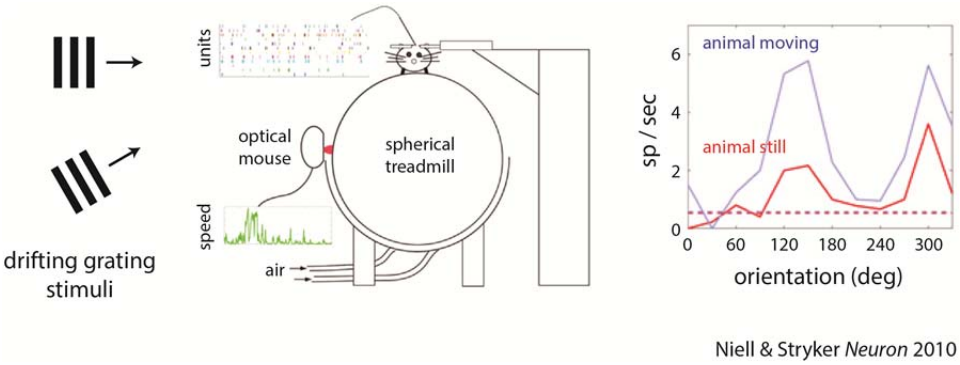
Janelia-JHU Bootcamp 2017
Module on intracellular place field analysis
Albert Lee, Jeremy Cohen

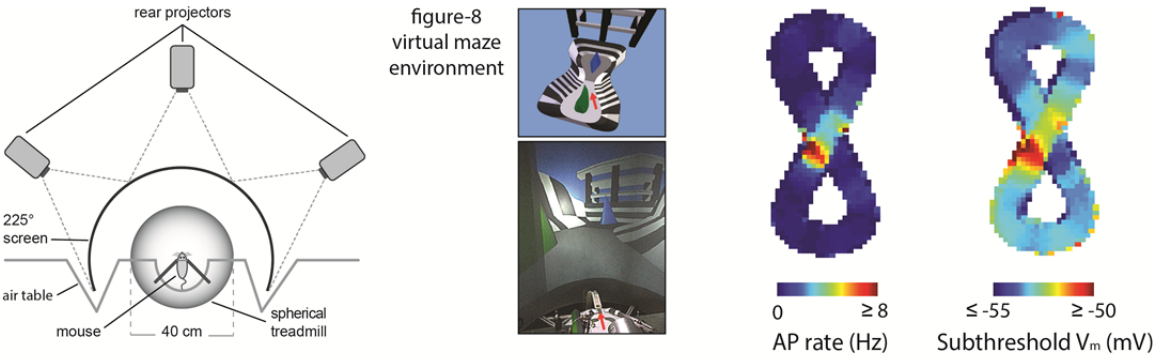
Receptive fields (RFs) are one of the cornerstones of neuroscience. The standard RF of a neuron for a given stimulus space is the neuron's average firing rate as a function of each stimulus value.

> ***Visual RF example*: *Orientation selective neuron*** – which fires at a higher rate to drifting grating stimuli of a particular range of orientations (generally at a particular location in the visual field). This receptive field is plotted as the average firing rate for each orientation.



Niell & Stryker *Neuron* 2010

> ***Hippocampal RF example*: *Place cell*** – which fires at a higher rate when the animal is at a particular location within an environment (called the place field of that cell). This receptive field is plotted as the average firing rate for each location in the environment.



Cohen, Bolstad & Lee *eLife* 2017

For this module, you will compute the RF of a hippocampal CA1 place cell and reveal the neuron's place field, i.e. the figure-8 plot that is second from the right above.

More background on place cells:

-Traditionally, place fields are determined using extracellular recording in freely moving animals
-But you can also get intracellular recordings of place cells in freely moving animals
-And you can get place cells in head-fixed animals navigating virtual reality (VR) environments
-These can be combined to give you intracellular recordings of place cells in VR

You'll be given (in Matlab format):

-the intracellarly recorded membrane potential data from a hippocampal CA1 place cell recording in an animal navigating a VR environment
-the simultaneously measured position data of the animal in the VR environment
-the occurence times of the APs that have been extracted from the intracellular data

Your primary goal will be to compute the AP firing rate as a function of the animal's location in the VR environment, using Matlab to accomplish this. If there is additional time, you can extract the AP firing times directly from the intracellular trace, estimate the AP threshold, bandpass filter the intracellular trace, remove the APs from the trace and compute the subthreshold RF, and so on.

We'll start by going through the data with some exercises that will also familiarize you more with Matlab.

1. Go into the intra_bootcamp2017 folder

2. If you haven't already, familiarize yourself with the material in the **Matlab Get Started Primer, pages 1-3 to 1-30**

3. Create folders "Data", "Programs", and "Results" inside the intra_bootcamp2017 folder

4. Move the Vm_data.mat, position_data.mat, and AP_data.mat files into the "Data" folder

5. Start Matlab

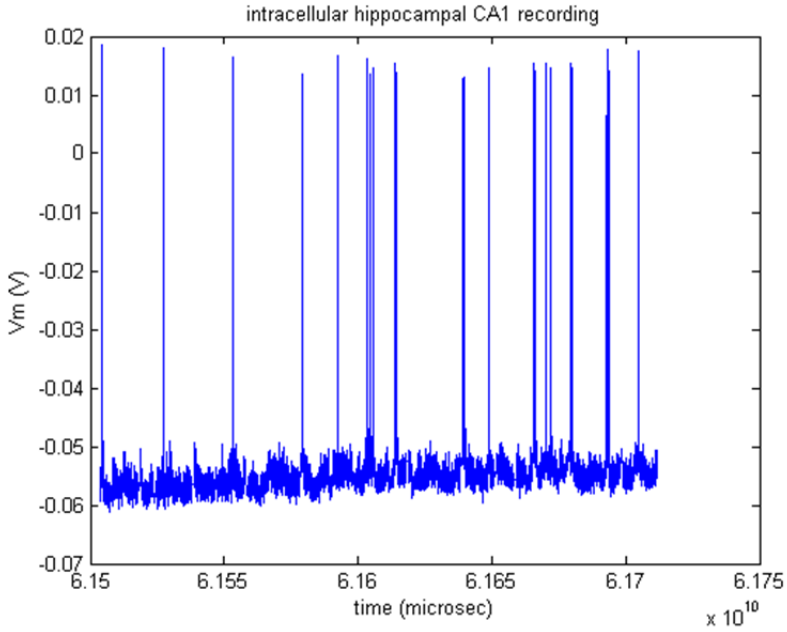
6. At the Matlab command line, type >> **cd** [full path to the Data folder you just made]
e.g. cd C:\Users\leeal\Desktop\intra_bootcamp2017\Data

7. Load in the intracellular data by typing >> **load** Vm_data
You can also **drag and drop** the Vm_data file into the Matlab Command Window
Also load in position_data.mat and AP_data.mat
Then type >> who
This will show you the variables from the loaded data, e.g. t_V, V, t_XYL, X, Y, t_AP_peak.

8. Type >> cd [full path to the Results folder you just made]
This means that the "Results" folder will be your default folder where any results you save will be stored.

7. Type >> **addpath** [full path to the Programs folder you just made]
e.g. addpath C:\Users\leeal\Desktop\intra_bootcamp2017\Programs
This means that any Matlab programs you write or put in "Programs" will be able to be run at the Matlab command line. Without this, Matlab can't find these programs.

8. Type >> **figure**(1); **plot**(t_V,V)
Then to add appropriate labels, type:
>> xlabel(['time (microsec)'])
>> ylabel(['Vm (V)'])
>> title(['intracellular hippocampal CA1 recording'])



That is, the **membrane potential (Vm)** data, **V**, is in units of **volts**, and the time, **t_V**, is in units of **microsec**. The Vm goes from ~-60 to +20 mV at the peaks of APs (i.e. -0.06 to 0.02 V). Note that the time does not start at "0".
You can click on the **"+" icon** and magnify parts of the figure, such as individual APs, and **double click** the mouse to return to the original scale.
Then type:
>> **size**(t_V)
>> size(V)
This shows that t_V and V are both column vectors of the same size. **Entry 1 of t_V, i.e. t_V(1), is the time corresponding to entry 1 of V, i.e. V(1).** This is why plotting with plot (t_V,V) is correct.

9. Type:
>> figure(2); plot(X,Y)
>> xlabel(['X (cm)'])
>> ylabel(['Y (cm)'])
>> title(['animal position'])
This shows the animal's position (in **cm**) during the recording as it runs around a (virtual) figure-8-shaped track.

10. Type:

```
>> figure(3); plot3(X,Y,t_XYL)
>> xlabel(['X (cm)'])
>> ylabel(['Y (cm)'])
>> zlabel(['time (microsec)'])
>> grid on
```

This shows the X,Y position over time. The unit of time here is also **microsec**. The time values of position and membrane potential, V, are in register. But the gap between position time points is much larger than Vm time points because behavioral data is sampled at a much lower rate. Note that the X and Y position data and the t_XYL time data all have the same size. At time t_XYL(1), the animal's position is X(1),Y(1).

11. Let's convert to more convenient and standard units, then replot the data. Comments of what is being done in each line are given at the ends of the typed lines.

To convert, type:

```
>> t_expt_start = t_V(1);  % the time of the first data point
>> t_V = t_V - t_expt_start;  % adjust time to start at 0
>> t_V = t_V/1e6;  % convert time from microsec to sec
>> t_XYL = (t_XYL- t_expt_start)/1e6;  % adjust start and convert to sec
>> t_AP_peak = (t_AP_peak - t_expt_start)/1e6;  % adjust start and convert to sec
>> t_AP_thresh = (t_AP_thresh - t_expt_start)/1e6;  % adjust start and convert to sec
>> V = V*1e3;  % convert membrane potential from V to mV
>> V_AP_peak = V_AP_peak*1e3;  % convert to mV
>> V_AP_thresh = V_AP_thresh*1e3;  % convert to mV
```

To replot, type:

```
>> figure(1); plot(t_V,V)
>> xlabel(['time (sec)'])
>> ylabel(['Vm (mV)'])
>> title(['intracellular recording'])
>> figure(3); plot3(X,Y,t_XYL)
>> xlabel(['X (cm)'])
>> ylabel(['Y (cm)'])
>> zlabel(['time (sec)'])
>> grid on
```

12. Type:

```
>> figure(1); axis([0 8 -70 20])
```

Note that typing "figure(1)" returns control of the command line to figure #1. This is another way of magnifying parts of the data. But axis just affects the visualization; all the data is still there.

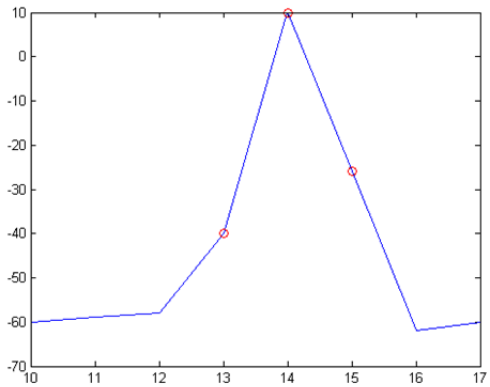
But what if you want to analyze only the data from the magnified time period? For instance, maybe you want to know the median of the Vm during that period. To do this, you need to be able to select specific parts of the data.
The key Matlab command you need for this is "**find**", which you will also find is very useful for many other things. "Find" gives you the entry numbers (i.e. indices) that satisfy the rule(s) that come after it.

Practice code for using "find" and I've written comments after some commands using the "**%**" comment character:

```
>> list = [2 4 6 8 10 12 14];
>> length(list)  % "length" gives you the size of a row or column vector
>> find(list > 10)
>> find(list >= 8 & list < 12)  % The "&" operator is Matlab's character for the "logical AND" function
>> find(list <= 8 | list >= 12)  % The "|" operator is Matlab's character for the "logical OR" function
>> find(list <= 8 & list >= 12)
>> indices = find(list <= 8 | list >= 12)  % This allows you to store the indices
>> new_list = list(indices)  % Here you can use the indices to get the corresponding values from list
```

Now let's practice with a simple cartoon version of the Vm data:

```
>> t_Z = [10 11 12 13 14 15 16 17]
>> Z = [-60 -59 -58 -40 10 -26 -62 -60]
>> figure(4); plot(t_Z,Z)  % This shows something that looks like an AP with poor sampling rate
>> indices = find(t_Z>=13 & t_Z<16)
>> t_selected_Z = t_Z(indices)
>> selected_Z = Z(indices)
>> hold on; plot(t_selected_Z,selected_Z,'ro')
```



The command "hold on" allows you to plot on top of what's already there. The 'ro' means instead of connecting the points with line segments, each point is plotted separately as a red ("r") circle ("o").

13. It's time to write a Matlab function.

Go to the Matlab window and in the upper left select tab "New -> Function"

You don't need the "end" that they put at the end of the function template

Let's start it out this way:

```
function [t_selected_data,selected_data] = ...
select_data_in_time_range(t_data,data,t_start,t_end)
  % Inputs t_data and data must be vectors of the same length
  % File name of function should be same as function name here
  % The "..." lets you continue the command on the next line
  % It's good practice to comment what you are doing
  % It helps to use variable names that are informative
  % Check "help [function name]" and "help [variable name]" to make sure
  %   you aren't naming a custom function or variable after one that
  %   already exists

  % Find indices that are between t_start and t_end, using interval of
  % form [ , )
  indices = find(t_data >= t_start & t_data < t_end);

  % Get the time values in the interval
  t_selected_data = t_data(indices);

  % Get the corresponding data values for those selected times
  selected_data = data(indices);
```

When you've typed this in, save it in your "Programs" folder, using the name "select_data_in_time_range", which will save it as a ".m" file. In this case the .m file is a function.

Now you can try the function, first on the cartoon data:

>> [t_selected_Z_new,selected_Z_new] = select_data_in_time_range(t_Z,Z,13,16)

>> figure(4); plot(t_selected_Z_new,selected_Z_new,'gs')

The last command should overlay green squares on top of the red circles

Now try the function on the intracellular data:

>> [t_selected_V,selected_V] = select_data_in_time_range(t_V,V,0,8);

>> figure(5); plot(t_selected_V,selected_V)

This should look the same as after the "axis" command in step #12

It's good to check that the time values and so on were registered correctly:

>> figure(1); hold on; plot(t_selected_V,selected_V,'r-'); axis([0 200 -70 20])

This overlays red ("r") line segments ("-") over the original (blue) Vm trace, just for the selected data

We can now compute the median for the selected data:

>> **median**(selected_V) % the answer should be -57.2 mV

14. Another key Matlab function is "**for**", as in for loops

Type >> help for

Note that you can always type "help [command name]" to get documentation on how to use any command

For the next set of commands, and for any long series of commands, it's usually easier to open a new Matlab script (upper left tab "New -> Script") and type for commands there, then copy and paste those lines into the Matlab window or run the script by name (which will help identify where errors are)

```
% script.m
%
% Plot a sequence of figures each with consecutive segments of 5 sec of Vm
% data
t_start_list = 0:5:15
num_starts = length(t_start_list)
for i_t = 1:num_starts
  t_start = t_start_list(i_t);
  t_end = t_start + 5;
  [t_selected_V,selected_V] = select_data_in_time_range(t_V,V,t_start,t_end);
  figure; plot(t_selected_V,selected_V)
  axis([t_start t_end -70 20])
  pause
end
```

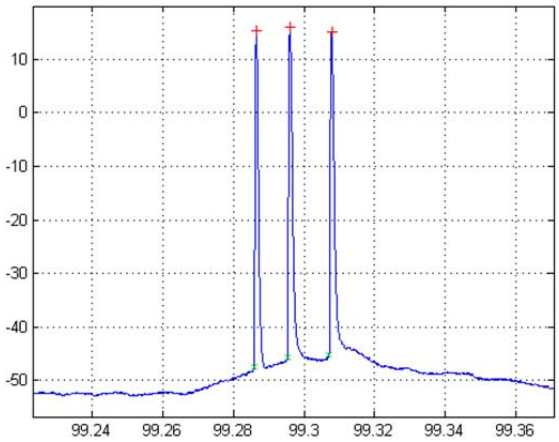
The "**pause**" command means the code will wait until you press any key

15. Now look at the AP data that includes the times of each AP, t_AP_peak, which gives the time value at the peak of each AP.

>> figure; plot(t_V,V); hold on

>> plot(t_AP_peak,V_AP_peak,'r+'); plot(t_AP_thresh,V_AP_thresh,'gx'); grid on

Magnify any part of the trace to see that the AP peaks are correctly identified (red +'s). Also note that the estimated AP threshold is plotted (green x's).

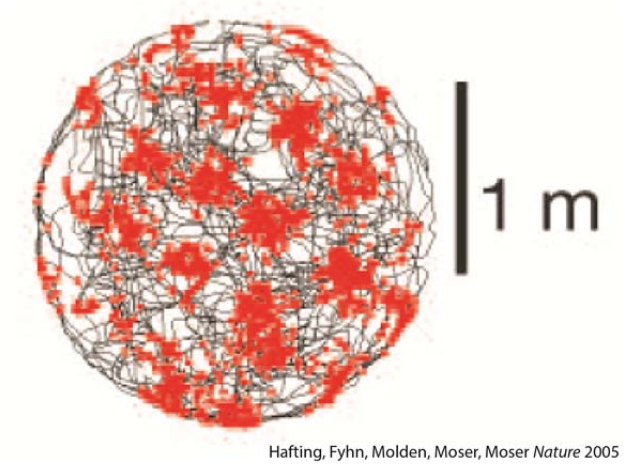


16. Now start a new script called plot_place_field_spikes.m
In this script, write code that finds the X,Y position of the animal when each AP occurs by finding the X,Y position at the position sample just before the AP
(Advanced exercise: Use the Matlab command **interp1q** to find the X,Y position of every AP in one line)

Write code

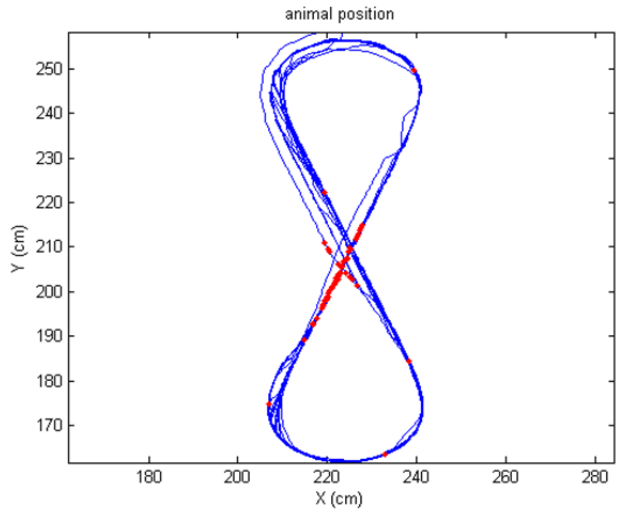
A useful Matlab command is "**zeros**", which will allow you to initialize key variables, such as:
>> X_AP = zeros(num_APs,1); % where num_APs is the total number of APs, which you should get first

Then plot the location of each AP on top of the animal's position trajectory as is often done in papers on hippocampal place cells, entorhinal grid cells, etc:



Hafting, Fyhn, Molden, Moser, Moser *Nature* 2005

In our case:



17. Now the final step: Start a new script called calculate_place_field.m

Write code

Here are the names of the key variables you'll need:
X_bin_center_list % vector listing the centers of bins along the X axis, e.g. 207.5, 212.5, ..., 242.5
X_bin_left_list % e.g. 205, 210, ..., 240
X_bin_right_list % e.g. 210, 215, ..., 245
Y_bin_center_list % e.g. 162.5, 167.5, ..., 257.5
Y_bin_left_list
Y_bin_right_list
total_num_APs_XY_matrix % a matrix containing the number of APs in each of the X,Y bins
total_time_XY_matrix % the total amount of time in sec the animal was in each X,Y bin

For example, the firing rate would be computed for 5 x 5 cm bins
A useful Matlab command for one part of the calculation is "**diff**"
Then plot the place field results to be similar to the plot on page 1 using the Matlab command "**surf**" and "**view(0,90)**". Other useful programs are "**colormap jet**" (or "**colorbar hot**", etc) and "**colorbar**". Play around with different binnings and/or smoothings.

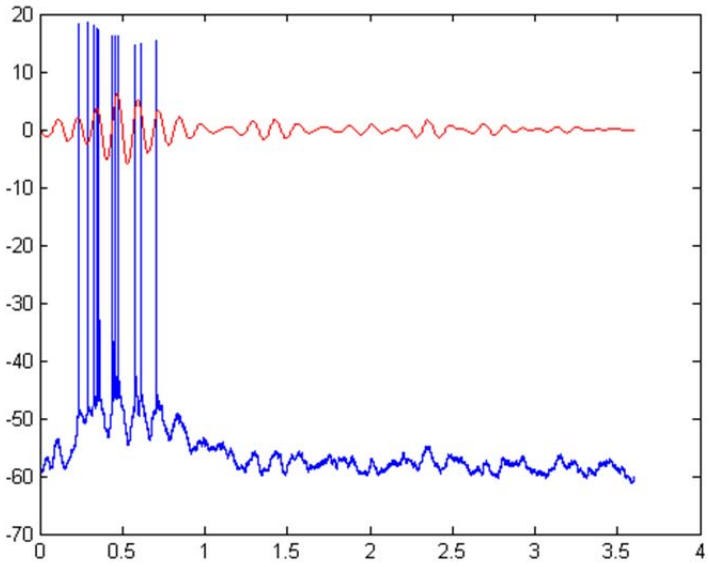
Save the main results in the "Results" folder:
>> save place_field_results ...
X_bin_center_list ...
Y_bin_center_list ...
total_num_APs_XY_matrix ...
total_time_XY_matrix

This saves the variables X_bin_center_list, Y_bin_center_list, total_num_APs_XY_matrix, and total_time_XY_matrix in a "**.mat**" file that you can load again later
Also save the figure separately using the tab at the upper left of the figure window: "**File -> Save As**" (and put it in the "Results" folder too). There are several formats, such as ".pdf". The ".fig" format is a Matlab format that will allow you to click on it and open the figure exactly as before, which you can manipulate like the original (e.g. magnify parts of it).

18. Bandpass filtering
Let's bandpass filter the intracellular data in the theta frequency band (5-10 Hz) using the Matlab commands "**fir1**" and "**filtfilt**"
Start a new script:

```
% Bandpass filter data in the 5-10 Hz frequency band
%   To save time, can run on a few sec of data
%     If one were to filter a lot of data in the 5-10Hz band, then one
%       should first decimate the signal to a lower sampling rate than the
```

```
%       original 25kHz rate, say to a few hundred Hz, but don't do here
t_V = t_V(1:90001); V = V(1:90001);  % first ~3.5 sec of data
V_samplingrate = 1/median(diff(t_V))  % in Hz
f_lo = 5;  % in Hz
f_hi = 10;  % in Hz
n = round((6/f_lo)*V_samplingrate)
  % make filter as long as 6x the lowest frequency (10 is better)
b = fir1(n,[f_lo f_hi]/(V_samplingrate/2));
  % Note: Syntax is b = fir1(n,Wn)
  %         where n  is the order
  %               Wn is a number (2 numbers for bandpass, etc) that is
  %                  between 0 and 1, with 1 corresponding to the Nyquist
  %                  frequency
  %               b  is a row vector b containing the n+1 coefficients of
  %                  an order n lowpass FIR filter (this is a
  %                  Hamming-window-based, linear-phase filter with
  %                  normalized cutoff frequency Wn), with output filter
  %                  coefficients, b, ordered in descending powers of z:
  %                    B(z) = b(1) + b(2)*z^(-1) + ... + b(n+1)*z^(-n)
a = 1;  % for FIR filter
V_filtfiltered = filtfilt(b,a,V);
  % Note: Syntax is y = filtfilt(b,a,x)
  %         which performs zero-phase digital filtering by processing the
  %           input data, x, in both the forward and reverse directions.
  %         where b is a vector that provides the numerator coefficients of
  %                 the filter
  %               a is a vector that provides the denominator coefficients
  %         If you use an all-pole filter, enter 1 for b.
  %         If you use an all-zero filter (FIR), enter 1 for a.
[h,t] = impz(b,a); figure; plot(t,h)
  % impulse response
[h,w] = freqz(b,a,n); figure; plot(w*V_samplingrate/(2*pi),abs(h))
  % frequency response, magnify x axis to 0 to 20 Hz range
figure; plot(t_V,V); hold on; plot(t_V,V_filtfiltered,'r')
```



19. Additional exercises:

Extract the AP times from the intracellular Vm trace

Determine the threshold of each AP using the threshold 10 V/s

Plot a dV/dt vs V phase diagram for an individual AP to see the threshold

Compute the place field from APs in 1D using the t_XYL and L data in position_data

Compute the 1D AP firing rate on a lap by lap basis as in Figure 6A of Cohen et al., 2017

Remove APs and compute the subthreshold RF (that reflects inputs into the cell)

Use the Matlab command "**spectrogram**" to compute the STFT of the intracellular data