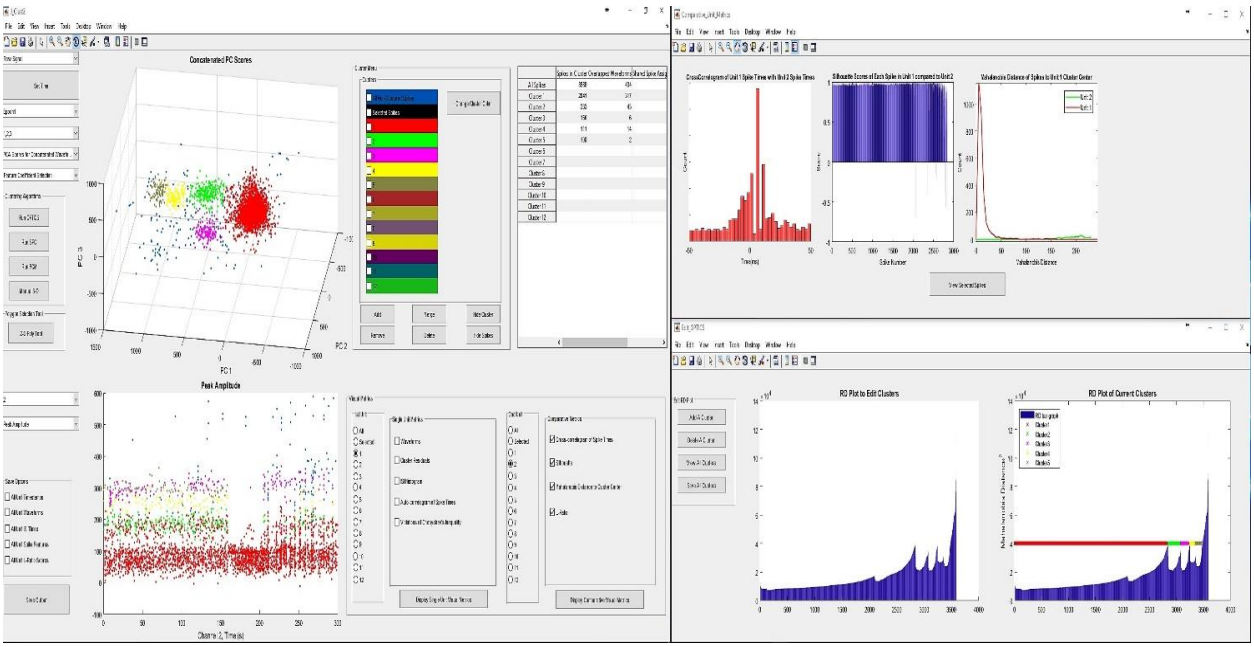
J-Clust2 Manual

Documentation for version 2.0

(September 2017)



Jai Bhagat – jaib1@mit.edu

Introduction

Welcome to J_Clust2, the world's premiere clustering-based spike sorting software!

Requirements:

MATLAB 2014B, including Signal Processing, Statistics and Parallel Computing Toolboxes.

Disclaimer/License:

Table of Contents

I) Loading Data 4

II) Spike Event Detection 5

III) Spike Features

IV) Clustering Algorithms

V) Manual Curation of Clusters

VI) Cluster Metrics

VII) Saving Output

VIII) References

I) Loading Data

J_Clust2 allows you to upload your tetrode in one of three formats: 1) the raw extracellular signal from all four channels of each tetrode; 2) the filtered extracellular signal from all four channels of each tetrode; 3) pre-detected spike waveforms for each spike, and their corresponding timestamps. **Because of the optimal adaptive filtering and spike detection implemented by J_Clust2 (for more details, see section on Spike Event Detection), the first format mentioned above is strongly** preferred**.**

If you are loading a raw or filtered continuous signal, the file should contain a single MATLAB variable. The variable should be a 2-D, [4 x n] array, where n = the number of samples in the recorded data. (E.g. If you have a 10-minute recording sampled at 30000 Hz, n = 18000000).

If you are loading pre-detected spike waveforms and their corresponding timestamps, the file should contain two MATLAB variables. One of these should be a 3-D, [4 x j x k] array, where j = number of samples in a spike waveform, and k = number of spikes in the dataset. The other variable should be a [1 x k] vector of spike times. The names of these variables does not affect the loading.

Note1: If one or more of your tetrode channels is noisy or bad, set all data in that channel to 0. (E.g. if channel 4 is bad, and you're loading in the raw or filtered signal ('**sig**'), enter the command:* ***sig(4,:) = 0. *Similarly, if you're loading in spike waveforms ('**waveforms**'), enter the command:* ***waveforms(4,:,:) = 0.***

**Note2: In the next updated release, J_Clust2 will also be able to directly read in data from Neuralynx and OpenEphys tetrode files.*

II) Spike Event Detection

After using the 'Load Data' drop-down menu to select the file containing your tetrode data, you will click the 'Set Time' push-button to enter in the start and end time for all spikes in the current recording session you wish to view. If you have loaded in a file containing only the raw extracellular signal from your tetrode - congratulations! - you have chosen to take advantage of J_Clust2's optimal spike event detection algorithm.

The adaptive filtering in J_Clust2 (which uses an FIR filter implementing a Hamming Window with bandpass of 400-5000 Hz, with at least 60dB of attenuation in the stop-band frequencies and a roll-off of less than 200 Hz between stop-band and pass-band frequencies) sets the minimal filter length needed to achieve these values based on your sampling rate. **This assures that all detected spikes will be smooth and noise-free, compared to most online implementations of spike detection, which use IIR filters that have non-linear phase and can cause distortions in the shapes of spike waveforms.**

Another issue with common spike event detection is that most neural electrophysiology acquisition systems (e.g. AD, Neuralynx, OpenEphys) capture spike waveforms in the following manner: the raw voltage signal is continuously filtered and the last 'm' samples within the signal are kept in memory. As soon as the voltage of the filtered signal from one of the tetrode's four channels crosses the threshold value, the acquisition system starts recording from all channels simultaneously. The system captures the 'm' samples before threshold crossing and an additional 'n' samples after, which yields a total of 'm+n+1' samples in the spike waveform. This can lead to a failure to detect spikes when two or more neurons fire in close temporal sequences – specifically, when a second spike is fired within the 'n' sample window following the first spike. Acquisition systems tend to handle this in one of two ways: the first is to simply "ignore" a second threshold crossing, and assign all of the samples following the initial crossing to the first spike; the second is to assign a "retrigger" period after the initial threshold crossing, wherein if a

second threshold crossing occurs after the first, spike detection is "retriggered" in order to now capture the 'm' samples before the second threshold crossing, and 'n' samples after this threshold crossing. **In either case, one of the two spikes ends up being missed.**

J_Clust 2 solves this problem via the following implementation of spike detection: First, the algorithm scans over the filtered trace to find every sample that is above the threshold value. Consecutive samples with values above the threshold crossing are marked as belonging to the same spike, while the first nonconsecutive sample above threshold indicates the start of a new spike. At this point, each spike will be indexed corresponding to its first sample that crosses the threshold value. For each spike, the algorithm continues to capture samples following the spike's initial threshold-crossing until the spike reaches its max amplitude value, and the spike is then re-indexed at the sample of max amplitude. 'K' samples are then taken before a spike's max amplitude, and 'l' samples after to complete the spike waveform (where 'k' > 'm' and 'l' < 'n'). After constructing the spike waveform, the algorithm checks if there are any subsequent threshold crossings within 'l' samples of the spike's max amplitude (which would be indicative of an overlap between two or more spikes). Because J_Clust2 has the entire voltage trace, the waveform created by the first threshold crossing is kept. Then, for subsequent threshold crossings within 'l' samples, the algorithm takes the same approach outlined above – it finds the peak following a subsequent threshold crossing, and takes 'k' samples before and 'l' samples after. This will result in some overlap in terms of the samples shared by two spikes (e.g. if there is a subsequent threshold crossing detected with a maximum peak '15' samples after the maximum peak of the first spike, these two spikes will share 'k+l-15' samples ('10' + '21-15'); however, **we can now detect spikes that would otherwise be false negatives as a result of two or more neurons firing in close temporal sequence.**