**My user in cluster and how to log in:**

First connect to the jump server:

ssh [allencenter@223.3.33.234](mailto:allencenter@223.3.33.234) or [allencenter@223.3.34.110](mailto:allencenter@223.3.34.110)

password: seu@jump!allen

Then connect to cluster: (my user account)

ssh [ylycenter@172.16.1.100](mailto:ylycenter@172.16.1.100)

password: seu@jump!yly

**Path to my folder:**

In the jump server, after logging in, the title should be [allencenter@MiWiFi-RA71-srv ~]$.

To get to my folder, first cd ../.. then cd data/PBS/SEU-ALLEN/Users/lyuyin

These are the documents I have in the jump server

:

文本

描述已自动生成

In cluster, after logging in, the title should be [ylycenyer@admin ~]$. This is my folder in cluster. The path after typing pwd is: /public/home/ylycenter

These are the documents I have in cluster:

文本

描述已自动生成

The files are a bit messy here. The downloaded files from Dropbox are in ‘v1\_biophysical’. Mainly these are the original files downloaded without any changes. But several weeks ago Penghao also used my account to run the network, so the files are messed together.

图形用户界面, 文本, 应用程序

描述已自动生成

The link to Dropbox original data is as follows:

<https://www.dropbox.com/scl/fo/9d33eedau0iavt7afwzgn/AKBfpIknOVdPM4GNrZm8K-0?rlkey=gsz56gzw37lq29mf7faxtn7b1&e=1&dl=0>

on the link, we go to simulations -> v1\_biophysical, these are the files we need. Information and explanation of each file are in the Readme.txt they provided.

Here in the cluster in v1\_biophysical, the ‘output’ document is a folder for the simulation output (for running the simulation step). Every time we run the simulation; the ‘output’ file will be overwritten. You can change the folder name other than ‘output’ to save the result. Here I have ‘output’ and ‘output100’, where ‘output’ is for original data, and ‘output100’ is for the data where ‘syn\_weight’ column in ‘v1\_v1\_edge\_types.csv’ is changed to be 100 times larger than original data.

The ‘network’ folder is generated after rebuilding the network by running ‘build\_network.py’. This folder will also be overwritten every time we rebuild the network. It includes files like ‘v1\_nodes.csv’, ‘v1\_node\_types.csv’, ‘v1\_v1\_edge\_types.csv’, etc.

For running the simulation (the first task I completed in cluster), it can be divided into 2 parts:

-building the network and generate the ‘network’ folder

-use the data generated in the ‘network’ folder to run simulation

At first, we don’t want to rebuild network and merely change one of the files generated under the ‘network’ folder, ‘v1\_v1\_edge\_types.csv’, change the column ‘syn\_weight’ to be 100 times larger, and create a new csv called ‘v1\_v1\_edge\_types\_100.csv’. Then we changed corresponding code in config.json: in line 75 of the code, change the corresponding edge type file to be ‘v1\_v1\_edge\_types\_100.csv’, and run the simulation. The file ‘v1\_v1\_edge\_types\_100.csv’ is saved in ‘network\_old’ folder. (v1\_v1\_edge\_types\_10.csv and v1\_v1\_edge\_types\_1000.csv are 10 and 1000 times larger, respectively)

图形用户界面, 文本

描述已自动生成

Other folders like ‘network\_rebuilt’ are created by Penghao to save the network output.

The ‘output’ files are output for running the simulation. It includes ‘config.json’, ‘log.txt’, ‘spikes.csv’, and ‘spikes.h5’.

‘spikes.csv’ and ‘spikes.h5’ are almost the same, with three columns ‘timestamps’, ‘population’, and ‘node\_ids’. This is to say, to represent a spike we need three things: the id of the node (e.g. cell) that spikes, the name of the population (in this case the area of the brain where the node came from), and the time of the spike. By default, Sonata assumes the units are in milliseconds but that can be changed in the format.

Visualizations can then be made based on these data. I will elaborate it later.

To understand the code in ‘build\_network.py’, there are some links and tutorials provided by AllenInstitute:

Article link: (star methods on page 18)

<https://www.cell.com/action/showPdf?pii=S0896-6273%2820%2930067-2>

**AllenInstitute bmtk NetworkBuilder Intro:**

<https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/NetworkBuilder_Intro.ipynb>

This demonstrates basic code for creating network.

<https://github.com/AllenInstitute/bmtk/blob/develop/docs/tutorial/Simulation_Intro.ipynb>

This includes tutorial for setting up a configuration file.

2024 BMTK workshop:

<https://github.com/AllenInstitute/bmtk-workshop/blob/418b59132cad0d4424ab6d56532098721029d735/Ch2_single_cell/2.%20Single%20Cell.ipynb>

**AllenInstitute/sonata**

<https://github.com/AllenInstitute/sonata/blob/master/tutorials/pySonata/spike-reports.ipynb>

This provides information on how to read sonata h5 format and analyze it.

Then if we go back to ylycenter cluster to look at other files,

文本

描述已自动生成

It looks a bit messy because later, Penghao and I use the same account together. He added some of the files when trying to rebuild the network. The ‘tar.gz’ documents are required packages downloaded and can be ignored, the three‘buildnetwork.out’files should be created by Penghao and are the output txt after rebuilding the network.

Most of the **important files** are on the right side:

v1\_biophysical: includes all documents that are downloaded from Dropbox

v1\_point: includes all documents that are downloaded from Dropbox

Files for point model:

test\_point.sh: running the simulation

Files for biophysical model:

test.sh: running the simulation:

run\_build\_network.sh: rebuilding the network (in parallel)

run\_build\_network\_new.sh: rebuilding the network (single core no parallel)

out: output txt for rebuilding the network (in parallel)

out\_new: output txt for rebuilding the network (single core no parallel)

Other files are not important and can be ignored.

Now we want to scale E2/3 and E5 connection probabilities in ‘v1\_v1\_edge\_models.csv’ according to the relative distribution of boutons and/or spatial extent of dendrites/axons in these layers. I’m not sure about the specific scaling since Penghao said he has already done this part, and he already has code for changing the data. The problem now would be that we cannot rebuild the network. Penghao said, once we can run the rebuild command successfully, we can change original data with our data, and rebuild the network. Then we can run the simulation and see the results.

So this last week, I was focusing on this and tried to troubleshoot, but there were no output generated under ‘network’ folder. I’ve just sent the email to AllenInstitute. The detailed issues are described there.

Github link that I asked the question about rebuilding network:

<https://github.com/AllenInstitute/bmtk/issues/380>

For Penghao’s part, I saw that he sent a weekly work report in the group on Thursday. I think he has mentioned his latest updates.

Below are the **details for analysis and visualization**: (mainly based on spikes.h5/spikes.csv output file)

I only did visualizations for the original output file (without changing weights). Comparison can be made by drawing same graph with changed data and compare the graphs.

表格

描述已自动生成

Again to elaborate, both the csv and the h5 file represent spikes in the network. To represent a spike, we need three things: the id of the node (e.g. cell) that spikes, the name of the population (in this case the area of the brain where the node came from), and the time of the spike (by default in milliseconds). These are the 3 variables in the simulation output (both csv and h5). Here all populations are from v1. So we visualized node id vs time directly.

Before visualization, comparisons can be made based on number of nodes and total number of spikes. I’ve done a comparison for h5 files before and after x100: (those files are in v1\_biophysical -> output -> spike.h5 and v1\_biophysical -> output100 -> spikes.h5)

From the h5 files, we see that with original data, there are 171900 nodes in the network, meaning that there are 171900 cells join the connection. The total number of spikes is 1875007. So the average spike rate is 1875007/171900=10.9075~11, meaning that on average, each neuron fires 11 times within 3 seconds.

After changing the weights to be 100 times larger, we see huge increase in the spikes. There are 193496 nodes in the network, meaning that there are 193496 cells join the connection. The increase is due to increase in synaptic weights (connection probability)— the connection probability increases 100 times, so number of cells that have connections within the network also increase. The total number of spikes is 168135773. So the average spike rate is 168135773/193496=868, meaning that on average, each neuron fires 868 times within 3 seconds.

This demonstrates huge increase in the spikes rate. It also reveals the reason why the simulation runs much more slowly this time—due to large increase in number of connections.

**Visualization:** (for the visualization part I downloaded all the output files and run the code on local computer) we need to import both spikes.csv and v1\_nodes.csv so that the node ids correspond. The code are mainly taught and written by Penghao.

With Penghao’s help, I did a simple visualization. Here is a typical graph generated using the csv output from original data. We took the first 10000 rows of data.

图表, 散点图, QR 代码

描述已自动生成

Then we want to divide the node by inhibitory and excitatory cell types and visualize using different colors. Here’s a figure generated after adding colors to each cell type. (red-excitatory, blue-inhibitory)

图表, 散点图, QR 代码

描述已自动生成

The **code** for the above two graphs is in the file ‘visualization 1&2.py’.

I have 2 codes for making the second graph. To draw the first graph, simply delete the color\_list.

We also did a visualization of excitatory and inhibitory neurons in 3D space.

We want to see the location of each spike in the brain using the coordinates provided and divide them by excitatory and inhibitory cell types. We divided the timestamps by 400 milliseconds and took visualizations of spikes in first 2000 milliseconds.

图示, 散点图

描述已自动生成

The **code** for 3d scatter plot is in the file ‘3d scatter plot.py’.

link for Matplotlib 3D scatterplot:

<https://matplotlib.org/stable/gallery/mplot3d/scatter3d.html>

This provides code for making 3D scatterplot that can be used for spikes.h5 output.

Above are the main types of graphs we can create for visualizing the output and making comparison. I did not continue making more graphs since the changes didn’t have biological meaning. Once the connection probabilities are scaled, similar visualizations can be done, and comparisons can be made.