Additional Setup Information for Configuration/Build files

General Information

1. In the directory where you ran setup there should be several new files and folders.
    a. *run_bionet.py*
        i. Used from the command line to start a simulation
    b. *circuit_config.json*
        i. Specifies where BMTK will look for files related to:
            1. Cell morphologies
            2. Synaptic models
            3. mechanisms (compiled mod files)
            4. biophysical neuron models
            5. point neuron models
            6. Network nodes and edges
        ii. This file will remain relatively unchanged through simulations
    c. *simulation_config.json*
        i. You can create as many of these files as you want as it's specified when running each simulation using run_bionet.py. You can specify separate simulations with different
            1. run configurations (stop time, spike threshold, etc)
            2. conditions (temperature, v_init)
            3. inputs (current clamp, artificial)
            4. output (spikes from different cell types)
            5. reports
    d. *network/*
        i. When you add nodes and build the network (later) the csv and h5 files reside here until you run the network
    e. *biophys_components/biophysical_neuron_templates/*
        i. Holds fit **cell json files** defining cell properties/parameters like conductance
        ii. When creating a network and specifying new cells using the ***add_cells()*** method, the parameter ***dynamics_params*** looks here for a json file
    f. *biophys_components/mechanisms/*
        i. Neuron mkdll or nrnivmodl **compiled code or dll file must** be placed here
    g. *biophys_components/mechanisms/modfiles/*
        i. Location of **mod files**
    h. *biophys_components/morphologies/*
        i. **SWC files** for biologically realistic cells placed here.
    i. *biophys_components/synaptic_models/*
        i. Synapse files

Network Build Script for SWC/JSON

2. To begin, **you'll need to create a new script file**, **build_network.py**, responsible for specifying the cells to be used in your network. You can use the following as a template:

build_network.py

```
1
2  from bmtk.builder.networks import NetworkBuilder
3
4  net = NetworkBuilder('mcortex')
5  net.add_nodes(cell_name='Scnn1a_473845048',
6                potental='exc',
7                model_type='biophysical',
8                model_template='ctdb:Biophys1.hoc',
9                model_processing='aibs_perisomatic',
10               dynamics_params='472363762_fit.json',
11               morphology='Scnn1a_473845048_m.swc')
12
13 net.build()
14 net.save_nodes(output_dir='network')
15
16 from pprint import pprint
17 for node in net.nodes():
18     pprint(node)
19
20
```

a. ***Line 5*** - cell_name (optional) – record keeping and arbitrary to your model
b. ***Line 6*** - potential (optional) – Indicate it's an excitatory cell ('exc','inh')
c. ***Line 7*** - model_type – can be ('biophysical', point_process', 'point_soma', 'virtual')
    i. Reference: https://github.com/AllenInstitute/bmtk/blob/develop/bmtk/simulator/bionet/bionetwork.py#L54
    ii. At this point (11/18) 'point_soma' is not implemented
d. ***Line 8*** - model_template – signals we're using a biophysical cell
    i. Biophys1.hoc is located in BMTK installation at /bmtk/simulator/bionet/default_templates/Biophys1.hoc
    ii. (BioAxonStub.hoc, Biophys1.hoc, advance.hoc) are also available.
e. ***Line 9*** - model_processing – internal method for processing Allen institute files. See bmtk\simulator\bionet\default_setters\cell_models.py for all methods you can use.
f. ***Line 10*** - dynamics_parameters – **parameters json file** used to specify parameters of a cell located in (circuit_config.json -> biophysical_neuro_models_dir) typically (*./biophys_components/biophysical_neuron_templates/*)
g. ***Line 11*** - morphology – **swc morphology file** located in
h. ***Line 14 –*** output_dir was the location specified in "Installation/Setup" step 3.

Cell SWC and JSON Config Files

3. Preparing Dynamics parameters and SWC Files
    a. Initial notes:
        i. SWC Files are 3d representation files of real neuron cells
            1. See http://www.neuromorpho.org/myfaq.jsp for more
        ii. **Swc files** have to be in biophys_components/morphologies directory
        iii. **Cell json files** have to be in biophys_components/biophysical_neuron_templates
        iv. Names of these files need to be the same as what you specified in the **build_network.py** script
        v. You can download a base set of cells to play with from:
            1. http://celltypes.brain-map.org/neuronal_model/download/482934212
            2. These cells fit in the example provided in https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/01_single_cell_clamped.ipynb and referenced through this tutorial.
            3. Info: https://www.neuron.yale.edu/phpBB/viewtopic.php?t=3477
    b. SWC Files
        i. Format

http://research.mssm.edu/cnic/swc.html

n T x y z R P
n is an integer label that identifies the current point and increments by one from one line to the next.
T is an integer representing the type of neuronal **segment you're starting**, such as soma, axon, apical dendrite, etc. The standard accepted integer values are given below.

- 0 = undefined
- 1 = soma
- 2 = axon
- 3 = dendrite
- 4 = apical dendrite
- 5 = fork point
- 6 = end point
- 7 = custom

x, y, z gives the Cartesian coordinates of each node.
R is the radius at that node.
P indicates the parent (the integer label) of the current point or -1 to indicate an origin (soma).

        ii. The cells MUST contain an axon component (2) I'm not sure if this will be a problem for some models. There must be 3 items identified.

biophys_components/morphologies/simple.swc

```
#Sample cell morphology
1 1 0 0 0 5 -1
2 2 1000 0 0 5 1
3 2 1000 0 0 0.1 2
```

iii. SWC files are required
    1. The above cell is a soma with length 1000 and radius 5 (diam=10) with an unrealistic required axon (cm will be 0 later) due to neuron 3d limitations.
iv. I used NeuTube http://www.neutracing.com for editing manually initially.
v. Allen Institute references **VAA3D** for viewing
    1. Here: https://alleninstitute.org/bigneuron/algorithms/
    2. See homepage: http://home.penglab.com/proj/vaa3d/home/index.html
    3. Downloaded from https://github.com/Vaa3D/release/releases/

c. Cell JSON files

i. The cell properties like axial resistance, capacitance and channel conductance are all editable from here
    1. For a full example to be used in your model, download http://celltypes.brain-map.org/neuronal_model/download/482934212
ii. A json file to go with the **simple.swc** specified above would look like:

biophys_components/biophysical_neuron_templates/simple.json

```
{
  "passive": [
    {
    "ra": 1,
      "cm": [
        {
          "section": "soma",
          "cm": 1.0
        },
        {
          "section": "axon",
          "cm": 0.0
        }
      ],
      "e_pas": -50
    }
  ],
  "conditions": [
    {
      "erev": [
        {
          "ena": 50.0,
          "section": "soma",
          "ek": -80.0,
          "eleak":-50
        }
      ]
    }
  ],
  "genome": [
    {
```

```
            "section": "soma",
            "name": "gbar_leak",
            "value": 0.00003,
            "mechanism": "leak"
          },
          {
            "section": "soma",
            "name": "gbar_na",
            "value": 0.3,
            "mechanism": "na"
          },
          {
            "section": "soma",
            "name": "gbar_kdr",
            "value": 0.3,
            "mechanism": "kdr"
          }
        ]
}
```

iii. Soma and Axon must be defined

Simulation Environment (Compilation and Reports)

4. Setting up the simulation environment
    a. **Windows**
        i. Methods specified in the official documentation don't work because nrnivmodl doesn't exist in Windows. Use **mknrndll gui** instead.
        ii. Edit the simulation_config.json file to specify the reports you want to run

simulaton_config.json

```
 1
 2   "reports": {
 3     "membrane_report": {
 4       "module": "membrane_report",
 5       "cells": "all",
 6       "variable_name": [
 7         "v"
 8       ],
 9       "file_name": "cell_vars.h5",
10       "sections": "soma"
11     }
12   },
13
14
```

iii. Use **mknrndll** to compile the components/mechanisms/modfiles directory
iv. Move the nrnmech.dll file up one directory to components/mechanisms/

    b. **Linux/Mac**
        i. You can either:

1. Compile manually and edit a config file

```
cd components/mechanisms
```

```
nrnivmodl modfiles
```

simulaton_config.json

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

2. Build from the command line:

```
python -m bmtk.utils.sim_setup -n network --
membrane_report-vars v,cai --membrane_report-sections
soma --tstop 2000.0 --dt 0.1 bionet
```

3. Or create a python file with build instructions and run

build_network.py

```
from bmtk.utils.sim_setup import build_env_bionet

build_env_bionet(network_dir='network', tstop=2000.0, dt=0.1,
                 reports={'membrane_report': {
                         "module": "membrane_report",
                         'cells': 'all',
                         'variable_name': ['cai', 'v'],
                         "file_name": "cell_vars.h5",
                         "sections": "soma",
                     }})
```

Current Clamp Input

5. Current clamp input (**OPTIONAL**)
    a. In the **simulation_config.json** file place the following code block in the input section

simulaton_config.json

```
```

```
"inputs": {
     "current_clamp": {
      "input_type": "current_clamp",
      "module": "IClamp",
      "node_set": "all",
      "amp": 0.120,
      "delay": 500.0,
      "duration": 1000.0
    }
},

```

i. "node_set" can be set to a list. Using [0] would reference the soma for all models

b. It's possible to have multiple clamps at different times and amps through a single simulation. Just duplicate the "current_clamp" section, separated by a comma. Ex:

simulaton_config.json

```
"inputs": {
     "cclamp1": {
      "input_type": "current_clamp",
      "module": "IClamp",
      "node_set": "all",
      "amp": 0.150,
      "delay": 0.0,
      "duration": 400.0
    },

    "cclamp2": {
      "input_type": "current_clamp",
      "module": "IClamp",
      "node_set": "all",
      "amp": 0.300,
      "delay": 500.0,
      "duration": 400.0
    }
},
```

Virtual Input

6. Virtual spike input (**OPTIONAL**)

https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/02_single_cell_syn.ipynb

Networked Cells

https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/03_single_pop.ipynb

https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/04_multi_pop.ipynb

Running the simulation and Output

Running Bionet

1. To **Run** your network execute:

```
python run_bionet.py simulation_config.json
```

Viewing Results

1. View spike times:

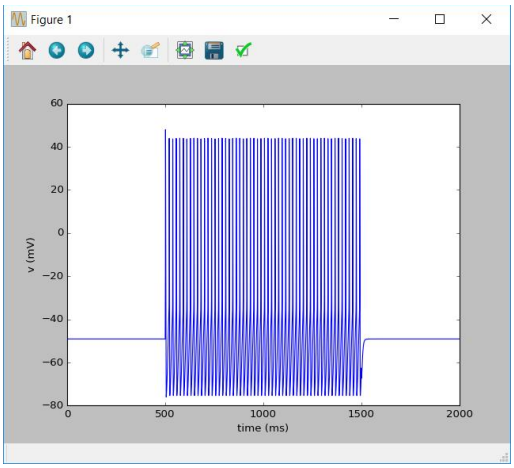
spikes.py

```
1
2  from bmtk.analyzer.spike_trains import to_dataframe
3
4  to_dataframe(config_file='simulation_config.json')
5
```

2. View output graphs specified in the “reports” section

graphs.py

```
1
2  from bmtk.analyzer.cell_vars import plot_report
3
4  plot_report(config_file='simulation_config.json')
5
```



3. View raster plot for population of cells

raster.py

```
1
2  from bmtk.analyzer.spike_trains import raster_plot
3
4  raster_plot(cells_file='network/mcortex_nodes.h5',
5  cell_models_file='network/mcortex_node_types.csv',
6  spikes_file='output/spikes.h5')
7
```

Additional References

https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/01_single_cell_clamped.ipynb

https://github.com/AllenInstitute/sonata/blob/master/docs/SONATA_DEVELOPER_GUIDE.md#nodes---required-attributes

https://github.com/AllenInstitute/bmtk/tree/develop/docs/tutorial

Things to remember

1. Compilation needs to be done manually for Windows systems
2. Nrnmech.dll has to be in the directory above modfiles – biophys_components/mechanisms
3. Swc files have to be in the morphologies directory
4. Fit_json files containing cell parameters have to be in biophys_components/biophysical_neuron_templates

Other notes:

Bmtk/simulator/bionet/default_templates/BioAxonStub.hoc is missing load_file("import3d.hoc") at begining (https://www.neuron.yale.edu/phpBB/viewtopic.php?t=3257)

Useful: SWC Viewer from different databases. No downloading

https://neuroinformatics.nl/HBP/morphology-viewer/#

Questions for Allen Institute

1. Single compartment cells - a couple of our models are very small and the

morphology/electrophysiology is entirely defined in hoc. How can BMTK handle something simple like a single hoc file? Much of BMTK is centered around JSON and SWC for the Allen Institute's cell database.

2. Is there any way around using SWC files for cell morphology? I noticed NeuroML is an acceptable format, which may make question 1 possible through hoc to nml conversion but the morphology reader in BMTK hasn't been implemented. Are there any SWC alternatives?

1 & 2)
In theory bmtk should be able to read in any NEURON template file. In the nodes.csv file update model_template to "hoc:MyNRNTemplate.hoc", where MyNRNTemplate.hoc is the name of your template file (if the path is not absolute or in your working directory it will look under "components/templates" in the config.json). The template must have a "soma" section, but it will ignore the SWC file and assume the morphology is being built into the template

However it that doesn't work you can override the way bmtk loads each individual cell. To do so add the following to the run_bionet.py script (before the __main__ section:

```
def loadHOC(cell, template_name, dynamics_params):
    param1 = cell['param1'] # Use cell dict to get each cell's
attributes,
    tau = dynamics_params['tau'] # dynamics params values store in
json file (is not None)
    hobj = ... # Create and return a NEURON HOCObject
    return hobj
bionet.pyfunction_cache.add_cell_model(loadHOC, directive='hoc',
model_type='biophysical')
```

Inside the loadHOC method you can call another NEURON hoc file, or build the cell directly using neuron's python api. It will ignore the swc files unless you want it too. BMTK will execute this function for every cell with a model_template parameter that starts with "hoc:". If you run into any issues overriding the loadHOC method please feel free to make or github issue or just email me directly (kaeld@alleninstititute.org) and we can try to fix the issue.

3. If we were to implement the previously mentioned morphology reader for NML files would this be a complete solution for the previous questions? Is this something that has been considered for near future development there?

You can use the same method as above to update the way NML templates are loaded. Here is a link to the current way bmtk loads NML based cells.

There is no immediate plans to read and load the morphology from the NML method. And unfortunately, last time I checked, there doesn't seem to be an easy way in python to convert a NML <cell> into a HOC object (I had to write a primitive parser just to read the membrane properties). But if

you have a method to do so you can override the NMLLoad function - and if you do let me know because I think it would be a good feature to add to the main branch.

4. How customizable are the synapse JSON files? Specifically, for neuromodulation effects for short/long term plasticity.

You can use any synapse mechanism built into NEURON or that you've created and compiled yourself. The json files just have to match up with the parameters for that given synapse.

We actually have our own custom stp synapses, and are used in the bio_stp_models examples. If you look under network/ext_to_slice_edge_types.csv you'll see for a certain number of connections the model_tempate is set stp2syn which uses different json file than the standard exp2syn.

5. Is it possible to record LFPs from a set of cells?

You can turn on LFP recordings in the config.json file, as part of the "reports" section. Here is an example config with lfpy recordings of the first 10 cells.

You will likely need to adjust the electrode.csv to coordinates that are more appropriate for your network. After the simulation is finished it will record and sum up the lfp's from all "cells" as an hdf5 into file_name specified (which is just a table where each row is a given time step and each column a different electrode). It will also store each's cell's individual lfp contribution into the "contributions_dir", unless that value is null or removed from the config.