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
mkdir network
python -m bmtk.utils.sim_setup -n ./network 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
```

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

```
C:\Users\Tyler\Desktop\my_bmtk_model\biophys_components\mechanism
s
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 (`\biophys_components\mechanisms\modfiles\`) directory then copy the resulting dll file to the parent directory (`\biophys_components\mechanisms\`)

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 (`.\biophys_components\mechanisms\modfiles\`)

4. Run nrnmkdll in (`.\biophys_components\mechanisms\modfiles\`)

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

5. Next, tell BMTK where to find your template files. In the root of your model directory, edit **circuit_config.json**. Add the `templates_dir` key under components. Your file should look similar to the following:

circuit_config.json

```
{
  "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/hoc_templates"
  },
  "networks": {
    "nodes": [],
    "edges": []
  }
}

```

6. Create a **hoc_templates** directory under `.\biophys_components`

7. Create a new file named **HCOCell.hoc** in your new hoc_templates directory and paste the following code into that file:

HCOCell.hoc

```

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

```

```
16
17
18          proc init() {
19
20          all = new SectionList()
21          somatic = new SectionList()
22          basal = new SectionList()
23          apical = new SectionList()
24          axonal = new SectionList()
25
26
27          for i=0,NumSoma soma[i] { all.append()
28              somatic.append()}
29
30                  /////// geometrical properties //////
31                  soma[0] {
32                          nseg=1  // create only one segment in the soma
33                          // gives area of .628e-3 cm^2
34                          L=1000          // (micrometer)
35                          diam=9.99593    // (micrometer)
36                  }
37
38                  /////// biophysical properties //////
39                  soma[0] {
40                          cm = 1  // (microF/cm2)
41
42                          //insert the appropriate channels and give them
43  reversal potentials and conductances
44                          insert leak
45                          insert cat
46                          insert cas
47                          insert kdr
48                          insert ka
49                          insert kca
50                          insert capool
51                          insert hyper
52
53                          eleak = -50 // (mV)
54                          gbar_leak = .03e-3 // (siemens/cm2)
55
56                          cao = 3
57                          cai = 50e-6
58                          gbar_cat = .02 //(.005~.01 siemens/cm2)
59                          gbar_cas = .01 //(.001~.005 siemens/cm2)
60
61                          ek = -80
62                          gbar_kdr = .1 // (.1~.5 siemens/cm2)
63                          gbar_ka = .3 // (.1~.5 siemens/cm2)
64                          gbar_kca = .01 //// (.01~.05 siemens/cm2)
65
66                          eh=-20
67                          gbar_hyper = .0002 // (.0001~.0003 siemens/cm2)
68
69                          insert na
70                          ena = 50                // (mV)
71                          gbar_na = 0.5   // (siemens/cm2)
72
```

```
73                  }
74
75          define_shape()
76
77          }
78  endtemplate HCOcell
79
```

8. Things to note about this file:
    a. There are public section references used by bmtk:
        i. `Line 11: public all, somatic, basal, apical, axonal`
    b. The sections are indexable, i.e., soma is not a single object but an array.
        i. `Line 8: create soma[NumSoma]`
        ii. `Line 27`: append your sections to the correct section list
    c. This can be any hoc file that specifies a template. This will be where a good majority of your model customization will be.
    d. **`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.

9. 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.

blank.swc

```
1
```

10. 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  from bmtk.builder.networks import NetworkBuilder
 3
 4  net = NetworkBuilder('hco_net')
 5  net.add_nodes(cell_name='HCOCell',
 6                model_type='biophysical',
 7                model_template='hoc:HCOcell',
 8                morphology='blank.swc',
 9                HCOCell='HCOCell'
10               )
11  net.build()
12  net.save_nodes(output_dir='network')
13
```

11. 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.

12. 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

```
{
  "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":"$COMPONENTS_DIR/hoc_templates"
  },
  "networks": {
    "nodes": [
      {
        "nodes_file": "$NETWORK_DIR\\hco_net_nodes.h5",
        "node_types_file": "$NETWORK_DIR\\hco_net_node_types.csv"
      }
    ],
    "edges": [
    ]  }
}
```

13. 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

```
# -*- coding: utf-8 -*-
```

```
 2
 3  """Simulates an example network of 14 cell receiving two kinds of
 4  exernal input as defined in configuration file"""
 5
 6  import os, sys
 7  from bmtk.simulator import bionet
 8  from bmtk.simulator.bionet.default_setters.cell_models import loadHOC
 9
10  bionet.pyfunction_cache.add_cell_model(loadHOC, directive='hoc',
11  model_type='biophysical')
12
13  def run(config file):
14      conf = bionet.Config.from_json(config_file, validate=True)
15      conf.build_env()
16
 …  …
```

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

    ```
    python run_bionet.py simulation_config1.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
```

15. If you receive “PermissionError: [WinError 5] Access is denied: './output'” just run the network again.

16. At this point your network should have ran. From here we can customize the output and view results.

17. Edit the reports section of the **simulation_config.json** file in the root of your model directory to look like:

simulation_config.json

```
…
"reports": {
      "membrane_report": {
      "module": "membrane_report",
      "cells": "all",
      "variable_name": [
        "cai",
        "v"
      ],
      "file_name": "cell_vars.h5",
      "sections": "soma"
    }
}
…
```

18. 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`)

19. Create a new file named **plot_test.py** in the root of your directory and paste the following code into it:

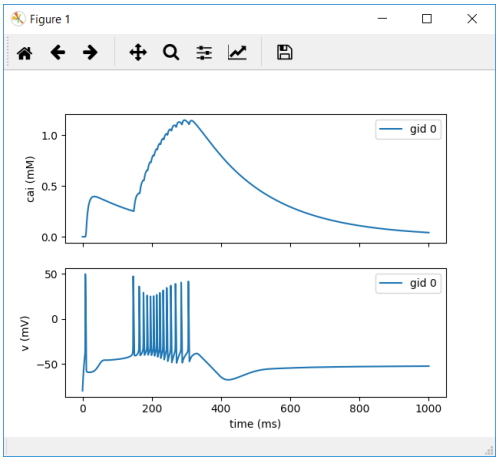
plot_test.py

```
from bmtk.analyzer.cell_vars import plot_report

plot_report(config_file='simulation_config.json')

```

20. Executing this file with (`python plot_test.py`) will return a plot like the following.



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