Contents

UDFAnalysis Examples

November 12, 2009

1 Introduction

This document contains various examples of the usage of the **UDFAnalysis** software package. These illustrate how to setup plots within the parent plot window, how to access data, how to use functions to manipulate data, how to plot data, and how to define plot labels. The UDFAnalysis menu file used to generate the output associated with each example and any required data files are found in the **Appendix**. These can be downloaded separately and the examples run locally. Data being input from UDF/IDFS data sets should and autopromote when needed and are not found in the Appendix. Some examples, however, use input from ASCII based files, particularly from files containing plasma moments generated by the MOments program and these need to be downloaded. They should be placed in the same directory which the menu files are downloaded. If the data file is used in multiple examples you only need to download it once.

Each example consists of a brief description of the important options used together with a figure showing the generate output. Discussion of option settings assume that you have downloaded the example menu file and can view the option settings in the various definition menus.

2 Plot Layouts

The following set of examples show how to set up plot layouts. None of these examples make use of any external data. The examples show how to define basic sets of plots (and some not so basic) and how to annotate them. Remember, to change the option settings for the layout of any plot you need to highlight the plot, click on edit, make the changes, and click on accept. At this point you can click on **RUN** in the Main Menu to see the effects of the changes.

2.1 A Basic 4x1 Plot Grid

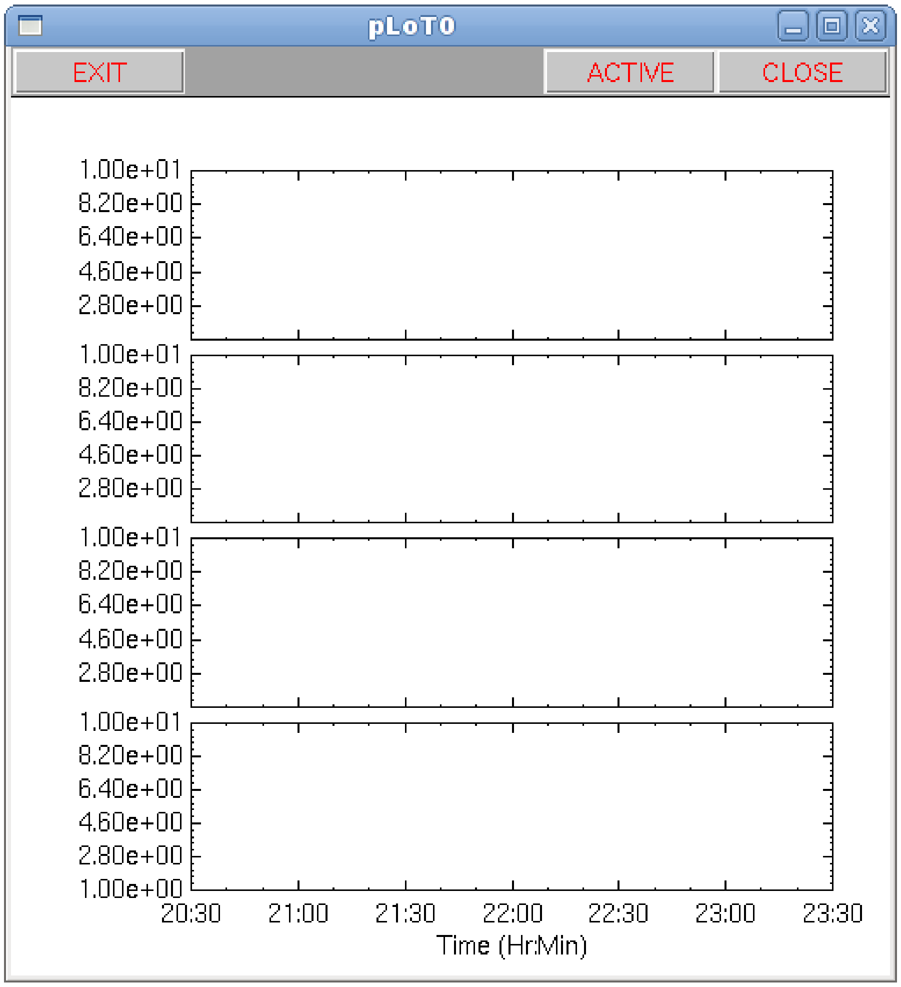


Figure 1: 4x1 plot grid

Figure 1 shows a column of four plots. Each plot is fully defined within the **Plot Layout** Menu. The four plots are all placed in a single plot column and in successive plot rows. The small gap between the plots is produced by setting the Top X gap of each plot to 1.0. This includes the setting a gap for the top plot even though there is nothing plotted above it. Leaving it at 0 would make this plot extend slightly higher in Y than the lower three plots. The right Y axis annotation has been turned on in all plots and the **Omit Number** option for that axis in the top three plots is set to **MIN** which causes the lowest numerical label to be dropped. This prevents the lower labels of one plot from interfering with the upper label of the plot below it. This problem could have been avoided by using a larger gap between the plots. Only on the lower plot is the bottom X axis annotation set to **YES**.

There are several options which can be explored using this menu. You can change the annotation color, change what annotation is being output, change the axis scaling, and the axis min and max values. (Since no data is being plotted the plots use the axis values specified in the plot layout menu.) If you add annotation along the right-hand axis of a plot you will find that it will shrink in X relative to the other plots. For all the plots to have an identical X axis length each would have to be annotated along the right-hand Y axis. Of course if you didn't want show the annotation you set the **Omit Number** option for that axis to **ALL**. One last thing to try is to change the row of the bottom plot from 4 to 5. This creates a 5x1 plot grid with an empty grid at row 4. Many of these options will be exercised in some of the examples below.

2.2 A Basic 2x2 Plot Grid

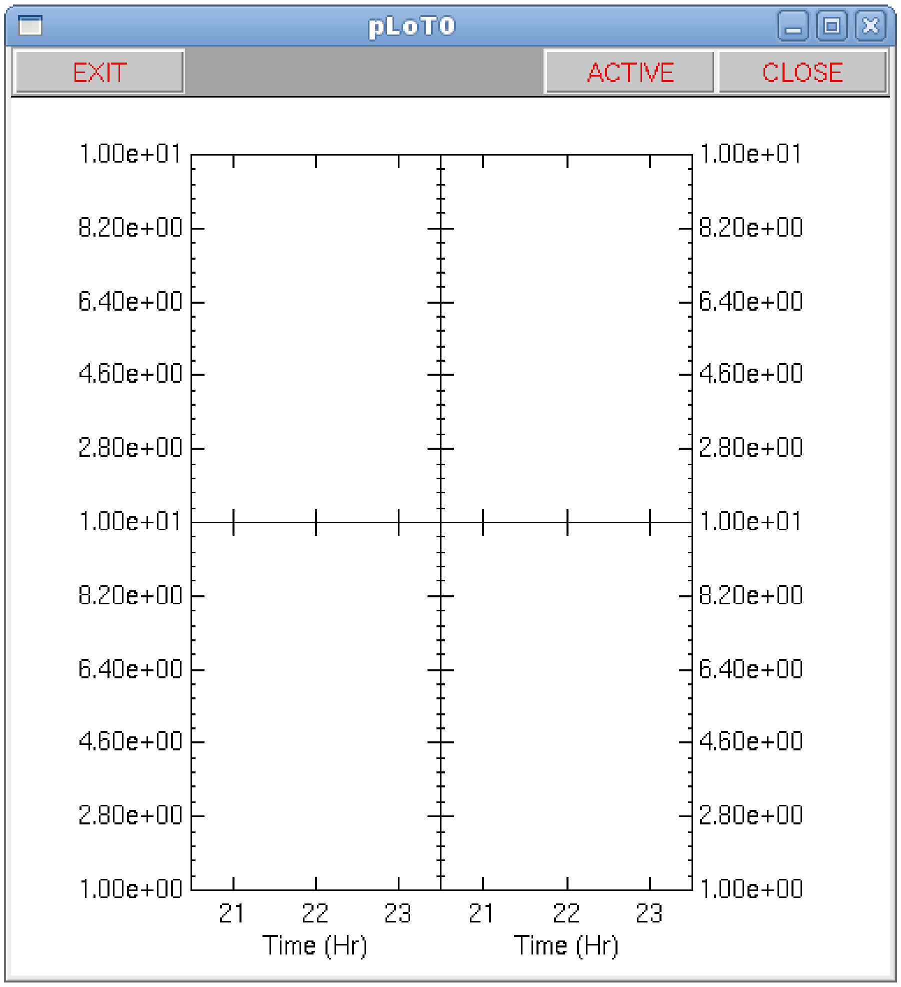


Figure 2: 2x2 plot grid

Figure 2 shows a 2x2 grid of plots. Each plot is fully defined within the **Plot Layout** Menu. The plots are 4 placed in a 2 by 2 square with plots P1 and P3 output in the first column of the plot grid and P2 and P4 output in the second. All of the plot gap options have been left at 0.0 causing the axes of adjoining plots to abut one another. Annotation on the plots whose Y axes which lie on the left and right edges of the plot grid have been turned on as has the annotation on the the plots which define the lower X base of the plot grid. The

Omit Number option for the Y axes of the plots in top row are set to **MIN** to keep their lower annotation labels from overwriting the upper ones on the lower plots.

2.3 Spanning Rows and Columns

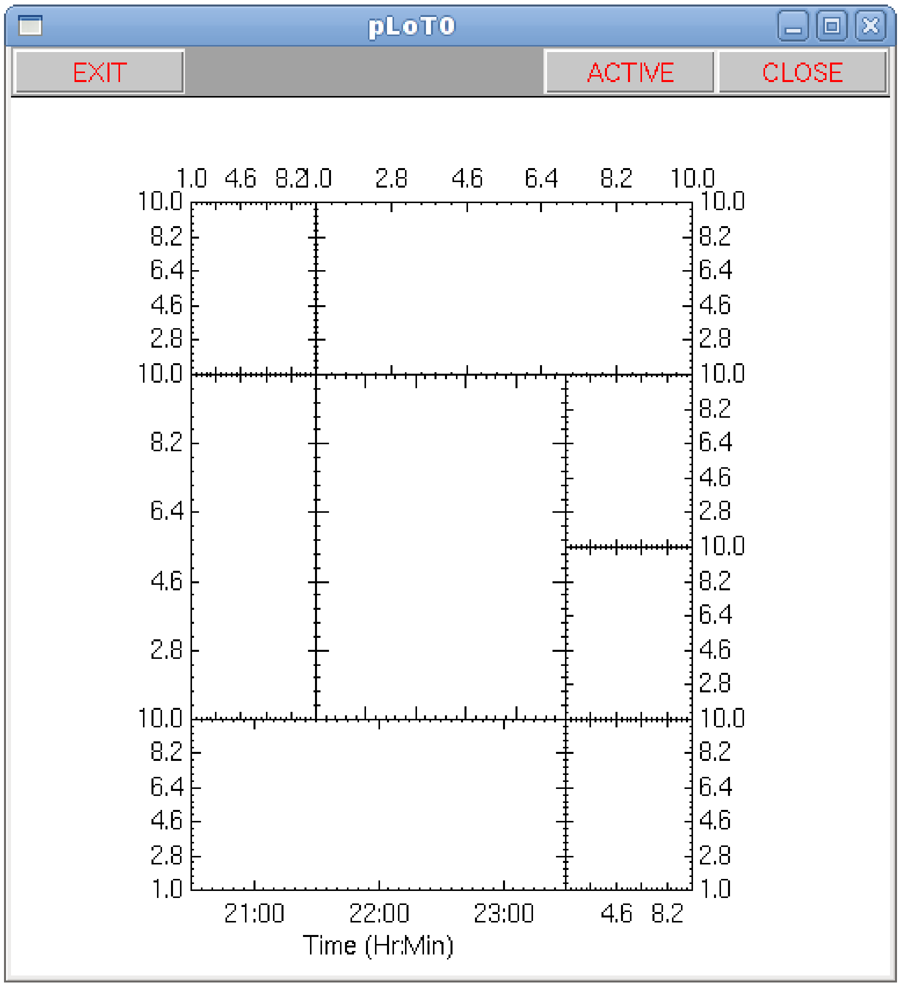


Figure 3: Plot grid with spanned rows and columns

Figure 3 shows a mixed bag of plots defined in an overall 4x4 plot grid. While this is probably not usable plot configuration it does illustrate how to make plots span rows and columns within a grid.

While all cells within a plot grid have approximately the same dimension, the plots defined in the grid can have different dimensions. This is done by having a plot span more than one grid row and/or column. The Plot Layout defines 4x4 plot grid since the largest outer most column and lowest row occupied by a plot is 3 and 3 respectively.

Much of the pertinent layout information for this example seen in the plot layout text window. Both P1 (the upper right plot in the layout) and P7 (the lower left plot in the layout) span 3 columns in the plot grid; P3 (the middle plot on the left) spans 2 rows in the plot grid; and P4 (the center plot) spans 2 columns and 2 rows. There are several other things to note in this example. With the exception of P6 all of the plot have been defined to be non-time based. P6 has a time based annotation along its X axis while the other annotated X axes show the range set in their **Minimum** and **Maximum** axis settings. The annotation format for all annotated axes has been set to **%.1f**. To prevent overlap with annotation on a adjoining axes, some of the annotated axes have had the minimum or maximum annotation dropped.

2.4 Empty Plot Cells

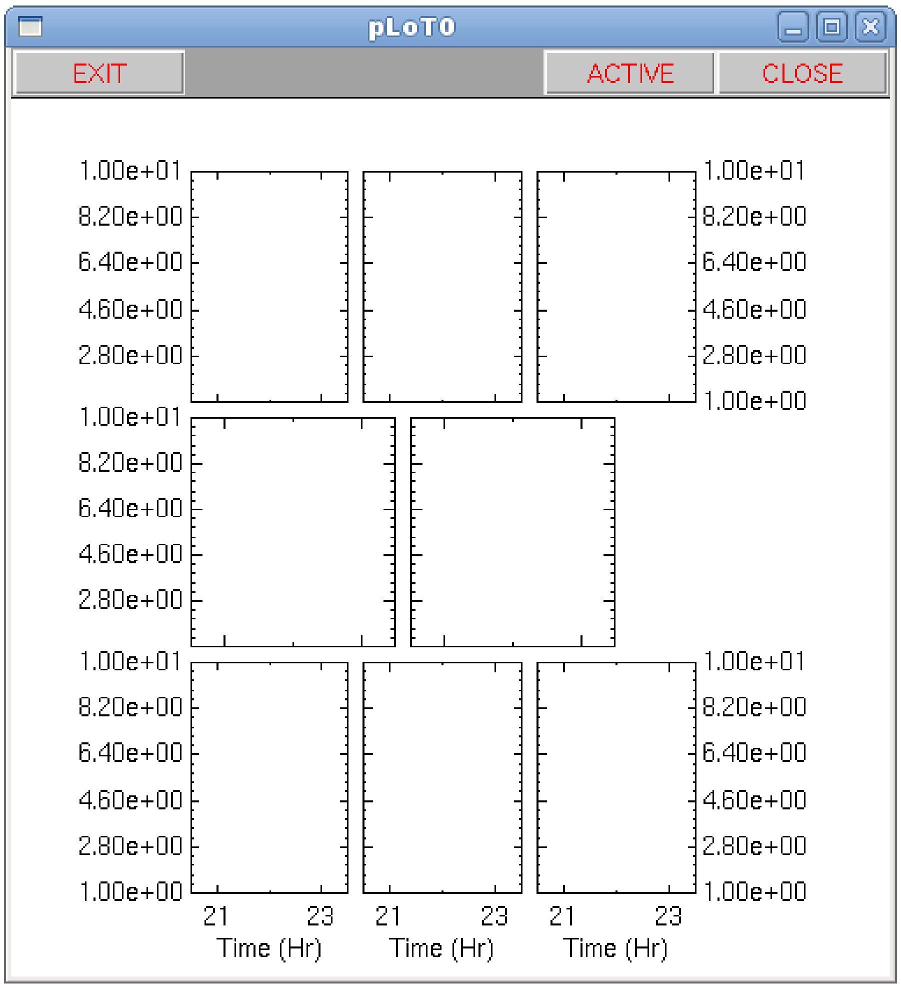


Figure 4: Plot grid with an empty plot cell

Leaving plot cells within the plot grid empty can sometimes produce unexpected results as seen in Figure 4. This is a simple 3x3 plot grid in which the cell at Row 2, Column 2 has no plot definition associated with it. The problem is that the plots in the middle row do not now align with the plots in the row above and below it.

Why this happens is due to the way the plot cell widths are computed. A very simple explanation (not generally correct but is in this case) is that cell widths in a given row are determined by dividing the available space in a row by the number of cells in the row. The space available is the unused space **after** space for all labels has been accounted for. Because the middle row has only to account for one set of Y axis labels while the other two rows need to account for two sets of labels, it has more space to allocate to its plot cells. As a result the plots in the middle row are wider. It should be noted that we would get much the same effect if a plot had been defined in the empty plot but its right Y annotation turned off.

To force alignment in the case of the empty cell is to fill the cell with a dummy plot with its right Y annotation turned on. This is shown in the next example. The solution when a plot is defined in the cell is to turn on it right Y annotation and set the **Omit Numbers** for that axis to **ALL**. This in essence leaves room for labels which are then not output.

2.5 Dummy Plots

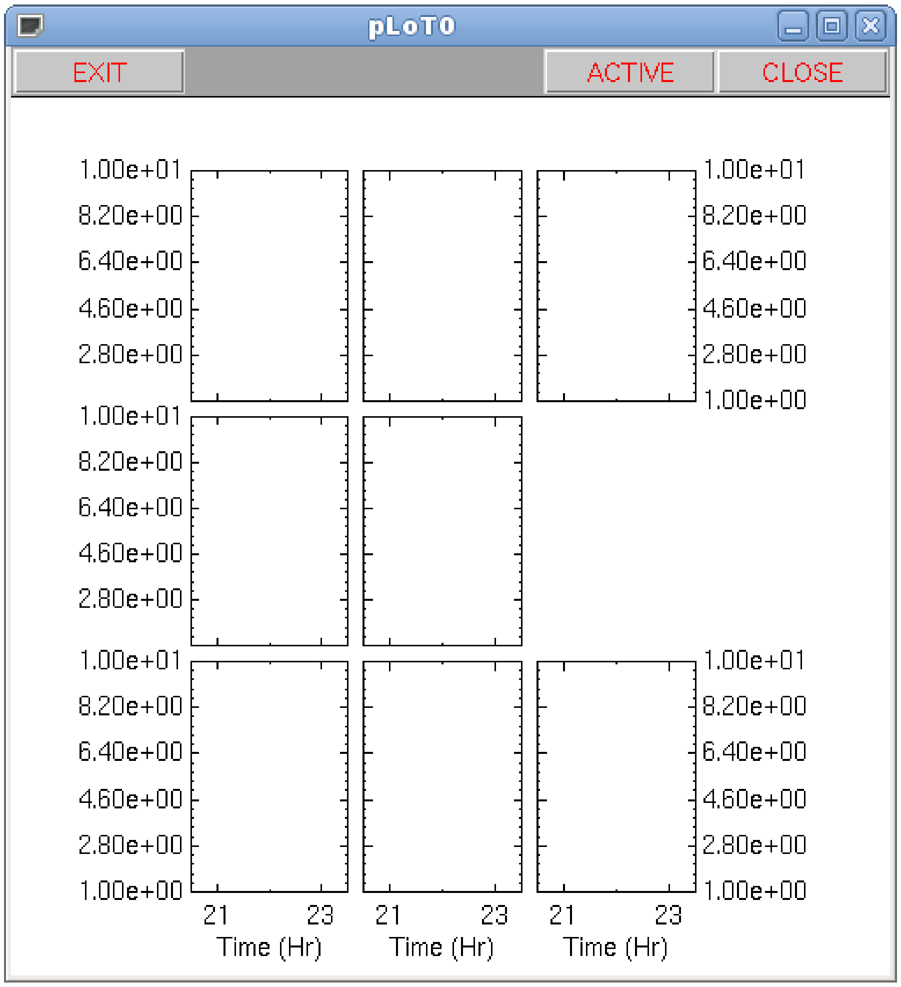


Figure 5: Plot grid with dummy plot added

Dummy plots are used as place holders in a plot grid where empty cell(s), as in the above example, cause unwanted misalignment in the displayed plots. Figure 5 shows the same plot layout as in Figure 4 but with a dummy plot placed in what was the empty plot cell.

The dummy plot is the last entry in the plot layout text window. Dummy plots always have a **Plot ID** of **_DP_**. Editing the dummy plot shows its option settings in the work area. What is important to note is that the dummy plot has its right Y axis being annotated. This second set of defined Y axis labels in this row gives it the same available space for the plot cells as for the top and bottom rows. With identical cell widths, all the plots align. If the right Y axis annotation were turned off in the dummy plot the resultant plot layout would look the same as if it had not been defined.

2.6 Negative Plot Gaps

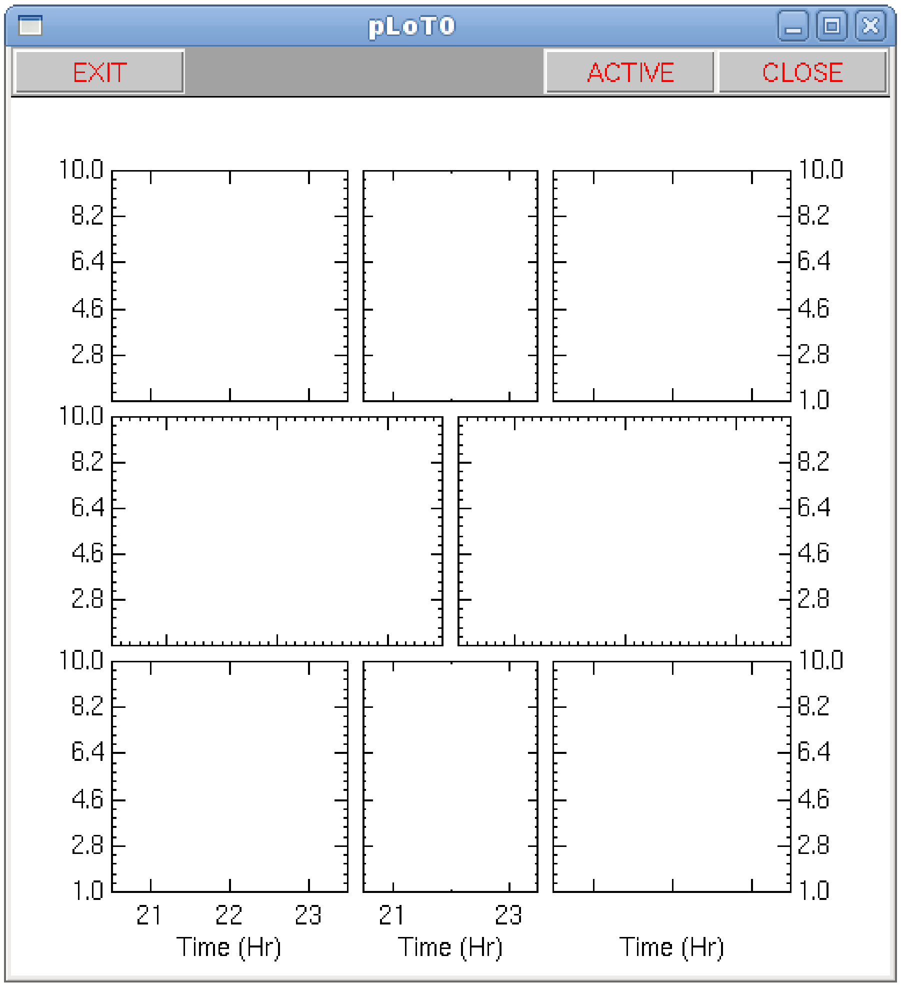


Figure 6: Plot grid with plots expanded using negative gaps

One of the uses if the axis gap options is to allow a plots to expand into blank or unused space within a plot grid. Figure 6 shows a 3x3 plot grid in which the center plot is not defined. It then uses a negative plot gap to pull the Y axes of the two plots on either side of the unfilled cell into the vacant area. In addition using a **%.1f** format for the numerical annotation along the Y axes leaves more room than necessary along the outer Y boundary of the plot grid (program budgets space for Y axis labels to fit a format of **%.2e**) and a negative plot gap is applied to all of the outer Y axes to recover some of this space.

Both of the negative gap applications are seen in the plot layout definitions. All of the plots in column 0 in the plot grid have a gap of -5 associated with their left Y axis. This pulls those axis 5 characters to the left to recover some of the extra space created by using a **%.1f** annotation format. Keeping the gap the same for these plots keeps their left edges in alignment. The center left plot has a gap of -5 along the right Y axis to pull that axis to the right to cover part of the hole in the center created by the unfilled plot cell. The same gap is applied to the left Y axis of the right center plot to let it expand into the vacant center cell. All of the plots in column 2 have a right Y gap of -5 to recover some of the extra space allowed for the default annotation format.

2.7 Layout Options - Part 1

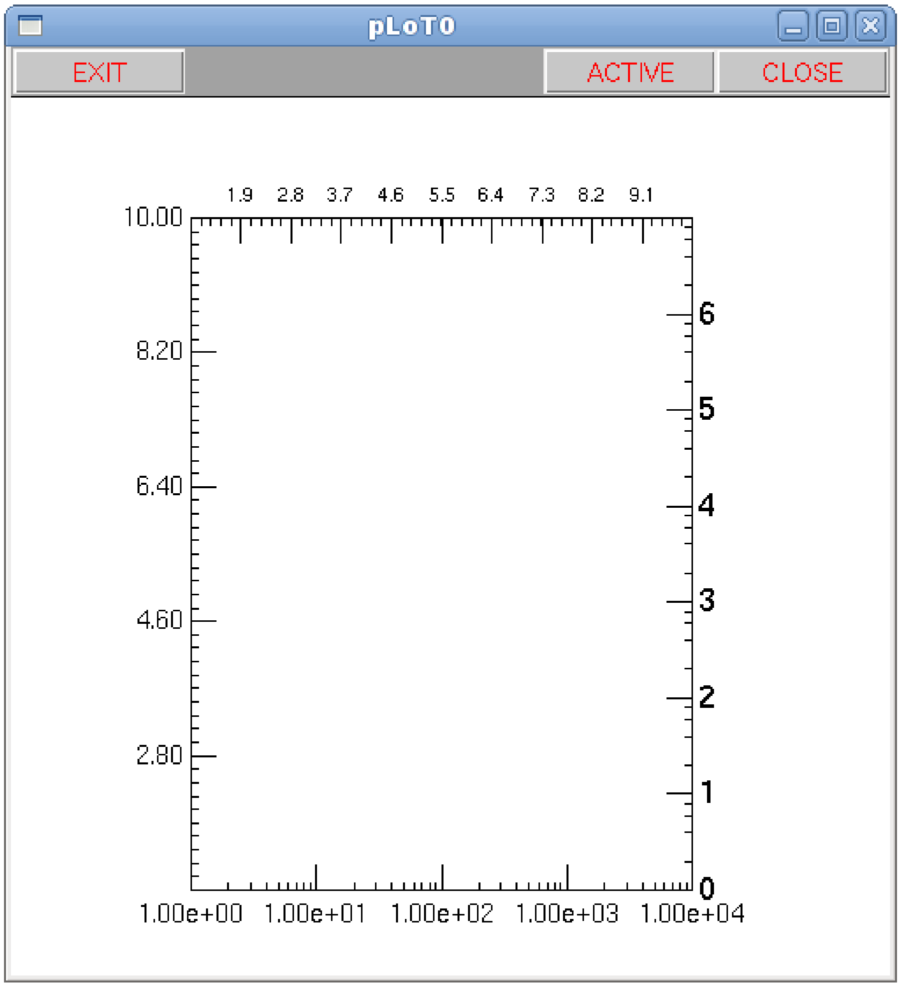


Figure 7: Plot illustrating various Axis Options

There are a number of options which deal with the plot axes and the associated numerical annotation. Figure 7 illustrates the effects of several options by applying them to the four axes in the displayed plot. The example shows the effects of changes in **Scaling**, **Major** and **Minor Ticks**, **Number Format**, and **Text Size** options.

2.8 Layout Options - Part 2

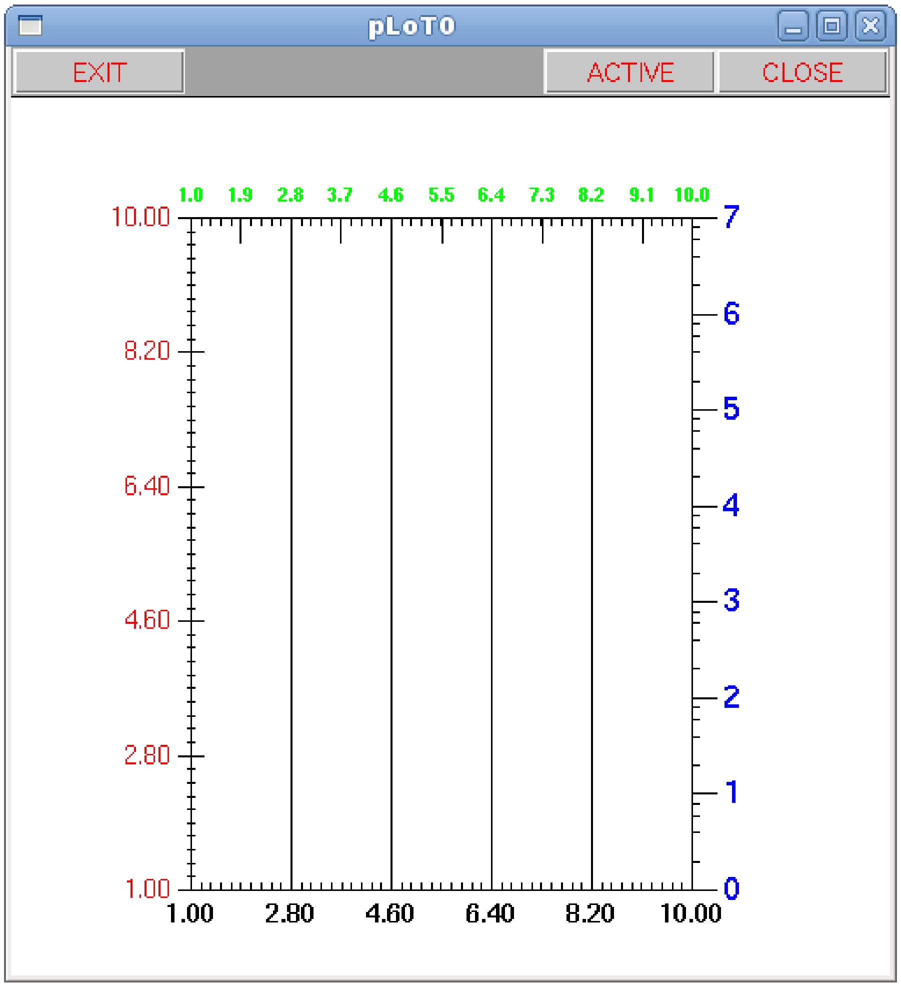


Figure 8: Plot illustrating various Axis Options

Figure 8 shows the same basic plot layout as shown in Figure 7 but shows the effects of changes in the options **Number Color**, **Tick Format** and **Bold Number**. The tick format options apply to both the major and minor tick marks except **SPAN** which applies only to the major tick marks. In the latter case the minor tick marks always assume a format of **INSIDE**.

2.9 Grids and ColorBars

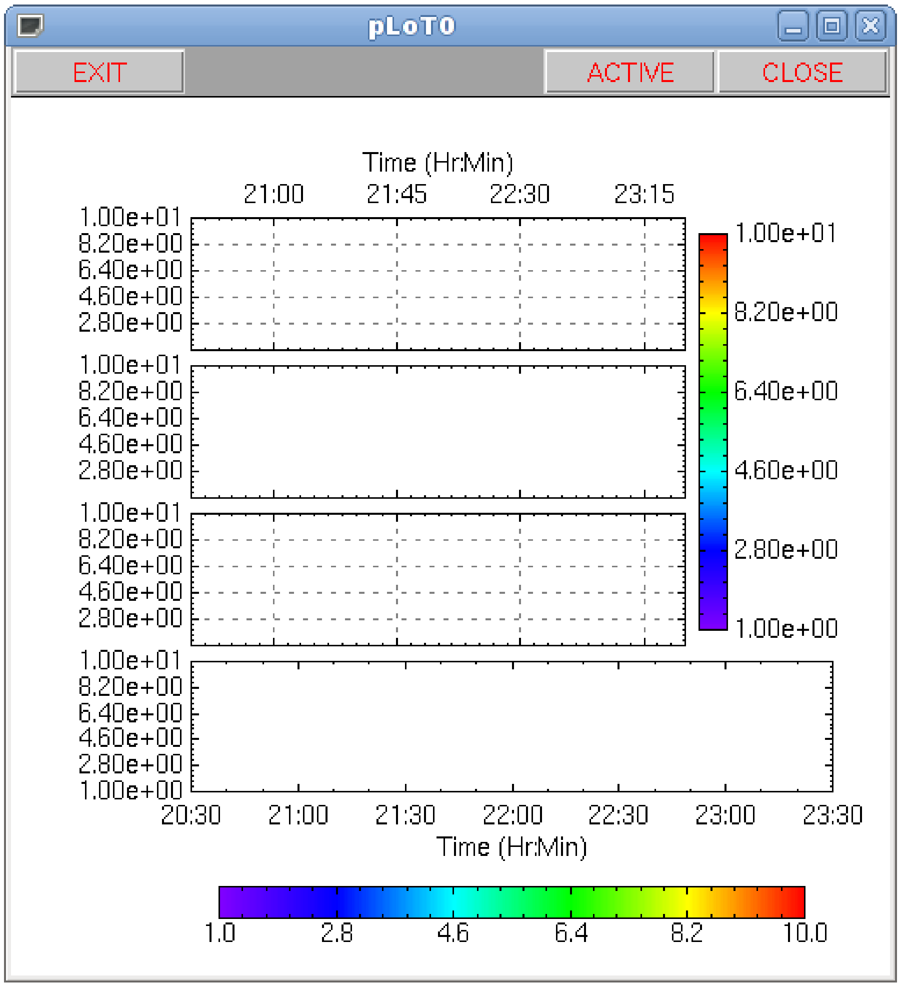


Figure 9: Plot illustrating Color Bar and Grid Options

Adding a colorbar and/or a grid to a plot is done within the plot layout menu. Figure 9 shows the addition of two colorbars as well as grids within two plots.

Plot grids have been added to the plots P1 and P3. This is done by setting the **Grid** option to **YES** in those plot definitions. The grid characteristics (color, line style, direction, overlay or underlay) are set globally in the **Plot SetUp** menu. Setting the Show Colorbar option to YES in P1 adds a colorbar to the plot. The colorbar placement was to the right of the plot and its orientation was vertical. Setting Span to 3 allows the colorbar to output against the first three plots in the column. The colorbar characteristics (annotation, etc) were set under the **CBar** axis options.

The colorbar on the lower plot (P4) was set to be output below the plot and with an orientation of RHORIZONTAL. The format was changed from the default %.2e to %.1f.

If you want all of the plots to align within the grid they you would specify the colorbar span in P1 as 3,1 rather than 3. The ,1 adds room for the color bar to the next row without it being output in that row.

3 Plot Setup

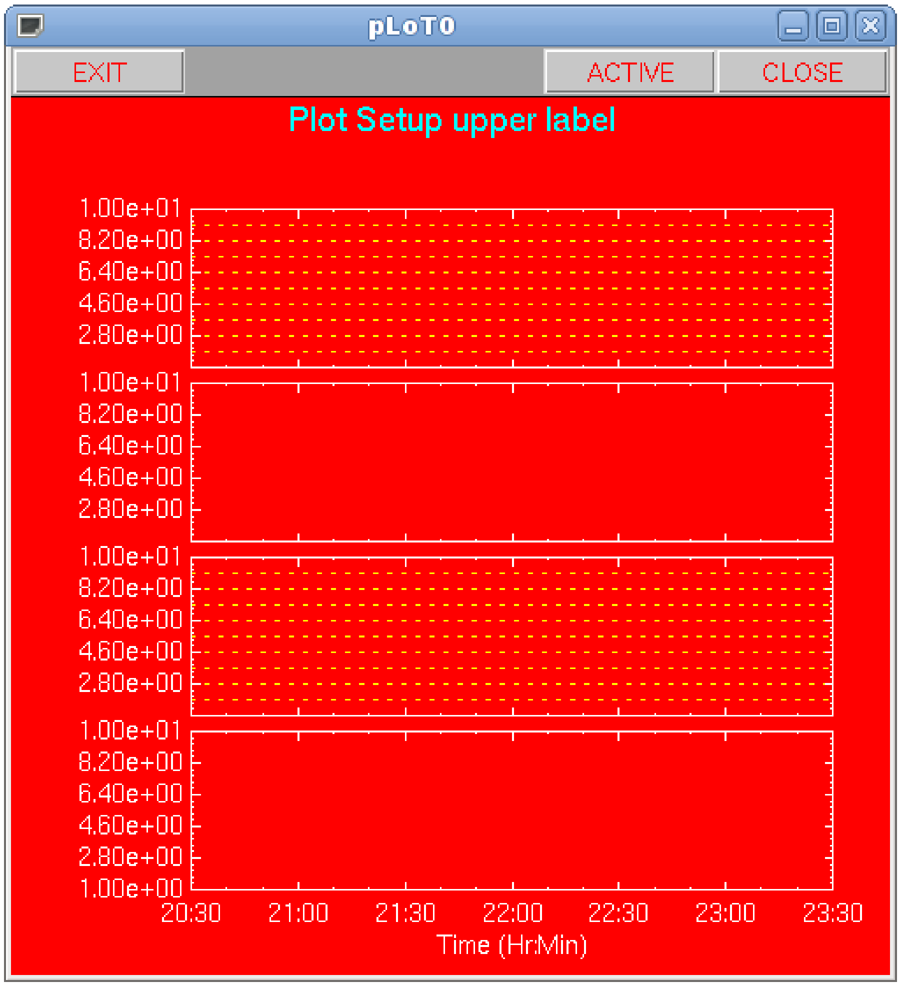


Figure 10: Plot Setup Options

The plot setup menu contains several options which can effect the overall plot display. These include the ability to set the background color, define the plot grid characteristics, and add an overall upper label to the plot window. Figure 10 shows the effects of several of the options including: adding an overall top label, setting the background of the parent plot window and the grid definition.

There are two things to notice in the example. The first is that the top window label defined in the plot setup menu is centered with respect to the plot grid and not to the plot outlines. The second is that the program internally checks to see if any item is being output in the background color and if so will use the complement of that color. That is why the red top label is output in cyan on the red background.

4 Labels

The following examples show how to add labels to and to manually move labels in a plot. These examples use the same plot layout shown in Figure 8 but with the tick format on the bottom axis set to INSIDE rather than SPAN.

4.1 Basic Labeling

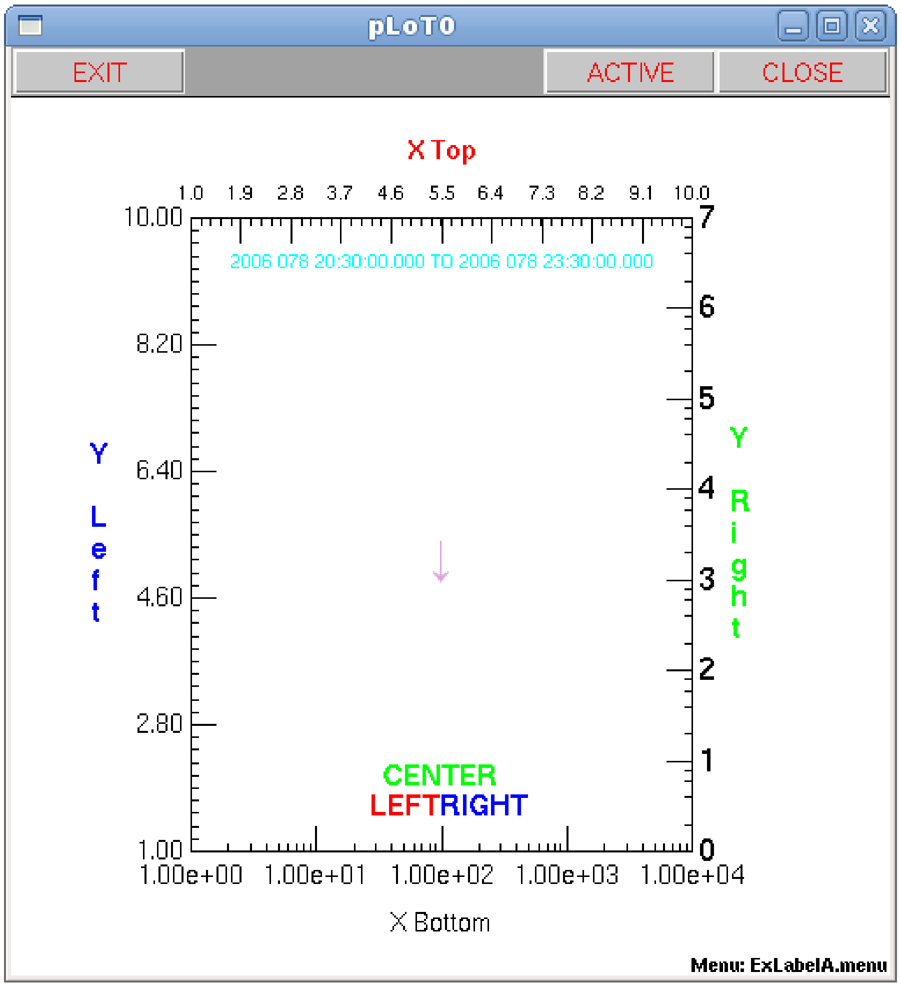


Figure 11: Basic Labeling of a plot

This example shows how to add and position labels within a plot as well as how to include unicode characters and internal variables in a label. The labels are defined in the **PLOT LABELS** menu. A subset of the option settings for each label definition are shown in the text window but to see the full set of option settings hilight the label entry and the click on **EDIT** to copy the settings into the work area.

The first four labels are standard axis labels. Each is centered on its respective axis and output exterior to the plot box. The Y axis labels have an X offset of 1.5 characters and the

X axis labels have a Y offset of 1.5 characters. The offsets move the labels away from their respective axes in the direction indicated by the **Pos** option.

The label in slot 4 shows how to include an internal variable in a label. The variable $apANS(miFile) contains the name of the input menu file. The label is output with a point size of 8 in the lower right-hand corner of the plot window.

The label in slot 5 shows the use a predefined label (**_SE_TiMe_**). This label holds the start and stop time entered by the user in the Time Definition Menu. The label is output inside its anchor axis.

The label in slot 6 contains a unicode character, in this case a downward pointing arrow. Note the large point size used as well as the custom color.

The last three label definitions demonstrate the effects of label justification. Each is centered on the lower axis and then justified according to its label. The last label in the group was moved upward 1.5 characters to prevent it from overlapping the lower two. The **LEFT** and **RIGHT** labels clearly show how the anchor point automatically shifts beyond any tick marks and numerical annotation in the direction of output.

4.2 Manually Moving a Label

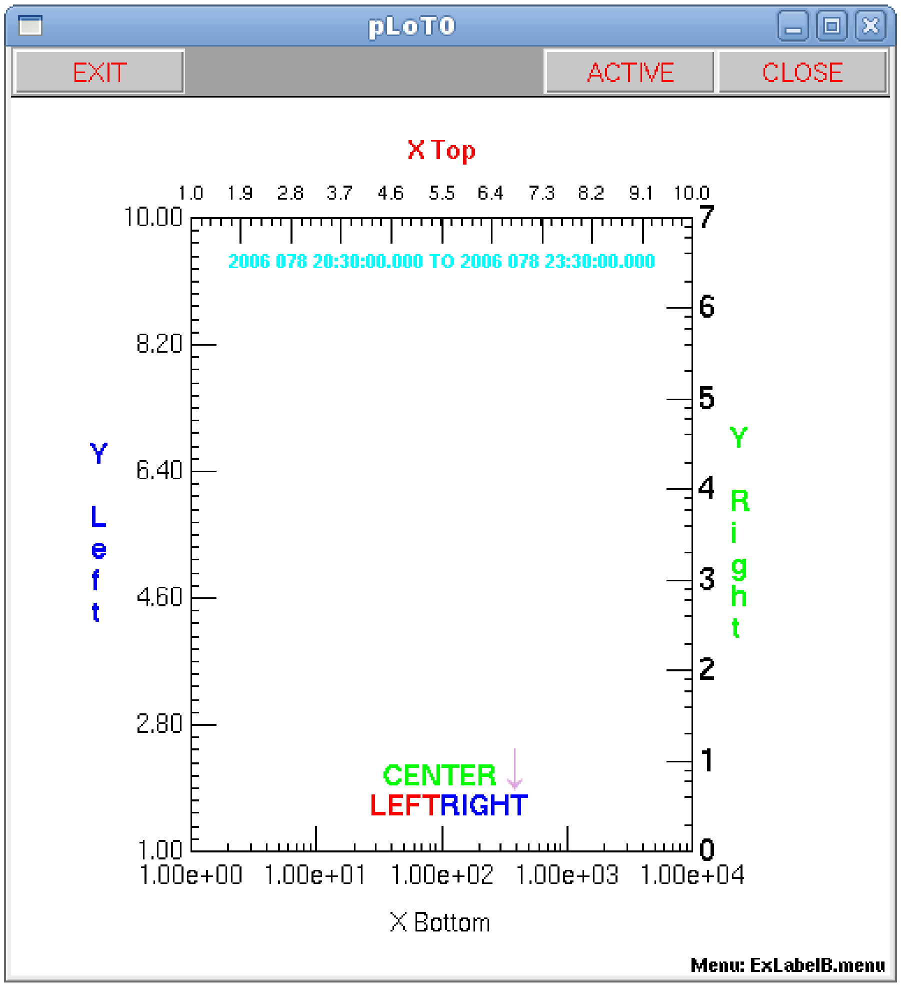


Figure 12: Basic Labeling of a plot

You can manually reposition any label in the plot window by placing the cursor over the label, left click, move the cursor to the position that you want the label moved and then right click. The selected location becomes the new label anchor point and the label is justified to that point according to its **Justify** option setting. This was done to the downward pointing arrow in Figure 11 moving it to above the T is RIGHT as shown in Figure 12.

Moving a label modifies its PlotID, X-Off, and Y-Off option settings in the **Label Menu** as seen in the menu. The PlotID option will be set to **_PW_** which indicates that the label anchor position is given in absolute coordinates in the parent plot window. The new coordinates are given in the X-Off, and Y-Off option settings. When you move a label you need to re-save the menu so that the label will appear in its new position the next time UDFAnalysis is invoked using this menu as input. **Note:** The **Anchor** and **Pos** options are not used when outputting a manually positioned label and the **Axis** option is only used to determine the direction the label is output: horizontal if an X axis is selected and vertical for a Y axis.

It should be emphasized that when a label is positioned absolutely in a window, changing the window size will cause the label to be repositions with respect to the plots in the window. Anchoring a label to a plot is generally a better positioning method since it overcomes the rescaling problem.

5 Plotting

The following examples show how to plot variables and illustrate various plot output options. More examples can be seen under sections which demonstrate how to setup various function calls.

5.1 1D Time Based Plots

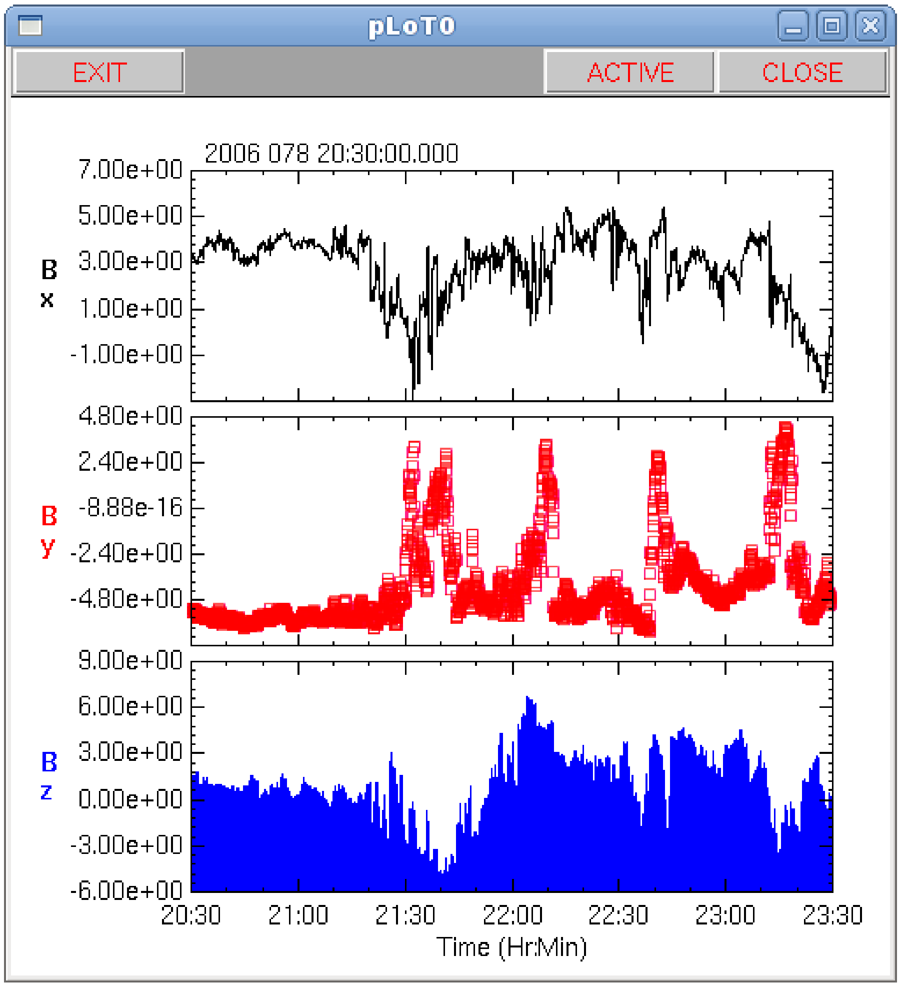


Figure 13: Time plots showing line, symbol, and bar format

Figure 13 shows a set of time based plots. The data plotted comes from the Cluster C1 magnetometer experiment. Any time based scalar variable can be used as input to a time based plot. If the variable has been stored in a time based grid such as the system-wide time grid then only the parameter being plotted against the Y axis needs to be specified. The time information is obtained from the grid.

The data being plotted and the characteristics of the plots are defined in the **PLOT DEFINITIONS** menu. The menu contains three definitions, one being output in each of the three plots set up in the **plot layout** menu. Each plot uses a different output format format and is output in a different color. For all plots the **Grid** option has been left set to **NO** and the autoscale option has been set to **YES**. Hilight a plot definition and then click **EDIT** to see the full set of option definitions for and of the plot in the work area. The plot color for both the top and bottom plot is set in the **Line Color** option while the middle plot color, which is a **POINT** plot is set in the **Symbol Color** option.

5.2 Grid to plot option

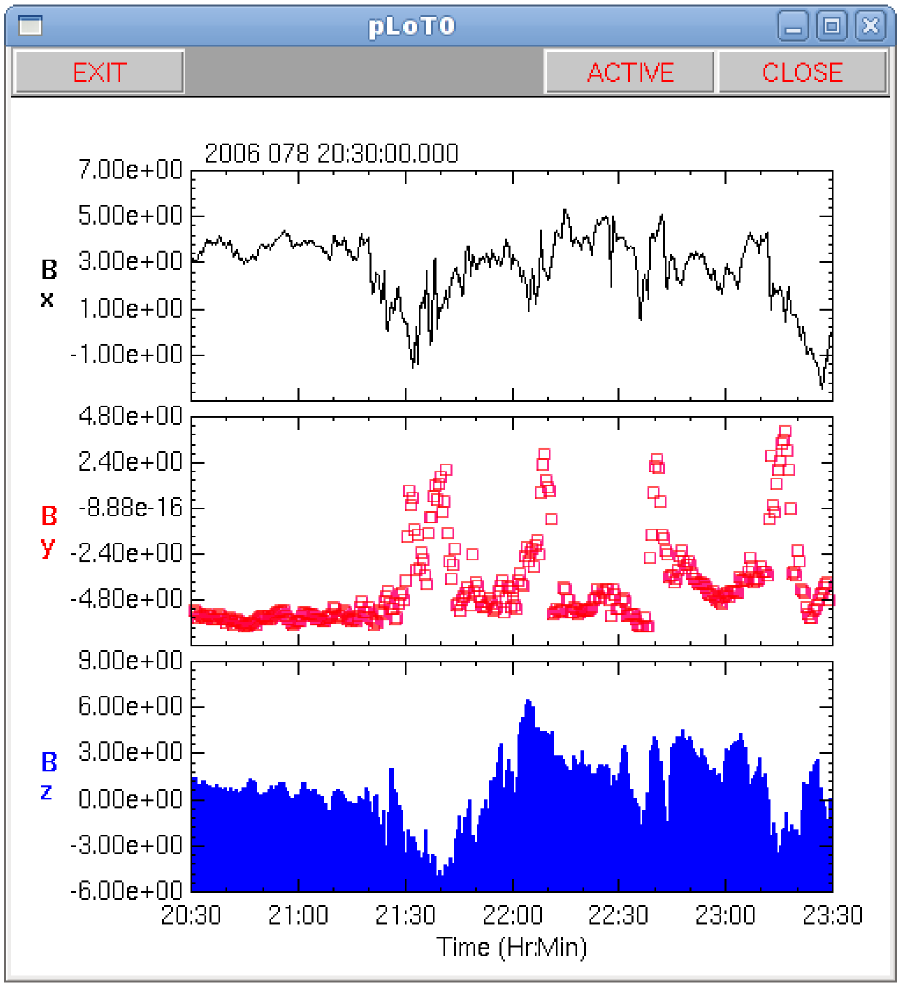


Figure 14: Time plots showing line, symbol, and bar format

Figure 14 shows the same plots as in Figure 13 with the exception that the **Grid** option in each plot definition is set to **NO**. When the **Grid** option in a plot definition is set to **NO** the data being plotted is displayed in the plot at its native time resolution or the time resolution of the time grid its stored in. In many cases there are more data points in the plot than there are pixels across the plot. This can produce noisy looking plots at times since there will be an lot of overplotting. Setting the **Grid** option in a plot definition to **YES** causes the input data to be regridded at the pixel resolution across the plot.

5.3 XY plot

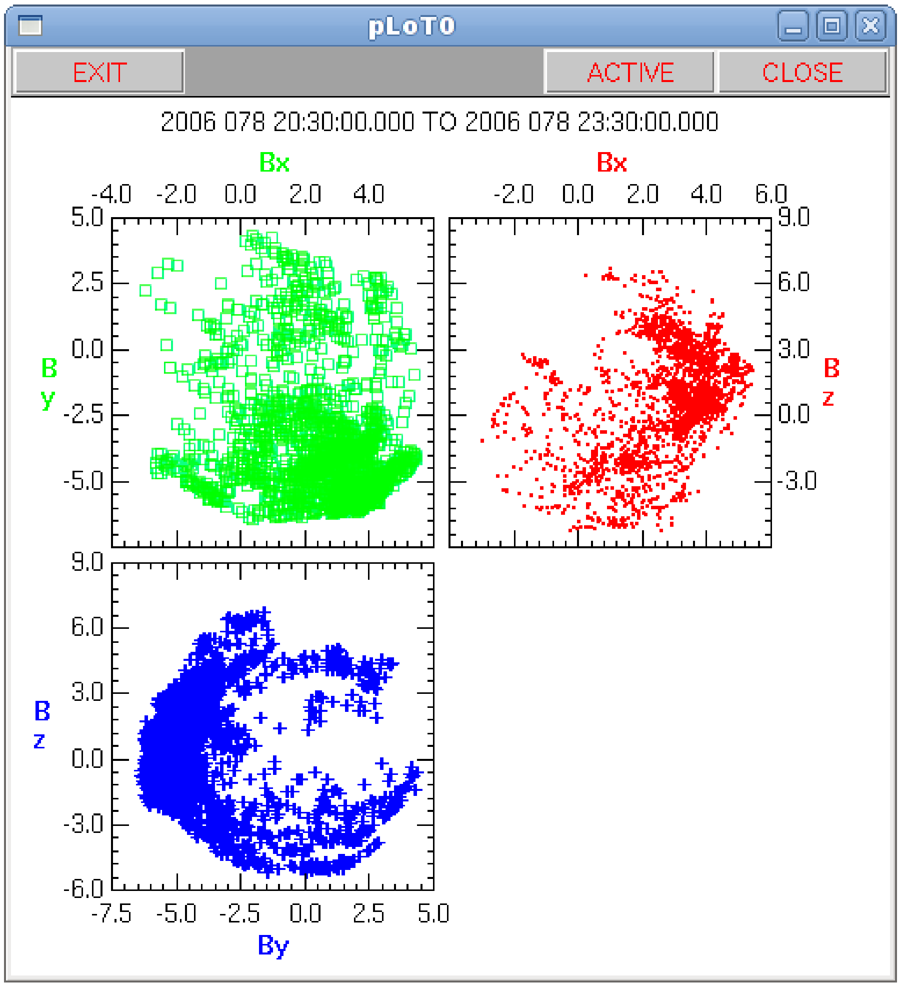


Figure 15: XY plots

Figure 15 shows a set of XY plots. XY plots are non-time based plots (the **Time** option in the plot layout definition for the plot is set to **NO**). They require both an X and Y input variable in their plot definition. This example uses the same data set as in the examples of the time-based plots but plots the data components one against the other.

All three plots use the **POINT** plot format. Both axes are autoscaled in all plots and different colors are used for each plot. A dummy plot was defined to fill the empty lower right cell in the plot grid to keep the displayed plot axes aligned.

5.4 SpectroScalar plot

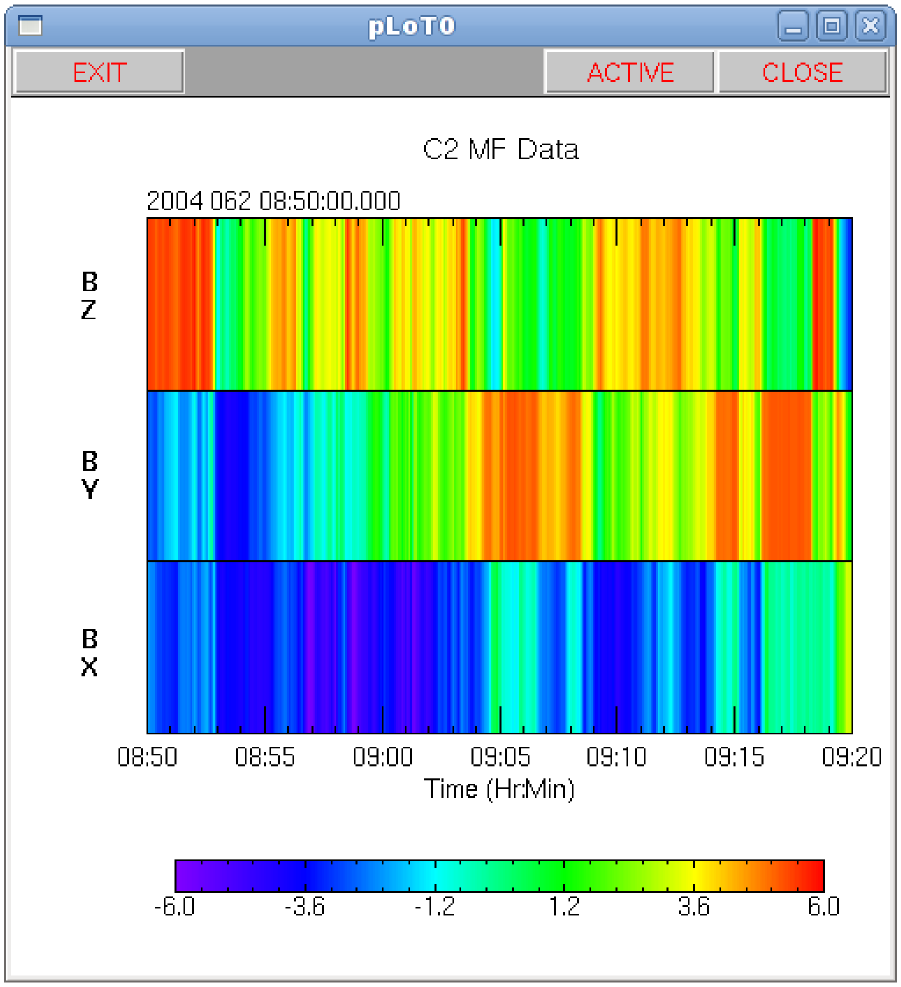


Figure 16: Spectro Scalar plots

Figure 16 shows a set of 3 spectroscalar plots. Spectroscalar plots are plots of scalar quantities in a spectrogram format. Outputting a set of spectroscalar plots under UDFAnalysis takes a bit more effort and thought than outputting other types of plots and for that reason we will describe how the plots were set up in some detail.

A spectroscalar plot is a 3D data representation with each plot definition requiring an **X**, **Y** and **I** variable designation. Here **X** is time, **the Y** variable gives the Y range over which the spectroscalar extends in the plot panel and the **I** variable is the intensity which is translated to color. The **Y** variable is not part of the data read in and so needs to be created. This is done using the **SetV** Function.

Before setting up the Y variable consider how to arrange the plot. The easiest way to get the three spectroscalar plots into a single plot panel is to scale the Y axis of the plot to run from 0 to 3 and then to set the Y range for the three spectroscalar plots to run from 0 to 1, 1 to 2, and 2 to 3 respectively. Setting up a Y range means setting up a pair of Y variables for each plot to hold the starting and stopping positions.

If you look at the **SetV** function definition in the menu you will see three pairs of order 2 variables defined. Each pair represents the start and stop Y value for one of the three spectroscalar plots. These become the Y variables in the plot definitions.

In the plot layout menu there is a single defined layout with its Y axis scaled from 0 to 3. There are only three major tick marks along the Y axis which will divide the three spectroscalar plots. The tick format has been set to **SPAN** so that each tick runs across the X direction. A colorbar is output below the lower X axis, all numerical annotation along the Y axis has been turned off and the left and bottom gaps have been increased. The increase in the left gap is for annotation and the increase in the bottom gap is to add extra room for the colorbar.

Since the input data has been stored in the system-wide time grid there is no X variable for any if the entries in the **PLOT DEFINITIONS** menu. The Y variables are those defined in the **SetV** function. These are of order 2 and represent the start and stop values associated with the measurements. The intensities are the scalar components of the vector magnetic field. Autoscaling of the Y axis is turned off for all plot definitions.

One variation on the above would be to define the Y extent of each of the three magnetic field components to show a gap in the plot, adding separation. In this scenario the ranges for the three variables might be set to run from 0 to .97, 1.03 to 1.97, and 2.03 to 3 respectively.

5.5 Spectrogram plot

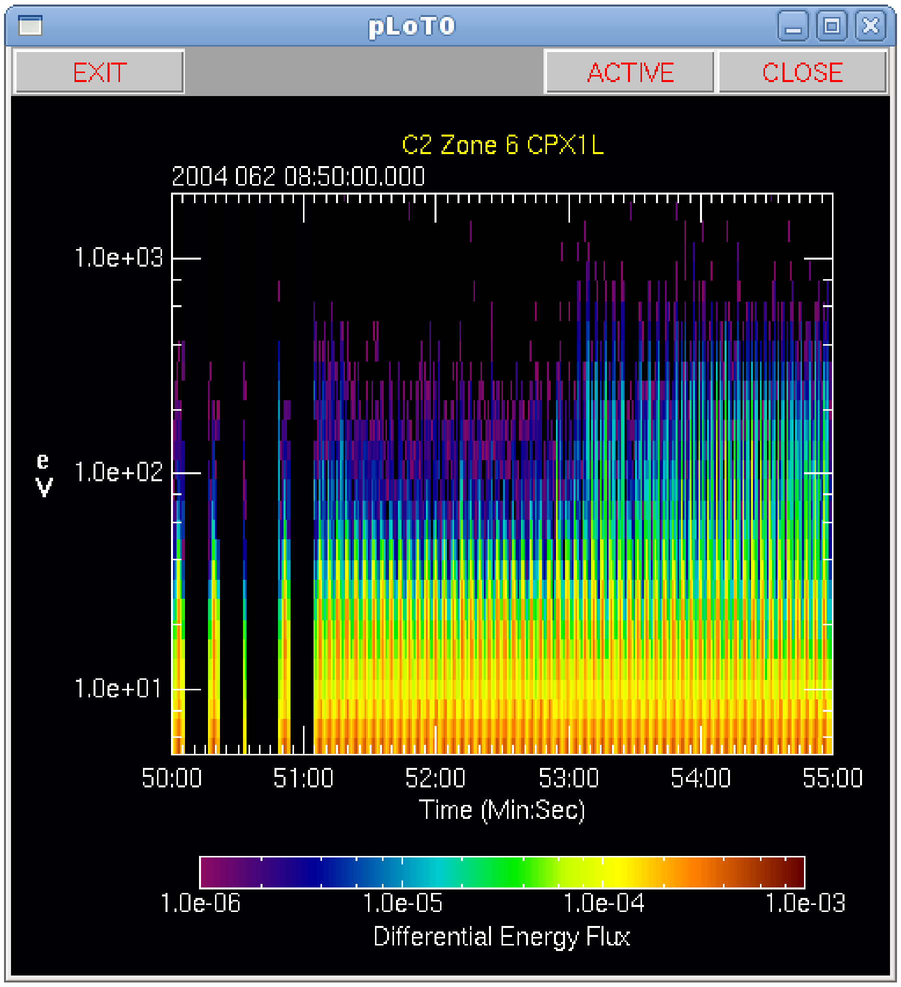


Figure 17: Spectrogram plot

Figure 17 shows a spectrogram of electron data from the Cluster C2 PEACE electron experiment. The basic difference between a spectrogram and a spectroscalar plot is that the data is truly 2D and has an inherent Y scaling associated with it. This is obtained when the data is accessed. You can see this in the set up menu associated with the variable definition.

One thing you will become aware of if you run this example is that it takes an quite a while for the plot to come up. This is a function of how Tk renders its plots and not slowness in acquiring the data.

6 Functions

The following examples show usage and output from a number of the function plugins available within the UDFAnalysis package.

6.1 Bin Function

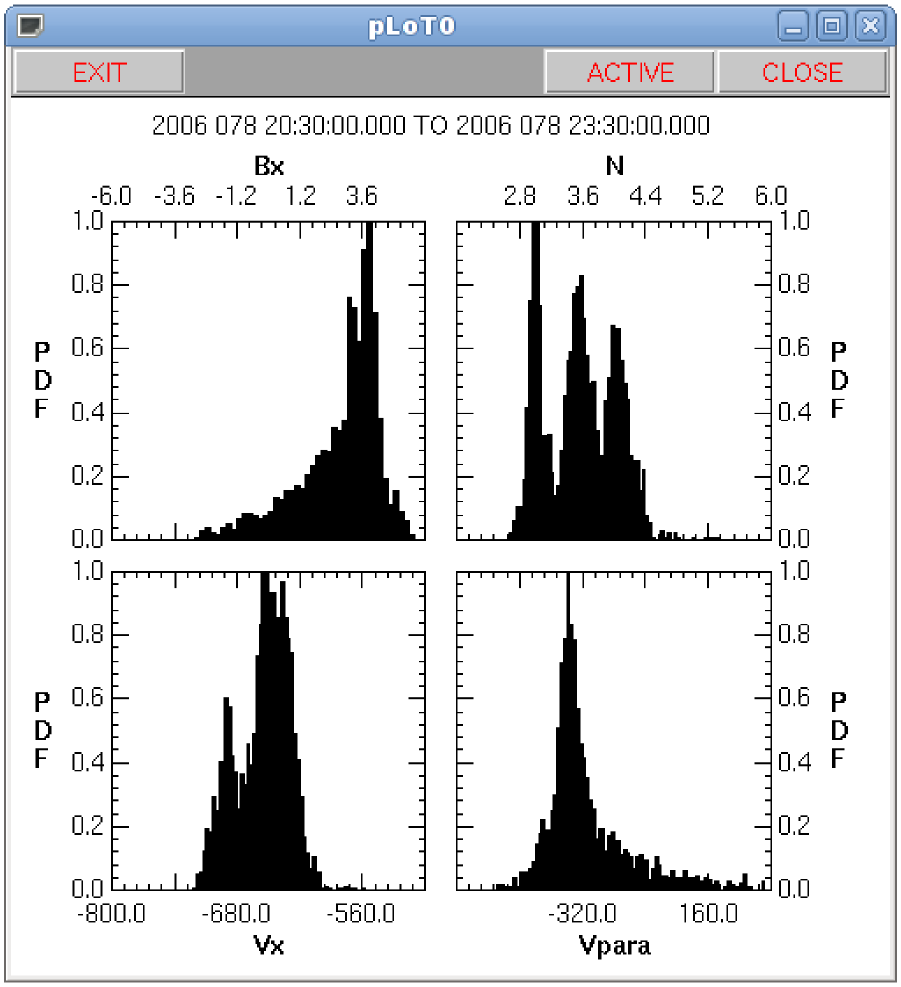


Figure 18: Plots of binned data

Figure 18 shows a set of 4 plots of Probability Distribution Function (PDF). The probability distributions were computed using the **Bin Function** which bins variables within a specified interval. Both the interval and the number of bins within the interval are selectable. This example shows binning of 4 different variables, the x component of the magnetic field, the plasma electron density, the x component of the electron bulk velocity in GSE coordinates and the component of the electron bulk velocity parallel to the magnetic field vector. The time range over which the data was accumulated is shown in the label above the plots.

The Bin Function Menu shows four separate bin definitions. The **Input** variables were read in from individual data sources. The **BinX** variables are the returned bin centers and the **BinY** variables are the number of events in each bin. **Min** and **Max** define the extent of the interval over which the data is being binned and **Bins** is the number of bins to divide the interval into. These vary with the expected range and fineness of the data. All of the defines have **Norm** set to **YES**. This normalizes the PDF to run from 0 to 1 which is useful if you are plotting many PDF's which share a common Y axis.

In the plots definitions autoscaling was turned off for all axes since the ranges covered by each plot are known from the binning definitions.

You should take some time to look at the **VARIABLE** definition menu and submenus under it as they show how the various data sets are being imported and how the data variables themselves are defined. Note that because the some of the moment data is to be returned in magnetic coordinates, the magnetic field data but be input before the moment data.

One think you will note if you run this menu is that it takes a substantial amount of time to read in moment data. This is caused by the large number of coordinate transformations which are being executed as the data rotated from the spacecraft frame into a GSE based coordinated and a subset of that data is further rotated in a magnetic field based coordinate system. Addition of some C backing code in future releases my mitigate this somewhat.

6.2 Filter Function

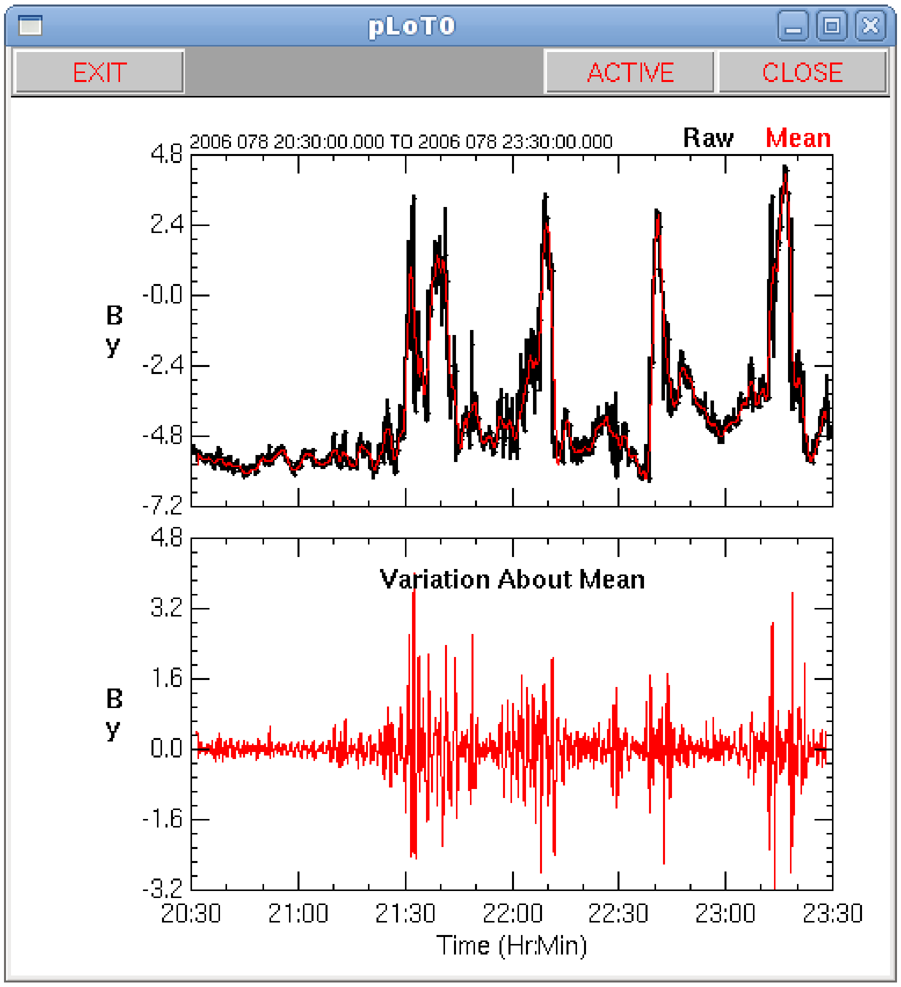


Figure 19: Plots of filtered data

Figure 19 shows a set of 2 plots illustrating results of the **Filter Function**. The filter function returns data within, above and below a specified frequency band. The band is often set to have a zero width in which case only data above and below the selected frequency is returned. In this mode the function can be used to compute the moving mean of a variable and the variation about the mean which is what is done in this example.

The **Filter Setup** menu under the **Function** menu shows the option settings used in generating the filters. The input being filtered is the y component of the magnetic field. Because the upper and lower band edges are set to the same value the function will return can only the filtered data below and above 0.005 Hz. Return is set to **LH** meaning that both the data in the **L**ow and **H**igh frequency ranges are returned. Since two variables are being returned the output variable (fBy) is a variable of order 2. The low frequency component is returned in fBy0 and the high frequency component in fBy1. The smoothing order is left at the default value of 2.

The upper plot in the Figure 19 contains the unfiltered input variable **By** (black) and the filtered low frequency component of the data Red. The lower plot contains the high frequency component of the data.

6.3 Vector Function

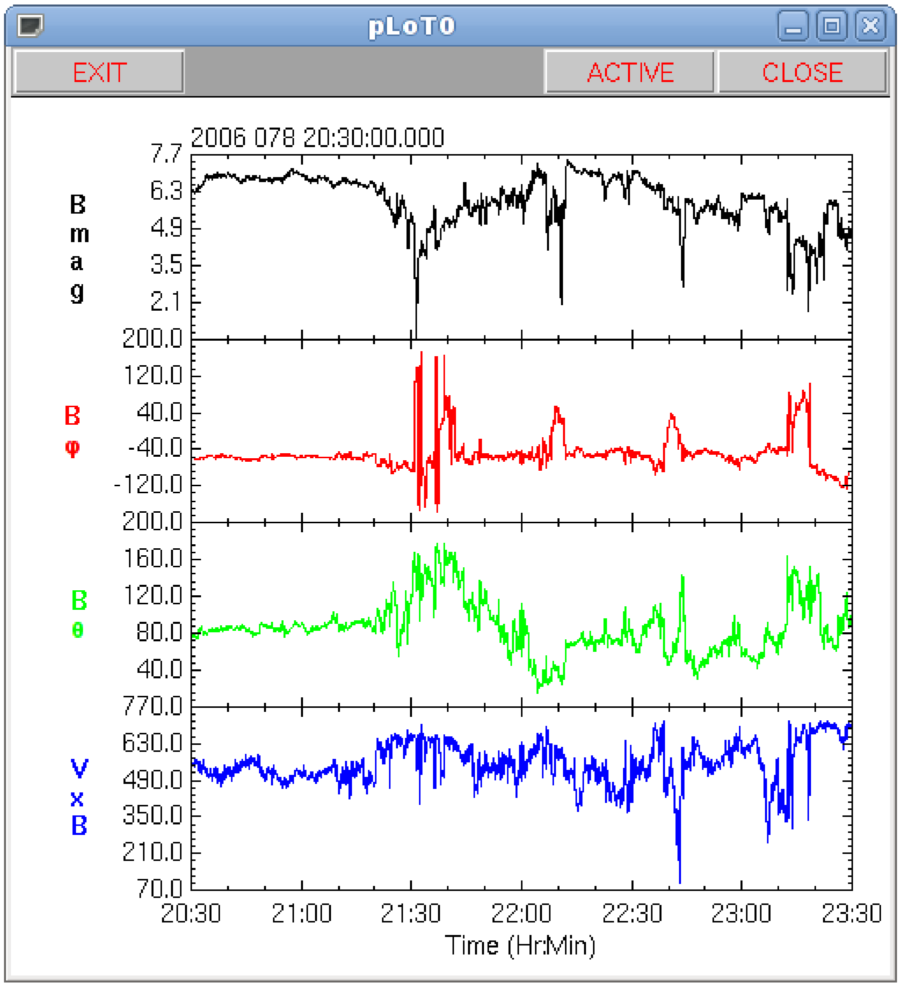


Figure 20: Plots of the results of vector operations

Figure 20 shows a set of 4 plots of output generated by the **Vector Function**. The vector function provides a number of operations specific to vectors or order 3 variables. This example uses both magnetic field data and plasma moment data to illustrate the usage of four vector operators. The four operations shown in the **Vector Setup** menu which is a submenu under the **FUNCTIONS** menu. In the first defined operation the magnetic field vector which is input in Cartesian coordinates is changed to spherical coordinates and output in the vector variable **BP**. The components of this vector are plotted in the upper three panels in the example output. Remember that the polar angle is returned in the range $0° \rightarrow 180°$. In the next operation the input magnetic field is converted to a unit vector which is stored in **uB**. This is not output but is used in the next operation which takes the cross product of the plasma bulk velocity with the unit magnetic field vector. The cross product is output in the variable **VxB**. The last operation computes the magnitude of the cross product which is stored in **VxBM**. This the plasma velocity perpendicular to the magnetic field.and is output in the lower panel in Figure 20. Of the operators used only the cross product operator requires the input of two vectors. The other operators work on a single vector.

6.4 Fit, Set Value and Variable Map Functions

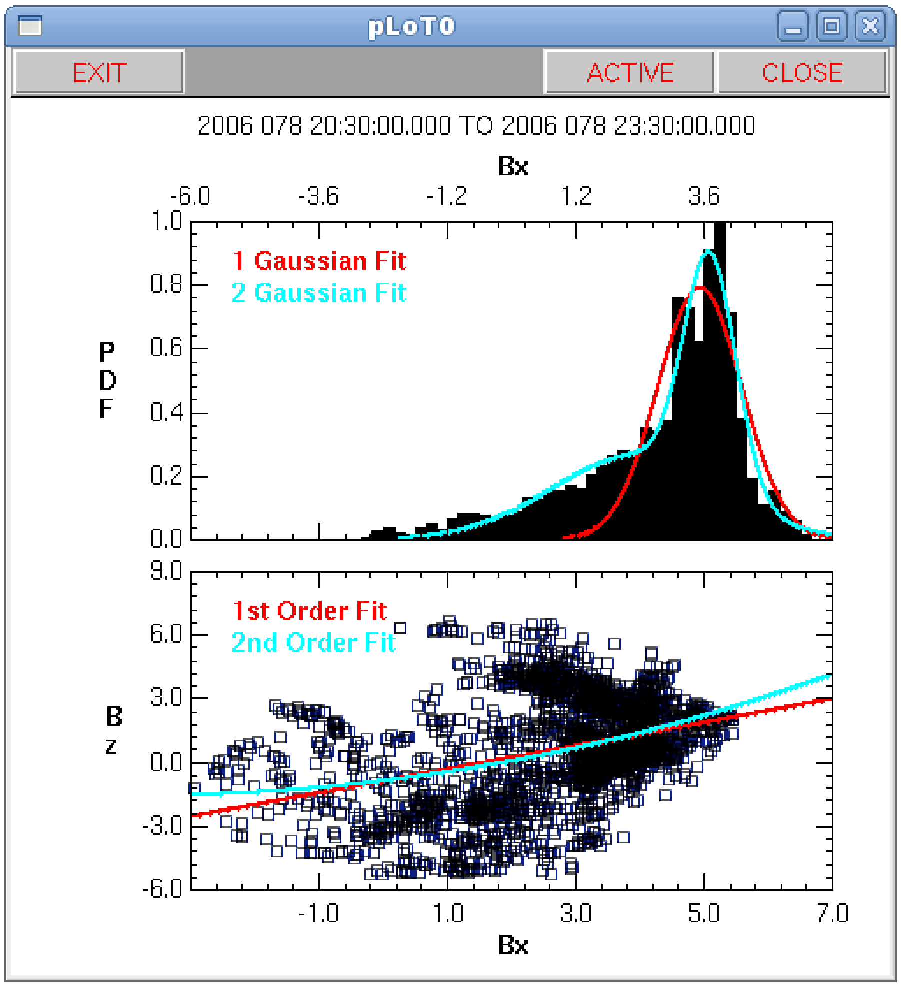


Figure 21: Plots of the results of fit function

As its name implies, the fit function fits a set of data to a user specified model function. Use of the fit function often requires previous calls to the **VMap** function and sometimes to the **SetV** function both of which will be introduced in this example. Figure 21 shows a set of 2 plots which include fits to data generated by the **Fit Function**.

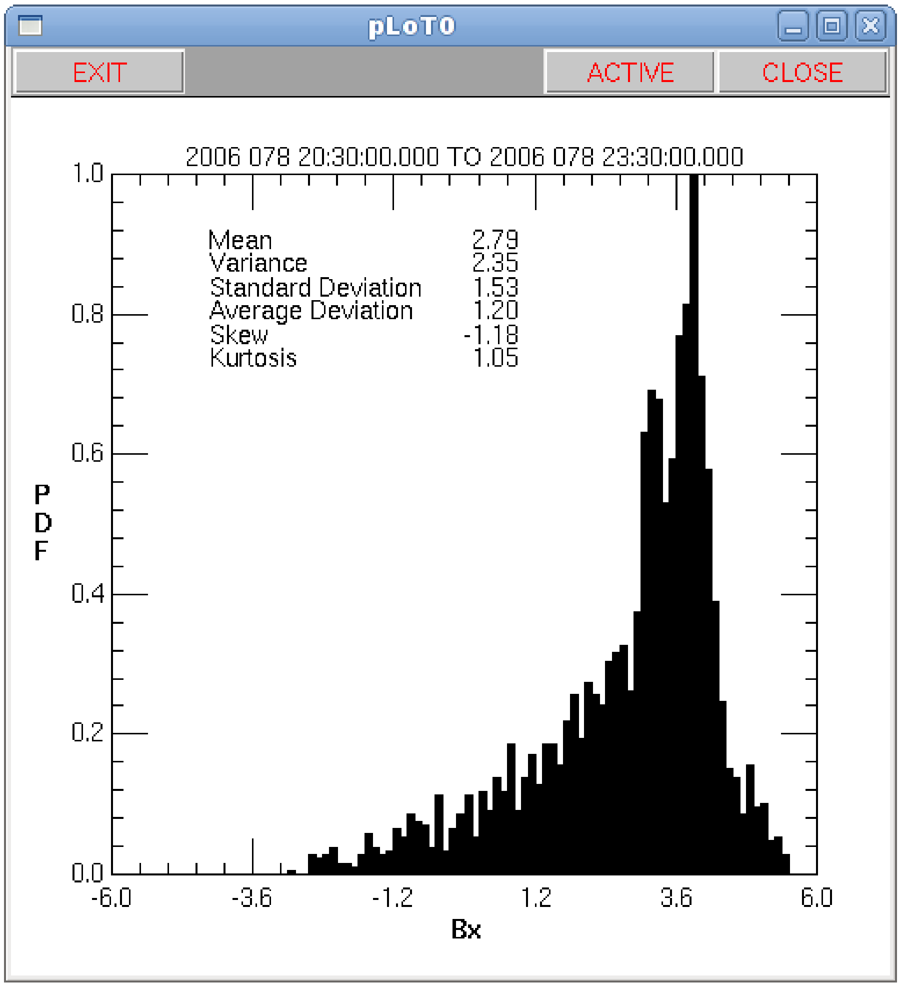
There are four separate fits defined in the fit menu, all of which have a dimension of 1, that is the fits are of the form y = f(x). The first two fits in the menu text window a linear fit algorithm to fit the same input data set to two different polynomial functions. From the number of coefficients, the first definition fits the input data to straight line and the second to a function of the form $A_0 + A_1x + A_2x^2$. The data being fit is a set of Bx and By values. The variables holding the data have been mapped into the second order variable **bFit** using the **VMAP Function**.

The next two fit definitions use a non-linear fitting algorithm and also fit a common set of data. The first non-linear fit is to a single Gaussian function and the second to a pair of Gaussians. The input data is the Probability Distribution Function of the x component of a set of magnetic field data which was shown in the Binning example in section 6.1.

Non-linear fits require an initial guess of the fit coefficients, three in the case of a single Gaussian fit and double that for a two Gaussian fit. The guesses are held in the variables **gF** and **gF2**. Both of these variables are created using the **SETV Function** setup menu. Since you can only assign a maximum of 4 values per line it takes two lines to completely specify the 6 values in **gF2**.

The **Plot Definition Menu** shows how to plot the fits. The fit output variable specified in the **Fit Setup** Menu is entered as the X Variable in the plot with no other variables specified. All of the information needed to reconstruct the fit is contained in output variable generated by the fit function.

6.5 Statistics Function



```
Mean                  2.79
Variance              2.35
Standard Deviation    1.53
Average Deviation     1.20
Skew                 -1.18
Kurtosis              1.05
```

Figure 22: Plot showing statistics data

Figure 22 shows output generated by the **Statistics Function**. The statistics function computes and returns a number of statistical parameters for an scalar variable. These are not plottable quantities but as seen in the figure can be added to the plots as annotation. The input data in this case is a set of Bx magnetic field values and what is plotted is the Probability Distribution Function of the values as computed using the **BIN** function.

The **Plot Label** illustrates how to output the returned statistics information. The statistic definitions and values were output as separate columns of labels which allowed the alignment seen in the figure.

6.6 Define Grid and Grid Fill Functions

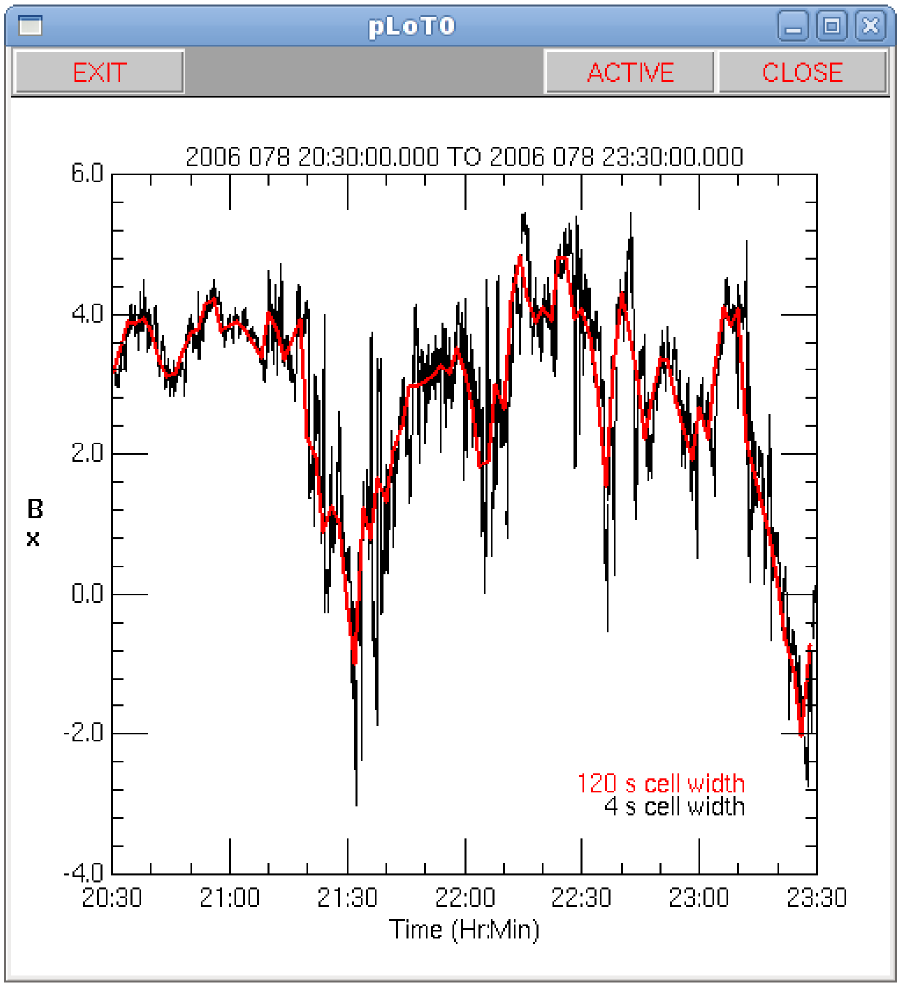


Figure 23: Plot showing data gridded at different time resolutions

The **Define Grid** function provides a mechanism to define a grid information array which can then be applied to arbitrary data sets to put them in a regular grid. The gridding of the data is done through the **Grid Fill** Function. The output of the **Define Grid** function is not plottable but that of the **Grid Fill** is. Figure 23 shows output generated by the **Grid Fill Function**.

The input data to the example is a set of ungridded Bx values. Both the data and the start and stop times are read in and saved. The latter is needed in the **Grid Fill** function. Then two nearly identical time based grids are defined. Each grid definition occupies two lines in the **DefGrid** setup menu text window. Leaving both the **X Min** and **X Max** options blank causes the X extent of the grid to match the X extent of the system-wide time grid. The first grid definition is stored in gIA and sets the grid cell width to be 4 seconds. The second grid definition is stored in gIB and sets the grid cell width to be 120 seconds.

Application of the grid definitions is made in the **Grid Fill** setup menu. The input data is placed into two grids: **BxA** which was defined according to the options set in **gIA** and **BxB** which was defined according to the options set in **gIB**. Both of these data sets are then plotted in the example output plot, the lower resolution grid in red and the higher resolution grid in black.

6.7 Dynamic-PS Function

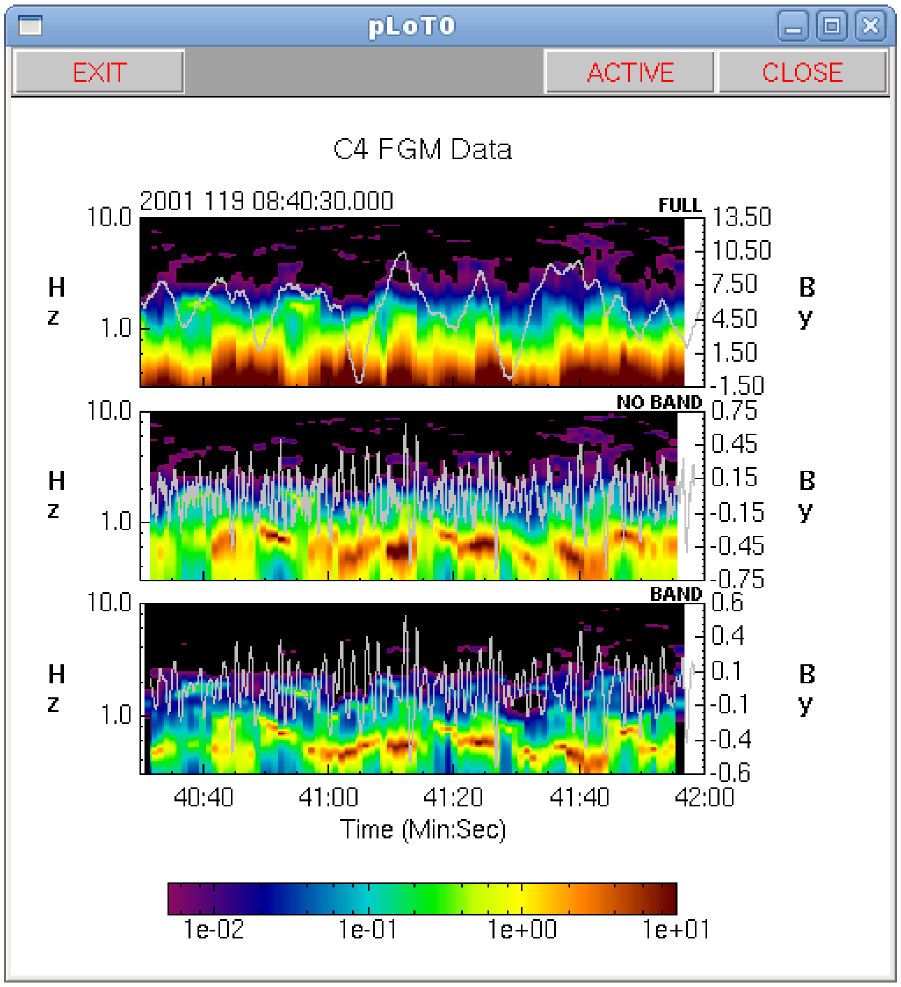


Figure 24: Plot showing dynamic power spectra

The dynamic-ps function creates a dynamic power spectra from any scalar variable stored in a time-based grid. The input is often data which has been passed through the **Filter Function** either to remove the low frequency component in the data or to restrict the frequency range in the data prior to creating the power spectra.

Figure 24 shows output generated by the **Dynamic-PS Function**. The input data is a set of high resolution FGM magnetic field data obtained from the Cluster Active Archive. All of the dynamic spectra involve the Y component of the field only. There are three sets of Dynamic Power Spectra shown in the figure: one derived from the raw input and two from frequency filtered components of the data. The **Time Menu** has the cell width of the time grid set 0.015s which is about the time resolution of the high resolution magnetic field data at this time.

The data was filtered using the **Filter Function**. The first definition in the setup menu filters the data into two components, one below 0.3 Hz and one above 0.3 Hz. Only the upper component is returned as a named variable. The second definition filters the data into three components, one below 0.3 Hz. one between 0.3 Hz and 1.0 Hz, and one above 0.3 Hz. Only the band and upper components are returned as named variables.

The dynamic power spectra setup menu contains four definitions of the dynamic power spectra of which the middle two definitions are coupled, that is their results are returned in the same variable combination. The returned variables are of order 4. The first two variable components hold the start and stop time of the window in which the individual power spectra are computed, the third component contains the frequencies at which the power is returned, and the last component contains the power.

The dynamic power spectra are spectrogram type plots and require an **X** (in this case the time), a **Y** (the frequency steps) and an **I** (the power) input. This is seen in the **PLOT DEFINITION** menu. The upper plot in the output is the dynamic power spectra derived from the raw data, the next from the filtered data > 0.3 Hz, and the last from the combined 0.3 to 1.0 Hz and > 1.0 Hz components. The first two plots have the input to the dynamic power spectra function overplotted while the last plot has just the signal within the band overplotted. **Note:** When plotting a line plot over a spectrogram type plot the line plot must be specified after the spectrogram plot in the plot definition list.

The gaps at the end of the dynamic spectra plot are due to the finite window width in which the spectra are computed. Gaps beginning and end of the line plots of the filtered data are due to the finite with of the filters used in the processing of the data.

6.8 Collapse Function

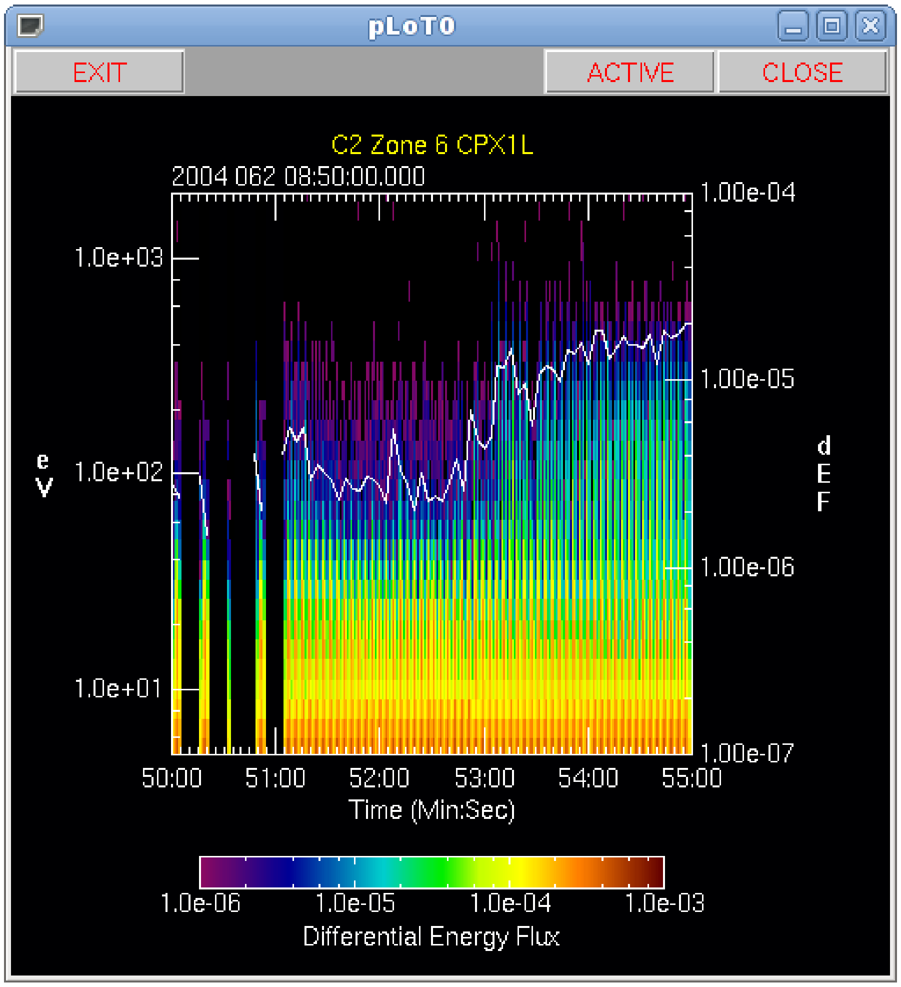


Figure 25: Collapse Function Example

The Collapse Function allows a 2D grid to be collapsed down to a 1D grid. The collapse direction and range in the input grid is selectable. Figure 25 shows output generated by the **Collapse Function** overlayed on spectrogram plot set up in the example in section 5.5.

Collapsing over the spectrogram requires two subsequent function calls. Because the input data is not gridded, a grid must be defined and the data gridded prior to the collapse. The gridding of the data in the spectrogram plot is done in the plotting routine and is not available as a variable that can be used. The grid is time based, runs over the plot time, and has 4 second resolution. There are more grids in Y than energy channels so that when the data is gridded it needs to be filled in the Y direction to remove any gaps. The collapse is done in the Y direction in the range 50 to 300 eV. The collapsed grid is output in the variable **NoSw** which can be plotted as 1D time based variable.

Appendix

Section(s)	Menu File Links	Data Links
2.1	Ex4x1Layout.menu	
2.2	Ex2x2Layout.menu	
2.3	ExSpanLayout.menu	
2.4	ExECell1Layout.menu	
2.5	ExDumLayout.menu	
2.6	ExNGapLayout.menu	
2.7	ExAxis1Layout.menu	
2.8	ExAxis2Layout.menu	
2.9	ExCBGLayout.menu	
3	ExPSet1.menu	
4.1	ExLabelA.menu	
4.2	ExLabelB.menu	
5.1	ExTimePlot.menu	
5.2	ExTimePlotA.menu	
5.3	ExXYPlot.menu	
5.4	ExSpecScaPlot.menu	
5.5	ExSpecPlot.menu	
6.1	ExBinFunc.menu	C12006078Full.mom
6.2	ExFilterFunc.menu	
6.3	ExVecFunc.menu	C12006078Full.mom
6.4	ExFitFunc.menu	
6.5	ExStatFunc.menu	
6.6	ExDefGridFunc.menu	
6.7	ExDypsFunc.menu	C4_FGM.CAA
6.8	ExColFunc.menu	