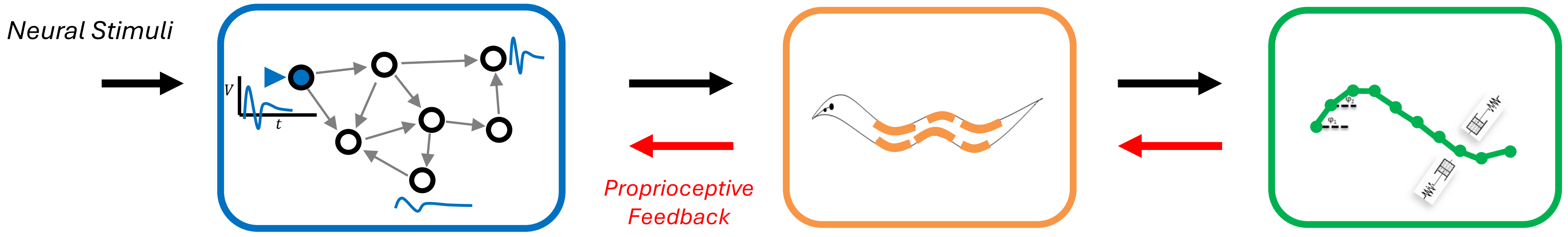
(Jupyter Notebook: 1.3 Quick start (Full model).ipynb)

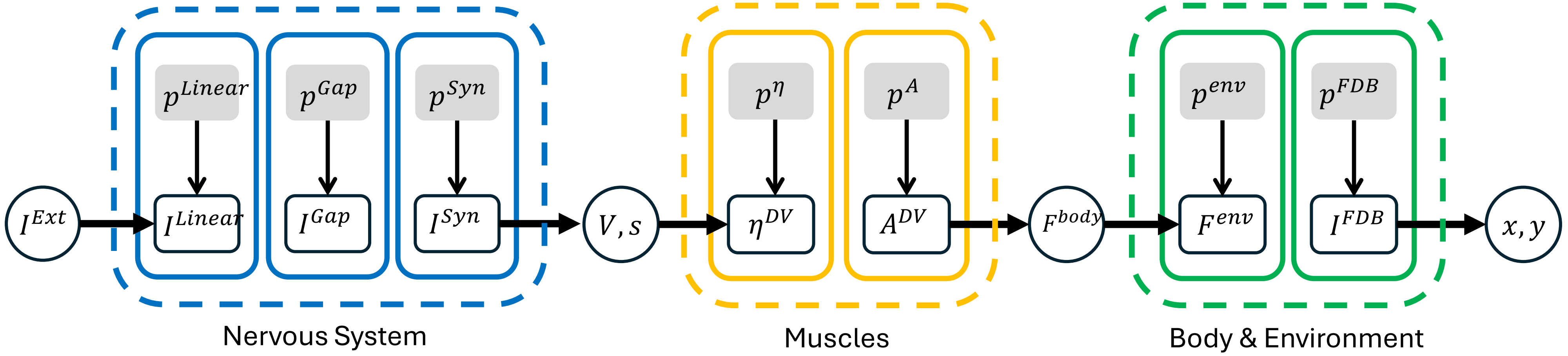




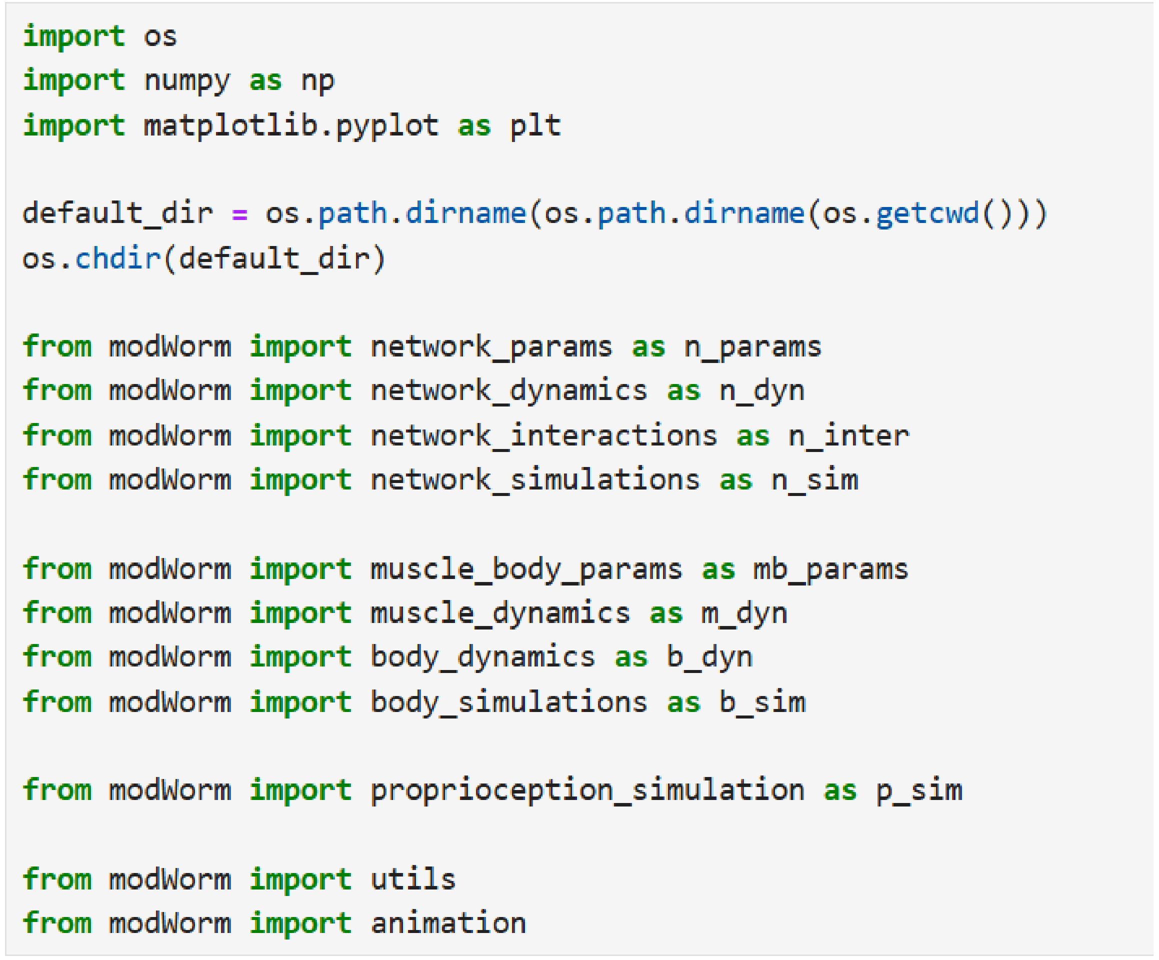
Linking Nervous System and Biomechanics

(Jupyter Notebook: 1.3 Quick start (Full model).ipynb)





1. Import necessary modules



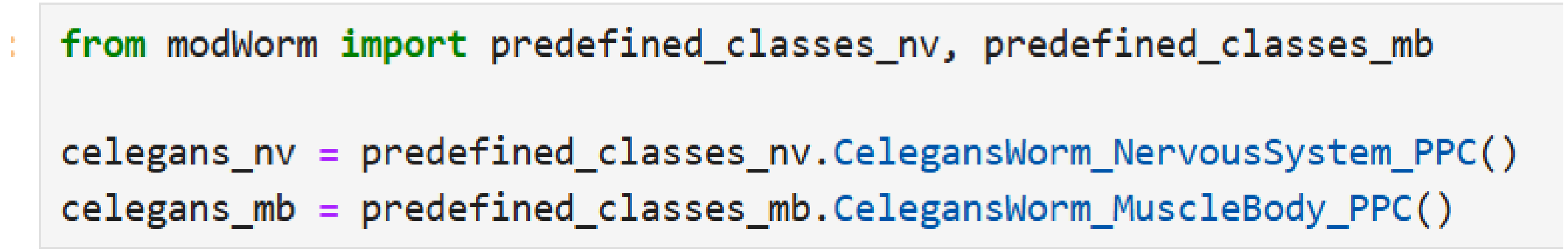
```python
import os
import numpy as np
import matplotlib.pyplot as plt

default_dir = os.path.dirname(os.path.dirname(os.getcwd()))
os.chdir(default_dir)

from modWorm import network_params as n_params
from modWorm import network_dynamics as n_dyn
from modWorm import network_interactions as n_inter
from modWorm import network_simulations as n_sim

from modWorm import muscle_body_params as mb_params
from modWorm import muscle_dynamics as m_dyn
from modWorm import body_dynamics as b_dyn
from modWorm import body_simulations as b_sim

from modWorm import proprioception_simulation as p_sim

from modWorm import utils
from modWorm import animation
```

Nervous system modules

Biomechanics modules

Import **proprioception_simulation** module for closed loop nervous system – body simulation

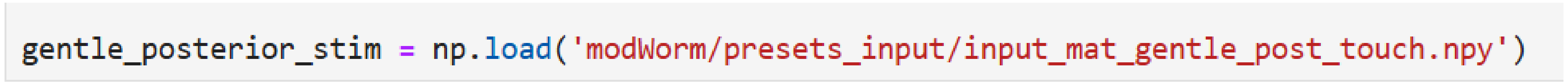
2. Load pre-defined nervous system model



```
from modWorm import predefined_classes_nv, predefined_classes_mb

celegans_nv = predefined_classes_nv.CelegansWorm_NervousSystem_PPC()
celegans_mb = predefined_classes_mb.CelegansWorm_MuscleBody_PPC()
```

Load nervous system and biomechanics model with **proprioception option** (_PPC)

3. Define input current stimuli



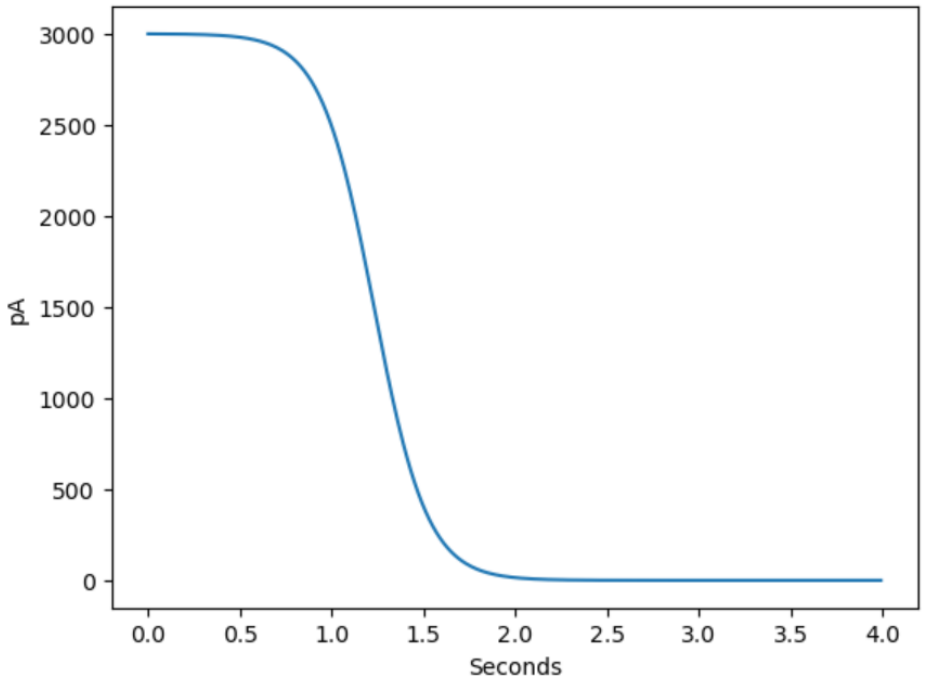
```python
gentle_posterior_stim = np.load('modWorm/presets_input/input_mat_gentle_post_touch.npy')
```

Load the pre-defined stimuli array for gentle posterior touch (PLM neurons)
(14 seconds, 1400 simulation timesteps)

3. Define input current stimuli

```
gentle_posterior_stim = np.load('neuralEngine\\presets_input\\input_mat_gentle_post_touch.npy')
```

Load the pre-defined stimuli array for gentle posterior touch (PLM neurons)
(14 seconds, 1400 simulation timesteps)



Input stimuli to PLM neurons decay over time

4. Simulate nervous system

Proprioception_simulations module

```
solution_dict_fwd = p_sim.run_network(celegans_nv, celegans_mb, gentle_posterior_stim)
```

Nervous system model class

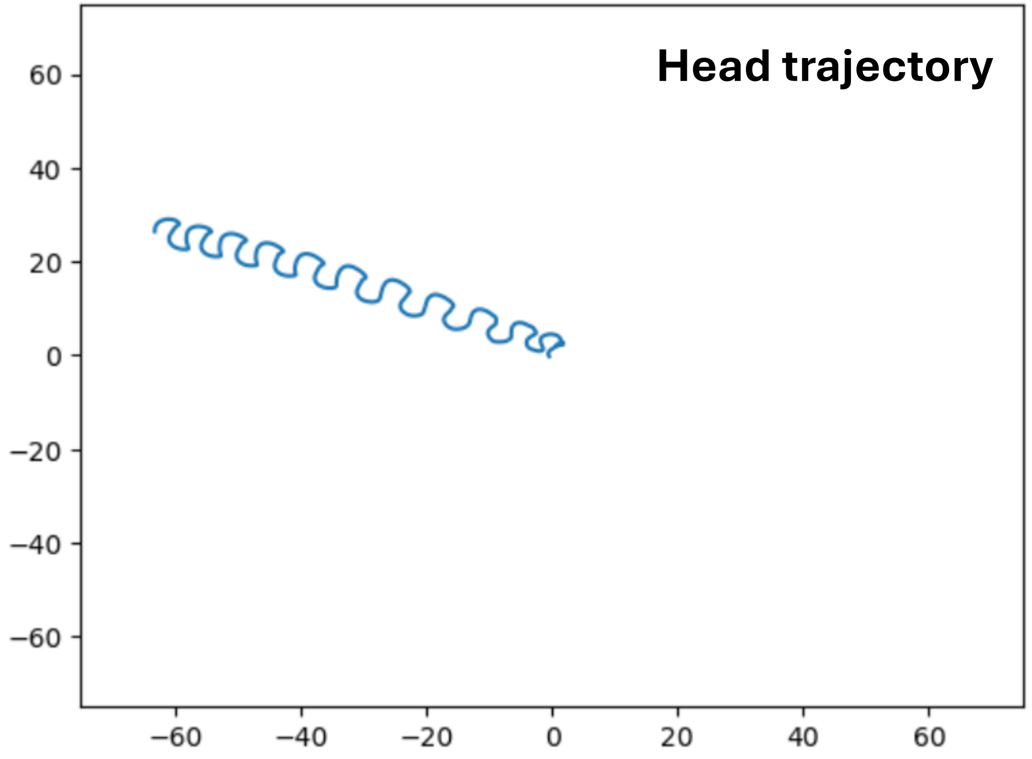
Biomechanics model class

Input stimuli array

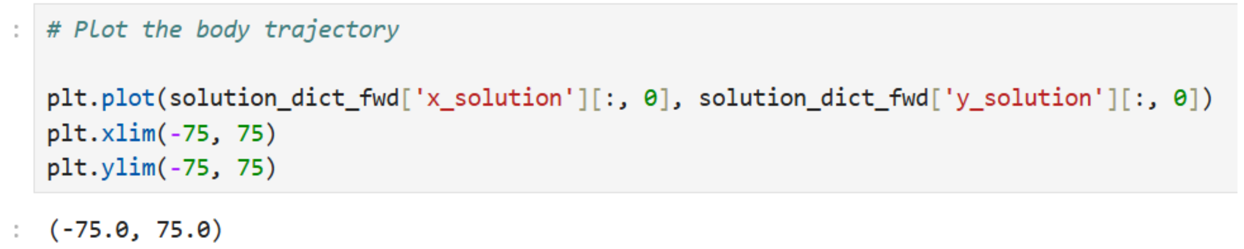
5. Analyze simulation results

```
# Plot the body trajectory

plt.plot(solution_dict_fwd['x_solution'][:, 0], solution_dict_fwd['y_solution'][:, 0])
plt.xlim(-75, 75)
plt.ylim(-75, 75)
```

```
(-75.0, 75.0)
```

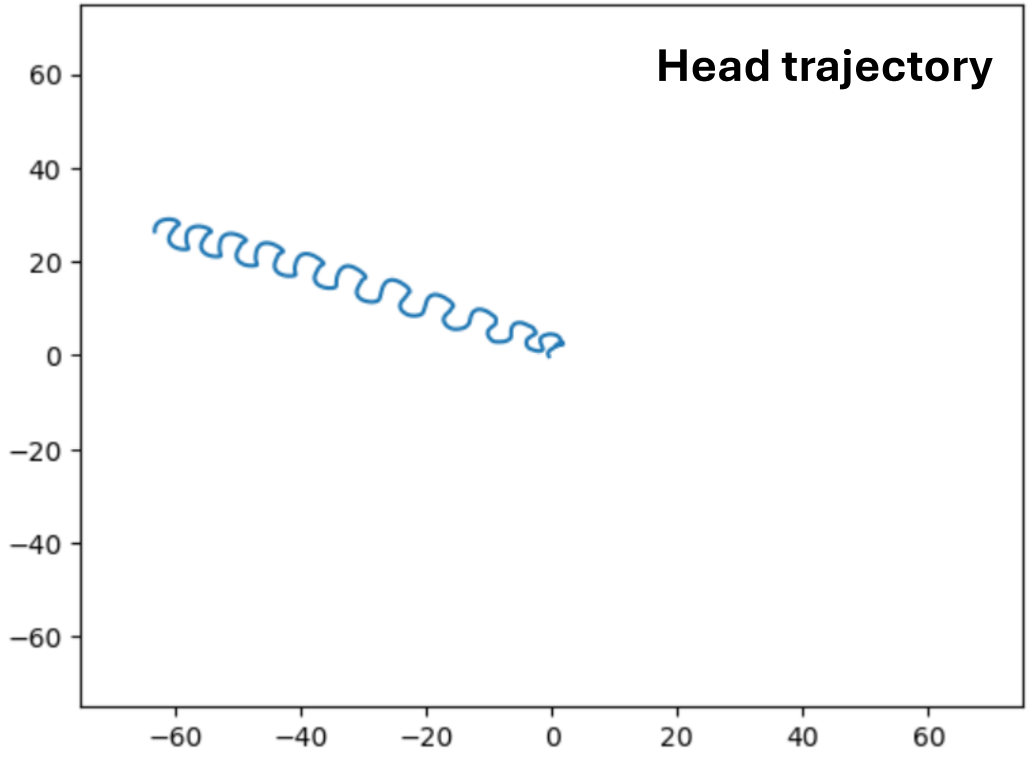


5. Analyze simulation results



```
# Plot the body trajectory

plt.plot(solution_dict_fwd['x_solution'][:, 0], solution_dict_fwd['y_solution'][:, 0])
plt.xlim(-75, 75)
plt.ylim(-75, 75)
```

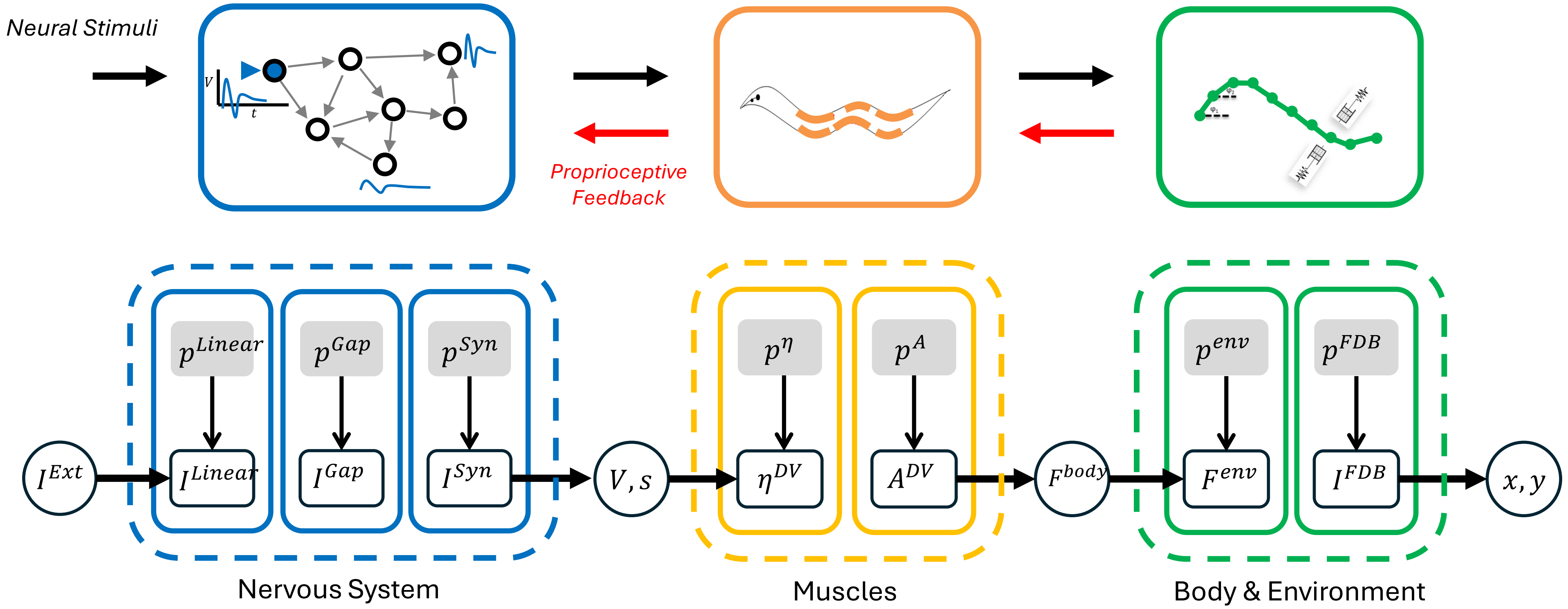
```
(-75.0, 75.0)
```



Body simulation animation

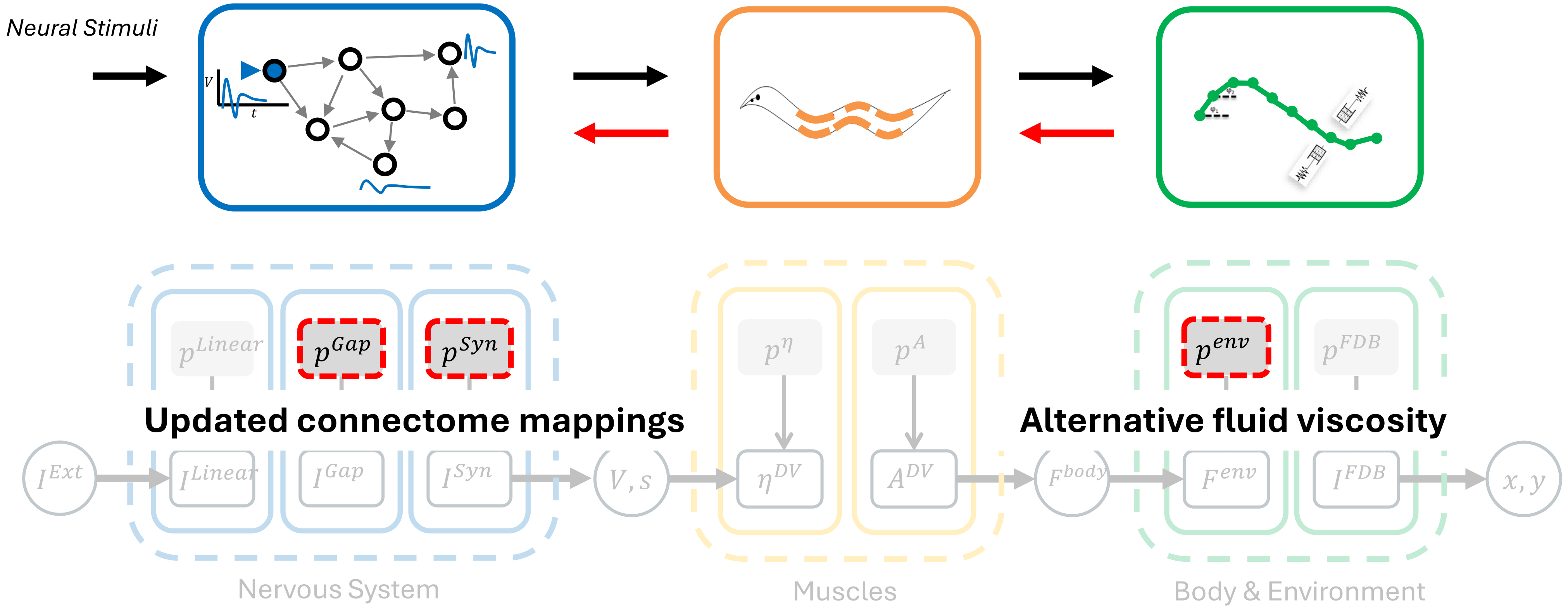
Customizing Model Components

(Jupyter Notebook: 1.4 Custom start (Full model).ipynb)



Customizing Model Components

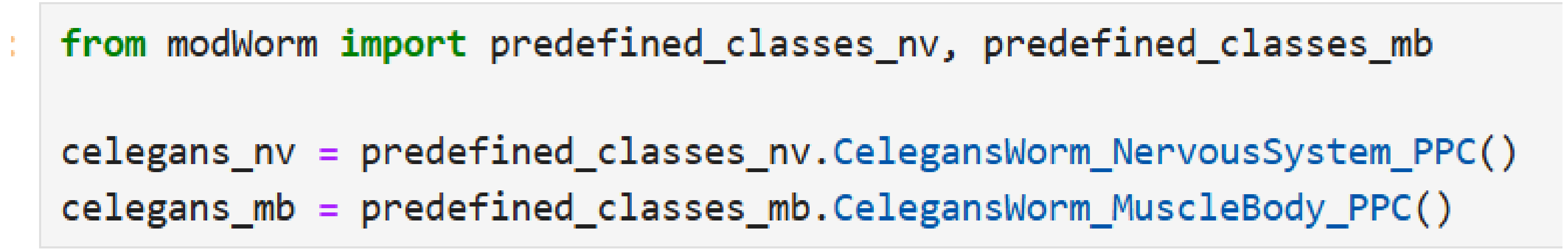
(Jupyter Notebook: 1.3 Custom start (Full model).ipynb)



Updated connectome mappings

Alternative fluid viscosity

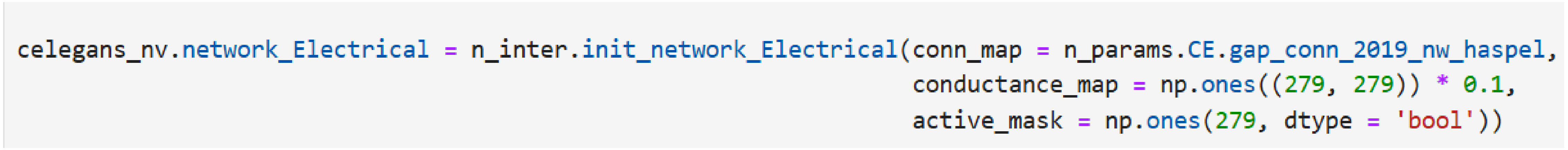
2. Load pre-defined nervous system model



```
from modWorm import predefined_classes_nv, predefined_classes_mb

celegans_nv = predefined_classes_nv.CelegansWorm_NervousSystem_PPC()
celegans_mb = predefined_classes_mb.CelegansWorm_MuscleBody_PPC()
```

Load nervous system and biomechanics model with proprioception option (PPC)

3. Modify model components

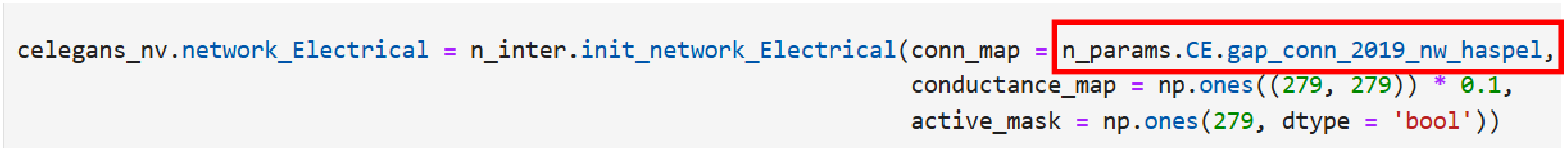


```python
celegans_nv.network_Electrical = n_inter.init_network_Electrical(conn_map = n_params.CE.gap_conn_2019_nw_haspel,
                                                                 conductance_map = np.ones((279, 279)) * 0.1,
                                                                 active_mask = np.ones(279, dtype = 'bool'))
```

- **conn_map**: connectivity mapping adjacency matrix (weight = # of synapses or other metrics)
- **conductance_map**: conductance multiplier applied to conn_map (default = 0.1nS)
- **active_mask**: 1D Boolean array (True, False) for turning ON/OFF neurons

3. Modify model components

Varshney et al, 2011 → Cook et al, 2019

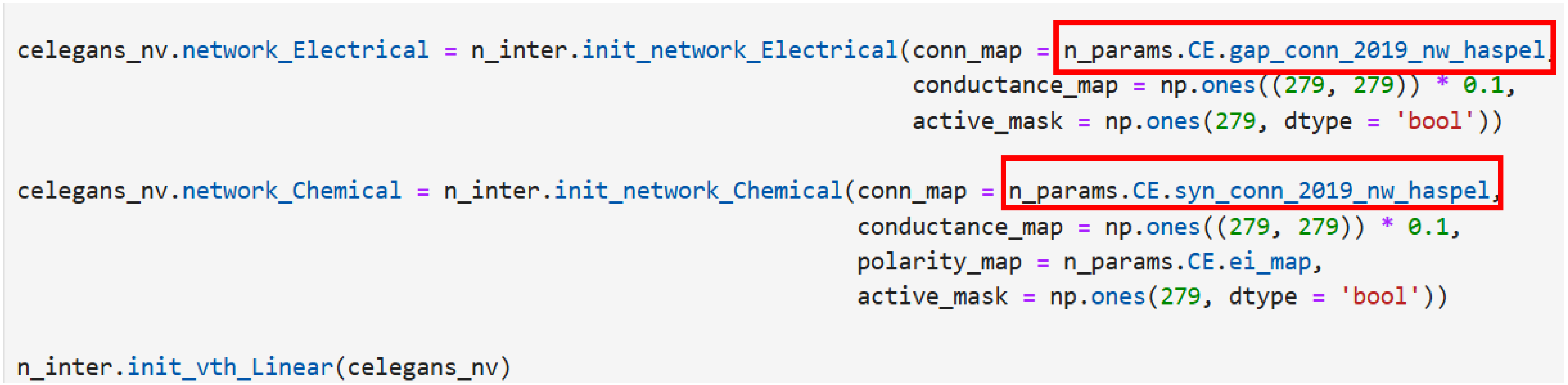


```
celegans_nv.network_Electrical = n_inter.init_network_Electrical(conn_map = n_params.CE.gap_conn_2019_nw_haspel,
                                                                 conductance_map = np.ones((279, 279)) * 0.1,
                                                                 active_mask = np.ones(279, dtype = 'bool'))
```

- **conn_map**: connectivity mapping adjacency matrix (weight = # of synapses or other metrics)
- **conductance_map**: conductance multiplier applied to conn_map (default = 0.1nS)
- **active_mask**: 1D Boolean array (True, False) for turning ON/OFF neurons

3. Modify model components

Varshney et al, 2011 → Cook et al, 2019

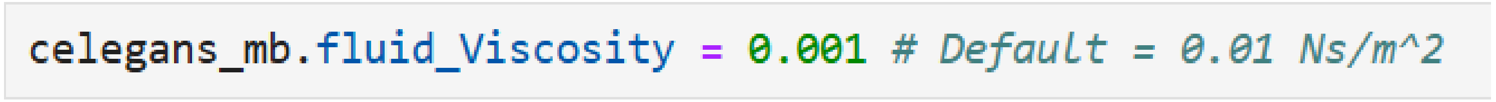


```
celegans_nv.network_Electrical = n_inter.init_network_Electrical(conn_map = n_params.CE.gap_conn_2019_nw_haspel,
                                                                 conductance_map = np.ones((279, 279)) * 0.1,
                                                                 active_mask = np.ones(279, dtype = 'bool'))

celegans_nv.network_Chemical = n_inter.init_network_Chemical(conn_map = n_params.CE.syn_conn_2019_nw_haspel,
                                                             conductance_map = np.ones((279, 279)) * 0.1,
                                                             polarity_map = n_params.CE.ei_map,
                                                             active_mask = np.ones(279, dtype = 'bool'))


n_inter.init_vth_Linear(celegans_nv)
```

- **polarity_map** : Excitatory/Inhibitory mapping for synapses (279 * 279)

Run **neural_interactions.init_vth_Linear()** to finalize the nervous system model reconfiguration

3. Modify model components



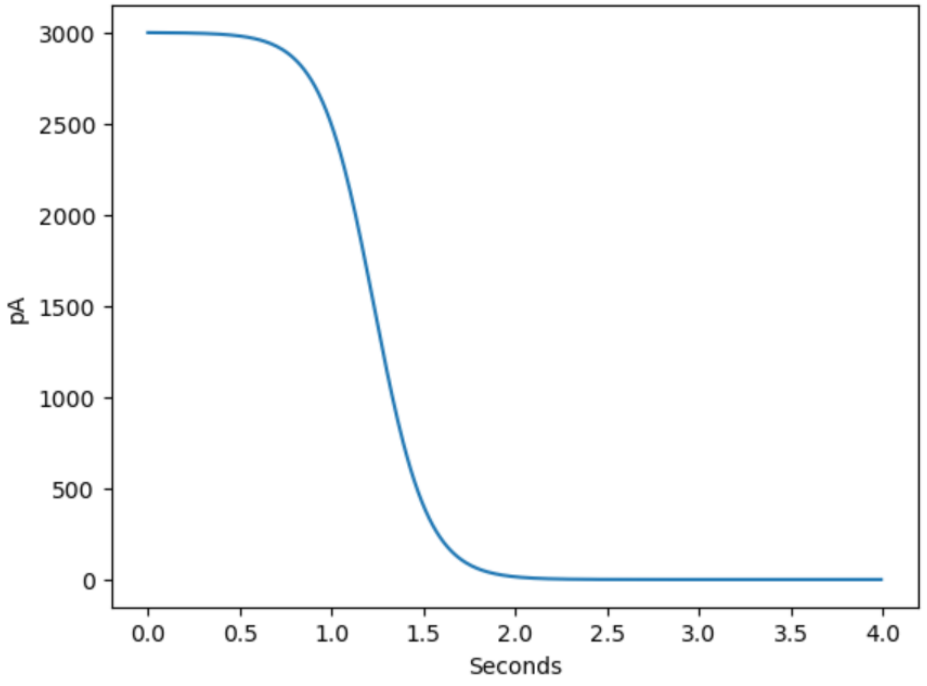
```
celegans_mb.fluid_Viscosity = 0.001 # Default = 0.01 Ns/m^2
```

Change **fluid viscosity** from 0.01 Ns/m^2 to 0.001 Ns/m^2

3. Define input current stimuli

```
gentle_posterior_stim = np.load('modWorm/presets_input/input_mat_gentle_post_touch.npy')
```

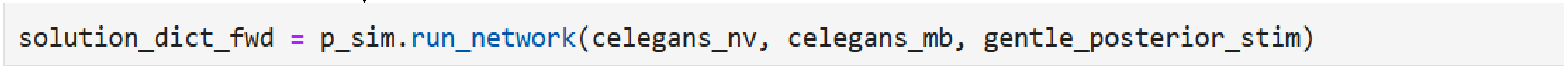
Load the pre-defined stimuli array for gentle posterior touch (PLM neurons)



Input stimuli to PLM neurons decay over time

4. Simulate nervous system

Proprioception_simulations module



```
solution_dict_fwd = p_sim.run_network(celegans_nv, celegans_mb, gentle_posterior_stim)
```

Modified nervous system model class

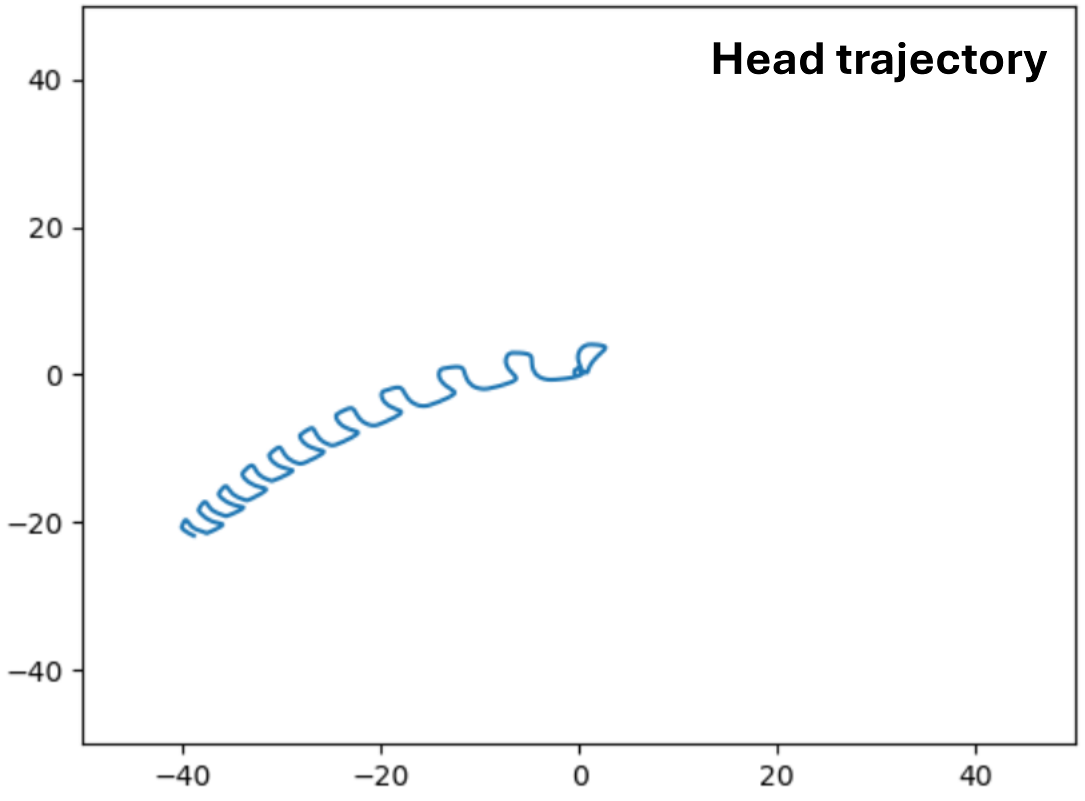
Modified Biomechanics model class

Input stimuli array

5. Analyze simulation results

```
# Plot the body trajectory

plt.plot(solution_dict_fwd['x_solution'][:, 0], solution_dict_fwd['y_solution'][:, 0])
plt.xlim(-50, 50)
plt.ylim(-50, 50)
```

```
(-50.0, 50.0)
```



Body simulation animation