

Revision: 01
24 March 2014

# Electrical Recording Code Library (Urkel)

(Working Title)

Kevin J. O'Neill III
University of Utah, Salt Lake City, 84112

kjon.oneill@gmail.com



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

NeuraLynx Code Manual

Revision: 01
24 March 2014





This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# I. Introduction and Programming Standards

## a. Purpose of Manual

The purpose of this manual is meant to guide and teach a user how to utilize a code library created to work with the NeuraLynx data acquisition system.

## b. Programming Standards

This code set utilizes a programming standard in order to clarify what type of data is contained in variables and to distinguish between variables and functions. The programming style is as follows:

**Naming Conventions:**

All variable names are descriptive. Temporary variables are labeled by having a prefix temp. You will not see single letter variable names unless the variable is a looping control or is used in the construction of patch plots. There are very few two letter variables, most of these are described in I.c

| Item | Format | Example | Exceptions |
|---|---|---|---|
| **Function** | FirstLetterCaps() | FindLaserEvents() | ncs2mat(), nse2mat(), c2e() |
| **Variable** | firstWordLower | timeBounds | - |
| **Structure** | FirstLetterCaps | CSCData, SpikeData | - |
| **Object** | m_myObject | m_errData | - |
| **Looping Control** | i j k l | i j k l | chan |

**White space standards:**

This code utilizes several white space standards in order to provide clarity and make the code easier to read for the user.

| Item | Format |
|---|---|
| **Variable Assignments** | A = B; |
| **Section Segregation** | |
| | |

## c. Explanation of Common Variables



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

| Variable | Data Type | Use | Functions |
|---|---|---|---|
| CSCData | Structure | Structure containing information about the continuously sampled data. Saved as a .mat. | Most function (raw data) |
| SpikeData | Structure | Structure containing information about the spike event data. Saved as a .mat. | Many functions (spike data) |
| overwrite | Boolean | Boolean flag to overwrite the previously saved CSCData and SpikeData .mat files | nse2mat()<br>ncs2mat() |
| saveData | Boolean | Boolean | |
| channels | Double Vector | This vector is used to determine which channel(s) that the code will analyze. Some functions will only accept one channel, others will use all 16/17 as default. | Many functions |
| timeBounds | Double [1x2] vector | | Many functions |
| events | Double [nx1] vector | | |

### d. Dependencies

This code set has one primary dependence. In order to use this code library, please download the Chronux library (http://www.chronux.org/). The spectral and PSTH plots use this library.

In order to correctly use this library, it must be added to the **bottom** of the MATLAB path. To do this use the following code **(Please fill in the correct path to the Chronux library)**:

```
addpath(genpath('C:\...\...\...\chronux'),'-end');
```

The other dependence is the NeurLynx code library. This library is used to extract data from the .ncs and .nse files. Please download the code library from NeuraLynx ('Matlab Import and Export MEX files' from:
http://neuralynx.com/research_software/file_converters_and_utilities).

For the PrarieView functions, the PrarieView library must be downloaded. Specifically the file required is PrarieView_Tiff.m function as it is the only one used. This dependency has several assumptions. The first and foremost of which is that there are an equal number of images for each part of the protocol.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### e. Code Expectations

Consistent units

Good documentation of experiments (mouse, stimulation type, etc).

Do not use more than one stimulus trial type unless well documented. Neuralynx may not save the markers for different trials, therefore the code does not know about them.

etc

### f. Brief Math Explanations

| Desired Value | Equation | Functions |
|---|---|---|
| Signal-to-Noise Ratio (SNR) | $SNR = \frac{pk2pk}{2 \times \sigma(noise)}$<br><br>SNR is equal to the peak-to-peak value of a spike over the term (2 times the standard deviation of noise) | MouseSNR() |
| Standard Error (stdErr) | $stdErr = \frac{\sigma(x)}{\sqrt{n}}$<br><br>Standard error is equal to the standard deviation of your data ($x$) over the square root of the number of samples ($n$) | Many Functions |
| Correlation Coefficients (CorrCoef) | $R(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)\,C(j,j)}}\ where\ C = cov(X)$ | MouseCorrCoef() |



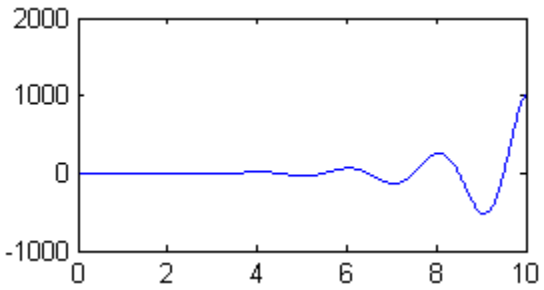
This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

## g. Plotting Tips and Tricks

**Lines**

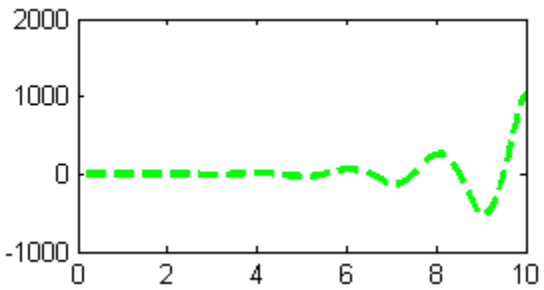
Whenever you plot a line you can capture its handle in order to change its properties.

```
lData = plot(x, y); % Plots y versus x
```




Changing the line properties are accomplished using the set() command:

```
set(lData, 'Color', 'g');        % Changes the color of the line
set(lData, 'LineStyle', '--');   % Changes the style of the line
set(lData, 'LineWidth', 2.75);   % Adjusts the line thickness
```



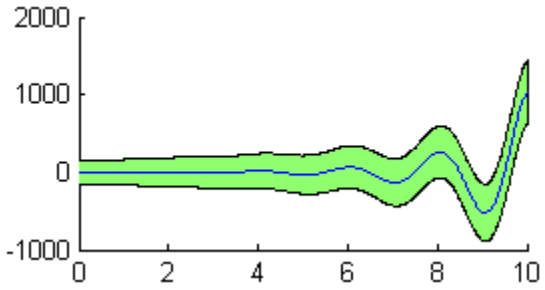



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Patch**

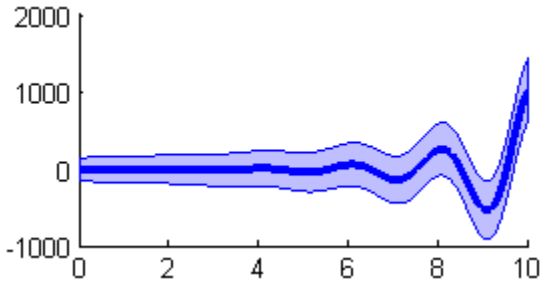
Patches provide a way to display additional information around a line plot. Usually the line plot is the mean and the patch is twice the standard error. Patched must be constructed such that you plot the boundary of a polygon. It is best to plot the line over the patch.

```
xx = [x, fliplr(x)];     % x-values for each of the y-values
yy = [[y + 2*stdErr], fliplr([y - 2*stdErr])]; % y-values form an n-gon

hold on
pData = patch(xx, yy, 1); % Plot patch
lData = plot(x,y);        % Plot line over patch
hold off
```




After plotting the patch and line, you can use the captured handles to control the figure.

```
set(pData, 'FaceColor', 'b');    % Changes the color of the patch
set(pData, 'EdgeColor', 'b');    % Changes the patch edge color
set(pData, 'FaceAlpha', 0.25);   % Changes the patch transparency

set(lData, 'Color', 'b');        % Changes the color of the line
set(lData, 'LineStyle', '-');    % Changes the style of the line
set(lData, 'LineWidth', 2.75);   % Adjusts the line thickness
```

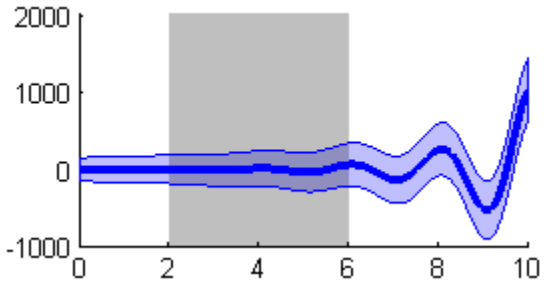





This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

You can also use patches to signify when a trial happens. To do this place a patch in the background.

```
hold on
% Background patch
backData = patch([2, 6, 6, 2], [2000, 2000, -1000, -1000], 1);
pData = patch(xx, yy, 1); % Plot patch
lData = plot(x,y);        % Plot line over patch
hold off

set(backData, 'FaceColor', 'k');   % Changes the color of the background
set(backData, 'EdgeColor', 'none');% Changes the background edge color
set(backData, 'FaceAlpha', 0.25);  % Changes the background transparency

set(pData, 'FaceColor', 'b');   % Changes the color of the patch
set(pData, 'EdgeColor', 'b');   % Changes the patch edge color
set(pData, 'FaceAlpha', 0.25);  % Changes the patch transparency

set(lData, 'Color', 'b');       % Changes the color of the line
set(lData, 'LineStyle', '-');   % Changes the style of the line
set(lData, 'LineWidth', 2.75);  % Adjusts the line thickness
```






This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Plot Formatting:**

```
title(sprintf('Electrode: %d', electrodes(k)), 'FontSize', 15)

ylabel('Firing Rate, Hz', 'FontSize', 15)
xlabel('Time, sec', 'FontSize', 15)

set(gcf, 'Units', 'Inches')
set(gcf, 'Position',[2 2 6 3])
% set(gcf, 'PaperUnits', 'FontSize', 'PaperPosition',[2 2 6 3])

aPos = get(gca, 'Position');

p1 = get(gca, 'position');
legend({'Hello', 'World'}, 'Location', [0.79, 0.8, 0.125, 0.0625])
legend({'Hello', 'World'}, 'Location', 'NorthEast')
legend boxoff


set(gca, 'yTick', [0, 10, 20, 30, 40])
set(gca, 'yTickLabel', [0, 10, 20, 30, 40])
set(gca, 'xTick', [0, 1, 2, 3])
set(gca, 'xTickLabel', [0, 1, 2, 3])


grid on
set(gca,'XGrid','off','YGrid','on','ZGrid','off')


set(gca,'FontSize',15)

h_xlabel = get(gca,'xLabel');
set(h_xlabel,'FontSize',15);

h_ylabel = get(gca,'yLabel');
set(h_xlabel,'FontSize',15);

print('-dpng', ['MyImage_', num2str(k), '.png'], '-r100');
saveas(gcf,['MyImage ', num2str(k), '.png']);
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# II. First Phase Code – Data Extraction

## a. ncs2mat()

ncs2mat() is used to extract data from the NeuraLynx continuously sampled data.

```
function [] = ncs2mat( override )
```

**Inputs**

In order to use ncs2mat() please use the following inputs

| Variable | Data Type | Required | Use |
|---|---|---|---|
| override | Boolean | yes | Used to overwrite the already saved information. This is to make sure that the user does not waste time re-extracting data. |

**Outputs**

| Variable | Data Type | Use |
|---|---|---|
| CSCData | Structure | This file contains the header information as well as timestamps for the .ncs data files. The file naming convention is "expYYYY-MM-DD_HH-MM-SS_CSC.mat" |
| HDF5 samples file | HDF5 | This is a scientific file type used to store the channel voltages for later use by other functions. The file naming convention is "expYYYY-MM-DD_HH-MM-SS_CSC.h5" |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Select Data File (Line 69)

The code below prompts the user to select one file of type .ncs. The .ncs files are the continuously sampled data from the NeuraLynx data acquisition software and equipment.

```
% Prompts the user to select *ONE* file.
[fileName, pathName] = uigetfile('*.ncs', 'Select a CSC (.ncs) file');
```

Memory Restriction (Lines 212-225)

The code below looks at the available RAM memory to MATLAB and calculates how much data is can extract from the data files without causing an 'Out of memory' error.

```
% Determining system memory to maximize data segments
[userview, ~] = memory;
maxBytes = userview.MaxPossibleArrayBytes;


% Calculating maximum channel samples to load into memory
segmentLength = maxBytes*0.25 / 8 / 512; % 8 bytes is the length of a double
% segmentLength = 512*1024/512;
dataLen = length(Header.timeStamps);

segNum = floor(dataLen/segmentLength) + 1;
segRemLen = rem(dataLen,segmentLength);

dataSize = dataLen*512*8*numel(fileList)/1024/1024/1024;
```

Saving Data to File (Line 270)

In the following code MATLAB saves the processed data to an HDF5 file type. This is a scientific file type that allows the parsing of terabytes of data. We are using it due to 32-bit MATLAB's maximum array size being ~600 megabytes.

```
h5write(fullCSCMAT, ['/', tempChanName], tempChan,...
        [1, (j-1)*segmentLength + 1], [1, size(tempChan,2)]);
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. nse2mat()

nse2mat() is used to extract data from the NeuraLynx spike event.

```
function [] = nse2mat( override )
```

**Inputs**

In order to use nse2mat() please use the following inputs

| Variable | Data Type | Required | Use |
|---|---|---|---|
| override | Boolean | yes | Used to overwrite the already saved information. This is to make sure that the user does not waste time re-extracting data. |

**Outputs**

| Variable | Data Type | Use |
|---|---|---|
| SpikeData | Structure | This file contains the header information, timestamps, and voltages for the identified spikes from the .nse data files. The file naming convention is “expYYYY-MM-DD_HH-MM-SS_SE.mat” |



This document and the information contained herein are proprietary to Kevin J. O’Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O’Neill III.

**Important Landmarks:**

Select Data File (Line 43)

The code below prompts the user to select one file of type .nse. The .nse files are the spike event data from the NeuraLynx data acquisition software and equipment.

```
% Prompts the user to select *ONE* file.
[fileName, pathName] = uigetfile('*.nse', 'Select a SE (.nse) file');
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### c. DownFilter()

DownFilter() is used to downsample and/or filter the data

```
function [] = nse2mat( override )
```

**Inputs**

In order to use nse2mat() please use the following inputs

| Variable | Data Type | Required | Use |
|---|---|---|---|
| override | Boolean | yes | Used to overwrite the already saved information. This is to make sure that the user does not waste time re-extracting data. |
| CSCData | Structure | yes | Data file generated from ncs2mat(). |
| dsFlag | Boolean | yes | Determines whether the samples are down sampled. |
| dsFs | Double/Integer | yes | The sampling frequency for the down sampled data. The maximum frequency after down sampling is 1/2 dsFs. |
| filtFlag | Boolean | yes | Determines whether the down sampled data is then filtered. Or if dsFlag is false, filters the given data. |
| filtType | String | yes | 'highpass' or 'lowpass' are the only inputs. These allow the user to control the filtering object. |
| filtFs | Double/Integer | yes | The filter frequency used in the filters. |

**Outputs**

| Variable | Data Type | Use |
|---|---|---|
| CSCData | Structure | This file contains the header information, timestamps, and voltages for the .ncs files. |
| HDF5 Data File | HDF5 | This scientific file structure contains all of the samples for each channel. |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Select Data File (Line 113)

The following code is an example for building a lowpass elliptical filter.

```
N = 10;          % order
Fpass = dsFs/4; % Passband frequency
Apass = 1;       % Passband ripple (dB)
Astop = 80;      % Stopband attenuation (dB)
h = fdesign.lowpass('N,Fp,Ap,Ast',N,Fpass,Apass,Astop,CSCData.freq);
Hd = design(h,'ellip');
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# III. Second Phase Code – Find Events

## a. FindLaserData()

Find laser events provides the timestamps for when the laser turns off and on.

```
function [trialEvents, laserEvents ] = FindLaserEvents( CSCData, varargin )
```

**Inputs**

In order to use FindLaserEvents()

| Variable | Data Type | Required | Use |
|---|---|---|---|
| CSCData | Structure | yes | Data file generated from ncs2mat(). |
| threshold | Double | No | This value is used to determine when the laser is turned on. If not given to the function, the default value is: $0.5 \times \max(laserData)$. This assumption works fine for squarewave inputs. But for exponential or graded inputs please provide a known cutoff value. |

**Outputs**

This function outputs a variable. This function will also be modified to look for trials of repeated stimulation pulses. This will be done by looking at the difference between timestamps.

| Variable | Data Type | Use |
|---|---|---|
| TrialEvents | Double [nx2] matrix | This two column matrix contains the trial on-off event timestamps in a pair-wise fashion. |
| laserEvents | Double [nx2] matrix | This two column matrix contains the laser on-off event timestamps in a pair-wise fashion. Most of the time you will be using only the first (on) column. |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Load data (Line 28)
Loads data from the CSCData structure given.

```
% Loads timestamps
laserTime = CSCData.timeStamps;

% Loads laserData from HDF5 file (channel 17).
tempChanName = ['CSC',num2str(17)];
laserData = h5read(CSCData.hdf5FileName, ['/', tempChanName],...
    1, length(laserTime));
```

Find Events (Line 42)
Loads data from the CSCData structure given.

```
% Find events
laserData = laserData(:)'; % Unwrap data
laserIdx = laserData > threshold; % Convert to square wave
laserIdx = xor(laserIdx, [0, laserIdx(1:end-1)]); % Find transitions
```

Find TimeStamps (Line 50)
Finds the timestamps when the on/off values occur.

```
% Find TimeStamps
laserEvents = laserTime(laserIdx);

% Reshape to [nx2] matrix
laserEvents = reshape(laserEvents,length(laserEvents)/2, 2);
```

Find Trials (Line 56)
Finds the timestamps when the trial on/off values occur.

```
trialIdx = find(diff(laserEvents(:,1))>1e6); % Find gaps of around 1 second

trialIdxStart = [1, trialIdx+1]; % Shift index to start of trial.
trialIdxEnd = [trialIdx, size(laserEvents, 1)]; % Shift index to end of
trial.

% Creates the start and end pairs for trials.
trialEvents = [laserEvents(trialIdxStart, 1), laserEvents(trialIdxEnd, 2)];
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**b. FindMarkerEvents()**

Function does not exist as I do not know how NeuraLynx records trial markers, if it even does. **DO NOT USE MORE THAN ONE TRIAL TYPE DURING A RECORDING SESSION UNLESS YOU DOCUMENT IT WELL.**



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**c. c2e()**

Converts electrode numbers to channel numbers and vice versa. Also provides a special map for where the electrodes placed. Needs information from Sandeep to function.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# IV. Third Phase Code – Raw Data

## a. TrialRaw()

Plots raw trial data

```
function [] = TrialRaw( CSCData, saveData, channel, stimOn, stimTimes, ...
    stimColor, timeBounds, events)
```

**Inputs**

In order to use TralRaw() please follow the table below.

| Variable | Data Type | Required | Use |
|---|---|---|---|
| CSCData | Structure | Yes | Data file generated from ncs2mat(). |
| saveData | Boolean | Yes | |
| channel | Double [1xn] vector | Yes | |
| stimOn | Boolean | Yes | |
| stimTimes | Double [nx2] matrix | Yes but may be empty | |
| stimColor | String | | *Needs work* |
| timeBounds | Double [1x2] | Yes | |
| events | Double [nx2] | Yes | |

**Outputs**

This function outputs figues in several save formats.

| Variable | Data Type | Use |
|---|---|---|
| laserEvents | Double [nx2] matrix | This two column matrix contains the laser on-off event timestamps in a pair-wise fashion. Most of the time you will be using only the first (on) column. |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Load data (Line 28)

Loads data from the CSCData structure given.

```
% Loads timestamps
laserTime = CSCData.timeStamps;

% Loads laserData from HDF5 file (channel 17).
tempChanName = ['CSC',num2str(17)];
laserData = h5read(CSCData.hdf5FileName, ['/', tempChanName],...
    1, length(laserTime));
```

Find Events (Line 42)

Loads data from the CSCData structure given.

```
% Find events
laserData = laserData(:)'; % Unwrap data
laserIdx = laserData > threshold; % Convert to square wave
laserIdx = xor(laserIdx, [0, laserIdx(1:end-1)]); % Find transitions
```

Find TimeStamps (Line 50)

Finds the timestamps when the on/off values occur.

```
% Find TimeStamps
laserEvents = laserTime(laserIdx);

% Reshape to [nx2] matrix
laserEvents = reshape(laserEvents,length(laserEvents)/2, 2);
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. MouseSpectrum()

Code is complete but needs testing before initial publishing.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### c. MouseSpectrogram()

Code is complete but needs testing before initial publishing.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# V. Third Phase Code – Spike Data

## a. SpikeWaveform()

Plots raw spike waveform

```
function [] = SpikeWaveform( SpikeData, saveData, channel, unit)
```

**Inputs**

In order to use SpikeWaveform() please follow the table below.

| Variable | Data Type | Required | Use |
|---|---|---|---|
| SpikeData | Structure | Yes | Data file generated from nse2mat(). |
| saveData | Boolean | Yes | |
| channel | Double [1xn] vector | Yes | • Needs work* |
| stimOn | Boolean | Yes | |
| Unit | Double | Yes | |

**Outputs**

This function outputs figures in several save formats.

| Options | Data Type | Use |
|---|---|---|
| Spike patch | figure | Plots the spike waveform as a mean and patch (+- 2 times standard error) |
| Spike overlay | Figure | Overlays the spikes atop each other. |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Find mean and std error (Line 32)

```
meanWave  = mean(SpikeData.samples{channel}(:,:), 2);
errorWave = std(SpikeData.samples{channel}(:,:), 0, 2);
errorWave = errorWave/sqrt(numel(SpikeData.samples{channel}));
% Standard error
errorWave = 2*errorWave; % 2*standard error.
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. MouseSNR()

Computes the signal-to-noise ratio for given channels

```
function [sigNoise] = MouseSNR( CSCData, SpikeData, channel, noiseTime )
```

**Inputs**

In order to use SpikeWaveform() please follow the table below.

| Variable | Data Type | Required | Use |
|---|---|---|---|
| CSCData | Structure | Yes | Data file generated from nsc2mat(). |
| SpikeData | Boolean | Yes | Data file generated from nse2mat(). |
| channel | Double [1xn] vector | Yes | |
| noiseTime | Double [1x2] vector | Yes | This variable contains the start and end values for a baseline measurement. |

**Outputs**

This function outputs figures in several save formats.

| Options | Data Type | Use |
|---|---|---|
| sigNoise | Double [nx2] matrix | Contains the SNR and standard error for each channel, with each row containing the information for the given channels. |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Calculate the SNR for each channel (Line 34)

```
    for j = 1:numel(SpikeData.samples{chan})

        voltData = SpikeData.samples{chan,j};
        pk2pk = max(voltData) - min(voltData);

        tempChanName = ['CSC',num2str(chan)];
        noiseData = h5read(CSCData.hdf5FileName, ['/', tempChanName],...
                           noiseTime(1), noiseTime(1)-noiseTime(2));

        SNRTemp(j) = pk2pk / 2*stdev(noiseData);

    end % END FOR
```



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### c. Rastergram()

Code is complete but needs testing before initial publishing.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### d. RasterComp()

Code is complete but needs testing before initial publishing.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### e. MouseCorrCoef()

Code is complete but needs testing before initial publishing.

Computes the signal-to-noise ratio for given channels

```
function [ R ] = MouseCorrCoef( CSCData, saveData, stimData, timeBounds,
events, nameMod, calcOption, plotOption)
```

**Inputs**

In order to use SpikeWaveform() please follow the table below.

| Variable | Data Type | Required | Use |
|---|---|---|---|
| CSCData | Structure | Yes | Data file generated from nsc2mat(). |
| | | | |
| | | | |
| | | | |

**Outputs**

This function outputs figures in several save formats.

| Options | Data Type | Use |
|---|---|---|
| | | |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

Plot corrcoef(Line 34)



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### f. MousePSTH()

Code is complete but needs testing before initial publishing.

Computes the signal-to-noise ratio for given channels

```
function [] = MousePSTH( SpikeData, saveData, stimOn, stimTimes, stimColor,
timeBounds, events, fs)
```

**Inputs**

In order to use

| Variable | Data Type | Required | Use |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

**Outputs**

This function outputs figures in several save formats.

| Options | Data Type | Use |
|---|---|---|
| | | |



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

**Important Landmarks:**

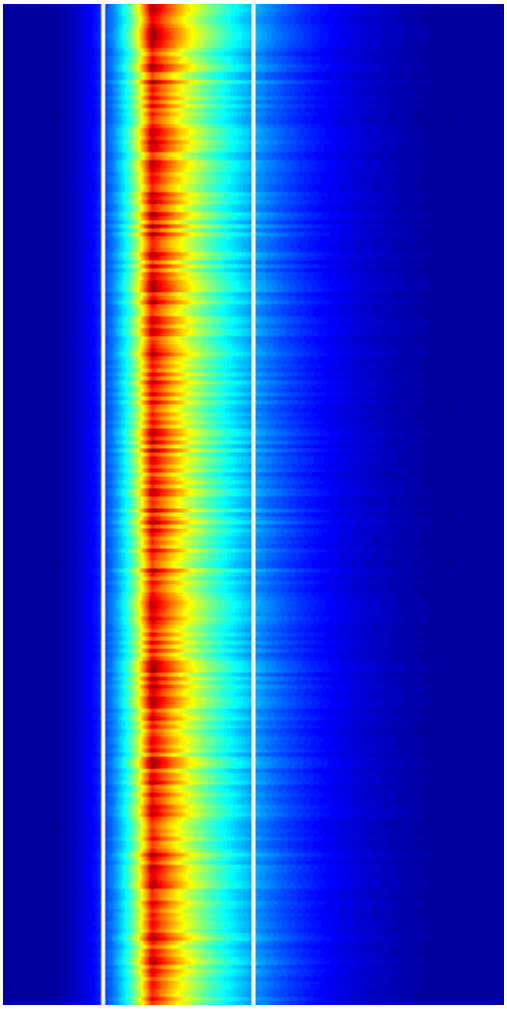
Plot PSTH(Line 34)



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### g. GenHeatmap()

Code is complete but needs testing before initial publishing. This code needs to be adjusted to fit with existing data structures.





This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# VI. Fourth Phase Code – Changes Over Time

## a. MouseAfterStim()

This code is meant to look at any sort of fatigue/recovery/depression/potentiation that occurs after a stimulus (excitatory or inhibitory). May utilize other programs to compute LFP, PSTH, etc from t= -k:n with respect to when the stimulus stops.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. MouseDurStim()

This code is meant to look at any sort of fatigue/recovery/depression/potentiation that occurs during a stimulus (excitatory or inhibitory). May utilize other programs to compute LFP, PSTH, etc from t= -k:n with respect to when the stimulus starts.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# VII. Fourth Phase Code – Changes Between Animals

## a. MouseWildMut()

This code is dependent upon reading an xlsx file that records which experimental day/times correspond to tested mice. It may attempt to compare the LFP, PSTH, etc between these two groups.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. MouseMF()

Male/female version if we use mix gendered cohorts. Otherwise it is redundant with above.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### c. MouseAge()

Follows mice throughout the experimental period, from first implantation to sacrifice. Looking at how the LFP, PSTH, etc change over time.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# VIII. Fourth Phase Code – Behavioral Analysis

## a. MouseGroom()

After a researcher carefully details when the mouse groomed (by watching a video) or from LABORAS. Use these times to plot the matching electrical recordings, LFP, PSTH, etc.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# IX. Miscellaneous Code

## a. Transient

Calcium transient code.
Code is complete but ugly.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### b. TwoPhoton

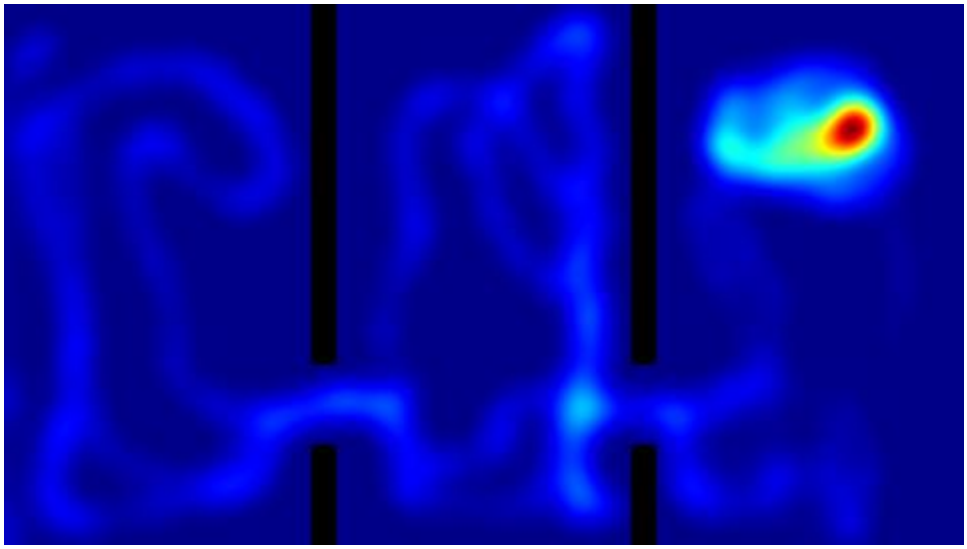
Two photon code.



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

### c. LaborasHeatmap

Convolves a 2D Gaussian with the x-y data from LABORAS.
Code is complete but needs to be transformed into a function.





This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.

# X. Appendix



This document and the information contained herein are proprietary to Kevin J. O'Neill III and are to be used in confidence. No copies are to be made without express written permission of Kevin J. O'Neill III.