

# Contents

<b>1</b>	<b>Overview</b>	<b>4</b>
<b>2</b>	<b>Internal Framework</b>	<b>4</b>
2.1	Data . . . . .	4
2.2	Functions . . . . .	6
2.3	Internal Definitions . . . . .	6
2.4	Display . . . . .	7
<b>3</b>	<b>General Menus</b>	<b>7</b>
3.1	Color Picker Menu . . . . .	7
3.2	Symbol Picker Menu . . . . .	7
<b>4</b>	<b>Main Menu</b>	<b>8</b>
4.1	Run . . . . .	8
4.2	Quit . . . . .	8
4.3	Input Menu . . . . .	8
4.4	Save Menu . . . . .	9
4.5	Comments . . . . .	9
4.6	TIME DEFINITION . . . . .	9
4.7	VARIABLE DEFINITION . . . . .	9
4.8	FUNCTIONS . . . . .	9
4.9	SETUP PLOTS . . . . .	9
4.10	PLOT LAYOUT . . . . .	10
4.11	PLOT DEFINITION . . . . .	10
4.12	PLOT LABELS . . . . .	10
<b>5</b>	<b>Time Definition Menu</b>	<b>10</b>
5.1	CLOSE . . . . .	10
5.2	Start . . . . .	11
5.3	Echo . . . . .	11
5.4	Stop . . . . .	11
5.5	Cell Time . . . . .	11
<b>6</b>	<b>Variable Definitions Menu</b>	<b>11</b>
6.1	Parent Menu . . . . .	11
6.2	ASCII Source Menu . . . . .	13
6.3	UDF/IDFS Source Menu . . . . .	20
<b>7</b>	<b>Functions Menu</b>	<b>26</b>
7.1	Close . . . . .	27
7.2	Function . . . . .	27
7.3	Add . . . . .	27
7.4	Edit . . . . .	27
7.5	Update . . . . .	27

7.6	Unset	27
7.7	Delete	28
<b>8</b>	<b>Function Plugins</b>	<b>28</b>
8.1	Bin Function	30
8.2	Collapse	32
8.3	Conversion	34
8.4	Cross Cal Function	36
8.5	Cross Helicity Function	38
8.6	DefGrid Function	40
8.7	Dynamic-PS Function	43
8.8	Equation	46
8.9	Filter Function	49
8.10	Fit Function	51
8.11	GridFill Function	58
8.12	Index Function	60
8.13	Mask Function	61
8.14	Math Function	64
8.15	MEM Function	67
8.16	Print Function	69
8.17	Random Function	71
8.18	Spatial-Derv Function	74
8.19	Setv Function	77
8.20	Statistics Function	80
8.21	Unset Function	82
8.22	Vector Function	83
8.23	Vmap Function	85
<b>9</b>	<b>Plot Setup Menu</b>	<b>86</b>
9.1	Device	87
9.2	Window	87
9.3	Label	88
9.4	Color	88
9.5	Grid	88
9.6	Offsets	89
9.7	3D Plots	89
<b>10</b>	<b>Plot Layout Menu</b>	<b>90</b>
10.1	The Text Window	93
10.2	Plot Layout Options	93
<b>11</b>	<b>Plot Definition Menu</b>	<b>97</b>
11.1	Plot Definition Options	99
11.2	Adding or Modifying Plot Definitions	102

<b>12 Plot Labels Menu</b>	<b>102</b>
12.1 Label Definition Options . . . . .	105
12.2 Adding or Modifying Label Definitions . . . . .	107

# UDFAnalysis

November 12, 2009

## 1 Overview

The UDF Analysis package provides an environment into which data can be imported from both ASCII and UDF/IDFS sources and then algebraically and functionally manipulated with the results available for display or output to an ASCII file. The package is written entirely in Tcl/Tk and is known to run under Unix, Linux, OSX, and Windows.

This manual describes how to use the package. It includes details of some of the underlying framework, how to access and manipulate data, how to set up a matrix of plots, how to output data, and how to add a new plugin. There is an additional document which contains working examples.

## 2 Internal Framework

While it is not absolutely necessary to understand the underpinning of how the package works, it is sometimes helpful, especially if you plan to design a plugin to provide additional data manipulation capability. It also helps to provide insight into the meanings of some of the terms that will be used later to describe different classifications of data.

### 2.1 Data

UDFAnalysis works with both time and non-time based data sets. When a set of data is imported into the UDFAnalysis framework it is mapped into a data grid. Each data grid is linked to a variable name which identifies the measurement in all subsequent usage.

#### 2.1.1 Non-time based data

Non-time based data sets are mapped into a data grid which for all intents and purposes is a linear array of measurements. The data are mapped into successive grid cells starting at cell 0 and ending at cell N-1 where N is the number of data points read in.

### 2.1.2 Time based data

Time-based data sets are mapped into a time based grid. The framework provides a general purpose time grid definition which is used to defined the storage array characteristics although this can be overridden. The purpose of the time grid is to allow data with dissimilar time resolutions to be placed in a common format which can be easily manipulated and combined.

The internal time grid begins at the user selected time  $T_{ub}$  and contains  $N$  cells of width  $T_c$ . The cell width is user specified. The number of cells is determined by the user selected end time  $T_{ue}$  and is given by:

$$N = \text{int}((T_{ue} - T_{ub})/T_c) + 1$$

if  $\text{fmod}((T_{ue} - T_{ub}), T_c)$  is non-zero and

$$N = \text{int}((T_{ue} - T_{ub})/T_c)$$

if it is. The ending time of the last cell in the time grid,  $T_{ce}$  is then:

$$T_{ce} = T_{ub} + N * T_c$$

which should contain the user end time to within the grid cell resolution.

### 2.1.3 Data Mapping

There are two selectable methods by which time based data can be mapped into a time grid which are termed **BAND** and **POINT** storage. The mapping may expand or compress depending on relationships between the data and grid cell temporal resolutions.

The **BAND** storage method requires that each data point have a start and stop time associated with it and spreads data over all cells in the grid covered by its duration. Each cell has added to it the time weighted average of the data. (The weighting factor is 1 if the measurement spans the entire cell.) Once all of the data have been stored in the grid, each grid cell is normalized by the accumulated weighting factors of the data stored in it. The **BAND** storage method preserves natural data gaps in the data. If you select the **BAND** method for data which has only a single time associated with it, it will be changed to **POINT** at the time of storage.

The **POINT** storage method can be used both with data sets which are have only a single associated time per value or with data sets in which each value has an associated start and stop time. The data are stored in the grid cell which contains the measurement time (using the start time when both the start and stop time are specified for a value. After all of the data have been stored each grid cell is normalized by the number of measurements which were stored in it. Data stored using the **POINT** method which has a lower time resolution than the storage grid will result in unfilled cells.

The option exists for both storage methods to fill unfilled grid cells by applying a linear interpolation across unfilled cells. This option should always be used when storing data using the **POINT** method.

### 2.1.4 Linked variables

All data variables within the UDFAnalysis framework are considered scalars. You can, however, link the scalar variables together to form arrays of variables. There are several methods ways to specify sets of linked variables. The most general way is by using the variable prefix **N.M**, which expands a variable name to N scalars with the values M, M+1, ... M+N-1 appended to them. The variable **4.5,T** is identical to the set of variables T5, T6, T7, and T8. This assumes that the variables T5, T6, T7, and T8 already exist as defined scalar variables.

In addition to the above general prefix there are two additional prefixes that can be used to link variables. The most useful is the **vec** (vector) prefix which expands a variable name into three scalars with x, y, and z appended to them. The variable **vec,V** is identical to the set of variables Vx, Vy, and Vz. The **ten** (tensor) prefix will expand the variable into nine scalars with xx, xy, xz, yx, yy, yz, zx, zy, and zz appended to them.

Setting up linked sets of variables is facilitated by using the **VMAP** function which allows variable names to be mapped alternate names. If you have the variables A, B, and C which need to be linked as a vector you can use **VMAP** to link A to Vx, B to Vy, and C to Vz, or alternatively A to V0, B to V1, and C to V2.

The number of linked variables associated with a variable name is its order. A scalar variable has a length or order of 1, vectors variables are of order 3, tensors variables are of order 9 and the variable **12.0,T** has an order of 12 while **4.5,T** has an order of 4. The individual variables which comprise a linked variable are known as the variable components. These can always be accessed individually and used anywhere that scalars would be.

## 2.2 Functions

Manipulation of data within the UDFAnalysis framework is performed by a set of plugins known as **FUNCTIONS**. Functions are designed to perform specific operations. The use of a plugin architecture for functions to makes it relatively easy to add new ones. Plugins add to the versatility of the package.

Functions operate on variables either cell by cell or on the grid as a whole. The former are generally arithmetic based functions (eg., adding two variables together) while the latter tend to be analysis based (eg., computing power spectra or fitting data). Results of functions are returned as variables which are available to be used in subsequent function calls.

## 2.3 Internal Definitions

The following table lists several internal variable definitions.

Variable	Value	Definition
apANS(BaD)	-1.0e51	Bad Data
apANS(BaDU)	1.0e50	Out of Limits High
apANS(BADL)	-1.0e50	Out of Limits Low

## 2.4 Display

The UDFAnalysis package provides a highly customizable set of display options. Multiple plots can be displayed within a single window. Labels can be placed anywhere within a plot and can include the values assigned to variables computed in function calls.

## 3 General Menus

There are several menus which are not directly accessible from within any UDFAnalysis menu. These are invoked internally to supply common options to options in other menus. They are described below outside the descriptions of the nominal UDFAnalysis menus.

### 3.1 Color Picker Menu

The Color Picker menu is used by menu options which involve setting colors such as text or line plot colors. The menu as it comes up is shown in Figure 1.

Colors can be selected in two ways, either by clicking on one of the predefined colors or by directly setting the Red, Green, and Blue intensities using the sliders as the bottom of the menu. The box below the CANCEL button shows the currently selected color while the box to its left shows the corresponding hex value. The hex definition is in the format **RRGGBB**. Selecting a predefined color will set the sliders to the corresponding RGB values allowing easy deltas off the color to be made. When setting a color using the sliders the color is continuously updated in the upper left-hand box.

Once the color has been defined click **SELECT** to apply it to the menu option which generated the menu. **CANCEL** will close the menu without applying the selected color.

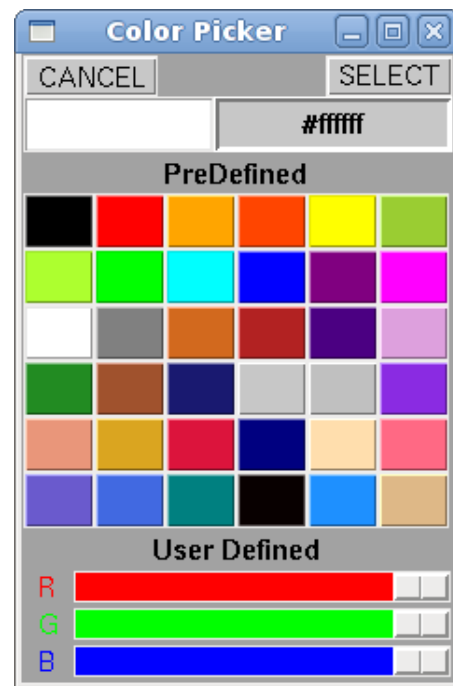


Figure 1: Color Picker Menu

### 3.2 Symbol Picker Menu

The symbol picker menu is invoked by right clicking the mouse in any menu option which involves defining either text labels or plot symbols. The menu is shown in Figure 2.

To select a symbol or character, click on the appropriate symbol. The selected symbol and its unicode value will appear in the upper entry box. Click **Select** to select the symbol and transfer it to the option from which the menu was invoked or click **Cancel** to cancel the selection process.

## 4 Main Menu

The UDF analysis program is invoked as:

*UDFAnalysis* [MenuFile]

The **MenuFile** command line argument is optional. If present is the saved set of options from a previous run of UDFAnalysis. Starting UDFAnalysis with a MenuFile preloads the UDFAnalysis menus with the settings stored in the file.

On startup the program brings up the GUI shown in Figure 3. If invoked with a **MenuFile** the menu file name will be displayed in both the **Input** and **Save Menu** entry boxes, otherwise these fields are blank as above. This menu is the springboard to the full set of menus and options available within the UDFAnalysis framework. The options in this menu are described below.

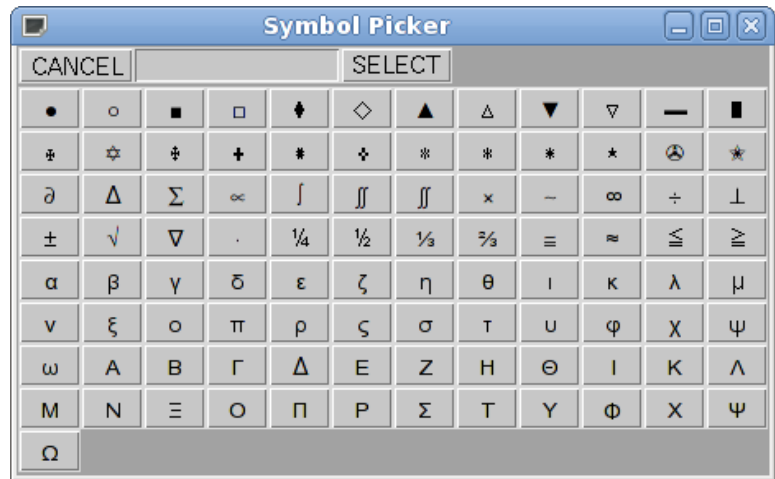


Figure 2: Symbol Picker Menu

### 4.1 Run

The **RUN** button begins execution of the program. Full execution should only be started once all applicable menus and options have been filled in. Execution includes reading in data, running all selected functions, and plotting and dumping selected results. **RUN** *does not* perform a menu save operation.

### 4.2 Quit

Quits and exits the program. This *does not* save any changes to the current menu made after the last menu save operation.

### 4.3 Input Menu

The name of the file which contains the menu options saved from a previous run

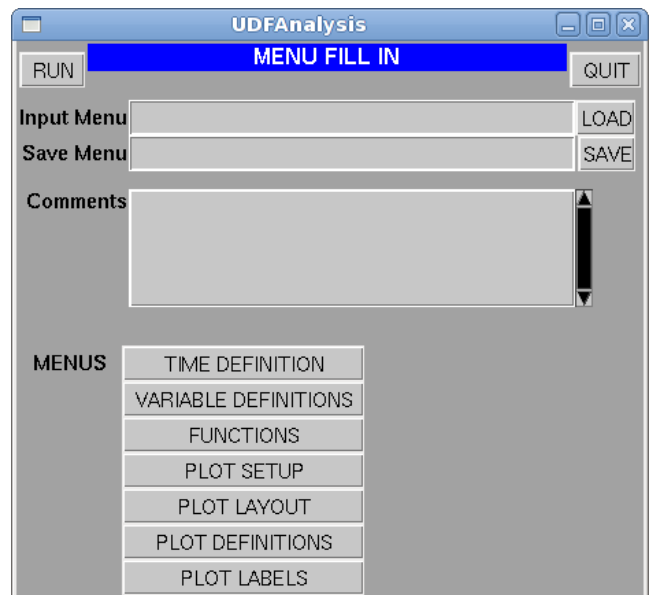


Figure 3: Main Menu



of the **UDFAnalysis** program. Click the **Load** button to the right to load the options into the current menu. **NOTE:** If the saved file contained unfilled options those may still need to be defined before running the application.

Hitting a return in the menu file entry box after you have typed in the file name will echo it down into the **Save Menu** box. If you are not loading options from a pre-existing save file there is no need to enter anything here.

#### 4.4 Save Menu

The name of the file which the current state of the menu options will be saved. Click the **Save** button to the right to save the current state of the menus to the file. The **Save** operation works on incomplete menu definitions. You can save at anytime during the process of filling in the various menus. **NOTE:** Saving to an existing file overwrites whatever was currently in the file without warning.

If the input and save file names are the same, saving will update the old options. If you wish to make changes to the menu but not loose the original menu settings the save file name must be different from the input file name.

#### 4.5 Comments

A free form text field which can be used to describe the purpose and contents of the menu definition.

#### 4.6 TIME DEFINITION

Invokes the GUI menu containing the timing options. These basically defines the timing grid by setting its beginning and ending time as well as its cell resolution.

#### 4.7 VARIABLE DEFINITION

Invokes the GUI menu used define the importing of external variables. External variables can have a UDF/IDFS or an ASCII source. The link between data and the variable names used to define the data within the UDFAnalysis framework is set up here.

#### 4.8 FUNCTIONS

Invokes the GUI menu containing the set of plugins available to manipulate the data.

#### 4.9 SETUP PLOTS

Invokes the GUI menu which defines all of the general plot characteristics. This includes the plot window size, the background color, grid overlay definitions and color table.

## 4.10 PLOT LAYOUT

Invokes the GUI menu which defines all of the plot specific characteristics. Most of this deals with the placement of the plot within the overall plot window, axis scaling options, and annotation options. This menu does not populate the plots with data. That occurs in the **PLOT DEFINITION** menu.

## 4.11 PLOT DEFINITION

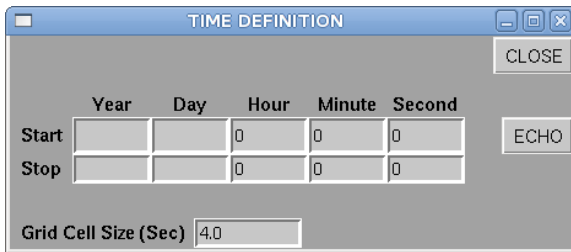
Invokes the GUI menu which links variables with plots. (Plots are defined in the **Plots Layout** menu).

## 4.12 PLOT LABELS

Invokes the GUI menu links labels to plots.

# 5 Time Definition Menu

The time menu is invoked by clicking the **TIME DEFINITION** button in the **Main Menu**. An unpopulated and populated version of the menu is shown in Figures 4 and 5 respectively.

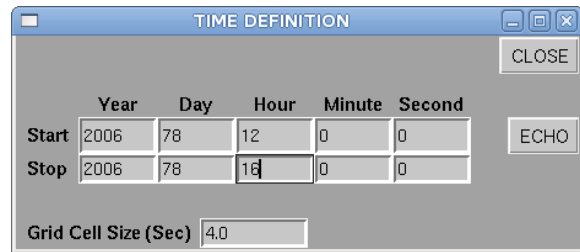


The screenshot shows a window titled "TIME DEFINITION" with a "CLOSE" button in the top right corner. The window contains a table with columns for "Year", "Day", "Hour", "Minute", and "Second". There are two rows: "Start" and "Stop". The "Start" row has empty fields for Year and Day, and "0" for Hour, Minute, and Second. The "Stop" row has empty fields for Year and Day, and "0" for Hour, Minute, and Second. To the right of the table is an "ECHO" button. Below the table is a "Grid Cell Size (Sec)" field with the value "4.0".

	Year	Day	Hour	Minute	Second
Start			0	0	0
Stop			0	0	0

Grid Cell Size (Sec) 4.0

Figure 4: Initial Time Definition Menu



The screenshot shows a window titled "TIME DEFINITION" with a "CLOSE" button in the top right corner. The window contains a table with columns for "Year", "Day", "Hour", "Minute", and "Second". There are two rows: "Start" and "Stop". The "Start" row has "2006" for Year, "78" for Day, and "12" for Hour, with "0" for Minute and Second. The "Stop" row has "2006" for Year, "78" for Day, and "16" for Hour, with "0" for Minute and Second. To the right of the table is an "ECHO" button. Below the table is a "Grid Cell Size (Sec)" field with the value "4.0".

	Year	Day	Hour	Minute	Second
Start	2006	78	12	0	0
Stop	2006	78	16	0	0

Grid Cell Size (Sec) 4.0

Figure 5: Filled Time Definition Menu

The menu establishes the parameters used to define the internal time grid as well as the start and stop times used when putting up time based plots. The time grid definition allows time based data to be stored at a common time resolution which facilitates mathematical operations between data sets. The options available in this menu are:

### 5.1 CLOSE

Close the menu window. The window can also be closed (and reopened) by clicking the **TIME DEFINITION** button in the **Main Menu**.

## 5.2 Start

A group of 5 options which sets the time of the leading edge of the first cell in the time grid. It is also used to set the position in UDF/IDFS data sets where data acquisition begins. In order, the inputs are: the full year, day of year, hour, minute and second.

## 5.3 Echo

Clicking the **Echo** button to copy the **Start** time fields down into the **Stop** time fields. Handy to cut down on the number of stop fields which may need to be filled.

## 5.4 Stop

A group of 5 options which, within time spanned by a single grid cell, is the time of the trailing edge of the last cell in the time grid. It is also used to terminate the acquisition of data from UDF/IDFS data sets. In order, the inputs are: the full year, day of year, hour, minute and second.

## 5.5 Cell Time

The time in seconds spanned by a single cell in the internal time grid. This sets the time resolution of the grid and is used to determine the number of cells within the grid. Fractions of seconds are accepted.

# 6 Variable Definitions Menu

Importing measurements into the UDFAnalysis framework is initiated from the **Variable Definition** GUI menu which is invoked from the **Main Menu**. This brings up a parent or top level menu. The complete set of accessible menus are described below.

## 6.1 Parent Menu

The parent menu is used to select the type of data being imported which will be either **UDF** or **ASCII**. Each data type has its own GUI interface in which the data file, associated measurements and any ancillary information are defined. They are described separately. An unpopulated and populated example of the parent GUI are shown in figures 6 and 7 respectively.

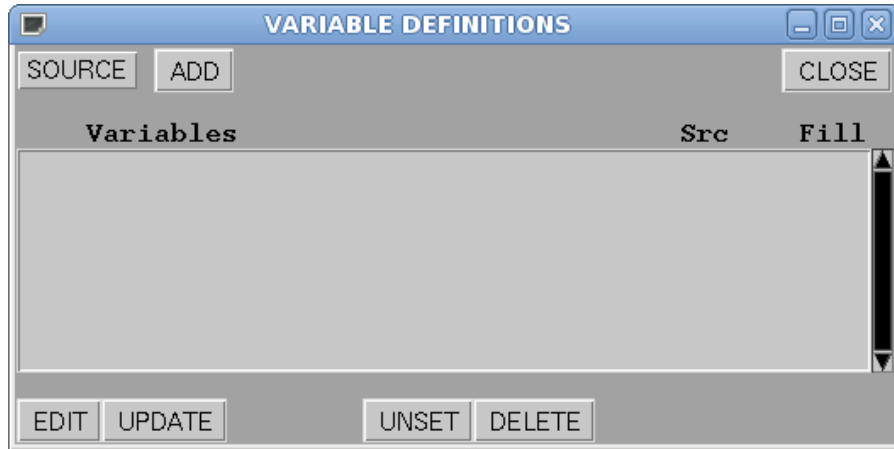


Figure 6: Unpopulated Parent Variable Definition Menu

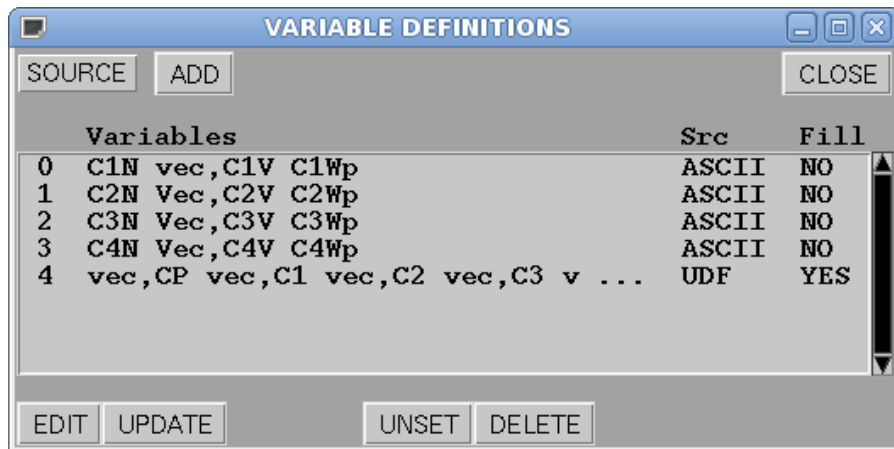


Figure 7: Populated Parent Variable Definition Menu

The menu consists of a central text window which shows a summary of the defined variables and a small subset of selected options. The first column of information in the text window is an offset into the variable definition structure where the options for this definition are kept. It is not editable by the user. This is followed by a list of the variables being imported into the UDFAnalysis framework, the data source type, and whether gaps in the data are being automatically filled.

Sources are read in the order shown in the text window, from top to bottom. You can change this order by highlighting a source and then using the up and down arrows to move it up and down in the list.

The options associated with the parent menu are described below.

### 6.1.1 Close

Close the menu window. The window can also be closed (and reopened) by clicking the **VARIABLE DEFINITIONS** button in the **Main Menu**.

### 6.1.2 Source

Sets the source of the next variable definition to be added. This is either **UDF** or **ASCII**. **UDF** includes the **IDFS** data format. Variables which have a UDF/IDFS source will auto-promote if they are not locally available when the menu is run. This entails locating a data source on the web and then retrieving the data and installing it locally. ASCII files must be physically present on the local system or manually retrieved.

### 6.1.3 Add

Creates an entry for the selected **SOURCE** type in the menu text window similar to those seen in the populated menu above. If there is a highlighted entry in the text window the new definition is placed above that entry. If no entry is highlighted the new definition is placed at the bottom of the list.

The variable field in the new definition is initially blank but the other fields are populated with default values. The Variable field is filled when data source options are set up. The GUI through which the options are accessed and set is brought up through the **EDIT** button.

### 6.1.4 Edit

This brings up the menu associated with the currently highlighted source definition in the text window. No action occurs if no source has been identified. The source definition menus are described in detail later and contain all of the options needed to both define the data source and to link named variables in the UDFAnalysis framework to the measurements being acquired.

### 6.1.5 Update

This updates the information on all of the lines in the text window reflecting any changes or additions to the variable name lists made when editing a definition.

### 6.1.6 Unset

This option unhighlights the currently highlighted source definition in the text window.

### 6.1.7 Delete

Deletes the currently highlighted source definition in the text window. This removes the entry line and clears all variables associated with the definition. *Not to be used lightly as there is no recovery.* No action occurs if no source has been identified for deletion.

## 6.2 ASCII Source Menu

**Editing** an ASCII source entry in the parent variable definition GUI brings up the menu:

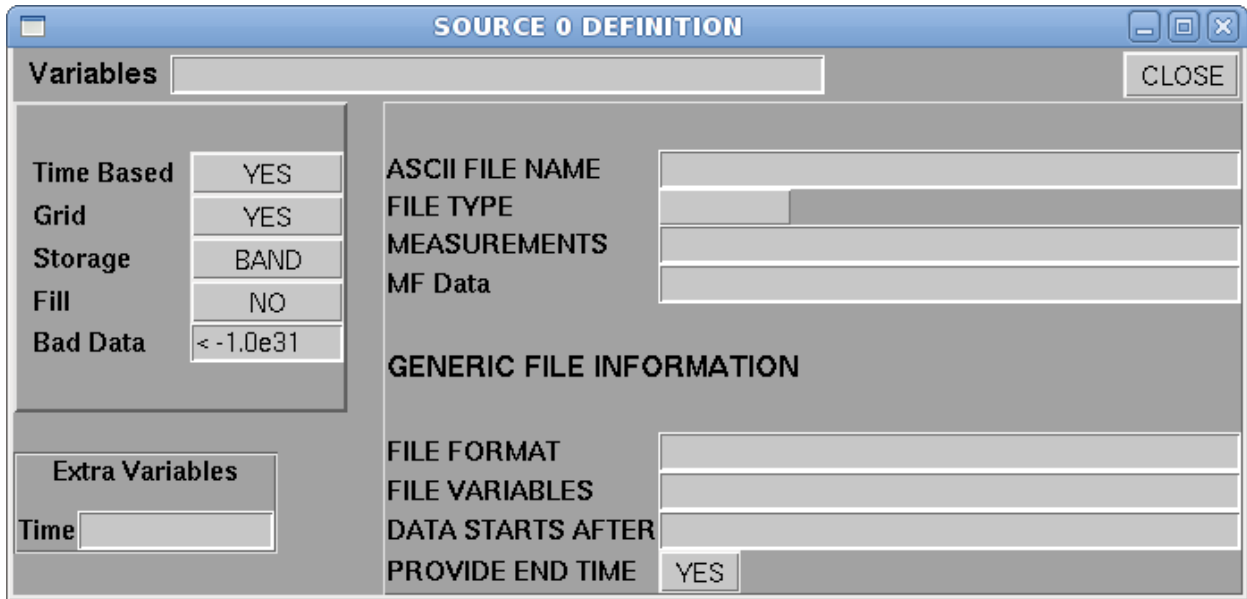


Figure 8: Unpopulated ASCII source menu

The menu consists of three sections, the definition of the internal variable names at the top, the variable storage options and extra variable definitions to the left, and the interface to the ASCII source at the right. You can have multiple versions of this window active at one time if you are editing multiple ASCII source definitions.

The options in the menu are described below.

### 6.2.1 Close

Close the menu window. The window can reopened by re-editing the variable entry.

### 6.2.2 Variables

The list of variable names associated with the input measurements. These names are used as references to the input measurements in all subsequent menus. There must be one variable name in the list for each measurement being input from this source. The first name in the list will be associated with the first measurement being imported, the second with the second measurement, and so on. Variable names are space separated and must be unique. You can use the **N.M.**, **vec**, or **ten**, prefixes to assign sets of measurements to sets of linked variables.

**NOTE:** Not all variables read in an ASCII file need to be imported into the UDFAnalysis framework.

### 6.2.3 Time Based

Set this option to **YES** if the measurements in the source are time based. Time based measurements have at least one time associated with them and possibly two (a start and a stop time).

#### 6.2.4 Grid

This option is valid only for time based measurements. Set this option to **YES** if you want the measurements to be stored in the system-wide time grid. If set to **NO** then the measurements will be kept at the resolution they are read in. In general you will want to keep this option set to **YES** but if you do elect to keep the variables at their native time resolution then you will probably also want to define variables for the associated measurement time under **Extra Variables**.

#### 6.2.5 Storage

The method used to store time based measurements in the time grid. This is a toggle option which switches between **BAND** and **POINT**. The **BAND** storage method spreads data across all of the cells in the grid which cover the duration of the measurement. Cells which are partially filled by a measurement are assigned a time weighted average of the value. Measurements which only have a single time associated with them use it as both the start and the stop time (zero time duration) in which case **BAND** storage is identical to **POINT** storage. With **POINT** storage, data is placed into the cell which contains the measurement starting time.

Data placed in a grid cell is added to any data which may already exist in it. Once the total data set has been accumulated each cell in the grid is normalized by the summed weighting factors of data stored in it.

**NOTE:** Non-time based data is always stored as an array of values beginning at 0 and ending at N-1 where N is the number of values read in.

#### 6.2.6 Fill

Set this option to **YES** to use a linear interpolation to fill unfilled cells in a data grid. Unfilled cells can occur for several reasons. Time based data stored using **POINT** storage can have unfilled cells when the measurements are stored in a grid which has a higher temporal resolution than the data itself. Data exclusion based on the next option can also result in unfilled cells in the data grids.

#### 6.2.7 Bad Data

This field establishes a data exclusion criteria. It consists of a conditional statement, either  $<$  or  $>$ , followed by a value. All measurement values which meet the condition are omitted from storage in the time grid. It is useful in many instances to remove fill data or data spikes. Any *holes* in the grid caused by data removal can be filled using the **Fill** option above.

#### 6.2.8 Extra Variables: Time

This option is used only if the measurements are **Time Based** and allows the start and stop time associated with the measurements be stored as independent variables. The times will be stored as deltas from the beginning time given in the **Time Definition Menu**. The first listed variable will hold the start time and the second the stop time. If there are no stop

times associated with measurements the second variable, if present, will be ignored. **Note:** This field should be left blank if no time variables need to be returned. The time variables are used if the data being input may need to be gridded or regridded.

### 6.2.9 ASCII File Name

The name of the file holding the measurement data to be read. If the file is not in the directory from which UDFAnalysis was invoked it must be prefixed with the full path name. It is always safest to use the full path name in the specification as this removes having to invoke UDFAnalysis from a specific location.

### 6.2.10 File Type

The type of ASCII file being accessed which can be either be **Generic** or **MOM-UDF**. **Generic** files have arbitrary format. The format as well as the file content is defined in the fields listed under the **Generic File Information** heading. **MOM-UDF** are ASCII files which contain plasma moments generated by the **MOments** application. For this type of file both the format and measurements are known so there is no need to describe how to parse the file or to specify its contents.

Figures 9 and 10 show examples of a populated ASCII definition GUI for a generic and MOM-UDF file respectively.

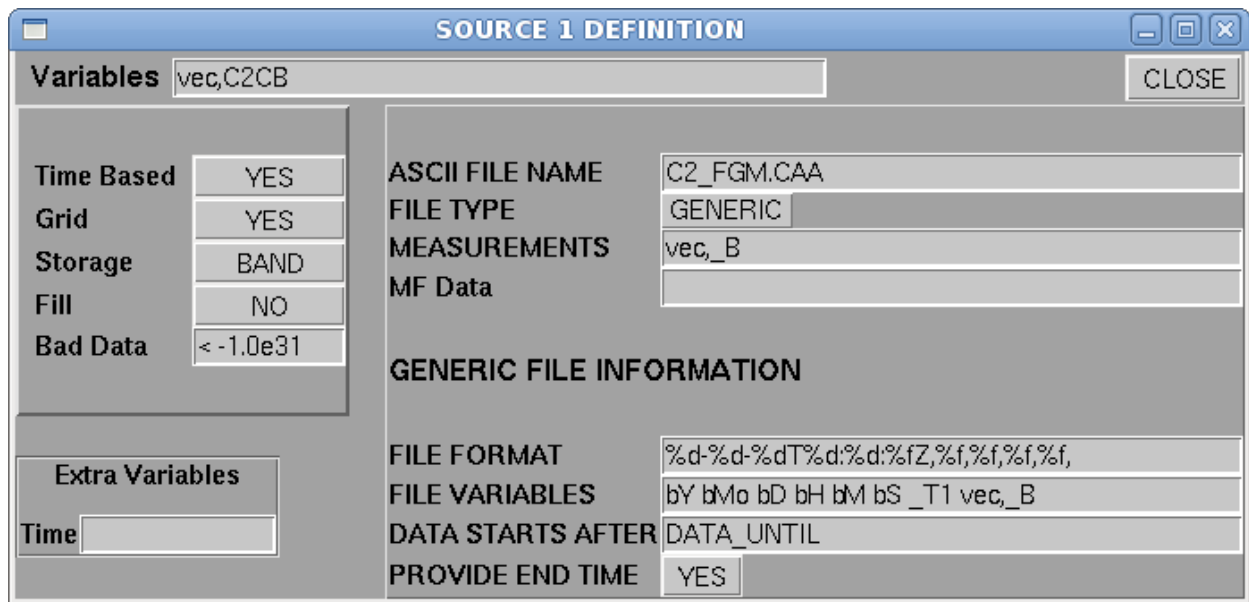


Figure 9: Populated Generic ASCII Source Example



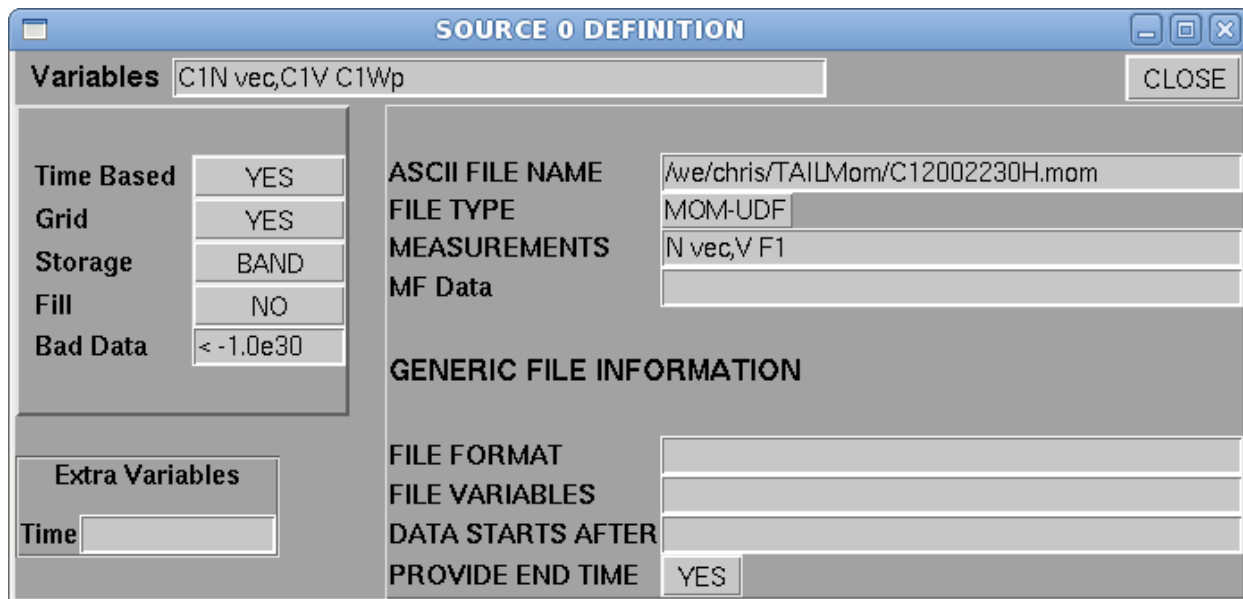


Figure 10: Populated Plasma Moments ASCII Source Definition

### 6.2.11 Measurements

The list of the measurements to be imported into the UDFAnalysis framework. This can be either all or a subset of the available measurements. There should be a one to one correspondence between the list of measurements being imported into the UDFAnalysis environment and the list of variable names given in the **Variables** field. The association between the imported measurements and the variable names follow the order in which they are specified.

With **generic** ASCII files the set of available measurements is listed in the **FILE VARIABLE** field under the **Generic File Information** heading in the menu. The measurements available to be imported into the UDFAnalysis environment have names beginning with an underscore (\_). Time variables, which do not have the preceding underscore, are not available for import.

When the source file is a **UDF-MOM** the measurements which can be imported are given the table below. All of the vector and tensor moments are returned in GSE coordinates unless the measurement name is preceded by a **b** (such as **bVx**) in which case the measurement is returned in a magnetic field based coordinate with the x direction along the field. You can import measurements in both coordinate systems. Unlike generic variables, these do not start with an underscore.

Measurement	Description	Units
N	Plasma Density	$m^{-3}$
Vx, Vy, Vz	Bulk Velocity Elements	m/s
Ex, Ey, Ez	Average Energy	eV
Vxx, Vxy, Vxz Vyx, Vyy, Vyz Vzx, Vzy, Vzz	Second Moment of Velocity	$m^2/s^2$

Tx, Ty, Tz	Plasma Temperature	eV
Pxx, Pxy, Pxz Pyx, Pyy, Pyz Pzx, Pzy, Pzz	Pressure Tensor Elements	Pascals
Vxxx, Vxxy, Vxxz Vxyy, Vxyz, Vxzz Vyyy, Vyyz, Vyzz Vzzz	Third Moment of Velocity	$m^3/s^3$
Qxx, Qxy, Qxz Qyx, Qyy, Qyz Qzx, Qzy, Qzz	Heat Flux Tensor Elements	eV-m/s
Qx, Qy, Qz	Equivalent to Qxx, Qyy, Qzz	eV-m/s
F1 thru F10	User Defined Additions	

The measurements **F1** through **F10** are associated with additional data which may have been added to the moments file when it was created. These were added at the users discretion and have no standard definition. Their contents will be listed in the header of the moments file.

### 6.2.12 MF Data

This option only pertains to UDF\_MOM ASCII files. If any of the measurements are being imported in the magnetic field aligned coordinate system then this field contains the variable name of the magnetic field vector components in GSE which are used to rotate the coordinate system. **This data must be read in prior to reading in the moment data.**

### 6.2.13 File Format

This field applies only to GENERIC ASCII files. A **generic** ASCII file is assumed to consist of a block of header information followed by a block of measurements. The block of measurements is assumed to of a set of identically formatted lines each containing a mixture of timing information and measurements of which the timing information is only present for time based data. The measurements in each line are assumed to be unique, that is the same measurement is not repeated in the line. The **File Format** field is the C format specification used to read the data line. As an example, consider the following line from an ASCII file of magnetic field data.

2007 020 13:59:59.9000, +1.070e-01, +2.557e+00, -1.490e+00

The format to read this line is given by

%d %d %d:%d:%f, %e, %e, %e

### 6.2.14 File Variables

This field links the elements associated with each format element specified in the **File Format** field to a variable name. The variable names are listed in the order that they are read on the line. The names associated with measurements (as opposed to time variables) are free form but must begin with an underscore (\_). Time variables when present on the line must conform to the following naming conventions:

Variable Name	Definition
bY	Beginning Year
bD	Beginning Day of Year or Day of Month
bMo	Beginning Month
bH	Beginning Hour
bM	Beginning Minute
bS	Beginning Second (may be fractional)
bMs	Beginning Millisecond
eY	Ending Year
eD	Ending Day of Year or Day of Month
eMo	Ending Month
eH	Ending Hour
eM	Ending Minute
eS	Ending Second (may be fractional)
eMs	Ending Millisecond

Keeping to the example of magnetic field data given in the previous section, the variables associated with the line of data shown could be listed as:

bY bD bH bM bS \_Bx \_By \_Bz

or similarly:

bY bD bH bM bS vec,\_B

were the **vec,\_B** expands to \_Bx, \_By, \_Bz

### 6.2.15 Data Starts After

This field identifies how to recognize the start of the data block or equivalently the last line in the file header information block. If there is no header information in the file and the data starts at the first line of the file then this field should be set to **\_FirstLine\_**. Leave this field blank if the data begins after the first blank line in the file otherwise provide the first word(s) on the line preceding the data. Note that these word(s) cannot appear as the first word(s) in any other line in the header block. As an example if the line **BEGIN DATA** precedes the data block then set the field to **BEGIN** or to **BEGIN DATA**.

**NOTE:** At times it might be necessary to add a line separating the header information from the data in the file if no unique separator exists. You also need to remove any trailing lines of information which may appear at the end of the file after the data block.

### 6.2.16 Provide End Time

If this is a time based set of data which only has a single time tag associated with it and if the data is regularly spaced in time then setting this option to **YES** will direct the read routine to supply an ending time to each measurement. This is done using an algorithm which looks at the time separation between adjacent the lines of data and determines the most likely value of the measurement duration to use for all data values. It should be emphasized that the ending time of a line is not just set to the beginning time of the next line. The use of a common measurement time to set the end times for each line ensures that time gaps in the data will be recognizable and preserved. The option is ignored both for non-time based data and if there is a second time tag provided in the data.

## 6.3 UDF/IDFS Source Menu

Editing a variable entry associated with an UDF/IDFS source opens the menu:

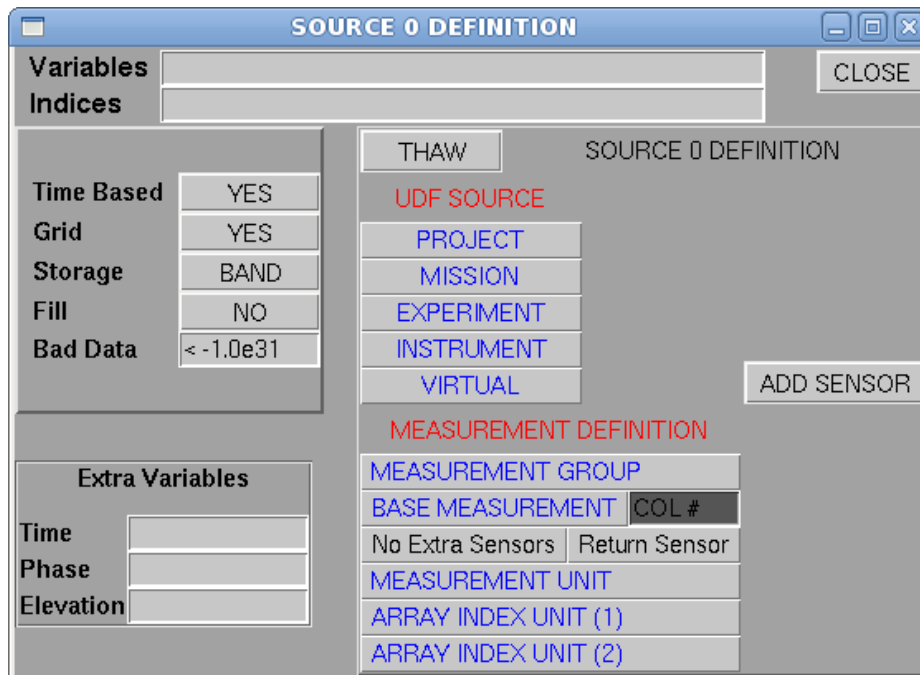


Figure 11: Unpopulated UDF/IDFS source menu

The menu consists of three sections, the definition of the internal variable names at the top, the variable storage options to the left, and the interface to the UDF/IDFS source at the right. The first two areas are identical to what is seen in the ASCII source menu above with the exception that the UDF supports a larger number of extra variables. The UDF definition portion of the menu is composed of two sections, the UDF source definition and a measurement definition. The latter is expandable. Clicking the **ADD SENSOR** button will duplicate the **MEASUREMENT DEFINITION** block allowing multiple measurements within the selected UDF source to be imported. You can have multiple versions of this

window active at one time if you are editing multiple UDF source definitions. A populated version of the menu is shown below followed by a detailed description of the menu options.

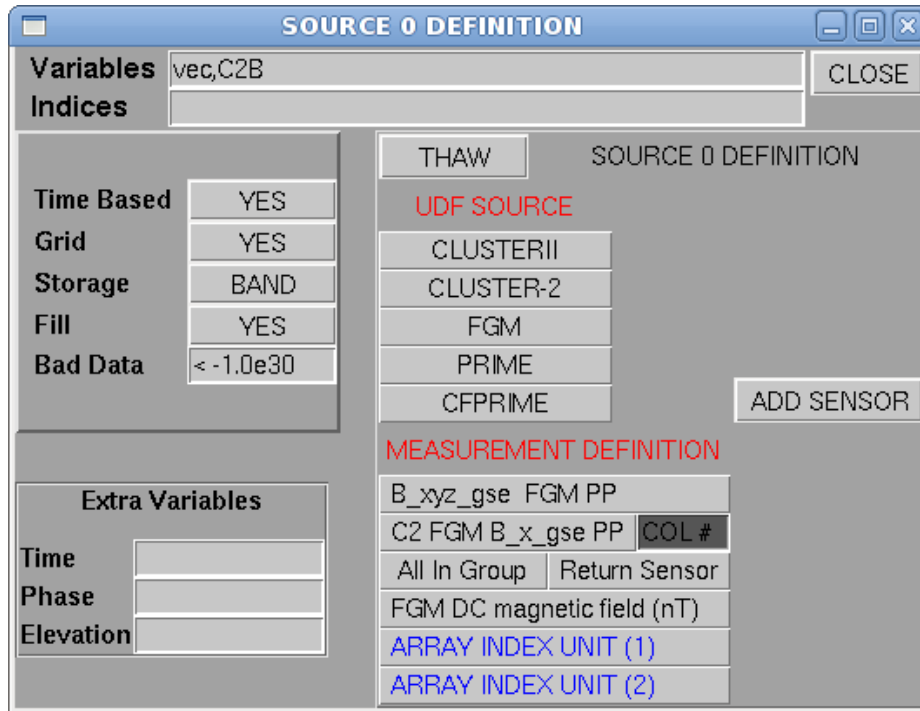


Figure 12: Populated UDF/IDFS source menu

### 6.3.1 Close

Close the menu window. The window can reopened by re-editing the variable entry.

### 6.3.2 Variables

The list of variable names associated with the input measurements. These names are used as references to the input measurements in all subsequent menus. There must be one variable name in the list for each measurement being input from this source. The first name in the list will be associated with the first measurement being imported, the second with the second measurement, and so on. Variable names are space separated and must be unique. You can use the **N.M.**, **vec**, or **ten**, prefixes to assign sets of measurements to sets of linked variables.

### 6.3.3 Indices

The variable name(s) associated with the input index measurements. The field should be left blank if no index measurements are being returned. When there are variables defined in this field, they will be filled on a first come first serve basis, that is, the first defined variable will be filled with the first defined index measurement, the second with the second, and so on until the defined variables are exhausted. Variable names are space separated and must

be unique. When multiple index variables are being returned you can use the **N.M.** prefix to assign measurements to a set of linked variables.

#### 6.3.4 Time Based

This option is ignored in a UDF/IDFS source since the data by definition are time based having both a start and stop time tag per measurement.

#### 6.3.5 Grid

Set this option to **YES** if you want the measurements to be stored in the system-wide time grid. If set to **NO** then the measurements will be kept at the resolution they are read in at. In general you will want to keep this option set to **YES** but if you do elect to keep the variables at their native time resolution then you will probably also want to define variables for the associated measurement time under **Extra Variables**.

#### 6.3.6 Storage

The method used to store time based measurements in the time grid. This is a toggle option which switches between **BAND** and **POINT**. The **BAND** storage method spreads data across all of the cells in the grid which cover the duration of the measurement. Cells which are partially filled by a measurement are assigned a time weighted average of the value. With **POINT** storage, data is placed into the cell which contains the measurement starting time.

Data placed in a grid cell is added to any data which may already exist in it. Once the total data set has been accumulated each cell in the grid is normalized by the summed weighting factors of data stored in it.

#### 6.3.7 Fill

Set this option to **YES** to use a linear interpolation to fill unfilled cells in a data grid. Unfilled cells can occur for several reasons. Time based data stored using the **POINT** storage method can leave unfilled cells when the measurements are stored in a grid which has a higher temporal resolution than the data itself. Data exclusion based on the next option can also result in unfilled cells in the data grids.

#### 6.3.8 Bad Data

This field establishes a data exclusion criteria. It consists of a conditional statement, either  $<$  or  $>$ , followed by a value. All measurement values which meet the condition are omitted from storage in the time grid. It is useful in many instances for removing fill data or data spikes. Any *holes* in the grid caused by data removal can be filled using the **Fill** option above.

#### 6.3.9 Extra Variables: Time

This option allows the beginning and ending times associated with the measurements to be stored as independent variables. The times are stored as deltas from the beginning time given

in the **Time Definition Menu**. This field should be left blank if no time variables need to be returned. The time variables are used if the data being input may need to be gridded or regridded. When there are variables defined in this field, they will be filled on a first come first serve basis, that is, the first defined variable will be filled with the beginning time of first defined measurement, the second with the ending time of the first defined measurement, and so on until the defined variables are exhausted. Since in most cases all of the defined measurements are returned at the same time, generally two variables will suffice.

### 6.3.10 Extra Variables: Phase

This option allows the beginning and ending phase angles associated with the measurements to be stored as independent variables. When there are variables defined in this field, they will be filled on a first come first serve basis, that is, the first defined variable will be filled with the beginning phase angles of first defined measurement, the second with the ending phase angles of the first defined measurement, and so on until the defined variables are exhausted. Since in most cases all of the defined measurements are returned at the same phase angles, generally two variables will suffice.

### 6.3.11 Extra Variables: Elevation

This option allows the beginning and ending elevation angles associated with the measurements to be stored as independent variables. When there are variables defined in this field, they will be filled on a first come first serve basis, that is, the first defined variable will be filled with the beginning phase angles of first defined measurement, the second with the ending phase angles of the first defined measurement, and so on until the defined variables are exhausted. Ideally there should be two variables per returned measurement. When there is only a center elevation for the measurement it will be returned in both the beginning and ending variables. If there is no elevation angle information available the values will be set to the current UDFAnalysis **bad data** value.

### 6.3.12 Freeze/Thaw

Disable (**FREEZE**) or enable (**THAW**) the cascade clearing of option settings below a changed UDF field. Changing the current **PROJECT**, **MISSION**, **EXPERIMENT**, **INSTRUMENT** or **VIRTUAL** setting causes all the lower options to be cleared (including the measurement options) and forces them to be reselected. This happens because the lower options generally depend on the current settings of the higher options. Turning on **FREEZE** allows you to change any UDF source option without the lower options being cleared. This is really only viable when dealing with a multi-satellite missions such as **CLUSTER** where the lower settings are often independent of the **MISSION**. (The options are the same for all spacecraft.) In this case **FREEZE** allows you to change the spacecraft without having to re-enter (in this case) all the same information to get to the same set of measurements.

In general use the **FREEZE** option with great care. Misuse can cause unforeseen problems.

### 6.3.13 Project

The **Project** under which the measurements to be imported are to be found.

### 6.3.14 Mission

The **Mission** under the selected **Project** which contains the measurements to be imported.

### 6.3.15 Experiment

The **Experiment** under the selected **Mission** which contains the measurements to be imported.

### 6.3.16 Instrument

The **Instrument** under the selected **Experiment** which contains the measurements to be imported.

### 6.3.17 Virtual

The **Virtual Instrument** under the selected **Instrument** which contains the measurements to be imported.

### 6.3.18 Measurement Group

A list of groups of measurements. Select the group which contains the specific measurement to be imported.

### 6.3.19 Base Measurement

A list of measurements under the selected **measurement group**. You will have the option to return either a single sensor from the list or all of the measurements in the list which have a common set of units. This is provided in the **EXTRA SENSORS** option. To return a single measurement, select that measurement. To return all measurements under the group with a common set of units select the first measurement in the list which fits your criteria.

Each of the measurements being returned must have an associated variable name in the **Variables** field.

### 6.3.20 Extra Sensors

This option allows you to import multiple measurements from a single measurement definition. To import only the selected measurement set the option to **No Extra Sensors**. To import all the measurements in the selected **measurement group** which have the same units as the selected measurement set the option to **All in Group**. To import all of the measurements in all groups which have the same units as the selected measurement set the option to all **All Defined**. With the latter setting you should select the base measurement from the first group in which a measurement you want imported is found.



It is important to understand when multiple measurements are being imported the order in which that takes place to correctly assign the variable names in the **Variables** field. Measurements are returned measurement block by measurement block where a measurement block is the defined set of options under a **MEASUREMENT DEFINITION** heading in the menu. There can be multiple defined blocks of measurements through the use of the **ADD SENSOR** button. Measurements in the first defined block are returned first followed by the measurements in the following blocks. Within each measurement block the first measurement returned is the selected base measurement. This is then followed by any additional measurements indicated by using the **All in Group** or **All Defined** option. With the **All in Group** option, the measurements within the group with the same units as the base measurement are returned beginning with the first measurement after the base measurement and proceeding cyclically back to the base measurement. With the **All Defined** option the measurements are returned as with the **All in Group** option except once the measurements in the selected **MEASUREMENT GROUP** have been exhausted the measurements in the remaining defined groups are looked at. Groups are searched cyclically beginning with the first group after the selected group.

The filled menu above illustrates the concept. Here the selected measurement group contains all three components of the magnetic field vector in the order Bx, By, Bz. By selecting **All in Group** all three sensors will be returned beginning with the selected sensor, Bx. These measurements will be associated with the variables C2Bx, C2By and C2Bz. Note the use of the **vec**, prefix to specify the components.

### 6.3.21 Return Sensor

Return either the selected measurement(s) or data associated with the selected measurement(s). The associated data varies but always includes the quality flags associated with the measurement. Most of the time you will leave this option in its default setting.

### 6.3.22 Measurement Unit

A list of units that the measurement can be returned in. Select the appropriate units that you want the measurement returned with.

### 6.3.23 Array Unit (1)

If the selected **base measurement** is returned as an array of values (as opposed to a single scalar value) then this option is a list of the units associated with the array indices. As an example, for a instrument which measures electrons as a function of energy this may be the energy or velocity associated with each element in the array.

### 6.3.24 Array Unit (2)

Allows the selection of an alternate **base measurement** to that selected in **Array Unit (1)**. This is often used when the array units being returned consist of an upper and lower measurement such as an upper and lower energy limit.

## 7 Functions Menu

Selecting and setting up functions to manipulate data within the UDFAnalysis framework is done through the **functions** GUI. The menu is accessed from the **Main Menu** and brings up the parent function menu. An unpopulated and populated version of the top level function menu is shown in Figures 13 and 14 respectively.

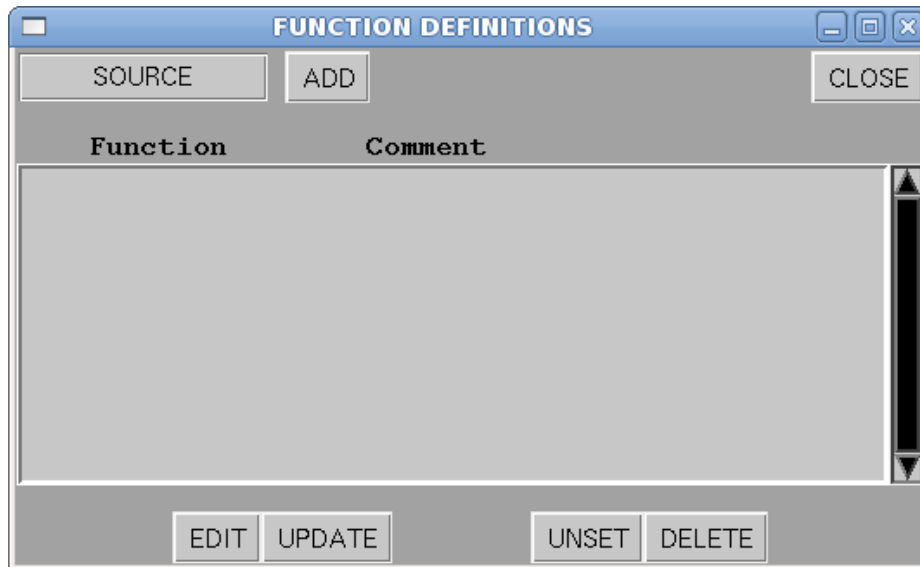


Figure 13: Unpopulated Function Definition Menu

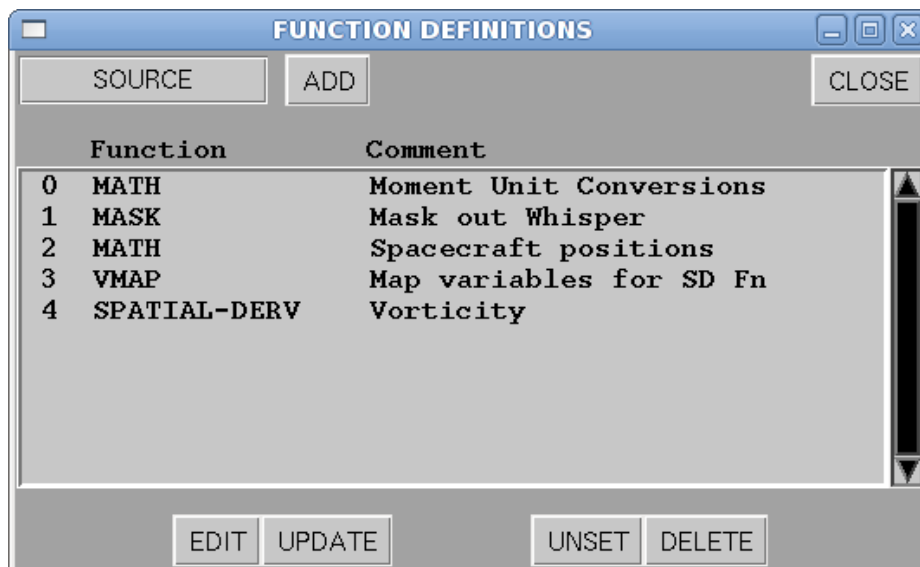


Figure 14: Populated Function Definition Menu

The menu is used to initiate and edit the options used in data manipulation. This consists of identifying the functions to run and setting up the function specific options. The latter is done in separate function specific set up menus invoked from this menu.

The top level function definition menu consists of a central text window which contains a summary of the defined functions and a number of options which add to or interact with the function definitions. The first column of information in the text window is an offset into the function definition structure where the options for this definition are kept. It is not editable by the user. This is followed by the function name and a short free form comment generally containing the purpose of the function call.

Functions are executed in the order shown in the text window, from top to bottom. You can change this order by highlighting a function and then using the up and down arrows to move it up and down in the list.

The options associated with this menu are described below.

## 7.1 Close

Close the menu window. The window can also be closed (and reopened) by re-clicking the **FUNCTIONS** button in the **Main Menu**.

## 7.2 Function

This is a list of the available functions. The function selected is added to the text window using the **ADD** button.

## 7.3 Add

Creates an entry for the selected **FUNCTION** in the menu text window similar to those seen in the populated menu above. If there is a highlighted entry in the text window the new definition is placed above that entry. If no entry is highlighted the new definition is placed at the bottom of the list.

The comment field in the new definition is initially blank and is filled when function options are set up. The options are accessed through using the **EDIT** button.

## 7.4 Edit

This brings up the menu associated with the currently highlighted source definition in the text window. No action occurs if no source has been identified. The set up menus are described in detail for each function in the section on **Function Plugins** below.

## 7.5 Update

This updates the information on all of the lines in the text window reflecting any changes or additions to the **comment field** made when editing a function definition.

## 7.6 Unset

This option unhighlights the currently highlighted source definition in the text window.

## 7.7 Delete

Deletes the currently highlighted source definition in the text window. This removes the entry line and clears all variables associated with the definition. **Not to be used lightly as there is no recovery.** No action occurs if no source has been identified for deletion.

## 8 Function Plugins

A function plugin is an application which modifies, combines or massages data in a specific way. There are a number of function definitions built into the UDFAnalysis framework which are invoked from the **Function Definition Menu**. Each function has its own unique setup menu. This defines the input variables to the function, supplies any options used to control how the function operates, and specifies the variables to hold the results. These will be described in detail later.

All function setup menus use a common layout format and share some common options. The differences are in the function specific options. Rather than repeat the usage of the common options under each function description we will describe them here. Figure 15 shows an example function setup menu, in this case for the MASK function.

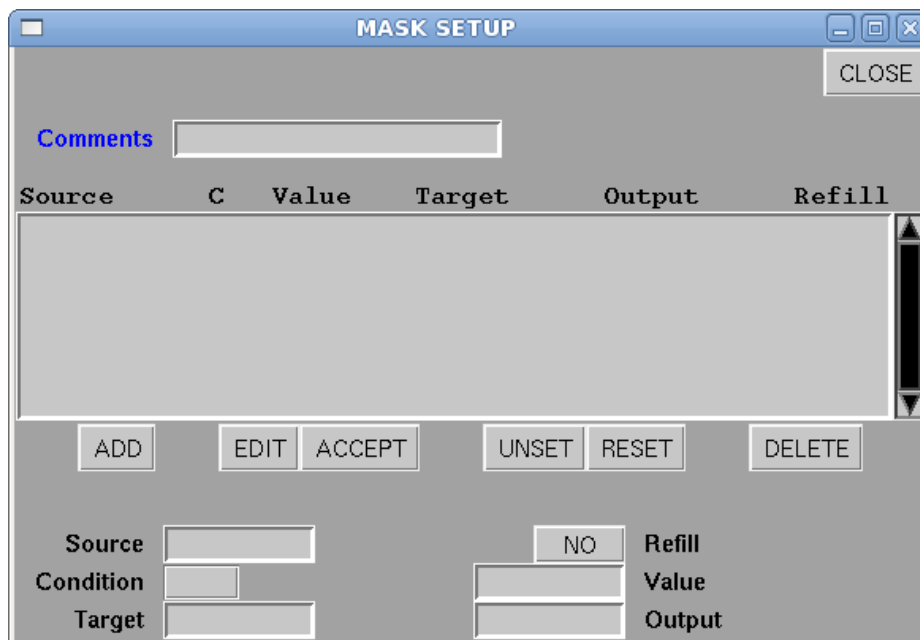


Figure 15: Example of function setup menu

The menu consists of three sections common to all function menus. At the top is the comment field. This is followed by a central text window below which is a set of buttons which allow interaction with information found in the text window. At the bottom is work area where the function specific inputs, outputs and any option settings are defined.

The first two sections of the menu are identical in all function setup menus. The only difference would be the contents of the text window which holds the function definitions. It

should be noted that any function definition can be set up to be executed multiple times using different input variables and options. This could equally well be accomplished by selecting the function multiple times from the **Function Definition Menu**.

The operations and options in these common menu setup sections are presented below.

### 8.0.1 Comment

A short free form comment field. This should describe the purpose of the function. It will be added to the function description line in the text window in the **FUNCTION DEFINITION** menu when you next update it.

### 8.0.2 Text Window

The text window contains the function definition. Each definition (and there may be multiple) occupies one or more lines. These contain the function option settings. When there are multiple definitions the function application is run multiple times starting with the first defined definition in the text window and continuing to the last. You can change this order by highlighting a function definition and then using the up and down arrows to move it up and down in the list. The buttons below the window allow new algorithm definitions to be added and current ones to be modified or deleted.

### 8.0.3 Add Button

Creates a new function definition line in the text box. The settings are taken from the current set of information in the work area.

### 8.0.4 Edit Button

Takes the options in the currently highlighted line and copies them down into the work area. There they can be modified. Clicking the **accept** button copies them back into the highlighted line. Edit can also be used to clone a function definition. **Edit** a definition, **unset** the highlight, and then **add** to create a copy of the definition. Edit the new definition to make what changes you want.

### 8.0.5 Accept Button

Copies the work area definitions into the currently highlighted line. It is used after editing the settings in a function definition to copy the changes back into the appropriate line in the text window. You can also clone the current work area setting into an already defined function by highlighting it and then clicking the **accept** button.

### 8.0.6 Unset Button

Unhighlights the currently highlighted line.

### 8.0.7 Reset Button

Resets the work area definitions to their default settings.

### 8.0.8 Delete Button

Deletes the function definition associated with the currently highlighted line. **Use with care.** There is no recovery of the options associated with this definition and no warnings that you are about to delete the line.

## 8.1 Bin Function

The **Bin** function takes a set of data associated with an arbitrary order variable and bins it according to value. Both the range over which the data is binned and the number of bins within the range are selectable. The routine is often used in setting up probability distribution functions. Components of the input variable are processed individually and produce separate outputs.

The binning is done by setting up a grid of  $N$  cells with the first cell beginning at  $V_b$  and the last cell at  $V_e$ . The cell width is given by  $(V_e - V_b)/N$ . The routine counts the number of values in the variable which fall into each bin cell and returns both a  $Y$  bin array (occurrences per bin) and an  $X$  bin array which contains the bin centers.

Figures 16 and 17 show examples of an unpopulated and populated set up menu for the bin function.

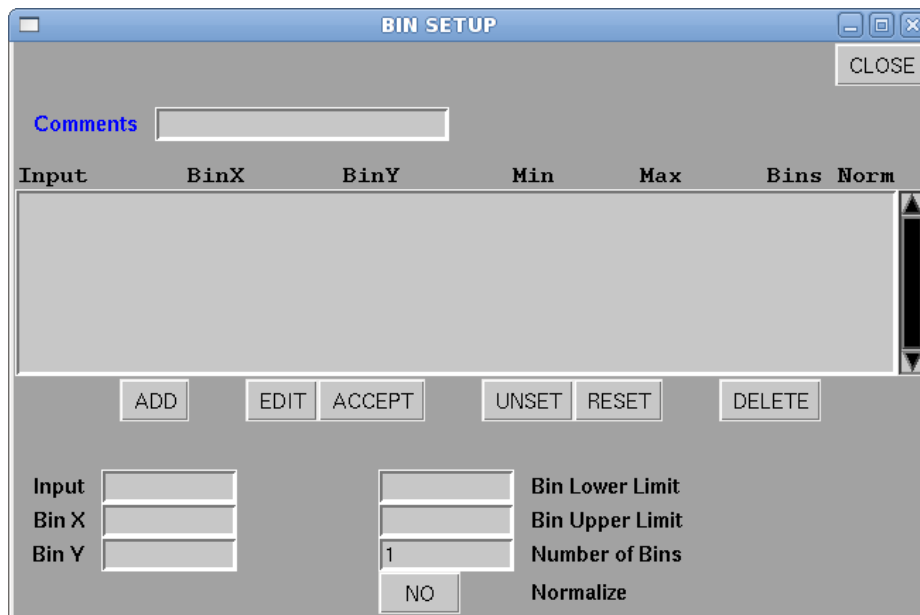


Figure 16: Unpopulated Bin Setup Menu

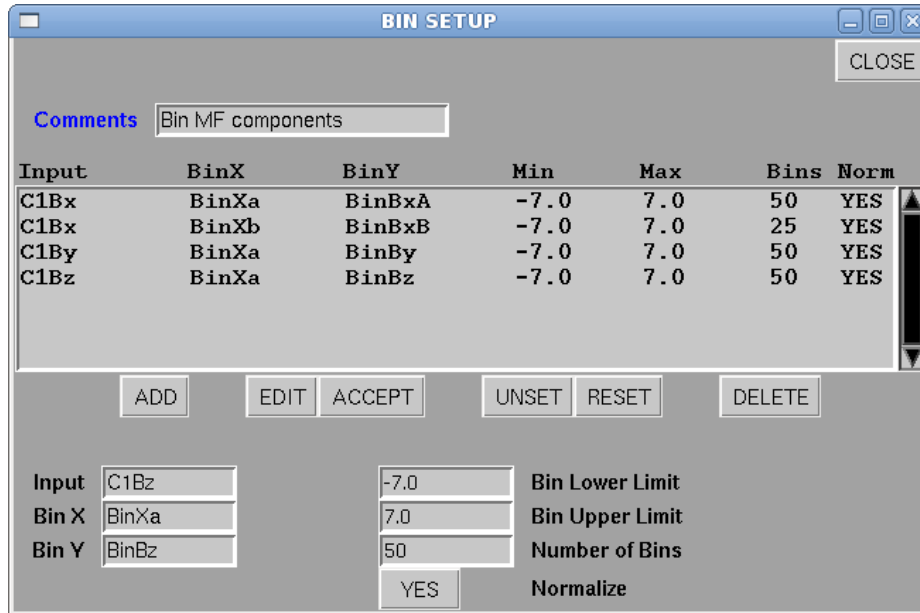


Figure 17: Populated Bin Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.1.1 Input

The variable(s) to be binned.

### 8.1.2 Bin X

The variable which will hold the returned bin centers. If running multiple instances of the function you can store the bin centers into the same variable for any definitions which bin over the same range and use the same number of bins. In each instance the bin centers will be the same in each execution.

### 8.1.3 Bin Y

This variable contains three separate fields of information. The array elements 0 through N-1, where N is the number of defined bins, holds the number of events in each bin. The array element **MaxP** contains the center value of the bin containing the maximum number of events and the array element **MaxV** contains the maximum. If the grid is normalized, the value in **MaxV** is what each cell was normalized to. This variable must be of the same order as the input variable.

### 8.1.4 Lower Limit

The value of the lower edge of the bin grid.

### 8.1.5 Upper Limit

The value of the upper edge of the bin grid.

### 8.1.6 Bins

The number of cells in the bin grid. The width of the bins is given by

$$\text{BinWidth} = (\text{UpperLimit} - \text{LowerLimit}) / \text{Bins}$$

### 8.1.7 Normalize

If set to **YES** the bin array is normalized to the bin with the largest number of occurrences.

## 8.2 Collapse

The **Collapse** function collapses a 2D grid over a specified range into a 1D grid. The collapse can be done over either the X or Y direction. The returned grid will have the same size as that of the uncollapsed axis.

Figures 18 and 19 show examples of an unpopulated and populated set up menu for the collapse function.

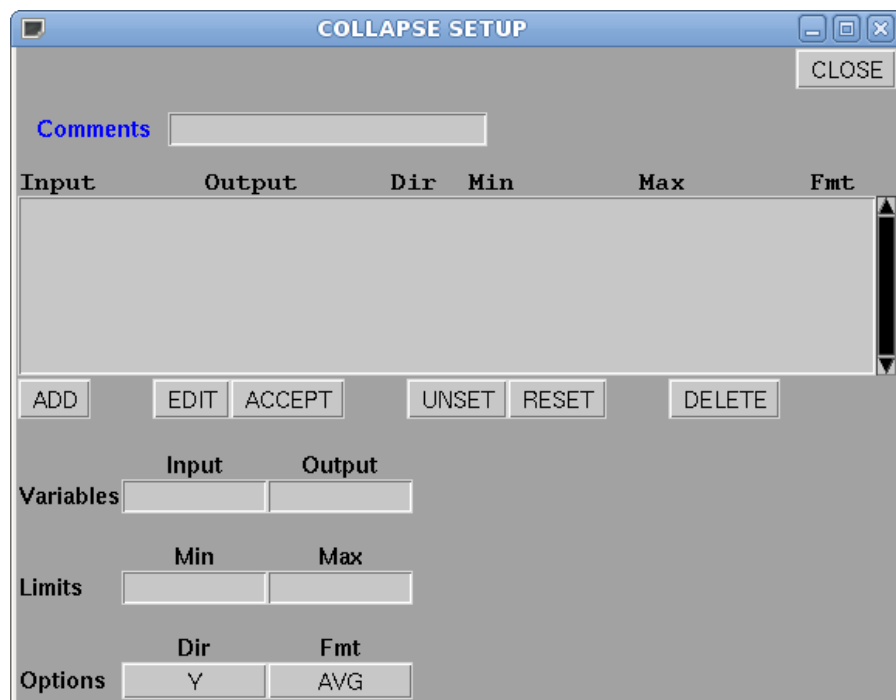


Figure 18: Unpopulated Bin Setup Menu



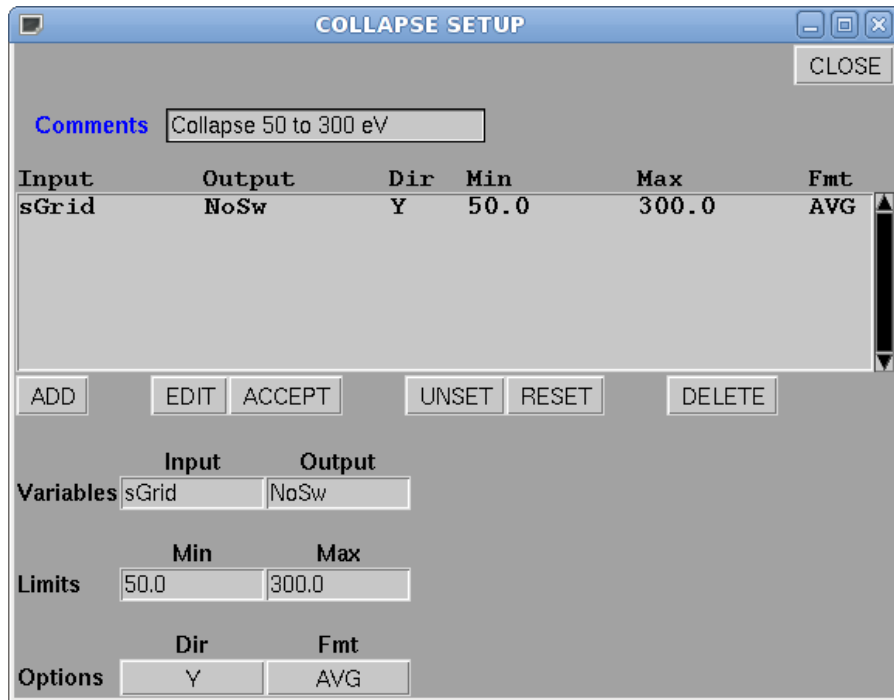


Figure 19: Populated Bin Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.2.1 Input Variable

The input variable representing the grid to be collapsed. The variable was presumably defined in a call to the **GridFill** function.

### 8.2.2 Output Variable

The variable which will contain the collapsed grid.

### 8.2.3 Min Limit

The minimum value at which to start the collapse.

### 8.2.4 Max Limit

The maximum value at which to end the collapse.

### 8.2.5 Dir

The direction in which to collapse the grid. This will be either **X** or **Y**.

### 8.2.6 Fmt

The format used to collapse the data. This will be either **AVG** or **SUM**. If the format is **SUM** then the data within the collapsed range is added together, otherwise it is added together and then divided by the number of bins within the collapsed range.

## 8.3 Conversion

The **Conversion** function performs basic unit conversions as well as some general conversions. The function works with arbitrary order variables.

Figures 20 and 21 shows an unpopulated and populated setup menu for the conversion function respectively.

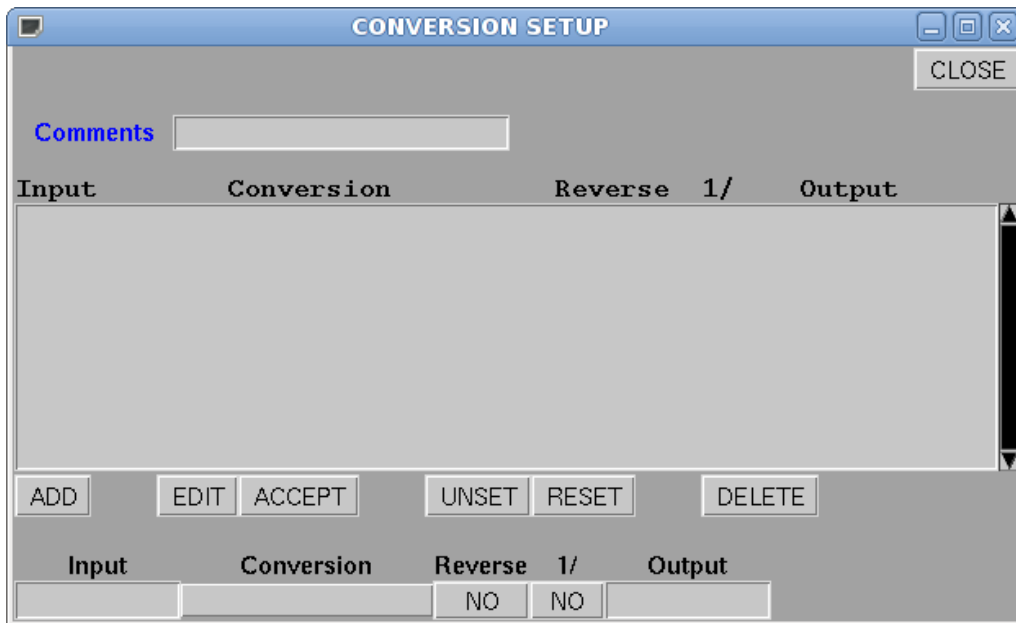


Figure 20: Unpopulated Cross Calibration Setup Menu

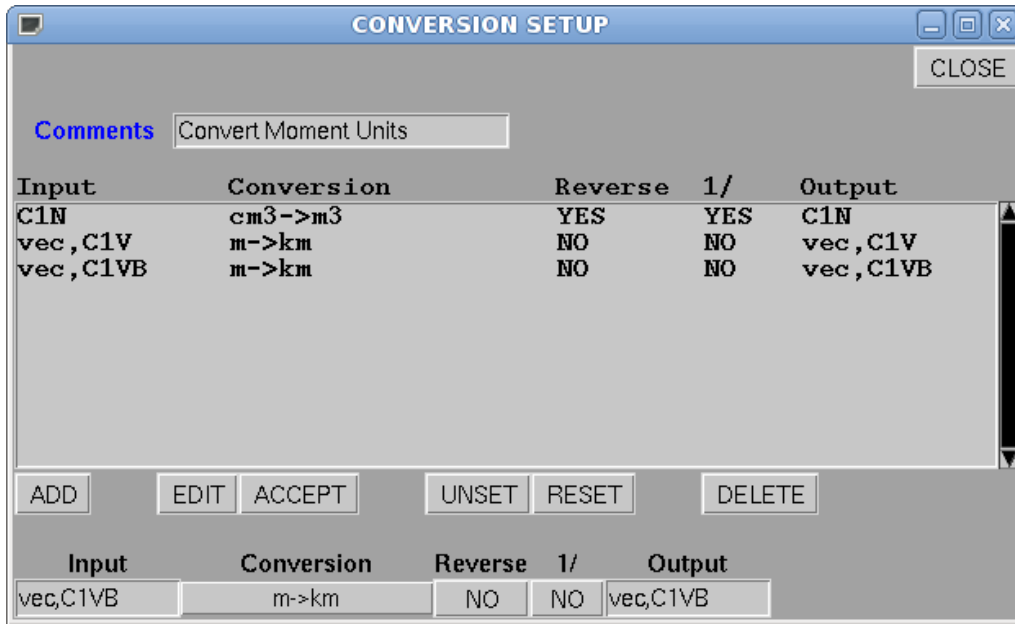


Figure 21: Unpopulated Cross Calibration Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.3.1 Input

The input variable. These are the variable(s) to which the conversion will be applied.

### 8.3.2 Conversion

The conversion to apply. These are shown in the table below.

Conversion	Action
Phi360->Phi180	Converts angles from $0^\circ \rightarrow 360^\circ$ to $-180^\circ \rightarrow 180^\circ$
Theta180->Theta90	Converts angles from $0^\circ \rightarrow 180^\circ$ to $-90^\circ \rightarrow 90^\circ$
cm->m	Converts centimeters to meters
cm->km	Converts centimeters to kilometers
m->km	Converts meters to kilometers
cm2->m2	Converts centimeters <sup>2</sup> to meters <sup>2</sup>
cm2->km2	Converts centimeters <sup>2</sup> to kilometers <sup>2</sup>
m2->km2	Converts meters <sup>2</sup> to kilometers <sup>2</sup>
cm->m3	Converts centimeters <sup>3</sup> to meters <sup>3</sup>
cm3->km3	Converts centimeters <sup>3</sup> to meters <sup>3</sup>
m3->km3	Converts centimeters <sup>3</sup> to meters <sup>3</sup>
deg->rad	Converts degrees to radians

### 8.3.3 Reverse

Set to **YES** to reverse the specified conversion. As an example, the conversion cm->m would be applied as m->cm.

### 8.3.4 1/

Set to **YES** to perform 1 over the specified inversion. As an example, the conversion cm<sup>3</sup>->m<sup>3</sup> would be applied as  $cm^{-3} \rightarrow m^{-3}$ .

### 8.3.5 Output

The output variable(s) holding the results of the conversion. This must be of the same order as the input variable. The results can be stored back into the input variable.

## 8.4 Cross Cal Function

The **Cross Cal**(ibration) function aligns variables to a common base variable. This is done adjusting the average value of each variable to match the average of the base value. The adjustment is done by shifting each variable by a constant additive or multiplicative value. If  $\langle A \rangle$  is the average of variable **A** and  $\langle B \rangle$  the average of the base variable, then the function will modify each element in **A** as:

$$A_i = A_i + \langle B \rangle - \langle A \rangle$$

when using the additive approach or as

$$A_i = A_i \frac{\langle B \rangle}{\langle A \rangle}$$

when using the multiplicative approach. In both approaches if **A** = **B** there is no change in **A**. Vectors can be cross calibrated component by component or as a whole by matching and shifting the vector magnitudes. The the latter case the cross calibrated vectors are constructed by multiplying the cross calibrated magnitudes by the unit vectors of the input vectors.

Figures 22 and 23 shows an unpopulated and populated setup menu for the cross calibration function respectively.

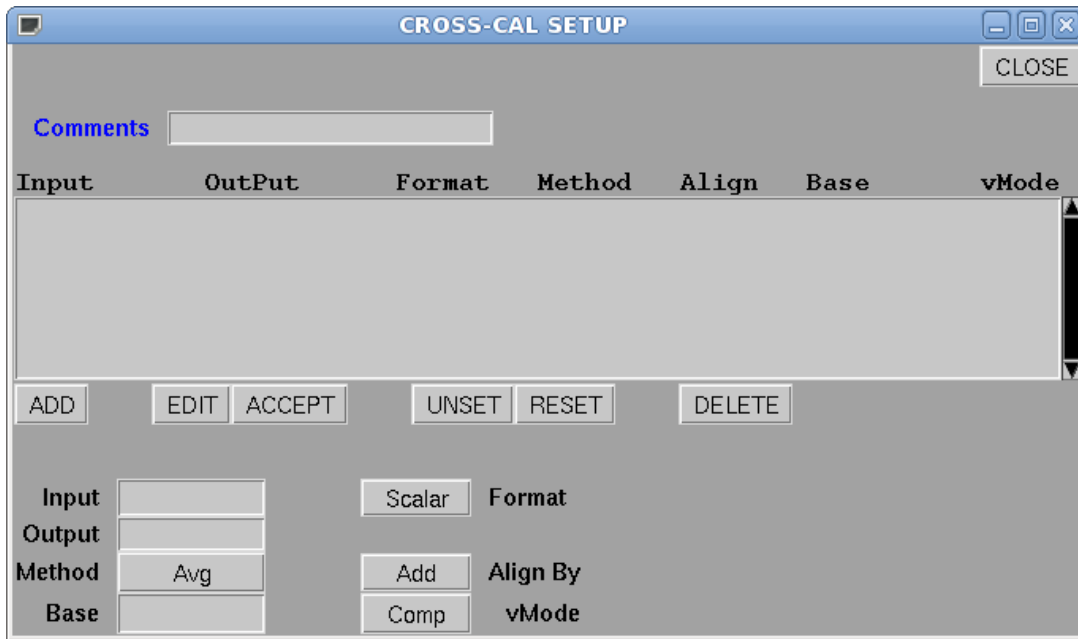


Figure 22: Unpopulated Cross Calibration Setup Menu

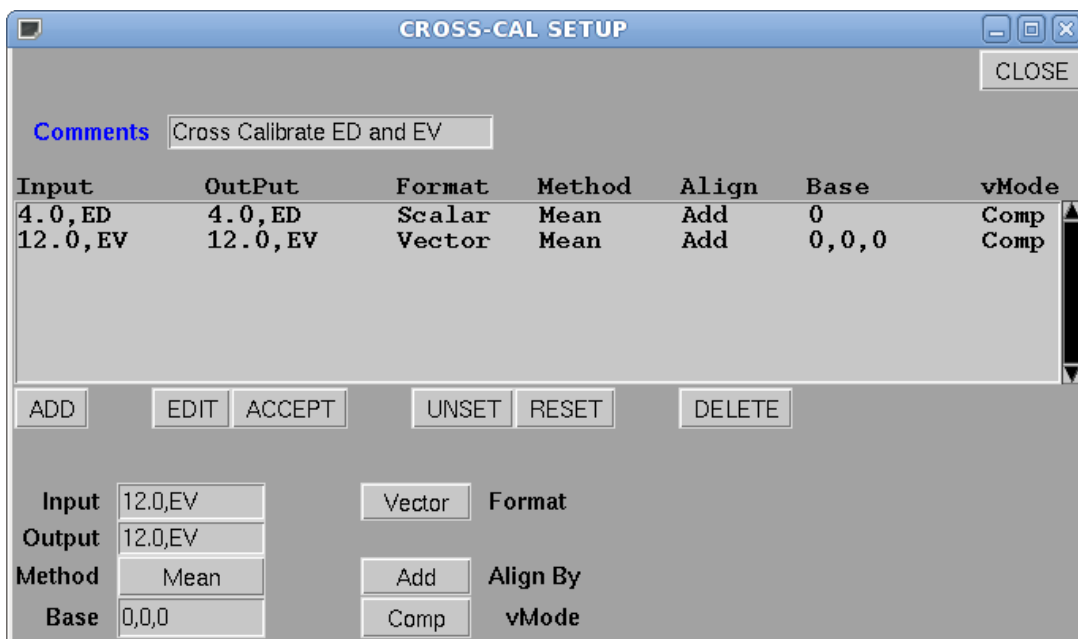


Figure 23: Unpopulated Cross Calibration Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.4.1 Input

A linked variable containing as components all of the measurements to be cross calibrated including the base measurement. The linked variables can be either vectors or scalars but not both.

### 8.4.2 Format

The type of variables which make up the input linked variable. This is either **Scalar** or **Vector**.

### 8.4.3 Output

The cross calibrated variables. This should be a linked variable of the same order as the input variable. The variables will be returned in the same order they are input.

### 8.4.4 Method

Determines how the base variable is set. The options are **Avg** and **Mean**. Use **Avg** if you want to calibrate to the average of all the input data rather than to a specific variable. Use **Mean** if you want to select the base variable from those in the Input variable.

### 8.4.5 Align By

Do the cross calibration by either adding an offset to the component variables or by multiplying them by a constant value.

### 8.4.6 Base

Select the base variable to use in the cross calibration if **Method** is set to **Mean**. For scalars this is an offset into the linked input variable to the variable to use as the base. For vectors this is a set of three offsets, comma separated without spaces, to the vectors whose x, y, and z variables are used as bases. Each component can come from a different vector.

### 8.4.7 VMode

Applicable only to vector variables. Set to **Comp** if the cross calibration is to be done component by component and set to **Mag** if only the vector magnitudes are to be used in the cross calibration.

## 8.5 Cross Helicity Function

The **Cross Helicity** function computes the vector Elsässer variables from the plasma density ( $\text{cm}^{-3}$ ), bulk plasma velocity ( $\text{km/s}$ ) and the local magnetic field ( $\text{nT}$ ). The equations used are:

$$\vec{E}_P = 21.8 * \frac{\vec{B}}{\sqrt{N}} - a\vec{B} + \vec{V} - a\vec{V}$$

$$\vec{E}_N = 21.8 * \frac{\vec{B}}{\sqrt{N}} - a\vec{B} - \vec{V} + a\vec{V}$$

where  $B$  is the magnetic field,  $N$  the plasma density and  $V$  is the bulk velocity. The term  $21.8 * \vec{B}/\sqrt{N}$  is the local Alfvén velocity and  $aV$  and  $aB$  are the mean plasma bulk and Alfvén velocities computed over the time period covered by the data. The removal of the mean in the equation is optional.

Figures 24 and 25 shows an unpopulated and populated setup menu for the cross-helicity function respectively.

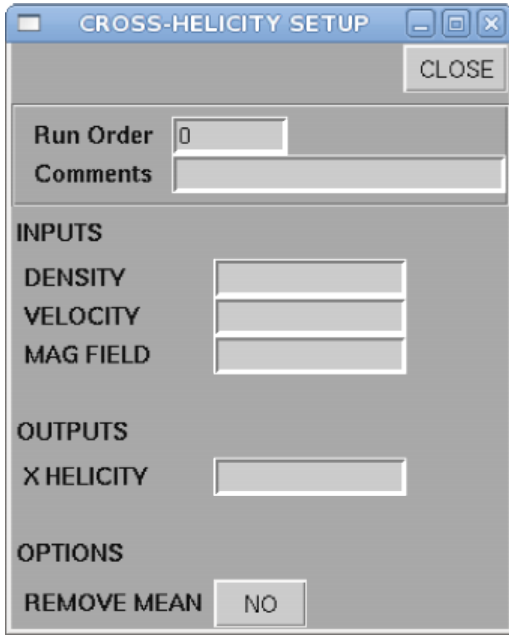


Figure 24: Unpopulated Cross-Helicity Setup Menu

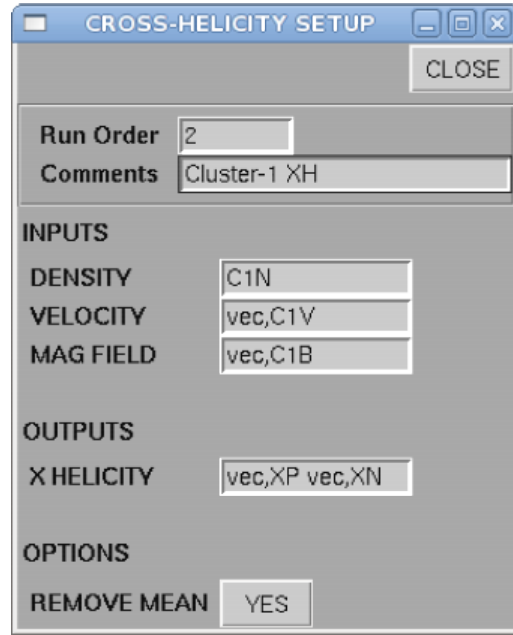


Figure 25: Populated Cross-helicity Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.5.1 Density

A scalar variable containing the plasma density in units of  $\text{cm}^{-3}$ .

### 8.5.2 Velocity

A vector variable containing the plasma bulk velocity components in units of  $\text{km/sec}$ .

### 8.5.3 Mag Field

A vector variable containing the magnetic field components in units of  $\text{nT}$ .

#### 8.5.4 Cross Helicity (+)

An vector variable which will contain the results of the first stated equation above (the outward propagating component of the the Elsässer computations).

#### 8.5.5 Cross Helicity (-)

An vector variable which will contain the results of the second stated equation above (the inward propagating component of the the Elsässer computations).

#### 8.5.6 Remove Mean

Set to **YES** to subtract the average bulk plasma and Alfvèn velocities in the Elsässer computations. If set to **NO** both aV and aB are set to 0 in the equations.

### 8.6 DefGrid Function

The **Def(ine)Grid** function sets up a grid information structure which may be used later to define data grids into which data can be stored. A data grid is a two dimensional set of cells extending over a fixed range in both the X and Y direction. Both the ranges and the number of cells in each direction can be different. The grid can be cyclic in one, both, or neither direction. The data which is being stored in the grid can be filtered to lie within a defined intensity range. If the measurements being stored have width in either X or Y they can be stored to cover multiple cells in the grid by setting the storage method to be **BAND**. A 1-D grid definition is possible by setting the number of cells along one of the two directions to 1. If the X range of the grid is left unset, the X-axis of the grid can be set to the same range as the internal time grid set up in the **Time Definition Menu**.

Figures 26 and 27 below shows both an unpopulated and a populated setup menus for the defgrid function.



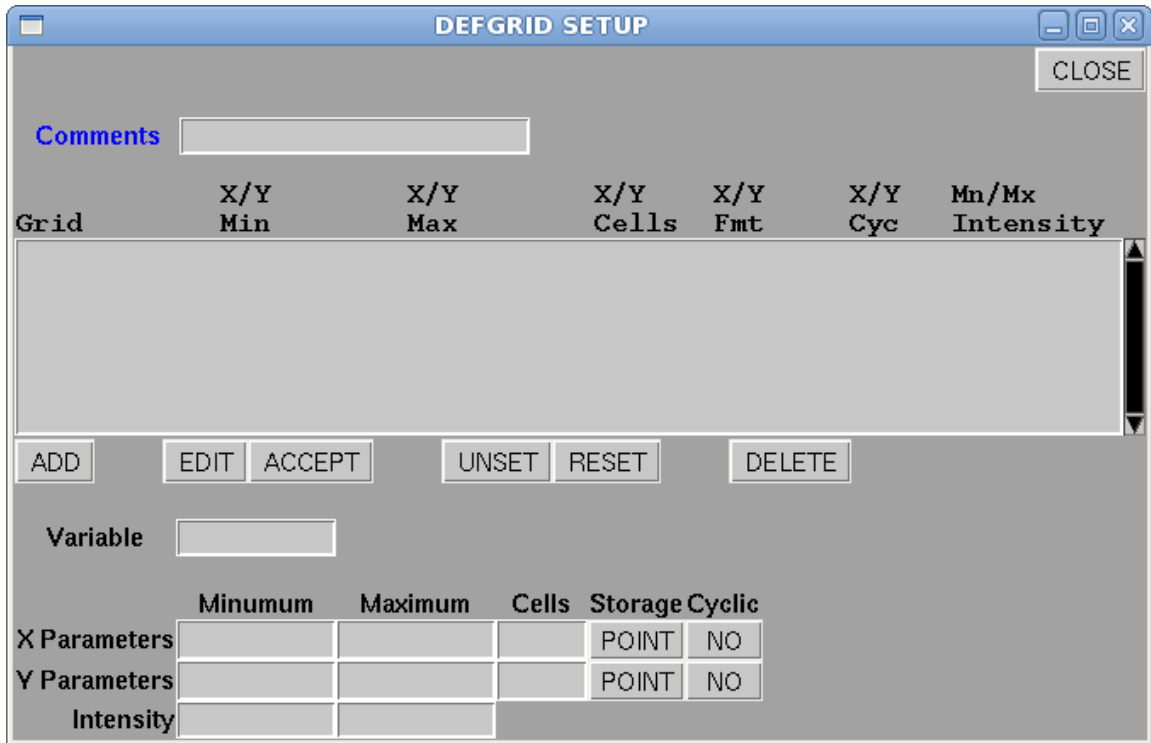


Figure 26: Unpopulated Define Grid Setup Menu

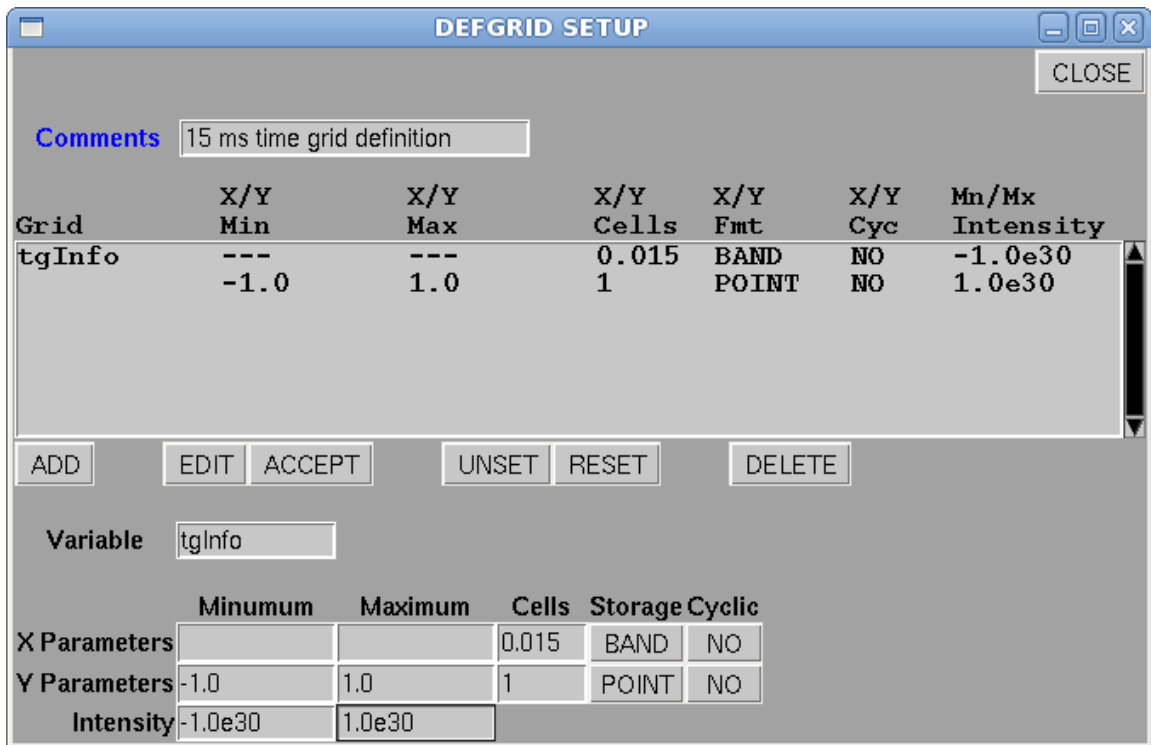


Figure 27: Populated Define Grid Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**. **Note:** A single function definition entries occupies two lines in the text window.

### 8.6.1 Variable

The variable name under which the grid definition will be stored.

### 8.6.2 X Minimum

This is the minimum value of the range covered by the X axis of the grid. If this value is left unset (blank) then it is internally set to the start time in the **Time Definition Menu**. If this field is left blank then the **X Maximum** value field should also be left blank.

### 8.6.3 X Maximum

This is the maximum value of the range covered by the X axis of the grid. If this value is left unset (blank) then it is internally set to the end time in the **Time Definition Menu**. This field should be left blank if the **X Minimum** value field has been left blank.

### 8.6.4 X Cells

If the X Minimum and Maximum values are defined then this is the number of grid cells along the X axis of the grid otherwise it is the time duration in seconds spanned by a single cell.

### 8.6.5 X Storage

The method to use when storing data along the X direction in the grid. This can be either **POINT** or **BAND**. The two methods are discussed in Section 2.1.3 with respect to the system-wide time grid but can be generalized to any measurement. The same caveats mentioned for time storage are valid for general storage.

### 8.6.6 X Cyclic

Set to **YES** if the X-axis of the grid is cyclic, **NO** otherwise.

### 8.6.7 Y Minimum

This is the minimum value of the range covered by the Y axis of the grid.

### 8.6.8 Y Maximum

This is the maximum value of the range covered by the Y axis of the grid.

### 8.6.9 Y Cells

The number of grid cells along the Y axis of the grid.

### 8.6.10 Y Storage

The method to use when storing data along the Y direction in the grid. This can be either **POINT** or **BAND**. The two methods are discussed in Section 2.1.3 with respect to the system-wide time grid but can be generalized to any measurement. The same caveats mentioned for time storage are valid for general storage.

### 8.6.11 Y Cyclic

Set to **YES** if the Y-axis of the grid is cyclic, **NO** otherwise.

### 8.6.12 Minimum Intensity

Any data with values less than or equal to this value are excluded from the grid.

### 8.6.13 Maximum Intensity

Any data with values greater than or equal to this value are excluded from the grid.

## 8.7 Dynamic-PS Function

The **Dynamic Power Spectra** function computes the dynamic power spectrum associated with a scalar variable. The variable must have been stored in a time-based grid, either the system time based grid or one created within the UDFAnalysis framework. The dynamic power spectrum is returned as a two dimensional grid of spectra with time along the X axis and frequency along the Y axis. There is no averaging of the output spectra. The number of cells along the X axis depend on the length of the input variable, the spectra window size and the size of the window advance. An estimate of the number cells can be obtained from the formula:

$$Cells = (N_p - W_p)/A_p + 1$$

where  $N_p$  is the total number of elements in the input variable,  $W_p$  is the number of points used per spectra, and  $A_p$  is the number of points to advance after computing a spectra.

The number of cells along Y in the grid as well as the frequency range covered depends on the method used to compute the spectra. When the spectra are computed using an FFT the number of cells is half the window length and the frequency range runs from 0 to the Nyquist Frequency. When using an MEM (Maximum Entropy Method) the number of cells is set to the number of frequency steps requested and the range will run from the specified beginning to ending frequency step.

Using an FFT requires that the window length be a power of 2. If this is not the case the number of points used are zero padded up to the nearest power of 2.

You can select to store multiple power spectra runs within a single power spectra grid. This makes sense only when using an MEM to produce the power spectra and when the input data has been frequency filtered into 2 or more frequency bands. When storing multiple dynamic spectra in a common grid the first definitions of how to compute the spectra will be used with the exception, in the case of MEM, of the number of coefficients to use.

Figure 28 and 29 below show an unpopulated and a populated setup menu for the dynamic power spectra function.

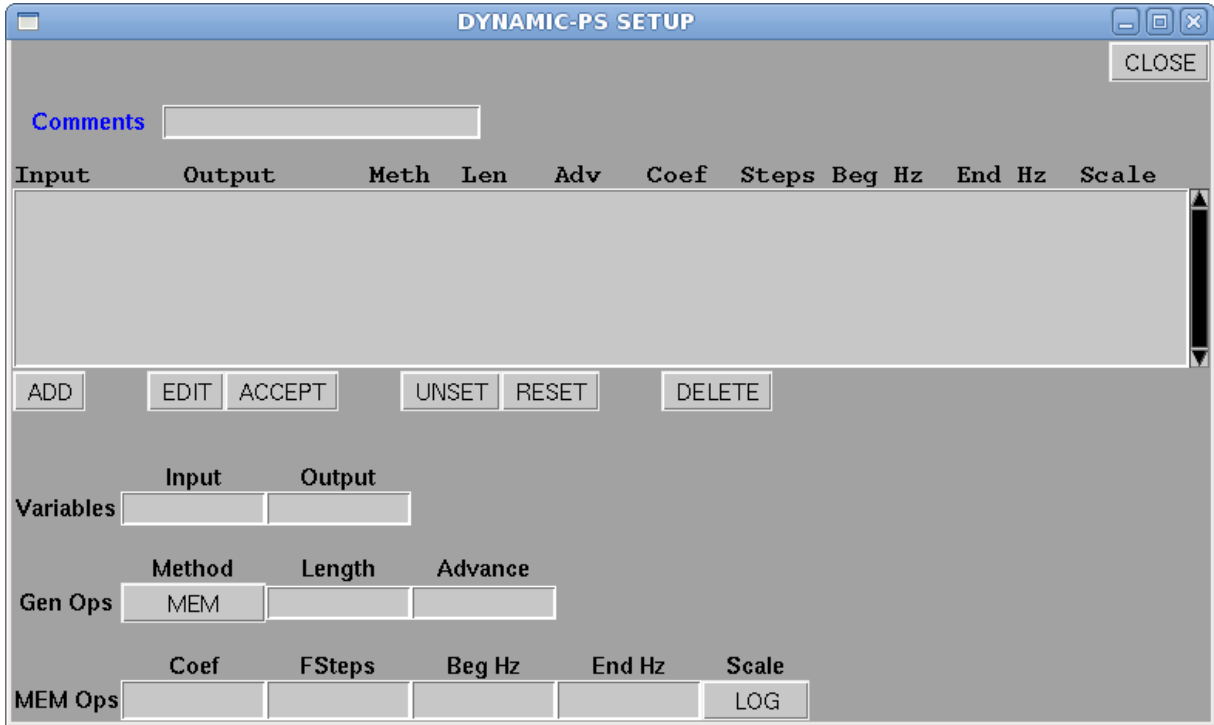


Figure 28: Unpopulated Dynamic Power Spectra Setup Menu

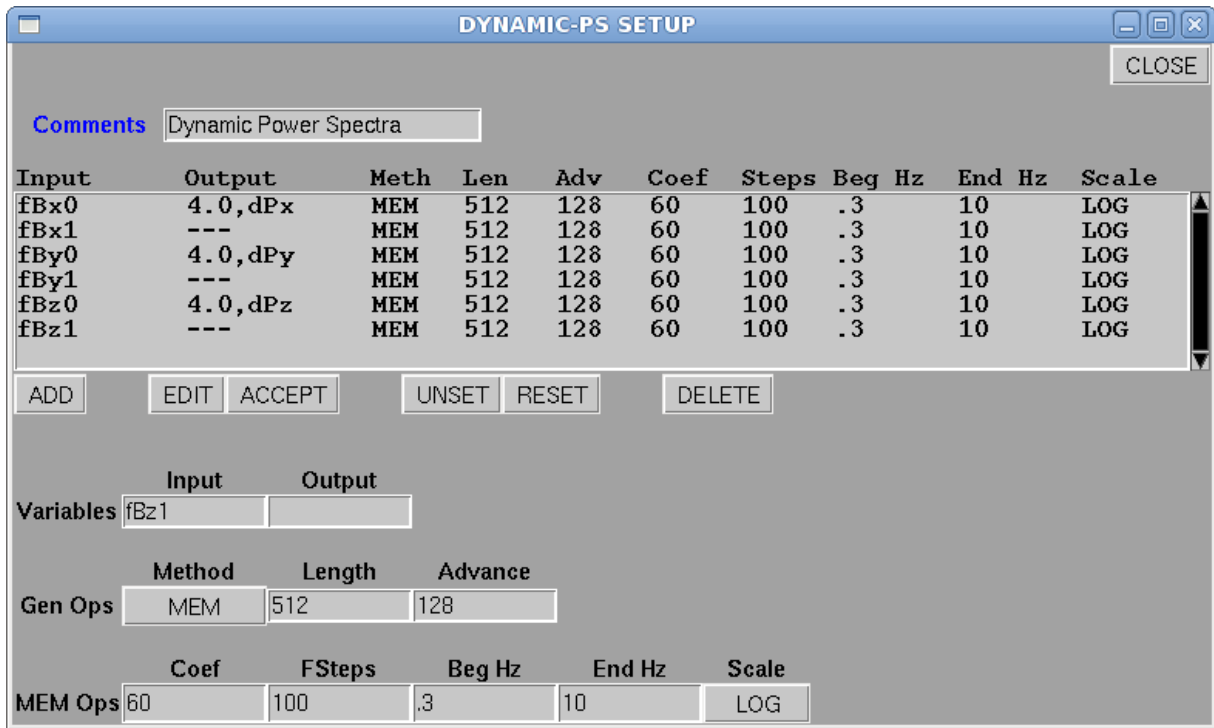


Figure 29: Populated Dynamic Power Spectra Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.7.1 Input Variable

The variable name of the measurement from which the dynamic power spectra will be constructed.

### 8.7.2 Output Variable

The variable name of the dynamic power spectra. Leave this field blank if you want the dynamic spectra to be added to the grid associated with the last defined output variable.

### 8.7.3 Method

The method used to compute the power spectra. This will be either **MEM** (Maximum Entropy Method) or **FFT** (Fast Fourier Transform).

### 8.7.4 Length

The window length which is the number of points to include in the computation of each spectra.

### 8.7.5 Advance

The number of points to advance the window after computing each spectra.

### 8.7.6 MEM Coef

The number of coefficients to use to compute an MEM based power spectra. This only has relevance if the selected method is **MEM**.

### 8.7.7 MEM FSteps

The number of frequencies at which to compute the power spectra when using the MEM method. The frequencies are equally spaced, either logarithmically or linearly between the specified start and stop frequencies. This value also sets the number of cells along the Y axis in the dynamic power spectra grid. This option only has relevance if the selected method is **MEM**.

### 8.7.8 MEM Beg Hz

The starting frequency in Hertz at which to compute the power spectra when using the MEM method.

### 8.7.9 MEM End Hz

The ending frequency in Hertz at which to compute the power spectra when using the MEM method.

### 8.7.10 MEM Scale

The scaling to use when dividing the frequency range to determine the locations at which to compute the power. This can be either **LINEAR** or **LOG**.

## 8.8 Equation

The **Equation** function solves arbitrary algebraic expressions. These can include constants and arbitrary order variables. All variables of order greater than 1 used in an equation must have the same length. Scalar variables can always be used with higher order variables. The expressions can include any of the normal C math functions such as sqrt, cos, atan2, log, exp, etc.

When variables of order greater than 1 are included in the equation, the expression is solved individually for each component.

Figures 30 and 31 below show an unpopulated and populated setup menu for the equation function.

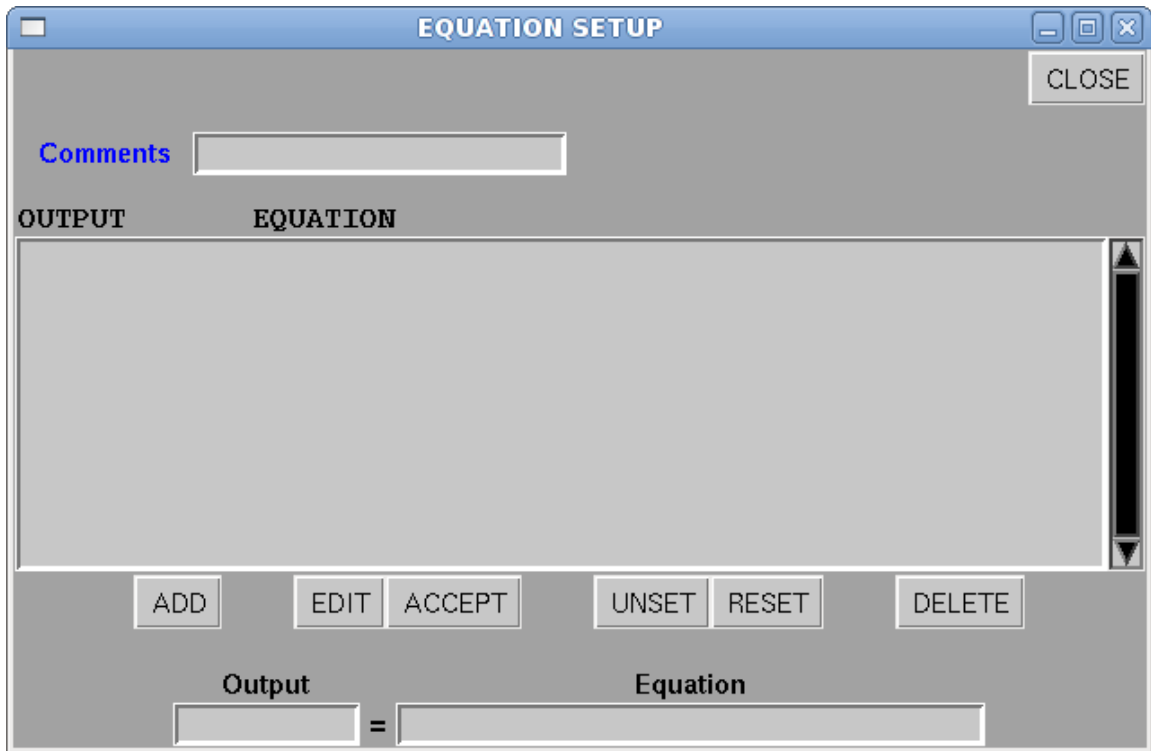


Figure 30: Unpopulated Equation Setup Menu

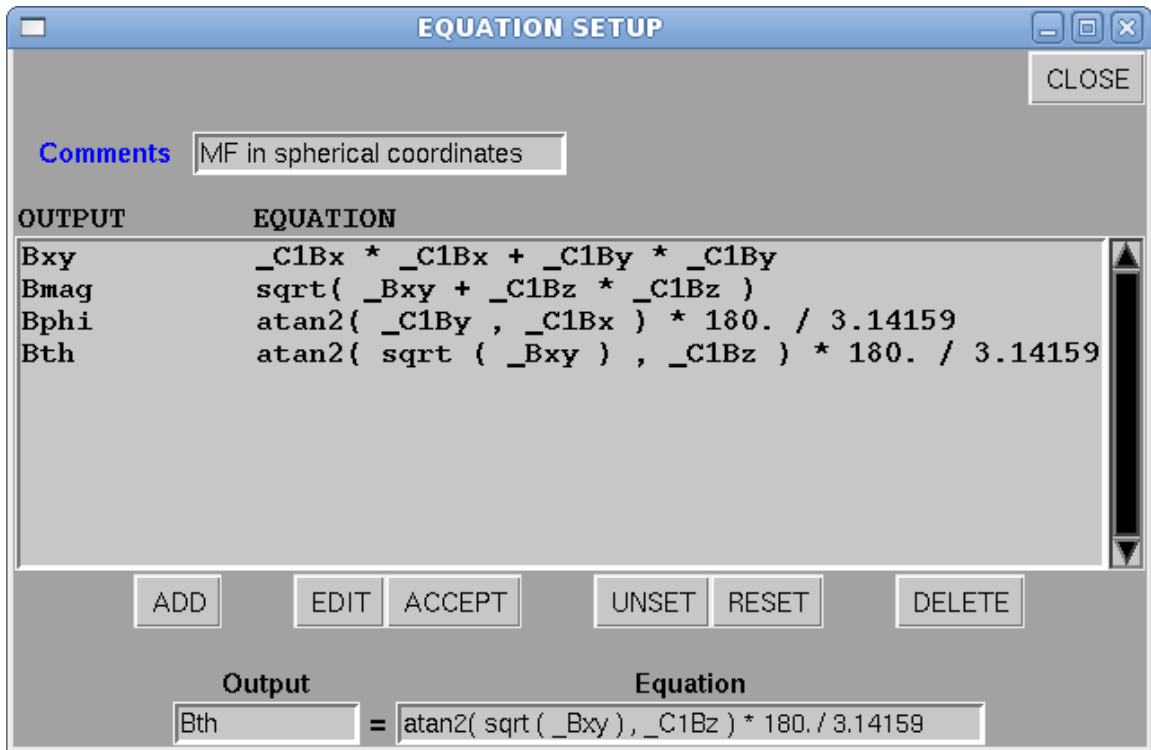


Figure 31: Populated Equation Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.8.1 Output

The variable which holds the solutions to the expression. This must be of the same order of the highest order variable included in the equation.

### 8.8.2 Equation

The expression to be solved. Variables are specified as defined within the UDFAnalysis framework except that they are preceded by an underscore (\_). The variable S would then be specified as `_S` and the vector variable `vec,V` as `vec,_V`. All variable names must be delimited by spaces unless they are the first or last item in the expression in which case the leading or trailing space can be dropped respectively.

Equations can contain constant values and any of the normal high level C functions such as `atan`, `exp`, `log`, `sin`, etc.

You can use variables of any order within an expression, however, with the exception of scalars you **cannot** mix variables within an expression of different orders. When using linked variables the expression is evaluated for each component. As an example, the equation

$$2.0 * \text{vec},_V + \_S + 1.0$$

expands to the three equations

$$2.0 * \_Vx + \_S + 1.0$$

$$2.0 * \_Vy + \_S + 1.0$$

$$2.0 * \_Vz + \_S + 1.0$$

and so requires that the output variable be of order 3.

The example below shows the the inclusion of a math function in an equation.

$$2.0 * \exp(\text{sqrt}(\_Vx * \_Vx + \_Vy * \_Vy + \_Vz * \_Vz))/ \_T )$$

**Note:** Only the variables need to be space delimited so that `2.0*exp(` for instance can be specified without spaces if desired.



## 8.9 Filter Function

The **Filter** function uses sets of low pass Savitsky-Golay filters to frequency filter data. The function can return data within, above, and below a defined frequency band. The pass band can be set to have zero width in which case there is no data within the pass band. The function is often used in the latter mode to remove the mean (i.e., very low frequency data) from a data set. For variable inputs of order greater than 1, the function is run individually for each component in the variable.

Figures 32 and 33 show examples of an unpopulated and populated setup menu for the filter function.

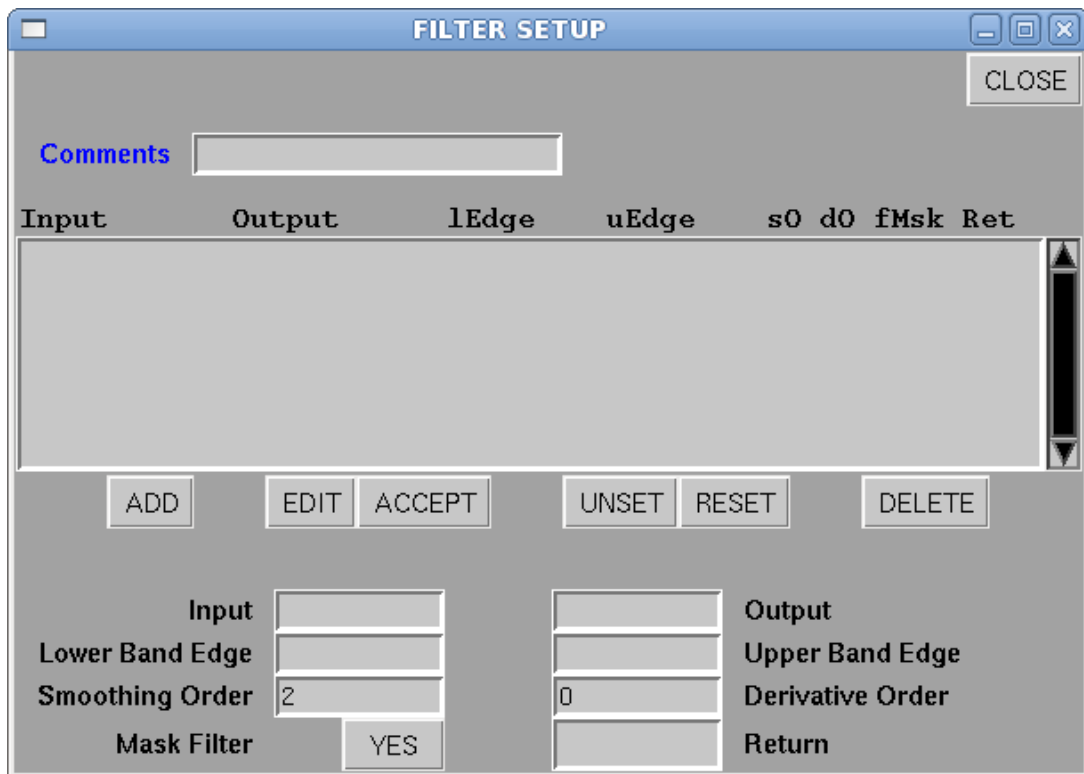


Figure 32: Unpopulated Filter Setup Menu

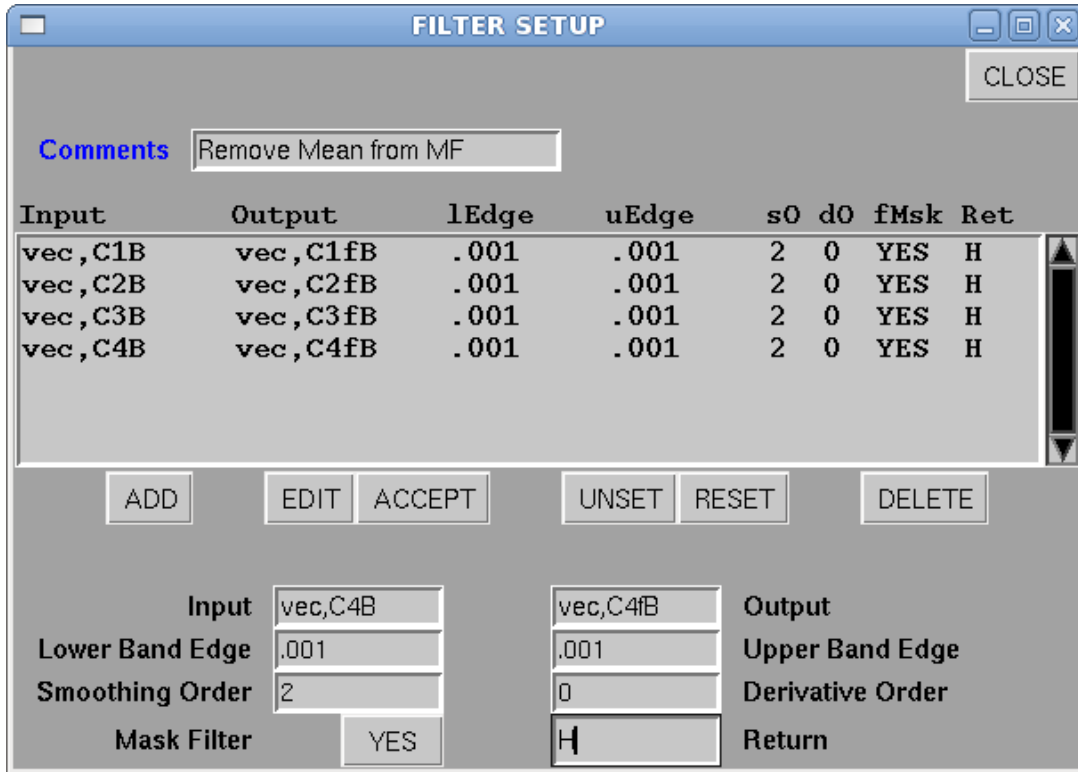


Figure 33: Populated Filter Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.9.1 Input

The variable name of the data to be filtered. The function accepts variables of any order.

### 8.9.2 Output

The variables which hold the filtered data. When computing a pass band filter, there can be a maximum of three variables returned, the data within, above and below the pass band. A maximum of two variables can be returned when the pass band has zero width. What it returned and the order it is returned in is determined by the **Return** option described below.

When using a linked input variable you need to make sure that there are enough output variables defined to hold the number of returns for each variables. Running a filter with a non-zero pass band and returning all three regions on a vector input requires output variable of order 9 (3 variables for each component of the input variable). The data is returned in the order specified in the **Return** option component by component.

### 8.9.3 Lower Band Edge

The lower edge of the band pass filter in **Hz**. If it is set to the **Upper Band Edge** then the band pass has zero width and only the filtered data above and below this frequency can be returned.

### 8.9.4 Upper Band Edge

The upper edge of the band pass filter in **Hz**. If it is set to the **Lower Band Edge** then the band pass has zero width and only the filtered data above and below this frequency can be returned.

### 8.9.5 Smoothing Order

The order of the fit polynomial used in the Savitsky-Golay computation. The default value of **2** is a good choice. Higher values tend to increase smoothing of narrow features in the data at the expense of broader ones.

### 8.9.6 Derivative Order

The order of the derivative to use in the Savitsky-Golay computation. For data smoothing you want leave this at the default value of 0. If you are looking to compute the numerical derivative of the data then you want to set this value to 1 or higher.

### 8.9.7 Mask

You can mask off the ends of the filtered data arrays. These are suspect due to the finite filter lengths used in the filtering. The amount of data masked off is half the used filter length which varies depending on the specified filter frequencies. If the option is set to **YES** the suspect cells in the data grids are set as unfilled.

### 8.9.8 Return

A string of maximum 3 characters specifying what quantities are to be returned. **H** indicates that the filtered data above the upper frequency cutoff is to be returned, **L** that the filtered data below the lower frequency cutoff is to be returned, and **B** that the filtered data within the pass band is to be returned. The latter only had meaning with a non-zero band width. The data is returned in the order specified in the option, hence, **LBH** would return the filtered data below the band in the first component of the output data, the filtered data in the band data in the second component of the output data, and the filtered data above the band in the third component of the output data. A return setting of just **L** would return only the filtered data below lower frequency cutoff.

## 8.10 Fit Function

The **Fit** function performs least square and non-linear 1-D fits to arbitrary data models and , 2-, and 3-D least square fits to arbitrary order polynomial models. The function returns

the fit coefficients as well as appropriate goodness of fit values ( $\chi^2$ , standard deviation, covariance values). The routine can also return the fit expanded and stored within a data grid identical in size to that of the input data. This is needed when the fit is to be used as input to other function calls but not if it is just to be plotted. The plotting routines will automatically expand a fit from the coefficients prior to output.

The fitting routine accepts data of the form  $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{V})$  and fits it to a model  $\mathbf{F}$  such that  $\mathbf{V} = \mathbf{F}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ . Fitting is carried using one of three different algorithms. Which algorithm is used depends both on the type of fit requested as well as the dimensionality of the data. The model characteristics are must be supplied to the routine as a Tcl procedure. There are a number of models currently available within the UDFAnalysis framework, however, if a suitable model is not found it is up to the user to provide the necessary procedure describing the model characteristics. The procedure specifics depend on the type of fit being performed and are described below.

**1D Least Squares Fit.** This fits a set of  $(\mathbf{X}, \mathbf{V})$  data to a generic linear model. The characteristics of the model are supplied to the fit routine through an external Tcl procedure. The procedure computes and returns the model basis function values at any  $\mathbf{X}$ . The interface to the procedure has the form:

```
proc FNAME X cM nC
```

where **FNAME** is the procedure name, **X** is the input value at which the model basis functions are computed, **cM** is the returned array of computed basis functions, and **nC** is the number of coefficients used in the fit. An example procedure **TUpolyFunc** is shown below. The basis functions for a polynomial are easy to compute being just  $\mathbf{X}^N$  for **N** running from **0** to **nC - 1**.

```
proc TUpolyFunc { X cM nC } {
  upvar $cM A
  # FIRST BASIS function is 1.0
  set A(0) 1.0   set J 0
  # COMPUTE succeeding basis functions by multiplying X by
  # the previous basis function.
  for { set I 1 } { $I < $nC } { incr I ; incr J } {
    set A($I) [expr $A($J) * $X]
  }
}
```

Routines similar to that shown above can be written for any model which can be expressed by a recursive algorithm such as those based on Bessel functions or Legendre polynomials. If there is no recursive relationship for the function you need to limit the number of coefficients to some reasonable value to prevent the procedure from becoming monolithic in size.

**1D Non-Linear Fit** This fits a set of  $(\mathbf{X}, \mathbf{V})$  data to a generic non-linear model using the Levenberg-Marquart method of solution. The characteristics of the model are supplied to the fit routine through an external Tcl procedure. The procedure computes both the value of the model fit and the values of the derivatives of the model with respect to the fit coefficients at any  $\mathbf{X}$ . The interface to the procedure has the form:

proc **FNAME** **nA** **dFdC** **InFo**

where **nA** is the number of coefficients being solved for, **dFdC** are the values of the derivative of the model with respect to each coefficient, and **InFo** is an array provided by the fitting routine containing values needed in the computation of both the value of the function and **dFdC**. The value of the function is returned through the procedure.

The procedure needs to have the array of X variables being fit as a global variable. This variable is always **\_X\_**. The **InFo** array contains the index of the **X** variable to use in forming the returned solutions in InFo(4) and and the current estimates of the fit coefficients in InFo(5) through InFo(5+nA-1).

An example of the procedure used in the fit of a set of data to the model

$$V = (AX^2 + BX + C)e^{-DX^2}$$

where **A**, **B**, **C** and **D** are the coefficients being solved for is shown below.

```

proc FunC { nA dFdC InFo } {
  global _X_
  upvar $dFdC dA
  upvar $InFo iN
  # THIS is the X value to solve the equations for
  set xV $_X_($iN(4))
  # THESE are the current values of the coefficients
  set A $oP(5)
  set B $oP(6)
  set C $oP(7)
  set D $oP(8)
  # COMPUTE X**2, the value of the exponent, and the terms
  # multiplying the exponent.
  set xS [expr $xV * $xV]
  set Exp [expr exp(-$D * $xS)]
  set Poly [expr $A * $xS + $B * $xV - $C]
  # THE value of the function which is returned through
  # the procedure is
  set yV [expr $Poly * $Exp]
  # THE derivitive with respect to A [X^2 * exp(-D * X^2)]
  set dA(0) [expr $xS * $Exp]
  # THE derivitive with respect to B [X * exp(-D * X^2)]
  set dA(1) [expr $xV * $Exp]
  # THE derivitive with respect to C [-exp(-D * X^2)]
  set dA(2) [expr -$Exp]
  # THE derivitive with respect to D [X^2 * yV]
  set dA(3) [expr -$xS * $Poly * $Exp]
  return $yV
}

```

In addition to the external routine which supplies the characteristics of the model, the non-linear fit also requires a initial guess of the model coefficients. The better the guess the faster the convergence is reached in the solution. If the guess is to far off sometimes convergence is never reached.

**2 and 3D Least Squares Fits** These fits work on 2 and 3D data sets. Because they only fit to a model polynomial neither requires an external user supplied function.

Figures 34 and 35 shows an unpopulated and populated setup menu for the fit function.

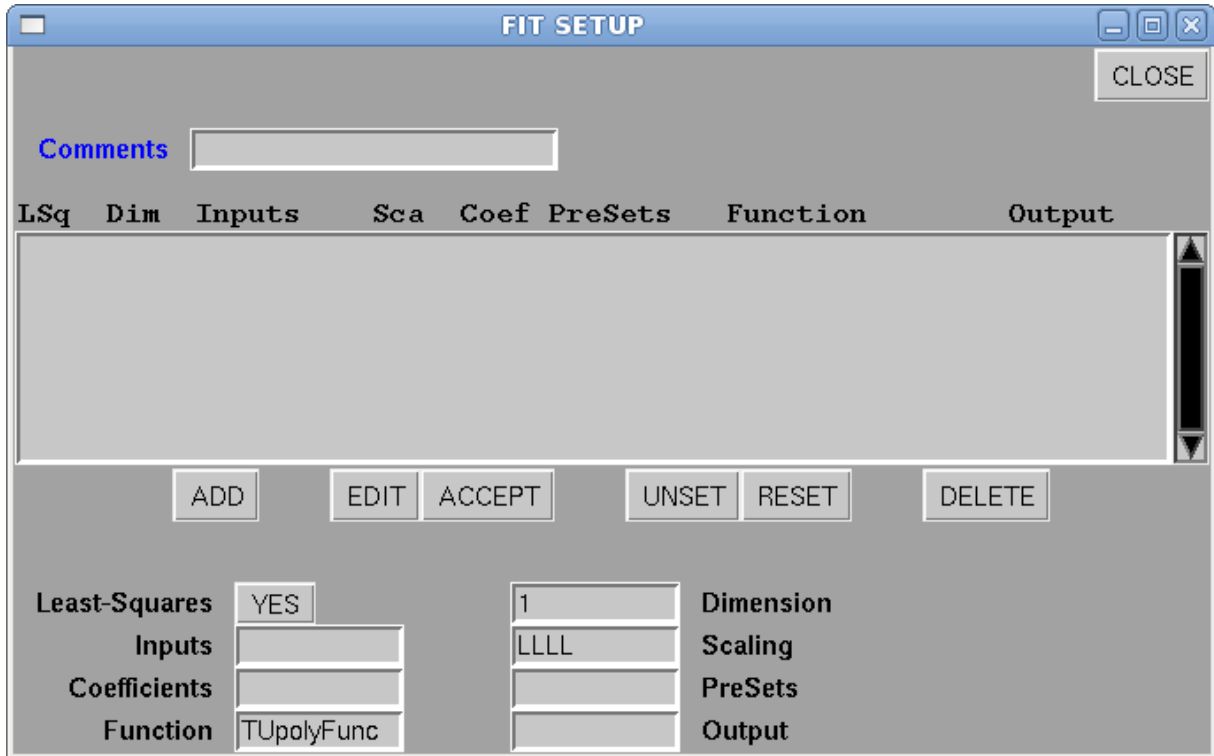


Figure 34: Unpopulated Fit Setup Menu

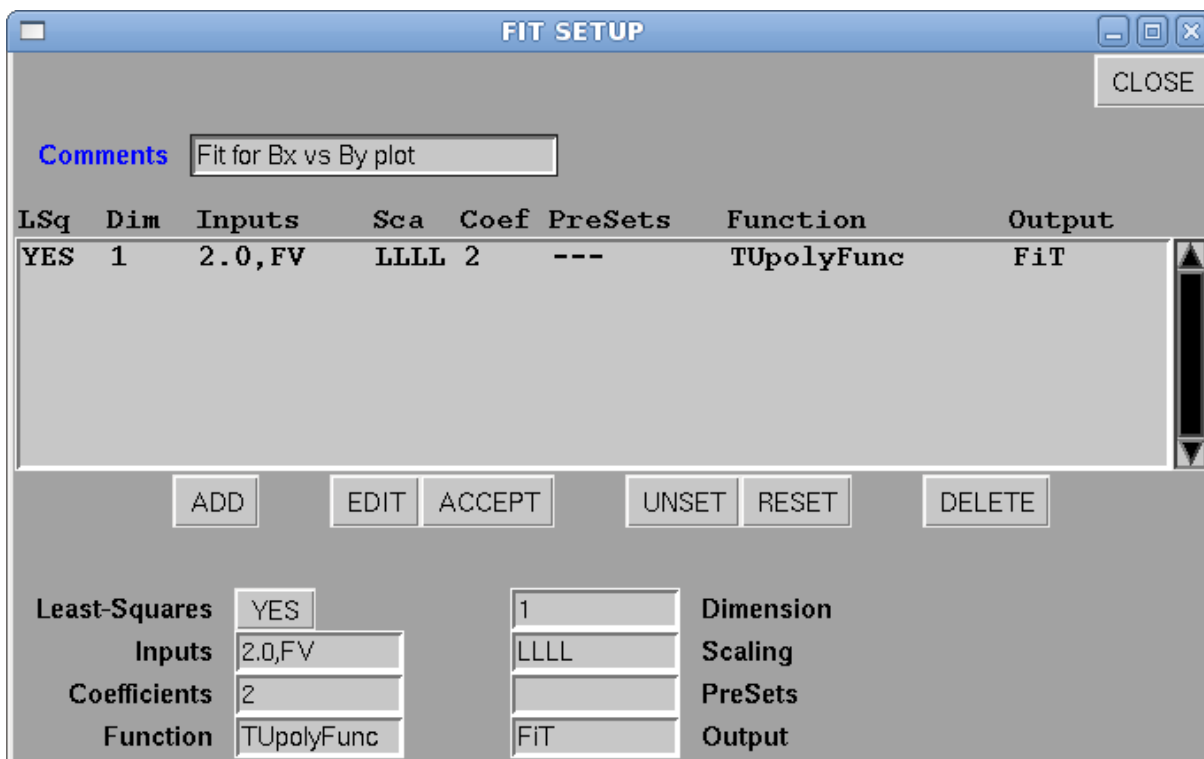


Figure 35: Populated Fit Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.10.1 Least -Squares

Set to **YES** if the fit is to be made using the least-squares fitting algorithm. Set to **NO** if the Non-Linear fitting algorithm is to be used. The Least Squares algorithm works with any dimension data while the Non-Linear algorithm works only with 1D data. Both Least Squares and Non-Linear fits to 1D data sets require an external Tcl routine to describe the model characteristics.

### 8.10.2 Input

The input data as a linked variable which is generally set up using the **VMAP** function. The number of components depends on the dimension of the function being fit and is generally 1 larger than the dimension. In their component order in the input variable, 1D fits should contain the X then V variables, 2D fits the X, Y, and V variables, and 3D fits the X, Y, Z, and V variables. If there is a Weighting function associated with the V variable it should be included as the last component in the input variable.

### 8.10.3 Scaling

A 4 character string specifying the scaling to use for the input data to the fit. The first character in the string is associated with the X variable, the second the Y, the third the Z and the fourth the V. If the variable is not included in the fit then its scaling setting is ignored. Variables which have their scaling set to **L** (linear) are used in the fit as input. If its set to any other character then the scaling is assumed to be logarithmic and the  $\log_{10}$  of the variable is used in the fit. In the latter case variables which have values which are  $\leq 0$  are not included in the fit.

### 8.10.4 Coefficients

For 1D fits, either least-squares or non-linear, this is the number of coefficients used in describing the model. For 1D fits to a polynomial this is 1 larger than the order of the polynomial to be fit to. For 2D and 3D fits this is the order of the polynomial being fit to (the function will determine the actual number of coefficients).

### 8.10.5 PreSets

When performing a 1D Non-Linear fit the user must supply a set of initial guesses for each of the model coefficients. These are set in the variable given here. Use the **SETV** function to set up the values.

### 8.10.6 Function

The name of the external Tcl procedure describing the model being fit to. This field is only used in 1D linear and non-linear fits. It is not used for either 2D or 3D least squares fits. The UDFAnalysis framework contains several built-in procedures to describe various models. These are shown in the table below.



Procedure	Notes	Expression
TUpolyFunc	1D Linear	$V = A_0 + A_1X + A_2X^2 + \dots + A_NX^N$
TUlgp0Func	1D Linear	$V = A_0P_0^0 + A_1P_1^0 + \dots + A_NP_N^0$
TUlgpFunc	1D Linear $-1.0 \leq X \leq 1.0$	$V = \sum_{m=0}^{m=N} \sum_{l=m}^{l=N} A_i P_l^m(X)$
TUsphFunc	2D Linear $Y = \cos\theta$ $X = \phi$ (deg)	$V = \sum_{l=0}^{l=N} A_i P_l^0(Y) + \sum_{m=1}^{m=N} \sum_{l=m}^{l=N} \{A_{N+j} P_l^m(Y) \sin(mX) + A_{N+j+1} P_l^m(Y) \cos(X)\}$
APgaussFunc	1D Non-Linear	$V = A_0 e^{-\left(\frac{X-A_1}{A_2}\right)^2}$
APtgaussFunc	1D Non-Linear	$V = A_0 e^{-\left(\frac{X-A_1}{A_2}\right)^2} + A_3 e^{-\left(\frac{X-A_4}{A_5}\right)^2}$
APlognormalFunc	1D Non-Linear	$V = \frac{A_0}{X} e^{-\left(\frac{\ln(X)-A_1}{A_2}\right)^2}$
APkappaFunc	1D Non-Linear	$V = A_0 \left[1 + \left(\frac{X}{A_2}\right)\right]^{-A_1}$

### 8.10.7 Output

The output variable which may be of order 1 or 2. The first variable holds both the returned fit coefficients as well as other information returned by the fitting routine. The second variable when present will contain the expanded model fit.

The returned data from the fitting routine has its information stored in the indices indicated in the table below.

Index	Contents
aType	Identifies the variable type. It is set to <b>FIT</b> .
fDEF	Identifies the function definition used in the.
fDim	The dimension of the fit
fFunc	The external function used in 1D Least-Squares and Non-Linear fits
LSq	YES if a Linear-Squares fit was performed
xSca	X variable scaling (Linear or Log)
ySca	Y variable scaling (Linear or Log)
zSca	Z variable scaling (Linear or Log)
vSca	V variable scaling (Linear or Log)
nC	Number of coefficients returned
gFit	A goodness of fit value. There is no value returned with non-linear fits. With a 1D least squares fit this is $\chi^2$ and with 2D and 3D least squares fits this the standard deviation of the data from the model.
CV#	The covariance values for each of the coefficients in the model. These are returned only for the 1D least squares and non-linear fits. # runs from <b>0</b> to <b>nC - 1</b>
#	The model coefficients. # runs from <b>0</b> to <b>nC - 1</b>

### 8.11 GridFill Function

The **Gridfill** function either creates and fills a new grid from the input variables or adds them into already existing grid. Empty cells in a grid can be filled using a linear interpolation scheme.

Figures 36 and 37 show examples of an unpopulated and populated setup menu for the grid fill function.

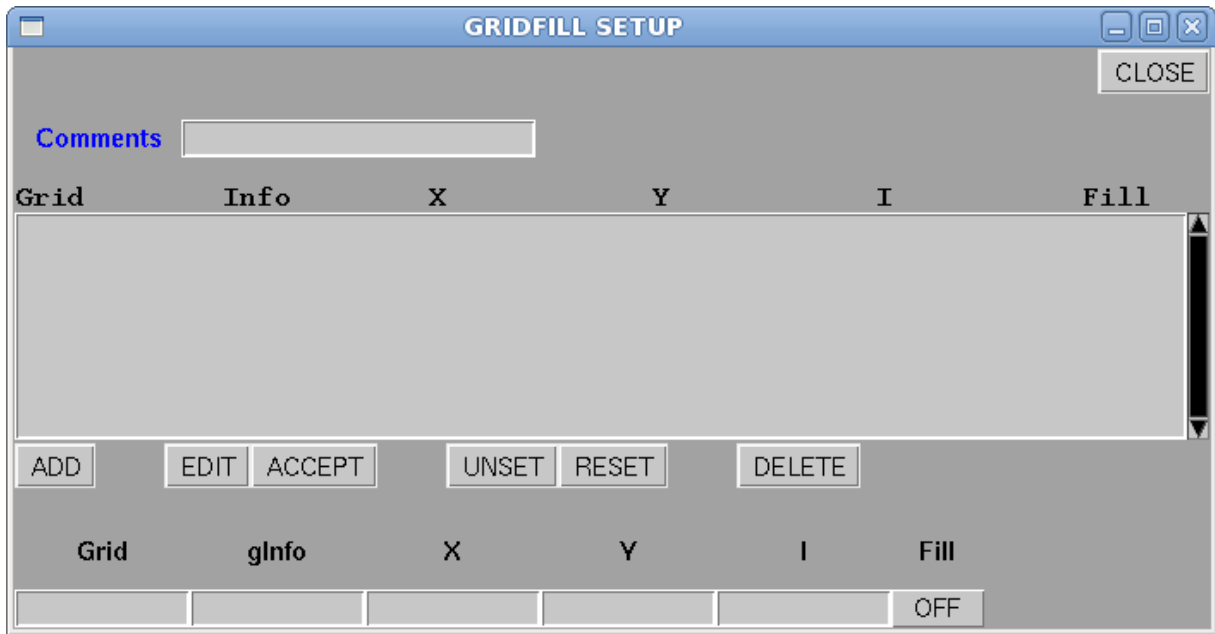


Figure 36: Unpopulated GridFill Setup Menu

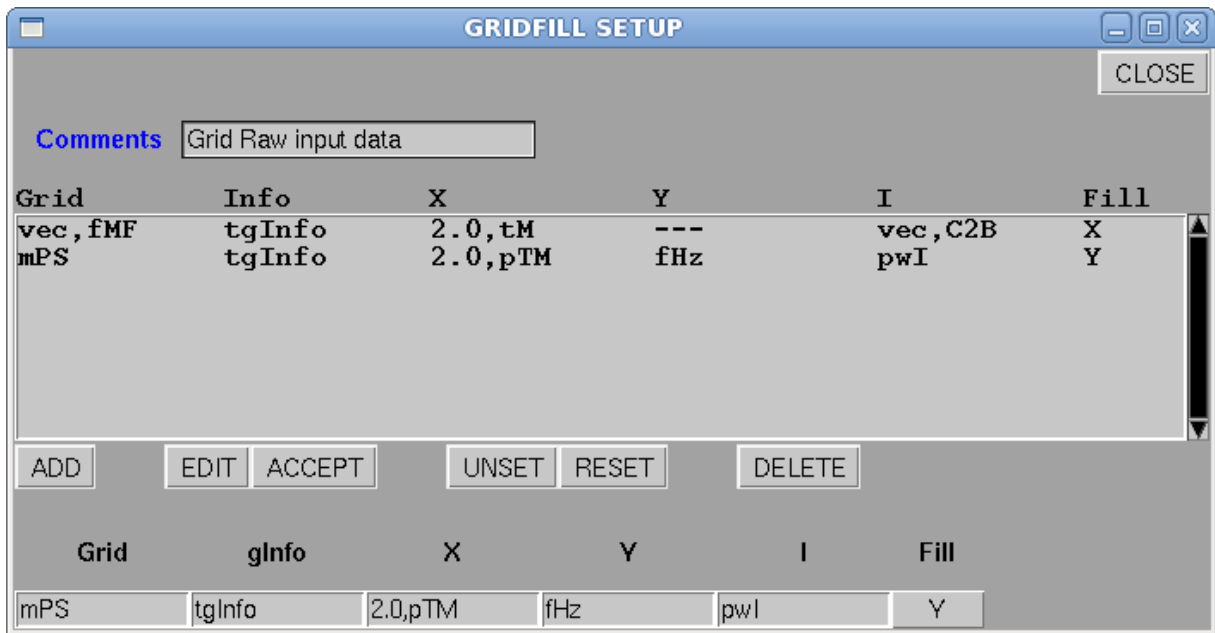


Figure 37: Populated GridFill Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.11.1 Grid

The variable representing the grid(s) to be created or added to. The routine accepts variables of arbitrary order. If this entry is left blank it reverts to the last defined instance. This allows multiple variables to be placed within a single grid definition.

### 8.11.2 gInfo

If a grid is to be created from the input **X**, **Y**, and **I** variables then this is the variable holding the grid information structure which was presumably created through the **DefGrid** function. Leave this field blank if you want just to fill an existing grid. In this case the **Grid** variable should already exist and will be appended to according to the grid information under which it was created.

### 8.11.3 X

The X variable to use when creating the grid. This should be a variable of order 2 if the measurements have width in the X direction (i.e, a start and stop value). If the variable has order 1 then the stop values are set to the start values which makes this a point measurement. When constructing multiple Grids, the same set of X values is used in each construction.

### 8.11.4 Y

The Y variable to use when creating the grid. This should be a variable of order 2 if the measurements have width in the Y direction (i.e, a start and stop value). If the variable has order 1 then the stop values are set to the start values which makes this a point measurement. When constructing multiple Grids, the same set of Y values is used in each construction.

### 8.11.5 I

The Intensity variable(s) to use when creating the grid. There should be one defined intensity variable for each grid being constructed.

### 8.11.6 Fill

The fill option. This is **OFF** if no fill is to be performed, **X** to fill just along the X direction, **Y** to fill just along the Y direction, **XY** to fill first along the X direction and then along the Y direction and **YX** to fill first along the Y direction and then along the X direction.

## 8.12 Index Function

The **Index** function constructs an indexed array for a scalar variable. The index array has the same length as the parent variable and runs sequentially from 0.5 to  $N - 0.5$  where N is the array length.

Figures 38 and 39 show an unpopulated and populated setup menu for the index function respectively

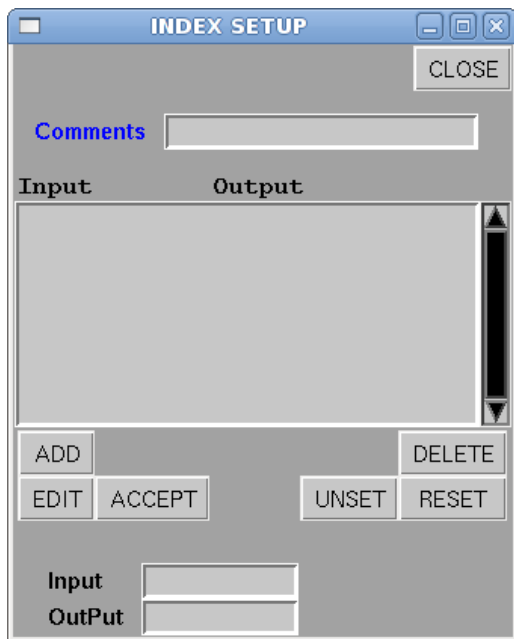


Figure 38: Unpopulated Index Setup Menu

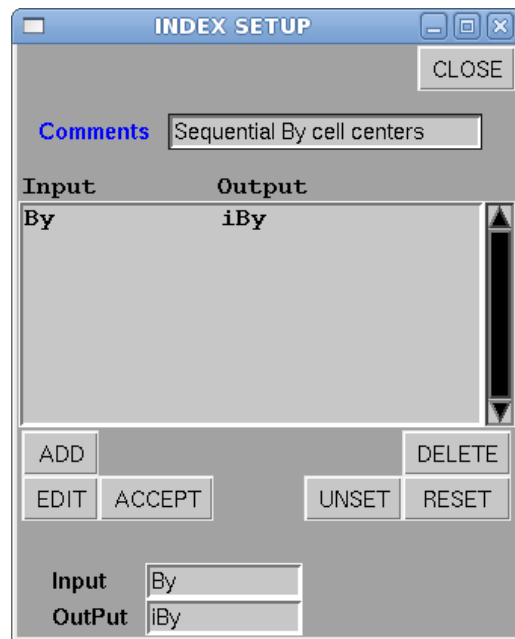


Figure 39: Populated Index Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.12.1 Input

The scalar variable for which the index array is generated.

### 8.12.2 Output

The index array.

## 8.13 Mask Function

The **Mask** function constructs a data mask from a scalar variable which can then be applied to other variables to mask out specific elements in their array. A mask is a binary grid (0 or 1) of the same length as the variable from which it is constructed. Elements which are 0 in the mask correspond to the cells in the variable from which the mask was built which meet a defined condition. When the mask is applied to other variables, all cells in the variable for which the mask cell is 0 are set to empty (unfilled). The masked variable can be returned either in the input variable or in a new variable (preserving the input variable).

The mask function allows for continued function definitions. To continue a definition leave the source option blank. The function appends the option definitions on this line to those of the last function definition in which a source was specified. The current Condition

and Value options in the continued definition line are ignored. This improves execution speed for function definitions which would use the same mask.

Figures 40 and 41 shows an unpopulated and populated setup menu for the mask function.

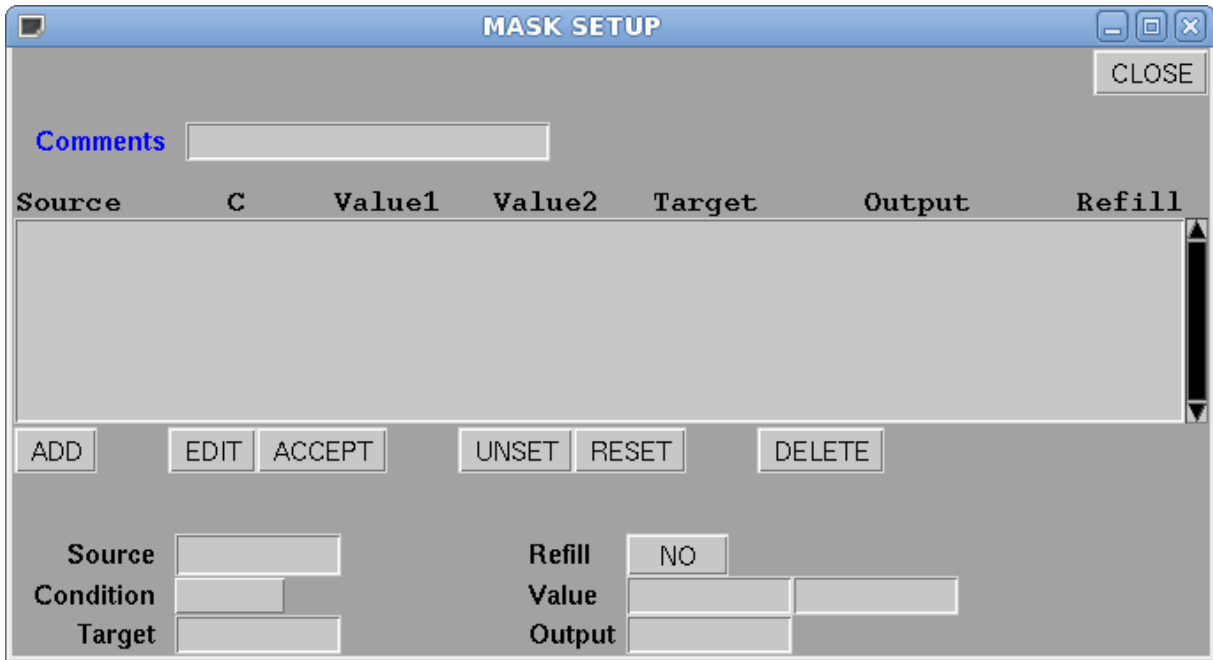


Figure 40: Unpopulated Mask Setup Menu

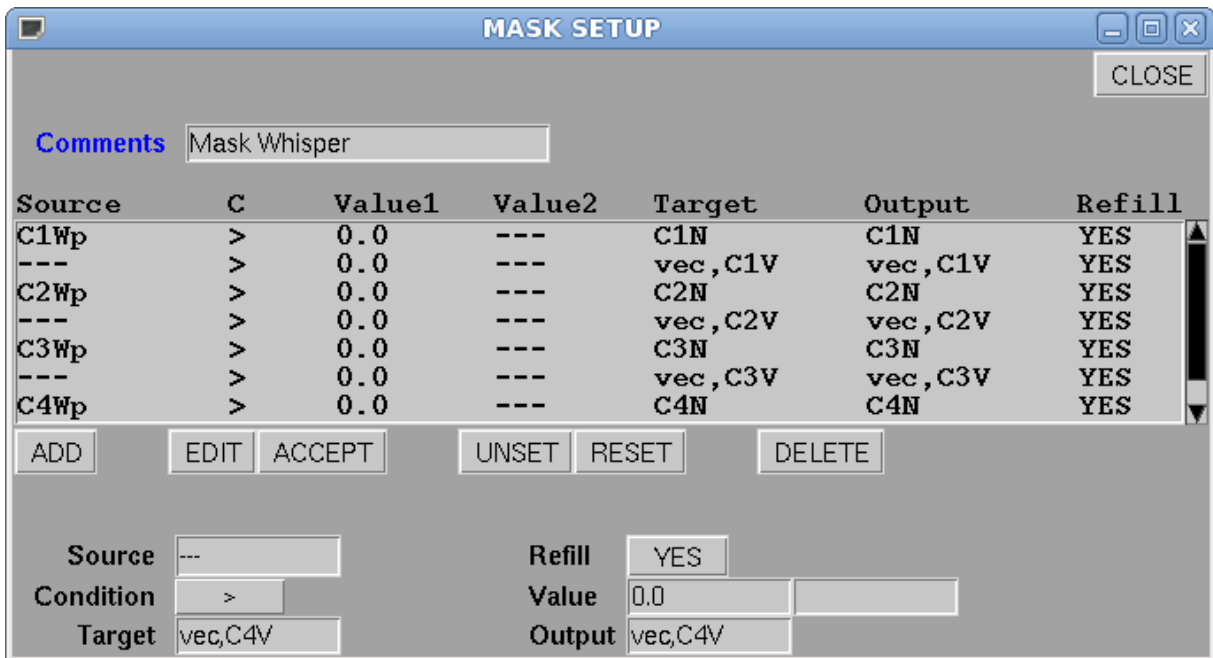


Figure 41: Populated Mask Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.13.1 Source

The name of the scalar variable from which the mask is created. Leave this option blank if you want to use the mask generated from last execution of the function.

### 8.13.2 Condition

The conditional statement to use when constructing the mask. This is ignored if no source is defined. The conditions available and their usage are shown in the following table where D is the data being tested and V1 and V2 are the conditional values defined in the next entry. The mask entry is set to 0 if the condition is met.

Condition	Definition
>	D > V1
>=	D >= V1
==	D == V1
!=	D != V1
<=	D <= V1
><	(D1 > V1) && (D1 < V2)
>=<=	(D1 >= V1) && (D1 <= V2)
>=<	(D1 >= V1) && (D1 < V2)
><=	(D1 > V1) && (D1 <= V2)

### 8.13.3 Value

The constants used in the condition tests according to the table above. All conditional tests make use of the first constant while only the last four conditions listed in the table make use of the second constant value. This option is ignored if no source is defined.

### 8.13.4 Target

The variables to apply the mask to. The routine accepts variables of arbitrary order. The variable components are individually processed.

### 8.13.5 Output

The variable holding the masking results. The output variable must be of the same order as the target variable. The results of the masking may be stored back into the target variable which would overwrite its current contents.

### 8.13.6 Refill

Set this option to **YES** if you want any masked out cells in the masked variable to be filled using a linear interpolation. Leave at **NO** otherwise.

## 8.14 Math Function

The math function provides constructs of the form

$$A \text{ op } B = C$$

where **A** and **B** are variables defined within the UDFAnalysis framework, **op** is an mathematical operation to perform and **C** is the variable in which the result is returned. You can substitute a constant for the **B** variable, but not for **A**. The function is designed to work with variables of arbitrary order. How the expression is evaluated depends on the operation.

Figures 42 and 43 shows an unpopulated and populated setup menu for the mask function.

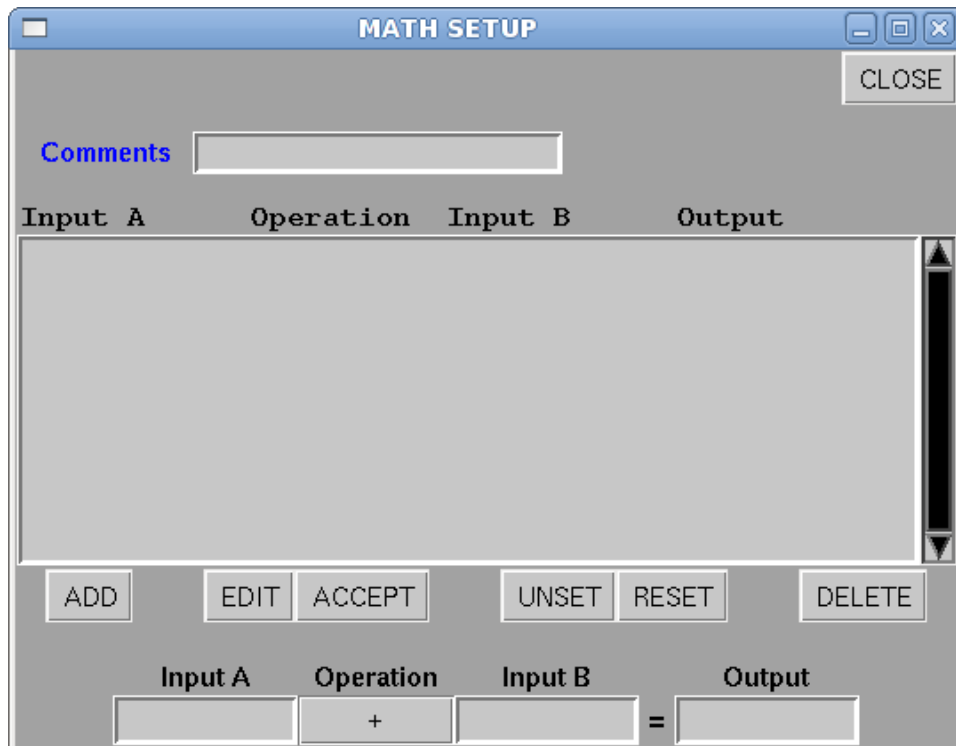


Figure 42: Unpopulated Math Setup Menu



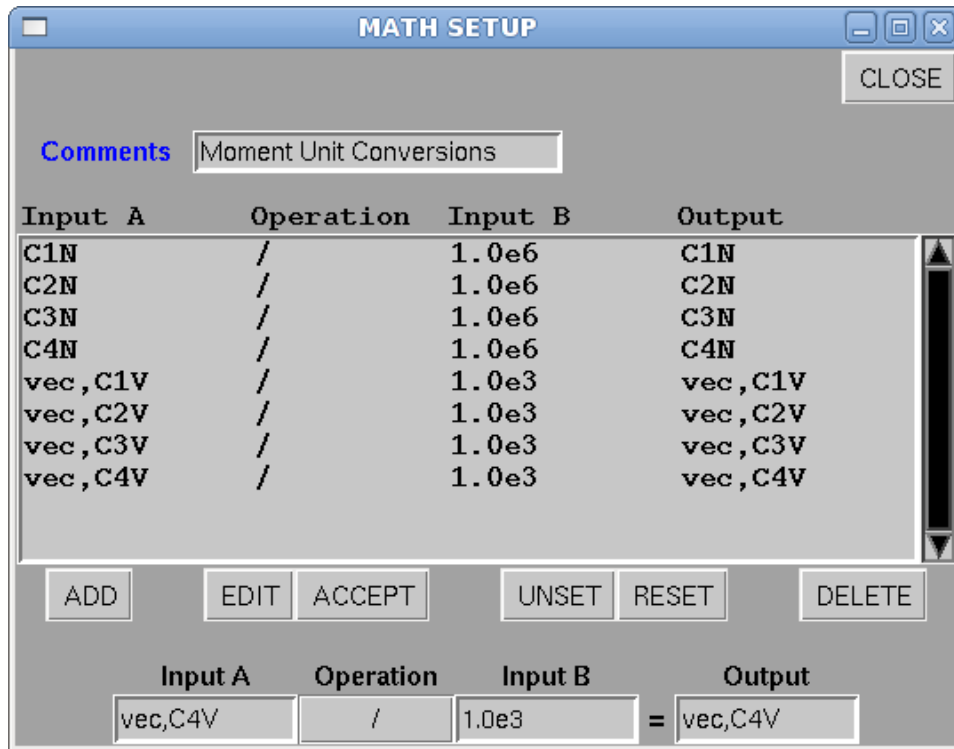


Figure 43: Populated Math Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

#### 8.14.1 Input A

The **A** variable in the math expression. This may be a linked variable. It cannot be a constant value.

#### 8.14.2 Operation

The math operation. These are summarized below. It should be understood that in the definitions **A** refers to Input A, **B** refers Input B and **C** refers to the Output variable. If Input B is not indicated as being used in the definition then it is not required in the expression and should be left blank.

Operator	Definition
+, -, *, /	The basic addition, subtraction, multiplication and division math operators. They require both <b>A</b> and <b>B</b> . If <b>A</b> and <b>B</b> are linked variables they must be of the same order. The expression is evaluated once for each pair of components in the <b>A</b> and <b>B</b> variables. If <b>B</b> is a scalar or constant then then expression is evaluated for each <b>A</b> variable using the same <b>B</b> variable in the evaluation. <b>C</b> must be the same order as the <b>A</b> .
ABS	The absolute value executed as $C = \text{abs}(A)$ . The expression executes once for each component. <b>C</b> must be of the same order as <b>A</b> .
ATAN	The arc tangent executed as $C = \text{atan2}(A,B)$ . with <b>C</b> returned in <b>radians</b> . <b>C</b> must be the same order as <b>A</b> . <b>B</b> must be of the same order as <b>A</b> or a scalar. The expression executes once for each component.
ATAND	The arc tangent executed as $C = \text{atan2}(A,B)$ with <b>C</b> is returned in <b>degrees</b> . <b>C</b> must be the same order as <b>A</b> . <b>B</b> must be of the same order as <b>A</b> or a scalar. The expression executes once for each component.
AVG	The average operator executed as $C = \frac{1}{N_A+N_B} (\sum A_i + \sum B_j)$ . The sum is over all <i>i</i> components of <b>A</b> and <i>j</i> components of <b>B</b> . N is the variable order. Input B is not required but is used if defined. <b>C</b> is a scalar variable.
DETREND	The detrend operator executed as $C = A - \langle A \rangle$ where $\langle A \rangle$ is the average over all elements in the <b>A</b> variable grid. The expression executes once for each component in <b>A</b> . <b>C</b> must be the same order as the <b>A</b> .
EXP	Base e exponentiation executed as $C = \text{exp}(A)$ . The expression executes once for each component. <b>C</b> must be the same order as <b>A</b> .
FMOD	The floating point modulus executed as $C = \text{fmod}(A,B)$ . <b>C</b> must be of the same order as <b>A</b> . <b>B</b> must be of the same order as <b>A</b> or a scalar. The expression executes once for each component.
LOG10	Base 10 logarithm executed as $C = \text{log10}(A)$ . Grid values in <b>C</b> are to undefined if the corresponding grid value in <b>A</b> is $\leq 0.0$ . The expression executes once for each component. <b>C</b> must be the same order as <b>A</b> .
LOGE	Base e logarithm executed as $C = \text{ln}(A)$ . Grid values in <b>C</b> are set to undefined if the corresponding grid value in <b>A</b> is $\leq 0.0$ . The expression executes once for each component. <b>C</b> must be the same order as <b>A</b> .
NORM	Normalization operator executed as $C = A / \text{Max}( A )$ where $\text{Max}( A )$ is the maximum absolute value in the <b>A</b> grid. The values in <b>C</b> will all lie between -1.0 and 1.0. <b>C</b> must be the same order as <b>A</b> .
POW	Exponentiation to an arbitrary base executed as $C = \text{pow}(A,B)$ where <b>A</b> is the base and <b>B</b> the exponent. <b>C</b> must be of the same order as <b>A</b> . <b>B</b> must be of the same order as <b>A</b> or a scalar. The expression executes once for each component.
SQRT	Square root executed as $C = \text{sqrt}(A)$ . Grid values in <b>C</b> are set to undefined if the corresponding grid value in <b>A</b> is $< 0.0$ . The expression executes once for each component. <b>C</b> must be the same order as <b>A</b> .
SUM	The summation operator executed as as $C = \sum A_i + \sum B_j$ . The sum is over all <i>i</i> components of <b>A</b> and <i>j</i> components of <b>B</b> . Input B is not

### 8.14.3 Input B

The **B** variable in the math expression. This may be a linked variable or a constant value.

### 8.14.4 Output

The variable which holds the result of the operation. The order of the output variable is determined by the operation used in the expression. See the table under the **Operation** option above.

## 8.15 MEM Function

The **MEM** function uses the **Maximum Entropy Method** to compute the power spectrum of a variable with evenly sampled data. Variables of arbitrary order are accepted with their components processed individually. The output of the MEM is a non-time based grid of length **Number of Steps**. It can only be used in Functions with variables of like length.

Figures 44 and 45 shows an unpopulated and populated setup menu for the MEM function.

MEM SETUP

Comments

Input	Power	Freq	Coef	Beg Hz	End Hz	Steps
-------	-------	------	------	--------	--------	-------

ADD EDIT ACCEPT UNSET RESET DELETE

Input  Coefficients   
Output Power  Begin Frequency (Hz)   
Output Frequency  End Frequency (Hz)   
Number of Steps

Figure 44: Unpopulated MEM Setup Menu

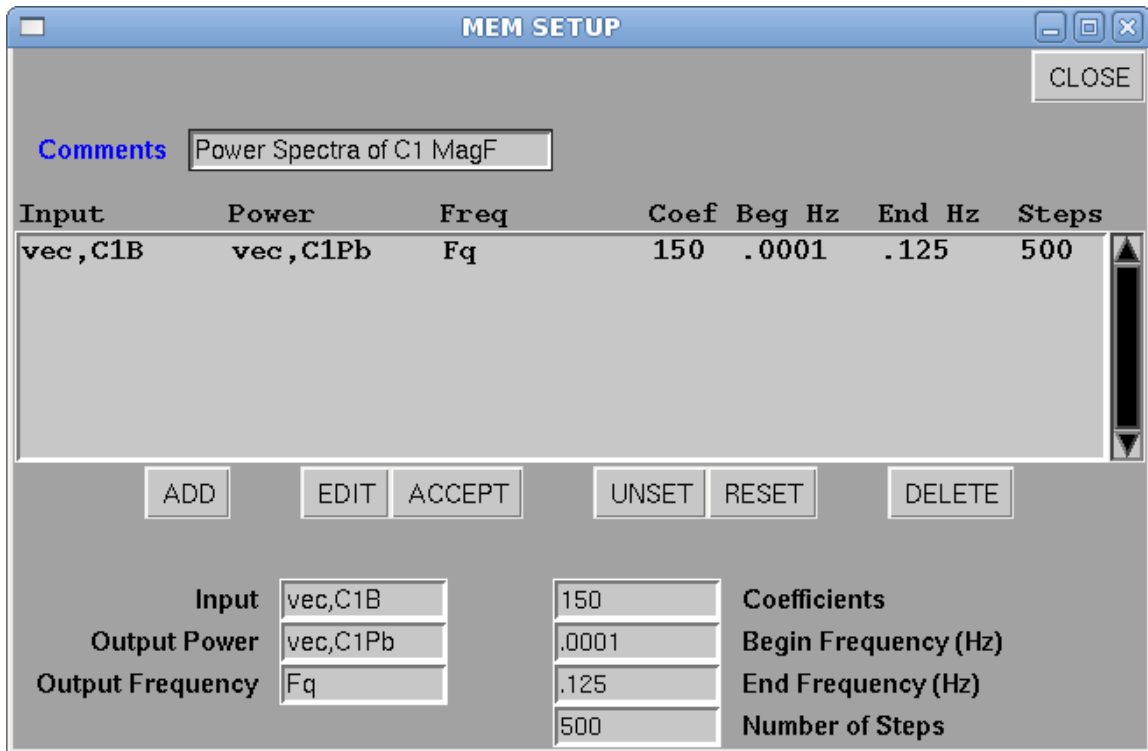


Figure 45: Populated MEM Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.15.1 Input

The variable that the MEM operates on. The routine accepts variables or arbitrary order. The components are individually processed to produce a power spectrum of each.

### 8.15.2 Output Power

The variable containing the output power spectral density. The variable must be of the same order as the **Input** variable.

### 8.15.3 Output Frequency

A scalar variable in which the frequency steps at which the power spectral densities are computed is returned. The function computes the power at the same frequency steps for each processed component in the input variable.

### 8.15.4 Coefficients

The number of linear prediction coefficients to use in computing the power spectral density. There is not a straightforward method to determine how many coefficients to use. Too small a value will omit or smooth high frequency peaks and sometimes will not split closely related

frequencies while too large a value tends to create noise in the spectra. A good value to start out with is something on the order of 3 to 6% of the total number of data points.

### 8.15.5 Begin Frequency

The beginning frequency in **Hz** of the power spectrum. This will be the first value in the **Output Frequency** array.

### 8.15.6 End Frequency

The ending frequency in **Hz** of the power spectrum. This will be the last value in the **Output Frequency** array.

### 8.15.7 Number of Steps

The number of frequencies at which to compute the power spectral densities. The spectral densities are computed starting at the beginning frequency and ending at the ending frequency in steps of

$$dF = \frac{eF - bF}{nF - 1}$$

where eF is **End Frequency**, bF is **Begin Frequency** and nF is **Number of Steps**.

## 8.16 Print Function

The **Print** function creates an ASCII dump file of selected variables. The file consists of a short header which lists the dumped variables followed by the data itself. The routine handles variables of arbitrary order. You can print both time and non-time based variables but they can't be mixed. All of the variables being output must have the same grid size, if not, the routine fails silently.

The print function allows for continued function definitions. To continue a definition leave the output file option blank. The function appends the option definitions on this line to those of the last function definition in which an output file was specified. This allows multiple variables to be included in the same file.

Figures 46 and 47 shows an unpopulated and populated setup menu for the Print function.

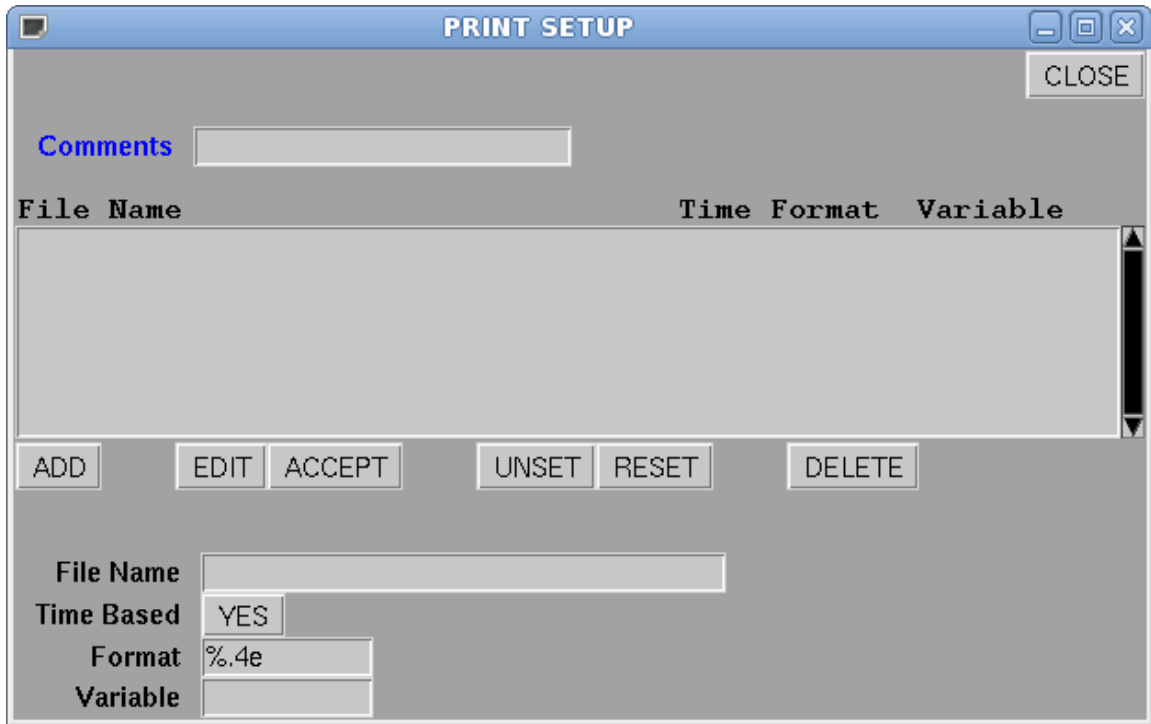


Figure 46: Unpopulated Print Setup Menu

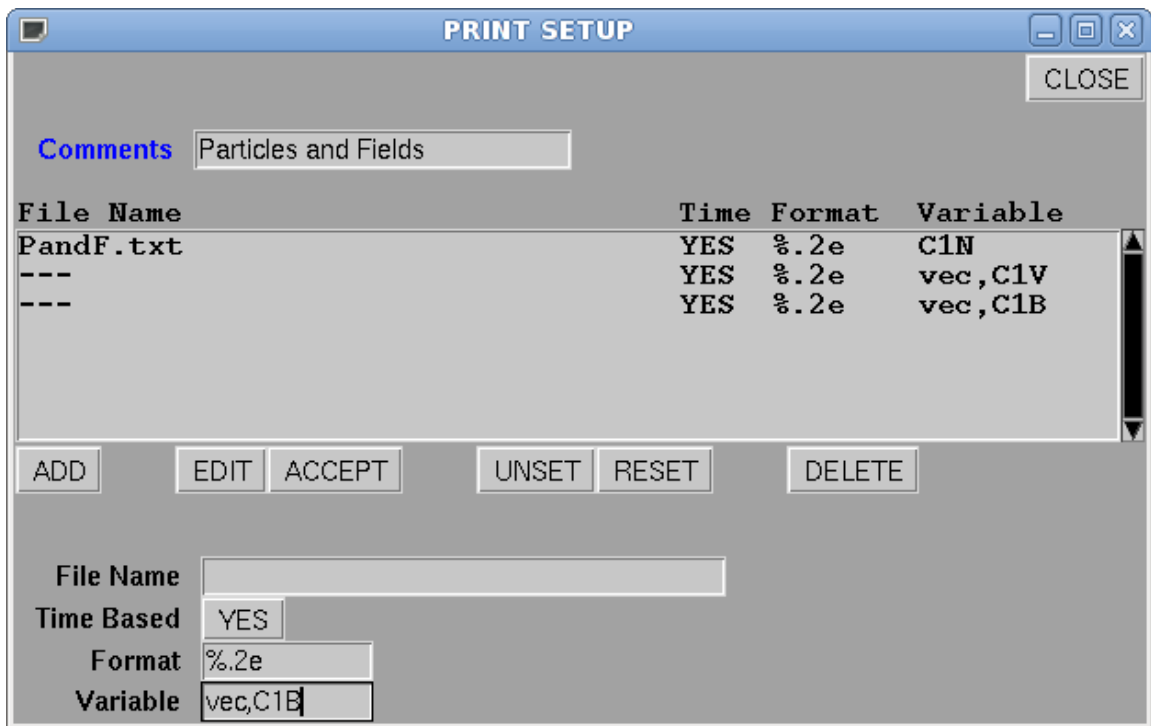


Figure 47: Populated Print Setup Menu

The work area options associated with this menu are described below. The other options

have already been described in the introduction to **Function Plugins**.

### 8.16.1 File Name

The name of the file ASCII output file. If you want the file in a directory other than the current working directory you need to include the full path name. If the file already exists it will be blatantly overwritten. If you leave this option blank the variables are output together with the variables in the last function definition which includes a defined output file.

### 8.16.2 Time Based

If the data being dumped is time based then select **YES** here. This will add the a start and stop time to each line of data in the file. You can say **NO** even if the data is time based in which case the start and stop times will be replaced a sequential counter beginning at 0. You should not mix time and non-time based data in the same dump file. The option is ignored if the **File Name** option is left blank.

### 8.16.3 Output Format

The C format you want the data to be output with. The same format is applied to all of the dumped variables in this definition. Different definitions can be output using different formats.

### 8.16.4 Variables

The variable to be dumped. This can be a variable of arbitrary order in which case all its components will be printed.

## 8.17 Random Function

The **Random** function constructs variable of random numbers. The length of the variable is set to be that of the input variable whose sole purpose is to provide the length. The random variables can be of an sign and will range up to a preselected magnitude. The seed to the random number generator is selectable.

Figures 48 and 49 shows an unpopulated and populated setup menu for the Random function.

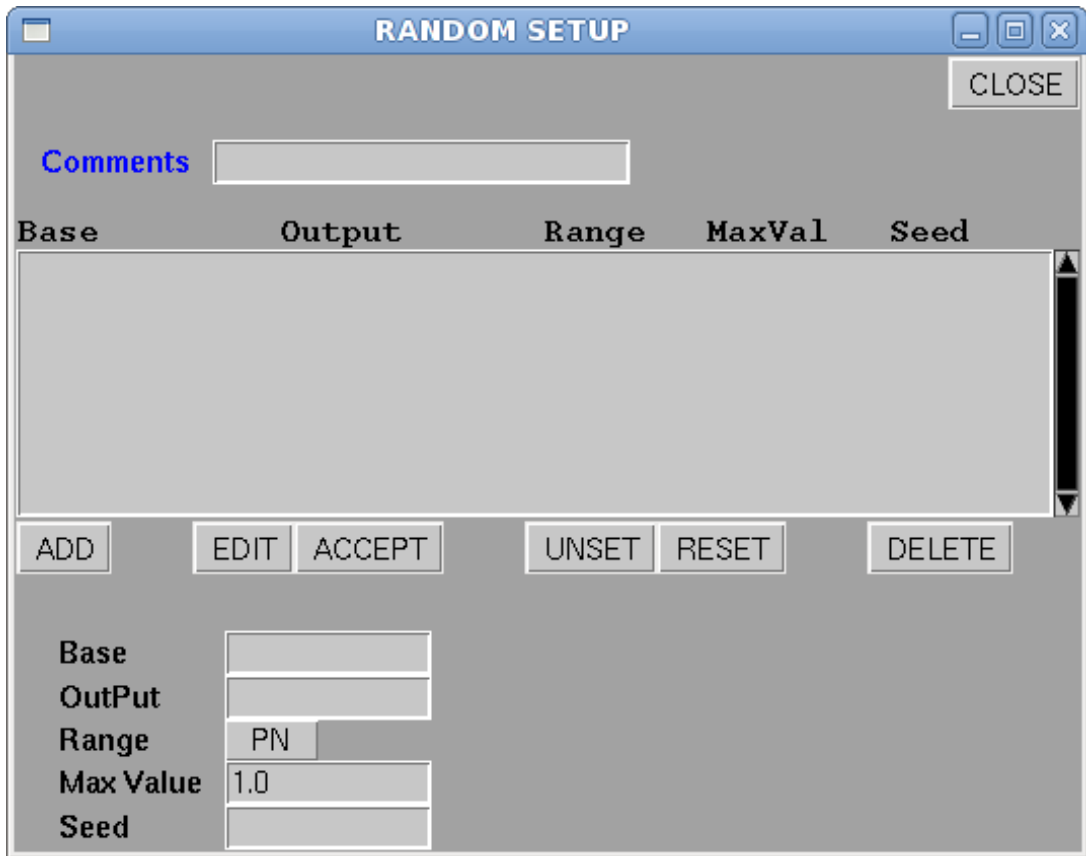


Figure 48: Unpopulated Index Setup Menu



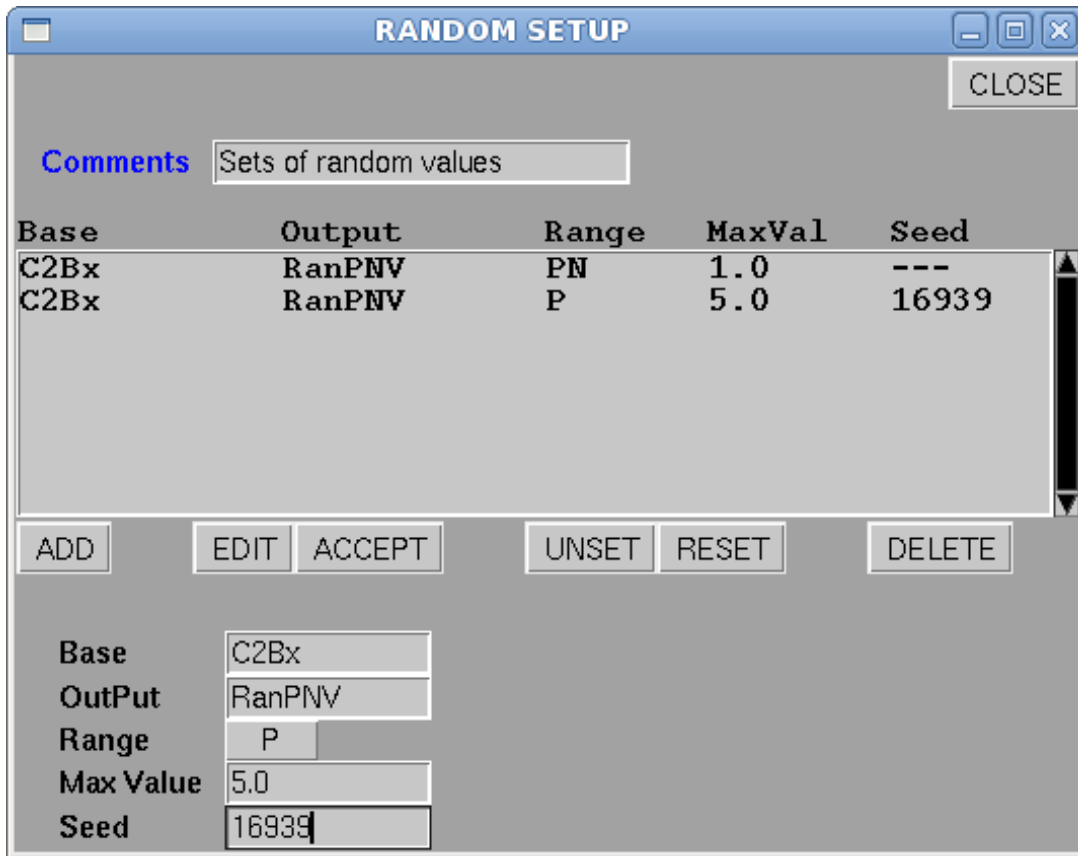


Figure 49: Populated Index Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

#### 8.17.1 Base

The scalar value used to provide the length of the random number variable.

#### 8.17.2 Output

The variable containing the random numbers.

#### 8.17.3 Range

The sign range over which to generate the random values. This can be **PN** for positive and negative values, **P** for positive only values and **N** for negative values only.

#### 8.17.4 Max Value

The maximum magnitude of the random values. If the range is set to **PN** then values will be generated between  $-(\text{Max Value})$  and  $(\text{Max Value})$ , if the range is set to **N** then values

will be generated between -(Max Value) and 0.0, and if the range is set to **P** then values will be generated between 0.0 and (Max Value).

### 8.17.5 Seed

The initial seed value to use when computing the random number. The value can be any number between 1 and 65536. If left blank, the current seed value will be used.

## 8.18 Spatial-Derv Function

The **Spatial Derivative** function computes the curl ( $\nabla \times$ ) and divergence ( $\nabla \cdot$ ) of a vector field or the gradient ( $\nabla$ ) and Laplacian ( $\nabla^2$ ) of a scalar field. A vector or scalar field is established by specifying the values of a vector or scalar value at multiple positions within a spatial volume and then fitting the data to a 3D polynomial model. The order of the polynomial is selectable, however, it must be low enough that the number of coefficients used in the fit is less than or equal to the number of defined spatial positions in the volume. The minimum number of positions for a first order fit is 4, for a second order fit 10, and for a third order fit 20. The positions used in the fit cannot be coplanar. The derivatives are computed at the arithmetic average position of all points making up the volume. The function can also return the vector or scalar value at the same position.

In most instances the data are fit to a first order 3D polynomial. This has the form:

$$V_i = A_{0i} + A_{1i}X + A_{2i}Y + A_{3i}Z$$

where i represents the x, y, or z component for vector variables and can be ignored in the case of scalar variables. The A's are the fit coefficients.

Computation of the divergence and curl as well as the gradient and Laplacian is straightforward. The divergence is given by:

$$\nabla \cdot \mathbf{V} = A_{1x} + A_{2y} + A_{3z}$$

the curl is given by:

$$\nabla \times \mathbf{V} = (A_{2z} - A_{3y})\hat{i} + (A_{3x} - A_{1z})\hat{j} + (A_{1y} - A_{2x})\hat{k}$$

the gradient is given by

$$\nabla V = A_1\hat{i} + A_2\hat{j} + A_3\hat{k}$$

and the Laplacian is 0. None of the derivatives have a spatial dependence in a first order fit. Fits to higher order polynomial models introduce a spatial dependence for all derivatives and allows for the possibility of a non-zero Laplacian.

At present this function only works for first and second order fits as it is lacking the code to compute the derivatives for higher order polynomials.

When working with space plasma data it is possible in certain situations to artificially **extend** the number of input positions. This can be done when **a)** the bulk plasma velocity is known and **b)** the Taylor frozen-in field theorem is valid. In its default mode the function computes one set of spatial derivatives at each grid point in the input data. Using the

frozen in theorem this can be expanded to include 1 or more neighboring cells. This is done by shifting by using the measured plasma velocity together with the time difference between the base and extended cells to adjust their positions to where the measured vector or scalar plasma moments would have existed at the time of the base measurement. This adds more measurement positions to the volume (as well as increases its size) which allows fits to higher order polynomial functions. It is one way, when there is insufficient data to fit to a second order polynomial, to allow such fits to be done and to obtain non-zero estimates of the Laplacian. If, for example, you know a measurement at 4 positions you need extend the volume to include the next two cells (8 extra points) to fit to a general second order polynomial.

Invocation of the frozen in theorem allows manipulation of the data in another manner. When the measurement points are or close to coplanar, an time delay can be added to one or more of the measurements to take them out of coplanarity. This does not increase the number of data points in the volume but changes their location.

The spatial derivative function allows for continued function definitions. To continue a definition leave the position variable blank. The function adds appends the option definitions on this line to those of the last function definition in which a position variable was specified. The Delay, Fit Order, and Extend options on this line will be ignored. This allows a number of measurements to be specified at the defined positions and also improves execution speed as the routine does not have to repeat computations which deal with the positions.

Figures 50 and 51 shows an unpopulated and populated setup menu for the Spatial Derivative function.

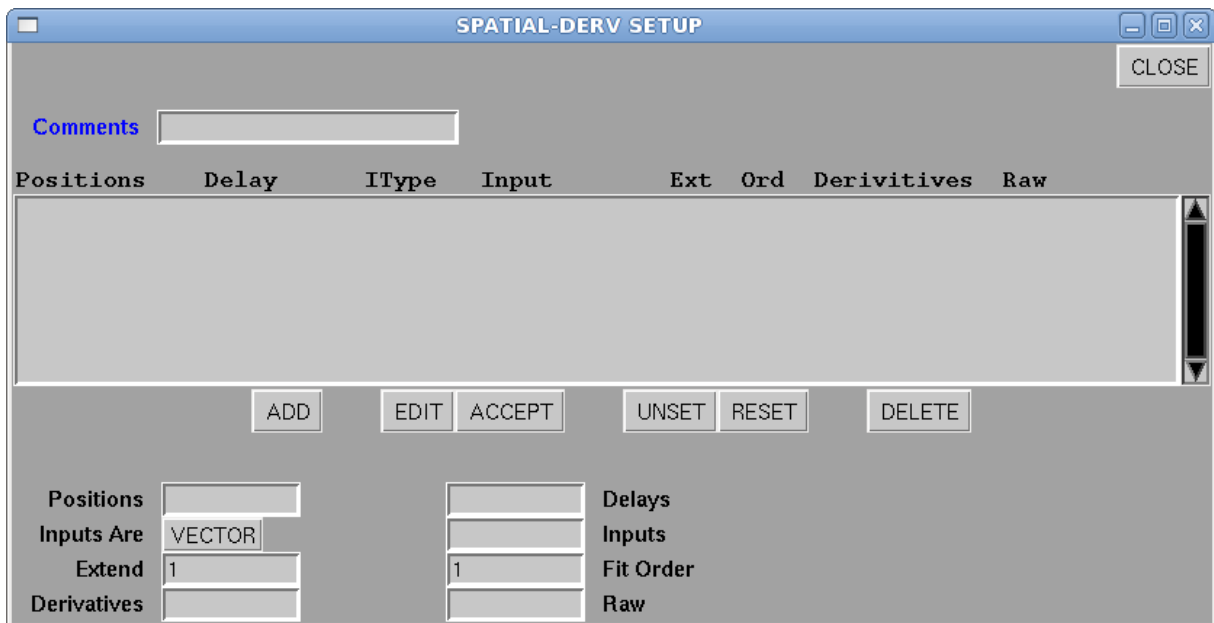


Figure 50: Unpopulated Spatial-Derv Setup Menu

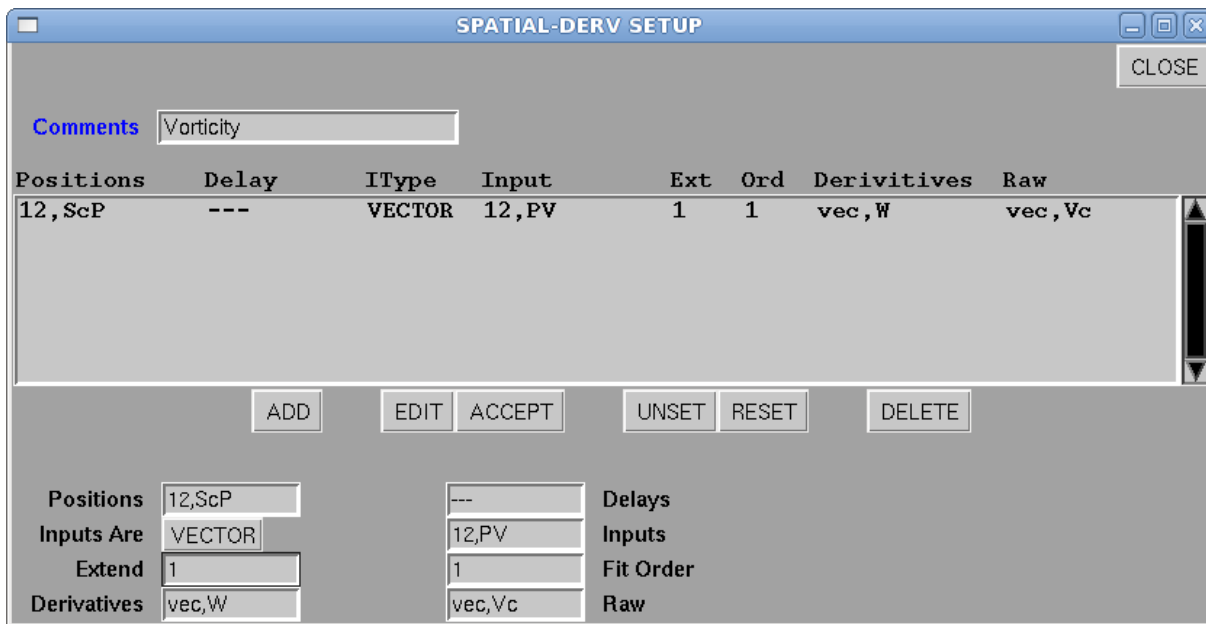


Figure 51: Populated Spatial-Derv Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.18.1 Positions

A linked variable containing the position vectors at which the input measurements are taken. Each position vector in the variable is given in the order X, Y, and Z respectively. There is a 1 to 1 correspondence between the position vectors and the input measurements. Leave the positions blank if the defined input information on the line to be grouped with the information from the last line for which a position variable was defined. The position variable is generally linked using the **VMAP** function.

### 8.18.2 Delay

An array of time offsets, one per position vector generally defined through the **SETV** function. These are used when working with plasma data to artificially move one or more of the measurement positions according to the Taylor frozen in field theorem. To disable this option leave it blank. It is also ignored if the **Positions** option has been left blank in which case the definition on line in which the positions are specified is used.

The computation of the new positions are made using the vector plasma bulk velocity which must be the first defined input variable in the function definition. Delays are given in units of seconds and are applied as

$$\vec{R} = \vec{R}_0 + \vec{V}T$$

where  $\vec{R}_0$  is the position at a delay of 0 seconds,  $\vec{V}$  is the bulk plasma velocity, T is the delay time, and  $\vec{R}$  is the the new position. **Delays** can be either positive or negative.

### 8.18.3 IType

The input variable type. This can be set to either **VECTOR** or **SCALAR**.

### 8.18.4 Input

A linked variable containing the set of vector or scalar measurements taken at each defined position. The type of measurements must match its **IType** setting.

### 8.18.5 Extend

The number of grid cells to use to form the volume. The default is 1. If you are fitting plasma data in which the Taylor frozen in condition is valid you can set this to higher values to increase the number of points defining the volume used in the computation. When doing this the first input variable in the function definition must be the set of plasma velocity vectors being used to reposition the added data points. The option is ignored if the **Positions** option has been left blank.

### 8.18.6 Fit Order

The order of the polynomial to fit to. It is defaulted to 1.

### 8.18.7 Derivatives

The results of the derivative computations. If the input variable is a vector this is the curl (a vector quantity) and divergence (a scalar quantity), otherwise it is the gradient (a vector quantity) and Laplacian ( a scalar quantity). The Laplacian is only returned if the fit order is greater than 1.

The number of components in the derivative variable can be used to limit what is returned. If the variable is a scalar only the scalar derivative will be returned, if it is a vector then only the vector derivative will be returned, and if its a fourth order variable both the vector and scalar derivatives are returned. In the latter case the three vector components are returned in the first three variable components and the scalar is returned in the fourth component.

### 8.18.8 Raw

The value of the scalar or vector input variable at the arithmetic average position of all points making up the volume. This is the location at which the derivatives are computed. The variable must be the same order as the input variable.

## 8.19 Setv Function

The **Setv** function allows constant values to be assigned to locations in a variable array. Up to 4 constants can be assigned at once beginning at a selectable array index and incrementing from there.

The setv function allows for continued function definitions. To continue a definition leave the variable option blank. The function appends the values on this line to the last defined variable continuing at the next index if no index position has been defined or begins at the defined index.

Figures 52 and 53 shows an unpopulated and populated setup menu for the Set Value function.

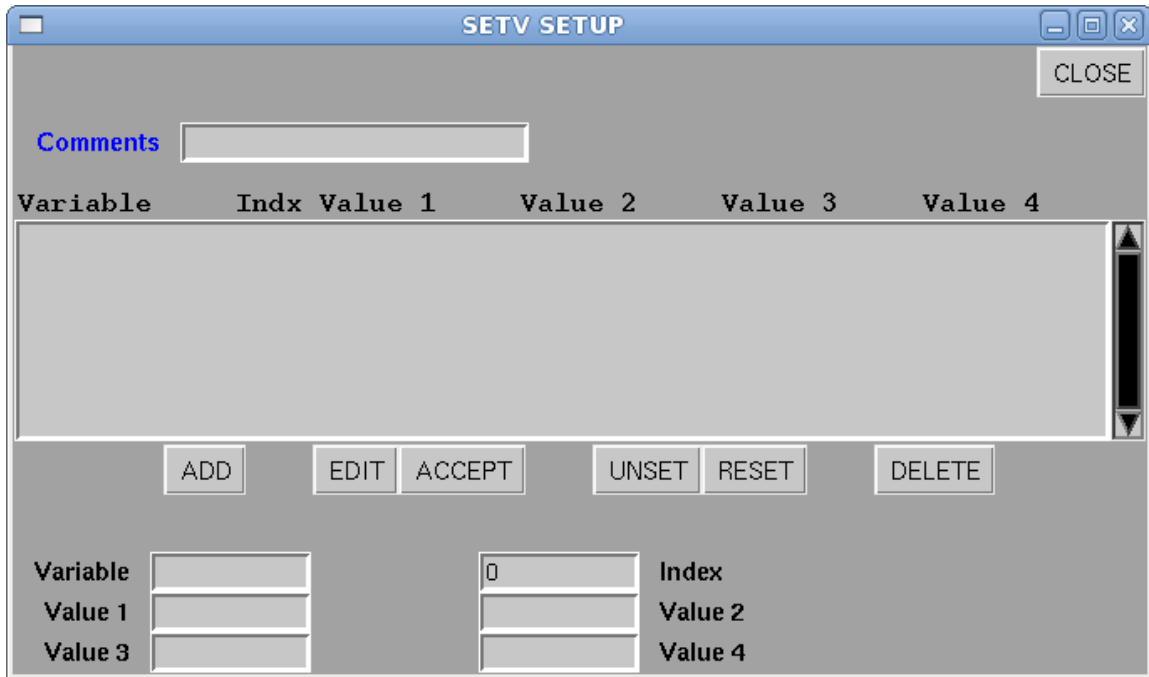


Figure 52: Unpopulated Setv Setup Menu

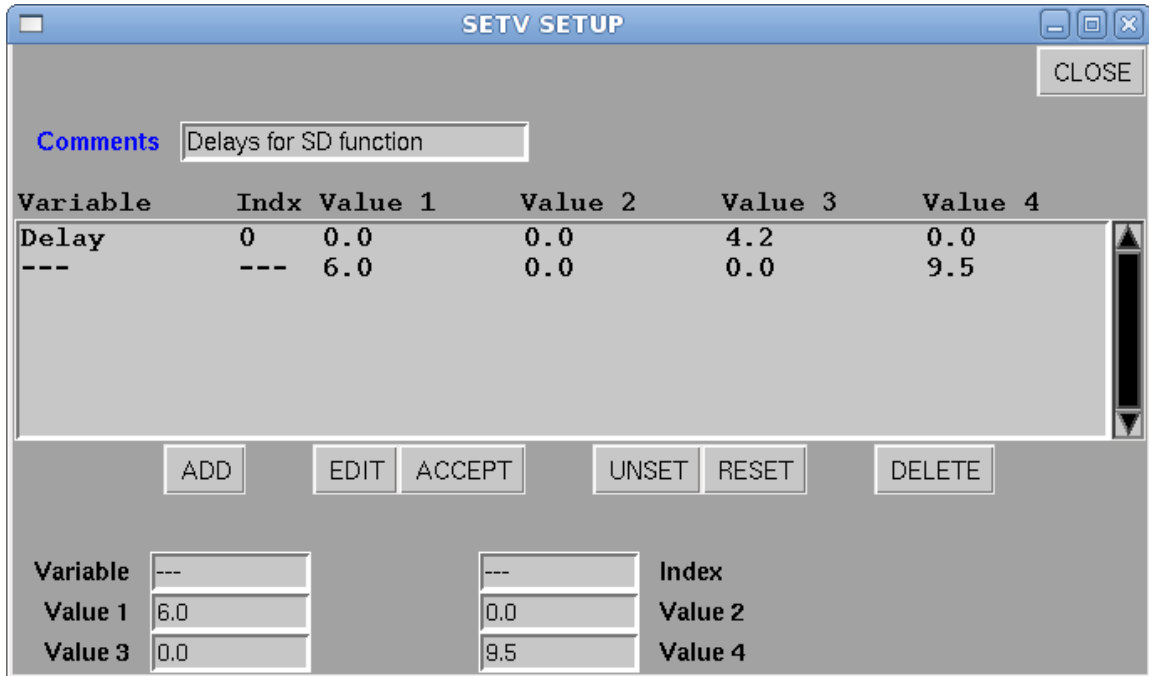


Figure 53: Populated Setv Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.19.1 Variable

The variable to assign the constants to. If this field is left blank the routine will use the variable assigned in the last function definition where field is defined.

### 8.19.2 Index

The array index at which the first defined value is to be stored. If both this and the **Variable** fields are blank then routine will continue filling the last defined variable beginning at the location at which it left off.

### 8.19.3 Value 1

The first constant value to be stored in the variable.

### 8.19.4 Value 2

The second constant value to be stored in the variable. Leave the field blank if there is no second value.

### 8.19.5 Value 3

The third constant value to be stored in the variable. Leave the field blank if there is no third value.

### 8.19.6 Value 4

The fourth constant value to be stored in the variable. Leave the field blank if there is no fourth value.

## 8.20 Statistics Function

The **Statistics** function computes the mean value, variance, standard deviation, average deviation, skewness and kurtosis of a variable. The information is returned in the output variable in the order listed above (array element 0 is the mean, element 1 the variance, etc). Needless to say this is a non-plottable variable.

The various quantities are defined below. In all the algorithms input X is the variable, N is the number of elements in X (number of cells in the data grid).

Quantity	Algorithm
Mean	$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x_n$
Variance	$V = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_n - \bar{x})^2$
Standard Deviation	$\sigma = \sqrt{V}$
Average Deviation	$\bar{\sigma} = \frac{1}{N} \sum_{n=0}^{N-1}  x_n - \bar{x} $
Skew	$S = \frac{1}{N} \sum_{n=0}^{N-1} \left( \frac{x_n - \bar{x}}{\sigma} \right)^3$
Kurtosis	$K = \left[ \frac{1}{N} \sum_{n=0}^{N-1} \left( \frac{x_n - \bar{x}}{\sigma} \right)^4 \right] - 3$

Figures 54 and 55 shows an unpopulated and populated setup menu for the Statistics function.



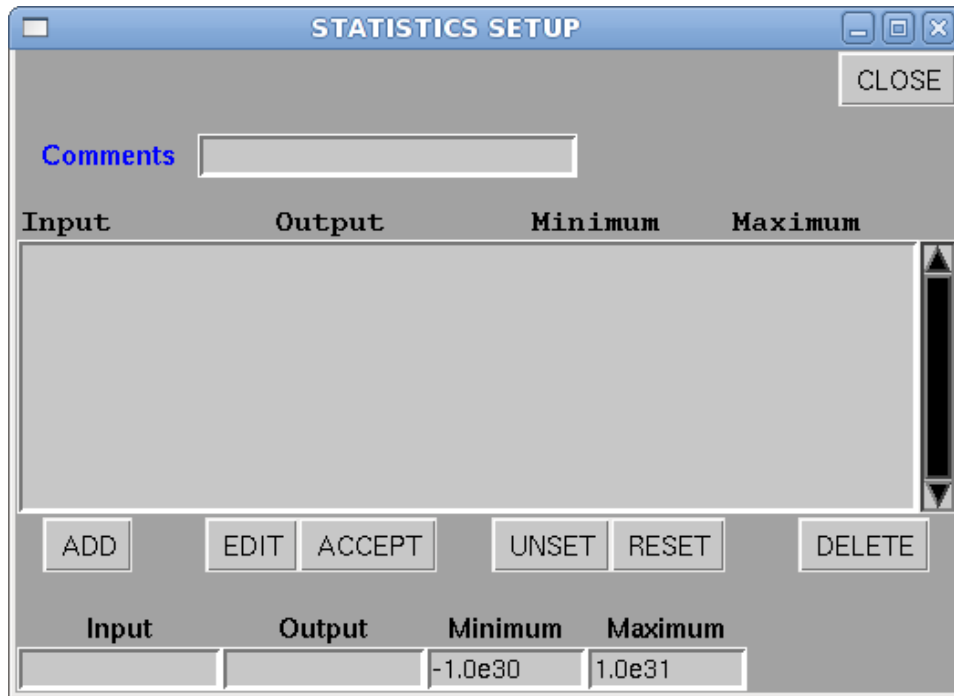


Figure 54: Unpopulated Statistics Setup Menu

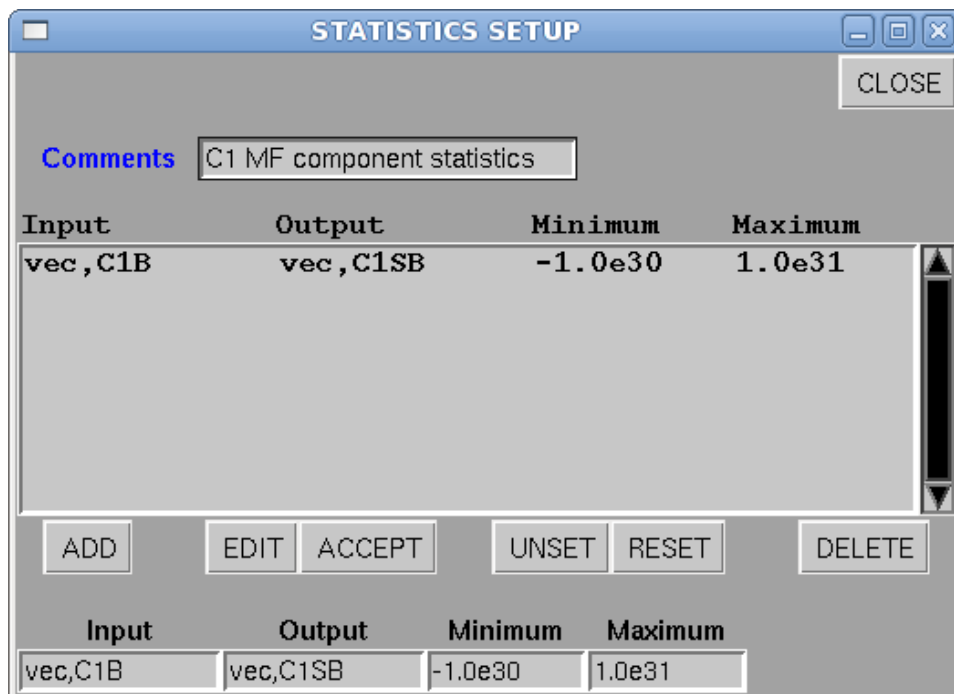


Figure 55: Populated Statistics Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.20.1 Input

Input variable to the statistics algorithms. This can be an arbitrary order variable. The statistics computations are performed independently on each component.

### 8.20.2 Output

The variable holding the results of the statistical computations. This must be of the same order as the input variable. Each component of the variable contains 6 elements. These are in order beginning at index 0 and ending at index 5, the mean value, the variance, the standard deviation, the average deviation, the skewness and the kurtosis.

### 8.20.3 Minimum

Data below this value are excluded in the statistics summations.

### 8.20.4 Maximum

Data above this value are excluded in the statistics summations.

## 8.21 Unset Function

The **Unset** function frees all memory associated with specified variables. This function is not meant to be used with variables defined through the **Variable Definition** menu but with variables defined in various function calls, especially temporary variables.

Figures 56 and 57 show an unpopulated and populated setup menu for the Unset function respectively



Figure 56: Unpopulated Unset Setup Menu



Figure 57: Populated Unset Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.21.1 Input

The variable to be freed.

## 8.22 Vector Function

The **Vector** function provides operations of the form

$$A \text{ op } B = C$$

where **A** and **B** are vector variables defined within the UDFAnalysis framework , **op** is a vector operation such as the dot or cross product, and **C** is the resultant scalar or vector.

Figures 58 and 5955 shows an unpopulated and populated setup menu for the Vector function.

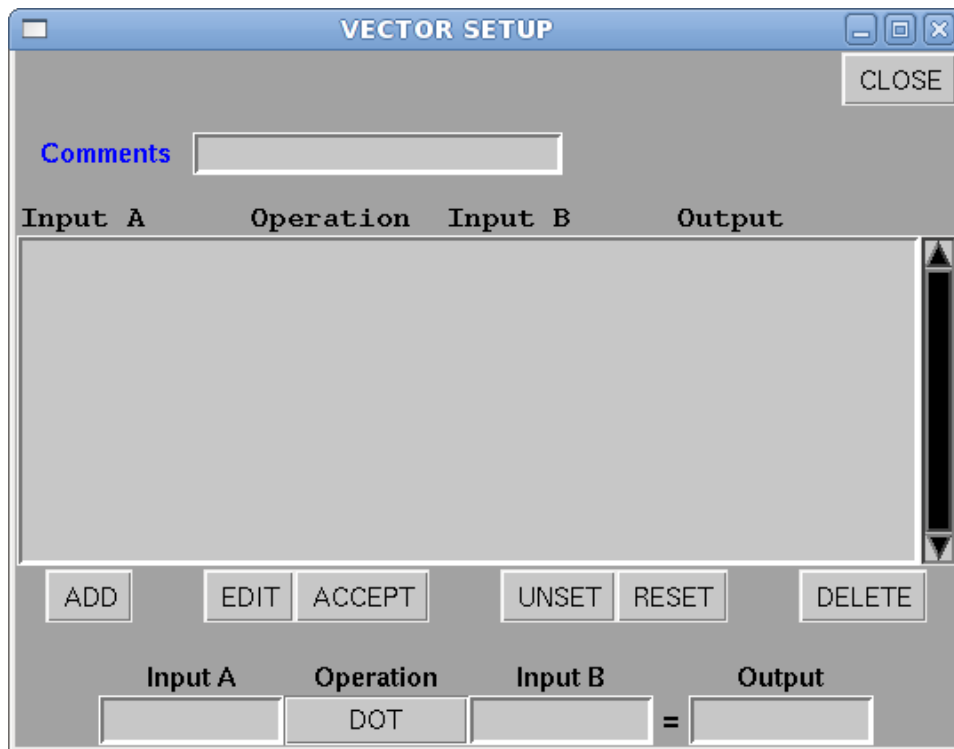


Figure 58: Unpopulated Vector Setup Menu

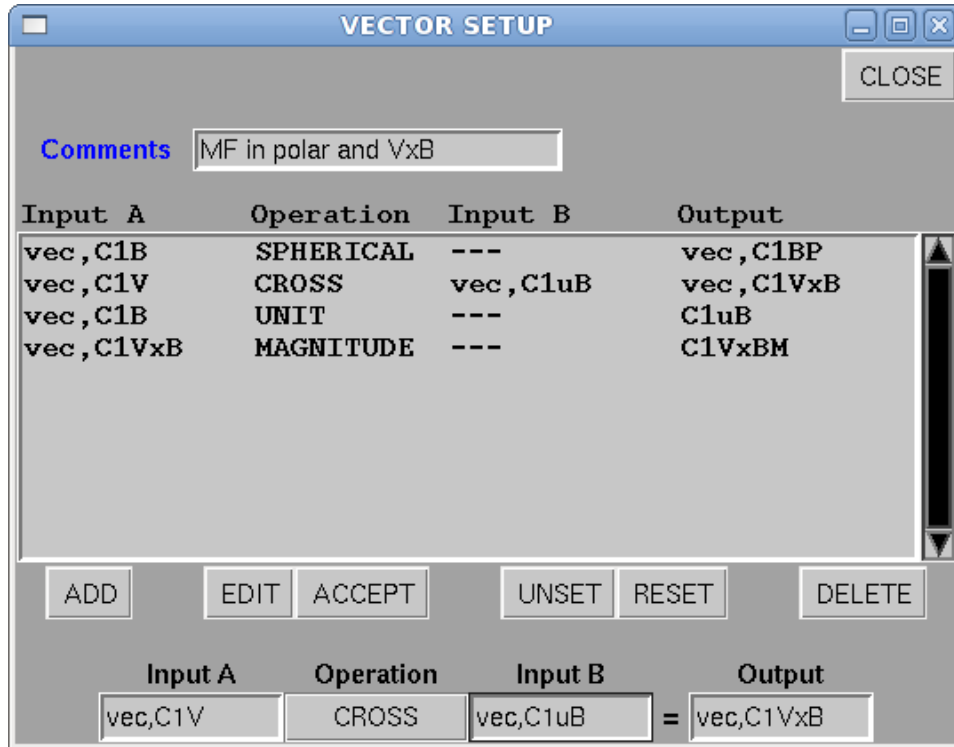


Figure 59: Populated Vector Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.22.1 Input A

The primary vector used in the algorithm.

### 8.22.2 Operation

The operation performed by the function. The available operations are described in the table below. It should be understood that in all formula **A** represents Input A, **B** represents Input B and **C** represents the output. If **B** is not shown in the formula then it is not used in the definition. The Output Type column indicates the order of the returned variable.

Operation	Description	Output Type
ANGLE	$C = \cos^{-1} \left( \frac{\mathbf{A} \cdot \mathbf{B}}{ \mathbf{A}   \mathbf{B} } \right)$ (degrees)	Scalar
CROSS	$C = \mathbf{A} \times \mathbf{B}$	Vector
DISTANCE	$C = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2 + (A_z - B_z)^2}$	Scalar
DOT	$C = \mathbf{A} \cdot \mathbf{B}$	Scalar
MAGNITUDE	$C =  \mathbf{A} $	Scalar
RECTOSPH	$(A_x, A_y, A_z) \rightarrow (C_r, C_\phi, C_\theta)$	Vector
SPHTOREC	$(C_r, C_\phi, C_\theta) \rightarrow (A_x, A_y, A_z)$	Vector
UNIT	$\frac{\mathbf{A}}{ \mathbf{A} }$	Vector

In the RECTOSPH operation the returned azimuth angle runs from  $-180^\circ$  to  $180^\circ$  and the returned polar angle from  $0^\circ$  to  $180^\circ$ . The angles are returned in degrees. The SPHTOREC operation expects the angle inputs to be in degrees and over the same ranges.

### 8.22.3 Input B

The secondary vector set used in the algorithm. It is not needed for all vector operations (see the above table).

### 8.22.4 Output

The variable holding the results of the vector operations. The output variable is either a scalar or a vector depending on the vector operation.

## 8.23 Vmap Function

The **Vmap** function maps one variable name into another. This is not a renaming of the variable but the creation of an alias. Both the original and new name can be used interchangeably. The main use of the function is in creating sets of linked variables. The function works with arbitrary order variables.

Figures 60 and 61 show an unpopulated and populated setup menu for the Variable Map function respectively

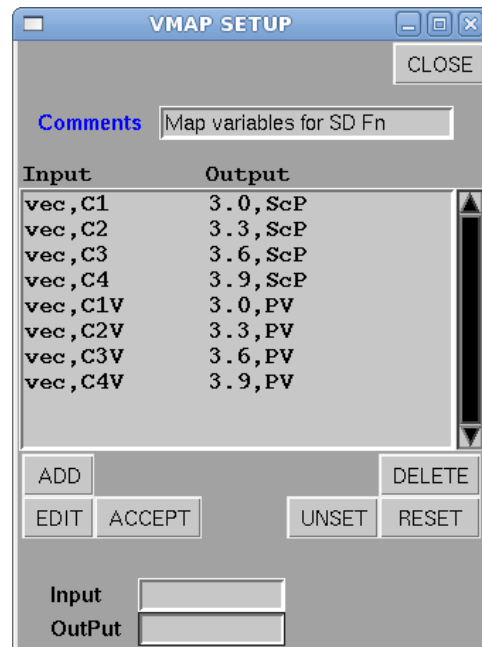
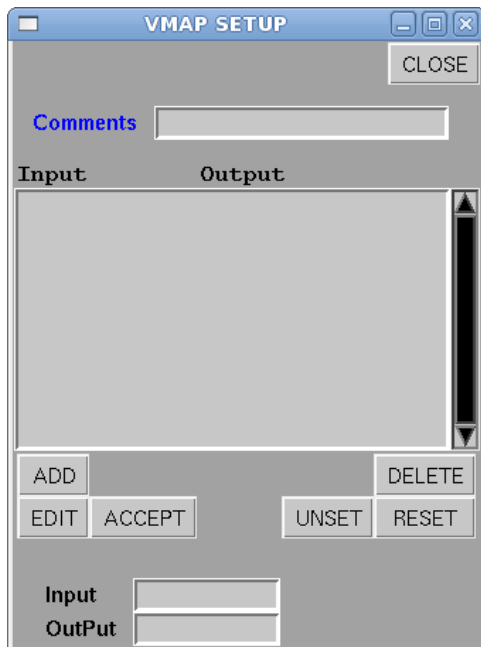


Figure 60: Unpopulated Vmap Setup Menu

Figure 61: Populated Vmap Setup Menu

The work area options associated with this menu are described below. The other options have already been described in the introduction to **Function Plugins**.

### 8.23.1 Input

The input variable name. This can itself be a linked set of variables.

### 8.23.2 Output

The output variable name. This is the alias being created for the input variable. It must be of the same order as the input variable.

## 9 Plot Setup Menu

Plots produced within the UDFAnalysis framework are output onto a single canvas called the parent plotting window. Most of the characteristics of this window, including its size, the size of any margins along the edge, as well as many generic plot specific properties are established in the **Plot Setup** menu. The menu is invoked from the **Main Menu**. Placement of the plots in the window is done in the **Plot Layout Menu** while the contents of each plot are defined in the **Plot Definitions** menu.

Clicking **Plot Setup** in the main menu brings up the menu shown in Figure 62.

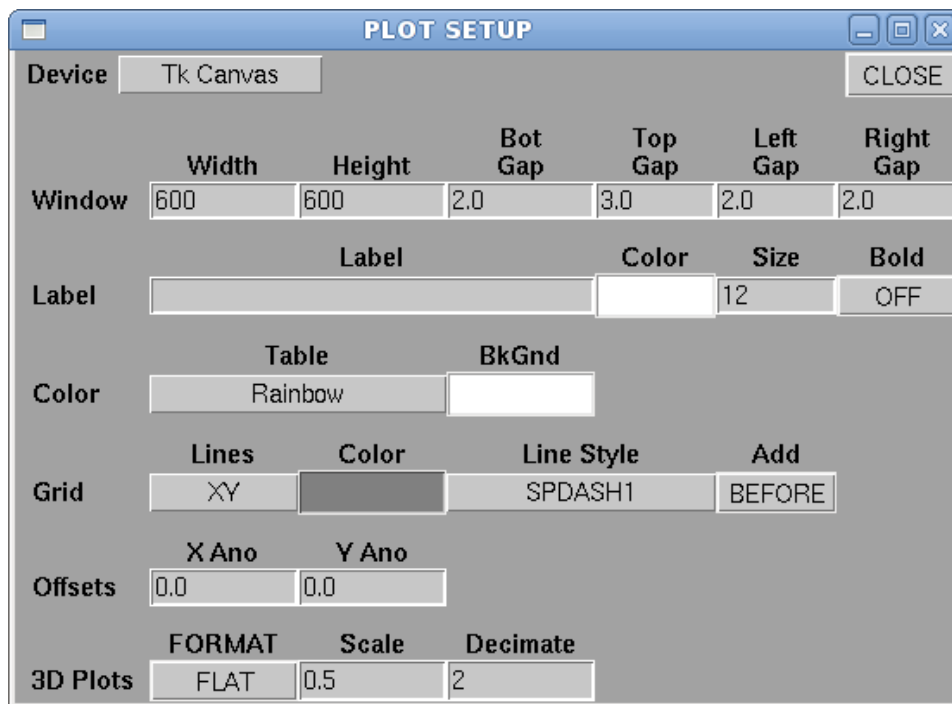


Figure 62: Unpopulated Plot Setup Menu

Default values are supplied for all options except for the main window label. The options are described in the sections below.

## 9.1 Device

The device driver to use to render the graphics. Unless you have VTK installed the only option currently supported is **Tk Canvas**. If you have VTK installed then you can also select **VTK Canvas**. The latter has extensions which allow for fully rotatable and zoomable 3-D plots and is required for the use of certain plot types. Unless there are 3D plots being output **Tk Canvas** is the better and more robust choice of the two.

## 9.2 Window

All plots are output in a single parent window. Arrangement of the plots within the window is described in the **Plot Layout** section. The following options describe some of the basic options which control the characteristics of the parent window.

### 9.2.1 Width

The width of the parent graphics window in pixels.

### 9.2.2 Height

The height of the parent graphics window in pixels.

### 9.2.3 Top Gap

The amount of white space to leave along the top of the parent plot window. The units are in terms of a 12 point character. The option is used to add extra space for text. If you include an overall plot label you should increase this from its default value of 3 to 6 or 7.

### 9.2.4 Bot Gap

The amount of white space to leave along the bottom of the parent plot window. The units are in terms of a 12 point character. This is used to add extra space for text other than the nominal plot axis numerical and text labeling.

### 9.2.5 Left Gap

The amount of white space to leave along the left hand side of the parent plot window. The units are in terms of a 12 point character. This is used to add extra space for text other than the nominal plot axis numerical and text labeling.

### 9.2.6 Right Gap

The amount of white space to leave along the right hand side of the parent plot window. The units are in terms of a 12 point character. This is used to add extra space for text other than the nominal plot axis numerical and text labeling.

## 9.3 Label

An optional overall label can be placed at the top center of the parent plot window. No label is output if the label field is left blank. When outputting a top label the top gap should be increased to about 6 to allow room for it.

### 9.3.1 Label

The label output at the top of the parent plot window. If the label is blank then no label is output.

### 9.3.2 Color

The color of the top label. This is set through the Color Picker Menu. Click the color to bring up the menu.

### 9.3.3 Size

The size of the top label in points. Default is 12 point.

### 9.3.4 Bold

This is option to **ON** if the label should be output in bold text, **OFF** otherwise.

## 9.4 Color

These options define the plot wide color selections.

### 9.4.1 File

This is a set of defined color tables. Plots which convert value to color do so according to the selected color table.

### 9.4.2 Plot Bkgnd

Set the background color of the parent plot window. This is set through the Color Picker Menu. Click the color to bring up the menu. Note that if an object is output in the background color the plotting code will automatically switch the color to its complement. As an example if the background is white then any white text will be output in black

## 9.5 Grid

These options define the characteristics of a background or foreground grid that can be added to any of the defined plots. Grid lines are drawn in conjunction with major tick marks which are drawn along the plot axes. Grids are turned off and on for a given plot in the **Plot Layout Menu**.



### 9.5.1 Lines

A set of menu options which establish which grid lines are to be drawn. Options are **X**, **Y**, or **XY** to include only the X grid lines, only the Y grid lines, or both the X and the Y grid lines respectively.

### 9.5.2 Color

The color of the grid lines. This is set through the Color Picker Menu. Click the color to bring up the menu.

### 9.5.3 Line Style

The line style to use when outputting the grid. The default setting is **SPDASH1**. The options for the most part are self explanatory. **SOLID** produced solid lines. The **DASH** options produce dashed lines with different dash widths. The **SPDASH** options produce **DASH** lines but add a larger amount of space between the dashes. The **DASHDOT** options produce lines with alternating dashes and dots (smaller dashes). The **DASH2DOT** options produce lines with dashes separated by 2 dots. The **DASHSPDOT** options produce **DASHDOT** lines but with larger spaces between the dashes and dots.

### 9.5.4 Add

This is a toggle option which specifies when to grid the plot. The options are **BEFORE** and **AFTER**. Use **BEFORE** to output the grid prior to plotting of any data and **AFTER** to output the grid after all data have been plotted.

## 9.6 Offsets

These options allow for additional shifts in the numerical labeling around plots from the default.

### 9.6.1 X Ano

Adds an additional offset of the X axis numerical annotation from the plot axis. The value has units of characters. This applies to all defined plots.

### 9.6.2 Y Ano

Adds an additional offset of the Y axis numerical annotation from the plot axis. The value has units of characters. This applies to all defined plots.

## 9.7 3D Plots

These options set global options to use when outputting 3D plots with with the VTK device driver.

### 9.7.1 Format

This option specifies whether spectrogram like plots should be plotted either as **FLAT** plots (2D representation) or **ELEVATED** plots (3D).

### 9.7.2 Scale

This option applies only if the **Format** option is set to **ELEVATE** and defines the scaling to use when computing out of plane elevation. A larger value exaggerates the elevations.

### 9.7.3 Decimate

This option applies only if the **Format** option is set to **ELEVATE**. The option allows for plots to be decimated in the X direction prior to being output. Elevated plots are generated from a 2D data grid which often are defined in the X direction to have one grid cell per plot pixel. This is often too dense to produce a good 3D raised representation of the data. Decimating the grid prior to plotting in these cases improves the plot. Set decimate to 1 for no decimation, 2 to use every other X grid cell, 3, to use every second X grid cell, and so on.

## 10 Plot Layout Menu

Plots are defined and positioned within the parent plot window through the **Plot Layout Menu**. (Note: what is actually output in each plot is defined in the **Plot Definitions** menu.) The parent plot window is divided into a rectangular grid of plot cells. This is a dynamic grid which expands and contracts with the number of defined plots. Plots are placed into the grid by specifying their row and column number in the plot grid. The upper left corner is row 0, column 0. Undefined cells in the grid produce blank areas in the plot window.

There are options to define the characteristics of all the axes against which data can be plotted. 2D plots have four independent scalable axes, the top and bottom X and the left and right Y axes. 3D plots make use of these axes plus the Z axis. Plots which use color to represent intensity will make use of the color bar axis options.

Clicking the **Plot Layout Menu** button in the main menu brings up the menu shown in Figure 63.

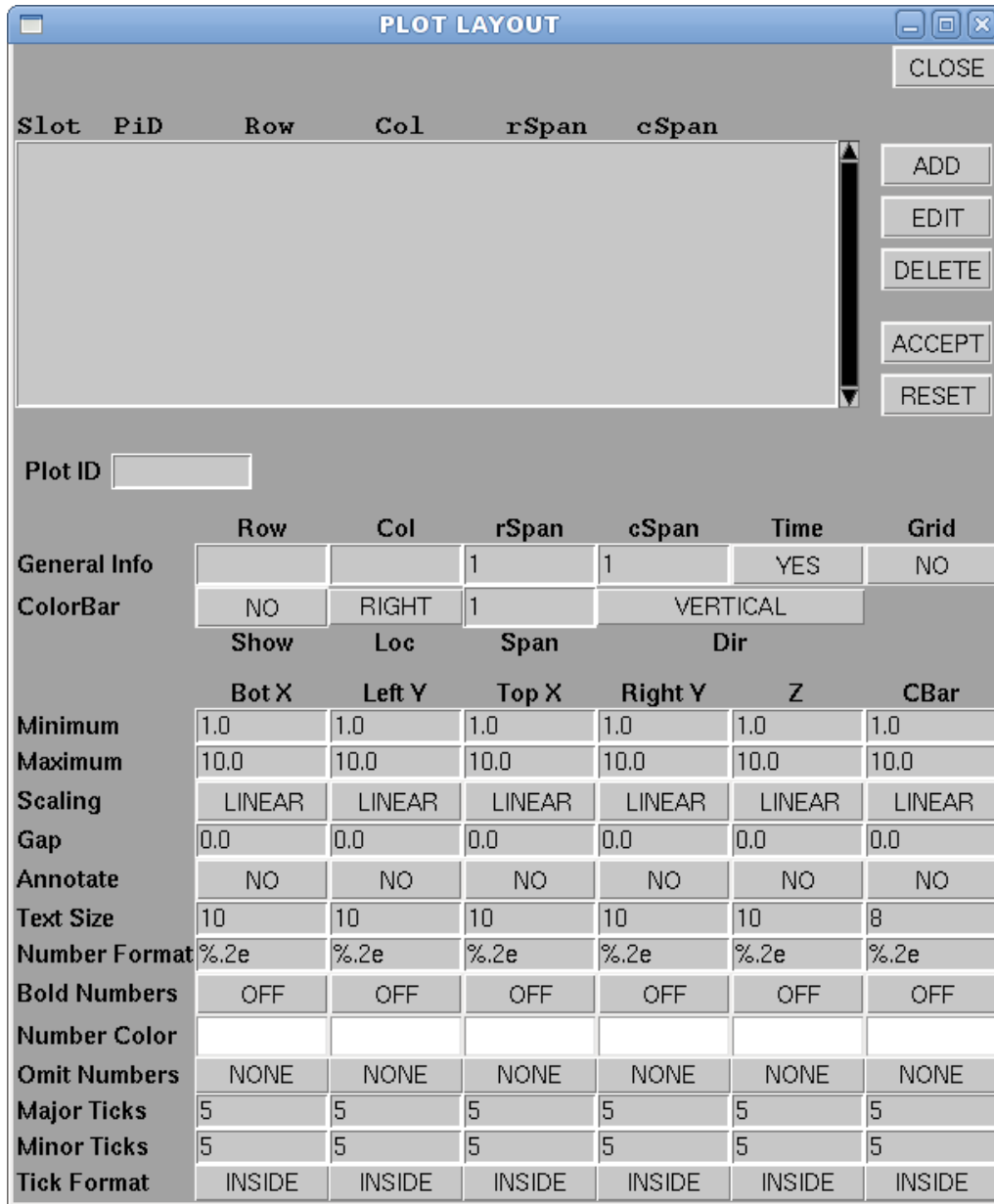


Figure 63: Unpopulated Plot Layout Menu

When first invoked the menu comes up empty with default values are supplied for most options. The menu itself is divided into two areas: a text window which holds a summary of the defined plots and below that the plot definition work area that contains the options associated with the current plot being edited. The text window is initially empty. The order in which the plots appear in the text window is the order that the plot outlines, ticking, and numerical annotation are output. This generally does not make any difference, however, a defined plot can be moved in the stack by highlighting it and then moving it using the up and down arrows.

When starting with an empty **plot setup** menu you can either fill in the plot options and then click **ADD** or just click **ADD** to start with an empty plot definition. In the latter case

you will need to highlight the entry and click **EDIT**. This transfers any current plot settings for the highlighted plot into the plot work area. Edit or set the various options and then click **ACCEPT**. This reflects part of the information back into the text window and transfers all the settings into the structure for that plot. Repeat the process to add new plots until you have defined all the plots required. Figure 64 below shows a populated **Plot Layout** menu with the highlighted plot's definition in the process of being edited.

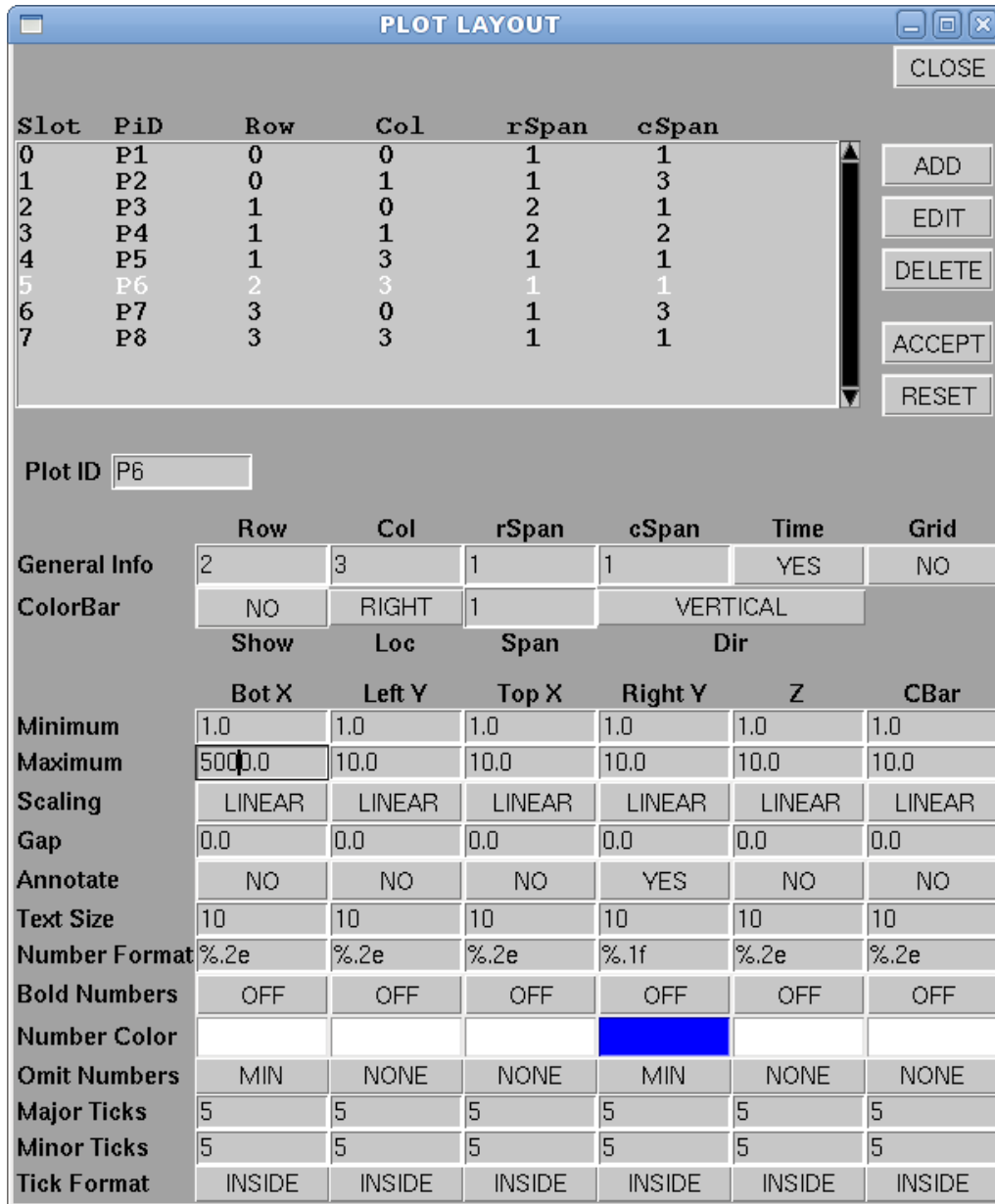


Figure 64: Populated Plot Layout Menu

The full set of options in this menu are described below

## 10.1 The Text Window

The text window holds a summary of the defined plots. It shows the assigned plot identifier, where the plot resides in the plot grid, and whether it spans multiple rows or columns. Also included in the summary is the slot position of the plot which is for internal information only. You can shift a definition around in the window by highlighting it and then moving it up or down using the arrow keys.

There are 5 control buttons associated with the text window. These are to its right and are used to add a new plot, or to modify and or delete an existing one.

### 10.1.1 Add

Adds a new plot definition. The plot is initiated using whatever information currently exists in the work area below the text window. Initiating a new plot does not require that any options be set, including the plot location. These can all be set or modified by editing the definition.

### 10.1.2 Edit

Edit the highlighted plot definition in the text window. Editing a plot transfers its current option settings into the work area below the text window. There you can modify any of the current settings. Click on **ACCEPT** to transfer the options back into the plot definition.

### 10.1.3 Delete

Deletes the currently highlighted plot definition in the text window. There is no undo.

### 10.1.4 Accept

This button accepts all changes made to a plot definition during an edit. It can be activated multiple times in the edit process. Changes do not become permanent and are not reflected in the text window until they are accepted.

### 10.1.5 Reset

This causes all of the options in the work area below the text window to be reset to their default values.

## 10.2 Plot Layout Options

The plot layout options are found in the work area below the text window. This set of options forms the default settings used when creating a new plot definition or contain the current option settings of a plot being edited. It should be emphasized that the work area option settings are not imported into the current plot definition and made permanent until they have been accepted by clicking the **ACCEPT** button.

The options are always in their default state when the **Plot Layout** menu is first invoked and can be returned to their defaulting settings at any time by clicking **RESET** button to the right of the text window.

In the menu the position set of options is used to place the plot in the plot grid in the parent plot window. The remaining options define the plot axes characteristics. The axis options are applied on a per axis basis. Each option runs across a set of columns with each column representing one of the plot axis. The color bar axis is applicable if the plots will have an intensity variable associated with it.

### 10.2.1 Plot Id

The plot identifier used to uniquely identify this plot definition in other menus. A plot identifier of **\_DP\_** signifies that the plot definition to be considered a dummy definition with no output. This is often useful as a place holder in the plot grid to preserve plot alignments in the columns and rows.

### 10.2.2 Row

This row number in the plot grid where this plot is anchored. The first row of plots across the top of the plot window is row 0.

### 10.2.3 Col

The column number in the plot grid where this plot is anchored. The first column of plots down the left hand side of the plot window is column 0.

### 10.2.4 Span Row

The number of rows the plot spans in the plot grid. This is defaulted to 1 so that the plot only occupies one cell in the grid row. Setting this to a larger value stretches a plot in the Y direction beginning at its anchor position.

### 10.2.5 Span Col

The number of columns the plot spans in the plot grid. This is defaulted to 1 so that the plot only occupies one cell in the grid column. Setting this to a larger value stretches a plot in the X direction beginning at its anchor position.

### 10.2.6 Time

Set this option to **YES** if this is to be a time based plot, set to **NO** otherwise.

### 10.2.7 Grid

Set this option to **YES** if a background or foreground grid is to be included with the plot. Set to **NO** if no grid is to be included. If a grid is to be included it is drawn with the grid characteristics set in the **Plot Setup Menu**.

### 10.2.8 Show Colorbar

Set this option to **YES** if you want a colorbar output in conjunction with this plot. The colorbar will be scaled and annotated according to the setting of the options in the **CBar** column of the menu. Set this option to **NO** if no colorbar is to be output.

### 10.2.9 Colorbar Loc

If the **Show Colorbar** option is set to **YES** this option sets the location relative to the plot where the colorbar is to be output. This can be either **RIGHT**, **LEFT**, **TOP**, or **BOTTOM** to output the colorbar to the right, left, above, or below the plot.

### 10.2.10 Colorbar Span

If the **Show Colorbar** option is set to **YES** this is number of rows in the plot grid to span for a vertically output colorbar or the number of columns in the plot grid to span for a horizontally output colorbar. You can add a dummy span value to the span value using the format S,DS where S is the span value and DS is the dummy span value. A dummy span value adds room for a colorbar to a plot without the colorbar being output. This is useful at times for plot alignment within the plot grid.

### 10.2.11 Colorbar Direction

If the **Show Colorbar** option is set to **YES** this is the orientation to output the colorbar. This can be **VERTICAL**, **RHORIZONTAL** or **LHORIZONTAL**. The vertical orientation should be used when the colorbar is being output to the left or right of a plot and the horizontal orientations when the colorbar is being output above or below the plot. **RHORIZONTAL** is a horizontal colorbar drawn from left to right. With a rainbow color scheme the colors would progress from violet to red from left to right. An **LHORIZONTAL** is the reverse progressing from violet to red from right to left.

### 10.2.12 Minimum

The minimum value associated with an axis. This setting will be ignored if a variable being output in the plot is set to be autoscaled against this axis. In that case the lower scaling will be set in the autoscaling. For **time based** axes this value is always ignored.

### 10.2.13 Maximum

The maximum value associated with an axis. This setting will be ignored if a variable being output in the plot is set to be autoscaled against this axis. In that case the upper scaling will be set in the autoscaling. For **time based** axes this value is always ignored.

### 10.2.14 Scaling

Sets the scaling method to use along an axis. This can be either **LINEAR** or **LOG**. Time bases axis are always scaled **LINEAR**. On **LOG** scaled axes both the axis minimum and

maximum value must be greater than 0.0.

### 10.2.15 Gap

Moves the default location of an axis either inward (positive) or outward (negative). It is generally used to create white space between neighboring axis but can be used to expand a plot to use unused area in a plot grid especially when too much white space is being left for labels. Annotated axes automatically have space left for the annotation. Gap is specified in units of 12 point characters. In general all plots within a row or column of the plot grid should have the same gaps assigned to them. This will give them a common X and Y axis length within the grid.

### 10.2.16 Annotate

Set this option to **YES** if the axis is to be numerically annotated. Set it to **NO** if the axis is to be left unannotated.

### 10.2.17 Text Size

The size of the text to use in the numerical annotation of an axis.

### 10.2.18 Number Format

The C format specification to use when numerically annotating an axis. One non-C format available for use is the format specification **expon** which just outputs the exponent of the numerical label. This should only be used on LOG scaled axes.

### 10.2.19 Bold Numbers

Set **ON** if you want the axis numerical annotation to be output as bold text, otherwise set to **OFF** which is the default setting.

### 10.2.20 Number Color

The color of the numerical annotation along the axis. This is set through the Color Picker Menu. Click the color to bring up the menu. The default text color is White.

### 10.2.21 Omit Numbers

This option allows for either or both of the extreme numerical labels along an axis to be omitted in the annotation. This is sometimes useful when the lower label of one plot partially overlaps the upper label of an adjacent plot. To output both the upper and lower labels set the option to **NONE**, to output neither the upper or lower labels set the option to **BOTH**, to omit just the upper label set the option to **MAX** and to omit just the lower label set the option to **MIN**. You can omit all numerical labels by setting the option to **ALL**.



### 10.2.22 Major Ticks

The number of major tick marks to output along an axis.

### 10.2.23 Minor Ticks

The number of minor tick marks to output between adjacent major tick marks.

### 10.2.24 Tick Format.

The format to use when outputting the axis tick marks. This can be **INSIDE**, **OUTSIDE**, **STRADDLE** or **SPAN**. The **inside** format draws the ticks starting at the axis towards the inside of the plot. The **outside** format draws the ticks starting at the axis towards the outside of the plot. The **straddle** format draws the ticks starting outside the axis and ending inside the axis (they straddle the axis). The **span** format draws the ticks inside the plot running from one axis to the corresponding opposite axis. With the exception of the **span** format the same format is used for both the major and minor tick marks. When using the **span** format only the major tick marks are out put with this format. The minor tick marks are output using the **inside** format.

## 11 Plot Definition Menu

The **Plot Definition Menu** links variables to plots. The plots themselves are set up in the **Plot Setup Menu**. The menu is is invoked directly from the **main menu**.

The analysis package currently supports a number basic plot types including, time based plots and XY plots. Each basic plot type comes in a number of variations. The basic plot type assigned to a plot is determined by the number of axes which have variables defined against them. Including only a Y axis variable in a plot will result in a time-based plot with time along the X axis. Setting up a plot with variables defined against both the X and Y axis produces an XY plot. Further adding a Z variable results in an XYZ plot which can only be displayed if you have VTK installed and have selected **Vtk Canvas** as the device driver to render the graphics. The intensity axis in a plot is represented by color and is activated when a variable is plotted against it.

Clicking the **PLOT DEFINITION** button in the **main menu** brings up the plot definition menu shown in Figure 65.

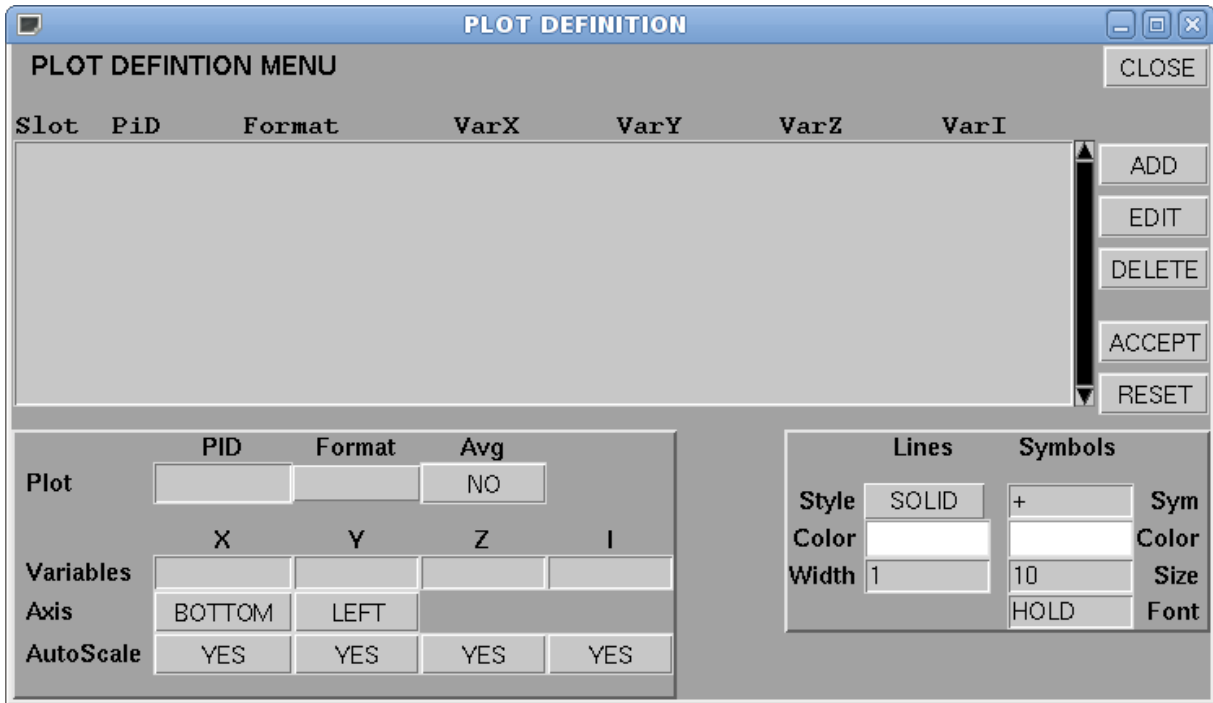


Figure 65: Unpopulated Plot Menu

The menu is divided into two sections. At the top is a text window with a set of controlling options to its right. The text window gives a summary of the plots which have been set up and the controlling options are used to add new plot definitions as well as to edit existing ones. Plots are output in the order of the plot definitions in the window. You can shift a definition around in the window by highlighting it and then moving it up or down using the arrow keys. The options below the text window constitute a work area. When a plot definition is being edited its current definition is copied to the work area where changes are made and then copied back into the definition.

To create a plot from an empty menu, click the **ADD** button. This creates a new plot definition in the text window populating it with the current work area settings. You can set some or all of the work area options settings before clicking ADD but that's not necessary. Highlight the added entry in the text window and click the **EDIT** button. This transfers the plot definition option settings into the work. Since you created the new plot definition from the current work area settings you won't see any change. Next make whatever changes to the plot definitions you want to make. When you are complete click **ACCEPT** to accept the changes. Some of the changes will be reflected back into the text window. Repeat the above steps until you have defined all of the plots you want to define. Figure 66 shows a populated **Plot Definition** menu in the process of being edited.

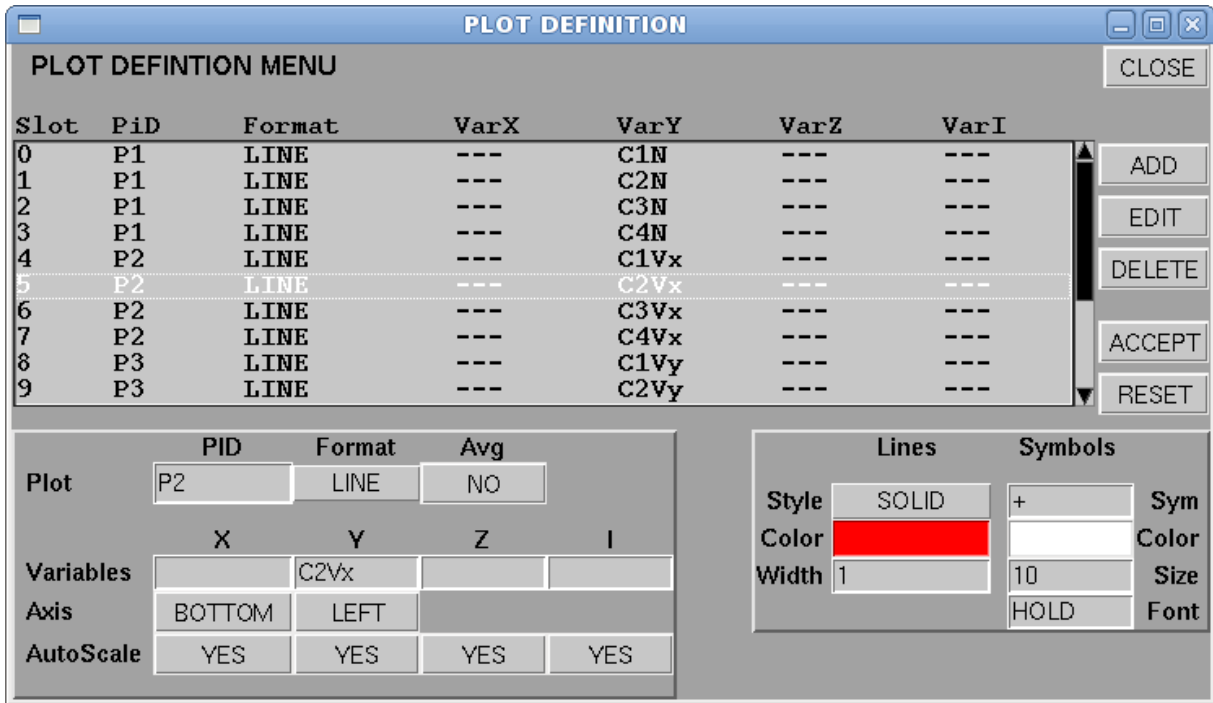


Figure 66: Populated Plot Menu

## 11.1 Plot Definition Options

The following describes the options found below in the work area below the text window. These options link a variable with a plot and define various output (drawing) options.

### 11.1.1 Plot PID

The plot into which the variable is output. It must match one of the plot ID's assigned to plots defined in the **PLOT SETUP** menu.

### 11.1.2 Plot Format

The output method used to display the data. This can be: **LINE**, **POINT**, **BARS**, **SOLID** or **CONTOURS**. The **contours** option is not currently implemented. The output formats applicable to a specific plot depend on both the plot and whether it is being output as a 2D (TK driver) or 3D (VTK driver) plot.

XY (line plots) can be output with any of the first three options. With the **LINE** option the individual data are connected by lines, with the **POINT** option the individual data are represented by symbols (no lines), and with the **BARS** option the data are represented as solid rectangles from the plot baseline to the data values.

Spectrogram format plots (XYI) can be output using the **SOLID** option. XYI plots can also be output using the **POINT** option which will produce a scatter plot with each symbol color coded according to its intensity.

Higher dimension plots require the use of the VTK graphics driver. XYZ plots can be output with either the **LINE** or **POINT** option and XYZI plots can be output with the **POINT** option.

### 11.1.3 Avg

For time based plots set this option to **YES** if you want the data being output within each plot pixel to be averaged over the pixel time width prior to plotting otherwise leave at the default setting of **NO**. Averaging reduces noise when there are multiple data values output within pixels.

### 11.1.4 VarX

The name of the variable to be plotted against the X axis of the plot. It is used in all plot types except **TIME** based plots in which case it should be left blank. The time variable for time based plots is an internal variable derived from the time grids.

### 11.1.5 VarY

The name of the variable to be plotted against the Y axis of the plot. All plot types require this variable.

### 11.1.6 VarZ

The name of the variable to be plotted against the Z axis of the plot. Leave this blank if there is no Z variable. Inclusion requires both that the VTK graphics package be installed on the local system and that the device driver used to render the graphics be set to VTK.

### 11.1.7 VarI

The name of the variable to be plotted against the intensity or color axis. Currently only plots in which the data is output as **POINTS** make use of this variable. Leave this field blank if there is no intensity data.

### 11.1.8 X Axis

The plot X axis the **VarX** data is to be plotted against. This can be either **TOP** or **BOTTOM**. A plot can have variables plotted against either or both axes.

### 11.1.9 Y Axis

The plot Y axis the **VarY** data is to be plotted against. This can be either **LEFT** or **RIGHT**. A plot can have variables plotted against either or both axes.

### 11.1.10 Autoscale

This set of options determines if an axis should be autoscaled from the values of the variable being plotted against. Set to either **YES** or **NO**. **TIME BASED** plots do not make use of this option. If multiple variables being plotted against this axis have autoscale set to **YES** then the axis maximum and minimum value is determined from the combined inputs of all the variables. If an axis is not autoscaled by any variable then the axis maximum and minimum value is set according to the axis scaling set in the **PLOT SETUP** menus.

### 11.1.11 Line Style

The line style to use in a line plot. The default setting is **SOLID**. The options for the most part are self explanatory. The **DASH** options produce dashed lines with different dash widths. The **SPDASH** options produce **DASH** lines but add a larger amount of space between the dashes. The **DASHDOT** options produce lines with alternating dashes and dots (smaller dashes). The **DASH2DOT** options produce lines with dashes separated by 2 dots. The **DASHSPDOT** options produce **DASHDOT** lines but with larger spaces between the dashes and dots.

### 11.1.12 Line Color

The color of the trace in a line plot. This is set through the Color Picker Menu. Click the color to bring up the menu. The default is white.

### 11.1.13 Line Width

The width of the line drawn in a line plot. The thinnest line has width of 1 (default value).

### 11.1.14 Symbol

The character or text to use when outputting data in the **POINT** format. In addition to the standard alpha/numeric characters you can also specify UDF-8 (unicode) characters as `\u#` where # is the hex UTF-8 specification. Right clicking in the window will bring up the Symbol Picker menu which contains a number of predefined symbols.

### 11.1.15 Symbol Color

The text color to use in **POINT** plots. This is set through the Color Picker Menu. Click the color to bring up the menu. The default is white. **NOTE:** This option does not apply to plots with an intensity input.

### 11.1.16 Symbol Size

The symbol size in points.

### 11.1.17 Symbol Font

The text font to use when outputting symbols in **POINT** plots. The menu default setting is **HOLD** which sets the font to the currently loaded font.

## 11.2 Adding or Modifying Plot Definitions

The buttons to the right of text windows allow for the addition, modification and removal of selected plot definitions. Their actions are described below.

### 11.2.1 Add

Adds a new plot definition in the text window. For a new label definition to be added a plot definition must have been selected for editing. The new label will be associated with that plot. Adding simply takes the current plot work area option settings and assigns them to the new plot or label.

### 11.2.2 Edit

Edit the highlighted plot definition. This copies the current plot options into the work area where they can be modified. After making any changes click the **ACCEPT** button to copy them back into the plot definition and update its the text window entry.

### 11.2.3 Delete

Delete a highlighted plot. If this is a plot definition then its associate label definitions will also be deleted.

### 11.2.4 Accept

Accepts the changes made in an edit. This copies the work area setting back into the plot settings and makes them the current plot settings. It also updates the definition in the text window.

### 11.2.5 Reset

Resets all of the options in the work area to their default values. If an plot is being edited this does not activate the changes until they are **ACCEPTED**.

## 12 Plot Labels Menu

The **Plot Labels Menu** contains options used to define and place labels within the parent plotting window. The menu is is invoked directly from the **Main Menu**.

Putting labels within a plot can be either easy or frustrating depending on where the label is to be placed. Labeling plot axes or putting up labels aligned with plot axes is generally straight forward while trying to place a label at a specific location within a plot is often

challenging. Labels within the UDFAnalysis framework are defined and output relative to plot axes, but, once output can be dragged and dropped into new locations on within the parent plot window. The final position is reflected back into the menus. Saving the menu after replacement of labels ensures that they will appear at their new location the next time the program is run from the menu. So for labels which need to be set at an exact location you just need to put it somewhere on the screen and then move it to where you want it when you first execute the menu.

Label positions are anchored to either the minimum, maximum or center position of any axis of a plot defined in the **Plot Setup** menu. This can be shifted to either inside or outside of the plot box which moves the label anchor point to be positioned at the edge of any axis tick marks or annotation extending in that direction. If, for example, a label is being output outside of a plot on an axis which is not numerical labeled and the tick marks extend inside the plot, then the anchor point will lie on the axis. The anchor can then be further offset in both X and Y. In practice this allows a label to be placed anywhere within the parent plot window.

To move a label move once it is displayed move the mouse cursor over the label and left click. Then move the mouse to the position on the plot where the label is to be moved and right click. This will move the label to the new position which becomes the new label anchor point. The label output at the new anchor point using the justification specified in the label definition.

If you are putting labels around the edge of the parent plot window or between plots it may be necessary to add extra white space to accommodate a label using the **Gap** options in the **Plot Setup** menu.

Clicking the **PLOT LABELS** button in the **Main Menu** brings up the plot label menu shown in Figure 67.

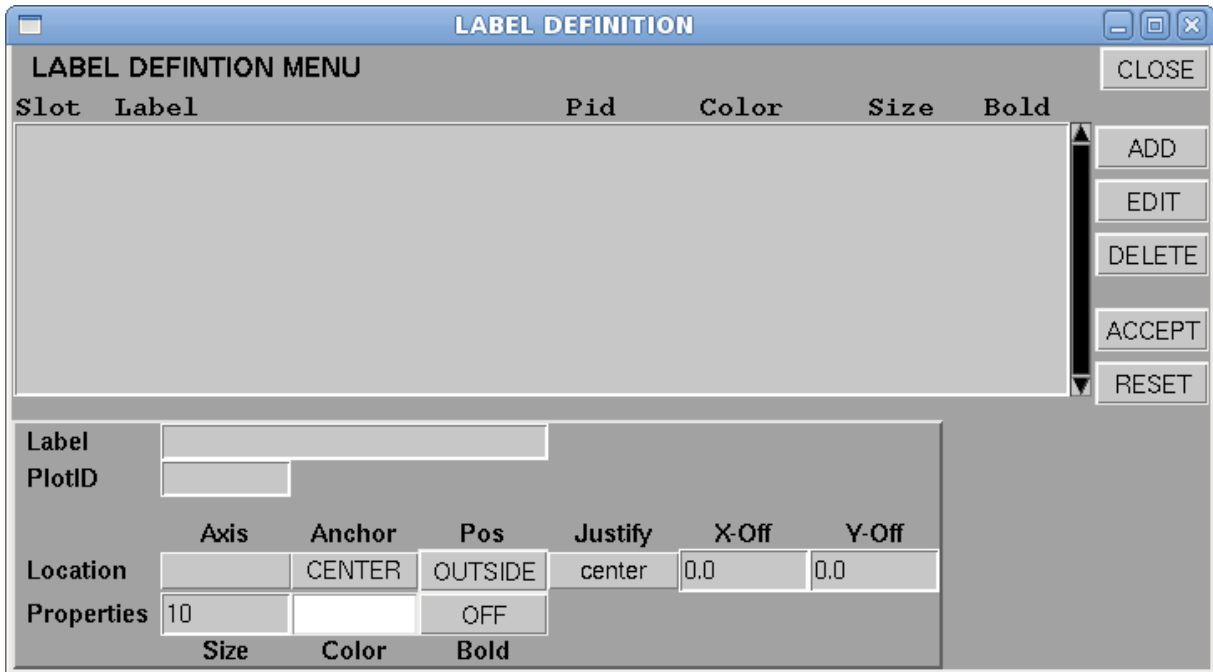


Figure 67: Unpopulated Label Menu

The menu is divided into two sections. At the top is a text window with a set of controlling options to its right. The text window gives a summary of the labels which have been defined and the controlling options are used to add new label definitions as well as to edit existing ones. Labels are output in the order of the label definitions in the window. You can shift a definition around in the window by highlighting it and then moving it up or down using the arrow keys. The options below the text window constitute a work area. When a label definition is being edited its current definition is copied to the work area where changes are made and then copied back into the definition.

To create a label from an empty menu, click the **ADD** button. This creates a new label definition in the text window populating it with the current work area settings. You can set some or all of the work area options settings before clicking ADD but that's not necessary. Highlight the added entry in the text window and click the **EDIT** button. This transfers the current label option settings into the work. Since you created the new label definition from the current work area settings you won't see any changes. Next make whatever changes to the label definitions you want to make. When you are complete click **ACCEPT** to accept the changes. Some of the changes will be reflected back into the text window. Repeat the above steps until you have defined all of the labels you want to define. The figure below shows a populated **Plot Labels** menu in the process of being edited.



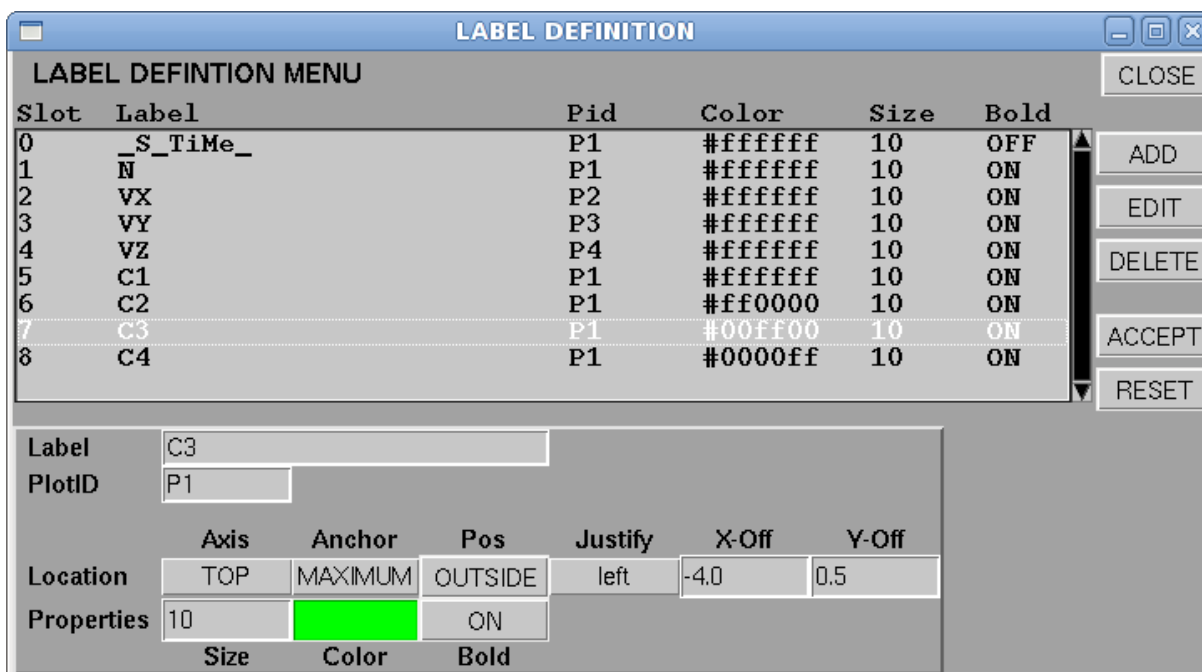


Figure 68: Populated Label Menu

## 12.1 Label Definition Options

The following describes the options found below the label text window.

### 12.1.1 Label

The label. This can contain both ASCII and UDF-8 (unicode) characters and data within internal variables. UDF-8 characters are specified as `\u#` where `#` is the hex UTF-8 designation. You can also right click within the label box which will bring up the Symbol Picker Menu which contains a number of predefined symbols and characters which can be added to any label.

There are 3 predefined labels which can be output. The label `_S_TiMe_` translates to the beginning time of a plot as **Year Day Hr:Mn:Sec.Msec**. The `_E_TiMe_` label does the same to the ending time of a plot. The label `_SE_TiMe` is equivalent to `_S_TiMe_ TO _E_TiMe_`. These find use both in time based plots where the initial time of the plot may not be labeled on the axis and in non-time based plots to show the time range covered by a plot.

Data stored in internal variables can included in a label by specifying it as `$V__Fmt` where `V` is the variable to be output and `__Fmt` is an optional format statement to use when converting the variable to text. If omitted the variable is output using the format `%s`. In the label the variable specification must be space delineated unless it is in the leading or ending position. **NOTE:** the `__` in the format is **2** underscores.

As an example, suppose you have run a fit on a set of data to a first order polynomial. The fit information was returned in the variable `fP`. You want to include both the slope of the fit (the fit coefficient at array index 1) and the goodness of fit parameter as labels to a

plot. The values should be output with three decimal places of accuracy. These labels can be stated as:

"SLOPE = \$fP(1)\_{%.3f}"

"CHI SQ = \$fP(gFit)\_{%.3f}"

### 12.1.2 PlotID

The identifier of the plot the label position is computed relative to. This was set in the **Plot Setup** menu when the plot was defined. If a label is manually repositioned in a plot this will be set to **\_PW\_** to indicate that the label is being output at an absolute position in the parent plot window. In this case neither the **Anchor** or **Pos** options are used, the **Axis** option is used to determine the direction the label is to be drawn (horizontal or vertical) and the **X-Off** and **Y-Off** hold the absolute anchor location of the label in the parent plot window.

### 12.1.3 Axis

The axis against which the label is to be output. This can be the **BOTTOM** or **TOP** X axes or the **LEFT** or **RIGHT** Y axes. Labels output against a Y axis are output vertically and those against an X axis horizontally. This option is not used if the **PlotID** options is **\_PW\_**.

### 12.1.4 Anchor

The anchor position of the label along the selected **Axis**. This can be at the axis **MINIMUM**, **CENTER**, or **MAXIMUM** position. This option is not used if the **PlotID** options is **\_PW\_**.

### 12.1.5 Pos

Indicates if the label anchor position it is to be positioned **INSIDE** or **OUTSIDE** of the plot box.

### 12.1.6 Justify

The label justification with respect to the anchor point. Labels can be **left**, **right**, or **center** justified. **Left** justification places the right-hand edge of the text against the anchor position (text is to the left of the anchor position), **right** justification text places the left-hand edge of the text against the anchor position (text is to the right of the anchor position) and **center** centers the text on the anchor point.

### 12.1.7 X-Off

When **PlotID** holds a plot identifier associated with plot defined in the **Plot Setup** menu this is an offset to move the label anchor point in the X direction. If the label is associated with a Y axis then positive offsets move the anchor point away from the axis otherwise positive offsets move the anchor point to the right and negative offsets move the anchor point to the left. If the **PlotID** options is **\_PW\_** then this option holds the absolute X position of the label in the parent plotting window.

### 12.1.8 Y-Off

When **PlotID** holds a plot identifier associated with plot defined in the **Plot Setup** menu this is an offset to move the label anchor point in the Y direction. The distance moved is given in units of characters. If the label is associated with an X axis then positive offsets move the anchor point away from the axis otherwise positive offsets move the anchor point upward and negative offsets move the anchor point downwards. If the **PlotID** options is **\_PW\_** then this option holds the absolute Y position of the label in the parent plotting window.

### 12.1.9 Size

The label size in points.

### 12.1.10 Color

The color of the label. This is set through the Color Picker Menu. Click the color to bring up the menu. The default is white.

### 12.1.11 Bold

Set to **ON** is if the label is to output in a bold font otherwise set to **OFF**.

## 12.2 Adding or Modifying Label Definitions

The buttons to the right of text windows allow for the addition, modification and removal of selected label definitions. Their actions are described below.

### 12.2.1 Add

Adds a new label definition in the text window. For a new label definition to be added a plot definition must have been selected for editing The new label will be associated with that plot. Adding simply takes the current label work area option settings and assigns them to the new plot or label.

### 12.2.2 Edit

Edit the highlighted label definition. This copies the current label options into the work area where they can be modified. After making any changes click the **ACCEPT** button to copy them back into the label definition and update its the text window entry.

### 12.2.3 Delete

Delete a highlighted label. If this is a plot definition then its associate label definitions will also be deleted.

### 12.2.4 Accept

Accepts the changes made in an edit. This copies the work area setting back into the label settings and makes them the current label settings. It also updates the definition in the text window.

### 12.2.5 Reset

Resets all of the options in the work area to their default values. If an label is being edited this does not activate the changes until they are **ACCEPTED**.