11- Advanced BMTK

# Instructions

This document assumes you have completed the necessary steps in **02-Single\_Cell\_Hoc\_BMTK** or **03-Networked\_Hoc\_BMTK**

# Custom Cell Positions

Position lists and random positions. (from bmtk.builder.auxi.node\_params import positions\_list)

# Custom Synapses

Included with the mod files from our HCO network from 03-Networked\_Hoc\_BMTK is a custom inhsyn.mod file. We can direct BMTK to use this file by taking the following steps

1. Copy the entire my\_bmtk\_model folder to a new folder called my\_bmtk\_model\_syn.
2. In this new folder, create a new file named synapses.py and add the following code to this file:

|  |  |
| --- | --- |
|  | **synapses.py** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  34  35 | from bmtk.simulator.bionet.pyfunction\_cache import add\_synapse\_model  from neuron import h  def InhSyn(syn\_params, sec\_x, sec\_id):  """Create a inhsyn synapse  :param syn\_params: parameters of a synapse  :param sec\_x: normalized distance along the section  :param sec\_id: target section  :return: NEURON synapse object  """  lsyn = h.inhsyn(sec\_x, sec=sec\_id)  if syn\_params.get('esyn'):  lsyn.esyn = float(syn\_params['esyn'])  if syn\_params.get('gmax'):  lsyn.gmax = float(syn\_params['gmax'])    return lsyn  def inhsyn(syn\_params, xs, secs):  """Create a list of inhsyn synapses  :param syn\_params: parameters of a synapse  :param xs: list of normalized distances along the section  :param secs: target sections  :return: list of NEURON synpase objects  """  syns = []  for x, sec in zip(xs, secs):  syn = InhSyn(syn\_params, x, sec)  syns.append(syn)  return syns  def load():  add\_synapse\_model(InhSyn, 'inhsyn', overwrite=False)  add\_synapse\_model(InhSyn, overwrite=False)  return |

1. Things to note:
   1. (To use this in your model simply change everywhere InhSyn and inhsyn is defined to your synapse name, with variable name changes to line 14+)
   2. Line 1: add\_synapse\_model function is called to add custom synapses to BMTK’s python function cache, allowing BMTK to “see” and use your synapse file
   3. Line 4: syn\_params will be a dictionary containing parameters defined in the json file referenced when creating edges (shown later)
   4. Line 12: h.inhsyn will instantiate the inhsyn neuron hobject
   5. Line 14-17: Set properties of the synapse by:
      1. Checking to see if the parameter has been defined
      2. Setting the synapse value to the supplied syn\_params value
   6. Line 19: Create an additional function for BMTK to handle lists of synapses, we simply link it to our previous InhSyn function to prevent code duplication
   7. Line 34: Call load() in your build\_network.py and run\_bionet.py scripts early on to notify BMTK that you have custom synapses. (also shown later)
2. Create a new file called my\_inhsyn.json in ./biophys\_components/synaptic\_models/ and place the following into it (Note how esyn and gmax appear in this file and the synapse function we defined previously)

|  |  |
| --- | --- |
|  | **my\_inhsyn.json** |
| 1  2  3  4  5 | {  "esyn":"-80",  "gmax":"40e-3"  } |

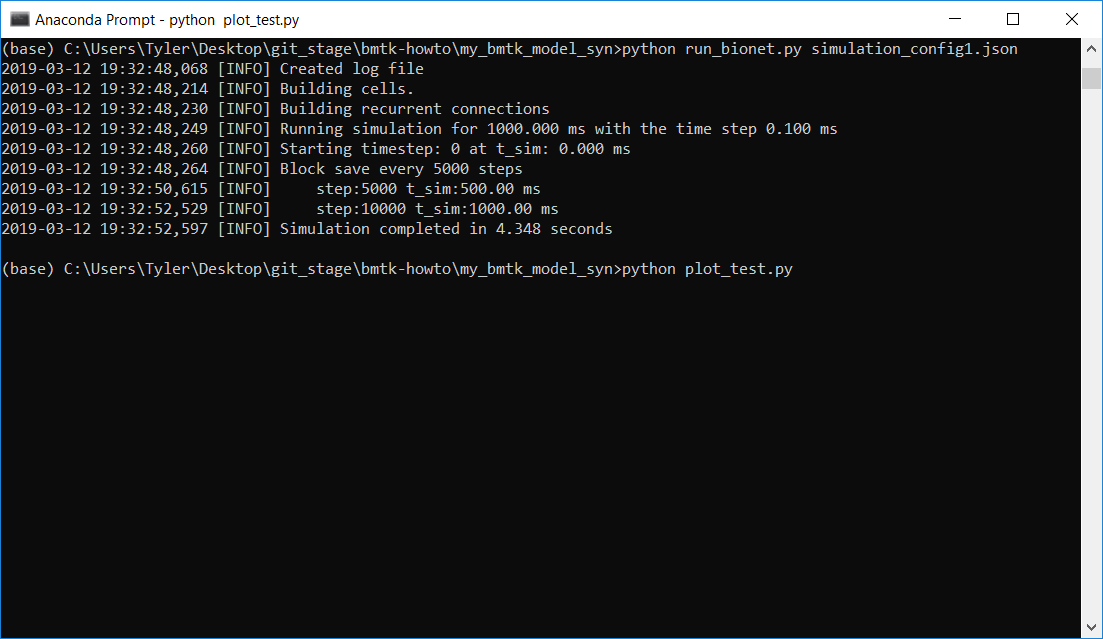
1. In build\_network1.py, we can now import this synapse file, call the load function, and use the synapse when defining our edges. See the complete file below with explanations.

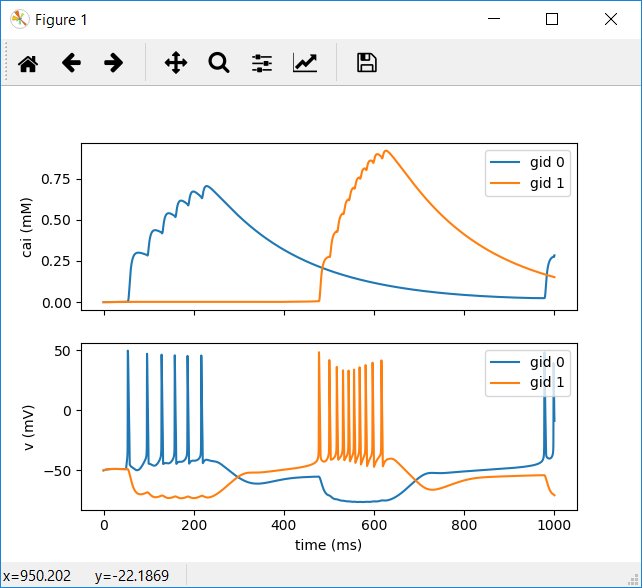
|  |  |
| --- | --- |
|  | **build\_network1.py** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | from bmtk.builder.networks import NetworkBuilder  import synapses  synapses.load()  net1 = NetworkBuilder('hco\_net')  net1.add\_nodes(N=1,  cell\_name='HCOCell1',  model\_type='biophysical',  model\_template='hoc:HCOcell',  morphology='blank.swc'  )    net1.add\_nodes(N=1,  cell\_name='HCOCell2',  model\_type='biophysical',  model\_template='hoc:HCOcell',  morphology='blank.swc'  )      net1.add\_edges(source={'cell\_name': 'HCOCell1'}, target={'cell\_name':'HCOCell2'},  connection\_rule=1,  syn\_weight=40.0e-02,  dynamics\_params='my\_inhsyn.json',  model\_template='inhsyn',  delay=0.0,  target\_sections=["soma"],  distance\_range=[0,999])    net1.add\_edges(source={'cell\_name': 'HCOCell2'}, target={'cell\_name':'HCOCell1'},  connection\_rule=1,  syn\_weight=40.0e-02,  dynamics\_params='my\_inhsyn.json',  model\_template='inhsyn',  delay=0.0,  target\_sections=["soma"],  distance\_range=[0,999])  net1.build()  net1.save\_nodes(output\_dir='network')    net1.build()  net1.save\_edges(output\_dir='network') |

1. Things to note:
   1. Lines 2 and 4: import the synapses file we just created and load the synapses by calling the load function.
   2. Lines 27,28, 36,37: reference the synapse name and dynamics\_params file json created earlier
2. Run `python build\_network1.py` to build the network.
3. Add the synapse load function to the top of run\_bionet.py like the following:

|  |  |
| --- | --- |
|  | **run\_bionet.py (snippet)** |
| 1  2  3  4  5  6  7  8  9 | import os, sys  from bmtk.simulator import bionet  from bmtk.simulator.bionet.default\_setters.cell\_models import loadHOC  import synapses  synapses.load()  bionet.pyfunction\_cache.add\_cell\_model(loadHOC, directive='hoc', model\_type='biophysical') |

1. You are now ready to run your network, run `python run\_bionet.py simulation\_config1.json` then `python plot\_test.py`. You should notice small changes in the dynamics of the synapse output from previous tests.





# Dynamic Synapse Properties

Edge properties like delay can be changed dynamically, per connection, rather than a blanket set value.

# Recurrent Synapses

Connect neurons back to neurons they’re connected to. (See <https://gist.github.com/tjbanks/8228e341e33f65d641bccc6f187e0895> until instructions can be made)

# Rule Based Synapses

If we want to define connections very specifically we can make use of the “all\_to\_one” iterator when adding edges.