**BioMEMs Lab**

Investigation of similarities between electrophysiology and calcium signaling in neuronal using microelectrode arrays and calcium tracking

Points: \_\_\_ / 30

## Module 1 – Background (4 points)

The most common MEMs devices involved in the nervous system are electrophysiological recording devices. In this course you learned about micro-electrode arrays, and here in this lab, you will get the chance to observe the outputs of one such device. In the lab, a microelectrode array was recorded synchronously with calcium using a custom setup. Now you are the engineer responsible for processing the resulting data to identify whether the calcium signal matches with the electrical signals. First, lets look into some background on the microelectrode array to develop a hypothesis. For the first 2 questions use: ‘MEA\_SPEC\_SHEET.pdf’

##### DELIVERABLE (0.5 points) Estimate the maximum number of neurons we could have on our electrode surface. Use average cell diameter of 15 µm.

##### DELIVERABLE (0.5 points) What photoresist is on the surface of the microelectrode array used in this experiment?

##### DELIVERABLE (1 point) Locate a published paper in which a micro-electrode array is used and identify the purpose of electrophysiology in the experiment. Give the citation. Make sure it is primary literature and not a review article! (Bonus point if you can identify the surface coating).

##### DELIVERABLE (1 point) Draw a neuronal action potential and show where Calcium influx is activated

##### DELIVERABLE (1 point) Hypothesize whether the signals will be correlated or not.

## Module 2 – Micro-electrode processing (4 points)

### MATLAB Microelectrode Data Setup

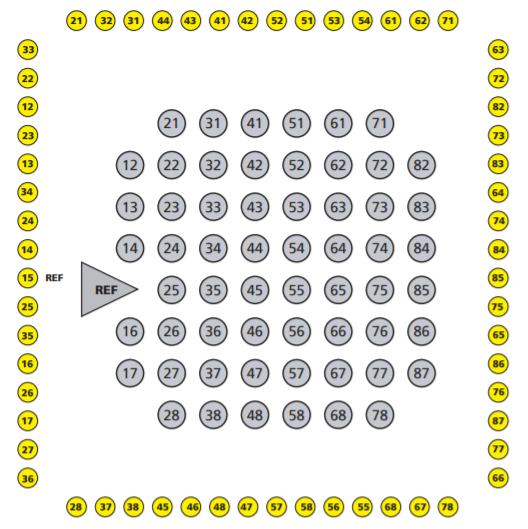
Microelectrode data is stored in the ‘.h5’ file within the folder. To extract the data from the file, we will call our function:

[[E, t, L] = load\_MEA();](#_[E,_t,_L]) **🡨 Control + Click me** or scroll down to appendix for more info!

Which will prompt a file selection. Here, select the ‘MicroElectrodeRecording.h5’ file for import.

Now that the data has been imported into MATLAB, we can observe an electrode over the time series. Below is the map of the microelectrodes over the recording area. You will notice the labeling is in array format, not a counting method. Because of this, we will need to call our map L to transition between the MATLAB indexing within our electrode data E, and our selection. To translate from the MEA label, we can select our electrode using:

E{L(‘12’)} where 12 is the electrode location on the map below.



To plot the electrical activity of the electrodes, you can call

[display\_Electrodes(t,E,L)](#_display_Electrodes(t,E,L)__)

to observe the entire electrode array. Now, find two **electrodes next to each other that seem interesting**

##### DELIVERABLE (1 point) Find two electrodes next to each other and plot them using: [display\_Electrodes(t,E,L,ID1,ID2)](#_display_Electrodes(t,E,L)__)

### Filter signals

Now with the electrode data in the MATLAB workspace, lets run a bandpass filter on the dataset to clear up the noise. Running:

[fE = filterElectrodes(E);](#_fE_=_filterElectrodes(E))

will provide you with a filtered dataset at a bandpass filter of 300-4000 Hz.

Now you can play with the low pass and high pass ranges by including

[fE = filterElectrodes(E,hp,lp);](#_fE_=_filterElectrodes(E))

Experiment a little with these two ranges and observe how this impacts your data on a specific electrode using [display\_Electrodes()](#_display_Electrodes(t,E,L)___1)

##### DELIVERABLE (1 point) The data is recorded at 20 kHz, what is the Nyquist frequency? Show how the data looks if you surpass this threshold.

### Detect Electrical Spikes

Now with filtered data we can detect electrical spikes. The most common spike detection method is falling edge thresholding. Here, we implement this within:

[[SpikeMask, SpikeTrain] = Electrode\_SpikeDetection(fE,L,t);](#_[SpikeMask,_SpikeTrain]_=)

Where E can either be the original electrode data or the filtered version.

Now let’s observe our detected spikes across the array using:

[display\_ElectrodeSpikes(SpikeTrain);](#_display_ElectrodeSpikes(SpikeTrain))

##### DELIVERABLE (1 point) Plot the detected electrode spikes.

Standard practice puts for spike detection puts the falling edge threshold 5 standard deviations below the signal. However, to adjust for more robust spiking we can increase this threshold by including a threshold as:

[[SpikeMask, SpikeTrain] = Electrode\_SpikeDetection(fE,L,t,threshold);](#_[SpikeMask,_SpikeTrain]_=)

##### DELIVERABLE (1 point) Select a second threshold and display how it impacts spike detection. Recommended range: 3-12

## Module 3 – Calcium Video processing (7 points)

### MATLAB Video Analysis Setup

As in Module 2, it is recommended to utilize the MATLAB application on a computer, however this can be completed in the online version as well. If you are using the desktop application, you will have to insure the image processing plugin is installed on MATLAB.

Analysis of videos can prove quite tricky with some software packages due to RAM limits. Here, we will work with video files small enough that you will not run into an issue but it is to note that most high resolution videos can not be completely stored on RAM and have to be loaded in during each analysis method. Here, let’s load our Calcium Fluorometry video into the MATLAB workspace using the function ‘load\_video’.

[V = load\_video()](#_V_=_load_video());

Call this function and select the appropriate ‘.tiff’ file to load.

### Accessing the Video in MATLAB

After loading the video, we can view it within MATLAB. First let’s understand how to display a single image.

Within the V architecture we can identify specific frames as standalone images within the video using:

V.Frame(index).I

Where index is the specific frame we aim to look at. To display this frame within MATLAB you can call the [imshow()](https://www.mathworks.com/help/images/ref/imshow.html) function at a specific index (i.e. V.Frame(1).Image shows frame 1).

\*need to convert the frame to a grayscale.

Use imshow(mat2gray(V.Frame(index).I))

##### DELIVERABLE (1 point): Display an image from the video.

### Segmentation

After loading in the video, we need to select specific cells to track. Here, we will locate these cells, more commonly identified as regions of interest (ROIs). Classically, ROI has a diverse meaning, however in this sense, we will locate individual neurons in the video file. The traditional and most common method for segmentation of a video file is performed by hand in the ImageJ platform. While this tends to yield the most optimal identification of cells, many engineers, mathematicians, and computer scientists have developed automated and sudo-automated methods to identify these locations in a fraction of the time. In this section we will use one of these algorithms to detect neurons.

Here we will use a table to automatically segment the video file using two separate methods: a standard thresholding and an activity dependent differential thresholding.

#### Standard thresholding:

During standard [thresholding](https://en.wikipedia.org/wiki/Thresholding_(image_processing)), each pixel within an image is converted into a binary based on the pixel intensity using:

Where the binary pixel, , at coordinates is identified as 1 if the intensity of pixel at coordinates is greater than the threshold . Here, standard thresholding method retains this design with a slight shift. As is to be discussed in the next method, neuron’s are dynamic in calcium levels, and thus might be below the threshold at one time, while above at another. To bypass this, we average the pixel intensity over the time series, and threshold the time series average.

To observe standard thresholding, let’s call our segmentation function:

[S = segmentation(V,t\_S,t\_D)](#_S_=_segmentation(V,t_S,t_D))

**For now, set t\_D to 0** to observe to sole effect of the standard threshold – t\_S

t\_S is scaled from 0 to 1, where 0 is the minimum pixel intensity and would create a mask of pure 1’s, while 1 is the maximum pixel intensity, which would create a binary mask of pure 0’s. **Now choose something a more in the middle.**

Now if you run this function, you will find your mask under the S architecture as:

S.threshold.bw1

We can observe the segmented regions using:

[display\_ROI(S)](#_display_ROI(S))

Here, we can see the standard threshold in the top left, the deviation threshold in the top right should be black if t\_D was set to zero.

##### DELIVERABLE (1 point): Show the Standard Threshold Binary image and record the threshold value

#### Differential thresholding

As briefly discussed above, the dynamic nature of neuronal calcium level can make standard thresholding difficult. Therefore, it can sometimes be advantageous not to look at the static calcium level to identify a cell, but the change in calcium levels over time. Here, we will call this differential thresholding, which is a gross misnomer as we are not implementing calculus. In fact, the proper terminology for this method is sum of squared differences (SSD), where is the frame.

To perform this, let’s rerun the segmentation function, however **now set t\_S to 0** to observe to sole effect of the differential threshold – t\_D

[S = segmentation(V,t\_S,t\_D)](#_S_=_segmentation(V,t_S,t_D))

Like the standard threshold, the differential t\_D is scaled from 0 to 1. Here, select an interesting value, run the segmentation, and plot the binary image, which is labeled as: S.threshold.bw2

##### DELIVERABLE (1 point): Use display\_ROI(S) to show the differential threshold binary image and record the threshold value used

#### Combined Thresholding

To optimize our segmentation, we can combine the two methodologies. Here, we will merge the standard and differential thresholds: S.threshold.bw1 and S.threshold.bw2 respectively into S.threshold.binary where

With this binary profile, we have a large toolbelt to fine tune our two thresholding values for an optimized segmentation protocol.

##### DELIVERABLE (1 point): Optimize the standard and differential thresholds to segment the image appropriately. Perform segmentation on the video and display the segmented image using display\_ROI(S)

### Analysis

Now, with our specific ROIs, we can track the intensity of each cell over time. We compute the average pixel intensity within each of the ROIs at each timeseries point to create a discrete time series dataset.

#### Observing Calcium Signals.

Call the analysis function:

[[A,Export]=analysis(V,S)](#_[A,Export]=analysis(V,S,Dev,Noise))

Where we will use the V & S objects derived above. Now we will add two more thresholding values: Dev & Noise. These thresholds will be used in the spike detection below.

For now, we will use the set parameters of Dev=3 & Noise=0.05

Here, we have two function outputs, A, which is optimized for object-oriented code, and export, which is optimized to take data out of MATLAB.

To view the average pixel intensity for all ROIs, call

[display\_F(A)](#_display_F(V,A))

##### DELIVERABLE (1 point) Show the average pixel intensity plot

Detection of Calcium flux events from the original fluorescent signal proves quite difficult as the baseline intensities of each ROI are intrinsically different. The two most common methods are to normalize each signal to its baseline, which is recognized as and the second is to observe the numerical derivative of the signal, identified as . The normalization method requires intensive computation for automation as is often not and requires locating with local minima. As a much faster computation, proves a signal for which you can observe shifts in the fluorescent intensity derived from calcium flux events.

Now display the numerical derivative plot of the data with [display\_dFdt(A)](#_display_F(V,A))

##### DELIVERABLE (1 point) Show the numerical derivative plot

#### Calcium Event Detection

The numerical derivative creates a much easier signal to extract calcium events as the are characterized by the positive peaks in the numerical derivative plot. Now we can detect these events by simple thresholding protocols. Here we will use a standard deviation () and noise threshold (). Calcium signals which surpass both thresholds will be detected in spikes such as:

In this case, we will use the standard deviation () of each signal so as not to filter out small fluctuations. This is then scaled by which we can implement as Dev in [analysis().](#_[A,Export]=analysis(V,S,Dev,Noise)) For the noise threshold, we only wish to filter out the high frequency oscillations which may come from electronic components to ensure they are not detected as spikes. While it may be more accurate to filter the fourier transform, it proves sustainable enough to threshold low magnitude signals. For this dataset, Noise=0.05 was manually set, and filters out any noise. This can be left unless you wish to see how shifting this will impact the signal.

Now let’s observe our detected calcium events. We already ran [analysis()](#_[A,Export]=analysis(V,S,Dev,Noise)) with Dev=4 and Noise=0.05 so we can observe the detected spikes stored in: A(ROI\_Index).Spikes.Frame(Time\_Index).Frame

Which yields an array of the timing events for detected events for the corresponding ROI.

Now to observe the detected events we call:

[display\_CalciumEvents(A)](#_display_CalciumEvents(A))

YAY! We now have spiking events which can correspond to the electrical events – kind of.

Real quick let’s play with our standard deviation threshold to see how this impacts our spike detection.

[A,Export]=analysis(V,S,dev)

##### DELIVERABLE (1 Point) Increase or decrease the standard deviation threshold and observe what happens to the event detection.

## Module 4 – Signal Correlation (5 points)

With both signals analyzed, we can delve further into processing. While there is a vast array of metrics to analyze the signals, in this lab we will focus on signal correlation. Both between datasets and across the two datasets.

### Electrical Signal Correlation

Initially, lets correlate electrical signals to identify correlated electrodes. To do this, we will observe the pearson correlation coefficient with the defined function:

E\_corr=continuous\_corr(Signals)

Where A is an input matrix with columns corresponding to variables and rows corresponding to a time series. Here, this matches the dimensions of our electrode “SpikeMask” and thus we can plug it in.

E\_corr=continuous\_corr(SpikeMask);

To view the correlation values, you can plot corr with imshow(E\_corr), which will display a 60x60 pixel grayscale image with white down the middle. This map can then be used to locate the correlation value of each of the electrodes.

##### DELIVERABLE (1 point): Show your electrode correlation map. What would happen if you input the original electrode signals instead of the detected spikes?

### Calcium Signal Correlation

Now, lets develop a similar map for the calcium signals. While we could ideally use the same methodology as above, the calcium signals we have are much more discrete with a 1 Hz sampling rate. **Here, we could only approximate a continuous signal from our discrete signal, so it would be ideal to develop a correlation style in reference to discrete sampling**. We have a discrete correlation function which implements [Sørensen-Dice](https://www.mathworks.com/help/images/ref/dice.html) correlation in:

[Ca\_corr=discrete\_corr(Export.SpikeTrain);](#_corr=discrete_corr(Signals))

##### DELIVERABLE (1 point): Show the calcium correlation map.

### Calcium Signal and Electrical Signal Correlation

We are really interested in comparing the Calcium and electrical signals to determine if they are correlated. To do this we will have to go through some preprocessing.

#### Sample Matching

The calcium signals are recorded at 1 Hz while the electrical signals are recorded at 20 kHz. To perform correlation, the signals need to have identical sampling. Thus, we will down sample our electrical signal using:

[[ds\_SpikeMask,ds\_SpikeTrain]=Downsample\_MEA(t,SpikeMask,Hz)](#_[ds_SpikeMask,ds_SpikeTrain]=Downsa)

Where we can input a **Hz of 1** to match the corresponding calcium frequency. The down sampling of the microelectrode array bins spikes together within each new sample. If any spikes are detected within that timeframe, the down sampled version retains a spike.

Now we can observe the electrode and calcium signals in unison. To do this, we need to know which electrodes are in the Calcium video. This is something that must be completed during the physical experiment as there are no identifying marks on the electrodes. Here, **the calcium video is recorded in between the electrodes ‘54’ and ‘55’.**

Let’s plot our down sampled electrodes and calcium signals using:

[display\_CaE(SpikeTrain,L,'54','55',Export.SpikeArray)](#_display_CaE(SpikeTrain,L,Electrode_)

##### DELIVERABLE (1 point): Display the electrode spikes and calcium events

#### Merging Calcium Events

Lets combine our calcium signals across every ROI to provide a single waveform for correlation between calcium and the electrodes. Use the [CalciumEvent\_Merger](#_CA_Merged_=_CalciumEvent_Merger(A,p) function to merge the events into a single waveform as:

[CA\_Merged = CalciumEvent\_Merger(A,percent);](#_CA_Merged_=_CalciumEvent_Merger(A,p)

where percent is the percent of spikes across ROIs required to count as a network event.

##### DELIVERABLE (1 point): Display the electrode spikes and merged calcium events using:

[display\_CaE(SpikeTrain,L,'54','55',CA\_Merged)](#_display_CaE(SpikeTrain,L,Electrode_)

#### Correlate Merged Calcium and Electrodes

Finally, we can correlate our merged calcium signal with our electrodes. Let’s call our [discrete\_corr](#_corr=discrete_corr(Signals)) as:

Corr\_CaE54=discrete\_corr(CA\_Merged,ds\_SpikeMask(:,L('54')))

Corr\_CaE55=discrete\_corr(CA\_Merged,ds\_SpikeMask(:,L('55')))

##### DELIVERABLE (2 point): Are the calcium events correlated to the nearby electrodes? Was your hypothesis correct or not? (Totally fine if not!)

## Module 5 – Experimentation (10 points)

With all of the tools, it is time to experiment a bit with the processed data. This is a much more open-ended component of the lab. Choose one of the provided Example projects below and investigate how the changes you introduce impact your data down to the final correlation. To present this data, you have 2 options: Graphical Presentation or Abstract. In the graphical presentation you will build a figure based on your results and write a small caption corresponding to the figure, whereas in the abstract, you will summarize your findings in a short writing format.

### Example Project 1 – Data Preparation Analysis

Investigate how preparing the calcium video can impact experimental results. Is there a limit to segmentation accuracy? How does a higher or lower standard deviation impact the correlation?

### Example Project 2 – Software Development

We briefly discussed the use of in [Module 3 - Analysis](#_Observing_Calcium_Signals.). Write a script that converts A(ROI\_Index).Fluorescence(Frame).F into by detecting the baseline intensity () and investigate the differences

### Example Project 3 – Correlation Experimentation

Investigate the correlation between the electrodes and individual cells instead of the combined calcium spikes.

\*Open to other projects, contact Connor Beck @ [Connorbeck1997@gmail.com](mailto:Connorbeck1997@gmail.com) for approval

*Option 1. Graphical Presentation:*

Figure: Create a graphical representation of what you performed. For this you are given artistic freedom. The goal is to show what you did and what you have gained from this study.

Caption(<100 words): Write in text what you can observe from the figure and explain the details.

*Option 2. Abstract:*

(250-500 words): Write a summary of the experiment you did. Include 2-3 sentences of **introductory** material that provides *background* and *motivation* for why you performed this experiment. 2-4 sentences of **methods** where you describe *what* you did. 3-4 sentences of **results** where you describe the *information yielded* from this study, and 1-2 sentences of **conclusion** where you summarize the *results* and *usefulness* of this information.

## Appendix

#### MICROELECTRODE FUNCTIONS

###### [E, t, L] = load\_MEA() opens file explorer for import

[E, t, L] = load\_MEA(filename) *import from filename*

Loads microelectrode data into the MATLAB workspace.

filename – filename and path

*Text,* *Filepath and name of ‘.tiff’ video file to import into MATLAB*

E –Electrode Data

*Cell, Contains the electrode values in µV.*

t – time series of Electrode Data

*Array, Contains the time at each electrode location*

L – Electrode Map

*Array, Map converts from the multichannel systems nomenclature to MATLAB Indexing.*

*\*i.e.* E(‘34’) *returns the MATLAB index* 18 *for which we can then insert to* E{18}

###### fE = filterElectrodes(E) filters electrodes with 300-5000 Hz bp

fE = filterElectrodes(E,hp,lp)

Filters the electrode signals to remove noise.

E –Electrode Data

*Cell*

See [**load\_MEA**](#_[E,_t,_L]) for details

hp –High Pass value

*Value*

*Filters out frequencies lower than hp in Hz*

lp –Electrode Data

*Value, Filters out frequencies higher than lp in Hz*

fE –Filtered Electrode Data

*Cell, Contains the filtered electrode values in µV.*

###### [SpikeMask, SpikeTrain] = Electrode\_SpikeDetection(E,L,t)

[SpikeMask, SpikeTrain] = Electrode\_SpikeDetection(E,L,t,thresh)

[SpikeMask, SpikeTrain] = Electrode\_SpikeDetection(E,L,t,thresh,refr)

Detects Electrical spikes using falling edge threshold detection.

E –Electrode Data

*Cell*

See [**load\_MEA**](#_[E,_t,_L]) for details

t – time series of Electrode Data

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

L – Electrode Map

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

Thresh – Standard deviation threshold

*Value, Falling edge threshold required to identify spikes. No value sets Thresh to 5*

refr – Refractory period

*Value, Refractory period in ms for which we ignore sequential spiking before the end of this period. No value sets refr to 2.*

SpikeMask – Binary map of spikes

*Matrix,*

A matrix for which 1 corresponds to a detected spike and 0 otherwise. Rows are time, columns are indexed as electrodes which can be identified using L

SpikeTrain – Time locations of spikes

*Cell array, Cell mapped electrodes with the time locations of detected spikes in each cell.*

*can access detected spikes for an electrode with* SpikeTrain{L(‘12’),1}

###### display\_Electrodes(t,E,L) *plots all electrodes*

display\_Electrodes(t,E,L,ID) *plots 1 electrode*

display\_Electrodes(t,E,L,ID1,ID2) *plots 2 electrodes*

Plots the voltage on the vertical axis and time on the horizontal axis. Can display single electrode, 2 electrodes for comparison, or the complete microelectrode array.

E –Electrode Data

*Cell*

See [**load\_MEA**](#_[E,_t,_L]) for details

t – time series of Electrode Data

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

L – Electrode Map

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

ID – Electrode ID

*Text, ID location of the specific electrode to plot in Multichannel Nomenclature i.e* ‘12’

###### display\_ElectrodeSpikes(SpikeTrain)

Plots the electrode spikes with electrode index on the vertical axis and time on the horizontal axis.

SpikeTrain – Time locations of spikes

Cell array,

See [**Electrode\_SpikeDetection**](#_[SpikeMask,_SpikeTrain]_=)for details

#### CALCIUM SIGNALING FUNCTIONS

###### V = load\_video()

V = load\_video(filename)

Loads ‘.tiff’ video file into the MATLAB workspace and identifies the video dimensions.

filename – filename and path

*Text,*

Filepath and name of ‘.tiff’ video file to import into matlab

V – Video Object

*Object, contains video, height, width, and length*

Opens File explorer to locate Calcium Video File. Currently limited to ‘.tiff’ video files. Once selected, will import the video file image by image into the workspace under the structure: V.Frame(index).Image

###### S = segmentation(V,t\_S,t\_D)

Identifies Cells or Regions of Interest (ROIs) to track during the video

V – Video Object

*Object, contains video, height, width, and length*

See [**load\_video**](#_Vid_=_load_video(filename)) for details

t\_S – Standard Segmentation Threshold

*positive value from 0-1*

Baseline intensity threshold used to create a primary mask of the image scaled to the maximum possible intensity. Intensity values above this threshold will be retained in S.threshold.bw1.

t\_D – Threshold Differential

*positive integer from 0-1*

Differential threshold used to create a mask of the image scaled to the maximum possible intensity. Intensity values above this threshold will be retained in S.threshold.bw2.

S – Segmentation Object

*Object, contains*

Explain Here

###### [A,Export]=analysis(V,S)

[A,Export]=analysis(V,S,Dev)

[A,Export]=analysis(V,S,Dev,Noise)

Analyzes the regions of interest across the video and detects Calcium events.

V – Video Object

*Object, contains video, height, width, and length*

See [**load\_video**](#_Vid_=_load_video(filename)) for details

S – Video Object

*Object, contains video, height, width, and length*

See [**segmentation**](#_S_=_segmentation(Vid,t_S,t_D)) for details

Dev – Standard Deviation threshold for Spike detection

*positive value > 0*

Standard value: 3 ; Recommended Range: 3-12

Standard deviation of the calcium signal required to surpass for initial identification of a spike.

Noise – Threshold Standard

*positive value > 0*

Standard value: 0.05 ; Recommended Range: Dependent on Experimental Parameters

Threshold to omit detecting events from non-active cells when Dev is insufficient.

A – Analysis Object

*Object, contains Fluorescence, DeltaF, Spikes*

Analysis is an object with rows corresponding to regions of interest (ROI), and 3 columns corresponding to (1) average fluorescent intensity of the ROI at each time point (2) the numerical derivative of the intensity over time and (3) frame number for detected spikes with the corresponding ROI

Export – Spiking Object

*Object, contains SpikeTrain and SpikeArray*

Export retains two separate methods for retaining the detected spikes,

*SpikeTrain:* Binary matrix of spikes, columns are regions of interest, rows are time points.

0 – no Spike, 1 - Spike

*SpikeArray:* Matrix containing the region of interest in the first column and frame for spike in the second colum. Easiest format to export for presenting in secondary software (Excel, R, Origin, etc.)

###### display\_ROI(S)

Displays the thresholding images and the segmented image

S – Video Object

*Object, contains video, height, width, and length*

See [**segmentation**](#_S_=_segmentation(Vid,t_S,t_D)) for details

###### display\_F(A)

Plots the fluorescent intensity of each region of interest over the time frame of recording

V – Video Object

*Object, contains video, height, width, and length*

See [**load\_video**](#_Vid_=_load_video(filename)) for details

A – Analysis Object

*Object, contains*

See [**analysis**](#_[A,Export]=analysis(V,S,Dev,Noise)) for details

###### display\_dFdt(A)

Plots the numerical derivative of the fluorescent intensity for each region of interest over the time frame of recording.

V – Video Object

*Object, contains video, height, width, and length*

See [**load\_video**](#_Vid_=_load_video(filename)) for details

A – Analysis Object

*Object, contains*

See [**analysis**](#_[A,Export]=analysis(V,S,Dev,Noise)) for details

###### display\_CalciumEvents(A)

Plots the detected calcium events with ROI index on the vertical axis and time on the horizontal axis.

A – Analysis Object

*Object, contains*

See [**analysis**](#_[A,Export]=analysis(V,S,Dev,Noise)) for details

#### CORRELATION FUNCTIONS

###### corr=continuous\_corr(Signals)

corr=continuous\_corr(Signal\_A,Signal\_B)

Correlates continuous signals using Pearson correlation. Can either use a matrix input with multiple signals indexed in the columns or can correlate 2 signals of same length. Returned value indicates perfect correlation at 1 and no correlation at 0.

Signals – Signals Matrix

*Matrix, contains a signal in each column of the matrix*

Input to create a correlation map of size(Signal\_array,2)x size(Signal\_array,2)

Signal\_A & Signal\_B – Signal Array

*Array, column formatted signal*

Input both to determine a specific correlation between two signals. Can be used with electrodes by calling:

corr=continuous\_corr(SpikeMask(:,L('12')), SpikeMask(:,L('13')))

corr – correlation matrix / value

*Matrix/value,*

contains Pearson Correlation values

Perfect Correlation: 1 No Correlation: 0

If input is more than 2 signals (matrix with more than 2 columns), corr will retain dimensions NxN where N is the number of input signals. The correlation value between 2 signals (i.e. column 10 and 12) can be located at the index corr(10,12) or corr(12,10). If input is 2 signals, corr will return a single value corresponding to the correlation.

###### corr=discrete\_corr(Signals)

corr=discrete\_corr(Signal\_A,Signal\_B)

Correlates discrete signals using [Sørensen-Dice](https://www.mathworks.com/help/images/ref/dice.html) correlation. Can either use a matrix input with multiple signals indexed in the columns or can correlate 2 signals. Returned value indicates perfect correlation at 1 and no correlation at 0.

Signals – Signals Matrix

*Matrix, contains a signal in each column of the matrix*

Input to create a correlation map of size(Signal\_array,2)x size(Signal\_array,2)

Signal\_A & Signal\_B – Signal Array

*Array, column formatted signal*

Input both to determine a specific correlation between two signals. Can be used with electrodes by calling:

corr=discrete\_corr(Export.SpikeTrain(:,1), Export.SpikeTrain(:,12));

corr – correlation matrix / value

*Matrix/value*

*contains Sorenson Dice Correlation values*

Perfect Correlation: 1 No Correlation: 0

If input is more than 2 signals (matrix with more than 2 columns), corr will retain dimensions NxN where N is the number of input signals. The correlation value between 2 signals (i.e. column 10 and 12) can be located at the index corr(10,12) or corr(12,10). If input is 2 signals, corr will return a single value corresponding to the correlation.

###### [ds\_SpikeMask,ds\_SpikeTrain]=Downsample\_MEA(t,SpikeMask,Hz)

Down samples micro-electrode array spiking data to match the frequency of a calcium video. The down sampled time point will include spikes if any spikes are identified in the original sample.

t – time series of Electrode Data

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

SpikeMask – binary map of spikes

*Matrix*

See [**Electrode\_SpikeDetection**](#_[SpikeMask,_SpikeTrain]_=) for details

Hz – Frequency to Match

*Value*

The desired frequency to downsample to.

ds\_SpikeMask – Downsampled Spike Mask

*Value*

The desired frequency to downsample to. Columns are identified as specific electrodes which can be located using L as in [**Electrode\_SpikeDetection**](#_[SpikeMask,_SpikeTrain]_=)**.**

ds\_SpikeTrain – Downsampled Spike Train

*Cell array*

Cell mapped electrodes with the downsampled time locations of detected spikes in each cell.

can access detected spikes for an electrode withSpikeTrain{L(‘12’),1}

###### CA\_Merged = CalciumEvent\_Merger(A,percent);

A – Analysis Object

*Object*

See [**analysis**](#_[A,Export]=analysis(V,S,Dev,Noise)) for details

percent – percent of ROIs containing calcium event

*Value*

Percentage of ROIs required to contain a calcium event at a single frame for it to be merged into a network calcium event

CA\_Merged – percent of ROIs containing calcium event

*Object*

Percentage of ROIs required to contain a calcium event at a single frame for it to be merged into a network calcium event

###### display\_CaE(SpikeTrain,L,Electrode\_ID1,Electrode\_ID2,Export)

SpikeTrain – Time locations of spikes

Cell array,

See [**Electrode\_SpikeDetection**](#_[SpikeMask,_SpikeTrain]_=) for details

L – Electrode Map

*Array*

See [**load\_MEA**](#_[E,_t,_L]) for details

Electrode\_ID – Electrode ID

*Text*

*ID location of the specific electrode to plot in Multichannel Nomenclature i.e* ‘12’

Export – Spiking Object

*Object*

See [**analysis**](#_[A,Export]=analysis(V,S,Dev,Noise)) for details