2 - Single Cell Hoc in BMTK

# Instructions

This document assumes you have completed the necessary steps in 01-Installing\_BMTK

## Setup

1. Now that BTMK is installed, visit the directory you want to **build** your model in and run BMTK’s directory setup. The network directory is where the node/edge configuration files will sit. For example, the following will change directory to your desktop, create a project directory (bmtk\_model), create a network directory for btmk to use, and initialize the directory for your further customization. **This step will only be run once!**

cd C:\Users\<your\_username>\Desktop

mkdir my\_bmtk\_model

cd my\_bmtk\_model

python -m bmtk.utils.sim\_setup bionet .

This will create the bmtk directory structure at the present location “$BASE.dir”, and create the following files and nested directories:

Nested directories: biophys\_coponents, network, output

Json files: circuit\_config.json, simulation\_config.json

1. In Windows you may be met with compilation errors like the following:

C:\Users\Tyler\Desktop\my\_bmtk\_model\biophys\_components\mechanisms

Was unable to compile mechanism in $COMPONENTS\_DIR/mechanisms

This is normal. You will need to compile your mod files any time they change. Run mknrndll in the (\components\mechanisms\modfiles\) directory then copy the resulting dll file to the parent directory (\components\mechanisms\)

In Linux, copy all mod files into the (components\mechanisms) directory and run nrnivmodl

## HOC Build

1. HOC template files can be used by BMTK. For this example, we’ll take a single HCO cell and run it with no input, in BMTK.
2. Download the HCO files from <https://github.com/tjbanks/two-cell-hco/archive/master.zip>
3. Extract the files in the zip directory and copy all .mod files into (.\components\mechanisms\modfiles\)
4. Run nrnmkdll in

(.\components\mechanisms\modfiles\)

Copy the resulting dll file from that directory into   
(.\components\mechanisms\)

1. Next, tell BMTK where to find your template files. In the root of your model directory, edit **circuit\_config.json**. Edit the templates\_dir key under components. Your file should look similar to the following (and may already exist):

|  |  |
| --- | --- |
|  | **circuit\_config.json** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | {  "manifest": {  "$BASE\_DIR": ".",  "$COMPONENTS\_DIR": "$BASE\_DIR/biophys\_components",  "$NETWORK\_DIR": "$BASE\_DIR/network"  },  "components": {  "morphologies\_dir": "$COMPONENTS\_DIR/morphologies",  "synaptic\_models\_dir": "$COMPONENTS\_DIR/synaptic\_models",  "mechanisms\_dir": "$COMPONENTS\_DIR/mechanisms",  "biophysical\_neuron\_models\_dir": "$COMPONENTS\_DIR/biophysical\_neuron\_templates",  "point\_neuron\_models\_dir": "$COMPONENTS\_DIR/point\_neuron\_templates",  "templates\_dir":"$COMPONENTS\_DIR/templates"  },  "networks": {  "nodes": [],  "edges": []  }  } |

1. Create a **templates** directory under .\components - if it doesn’t already exist.
2. Create a new file named **HCOCell.hoc** in your ‘templates’ directory and paste the following code into that file:

|  |  |
| --- | --- |
|  | **HCOCell.hoc** |
| 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  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79 | begintemplate HCOcell  public NumSoma  NumSoma = 1  public soma // declares object soma is a public object that can be accessed by any procedures/functions  create soma[NumSoma] // declares soma as a membrane compartment object    public all, somatic, basal, apical, axonal  objref all, somatic, basal, apical, axonal  objref this      proc init() {    all = new SectionList()  somatic = new SectionList()  basal = new SectionList()  apical = new SectionList()  axonal = new SectionList()      for i=0,NumSoma soma[i] { all.append()  somatic.append()}    /////// geometrical properties //////  soma[0] {  nseg=1 // create only one segment in the soma  // gives area of .628e-3 cm^2  L=1000 // (micrometer)  diam=9.99593 // (micrometer)  }    /////// biophysical properties //////  soma[0] {  cm = 1 // (microF/cm2)    //insert the appropriate channels and give them reversal potentials and conductances  insert leak  insert cat  insert cas  insert kdr  insert ka  insert kca  insert capool  insert hyper    eleak = -50 // (mV)  gbar\_leak = .03e-3 // (siemens/cm2)    cao = 3  cai = 50e-6  gbar\_cat = .02 //(.005~.01 siemens/cm2)  gbar\_cas = .01 //(.001~.005 siemens/cm2)    ek = -80  gbar\_kdr = .1 // (.1~.5 siemens/cm2)  gbar\_ka = .3 // (.1~.5 siemens/cm2)  gbar\_kca = .01 //// (.01~.05 siemens/cm2)    eh=-20  gbar\_hyper = .0002 // (.0001~.0003 siemens/cm2)    insert na  ena = 50 // (mV)  gbar\_na = 0.5 // (siemens/cm2)    }    define\_shape()    }  endtemplate HCOcell |

1. Things to note about this file:
   1. There are public section references used by bmtk:
      1. Line 11: public all, somatic, basal, apical, axonal
   2. The sections are indexable, i.e., soma is not a single object but an array.
      1. Line 8: create soma[NumSoma]
      2. Line 27: append your sections to the correct section list
   3. This can be any hoc file that specifies a template. This will be where a good majority of your model customization will be.
   4. **Line 75: define\_shape() must be called** if you don’t define a 3d morphology in the hoc otherwise. BMTK relies heavily on 3d locations.
2. Something important to note, which may be a limitation of BMTK: the morphology file is ALWAYS required. For any hoc file loaded you must specify an swc, however, it can be a blank file, as it will be ignored. Create the **blank.swc** file in .\biophys\_components\morphologies and leave this file blank.

Later versions of BMTK may not require the blank.swc file.

|  |  |
| --- | --- |
|  | **blank.swc** |
| 1 |  |

1. Next, you’ll need to create a cell builder script to tell BMTK the type of cells you want to use. Create the file **build\_network.py** in the root of your directory and add the following code:

|  |  |
| --- | --- |
|  | **build\_network.py** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | from bmtk.builder.networks import NetworkBuilder  net = NetworkBuilder('hco\_net')  net.add\_nodes(cell\_name='HCOCell',  model\_type='biophysical',  model\_template='hoc:HCOcell',  morphology='blank.swc',  HCOCell='HCOCell'  )  net.build()  net.save\_nodes(output\_dir='network') |

1. Now you should be ready to build your network. In your Anaconda Prompt, in the root of your directory execute the following command to build your network:

python build\_network.py

A successful run may not have any output.

1. Before running your simulation you will need to tell BMTK which generated network files are to be used in your simulation. ANY time you change your network configuration (networks, edges, etc) this will need to be updated. These files were generated in the previous step and exist in the network directory. Edit **circuit\_config.json**. Add the appropriate ”networks” key values that correspond to the files generated in the **network** directory. Your file should look like the following:

|  |  |
| --- | --- |
|  | **circuit\_config.json** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | {  "manifest": {  "$BASE\_DIR": ".",  "$COMPONENTS\_DIR": "$BASE\_DIR/biophys\_components",  "$NETWORK\_DIR": "$BASE\_DIR/network"  },  "components": {  "morphologies\_dir": "$COMPONENTS\_DIR/morphologies",  "synaptic\_models\_dir": "$COMPONENTS\_DIR/synaptic\_models",  "mechanisms\_dir": "$COMPONENTS\_DIR/mechanisms",  "biophysical\_neuron\_models\_dir": "$COMPONENTS\_DIR/biophysical\_neuron\_templates",  "point\_neuron\_models\_dir": "$COMPONENTS\_DIR/point\_neuron\_templates",  "templates\_dir":"$COMPONENTS\_DIR/templates"  },  "networks": {  "nodes": [  {  "nodes\_file": "$NETWORK\_DIR/hco\_net\_nodes.h5",  "node\_types\_file": "$NETWORK\_DIR/hco\_net\_node\_types.csv"  }  ],  "edges": [  ] }  } |

1. An important, step needed for the network to run correctly:  
   BMTK will need to have its default hoc loader overridden. You do this by editing your **run\_bionet.py** file to add the highlighted code:

|  |  |
| --- | --- |
|  | **run\_bionet.py** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  … | # -\*- coding: utf-8 -\*-  """Simulates an example network of 14 cell receiving two kinds of exernal input as defined in configuration file"""  import os, sys  from bmtk.simulator import bionet  from bmtk.simulator.bionet.default\_setters.cell\_models import loadHOC  bionet.pyfunction\_cache.add\_cell\_model(loadHOC, directive='hoc', model\_type='biophysical')  def run(config\_file):  conf = bionet.Config.from\_json(config\_file, validate=True)  conf.build\_env()  … |

1. Finally, run the simulation by executing the following in your Anaconda prompt, in the root of your project directory:

python run\_bionet.py simulation\_config.json

A successful run will output something like the following:

|  |
| --- |
| (clean) C:\Users\Tyler\Desktop\my\_bmtk\_model>python run\_bionet.py simulation\_config.json  2018-12-30 20:23:31,719 [INFO] Created log file  2018-12-30 20:23:31,779 [INFO] Building cells.  C:\Users\Tyler\Anaconda3\envs\clean\lib\site-packages\bmtk-0.0.7-py3.7.egg\bmtk\simulator\bionet\morphology.py:61: RuntimeWarning: invalid value encountered in true\_divide  r3dsoma /= n3dsoma  2018-12-30 20:23:31,796 [INFO] Building recurrent connections  2018-12-30 20:23:31,804 [INFO] Running simulation for 1000.000 ms with the time step 0.001 ms  2018-12-30 20:23:31,804 [INFO] Starting timestep: 0 at t\_sim: 0.000 ms  2018-12-30 20:23:31,810 [INFO] Block save every 5000 steps  2018-12-30 20:23:31,929 [INFO] step:5000 t\_sim:5.00 ms  2018-12-30 20:23:32,061 [INFO] step:10000 t\_sim:10.00 ms  2018-12-30 20:23:32,165 [INFO] step:15000 t\_sim:15.00 ms  2018-12-30 20:23:32,231 [INFO] step:20000 t\_sim:20.00 ms  …  2018-12-30 20:23:49,841 [INFO] step:985000 t\_sim:985.00 ms  2018-12-30 20:23:50,006 [INFO] step:990000 t\_sim:990.00 ms  2018-12-30 20:23:50,156 [INFO] step:995000 t\_sim:995.00 ms  2018-12-30 20:23:50,316 [INFO] step:1000000 t\_sim:1000.00 ms  2018-12-30 20:23:50,357 [INFO] Simulation completed in 18.55 seconds |

1. If you receive “PermissionError: [WinError 5] Access is denied: './output'” just run the network again.
2. At this point your network should have ran. From here we can customize the output and view results.
3. Edit the reports section of the **simulation\_config.json** file in the root of your model directory to look like:

|  |  |
| --- | --- |
|  | **simulation\_config.json** |
| …  26  27  28  29  30  31  32  33  34  35  36  37  … | "reports": {  "membrane\_report": {  "module": "membrane\_report",  "cells": "all",  "variable\_name": [  "cai",  "v"  ],  "file\_name": "cell\_vars.h5",  "sections": "soma"  }  } |

1. This will instruct BMTK to log the calcium and voltage levels for your cell.

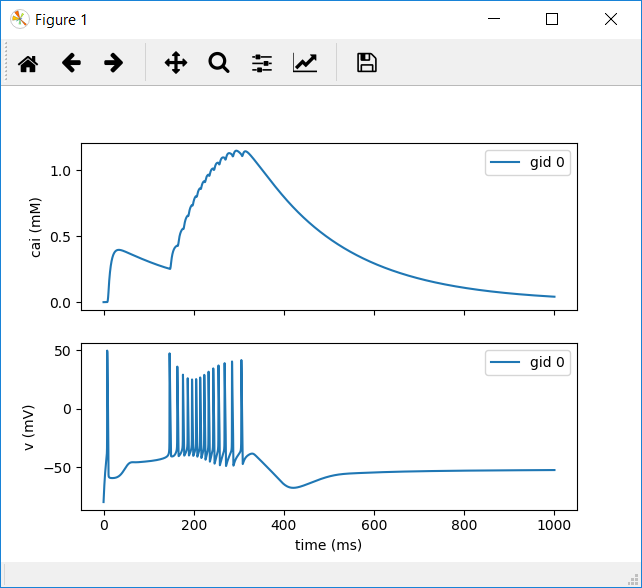
Run your network again, as we did previously, with (python run\_bionet.py simulation\_config.json)

1. Create a new file named **plot\_test.py** in the root of your directory and paste the following code into it:

|  |  |
| --- | --- |
|  | **plot\_test.py** |
| 1  2  3  4  5 | from bmtk.analyzer.cell\_vars import plot\_report  plot\_report(config\_file='simulation\_config.json') |

1. Executing this file with (python plot\_test.py) will return a plot like the following.

If you only see a single plot, that is ok, voltage is the important one here. Adding ‘cai’ to the simulation\_config reports section is for demonstration purposes only.



For additional resources and instructions on configuring BMTK see: <https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/Simulation_Intro.ipynb>