MClust

Spike sorting toolbox

A. David Redish
NSMA
University of Arizona
Tucson AZ 85724-5115

Documentation for version 2.0
Released January 2000

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).

MClust is a toolbox which enables a user to perform manual clustering on single-electrode, stereotrode, and tetrode recordings taken with the DataWave™ (DataWave Technologies Inc., Boulder CO) and Cheetah™ (Neuralynx, Tucson AZ) recording systems, including data generated from tetrodes, stereotrodes, and single electrodes. 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. Nothing has to be done to the DataWave™ or Cheetah™ files in order to load them into MClust.

Requirements

MClust is written in Matlab™ (The MathWorks Inc., Natick MA) and requires Matlab™ version 5.2 or higher. MClust has been tested on Sun UNIX workstations running SunOS™ (Sun Microsystems Inc.) and on PC workstations running Windows 95™ (Microsoft Corp.) and Windows NT™ (Microsoft Corp.). It should, however, be portable to any system capable of running Matlab™ with an ANSI-compatible C++ compiler.

The key to fast processing is large amounts of RAM. Typically, 512MB of memory can cut up to 200,000 spikes comfortably; 160 MB of memory can cut up to 50,000 spikes.

Disclaimer

Neither the author, the Neural Systems, Memory and Aging program, nor the University of Arizona 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.

Use of this code should be acknowledged in any paper that uses data analyzed with it. Acknowledgment should be in the methods section as "(MClust, A. D. Redish)".

This code may not be distributed without the express written consent of the author (A. David Redish). This code may not be modified without the express written consent of the author. [MClust is designed to facilitate the incorporation of new features and new cutting methods. If you have such a new feature or cutting method that you wish to include, please contact me. - adr]

Acknowledgments and contact information

This code is copyright © A. David Redish, 1998-2000. Additional code for MClust was written by Peter Lipa and Stephen Cowen. We thank the Neural Systems, Memory and Aging lab at the University of Arizona for support and for extensive testing. ADR was partially supported by a National Research Service Award from the National Institute for Aging.

Send bug reports, questions, and comments to redish@ahc.umn.edu. Neither the author, the Neural Systems, Memory and Aging program, nor the University of Arizona assume any responsibility for replying to email, to fixing bugs, or to maintaining this code. [However, I will reply if I have time. I use MClust regularly in my own work. I will do my best to fix bugs, answer questions, incorporate new features and cutters, etc. - adr]

Installing MClust

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

1. Create an *MClust* directory in the Matlab™ hierarchy.
2. Copy the files from the *MClust* directory on the *MClust CD* into the *MClust* directory that you just created.
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…

Using MClust

1. Start Matlab™.
2. Type MClust at the prompt. This will open the main MClust window (Figure 1).
3. Select the file type you are using. (The file type in Figure 1 is *TT sun*.) Use *TT sun* for files generated with the DataWave™ recording system or with the Cheetah™ SunOS™ system. Use *TT nt* for tetrode files generated with the Cheetah™ NT system. Use *ST nt* for stereotrode files generated with the Cheetah™ NT system. Use *SE nt* for single-electrode files generated with the Cheetah™ NT system.
4. Click on the *Load File* box. This will open a file-selection window from which you can choose the file you wish to cut. When the file has been correctly loaded, the *Load File* box will be checked.
5. Select which features you wish to use to cluster the spikes. Features are moved between the *IgnoreFeature* and *UseFeature* lists by clicking on them. If any features are in the *UseFeature* listbox, the *Choose Feature* checkbox will be checked.
6. Select which channels are valid for this recording. For single-electrodes, all channels but Channel 1 will be grayed out; for stereotrodes, Channels 3 and 4 will be grayed out; for tetrodes, all channels are available. Channels which are not checked will be ignored in further processing.
7. Click on the *Calculate Features* checkbox. When this box is checked, you are ready to cut clusters.
8. Click on *Cut: Convex Hulls*. This will open the Cluster Cutting Control Window (Figure 2).
9. Cut your clusters.
    a. Click on *Redraw Axes* to open the Cutting Window (Figure 3).

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 a 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 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.

10. Click on *Write Files* to save the processed clusters.

11. 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.

Components

Algorithms

MClust first defines *features* of the spikes and then selects them within the *cutter*. Within the cutter, two-dimensional projections of the high-dimensional space defined by the features can be projected and convex hulls defined within those projections. A cluster consists of the set of points falling within the intersection of the convex hulls on all two-dimensional projections. Up to 100 clusters can be cut from a single session.

File types

TT sun — tetrode files generated with the DataWave™ or Cheetah™ SunOS™ systems.

TT nt — tetrode files generated with the Cheetah™ NT system.

ST nt — stereotrode files generated with the Cheetah™ NT system.

SE nt — single-electrode files generated with the Cheetah™ NT system.

Features

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

Currently available features

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.*

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*.

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*.

sw — 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 *sw* can be negative. In the cutter, this feature will appear as *sw: Ch*.

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

wavePCAE — principle components of the waveform. Returns the first five principal components of each channel of each spike. Produces five parameters per valid channel per spike. In the cutter, this feature will appear as *wavePCx: Ch*, where *x* is the principal component (ranging from 1 to 5).

Cutting clusters with convex hulls

The cluster cutting engine consists of up to three windows (Cluster Cutting Control Window, Figure 2; Cluster Cutting Window, Figure 3; Contour Plot Window, Figure 4). 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 a 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.

Additional cluster features

Average waveform — In the functions menu for each cluster, shows the average waveform for that cluster.

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.

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

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.

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.

Hist ISI — In the functions menu for each cluster, shows the interspike interval histogram for that cluster.

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.

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.

Show info — In the functions menu for each cluster, shows number of spikes within that cluster and on which dimensions boundaries have been defined.

Undo — MClust has a one-step undo.

Variances — In the functions menu for each cluster, lists the variances of the spikes within the cluster on each dimension.

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.

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 (Figure 4) 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.

Using keyboard shortcuts

Within the cutting window, certain keyboard shortcuts have been defined:

c — show contour plot.

n — next projection.

p — previous projection.

r — redraw the current cutting window.

u — undo the last step.

v — view all dimensions. This is a toggle, it actually checks and unchecks the *View all dimensions* checkbox in the control window.

If all clusters but one have been hidden (i.e. hide all clusters then show only one), the following two shortcuts are also available:

a — add limit.

d — delete limit.

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 does not change any parameters in its input or output. All it does it is set some of the MClust selection boxes. It saves the color scheme used for the clusters, the features used versus features ignored, the channel validity, and the file type for input.

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.

Additional code supplied with the MClust package

In addition to the MClust program, this package includes, two C++ programs and two Matlab™ functions which are useful for cutting clusters.

SpikeCount (C++)

SpikeCount reports the number of spikes in an input file. It takes the following parameters.

`-in <fname>` — input file name

`-sun` or `-nt` — use to indicate which input file type is being used. Use `-sun` for DataWave™ and Cheetah™ SunOS™ files; use `-nt` for Cheetah™ NT files.

`-tt` or `-st` or `-se` — use to indicate which electrode was used. Use `-tt` for tetrodes, `-st` for stereotrodes, and `-se` for single-electrodes.

SubsampleTT (C++)

SubsampleTT takes an input file (of any compatible type) and outputs the same type of file only including every n^{th} spike. It takes the following parameters.

`-in <fname>` — input file name

`-out <fname>` — output file name

`-skip <n>` — select every n^{th} spike

`-sun` or `-nt` — use to indicate which input file type is being used. Use `-sun` for DataWave™ and Cheetah™ SunOS™ files; use `-nt` for Cheetah™ NT files.

`-tt` or `-st` or `-se` — use to indicate which electrode was used. Use `-tt` for tetrodes, `-st` for stereotrodes, and `-se` for single-electrodes.

SplitTT (Matlab™)

SplitTT applies a set of clusters to an input file (of any type) and creates a new input file for each cluster and a new input file for the spikes not in any cluster. These files are fully normal input files and can be used to cut clusters appropriately. *SplitTT* is particularly useful for input files with more spikes than your computer can comfortably handle. (See below.)

WARNING: The *wavePCAE* feature is dependent on the entire spike set. It should not be used with *SplitTT*.

ProcessTT (Matlab™)

ProcessTT applies a set of clusters to an input file (of any type) and creates a *.t* file for each cluster. These files are fully normal *.t* files. *ProcessTT* is particularly useful for input files with more spikes than your computer can comfortably handle. (See below.)

WARNING: The *wavePCAE* feature is dependent on the entire spike set. It should not be used with *ProcessTT*.

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?"

There are two options. Both require subsampling your data. Either you can subsample the data and split the data into separate input files which can then be cut directly, or you can subsample the data, cut the subsampled data, and then apply those clusters to the complete file.

So either (using *SplitTT*):

1. Subsample the data using *SubsampleTT*.
2. Cut the subsampled file into manageable pieces. It is okay here to have more than one cluster in each piece, but it is important not to have your cluster boundaries cross any clusters.
3. Save these "clusters" into another file such as *split.clusters*.
4. Use *SplitTT* to split the original file using the *split.clusters* file.
5. Cut each separate file as if it were a normal input file.

Or (using *ProcessTT*):

1. Subsample the data using *SubsampleTT*.
2. Cut the subsampled file as if it were the complete file.
3. Save these "clusters" into another file such as *split.clusters*.
4. Use *ProcessTT* to apply the *split.clusters* file to the original file.

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.

Other platforms

"I have LINUX on my PC. Can I still run MClust?"

MClust has only been tested on PCs running Windows 95™, PCs running Windows NT™, and on UNIX machines running SunOS™. It should run on any machine that can run Matlab™ 5.3 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.

Example screen-shots

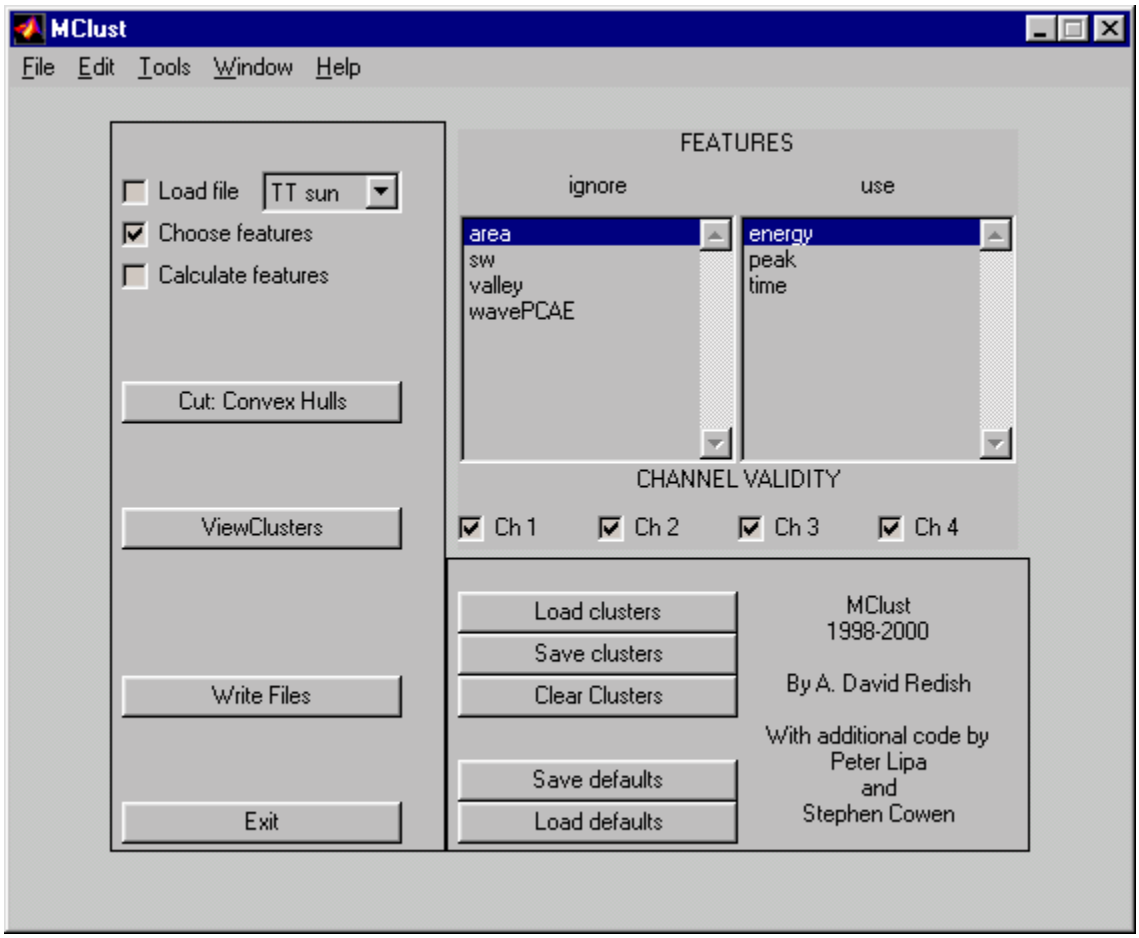

Figure 1: The main MClust window.

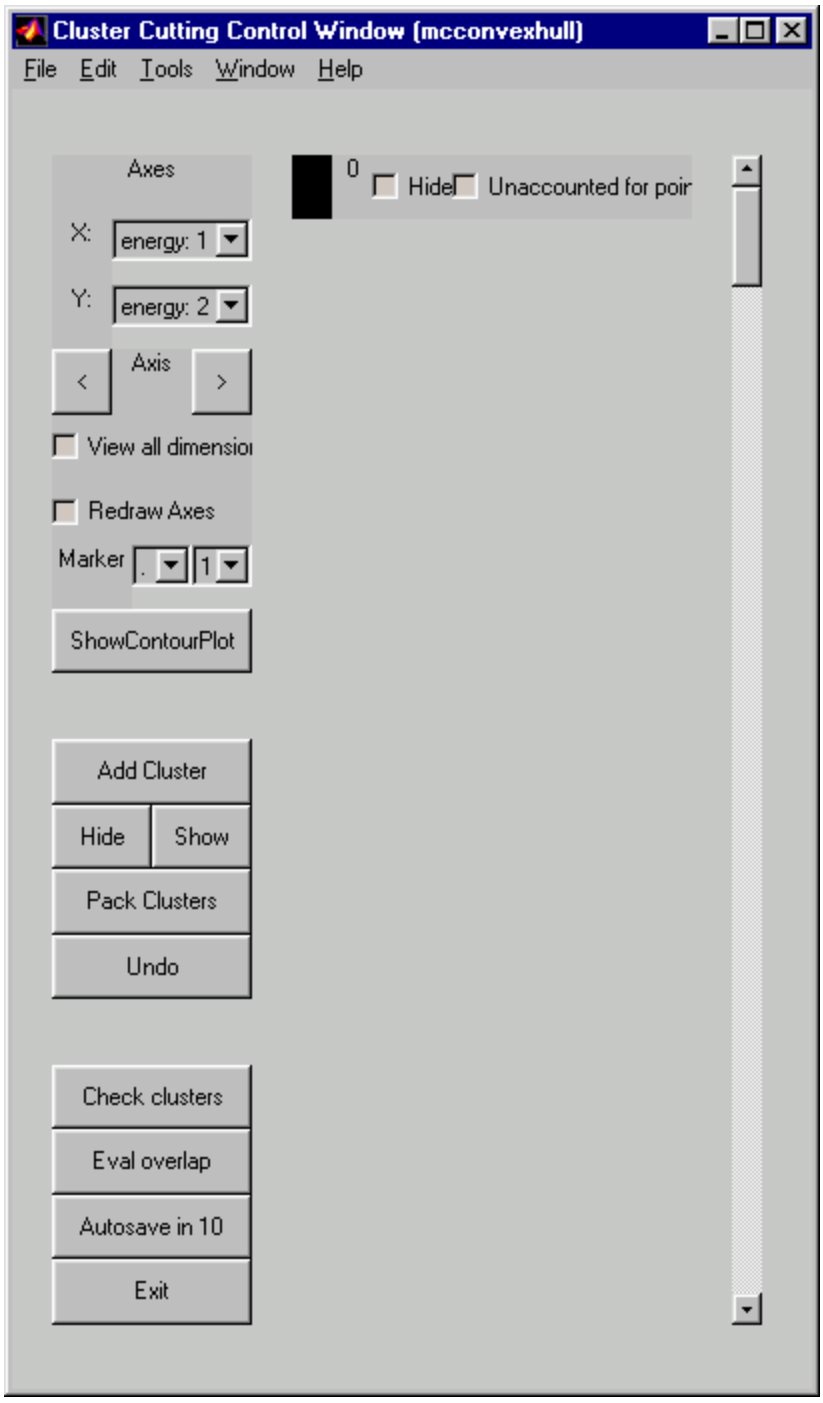

Figure 2: The Cluster Cutting Control Window.

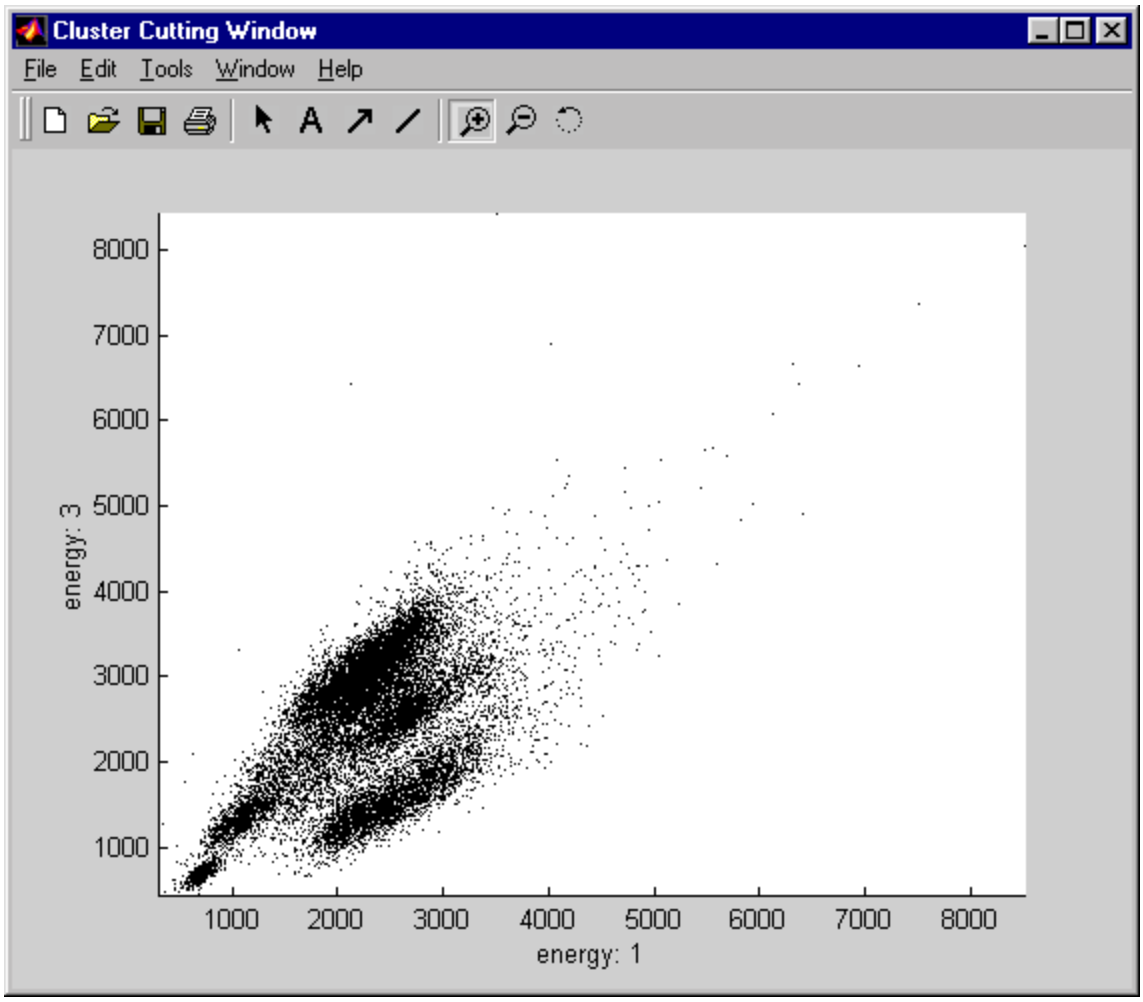

Figure 3: A typical Cluster Cutting Window.

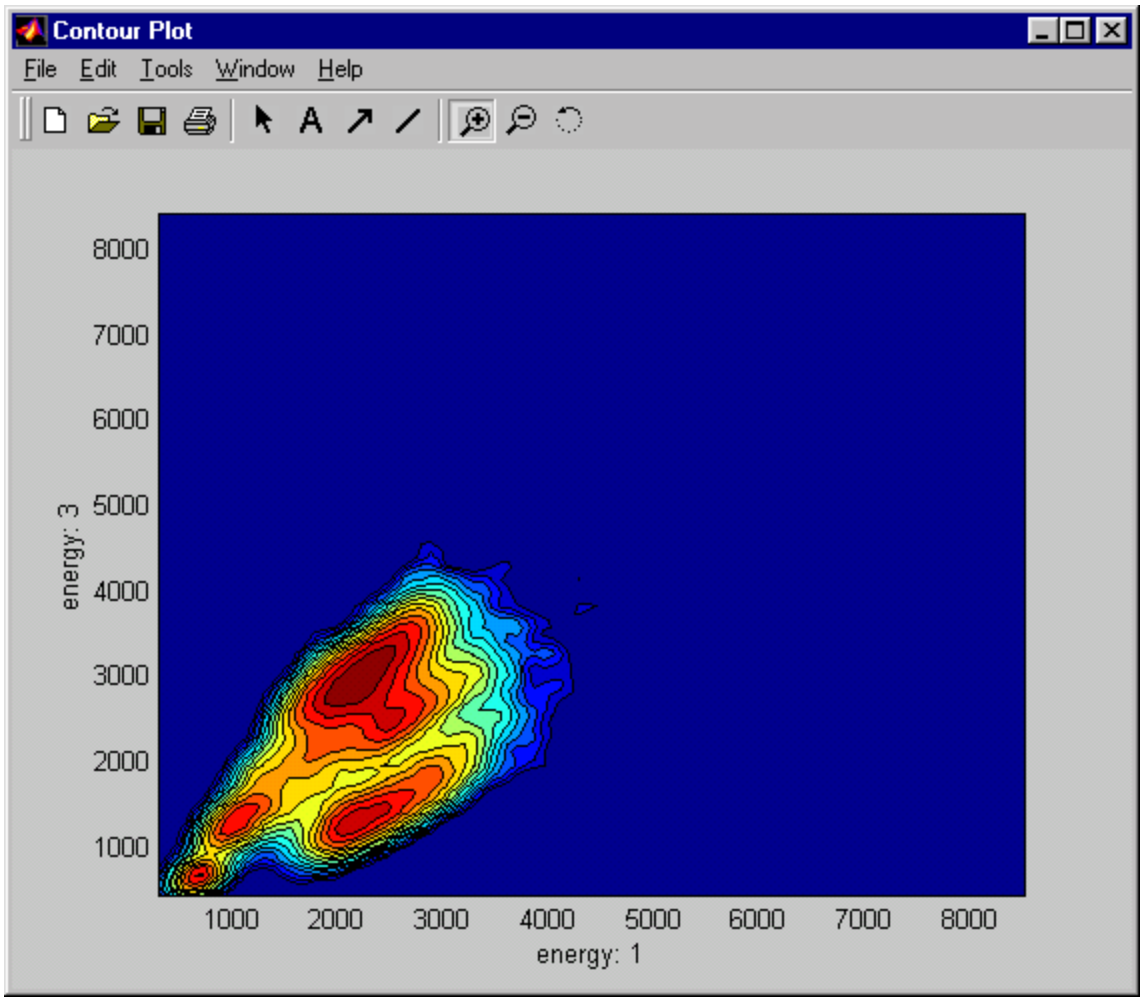

Figure 4: A typical Contour Plot window.