Lapish Lab Spike Sorting

Contents

[1 Introduction 3](#_Toc535574995)

[2 Version Management/History 3](#_Toc535574996)

[3 Gathering Electrophysiological Data 3](#_Toc535574997)

[4 Getting Started 3](#_Toc535574998)

[4.1 Getting a Big Red 2 Account 4](#_Toc535574999)

[4.2 Accessing Big Red 2 4](#_Toc535575000)

[4.3 Basic Operations in Matlab and Linux 4](#_Toc535575001)

[4.3.1 Executing a Command 4](#_Toc535575002)

[4.3.2 Starting and Exiting Matlab 4](#_Toc535575003)

[4.3.3 Changing Directories 5](#_Toc535575004)

[4.3.4 Editing Files 5](#_Toc535575005)

[4.4 Transferring and Copying Files 5](#_Toc535575006)

[4.5 Accessing the Data Capacitor 7](#_Toc535575007)

[4.6 Accessing IU Box 7](#_Toc535575008)

[4.6.1 Creating Your IU Box SFTP Password 7](#_Toc535575009)

[4.6.2 Accessing IU Box with SFTP or Web Browser 7](#_Toc535575010)

[4.7 Running an Interactive Job on Big Red 2 7](#_Toc535575011)

[4.8 Getting the Spike Sorting Software 8](#_Toc535575012)

[4.9 Loading the Necessary Modules on Big Red 2 8](#_Toc535575013)

[4.10 Installing Kilosort 9](#_Toc535575014)

[4.11 Testing Kilosort 9](#_Toc535575015)

[5 Directory Organization 10](#_Toc535575016)

[5.1 Data Capacitor 10](#_Toc535575017)

[5.2 lapishLabCluster Directory 11](#_Toc535575018)

[6 Creating Site Mapping Files 11](#_Toc535575019)

[6.1 siteMapCreate Script 12](#_Toc535575020)

[6.2 Simple Example Channel Mapping 13](#_Toc535575021)

[7 Spike Sorting an Individual Data Set 15](#_Toc535575022)

[7.1 Stage 1 15](#_Toc535575023)

[7.2 Stage 2 15](#_Toc535575024)

[7.3 Stage 3 16](#_Toc535575025)

[8 Spike Sorting Multiple Data Sets in Parallel 16](#_Toc535575026)

[9 Output File Information 17](#_Toc535575027)

[10 Stage 1 Review GUI 18](#_Toc535575028)

[11 Stage 2 Review GUI 19](#_Toc535575029)

[12 Kilosort File Change Log 21](#_Toc535575030)

[13 Common Errors/Problems 21](#_Toc535575031)

[13.1 FTP Transfer Fails 21](#_Toc535575032)

[14 Tips for Working on Big Red 2 22](#_Toc535575033)

[15 Using Phy to Visualize Data 22](#_Toc535575034)

[15.1 Installing Miniconda 22](#_Toc535575035)

# Introduction

This documentation and the accompanying software carry out spike sorting of open ephys data with kilosort using high performance computing clusters at Indiana University. This manual will walk through the steps of installing the software (Section 4), preparing the data for spike sorting (Sections 3 and 6), and spike sorting the data (Sections 7-11). The process of spike sorting can be carried out on an individual data set in an interactive fashion (Section 7) or in parallel on multiple data sets (Section 8).

The general work flow is shown below:

1. Put electrophysiological data on IU Box.
2. Stage 1
   1. Move data from IU Box to the Data Capacitor
   2. Gather basic data information
   3. User reviews raw voltage traces to look for noisy channels to reject
3. Stage 2
   1. Spike sort the accepted channels
   2. Calculate statistics from resulting spiking neurons
   3. User reviews neuron spike statistics to look for bad neurons to reject
4. Stage 3
   1. Organize the remaining neuron spike data
   2. Transfer data from the Data Capacitor to IU Box

# Version Management/History

Version 1: The first version of this software package was created throughout December 2018.

Version 2: The second version was created in January 2019. Improvements to this version included addressing a bug in transferring files via SFTP between the Data Capacitor and IU Box, implementing a stand-alone interactive spike sorting pipeline, moving the main stage functionality into core functions, improvements to the manual, and posting the software on github. Also, the name was changed to lapishLabCluster.

# Gathering Electrophysiological Data

This manual and analysis software assumes the data have been gathered with open ephys, but other formats could work as well with sufficient modification. The data for each individual recording (including all .continuous and supporting files generated by a recording) should be stored in its own directory on IU Box (see Section 4.6). No naming conventions are necessary other than the standard output file names from open ephys.

# Getting Started

## Getting a Big Red 2 Account

1. Go to <https://one.iu.edu/task/iu/account-creation?searchTerms=create%20accounts>
2. Click start, then indicate you want a Big Red 2 account (as well as Karst and Carbonate accounts if you don’t have them and you’re interested). Fill out the information and tell them you will cite to them and notify them of citations.
3. Wait 0 to 24 hours to get an account

## Accessing Big Red 2

1. How you access Big Red 2 is dependent on the operating system of your local machine.
   1. Windows
      1. Download Putty to use as an ssh client (<https://iuware.iu.edu/Windows/title/781>).
         1. Host Name: username@bigred2.uits.iu.edu
         2. Port: 22
         3. Connection type: SSH
         4. Click Open
         5. A terminal will open and request your IU passphrase.
         6. Duo will request secondary authentication.
   2. Linux/Mac
      1. Open a terminal and enter the following:

**ssh -X -L 5900:bigred2.uits.iu.edu:5901 username@bigred2.uits.iu.edu**

Replace the username with your IU user name. You will be prompted for your IU passphrase and a DUO authentication.

1. After using either of these ssh methods, the command line on a Big Red 2 login node will appear. From here you can start Matlab (see Section 4.3.2), start an interactive job (see Section 4.7), or submit and monitor parallel jobs (see Section 8).
2. See the Big Red 2 website for more information about using ssh or sftp to access Big Red 2 (<https://kb.iu.edu/d/bcqt>).

## Basic Operations in Matlab and Linux

### Executing a Command

The most basic operation in all of is executing a command. This process involves typing something into the command line on the terminal and pressing enter (see Section 4.2 for information about getting to a command line in a terminal via SSH). Throughout this manual, we will note this process by bolding some text. For instance, if you see **example**, you should type “example” (without quotation marks) into the command line and press enter.

### Starting and Exiting Matlab

After you log in to Big Red 2 (see Section 4.2), you will get a command line in a terminal. Regardless of your directory location or if you are in an interactive job (see Section 4.7), you can start Matlab by simply executing **matlab**. It may take a few moments to start up. To exit Matlab, simply execute **exit** within Matlab.

### Changing Directories

Changing directories on Big Red 2 (linux) or in Matlab use the same commands. Here are the relevant commands (shown in bold):

1. **ls**: Displays the files and directories in your current directory.
2. **pwd**: Displays the full path name of your current directory.
3. **cd directoryName**: Changes the directory to directoryName, which must be a subdirectory of your current directory.
4. **cd fullDirectoryPathName**: Changes the directory to fullDirectoryPathName, which can be any directory (where fullDirectoryPathName is the full path name of the destination directory).
5. **cd ..**: Changes the directory to the parent directory that holds your current directory.
6. **cd~**: Changes the directory to your main directory (/gpfs/home/n/m/username/BigRed2).

### Editing Files

The process of editing files will depend on the operating system of the machine your local machine.

1. Windows
   1. Text files can be easily edits within the winscp program. Simply locate the file you wish to edit and double click on it to bring up a text editor. Make sure to save the file when you are finished.
2. Linux
   1. Under construction…
3. In general, files can be edited on your local machine using the method of your choice and then transferred to Big Red 2 or the Data Capacitor.

## Transferring and Copying Files

1. How you transfer files between your local machine and Big Red 2 is dependent on the operating system of your local machine.
   1. Windows
      1. Download Winscp as an sftp client to transfer data back and forth (<https://iuware.iu.edu/Windows/title/785>).
         1. Host name: bigred2.uits.iu.edu
         2. User name: username
         3. Password: Your IU Passphrase
         4. Port number: 22
         5. Click Login
         6. Duo will request secondary authentication.
         7. A two panel window will appear that will allow you to move files between your local machine (left panel) and Big Red 2 (right panel) by simply selecting files and dragging them across.
         8. Coping files can be accomplished by right clicking on a file and selecting duplicate. Alternatively, you can use the cp command from the command line (see Linux/Mac section below).
   2. Linux/Mac
      1. For all the Linux/Mac commands, note these values:
         1. localPath: The full path name for the local directory or file you wish to transfer to Big Red 2.
         2. Bigred2Path: The full path name of the destination directory on Big Red 2.
         3. username: Your IU username.
      2. Use the commands below to transfer files and directories between your local machine and Big Red 2 from the command line in a terminal when logged into Big Red 2 (see Section 4.2).
         1. For these commands below, note these values:
            1. localPath: The full path name for the local directory or file you wish to transfer to Big Red 2.
            2. Bigred2Path: The full path name of the destination directory on Big Red 2.
            3. username: Your IU username.
         2. **scp -r localPath username@bigred2.uits.iu.edu:bigred2Path**
            1. Transfers the local directory localPath to the directory bigred2Path on Big Red 2.
         3. **scp localPath username@bigred2.uits.iu.edu:bigred2Path**
            1. Transfers the local file localPath to the directory bigred2Path on Big Red 2.
         4. **scp -r username@bigred2.uits.iu.edu:bigred2Path localPath** 
            1. Transfers the Big Red 2 directory bigred2Path to the directory localPath on your local machine.
         5. **scp username@bigred2.uits.iu.edu:bigred2Path localPath** 
            1. Transfers the Big Red 2 file bigred2Path to the directory localPath on your local machine.
      3. Use the following commands to copy files and directories between directories on Big Red 2 from the command line in a terminal when logged into Big Red 2 (see Section 4.2):
         1. **cp sourcePath destinationPath**
            1. Copies the file sourcePath to the directory destinationPath on Big Red 2.
         2. **cp –R sourcePath destinationPath**
            1. Copies the directory sourcePath to the directory destinationPath on Big Red 2.

## Accessing the Data Capacitor

You can access the data capacitor from Big Red 2 by simply changing directories to your Data Capacitor scratch directory (/N/dc2/scratch/username) (see Sections 4.2 and 4.3.3).

## Accessing IU Box

When using the spike sorting software, you will not need to directly access IU Box yourself. All interactions with IU Box are handled automatically in the code. However, it is necessary to place your raw data on IU Box and you must create a special sftp password for IU Box that you will be required to enter into the spike sorting code.

### Creating Your IU Box SFTP Password

You need to setup a special password in IU Box that is unique to just that account. This password will be stored unencrypted in Matlab variables throughout the analysis process, so make sure the password is unique to just IU Box. In other words, don’t use your regular email password or your bank password here.

Start by going to <https://uits.iu.edu/box> in a web browser and logging in with your standard IU username and passphrase. In the upper right hand corner, click on your initials, then down to Account Settings. Scroll down to Authentication and setup a new password to use just for SFTP with IU Box.

### Accessing IU Box with SFTP or Web Browser

In general, you can access IU Box using sftp (see Section 4.4). See (https://community.box.com/t5/Upload-and-Download-Files-and/Using-Box-with-FTP-or-FTPS/ta-p/26050) for additional support. If you do need to access IU Box directly, the username for sftp is username@indiana.edu, server (host name) is ftp.box.com, and port is 990. Additionally, IU Box can be accessed via a web browser by simply going to https://uits.iu.edu/box and logging in with your standard IU username and passphrase (not the special password created for sftp in IU Box).

## Running an Interactive Job on Big Red 2

To do anything more than basic operations on Big Red 2 (e.g., anything beyond starting and monitoring jobs on the normal queue), you’ll need to start what’s called an “interactive job”. Typically, when you run a job on Big Red2 (BR2) you’d submit it, log out, and then wait to be notified when it is completed. However, interactive jobs allow you to run (and debug!) software in real time while logged into BR2. Importantly, use the qsub command below to initiate an interactive session ensures you get on a node with a GPU, which is a requirement of how we’ve configured kilosort.

To start an interactive session run the following code when logged in to Big Red 2:

qsub -I -l walltime=02:00:00 -l nodes=1:ppn=4 -l gres=ccm -q gpu

What does this command mean?

1. nodes=1 means you’re requesting 1 node or core.
2. ppn=4 means you are requesting 4 processors per node. Increase this if you would like (up to the maximum of 16). Note that requesting more processors will make it take longer for the job to start.
3. walltime=2:00:00 means you are requesting 2 hours. Increase this if you would like (up to the limit of 12 hours, I think). Note that requesting more time will make it take longer to get a node.
4. –q gpu means you want a node with a GPU. If you don’t want or need a gpu, this can be removed.
5. gres=ccm means you want the ccm environment (which you do)

After waiting a while (just a few seconds to a few hours depending on how busy Big Red 2 is and the resources you requested), you should be told that the job has started and you’ll get a prompt. Execute **ccmlogin** (to tell the system you want a computation node) and then execute **matlab** (to start up MATLAB).

You are now running matlab on the interactive job and you can run the spike sorting software (see Section 7). If you don’t need to run matlab for whatever you’re doing, you can skip starting up matlab.

## Getting the Spike Sorting Software

1. If you haven’t done so already, download the lapishLabCluster directory to your local machine from github (https://github.com/nmtimme/lapishLabCluster). Where you locate it on your machine is up to you, but we would recommend somewhere logical (e.g., with the rest of your code, in a code related folder in your box sync folder, etc.).
   1. The directory can be downloaded using a web browser by going to https://github.com/nmtimme/lapishLabCluster and clicking “clone or download” on the right side.
   2. In Linux, the repository can be copied to your local machine from the command line. Go to the directory where you would like the software to be place and execute **git clone https://github.com/nmtimme/lapishLabCluster**.
2. Copy the lapishLabCluster directory from your local machine into your main Big Red 2 directory (see Section 4.4).
   1. Alternative: Log in to Big Red 2 and go to your main directory. Then, execute **git clone https://github.com/nmtimme/lapishLabCluster**.
3. Copy the parallel job scripts (spikeSortStageXJobVer1.txt, where X is the stage number (1, 2, or 3)) from the parallelJobs directory into your main Big Red 2 directory (see Section 4.4).
4. Edit the parallel job scripts (spikeSortStageXJobVer1.txt, where X is the stage number (1, 2, or 3)) in your main Big Red 2 directory to replace “username” on the third line with your IU username (see Section 4.3.4).

## Loading the Necessary Modules on Big Red 2

In your main Big Red 2 directories (/gpfs/home/n/m/username/BigRed2), edit the hidden text file .modules and add the following lines to the end of the file (see Section 4.3.4 for information about editing files, see <https://kb.iu.edu/d/bcwy> for more information about modules):

module load matlab/2016a

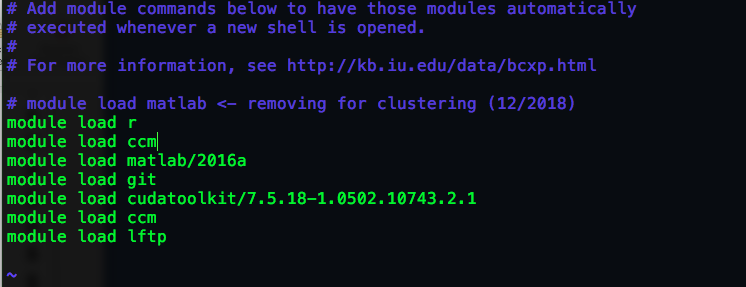
module load git

module load cudatoolkit/7.5.18-1.0502.10743.2.1

module load ccm

module load lftp

Note, if you’ve used Big Red 2 previously, you may have added the line “module load matlab” to your .modules file instead of “module load matlab/2016a”. Usually, “matlab” is preferable to “matlab/2016a” because the first selection will always load the most recent version of MATLAB. However, the kilosort software requires MATLAB 2016a, so using “module load matlab/2016a” is required for this spike sorting software. You can comment out “module load matlab”, as shown in the example below, rather than replace it to maintain a record of what you’re doing.

****

## Installing Kilosort

Log into Big Red 2, start matlab by entering “matlab” (without quotation marks) on the command line. Ensure that Matlab 2016 is being used – it should say this in a header when opening Matlab (see Section **Error! Reference source not found.**). Or you can double check with the **ver** command in Matlab.

Go to the CUDA directory in the kilosort directory (/gpfs/home/n/m/username/BigRed2/lapishLabCluster/kilosort/CUDA) (see Section 4.3.3).

Then, execute **mexGPUall**. You should get a bunch of warnings plus “MEX completed successfully” at the very end.

## Testing Kilosort

Now we are ready test the kilosort install. We’ll do this by using the “eMouse” data the kilosort developers provide. See the kilosort docs for more info on the eMouse data.

First, start an interactive job (at least 8 processors and at least 6 hours of walltime, gpu required) and then matlab. Next, go to the eMouse directory (/gpfs/home/n/m/username/BigRed2/lapishLabCluster/kilosort/eMouse) (see Section 4.3.3). Then, execute **master\_eMouse**. The test should run without error and produce some statistics about the test spike sort. Exploring this test can help you understand how kilosort works.

About this test, the kilosort github site says:

“You can verify that the code has been installed correctly by running master\_eMouse inside the eMouse folder. See first readme\_eMouse.txt. You can also use these scripts to understand how to pass the right settings into Kilosort (will depend on your probe, channel map configuration etc), and what you should be seeing in Phy during manual cleanup of Kilosort results. There are many parameters of the simulation which you can tweak to make it harder or easier, and perhaps more similar to your own data.”

# Directory Organization

## Data Capacitor

The spike sorting software will create a specific directory structure in your Data Capacitor directory. A typical complete spike sorting analysis might produce a directory and file structure like the one shown below. (Directories end with / and indents in the list below indicate nested directories. In other words, spikeSortTest1/ is a directory within /N/dc2/scratch/username/.) The user does not create these directories, but it is important to understand where your data is being stored.

* **/N/dc2/scratch/username/** <- This is your scratch directory on the Data Capacitor (username is your username). It is created by UITS when you request a Big Red 2 account.
  + **spikeSortTest1/** <- This is the main spike sorting directory for this analysis. It is set via the mainDC variable in the code. It can (and should) be set by the user. Stage 1 scripts will error if this directory already exists to prevent accidentally overwriting data.
    - **dataSet1/** <- This is the unique name given to the first data set. It is set using dataSetIDs in stage 1 scripts. If you are running an interactive job, there will only be only one data set. We recommend using names closely related to the data set, such as recording dates, animal numbers, and so forth. This directory will store the raw data transferred from IU Box, as well as all the files created throughout the spike sorting process for this data set. The contents of this directory will be copied back to IU Box to preserve all information about the spike sorting.
    - **dataSet2/** <- This is another data set.
    - **dataSet3/** <- This is another data set. Additional data sets can be added in stage1JobsToDo for parallel processing of multiple data sets.
    - **Stage1ResultsPreReview/** <- This is where reports from stage 1 that require user review on your local machine will be located.
    - **Stage1ResultsPostReview/** <- This is where reports from stage 1 that have been reviewed by the user on your local machine should be placed.
    - **Stage2ResultsPreReview/** <- This is where reports from stage 2 that require user review on your local machine will be located.
    - **Stage2ResultsPostReview/** <- This is where reports from stage 2 that have been reviewed by the user on your local machine should be placed.
    - **spikeSortingStage1Info.mat** <- This is a MATLAB file that records necessary information about stage 1 jobs (both interactive and parallel).
    - **spikeSortingStage2Info.mat** <- This is a MATLAB file that records necessary information about stage 2 jobs (both interactive and parallel).
    - **spikeSortingStage3Info.mat** <- This is a MATLAB file that records necessary information about stage 3 jobs (both interactive and parallel).

Note that not all of these directories will appear immediately at the start of the analysis and you do not need create them. The software will create these directories and files. Only spikeSortTest1 and the data set names (e.g., dataSet1, dataSet2,…) can be altered by the user in the code.

## lapishLabCluster Directory

The software package is contained in a directory called lapishLabCluster. Here is a description of the various subdirectories:

* coreFunctions
  + - This directory contains the main functions for stages 1, 2, and 3. This functions are utilized by both the interactive job scripts and the parallel jobs scripts. These functions do almost all the work in the spike sorting.
* interactiveJob
  + - This directory contains scripts necessary to run an interactive job spike sorting of a single data set.
* kilosort
  + - This directory contains the kilosort spike sorting software. Note, we did not create this software. We have chosen to include it in this package to ensure functionality by freezing this version of kilosort in place. More information about kilosort can be found on github (<https://github.com/cortex-lab/KiloSort>). We have also made some small changes to the kilosort code (see Section **Error! Reference source not found.**).
* npy-matlab
  + - This directory contains some necessary files for kilosort and functionality with phy (see Section 15). Note, we did not create this software. We have chosen to include it in this package to ensure functionality by freezing this version of npy-matlab in place. More information about npy-matlab can be found on github (<https://github.com/kwikteam/npy-matlab>).
* openEphysCode
  + - This directory contains code necessary to convert open ephys data into MATLAB format. We did not write this code. See open ephys for more information.
* parallelJobs
  + - This directory contains code necessary to run multiple spike sorting jobs in parallel on Big Red 2.
* reviewGUIs
  + - This directory contains the code for the GUIs that are used to review the results of stage 1 (see Section 10) and stage 2 (see Section 11).
* siteMapping
  + - This directory contains a script to create the necessary site mapping files (see Section 6).

# Creating Site Mapping Files

## siteMapCreate Script

During the spike sorting process, it is necessary to specify the physical location of electrodes in relation to one another. To do so, a mapping must be created of the electrode position in space and the conversion from electrode numbering on the probe to the electrode numbering in the open ephys system. (Note, a similar translation will be necessary with basically any recording system.)

The script siteMapCreate in the siteMapping directory helps to organize the site mapping process and produce the necessary variables used later in the analysis. This script is used on your local machine and produces files stored on your local machine.

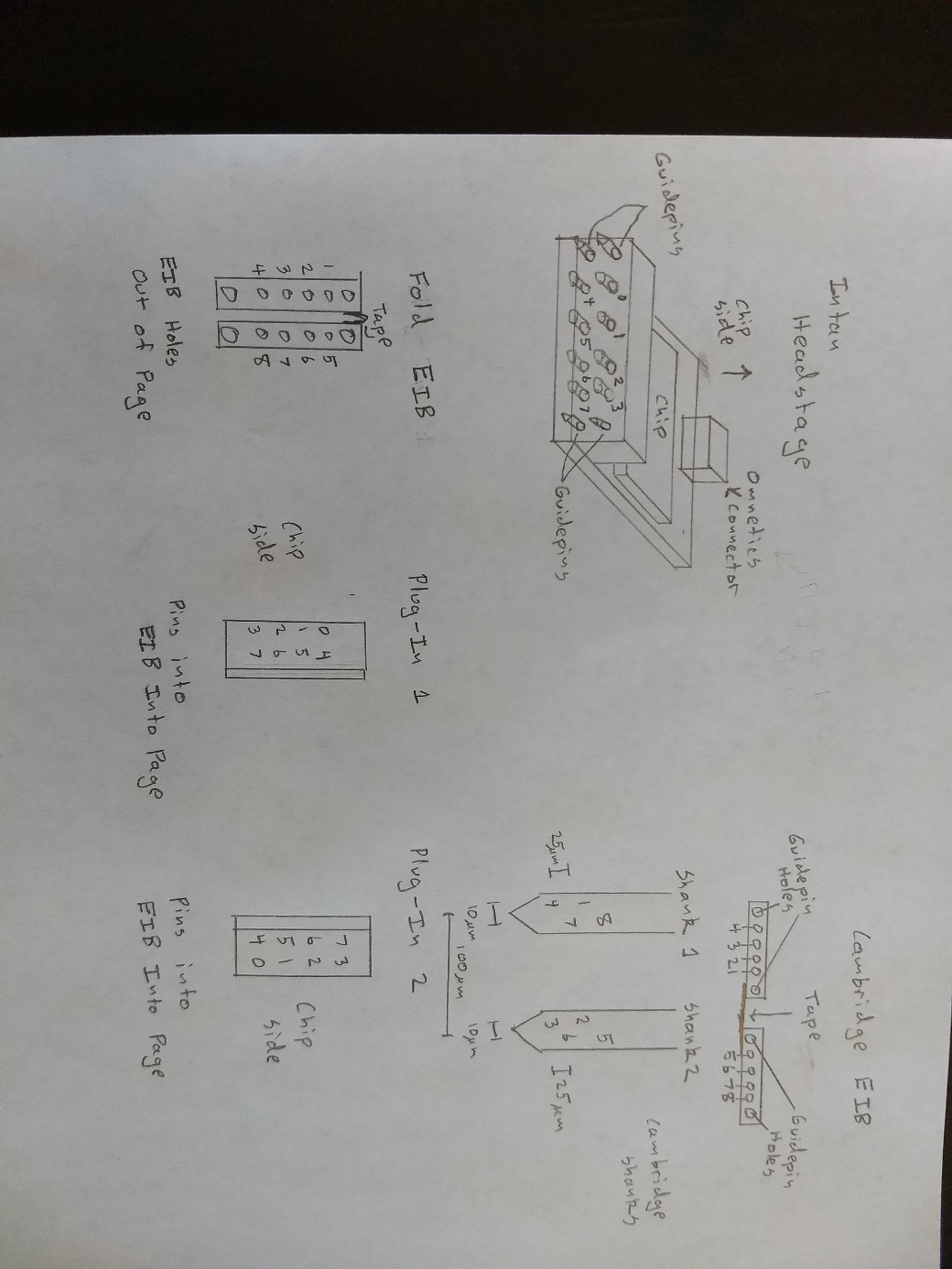
It is strongly recommended that you start with the basic siteMapCreate script and create a dedicated script (e.g., siteMapCreate\_ExpY\_AnimZ.mat) for each probe type, headstage orientation, and so forth. It is also recommended that you create a folder that holds all the site mapping files to be used later in the spike sorting process.

The script has multiple sections that must be updated for the user’s computer, the name for the site mappings, the notes about the mapping, the physical geometry of the probe, the translation between probe channel numbers and open ephys channel numbers, and the information about which shanks the channels are located on. In the process of creating the site map, there are two main numbering systems that you need to keep in mind. First, there is the probe number, which is the number of an electrode on the probe. This corresponds to the numbering provided by Cambridge for their probes (for example). Second, there is the open ephys number, which is the number of an electrode in the open ephys system. This corresponds to the numbers shown when displaying open ephys data and when saving it. We’ll walk through each of these variables below:

* siteMapDir: This is the full path name of the directory on your local machine where you wish to store the site map. We recommend making a dedicated directory for site mappings.
* siteMapName: This is the file name (not full path name) for the site map you are creating. We recommend creating a name that will help you identify the animal, experiment, etc. for which this site mapping will be used.
* siteMapNotes: This is a string that is saved with the site map that allows you to write further details to help you identify the site map.
* probeGeo: This is a number of electrode channels by two matrix that specifies the physical location of the electrodes in space using x and y coordinates. probeGeo(i,:) = [j,k] means that probe channel number I has an x coordinate of j and a y coordinate of k. Note that coordinates can be specified only within shanks or groups, or it can be specified universally across all shanks or groups (see probeChanShankID below).
* openE2probe: This is a number of electrode channels by one matrix that specifies the relationship between open ephys channels and probe channels. openE2probe(i) = j means that open ephys channel i corresponds to probe channel j. To determine this relationship, you will need to relate the channels on the Intan headstage you are using to how it was plugged in to the EIB on probe. Note that the open ephys channel number is identical to the channel number on the Intan headstage plus 1 (Intan headstage channels run 0-(n-1) and open ephys runs 1-n). To accurately create the openE2probe variable, you will need to know the orientation of the EIB on the animal as installed during surgery (e.g., fold side, omnetics label side). Also, you will need to know how the headstage was plugged in (e.g., chip side front, back, left, or right).
* probeChanShankID: This is a number of electrode channels by one matrix that specifies the shanks or groups for each probe channel. probeChanShankID(i) = j means that probe channel i is on shank (or in group) j. The channel mapping and spike sorting process allows for the electrodes to be broken up into different shanks. In reality, the term “shank” is probably not correct because there is nothing in the algorithm that requires electrodes on one “shank” to actually be on one and only one physical shank on the actual probe. In reality, the “shanks” are just groups. Thus, each shank will be spike sorted separately. If you wish, you can consider all the electrodes as being part of one “shank” by setting probeChanShankID to all be 1 and setting the geometry appropriately between all electrodes. Alternatively, you could break up electrodes on one physical shank into different “shanks”.

## Simple Example Channel Mapping

To help understand how to create channel mappings, we will walk through a simple example. Suppose we have a made-up probe with only 2 shanks, 4 electrodes per shank, and a made-up Intan headstage with 8 channels. Open ephys would find 8 channels (open ephys channels 1-8) and it would save 8 files 100\_CH1.continous through 100\_CH8.continuous. Suppose the Intan headstage and the probe EIB are numbered as shown below:



Several of the Cambridge probe EIBs are folded during surgery to create an EIB with two rows. (Note, not all probes are folded, so this issue may not apply to your data set, but plug in orientation will always affect the site mapping.) When connecting the EIB to the headstage, there are two orientations to plug it in. Based on the drawings above where the folded tape is assumed to be oriented upwards, the processor chip on the headstage can either be facing left (Plug-In 1) or right (Plug-In 2). (Note, in Plug-In 1 and Plug-In 2, the EIB plug on the headstage is oriented into the page and the holes in the EIB are opening out of the page. In other words, the EIB and headstage are oriented as if you are looking down on the animal plugging in the headstage.)

Each shank has 4 electrodes in two staggered columns. The columns are 10 um apart. The electrodes within the columns are 50 um apart with a staggering of 25 um. Assume the shanks are 100 um apart. Note that the numbers shown on the shank are probe channel numbers, not open ephys channel numbers.

The values of the variables in siteMapCreate will depend on two choices. First, are the shanks spike sorted separately or together. Depending on the selection, the probeGeo variable will have the following values (assuming the x direction is horizontal in the drawing and the y direction is vertical):

|  |  |  |
| --- | --- | --- |
|  | Sort Shanks Separately | Sort Shanks Together |
| probeGeo(1,:) | [0,50] | [0,50] |
| probeGeo(2,:) | [0,50] | [100,50] |
| probeGeo(3,:) | [0,0] | [100,0] |
| probeGeo(4,:) | [0,0] | [0,0] |
| probeGeo(5,:) | [10,75] | [110,75] |
| probeGeo(6,:) | [10,25] | [110,25] |
| probeGeo(7,:) | [10,25] | [10,25] |
| probeGeo(8,:) | [10,75] | [10,75] |

Depending on the selection, the probeChanShankID variable will have the following values:

|  |  |  |
| --- | --- | --- |
|  | Sort Shanks Separately | Sort Shanks Together |
| probeChanShankID(1) | 1 | 1 |
| probeChanShankID(2) | 2 | 1 |
| probeChanShankID(3) | 2 | 1 |
| probeChanShankID(4) | 1 | 1 |
| probeChanShankID(5) | 2 | 1 |
| probeChanShankID(6) | 2 | 1 |
| probeChanShankID(7) | 1 | 1 |
| probeChanShankID(8) | 1 | 1 |

In addition to sorting shanks together or separately, the plug-in orientation will affect the openE2probe variable. Depending on the orientation (Plug-In 1 or Plug-In 2), the openE2probe variable will have the following values (don’t forget that open ephys adds 1 to the channel number from the Intan headstage (numbers shown in drawing)):

|  |  |  |
| --- | --- | --- |
|  | Plug-In 1 (Fold Up, Chip Left) | Plug-In 2 (Fold Up, Chip Right) |
| openE2probe(1) | 1 | 8 |
| openE2probe(2) | 2 | 7 |
| openE2probe(3) | 3 | 6 |
| openE2probe(4) | 4 | 5 |
| openE2probe(5) | 5 | 4 |
| openE2probe(6) | 6 | 3 |
| openE2probe(7) | 7 | 2 |
| openE2probe(8) | 8 | 1 |

# Spike Sorting an Individual Data Set

This section addresses spike sorting an individual data set using an interactive job on Big Red 2. (For information about sorting multiple data sets simultaneously, see Section 8.) If you are new to spike sorting or this spike sorting routine, or if you are only interested in sorting a single data set once, we recommend starting here.

## Stage 1

Log into Big Red 2, start an interactive job (1 processor and at least 3 hours of walltime, gpu not required), and start Matlab (see Section 4.7). (Note, occasionally the transfer process is very slow and requires more than 3 hours. If this happens, request more time, delete the spike sorting directory for this data set on the Data Capacitor, and restart stage1.)

Edit the stage1 script in the interactiveJob directory with information about the data set to be spike sorted. This information includes where to save the data on the data capacitor (Settings Section) and information about the data set (Data Set Information Section).

In MATLAB on Big Red 2 in the interactive job, go to the interactiveJob directory and execute **stage1**. The software should prompt you for your IU Box password (see Section 4.2), report progress as the data is moved from IU Box to the data capacitor, and report progress in generating the stage 1 report on the data. For a 64 channel recording sampled at 30 kHz for an hour, this usually takes about 1.5 hours to run.

When the stage1 script is finished, it will display the full filename for the report file to be reviewed by the user. Transfer this file from the Data Capacitor to your local machine for review. We would suggest labeling the destination directory something like Stage1Results and only putting these types of files in that directory, but this isn’t strictly necessary. On your local machine, review the stage 1 report. See Section 10 for more details on stage 1 review.

## Stage 2

Log into Big Red 2, start an interactive job (at least 8 processors and at least 6 hours of walltime, gpu required), and start Matlab (see Section 4.7).

Edit the mainDC variable in the stage2 script in the interactiveJob directory so it matches the directory name used for stage 1.

In MATLAB on Big Red 2 in the interactive job, go to the interactiveJob directory and execute **stage2**. The software should report progress throughout the spike sorting process. If the data has been broken down into different shanks, each shank will be spike sorted individually. For a 64 channel recording sampled at 30 kHz for an hour, this will take about 1.5 hours to run, but it can take longer if many neurons are found.

When the stage2 script is finished, it will display the full filename for the report file to be reviewed by the user. Transfer this file from the Data Capacitor to your local machine for review. We would suggest labeling the destination directory something like Stage2Results and only putting these types of files in that directory, but this isn’t strictly necessary. On your local machine, review the stage 2 report. See Section 11 for more details.

## Stage 3

Log into Big Red 2, start an interactive job (1 processor and at least 3 hours of walltime, gpu not necessary), and start Matlab (see Section 4.7). (Note, occasionally the transfer process is very slow and requires more than 3 hours. If this happens, request more time, delete the spike sorting directory for this data set on the Data Capacitor, and restart stage3.)

Edit the mainDC variable in the stage3 script in the interactiveJob directory so it matches the directory name used for stage 1.

In MATLAB on Big Red 2 in the interactive job, go to the interactiveJob directory and execute **stage3**. The software should report progress as it organizes the data and transfers the data from the Data Capacitor to IU Box. For a 64 channel recording sampled at 30 kHz for an hour, this usually takes about 0.5 hours to run.

When the stage1 script is finished, all of the final spike sorted data (see Section 9) and the various files created throughout the spike sorting process will be located in a new directory on IU Box. This directory will be called SpikeSorting-DataSetName, where DataSetName is the name you set for this data set in stage 1.

# Spike Sorting Multiple Data Sets in Parallel

The spike sorting process described in Section 7 can be parallelized to process multiple data sets simultaneously. The general process is very similar with the analysis proceeding through stages 1, 2, and 3 with report reviewing (see Sections 10 and 11) after stages 1 and 2. However, instead of setting information in the stage1, stage2, and stage3 scripts in the interactiveJob directory, this information is set in stage1JobsToDo, stage2JobsToDo, and stage3JobsToDo in the parallelJobs directory. Note that multiple data sets can be listed in the Data Set Information variables in the stage1JobsToDo script. The only changes necessary for stage2JobsToDo and stage3JobsToDo are the mainDC scripts (similar to stage2 and stage3 in the interactiveJobs).

In general, for each stage, once the necessary edits have been made to the JobsToDo script, start an interactive job (1 processor and at least 1 hour of walltime, gpu not necessary), and start Matlab (see Section 4.7). Then, run the JobsToDo script for that stage (e.g., **stage1JobsToDo** in parallelJobs). This will create a special text file in your main Big Red 2 directory. For instance, stage1JobsToDo creates the file spikeSortStage1JobList.txt file that serves as a bash script to start all of the jobs.

Next, exit MATLAB and execute the following commands (where X is the stage number (1, 2, or 3)):

**chmod u+x spikeSortStageXJobList.txt**

**dos2unix spikeSortStageXJobList.txt**

**./spikeSortStageXJobList.txt**

This sequence of commands will submit multiple jobs to Big Red 2 to run simultaneously on the cluster. The number of jobs will be equal to the number of unique data set and parameters you input in the stage1JobsToDo script. How long it will take these jobs to run is dependent upon the current usage of Big Red 2. Stage 2 jobs require more resources, so they tend to take longer to run. You will receive emails for the cluster when jobs start and end. Also, text files will appear in your main Big Red 2 directory that record the outputs from the jobs (what you would normally see displayed on the command line in an interactive job) and any errors.

The resources requested for each job can be changed in the job scripts which you copied into your main Big Red 2 directory during installation (spikeSortStageXJobVer1.txt, where X is the stage number (1, 2, or 3)).

Remember that reports must be reviewed after stages 1 and 2, just like with an interactive job. Once stage 3 has finished running, the spike sorted data will have been transferred to the IU Box folder for each data set (see Section 9). If an individual data set has been spike sorted multiple times, multiple folders will appear with the unique data set label and a prefix number.

# Output File Information

After running the stage 3 jobs, all of the various spike sorting files from the Data Capacitor will be transported to the IU Box folder that holds the raw open ephys data. The folder will be called “SpikeSorting-DataSetID”. There are many files produced by the spike sorting algorithm, many of which are best understood by finding them in the code. Particularly important files are described below:

1. spkData.mat: A number of neurons by 1 cell array. spkData{x} contains the spike times in milliseconds for neuron x.
2. spkDataNeurInd.mat: A number of neurons 1 cell array. spkDataNeurInd{x} contains a list of the neurons from the raw spike sorting (see the stage 2 review) that were merged into this neuron to create neuron x in spkData.
3. xy.mat: Contains animal tracking information. xdata and ydata are vectors containing the x and y tracking positions as measured by voltage. timestamps provides the time in milliseconds for the corresponding elements of xdata and ydata.
4. licks.mat: Contains voltage traces from microphones on the sippers. ldata and rdata contain the voltage traces for the left and right sipper (respectively). timestamps provides the time in milliseconds for the corresponding elements of ldata and rdata. In some recordings, these voltage traces are not meaningful (e.g., delay discounting, recordings prior to installation of microphones on sippers).
5. maEvents.mat: Contains the med associates events in maEvents and the timestamps for the med associates events in maTimestamps. The precise meaning of these events is task and data set specific.

# Stage 1 Review GUI

On your local machine in matlab, startup the stage 1 review GUI (graphical user interface) by double clicking on stage1Reviewer in the reviewGUIs directory (or entering it into your command line). This GUI allows you to quickly and easily reject and accept individual electrode channels from the raw recording.



Start by clicking Load Report and finding the stage 1 pre-review report you wish to view on your local machine.

Go through each channel using the Previous Channel and Next Channel buttons. Each channel will automatically be marked as Accept unless you change it to Reject in the Channel Ruling section.

Three segments of data and their associated power spectra will be displayed for each channel. In addition, the standard deviation of the whole recording signal will also be displayed. Use this information to reject noisy channels. Any record of noisy channels gathered during the electrophysiology recording can assist you in this task. Note that the channel numbers displayed here are identical to those in open ephys.

The y-axis of the voltage and power plots can be controlled via the fields on the middle right.

At any point in the review process, load the site mapping file for this recording (see Section 6 on how to create this file) by clicking on Load Channel Map. When you do so, a dialog box will appear with the description for that site mapping that you created. Review this information to ensure you are using the right site mapping for this data set.

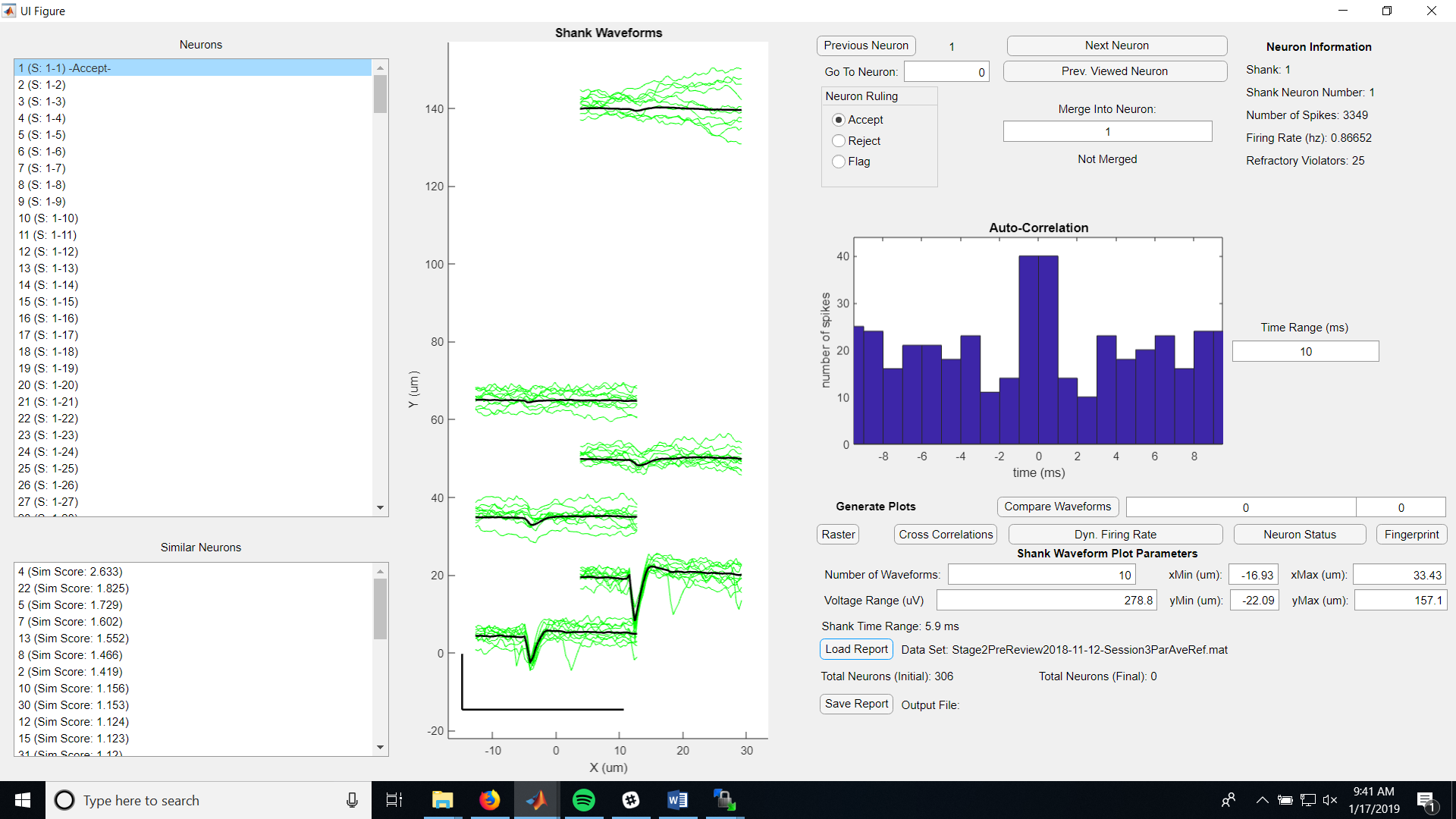
Once you have reviewed all the channels and loaded the appropriate site mapping, click on the Save Results button. This will create a file very similar to the file you loaded, expect it will say “PostReview” instead of “PreReview” indicating that it is the results of your review of the data.

Tips and Tricks: if you are spike sorting the same data set multiple times with different spike sorting parameters, you can avoid repeating the stage 1 review process by doing it once, then copying the post review file and changing its name. To do so, click Quick Duplicate and then select the pre-review report that should have a post-review report that is identical to the review you just performed. For instance, if you just reviewed prereviewFile1 and saved the results, but prereviewFile2 should have the same results, click Quick Duplicate and select prereviewFile2. postreviewFile2 will appear.

Once you have reviewed your data set, copy the postreview file to the Stage1ResultsPostReview directory on the data capacitor in the main spike sorting directory for this analysis. For instance, if the prereview file was in /N/dc2/scratch/username/spikeSortTest1/Stage1ResultsPreReview, the postreview file should go in /N/dc2/scratch/username/spikeSortTest1/Stage1ResultsPostReview.

# Stage 2 Review GUI

On your local machine in matlab, startup the stage 2 review GUI (graphical user interface) by double clicking on stage2Reviewer in the reviewGUIs directory (or entering it into your command line). This GUI allows you to quickly and easily accept and reject candidate neurons, as well as to merge neurons that have been incorrectly split.



Start by clicking Load Report and finding the stage 2 pre-review report you wish to view on your local machine.

Go through each neuron using the Previous Neuron and Next Neuron buttons. Each neuron will automatically be marked as Accept unless you change it to Reject or Flag in the Neuron Ruling section. You can also jump to a certain neuron via the Neurons menu in the upper left or switch back to the previously viewed neuron by clicking the Prev. Viewed Neuron button.

When a specific neuron is brought up in the GUI, example waveforms on the shank electrode locations will be shown in the middle plot. The number of waveforms can be controlled with the Number of Waveforms field. The more waveforms displayed, the slower the GUI updates the plots. The voltage range and time range are displayed beside the figure and correspond to the black bars in the lower left corner. The voltage range can be controlled with the Voltage Range (uV) field. The displayed x and y ranges on the shanks can be controlled with their corresponding fields.

In addition to the waveform, the GUI also displays the autocorrelation and numerous statistics about the neurons. Good neurons tend to have relatively few refractory period violators (low number of spikes between -1 and 1 ms in autocorrelation) and reasonable firing rates (less than 10 Hz).

Identifying neurons that have been improperly split is aided via the list of neurons in the lower right corner. These are other candidate neurons from the same shank ranked in descending order of average waveform similarity. Selecting another neuron on this similar neuron list will produce a plot of the cross correlation between the two neurons, as well as a plot of the average waveforms for direct comparison. A neuron can be merged into another neuron using the merge field. Simply put the number of the neuron that should receive the currently displayed neuron in the field and press enter. The text below the field should change to red and say “Merged”. Currently, if 3 or more neurons are to be merged, all neurons must point to one receiver (1 merged in 3 and 2 merged into 3 is good, 1 merger into 2 and 2 merged into 3 is not good). This will be made more flexible in future updates.

An example raster plot of all neurons with shank color coding can be produced by pushing the raster button.

Cross correlations (really spike triggered averages) for all other neurons in relation to the currently displayed neuron can be displayed by pushing the cross correlation button.

The firing rate in 1 minute segments for the displayed neuron throughout the recording can be displayed by pressing the dynamic firing rate button.

The status (unreviewed, accepted, rejected, or flagged, as well as merge status) for all neurons can be displayed by pressing the neuron status button.

Once you have reviewed all the neurons, click on the Save Report button. This will create a file very similar to the file you loaded, expect it will say “PostReview” instead of “PreReview” indicating that it is the results of your review of the data. Once you have reviewed all your data sets for this analysis, copy them to the Stage2ResultsPostReview directory on the data capacitor in the main spike sorting directory for this analysis.

Admittedly, the process of thoroughly reviewing a single data set can be time consuming. Nick spike sorted a data set from a 6 shank, 64 channel (only about 50 working) probe that found 304 neurons and it took him about 30 minutes to go through it. In the future, we hope to speed this up by using machine learning to perform an initial accept/reject assignment and to possibly guide merging.

# Kilosort File Change Log

The following kilosort files have been changed:

convertOpenEphysToRawBInary.m: This function has been extensively modified to fix path issues, to allow for the correct open ephys mapping, and to allow for referencing.

master\_eMouse.m: This script has been edited to reflect path structure on Big Red 2 for lapishLabCluster. Also, minor changes to file separators to make certain lines operating system independent.

# Common Errors/Problems

## FTP Transfer Fails

If the stage 1 or 3 code runs, but displays that the files can’t be transferred, this may be due to the fixed SSL libraries in stage1Core and stage3Core. These libraries were fixed in January 2019, but they will be updated in the future, which might cause the transfer to break. Please adjust the libraries or ask High Performance computing for help if this is the only remaining source of a failed FTP transfer. First, make sure your username and password are correct for IU Box. A key indicator that the SSL libraries are the source of the problem are if the bash script created by the stage1Core or stage3Core can be executed outside MATLAB correctly, but only fail when executed in MATLAB via the system command.

# Tips for Working on Big Red 2

1. Under construction…

# Using Phy to Visualize Data

1. As suggested by kilosort, it is possible to install phy to visualize the results. However, this is not necessary as we have created our own GUIs to review results. In addition, using phy requires many additional steps and, as of this writing, phy is not completely functional. However, if you wish to pursue this course, here are several notes.
   1. Because phy will produce a GUI, you will need to use the Research Desktop (https://kb.iu.edu/d/apum). So, you must set this up.
   2. Follow the steps on the phy github page (https://github.com/kwikteam/phy), installing miniconda (see dedicated section below) first.
   3. In addition to installing phy and phycontrib, you will also need to install klusta.
   4. When the GUI is called, try the following commands to correct a runtime error related to python 3 and ASCII coding:
      1. export LC\_ALL=en\_US.utf-8
      2. export LANG=en\_US.utf-8
   5. Nick had the most success running all of this on Carbonate, not Big Red 2. However, all three clusters (Karst, Carbonate, and Big Red 2) can be accessed via the Research Desktop, so it should be possible to do this in Big Red 2.

## Installing Miniconda

As mentioned in Section 15, to install Phy, you must first install miniconda. Directions on how to do so are shown below.

1. SSH into carbonate.
2. Input (i.e., copy what comes next into the Carbonate command line and hit enter): wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O miniconda3.sh
3. Input: bash miniconda3.sh -bp ~/conda
4. Input: unset PYTHONPATH
5. Input: echo ". ~/conda/etc/profile.d/conda.sh" >> ~/.bashrc
6. Close your terminal or SSH client and reconnect to Carbonate.
7. Input (NOTE, this is frequently needed to prevent errors. If you run into an unforeseen error trying to start up conda, there is a good chance you need to do this.): unset PYTHONPATH
8. Input: conda activate base
9. Input: conda update conda
10. It will prompt you to say whether you want to proceed with the update. Input: y
11. Input (this command exits an environment in conda and can be used generally): source deactivate