

# MClust

## Spike sorting toolbox

Documentation for version 3.3

Released October 2003.

Original MClust by

- *A. David Redish (ADR)*,  
current address, University of Minnesota, Minneapolis MN.

Major modifications that have been incorporated into versions 3.0 and 3.2 were made by

- *Neil Schmitzer-Torbert (NCST)*, University of Minnesota, Minneapolis MN.

Modifications that have been incorporated into version 3.0 were made by

- *Peter Lipa (PL)*, University of Arizona, Tucson AZ.
- *Stephen Cowen (SLC)*, University of Arizona, Tucson AZ.
- *Jadin Jackson (JCJ)*, University of Minnesota, Minneapolis MN.
- *Francesco Battaglia (batta)*, University of Arizona, Tucson AZ.

BubbleClust (automated spike-sorter) was written by

- *Peter Lipa*, University of Arizona, Tucson AZ.

KlustaKwik (automated spike-sorter) was written by

- *Ken Harris*, Rutgers University, Newark NJ.

## Spike sorting

Neurophysiological recordings usually include spikes occurring on multiple cells simultaneously. It is important to be able to separate the spike trains of each of these cells. Because spikes occurring on different cells should show different waveform parameters (peak height, total energy, waveform shape, etc.), the spikes from a single cell will form clusters in that high-dimensional space (McNaughton, O'Keefe, and Barnes, 1983, *J. Neurosci. Methods*, 8:391–7; Fee, Mitra, and Kleinfeld, 1996, *J. Neurosci. Methods*, 76:3823–31). Tetrodes and stereotrodes have also proven useful for differentiating spikes from multiple cells: different cells show different spike shapes on each channel of the tetrode or stereotrode (McNaughton, O'Keefe, and Barnes, 1983, Wilson and McNaughton, 1993, *Science*, 261:1055–8). Many advances have been made in spike sorting in the last decade. There is not room to list those advances here. Where specific citations have been incorporated into MClust, those citations are noted in the text.

MClust is a toolbox which enables a user to perform automated and manual clustering on single-electrode, stereotrode, and tetrode recordings. It allows manual corrections to automated clustering results. It outputs *t-files*, which contain (after a header) a list of timestamps in binary format. Timestamps are 32-bit longs at a resolution of 10 timestamps/ms.

## **Requirements**

MClust is written in Matlab™ (The MathWorks Inc., Natick MA) and requires Matlab™ version 6.0 or higher. MClust-3.0 has been tested on PC workstations running the Windows (Microsoft Corp.) family of operating systems (including Win95, Win2000, WinXP, and WinNT). It should, however, be portable to any system capable of running Matlab™ with an ANSI-compatible C++ compiler. Nathaniel Daw (CMU, Pittsburgh PA) has ported it to Linux.

New components added in version 3.3 require toolboxes. If you do not have these toolboxes, you can still use MClust, but will not have the complete functionality. Specifically, the *Statistics toolbox* is needed for measuring Cluster Separation (see below), while the *Signal Processing toolbox* is needed for smoothing the contour map. Warnings are given if the toolboxes are unavailable. These warnings can be suppressed with “warning off MCLUST:ToolboxUnavailable”.

## **Disclaimer**

This code is copyright © by the original authors (see above), 1998-2003.

None of the authors, nor their respective labs, nor their respective universities, assume any liabilities for this code. Use at your own risk. We have done our best to ensure that this code is correct, but do not make any guarantees. This program is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

Use of this code should be acknowledged in any paper that uses data analyzed with it. Acknowledgement should be in the methods section as “(MClust-3.3, A. D. Redish et al)”. Use of automated cutters should be acknowledged as “BubbleClust (P. Lipa)” or “KlustaKwik (K. Harris)”.

This code may be distributed freely. However, all distributed copies must include all components included in the original distribution. This code may not be modified without the express written consent of the author (contact A.D. Redish for permission).

[However, MClust is designed to facilitate the incorporation of new loading engines, new features and new cutting methods. If you have such that you wish to include, please contact me.  
- adr]

Send bug reports, questions, and comments to [redish@ahc.umn.edu](mailto:redish@ahc.umn.edu). Please do not send emails related to MClust to any of the other authors. None of the authors, nor their respective labs, nor their respective universities assume any responsibility for replying to such email, to

fixing said bug-reports, or to maintaining this code. This code is distributed as is. [However, I will reply if I have time. I will do my best to fix bugs, answer questions, incorporate new features and cutters, etc. Many of the updates to version 3.0 and 3.3 were done by others, see above. - adr]

## Changes from MClust 3.0

Changes from MClust 3.0 to MClust 3.3 are primarily driven by a desire for ease of use. However, there are several important additions: calculation of cluster separation, and a tool for waveform discrimination (WaveFormCutter).

### *Limits*

MClust 3.0 used 2 types of limits, Add Limit INCL and Add Limit EXCL. These limits have been replaced by Add Points and Add Limit. No functionality has changed, but the names were confusing some users. The new names should be clearer.

### *Defaults*

MClust 3.3 keeps track of the position and size of several windows, including windows used in the KlustaKwik interface and those of the cluster cutting windows. In the Cluster Cutting Control window, the selected marker and marker size are also saved. Modifications within a session will be preserved for the remainder of the session, and changes that are intended to be permanent can be saved by using the “Save Defaults” option in main MClust menu.

### *Clusters*

When clusters are saved, MClust 3.3 also saves the color and name of each cluster.

### *Cluster separation*

In order to help quantify how well separated a cluster is from other points in the data set, 2 measures of cluster quality are included.

### *Extras*

Additional features and components are available (MClust-3.3-Extras). To use, simply put each component in ClusterOptions folder.

MClust-3.3. now has a similar mechanism to enable the addition of new functionality to the GeneralizedCutter. Any function that takes no arguments and is included in the GeneralizedCutter directory will be available in the GeneralizedCutter.

### *Breaking the 32-sample barrier*

MClust-3.3 does not assume that input waveforms are 32 samples each. A loading engine which provides different sample-length waveforms should work. *Warning:* MClust does still require that all inputs are 4 channels in size.

## Installing MClust

This assumes that you have already installed Matlab on your computer.

1. Create an *MClust* directory in the Matlab™ hierarchy.
2. Unzip the archive (*MClust-3.3.zip*) into the new *MClust* directory.
3. Add the *MClust* directory to your Matlab™ path (see Matlab™ information for how to do this).
4. Start up Matlab™ and type MClust and you're off...

## Components

### *Loading Engines*

In MClust-3.0, we have split out the loading engine. Any .dll or .m file placed in the *LoadingEngines* directory will be declared a plausible loading engine. In the Mclust main window (Figure 1), there is a pop-up menu to select the loading engine.

### *Loading engine requirements*

A loading engine must be callable as a Matlab function. This means it must be either a .m function-file or a compiled mex function-file. It must take as input one or three inputs and provide one or two outputs. MClust-3.0 should work with any loading engine that supplies this functionality.

### *INPUTS*

- *fn* = file name string
- *records\_to\_get* = a range of values
- *record\_units* = a flag taking one of 5 cases (1,2,3,4 or 5)
  1. implies that *records\_to\_get* is a timestamp list.
  2. implies that *records\_to\_get* is a record number list
  3. implies that *records\_to\_get* is range of timestamps (a vector with 2 elements: a start and an end timestamp)
  4. implies that *records\_to\_get* is a range of records (a vector with 2 elements: a start and an end record number)
  5. asks to return the count of spikes (*records\_to\_get* should be [] in this case)

In addition, if only *fn* is passed in then the entire file should be read.

## OUTPUT

- [t, wv]
- t = n x 1: timestamps of each spike in file
- wv = n x 4 x 32 waveforms

## EXAMPLES

- [t,wv] = myLoadingEngine('myfile.dat', 1:10, 2) should return the time and waveforms for the first 10 spikes in the file.
- t = myLoadingEngine('myfile.dat') should return all the timestamps from the file.
- n = myLoadingEngine('myfile.dat', [], 5) should return the number of spikes in the file.

## Features

Features are processed parameters of spikes. Most features produce one number for each valid channel for each spike.

### *Currently available features*

energy — the energy contained within the waveform of each channel of the spike. Also known as the L2 norm. Produces one parameter for each valid channel. In the cutter, this feature will appear as *energy: Ch*.

peak — the maximum height of the waveform of each channel of the spike. Produces one parameter for each valid channel. In the cutter, this feature will appear as *peak: Ch*.

time — the time (in timestamps) of each spike. Produces one parameter per spike. In the cutter, this feature will appear as *time*.

wavePC1,2 — Returns for each waveform the contribution to the waveform that is due to the first (wavePC1) or second (wavePC2) principal component . In the cutter, this feature will appear as *wavePC1(or 2): Ch*.

### *Extra Features*

The following features are in the “Extra Features” directory. To add them to your feature list, copy them into the Features directory under MClust.

area — the area within the waveform of each channel of the spike. Area is defined as the integral of the positive component relative to zero plus the integral of the negative component relative to zero. Also known as the L1 norm. Produces one parameter for each valid channel. In the cutter, this feature will appear as *area: Ch*.

valley — the maximum depth of the waveform of each channel of the spike. Produces one parameter for each valid channel. In the cutter, this feature will appear as *Valley: Ch*.

peakValleyRatio — the ratio between the peak and valley parameters. Produces one parameter for each valid channel. In the cutter, this feature will appear as *peakValleyRatio: Ch*.

spikeWidth — the width of each channel of each spike. Width is defined as the position (out of 32 samples) of the peak minus the position (out of 32 samples) of the valley. Thus *spikeWidth* can be negative. Since *spikeWidth* is integral, a small random value ( $\pm 0.5$ ) has been added to *spikeWidth* to provide for scatter. This provides a better visual picture without changing the integrity of the *spikeWidth* parameter. In the cutter, this feature will appear as *spikeWidth: Ch*.

peakIndex — the sample at which the peak occurs in each spike; defined as the position (out of 32 samples) of the peak. Since *peakIndex* is integral, a small random value ( $\pm 0.5$ ) has been added to *spikeWidth* to provide for scatter. This provides a better visual picture without changing the integrity of the *peakIndex* parameter. In the cutter, this feature will appear as *peakIndex: Ch*.

peak6to11 — the maximum height of the waveform of each channel of the spike in samples 6 to 11 of the wave form. Produces one parameter for each valid channel. In the cutter, this feature will appear as *peak6to11: Ch*.

waveFFT — Returns a value based on the fast Fourier transform of the spike waveform. The 32-point discrete Fourier transform is obtained from the waveform. As the output of this transform is generally symmetrical (with points 1 to 16 highly similar to points 32 to 17), the last 16 points are folded over onto the first 16, and the weighted mean of these first 16 points is calculated. The value returned is in effect the centroid of the FFT. In the cutter, this feature will appear as *waveFFT: Ch*.

### ***Adding new features***

Any Matlab function that is named “Feature\_XXX.m” or “Feature\_XXX.dll” and is placed in the *Features* directory will be included as a possible feature with the name XXX. Features must take as input three parameters and return as output three parameters. MClust-3.0 should work with any loading engine that supplies this functionality.

### ***INPUT***

- V = A tsd of tetraode data (time = n x 1 array of times, data = n x 4 x 32 array of waveforms)
- ttChannelValidity = nCh x 1 of Booleans
- Parameters = a cell array of internal information

### ***OUTPUT***

- FData = feature-calculated data (nSpikes x number of valid channels)
- FNames = a cell array of names (one for each valid channel)

- Parameters = a cell array of internal information. If the feature needs to be stopped and recalled, it will be repassed into the function as Parameters.

## Manual cutting

1. Start Matlab™.
2. Type MClust at the prompt. This will open the main MClust window (Figure 1).
3. Select which features you wish to use to cluster the spikes. Features are moved between the *IgnoreFeature* and *UseFeature* lists by clicking on them.
4. Select the file to cut (you can select either the original data file or any feature data file from this file).
5. Click on ManualCut: Convex Hulls. This will open the Cluster Cutting Control Window (Figure 5).
6. Cut your clusters.
  - a. Click on *Redraw Axes* to open the Cutting Window.
  - b. To add a cluster, select the *Add Cluster* button. When a cluster is added, it contains no boundaries.
  - c. To add a boundary, under the functions selection menu for that cluster, select *Add an inclusive limit*. This will transfer the cursor to the cutting window. Select a set of points by clicking with the left button of the mouse. When you have selected the points, hit the enter key on your keyboard. For speed purposes, points are not drawn on the cutting window online. After hitting the enter key, the a boundary should appear. The boundary used is the convex hull of the points you selected. All spikes that fall within the boundary will change color to match that of the cluster, indicating that these spikes are now within that cluster. To add boundaries on other dimensions, change the axes drawn and add more limits. Spikes within a cluster are defined as those that fall within all of the boundaries defined for that cluster.
  - d. Continue adding clusters and boundaries (add boundaries with either inclusive or restrictive limits) until you are satisfied with the clusters. See below for additional features which allow deleting boundaries, copying clusters, merging clusters, checking cluster parameters, etc.
  - e. When you are done cutting clusters, exit the Cluster Cutting Control Window.
7. Click on *Write Files* to save the processed clusters.
8. Exit MClust.

At any time click on *View Clusters* in the main MClust window for more ways to visualize the data. Clusters cannot be cut in the *View Clusters* window, but they can be examined.

### ***Cutting clusters with convex hulls***

The cluster cutting engine consists of up to three windows (Figure 6). When the cutting engine starts up only the control window will be visible. To create a cutting window, click on *Redraw Axes*. Whenever the *Redraw Axes* checkbox is checked, any changes will be immediately shown in the cutting window. Unchecking and rechecking the *Redraw Axes* checkbox will redraw the cutting window.

Which 2D projection will be shown in the cutting window is controlled by the two selection boxes in the upper left corner of the control window. The arrows immediately below, steps through the possible projections. *View all dimensions* quickly steps through the projections.

Clusters are objects which define regions of the high-dimensional space. When the control window first opens, there will be no cutting clusters shown; only the *zero cluster* will be shown. The zero cluster is the set of all points. To add a cluster, select the *Add Cluster* button. Add clusters as necessary. Up to 99 clusters can be added.

When a cluster is added, it contains no boundaries. To add a boundary, under the *functions* selection menu for that cluster, select *Add inclusive limit*. This will transfer the cursor to the cutting window. Select a set of points by clicking with the left button of the mouse. When you have selected the points, hit the enter key on your keyboard. For speed purposes, points are not drawn on the cutting window online. After hitting the enter key, the a boundary should appear. The boundary used is the convex hull of the points you selected. All spikes that fall within the boundary will change color to match that of the cluster, indicating that these spikes are now within that cluster. To add boundaries on other dimensions, change the axes drawn and add more limits. Spikes within a cluster are defined as those that fall within all of the boundaries defined for that cluster.

You can exit and re-enter the cutting engine as many times as you want.

### ***Add points vs. Add limit***

Clusters in MClust 3.3 work by defining a set of points considered to be members of the cluster (using *Add points*), and refining that set of points by eliminating some of these cluster (using *Add limit*). Thus, the current members of a cluster are defined by the set of points that have been added that fall within the boundaries defined by the limits.

### ***Splitting a cluster***

If you have a cluster that you wish to split into two clusters, copy the cluster and add new restrictive boundaries.

### ***Additional cluster features***

Changing cluster color — Clicking on the color box next to the cluster will pop up a color control window allowing you to change the color used for the cluster. This color will be used for both boundaries and spikes falling within the cluster.



*Check cluster* — Shows key parameters for the cluster. Average waveform, ISI histogram, etc. See Figure 7.

*Delete limit* — In the functions menu for each cluster, *Delete limit* removes the boundary for that cluster on the currently shown projection.

*Delete all limits* — In the functions menu for each cluster, *Delete all limits* removes all boundaries for that cluster. This effectively removes the cluster.

*Delete cluster* — Clusters created with KlustaKwik and BubbleClust do not have limits and must be deleted with *DeleteCluster*.

*Copy cluster* — In the functions menu for each cluster, *Copy cluster* creates a new cluster with the same boundaries as the cluster in question.

*Merge with* — In the functions menu for each cluster, *Merge with* asks for a cluster number and then creates a new cluster in which the boundaries are the convex hull of all the boundaries for the two clusters.

Hiding clusters — When the *Hide* checkbox next to a cluster is checked, no boundaries or spikes falling within that cluster will be drawn on the cutting window. The *Hide* and *Show* buttons on the left panel of the control window, hide or show all the clusters.

*Pack clusters* — Selecting the pack clusters button, removes all clusters that contain no spikes. Other clusters are moved up the list to fill the blank spaces.

*Undo* — MClust has a one-step undo.

### ***Final checks***

After clusters have been cut, the clusters can be checked for key parameters using the following two buttons.

*Check clusters* — Shows key parameters for each cluster currently defined (average waveform, interspike interval histogram, etc.).

*Eval overlap* — Counts the number of spikes in each pair of currently defined clusters. The *Eval overlap* button pops up a window but also writes the table of overlaps to the Matlab™ control window. The formatting on the window can be misaligned, but the text output will always be correct.

### ***Cluster Separation***

At a dinner meeting organized by Cliff Kentros at SFN\*2002, it was noted that there was no standard technique for measuring recording quality with multi-channel electrodes (such as stereotrodes and tetrodes). At that meeting, it was noted that a measure of *cluster separation* would go a long way towards providing those quality measurements. MClust-3.3 includes two measures *isolation distance* (developed by KD Harris *et al. Neuron*, 32:141, 2001) and *L-ratio* (developed by NC Schmitzer-Torbert *et al. submitted-a*). Both measures assume that

the cluster forms a multi-dimensional Gaussian in feature space. *Isolation distance* measures how many sigma one has to go to double the number of points in the cluster. *L-ratio* measures the number of extra-cluster points, weighted by the expected  $\chi^2$  distribution expected from the Gaussian shaped cluster.

We have submitted a paper detailing these two measures and their respective properties (NC Schmitzer-Torbert *et al.* submitted-b). People wishing to report cluster separation values from MClust should cite Jackson *et al.* (*Society for Neuroscience Abstracts*, 2003).

### ***ClusterOptions***

In MClust-3.3, there is a new directory which contains “Cluster Options”. These are generally for viewing clusters, but can be used for any cluster function you wish. Under the *MClust* directory is a *ClusterOptions* directory. Under the *ClusterOptions* directory is an *Extras* directory. To use these options, move the file up from the *Extras* directory into the *ClusterOptions* directory. To remove these options, move the file back down.

These files will be visible in the functions list for each cluster. These include AverageWaveform, ISI histograms, etc.

Note: Any file that appears in the *ClusterOptions* directory will be available in the functions list for each cluster. Any file that takes as its sole input a cluster index (i.e. which cluster called it) and has no outputs can be used as an additional cluster option.

### ***GeneralizedCutterOptions***

In MClust-3.3, there is a new directory which contains “Options for the Generalized Cutter”. These are generally for manipulating clusters, but can be used for any function you wish. Under the *MClust* directory is a *GeneralizedCutterOptions* directory. Any file that appears in the *GeneralizedCutterOptions* directory will be available in the functions list for each cluster. Any file that takes no inputs and has no outputs can be used as an additional option.

### ***Autosave***

MClust keeps track every time a change is made to the defined clusters. After 10 steps, it automatically saves the current clusters and the current defaults. The current clusters are saved in *autosave.clusters* and the current defaults are saved in *autodflts.mclust*. Clicking on the *Autosave* button forces an immediate autosave.

### ***Contour plots***

Clicking on the *Show contour* button creates a contour plot showing the current cutting window. Each time the cutting window is updated, the contour plot will also be updated. The contour plot cannot be used for cutting. But it can be useful to help see whether clusters need to be separated. If the contour plot is not updated, clicking on *Show contour* will redraw the contour plot window.

## Automated cutting with manual touch-up

*MClust-3.0* includes two automated spike-separation algorithms, *BubbleClust* and *KlustaKwik*. Both algorithms find clusters of points in a high-dimensional space. Because they work in all dimensions simultaneously, they tend to separate noise and real spikes better than manual cutting. However, both algorithms can also miss entire clusters as well as incorrectly merging two clusters. Therefore *MClust-3.0* offers the user the chance to touch-up and correct the automated algorithms.

*BubbleClust* starts from small bubbles of similar spikes and merges the bubbles, bubble by bubble. This creates a tree of merges. In the *BubbleClustDecisionWindow* (FIGURE), the user selects which at which merges should be declared useful clusters. These clusters can then be exported to the manual cutting window (see above).

*KlustaKwik* performs an expectation-maximization fit of  $n$  Gaussians to the data. This creates a set of putative clusters. As noted above, *KlustaKwik* will often split a single cell's data into multiple clusters and may sometimes merge two clusters that are in fact separate cells. These can be corrected in the *KlustaKwikDecisionWindow*.

The two algorithms are slightly different, but their role in *MClust* is similar. Both *BubbleClust* and *KlustaKwik* are EXE programs which, once compiled run outside of Matlab. First, you must run the programs (the best way is with the batch-processing supplied with *MClust-3.0*, this can be done off-line without user interaction). Then the data is read in and the user selects/corrects the cut. Finally, the clusters are exported into the manual cutting system for further touch-up and write-out.

## Batch Processing

*MClust 3.0* supplies a program (*RunClustBatch.m*) that allows the user to apply *BubbleClust.exe* or *KlustaKwik.exe* to data files. The output of these programs can then be viewed and refined using *MClust 3.0*.

Using *RunClustBatch.m*, the user can generate clusters using *BubbleClust* or *KlustaKwik*. *RunClustBatch* loads settings from a batch file and then runs the automated clustering program. Default batch files are included in the *MClust/Batch* directory; *Batch\_KlustaKwik.txt* contains settings for running *KlustaKwik*, and *Batch\_BBClust.txt* contains settings for running *BubbleClust*. See below for a description of the batch file fields.

*BubbleClust* and *KlustaKwik* are not able to process unlimited numbers of spikes. Capacity depends on both the number of spikes and the number of features by number of valid channels. With 12 dimensions (3 features with 4 valid channels), *BubbleClust* on our machines can process files of up to 350,000 spikes, and *KlustaKwik* can process files of up to 1.3 million spikes (the largest that we have tested it with). If larger files are to be processed, the feature data files need to be split into a number of smaller pieces.

*RunClustBatch* uses a variable, *SubSampleToNSpikes* (see description below below) to identify the maximum number of spikes to send to the clustering program (*BubbleClust* or

KlustaKwik). If this threshold is exceeded, RunClust batch will split the feature data files into multiple files whose size will be no larger than the value of SubSampleToNSpikes. Each split file will be identified with a b\*, where \* is the number of subset (For example, TT4b1\_energy.fd is the energy feature data file for TT4.dat, and b1 identifies it as the first subset of that file). When using MClust3.0 with this type of output, it is recommended that you use the decision windows to find clusters of interest and export them to the Cluster Cutting Window. Then, save the clusters (do not write files). Do this for each split file (TT4b1, b2, b3, etc). Then, load in the original data file (In this case, TT4.dat). Load in each set of clusters from the split files. Now, you can use MClust to identify which clusters from each split file are the same and should be merged into a single cluster.

To use RunClustBatch,

1. Copy the appropriate default batch file (Batch\_KlustKwik.txt or Batch\_BBClust.txt) to the directory containing the data files which are to be processed.
2. Rename the batch file copy Batch.txt.
3. Start up Matlab, and move to the directory containing the data files to be processed.
4. Type RunClustBatch.

RunClustBatch will generate feature data files for each of the features specified in the batch file. The default location for these files is in a directory, FD, which is a subdirectory in the directory containing the data that is being processed. MClust3.0 assumes that feature data files are either in this subdirectory, or in the same directory as the data files.

### ***Batch file fields***

*ProcessingDirectory* — Directory containing the data to process. Usually the current directory.

*FeatureDataDir* — Directory to place the feature data files that are created in running RunClustBatch.

*FindFilesSearchString* — A string used to search the processing directory for files to process. (Example, TT\*.dat)

*AddToBatchList* — A space separated list of other files to add to the list of files to process. (Example, TT1.dat TT2.dat TT3.dat). Useful if you only want to process a few files in a directory.

*RemoveFromBatchList* — Files to remove from the list of files to process. Useful to remove files which do not contain any spikes.

*ClusterAlgorithm* — BBClust or KlustaKwik

*LoadingEngine* — Specifies the loading engine to use.

*UseFeatures* — Features to use in clustering, generally we use energy, wavePC1, wavePC2 and sometimes waveFFT.

*ExtraFeatures* — Other features which will be calculated, but not used for clustering. These are features you can calculate during the call to runclustbatch to save yourself time when loading files into MClust, because they will already be finished.

*ChannelValidity* — What channels to use in calculating feature data files.

*SubSampleToNSpikes* — Threshold for splitting feature data files. If files are too large, MatLab won't be able to load all of the spikes into memory at one time. Also, BubbleClust will choke if too many spikes or features are sent to it. On our machines, 350,000 is a good value for BubbleClust, and 2,000,000 is a good value for KlustaKwik. BubbleClust, in our experience, is slow for files approaching 350,000 spikes.

*NumberOfNearestNeighbors* — number of nearest neighbors to use with BubbleClust

*KKwik\_MinClusters* — Minimum number of clusters to initialize KlustaKwik with. For more information, see original documentation of KlustaKwik.exe

*KKwik\_MaxClusters* — Maximum number of clusters to initialize KlustaKwik with.

### **Using KlustaKwik or BubbleClust output**

Once BubbleClust or KlustaKwik has been run on a data file, the generated clusters can be viewed using MClust3.0.

1. Start Matlab™.
2. Type MClust at the prompt. This will open the main MClust window (Figure 1).
3. Select which features you wish to use to cluster the spikes. Features are moved between the *IgnoreFeature* and *UseFeature* lists by clicking on them.
4. Select the file to cut.
5. Click on BubbleClust Selection or KlustaKwik Selection. This will open a decision window for viewing the automatic clustering output.

### **KlustaKwik Decision Window**

The general purpose of the KlustaKwik Decision Window is to allow the user to examine the KlustaKwik clusters and select 1) which clusters are likely to be cells, and 2) which clusters result from the splitting of a single cell and need to be merged. Clusters can be either merged in the Cluster Cutting Window, or they can be merged on export to MClust.

Plotting: If the View2D, View3D or Contour boxes are checked, these plots will be updated anytime a relevant change is made. When checked, the current cluster is always shown, as is the held cluster, if there is one. If show all points is checked, all of the points will be shown in black, and if show keeps is checked, then the keeps will be shown as well.

## ***Window Options***

**Selecting a cluster:** By clicking on the cluster number (to the right of the colored box), the cluster can be selected and will change color to cyan. The text in the CURRENT CLUSTER box will be changed to that of the selected cluster, and the waveform and histISI will be shown below in blue (The waveform on invalid channels will be shown, but plotted in black).

**X,Y and Z axis** — Shows which features are plotted on each axis.

**View 2D** — Shows a two-dimensional plot of Y by X

**View 3D** — Shows a three-dimensional plot of X, Y and Z.

**Contour** — Shows a contour plot of the density of points of the current x and y axis selections.

**Correlation** — Shows the correlation of the waveform of the current cluster with the waveform of each other cluster and displays the correlation coefficients to the right of the cluster number. Correlation is based on the waveforms from all valid channels. Useful for identifying which clusters likely are from the same cell.

**Export all Clusters** — Loads the Cluster Cutting Control Window and creates one cluster for each cluster which has been checked in its keeps box in the KlustaKwik Decision Window.

**Export with Color Merge** — Loads the Cluster Cutting Control Window and creates one cluster for each set of keeps sharing the same color. If several clusters are judged to be the same cell and all are set to the same color, these clusters will be automatically merged into a single cluster. This saves the user a step after loading the Cluster Cutting Control Window and decreases the number of clusters to sort through in the cutting window.

**Exit** — Closes the KlustaKwik Decision Window. Does not save any of the keeps, or colors which have been selected for the Klusta Kwik Clusters.

## ***Cluster options***

**Hold** — Show the waveform and ISI of this cluster in red. Used to identify which clusters are identical to other clusters and should be merged.

**Keep** — Select this cluster to be exported to MClust when an Export option is chosen.

**Selecting a color:** Click on the colored box next to the cluster number. A palette of colors will be shown. Click on one and select ok to change the color of the cluster. Choose cancel to leave the color unchanged.

## ***BubbleClust Decison Window***

The BubbleClust Decision Window (see Figure 4) shows a merge tree where each box of the tree represents one cluster, or the merge of higher clusters. The lowest level shows the merge of all bubbles, and the highsEach cluster is referred to as a bubble. By clicking on a box, and clicking on statistics, information about that bubble is shown in the lower left-hand corner of the figure, including the waveform, interspike interval , and firing rate data. The goal is to find the

lowest level which is clearly a cell which at a higher level cannot be split into multiple cells, (For an example, see Figure 4, and the discussion below under *Hold 1/2*).

### ***Window Options***

*Redraw Bubble Tree* — Recreates the Bubble Tree, losing information on which bubbles are KEEPs and any bubble colors which have been changed.

*Save* — Saves the current state of the Bubble Tree (which bubbles are KEEPs).

*Load* — Loads a saved Bubble Tree (which were KEEPs).

*Export Clusters to MClust* — Loads the Cluster Cutting Control Window and creates one cluster for each bubble which is a KEEP.

*Exit* — Closes the Decision Window

*ID/Level* — Identifies the location of the current bubble. Level specifies the row, with row 1 being the bottom row. ID specifies which bubble in that row, with 1 being the left most bubble.

*Keep* — When checked, marks the current bubble to be exported to the Cluster Cutting Window when Export Clusters is selected.

*Statistics* — Shows descriptive data for the current cluster, including waveform, a histogram of interspike intervals, and firing rate information. When the checkbox to the left of the Statistics pushbutton is checked, the statistics will be updated each time a new bubble is selected.

*AutoCorr* — Generates an autocorrelogram for the current bubble, and displays it over short and long time scales.

*XCorr* — Lets the user examine the correlation in spike times between the current bubble and another, user specified, bubble.

*CheckBB* — Shows the cluster check sheet (same as in the Cluster Cutting Window) for the current bubble.

*Hold 1/2* — When checked, holds the waveform and interspike interval information of the current cluster. When a new cluster is selected, shows the information of Hold 1 in red and Hold 2 in green. Used to determine if the merged bubble is the same as higher level bubbles. See Figure 4 for an example: the bubbles labeled 1 and 2 are the two bubbles being held. In blue, is shown the bubble beneath those two, which represents the merge of those two bubbles. In this case, the higher level bubbles should likely be kept, as Hold 2 looks different from Hold 1 and the merge. On the XY axes, it can also be seen that the blue and red clusters (corresponding to Hold 1 and Hold 2 respectively), appear to separate on energy 1 by energy 2.

## View Clusters

Selecting *View Clusters* opens up the View Clusters Control window. This window contains most of the same controls that the Cluster Cutting Control window does, however, clusters cannot be modified in the View Clusters windows. If clusters are modified while the View Clusters Control window is open, the cluster list in the View Clusters Control window will not be modified. Select *Update Clusters* to bring the cluster modifications made with the cutting windows into the view clusters windows.

The main advantage the View Clusters windows brings is that it displays the data in three dimensions. Checking the *Rotate* box will rotate the 3D window. The 3D window can also be manually rotated with the mouse.

## Write files

When done cutting clusters, select *Write Files* to output the processed data. MClust writes out three file types.

*.t files* — Each cluster generates a corresponding *.t* file. This file contains a header (beginning with *%%BEGINHEADER* and ending with *%%ENDHEADER*) and then consists of a list of timestamps. These timestamps are the times at which the spikes in the cluster occurred. The file format for *.t* files is in binary.

*.cluster files* — Each input file generates a single *.cluster* file. This file is a Matlab binary file containing the cluster objects themselves.

*.cut files* — This is an ASCII list of which cluster each spike was assigned to. Spikes that do not fall into any cluster fall into the 0 cluster and are labeled with 0. Spikes that fall into multiple clusters are labeled with the error code -1.

## Additional features

### ***Loading and saving defaults***

The *defaults.mclust* file contains information for the way MClust looks. It saves the color scheme used for the clusters, the features used versus features ignored, the channel validity, and the loading engine.

Autosave saves the current default settings in *autodflts.mclust*. These can be loaded using the *Load defaults* button.

When MClust starts up it looks first for *defaults.mclust* in the current directory, then in the MClust directory.

### ***Loading, saving, clearing clusters***



Clusters can also be loaded and saved separately. When loading clusters, the features used and the channel validity must be identical to when they were saved. *Write Files* saves clusters automatically.

Clearing clusters deletes all limits and packs the clusters. There is no undo for clearing clusters.

### ***Clearing the workspace***

A button is supplied which clears all of the global variables used by MClust-3.0. Clicking this button is equivalent to exiting and restarting Matlab. It is recommended that the user click this button between cutting different data files (e.g., different tetrodes).

## **Cutting non-tetrode data**

Some people have used MClust to do spike sorting on single-electrode (SE) and stereotrode (ST) data. It should work on those data as well. In order to load SE or ST data, the user must (1) use an appropriate loading engine and (2) turn off the channel validity for channels 2, 3, & 4 (for SE) or 3 & 4 (for ST). Selecting an SE or ST loading engine does not automatically affect the channel validity.

A loading engine which loads SE or ST data must pad the waveforms with 0's to fill out four channels. [We are looking into the possibility of making the code compatible with different size waveform matrices, but it turns out to be more complex than we hoped. – adr]

## **FAQ (Troubleshooting / Warnings / Issues)**

### ***Files with more points than your computer can handle***

*“My computer memory can only handle 50,000 points,  
but my tetrodes have 200,000 each. What can I do?”*

MClust-3.0 is now designed to load files as needed which should greatly increase the number of spikes that a computer can handle. If this is still a problem, one can always add more memory.

The old issues of subsampling and splitting files have been removed because they do not work with the new loading-engine facility.

### ***Matlab™ storage issues***

*“Every time I load in a new tetrode, MClust gets slower and slower. Why?”*

Matlab™ has an inefficient heap storage algorithm. This means that between cutting each tetrode you should either run `clear; clear global; pack` or you should exit Matlab™ and restart it each time. Clicking the ClearWorkspace button between cutting tetrodes should also solve this problem. When in doubt, however, we recommend exiting and restarting Matlab between tetrodes.

## **Other platforms**

*“I have LINUX on my PC. Can I still run MClust?”*

MClust has only been tested on PCs running the Windows family of operating systems. However, it should run on any machine that can run Matlab 6.0 and has an ANSI-compatible C++ compiler. If you want to try porting MClust to another platform, contact me at “redish@ahc.umn.edu” and I will do my best to help you.

## **Matlab™ licenses**

*“Do I have to have Matlab™ to run MClust?”*

Yes. MClust runs within the Matlab architecture. Therefore you will need to have Matlab to run MClust.

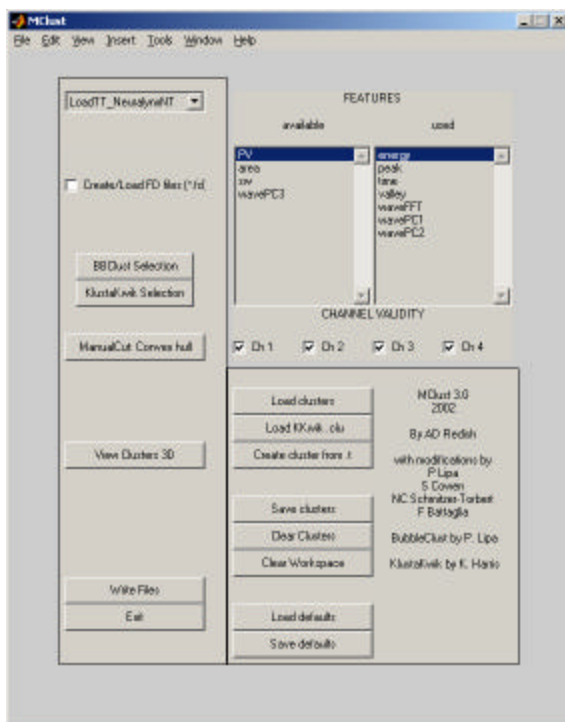
As noted above, some functionality requires toolbox licenses as well. If you do not have those toolboxes, MClust will still work and will still be able to cut clusters.

## **Cutting Engines**

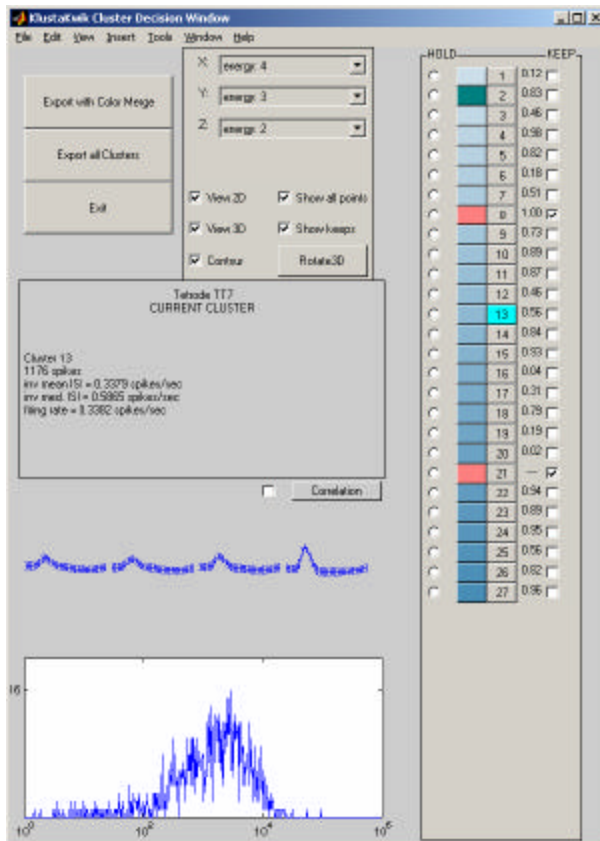
*“Which cutter should I use?”*

MClust-3.0 is supplied with three cutting engines (Manual, BubbleClust, KlustaKwik). The manual cutting engine provides essentially the same capabilities of MClust-2.0. BubbleClust is primarily used by the NSMA lab (U of AZ, Tucson AZ). The RedishLab (U of MN, Minneapolis MN) generally uses KlustaKwik. They all have their advantages and their disadvantages. A thorough discussion of the issues related to cutting engines is beyond the scope of this documentation, however, I will say that: The Manual Cutter has been extensively tested (through MClust-2.0). Cutting tetrodes with it tends to be slower than the either of the other automated systems because it requires more user interaction (obviously). We (RedishLab) have tested the KlustaKwik-MClust interface more thoroughly than we have tested the BubbleClust-MClust interface, but we believe that both work correctly.

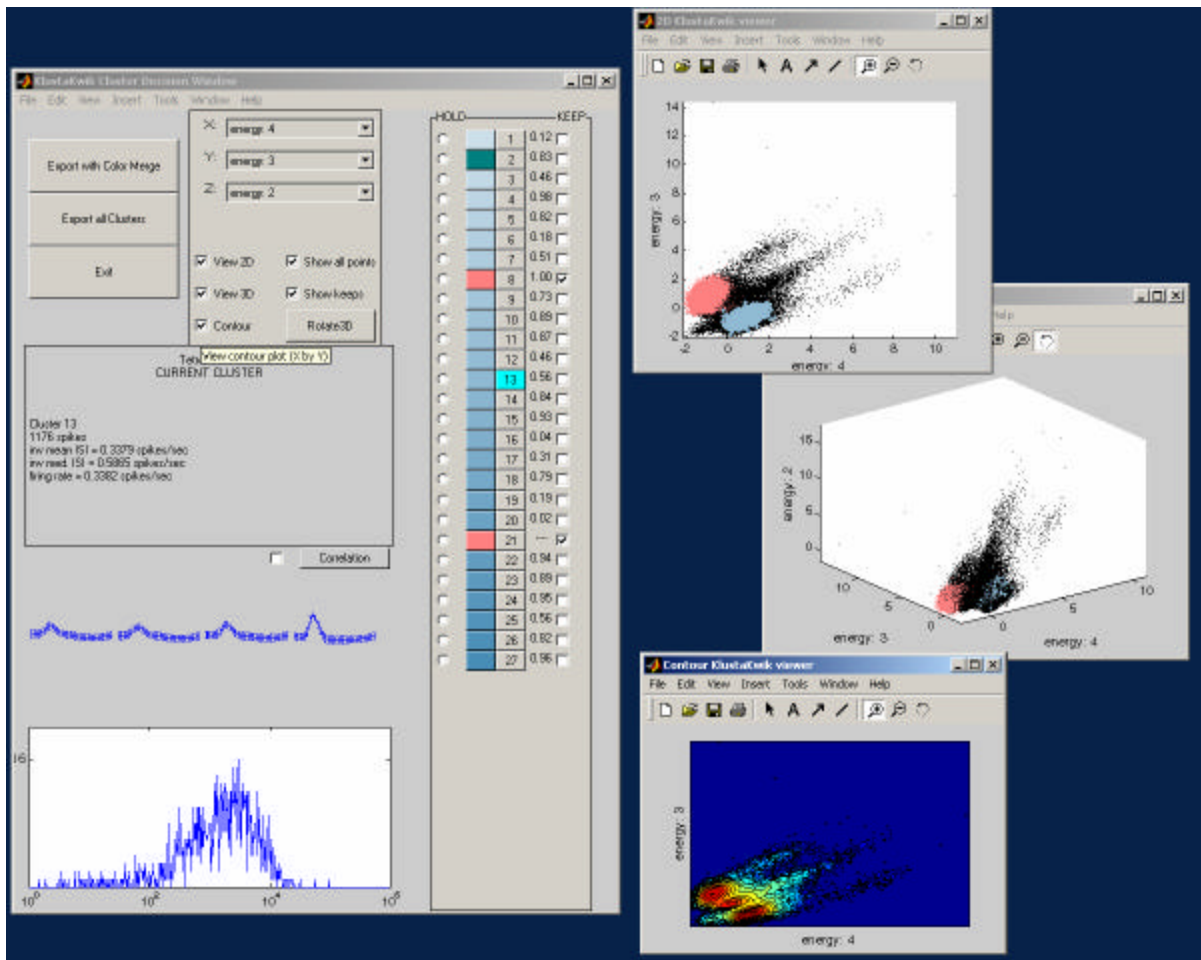
## Example screen-shots



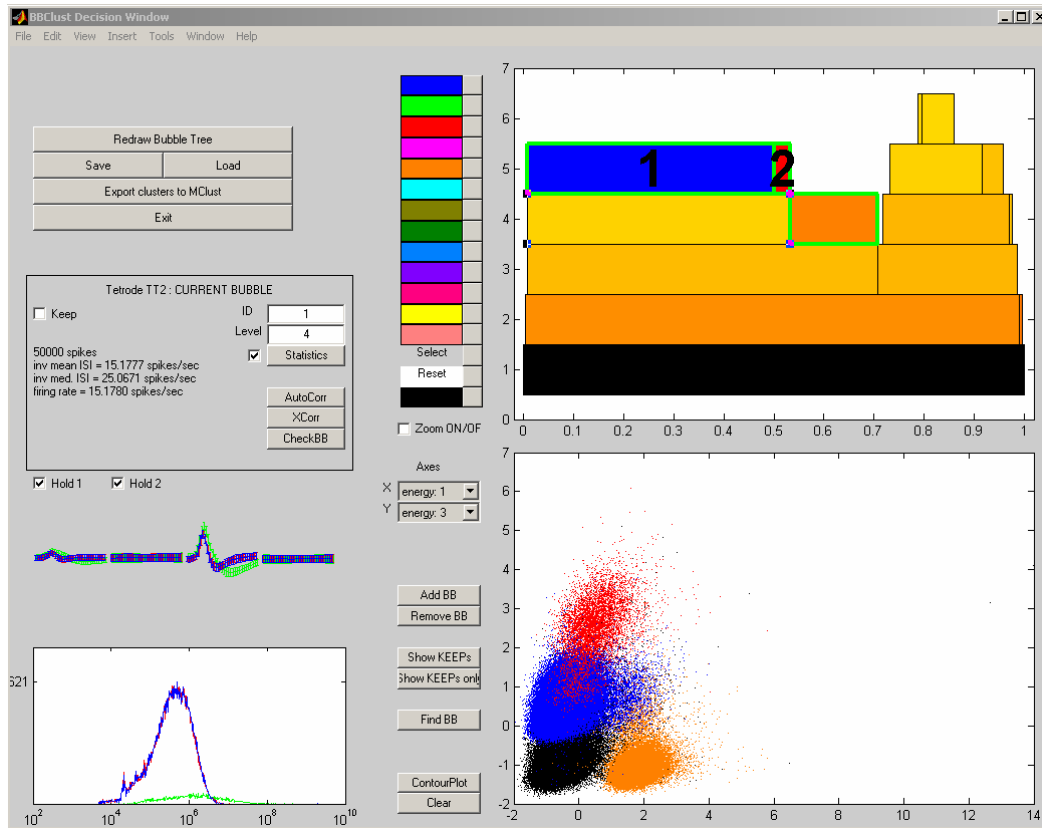
**Figure 1: The main MClust window.**



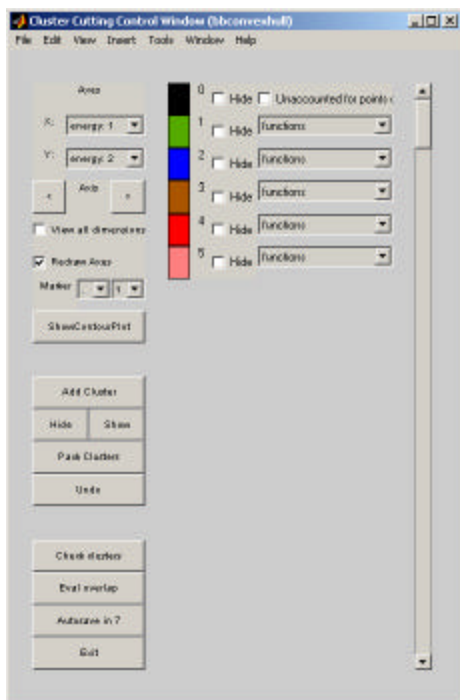
**Figure 2: The KlustaKwik Decision window.**



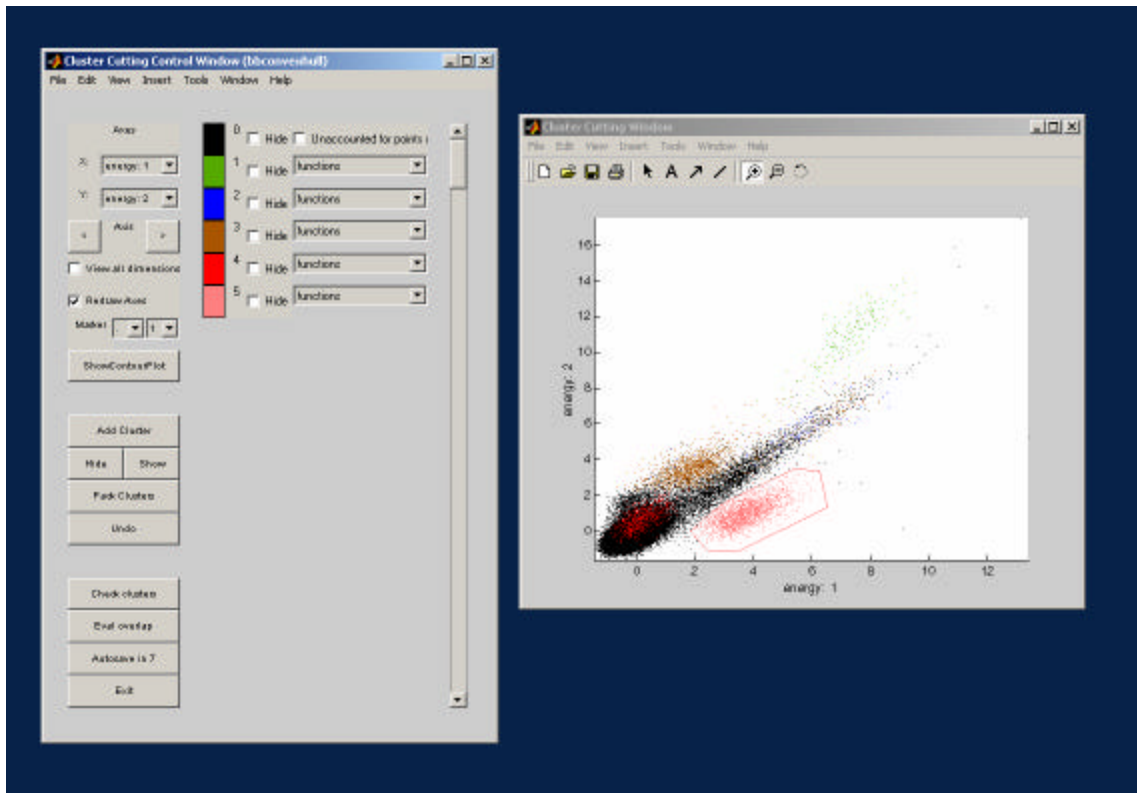
**Figure 3: A KlustaKwik Selection session.**



**Figure 4: The BBClust Decision Window.**

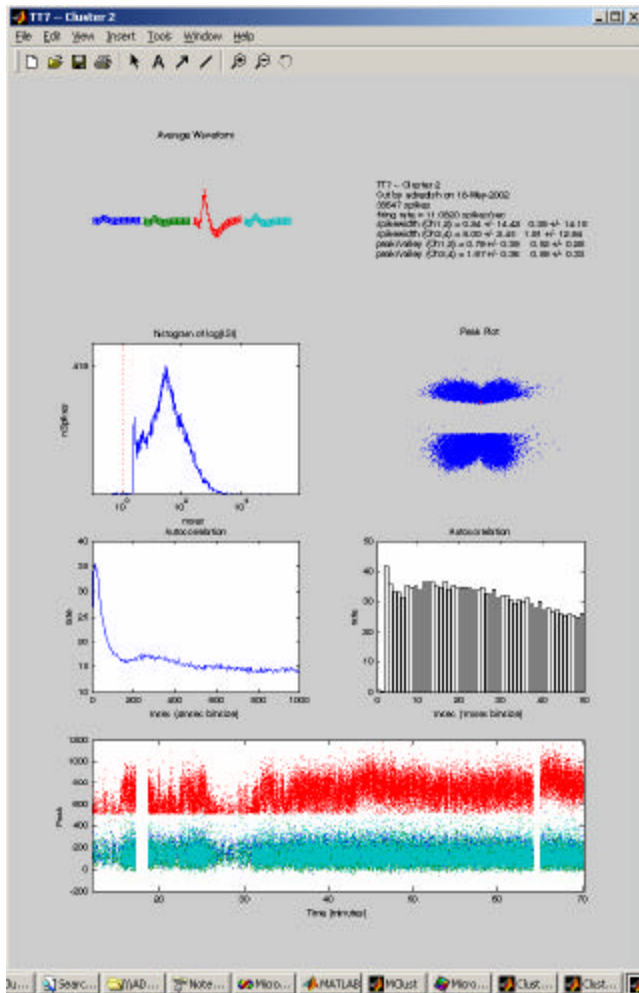


**Figure 5: The Cluster Cutting Control Window.**



**Figure 6: A Cluster Cutting Session.**





**Figure 7: A CheckCluster window.**