The Kst Handbook

Funding of this project was provided in part by the Canadian Space Agency.

COLLABORATORS

	TITLE : The Kst Handbook		*REFERENCE* :
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Duncan Hanson, Rick Chern, Philip Rodrigues, Barth Netterfield, Yiwen Mao, and Zongyi Zhang	2010-11-29	

Contents

List of Tables

Abstract

Kst is a data plotting and manipulation program with powerful plugin support.

Chapter 1

Introduction

1.1 What is Kst?

Kst is a data plotting and analysis program. Some of its features include:

- Robust plotting of live ‘streaming’ data, including support for automated monitoring of events and notification.
- Powerful plugin and extensions support.
- A large selection of built-in plotting and analysis functions, including: histograms, equations, curve fitting, filtering, and power spectra.
- Powerful graphical user interface, with unique view modes for easy plot manipulation.
- Color mapping and contour mapping capabilities for three-dimensional data.
- Support for several popular data formats including FITS, netCDF, and HEALPix, as well as ASCII. Additional formats can be accommodated with plugins.
- KstScript, a JavaScript based scripting language which allows automation of complicated workflows.

1.2 Getting Started

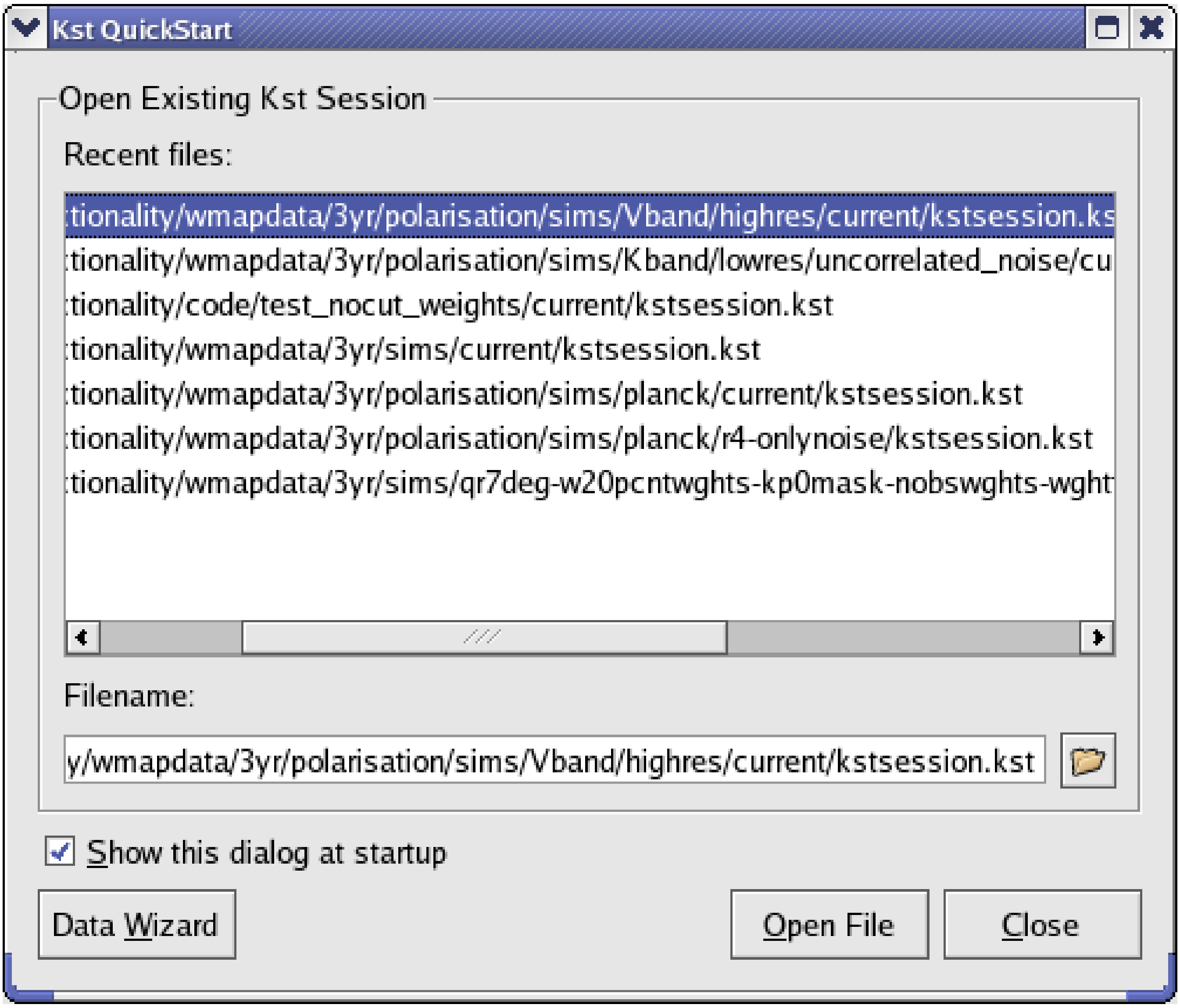
This is a brief introductory tutorial to the fundamental features of Kst. We will cover...

- Importing and plotting raw data
- Plot manipulation
- Basic data analysis with Kst

1.2.1 Importing Data

Let’s start by seeing how raw data can be imported into Kst using the graphical interface (this can also be done using the command line, for details see the appendix on Command Line Options).

Begin by running Kst. When you do this for the first time, a ‘QuickStart’ Dialog should appear by default.



:tionality/wmapdata/3yr/polarisation/sims/Vband/highres/current/kstsession.ks
:tionality/wmapdata/3yr/polarisation/sims/Kband/lowres/uncorrelated_noise/cu
:tionality/code/test_nocut_weights/current/kstsession.kst
:tionality/wmapdata/3yr/sims/current/kstsession.kst
:tionality/wmapdata/3yr/polarisation/sims/planck/current/kstsession.kst
:tionality/wmapdata/3yr/polarisation/sims/planck/r4-onlynoise/kstsession.kst
:tionality/wmapdata/3yr/sims/qr7deg-w20pcntwghts-kp0mask-nobswghts-wght

Filename: y/wmapdata/3yr/polarisation/sims/Vband/highres/current/kstsession.kst

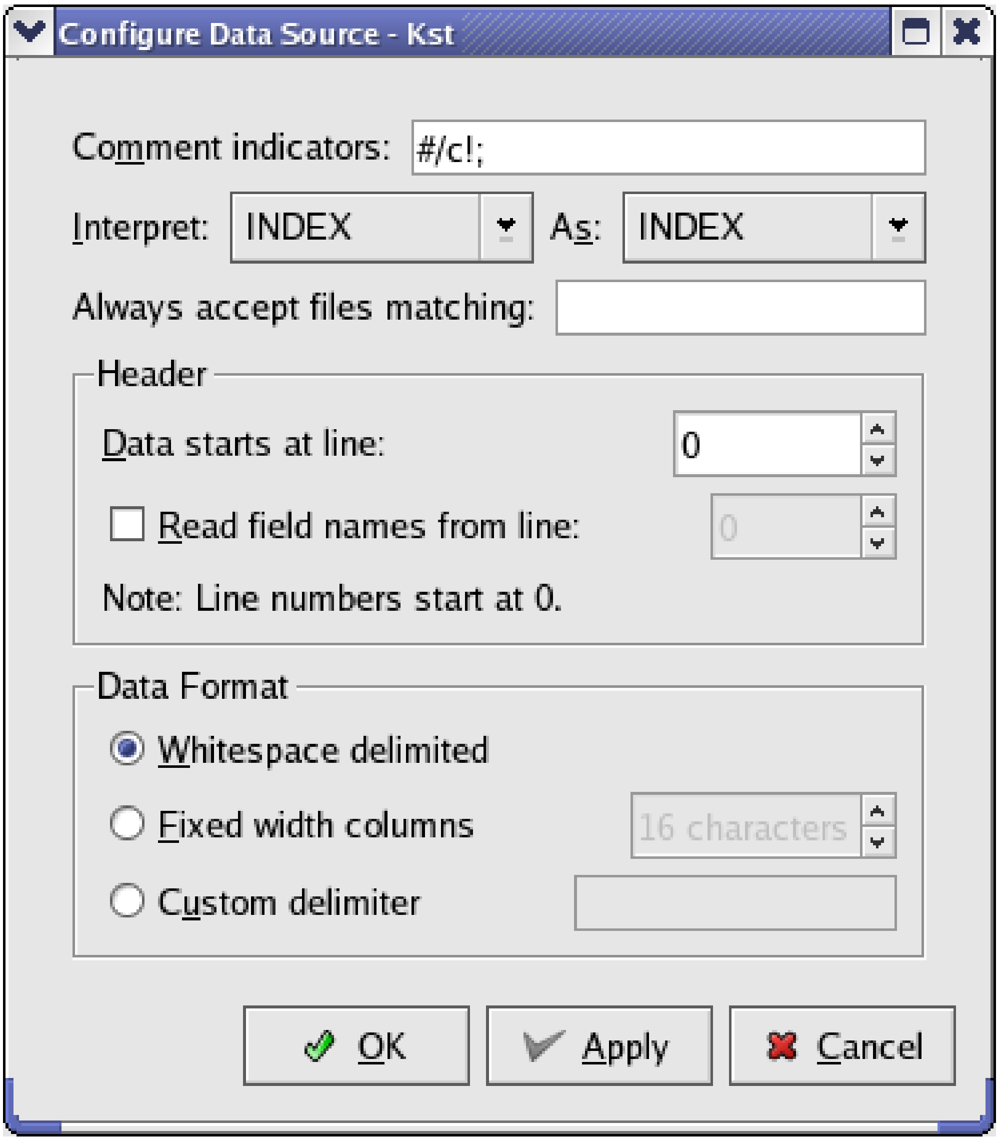
☑ Show this dialog at startup

This dialog has a list of your recently opened Kst sessions, as well as a link to the Data Wizard. The Data Wizard allows data to be imported into Kst using the graphical interface. We'll run through using it now.

Click the Data Wizard button. A data source dialog will appear which allows you to select a file or network resource (e.g. HTTP address) to use as a data source. Kst determines the format of the source by its extension and contents.

We will import data from a file called 'gyrodata.dat' which should be included in your Kst installation. The location of this file is `$KDEDIR/share/apps/kst/tutorial/gyrodata.dat`, where `$KDEDIR` is the location in which KDE is installed on your system (you can find this using the command **kde-config --prefix**).

Select `gyrodata.dat` for opening. Kst should report the data source as type ASCII, which means that the data is in delimited plain text format. More options can be seen by clicking the Configure button. The options for the gyrodata.dat file should be as shown below:



Configure Data Source - Kst

Comment indicators: #/c!;

Interpret: INDEX As: INDEX

Always accept files matching:

Header

Data starts at line: 0

□ Read field names from line: 0

Note: Line numbers start at 0.

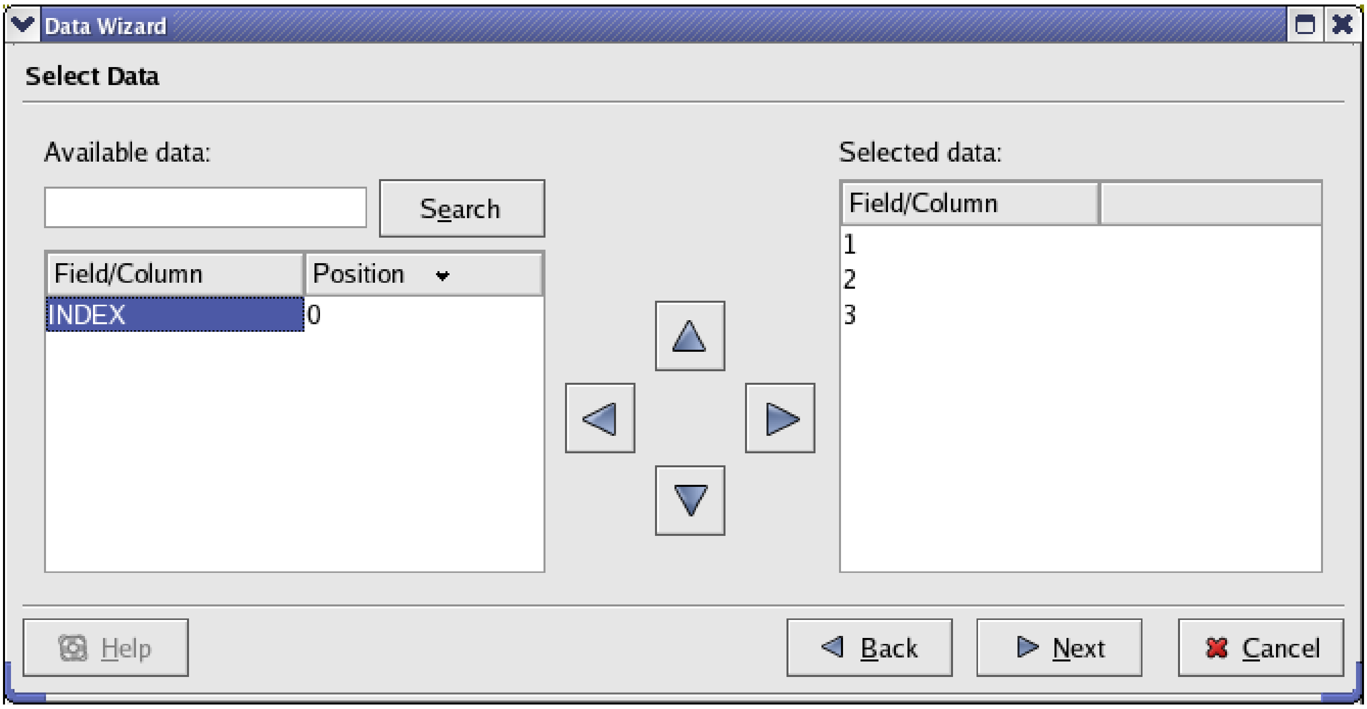
Data Format

● Whitespace delimited

○ Fixed width columns 16 characters

○ Custom delimiter

Close the Configure Data Source Dialog and click the Next button. This will lead to a pane where you can select the fields from the data source which you'd like to work with. In the case of ASCII data, each field is a column in the text file. Select the data from the 1st, 2nd, and 3rd fields. The INDEX field is a supplementary field created by Kst. It contains integers from 0 to N-1, where N is the number of frames in the data file. Once you've selected the proper fields, the selection pane should look like this:



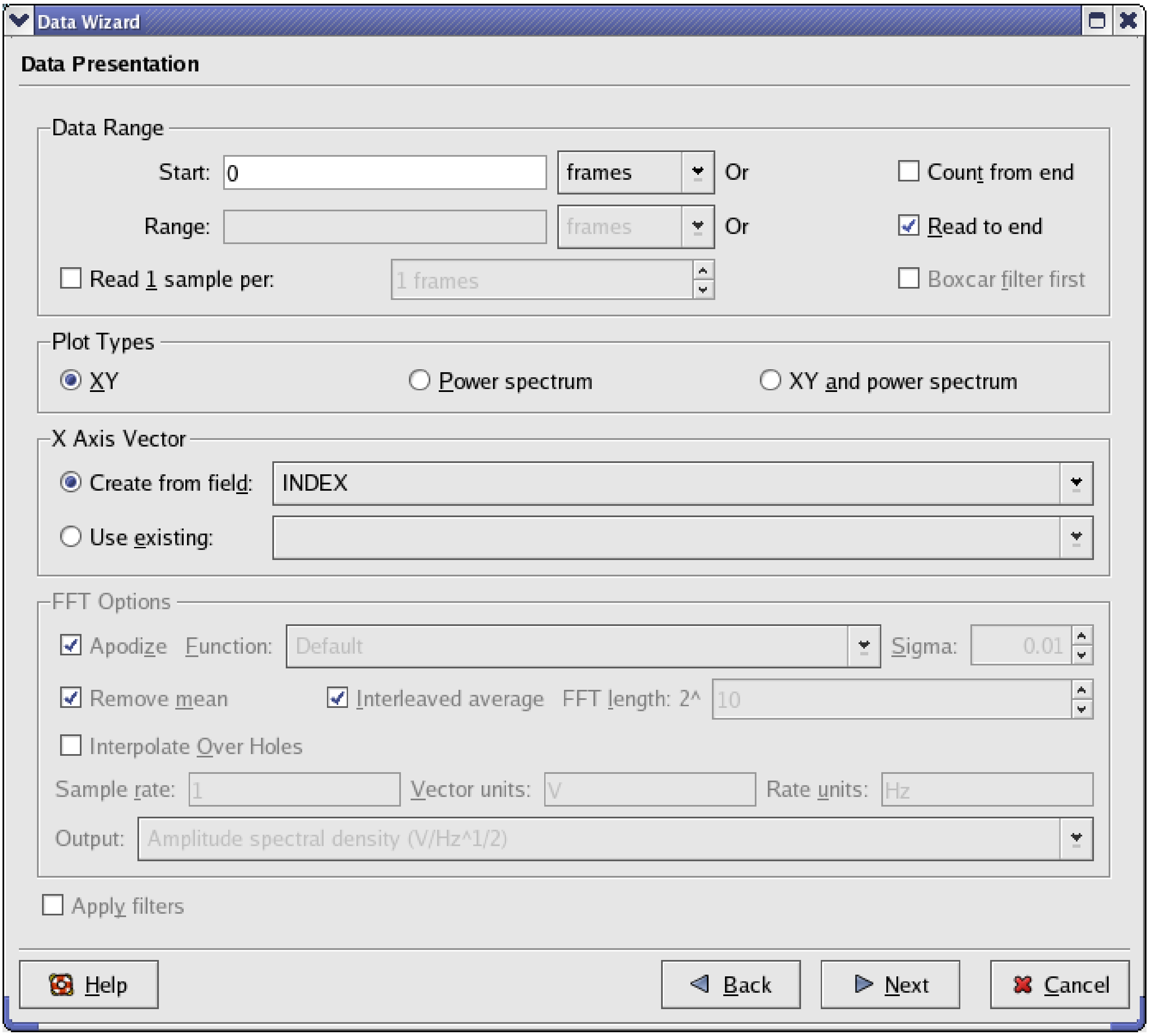
Field/Column	Position
INDEX	0

Field/Column
1
2
3

Click the Next button again, to get to the Data Presentation Pane. This allows you to select some basic and frequently modified options for the plots which are about to be created.

The Data Range section is used to specify the subset of data to read from the selected vectors in the input file. For more information about this pane see the Data Wizard documentation.

Configure the Data Presentation Pane to look like the following:



Data Wizard

Data Presentation

Data Range

Start: 0 frames Or ☐ Count from end

Range: frames Or ☑ Read to end

☐ Read 1 sample per: 1 frames ☐ Boxcar filter first

Plot Types

● XY ○ Power spectrum ○ XY and power spectrum

X Axis Vector

● Create from field: INDEX

○ Use existing:

FFT Options

☑ Apodize Function: Default Sigma: 0.01

☑ Remove mean ☑ Interleaved average FFT length: 2^ 10

☐ Interpolate Over Holes

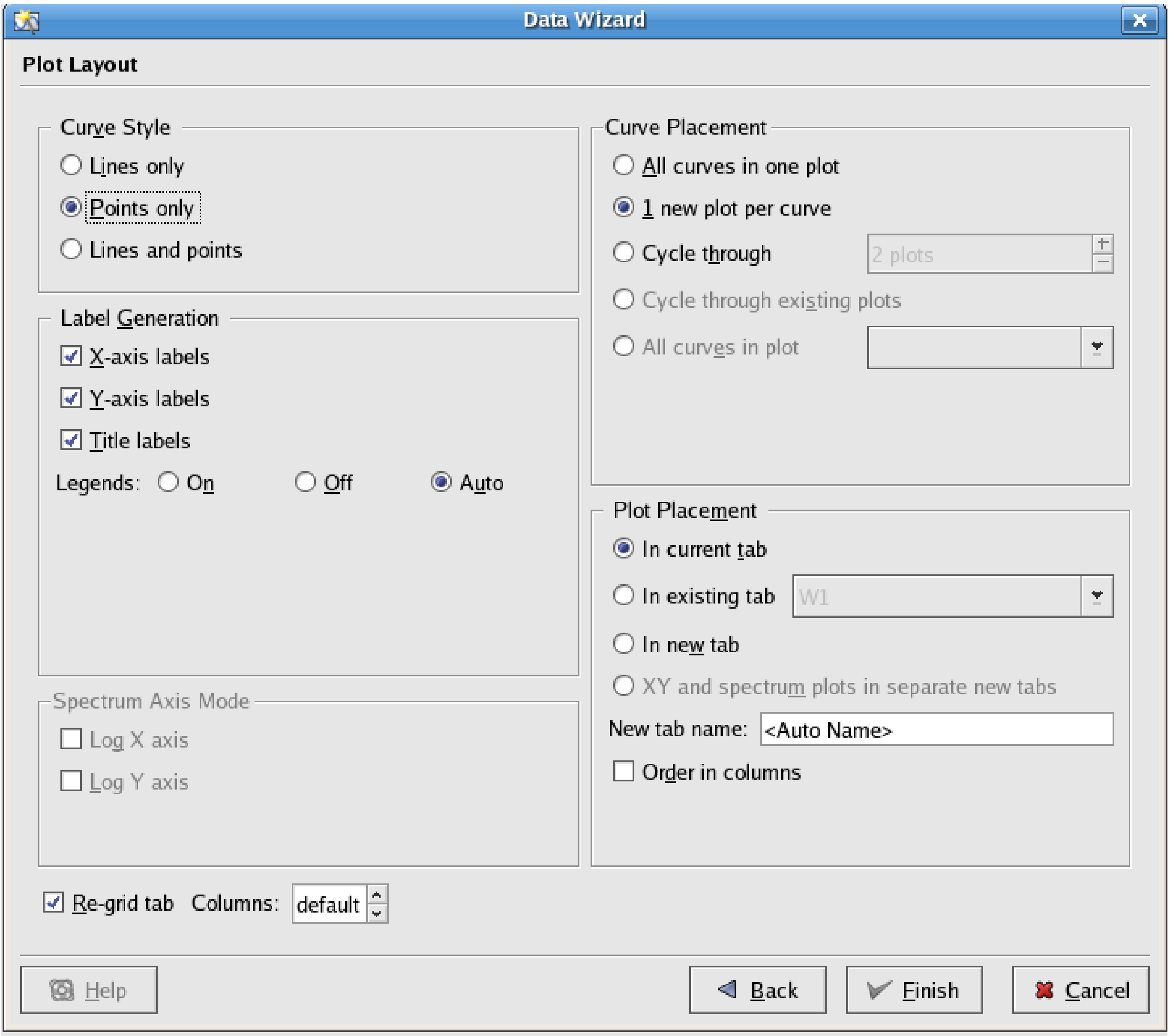
Sample rate: 1 Vector units: V Rate units: Hz

Output: Amplitude spectral density (V/Hz^1/2)

☐ Apply filters

This tells Kst that we would like to read all of the available data for our chosen fields, and that we would like to generate plots with the x-axis of the data as its INDEX. INDEX is an automatically generated vector which gives the position of the data elements within the file.

Clicking the Next button again takes us to the final Data Wizard pane, where there are options to determine the appearance of the plotted data. We can select a curve style, decide whether labels and legends should be generated from the available field names, and decide which plots the curves should be placed into. Again, the default options should be fine for now.



Plot Layout

Curve Style

○ Lines only

● Points only

○ Lines and points

Label Generation

☑ X-axis labels

☑ Y-axis labels

☑ Title labels

Legends: ○ On ○ Off ● Auto

Spectrum Axis Mode

□ Log X axis

□ Log Y axis

Curve Placement

○ All curves in one plot

● 1 new plot per curve

○ Cycle through 2 plots

○ Cycle through existing plots

○ All curves in plot

Plot Placement

● In current tab

○ In existing tab W1

○ In new tab

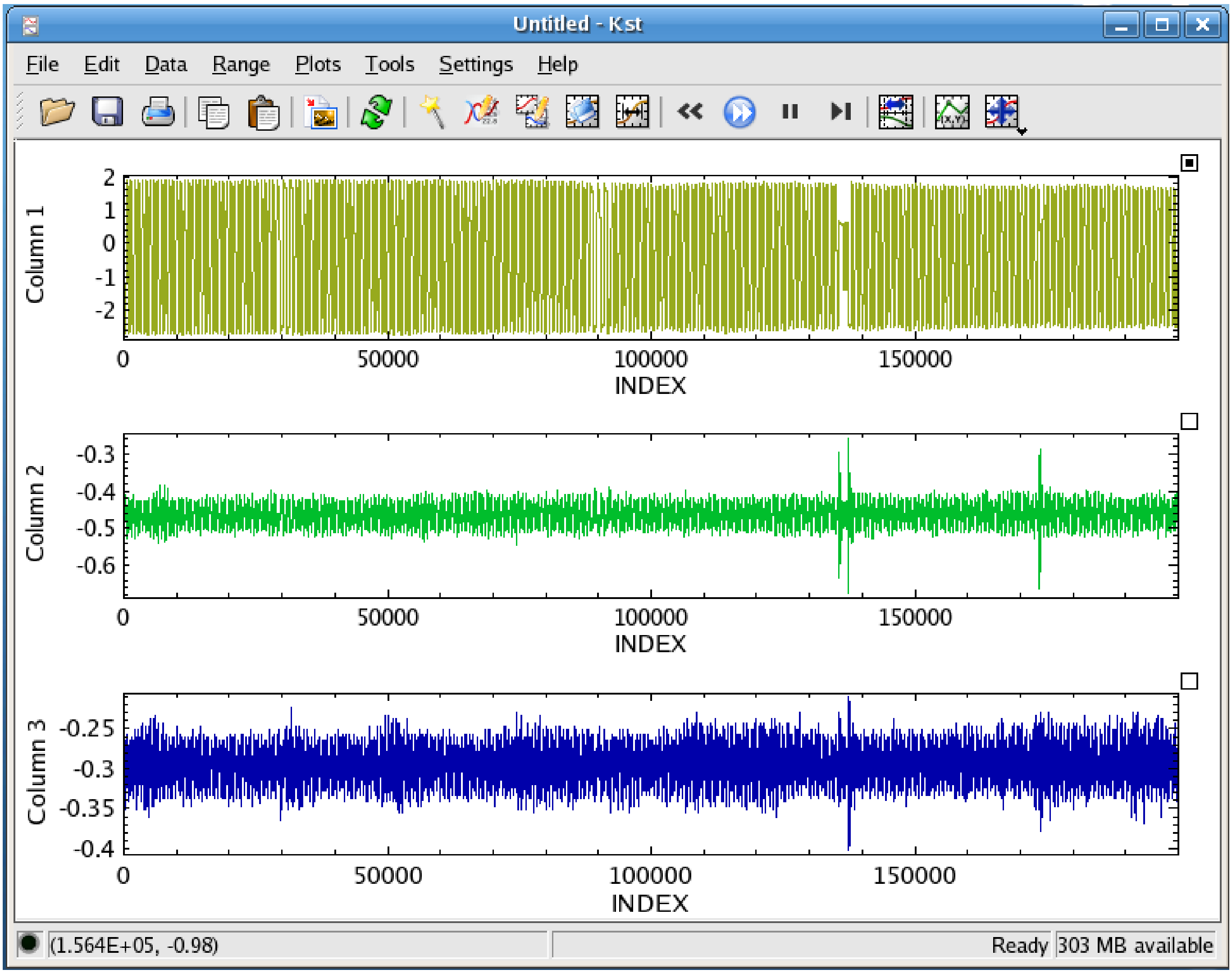
○ XY and spectrum plots in separate new tabs

New tab name: <Auto Name>

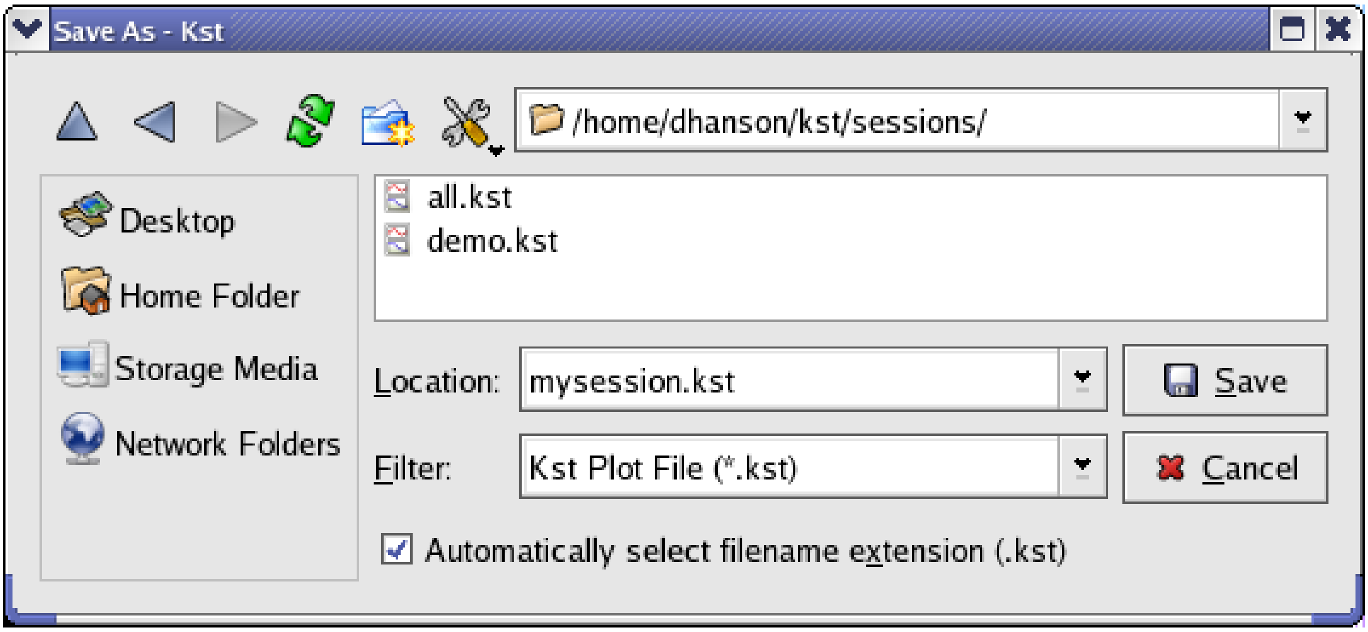
□ Order in columns

☑ Re-grid tab Columns: default

Now, click the Finish button to generate the plots. Your Kst session should now look something like this:



Congratulations! You've just done some basic plotting in Kst. Generating these plots took a bit of effort, so we should save the current Kst session (it will be used in the next section of this tutorial). Select Save As... from the File menu, and save the session as `mykstsession.kst`:



Location: mysession.kst

Filter: Kst Plot File (*.kst)

☑ Automatically select filename extension (.kst)

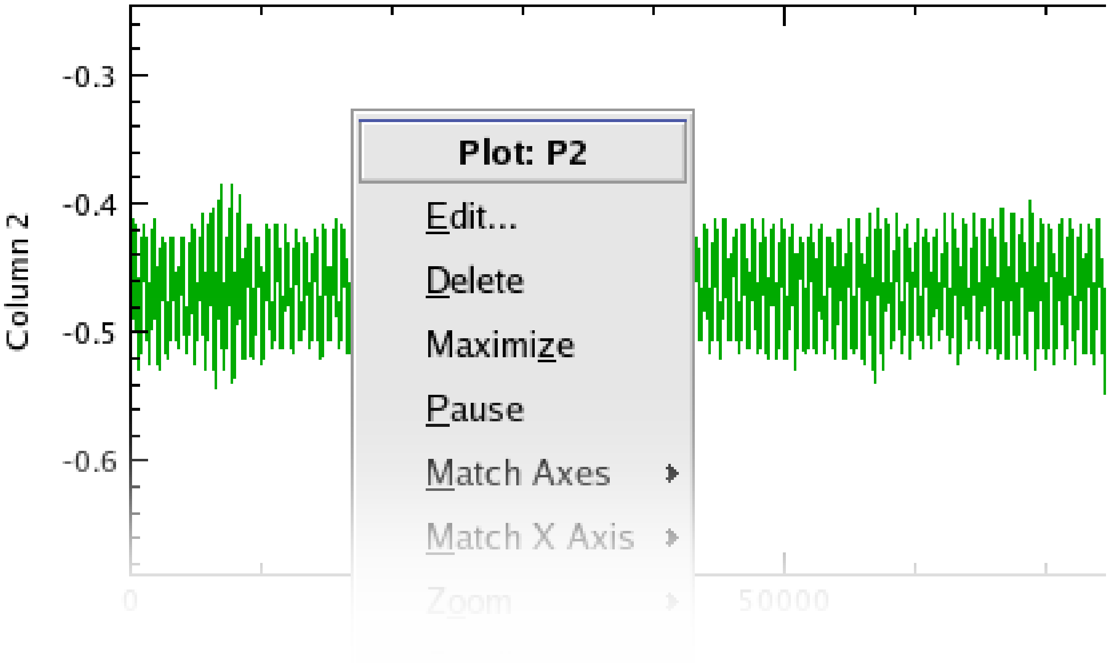
Saving the Kst session allows you to restore your plots later.

1.2.2 Plot Manipulation

Now that you are comfortable with creating plots from imported data in Kst, we can explore some of the plot manipulation features available through the graphical user interface. Start Kst from the command-line with the `mykstsession.kst` file you saved earlier:

```
kst mykstsession.kst
```

All the plots you created in the Importing Data section should now be loaded in Kst. Examine the plots with the y axis label of `Column 2`. To take a closer look at the plot, right click on it and select the Maximize menu item, as shown below:

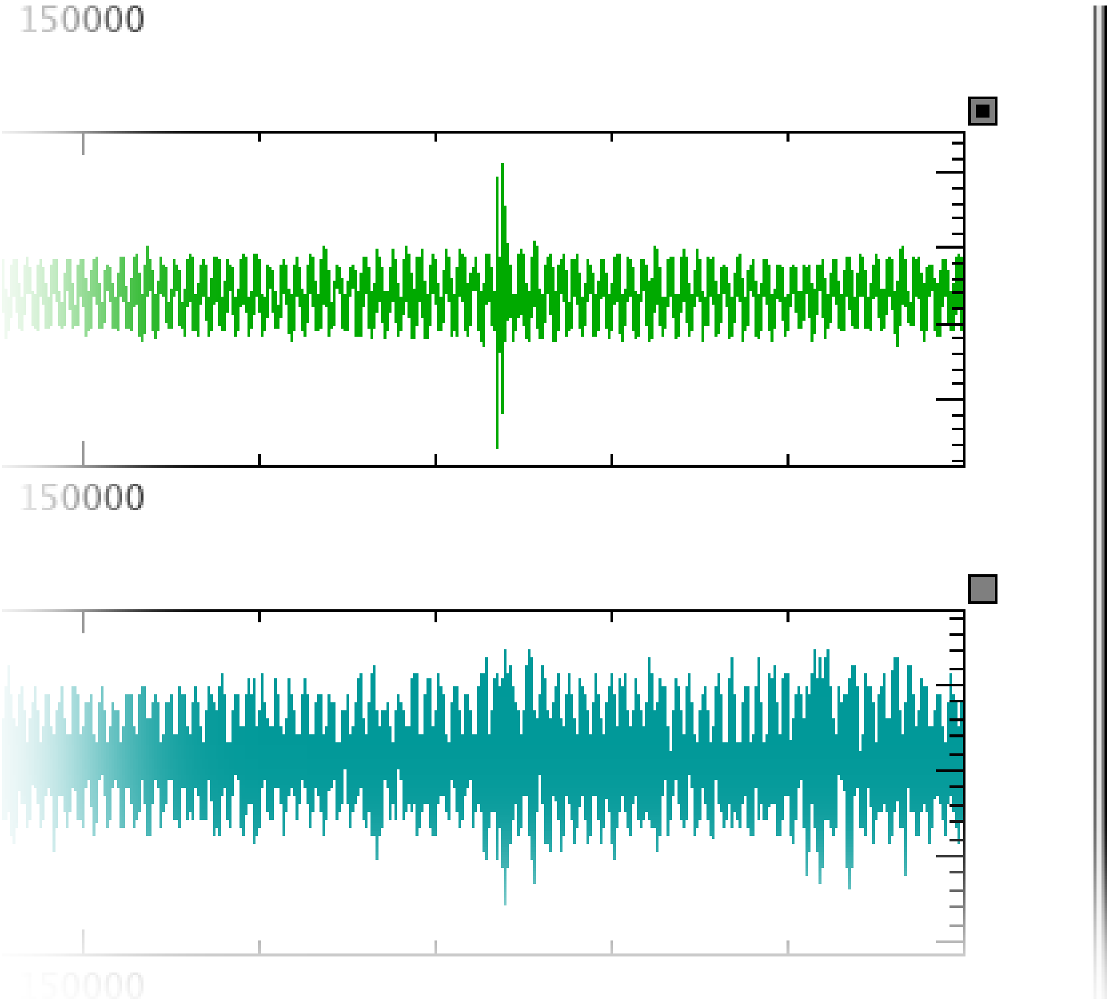


The plot you selected will now be maximized within the current window. Note that the data is still dense in this plot, so it would be useful to zoom in on an interesting area. To do so, make sure you are in XY Mouse Zoom mode (select XY Mouse Zoom from the Plots menu, or click the toolbar button). Then, simply drag a rectangle around the area you're interested in. Note that the coordinates of the mouse cursor are displayed in the lower right corner of the Kst window (if they are not, ensure that Show Statusbar is checked in the Settings menu).

The plot axes will change to 'zoom in' on the selected area of the plot. Suppose that you would like to view some data just outside the region you have zoomed into. This can be done with Kst's scrolling feature. Right-click on the plot and select Up, Down, Left, or Right from the Scroll submenu. The plot will scroll accordingly. Of course, it is usually easier to use the shortcut key associated with the menu item; this is true for most of the zooming and scrolling functions. The shortcut keys for scrolling are the Arrow keys, so the quickest way to scroll upwards would be to hold down the Up Arrow key. To return to maximum zoom at any time, right-click on the plot and select Zoom Maximum from the Zoom submenu (or type **M**, the shortcut key associated with Zoom Maximum).

Restore the size of the plot by right-clicking it and unchecking the Maximize option.

Now look at the plots with y axes labeled `Column 2` and `Column 3`. These are plots of the pitch gyro and roll gyro, respectively, from the 1998 BOOMERANG flight. Since these two plots are related, it can be useful to zoom and scroll them simultaneously. Click on the squares located at the top right corners of the two plots. They should turn gray in color to indicate that the two plots are now tied together:



Before we try zooming in, we should delete the ‘Column 1’ plot, which we won’t be working with any more. Right-click on this plot and select Delete. A hole will be left in the plotting window. We can remedy this by right-clicking anywhere inside the window and selecting Cleanup Layout → Default Tile. Now the two remaining plots should share maximal space inside the window. Return to XY Mouse Zoom mode when you are done.

Now try zooming in on any portion of the upper plot. You will find that the lower plot becomes blank. This is because the lower plot axes have changed to match the upper plot axes, and there is no data in that particular region of the lower plot. Type **M** while the mouse cursor is over either plot to return to maximum zoom on both plots. Now hold down **Ctrl** (this is equivalent to selecting X Mouse Zoom Mode from the Plots menu or clicking the toolbar button). The mouse cursor will change shape as visual feedback. While keeping **Ctrl** held down, drag a rectangle in the upper plot. Note that the height of the dotted rectangle is restricted so that only the x axis will be zoomed. Now both plots will display data when zoomed in, as the y axis for either plot was not changed.

Tip

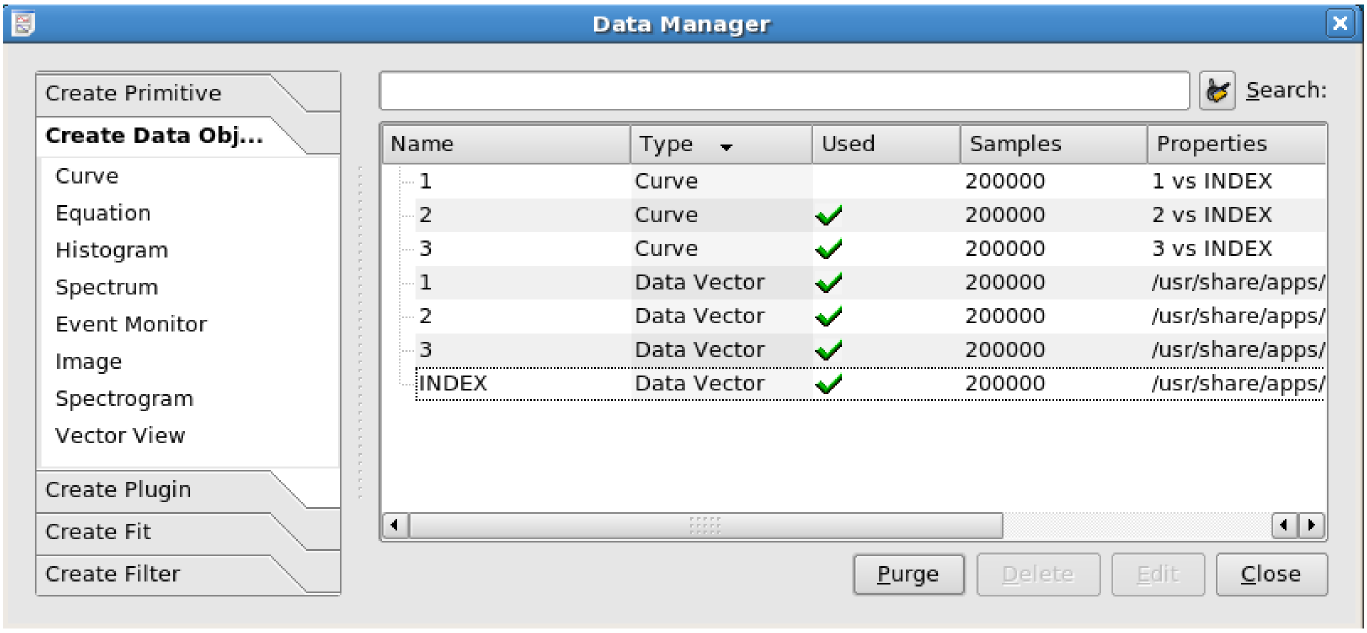
You can quickly tie or untie all the plots in the window by selecting Tied Zoom from the Plots menu or by clicking the toolbar button.

When you are finished experimenting with the zooming features, you can save the `mykstsession.kst` file again, with the changes which have been made.

1.2.3 Basic Analysis and the Data Manager

There is more to Kst than simple plotting and viewing of data. From the data wizard, you have already seen that Kst has the ability to create power spectra of data. In fact, Kst is also capable of creating other data objects such as histograms, equations, and plugins. A utility called the Data Manager can help you to do this.

Open the data manager now by clicking the toolbar button, or by selecting Data Manager from the Data menu.

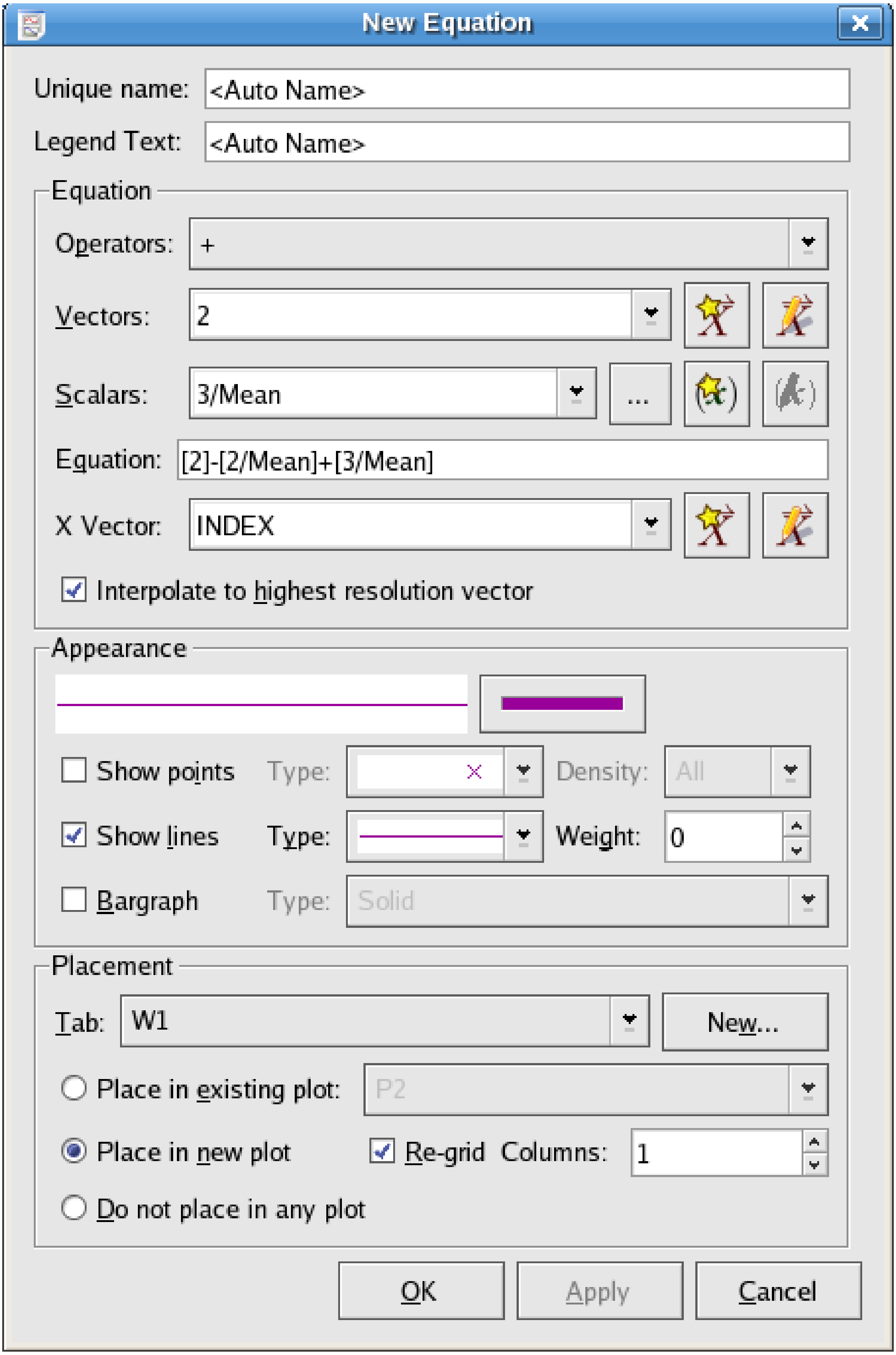


Name	Type	Used	Samples	Properties
1	Curve		200000	1 vs INDEX
2	Curve	✓	200000	2 vs INDEX
3	Curve	✓	200000	3 vs INDEX
1	Data Vector	✓	200000	/usr/share/apps/
2	Data Vector	✓	200000	/usr/share/apps/
3	Data Vector	✓	200000	/usr/share/apps/
INDEX	Data Vector	✓	200000	/usr/share/apps/

The Data Manager contains the definitive list of data objects in the current Kst session. It also allows you to edit or create new data objects. As you can see, there are currently 3 curves (each created from a pair of vectors) and four data vectors listed. A summary is given for each object; see the Data Manager section for more information.

We saw earlier that when we attempted tied-zoom in XY-Zoom-Mode that the Column 2 and Column 3 data have different y-offsets. Suppose that we would like to compare them on the same graph. To do this, we will have to shift the mean of one of the plots. We can do this easily using an Equation Object.

Begin the process of creating an Equation by clicking the corresponding button under the Create Data Object tab on the left side of the Data Manager. A dialog will appear to create the new object. Configure this dialog as shown below:



New Equation

Unique name: <Auto Name>

Legend Text: <Auto Name>

Equation

Operators: +

Vectors: 2

Scalars: 3/Mean

Equation: [2]-[2/Mean]+[3/Mean]

X Vector: INDEX

☑ Interpolate to highest resolution vector

Appearance

□ Show points Type: Density: All

☑ Show lines Type: Weight: 0

□ Bargraph Type: Solid

Placement

Tab: W1

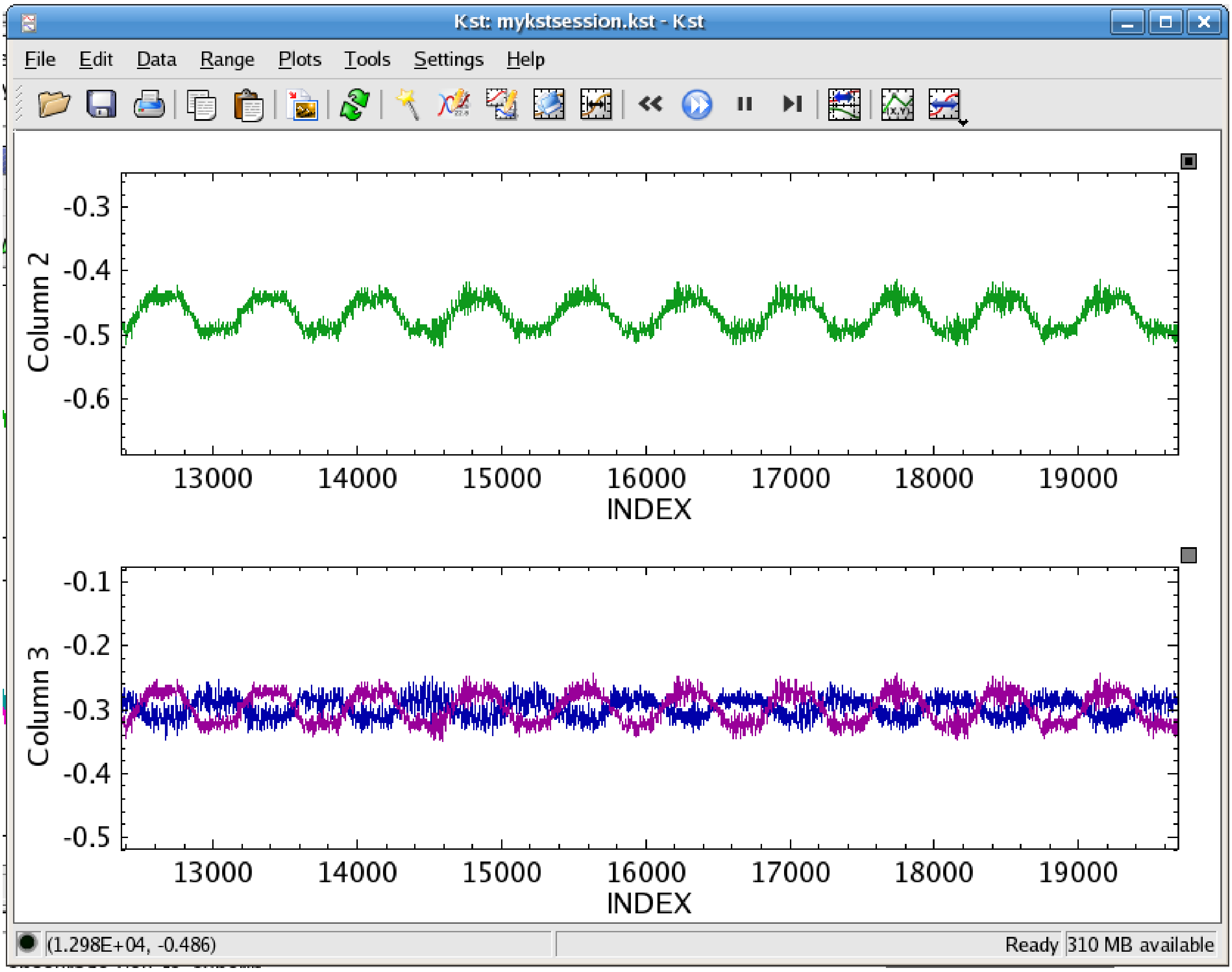
○ Place in existing plot: P2

● Place in new plot ☑ Re-grid Columns: 1

○ Do not place in any plot

This equation takes values from the data vector [2], subtracts the scalar [2/mean], then adds the scalar [3/mean] to make a new vector which is [2] shifted vertically to have the same mean as the data vector [3]. Note from the Scalars drop-down menu that Kst maintains a collection of simple statistics on all of its vectors (as well as many other objects). These can frequently prove useful.

Click the OK button of the New Equation dialog. Now a new curve has been added to the plot with y-axis label ‘Column 3’. If we return to the Kst plot window, we can zoom in on the ‘Column 3’ plot. Comparing the data, we can see what we suspected, that data vector [2] is essentially the same as [3], but exactly out of phase:



This concludes our basic introduction to Kst. We encourage you to experiment with the program now; there is clearly a lot of functionality in Kst which hasn’t been covered in this tutorial. Most of it can be found intuitively. You may want to look at some of the entries in the Common Tasks chapter. Good luck!

Chapter 2

Common Tasks

This chapter contains instructions for performing basic tasks with Kst. It runs through the method for performing each task in the simplest way. Links to the sections of the handbook with more in-depth information are also given.

2.1 Plotting Simple Graphs

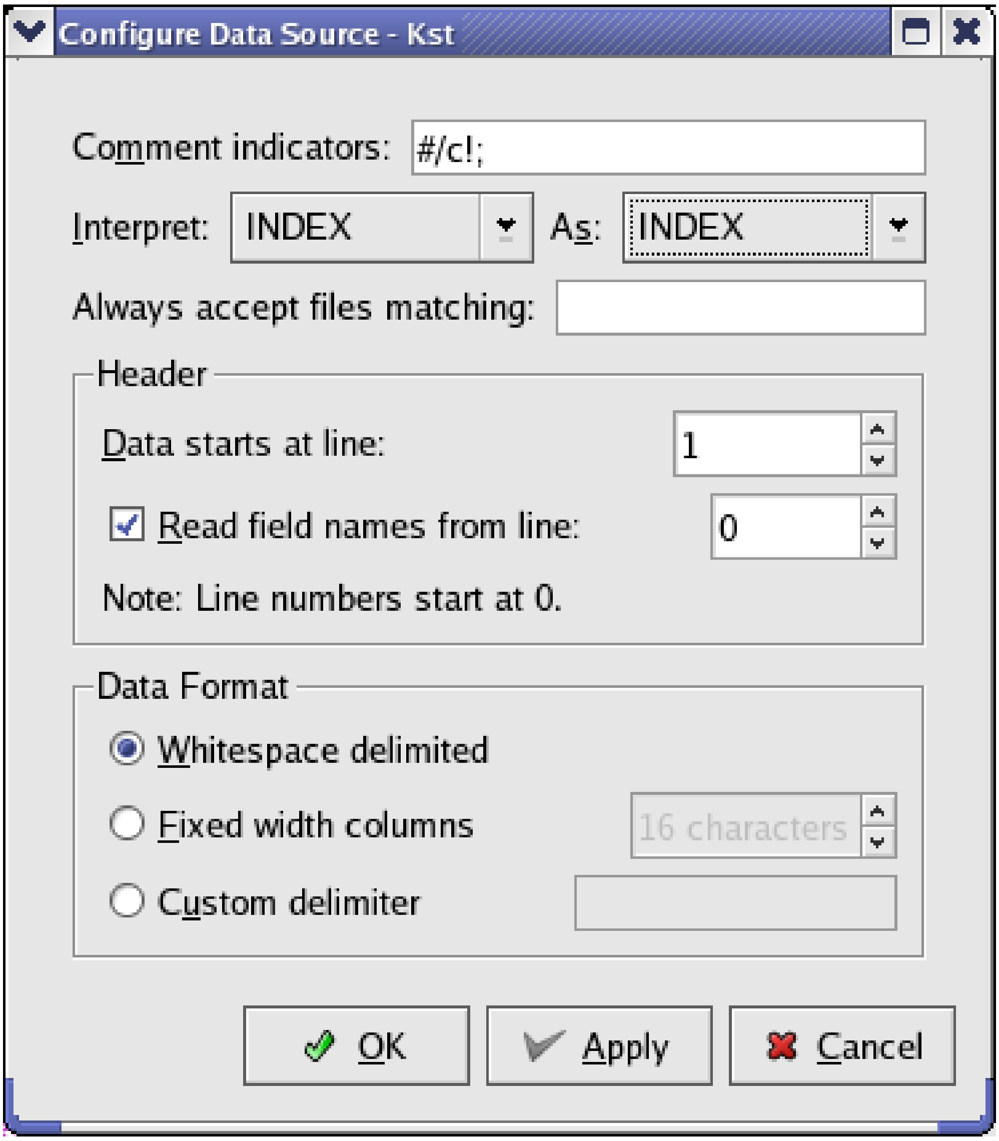
The first thing you'll want to do with Kst is probably to plot a simple graph. Here's the procedure:

1. Choose your data file. Kst will read many types of data file, the simplest being a plain text (ASCII) file with the data in columns. For this tutorial, we'll create a simple data file. Copy and paste this simple 10-line set of data into a text file, and save it as `simple.dat`:

```
x  y
-5 6.25
-4 4
-3 2.25
-2 1
-1 0.25
0 0
1 0.25
2 1
3 2.25
4 4
5 6.25
```

> **Note**
> ASCII is only one of the many file types Kst is capable of reading. For further details on file types, see the Datasources section.

2. Open Kst. If you are presented with the QuickStart dialog, then click on the Data Wizard button. If the QuickStart dialog does not appear, then you can click the 🪄 toolbar button to open the Data Wizard. In the wizard that appears, you can enter the pathname of the simple.dat file manually, or select it using the Open File icon 📂 which appears next to the path entry textbox. We have put x and y labels on the first line of our data file, so click the Configure... button and set up the data source as shown below.



Configure Data Source - Kst

Comment indicators: #/c!;

Interpret: INDEX As: INDEX

Always accept files matching:

Header

Data starts at line: 1

☑ Read field names from line: 0

Note: Line numbers start at 0.

Data Format

● Whitespace delimited

○ Fixed width columns 16 characters

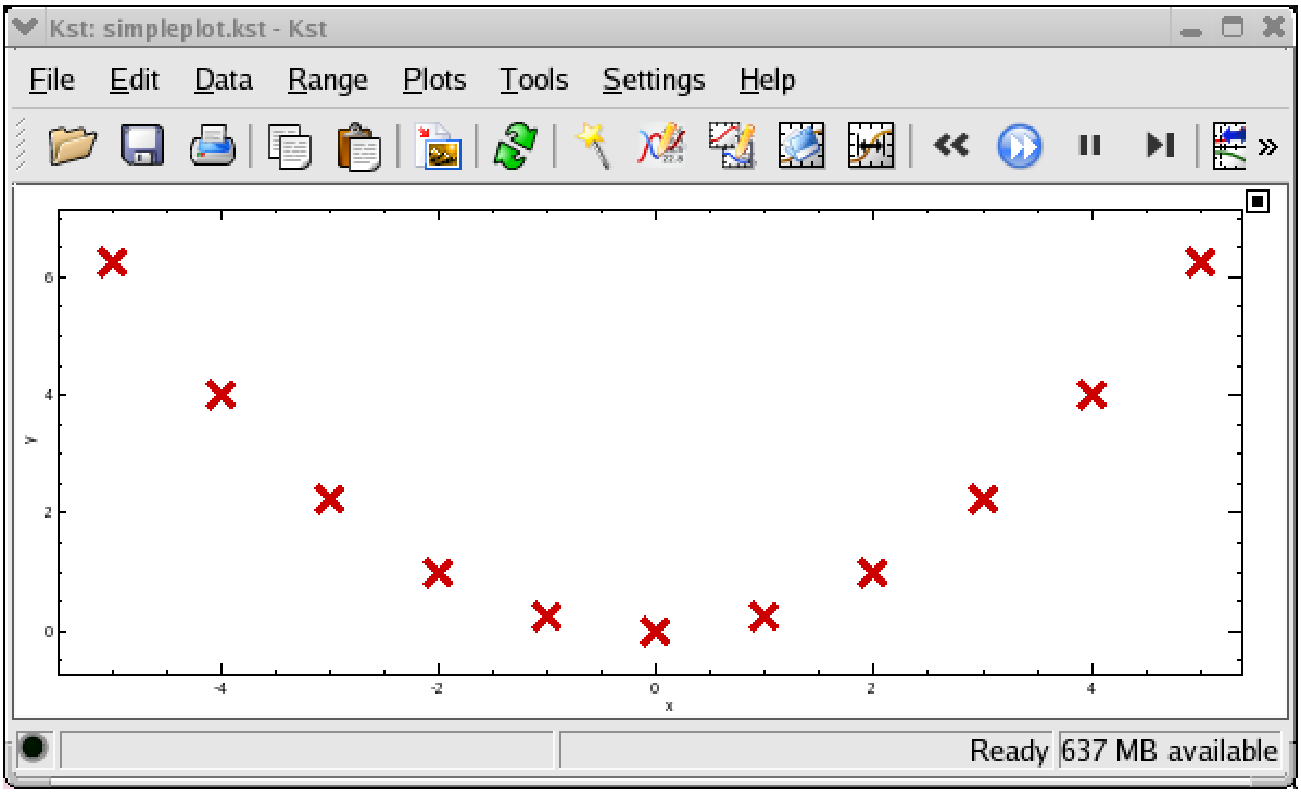
○ Custom delimiter

3. The Next page is where you select which data to use. If you've configured the ASCII datasource correctly, the field names will have been read from the first entry of the datafile and ‘x’ and ‘y’ will be entries in the Available list on the left-hand side of the page. Move the ‘y’ field to theSelected list using the arrow buttons in the center of the window. The INDEX field is a special field created by Kst; it can be used as X-axis data when the data source does not provide a data vector for the X-axis.

4. Clicking the Next button again allows you to select Data Presentation options. Under Plot Types, leave the default XY as it is, but change the X Axis Vector to Create From Field: x.

5. On the Next page, ’Plot Layout’, we'll just change one thing: under Curve Style, select Points only to draw just the data points, and not the lines joining them.

6. Click on Finish and you're done! The new plot appears in the main Kst window.



Tip
It's also possible to quickly create plots from the command line using Kst, similar to Gnuplot. In the case of the `simple.dat` file, we can produce a simple plot using the first column (or 'field') of the file as the x vector, with the second column as a y vector, by entering the following at the command line:

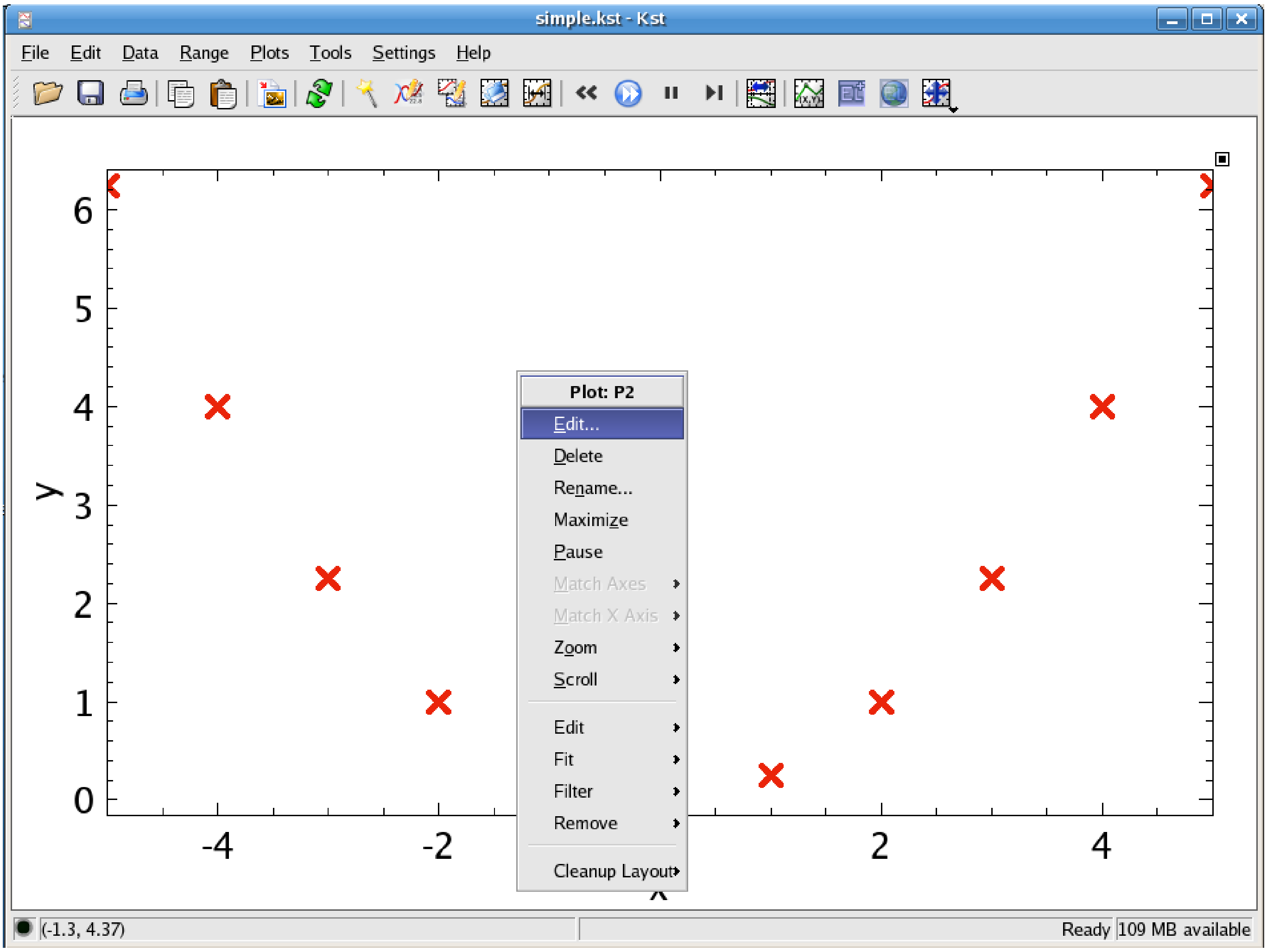
```
kst -x 1 -y 2 simple.dat
```

Numerous other command-line options are available — using the command-line only, plots of data can be printed directly to files, and simple data manipulation such as creation of power spectra can be performed as well. For a complete list of command-line options, see Command Line Usage and Examples.

2.2 Customizing Plot Settings

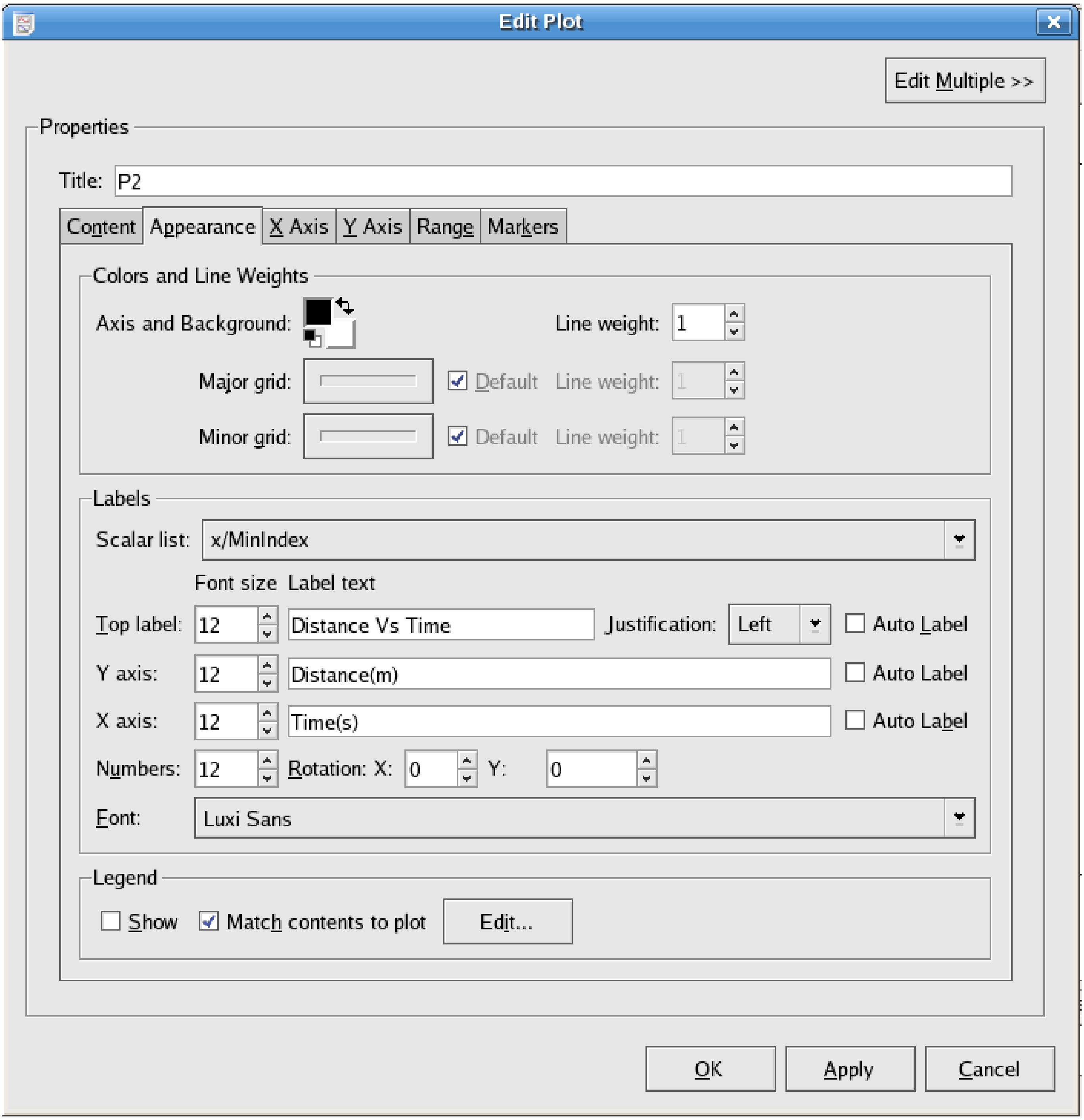
For the previous curve plot, suppose you want to change the default x and y labels, and add a title for it.

To change the default setting for the plot, first, right click anywhere on the plot, a context menu will appear as below.



Choose Edit..., and the Edit Plot dialog will appear.

To add or change the Y/X label and title of the plot, click on the Appearence tab.



Title: P2

Axis and Background: ■ □

Line weight: 1

Major grid: ☑ Default Line weight: 1

Minor grid: ☑ Default Line weight: 1

Scalar list: x/MinIndex

	Font size	Label text		
Top label:	12	Distance Vs Time	Justification: Left	□ Auto Label
Y axis:	12	Distance(m)		□ Auto Label
X axis:	12	Time(s)		□ Auto Label

Numbers: 12 Rotation: X: 0 Y: 0

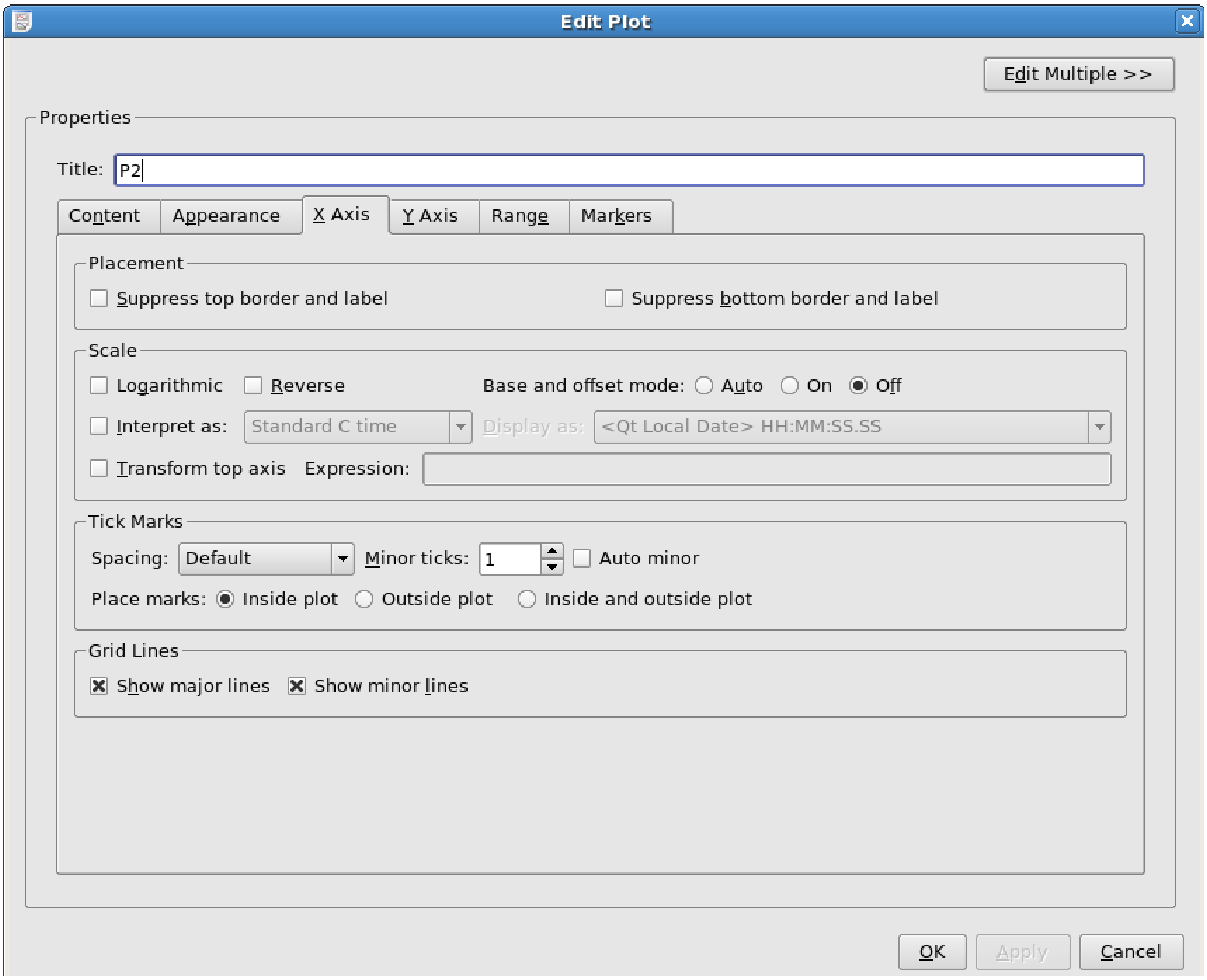
Font: Luxi Sans

□ Show ☑ Match contents to plot

In the textbox beside Y axis/X axis: enter the Y axis/X axis label you want. In this case, ‘Distance(m)’ and ‘Time(s)’.

In the textbox beside Top label: enter the title for the plot. In this case: ‘Distance vs Time’. In addition, you can choose the justification of the title by selecting Left, Right or Center in the Justification textbox.

To change the settings of the X or Y axis, such as choosing the scale of the X/Y axis or adding gridlines, click on the X Axis or Y Axis tab.



Edit Plot

Edit Multiple >>

Properties

Title: P2

Content | Appearance | X Axis | Y Axis | Range | Markers

Placement

□ Suppress top border and label □ Suppress bottom border and label

Scale

□ Logarithmic □ Reverse Base and offset mode: ○ Auto ○ On ● Off

□ Interpret as: Standard C time Display as: <Qt Local Date> HH:MM:SS.SS

□ Transform top axis Expression:

Tick Marks

Spacing: Default Minor ticks: 1 □ Auto minor

Place marks: ● Inside plot ○ Outside plot ○ Inside and outside plot

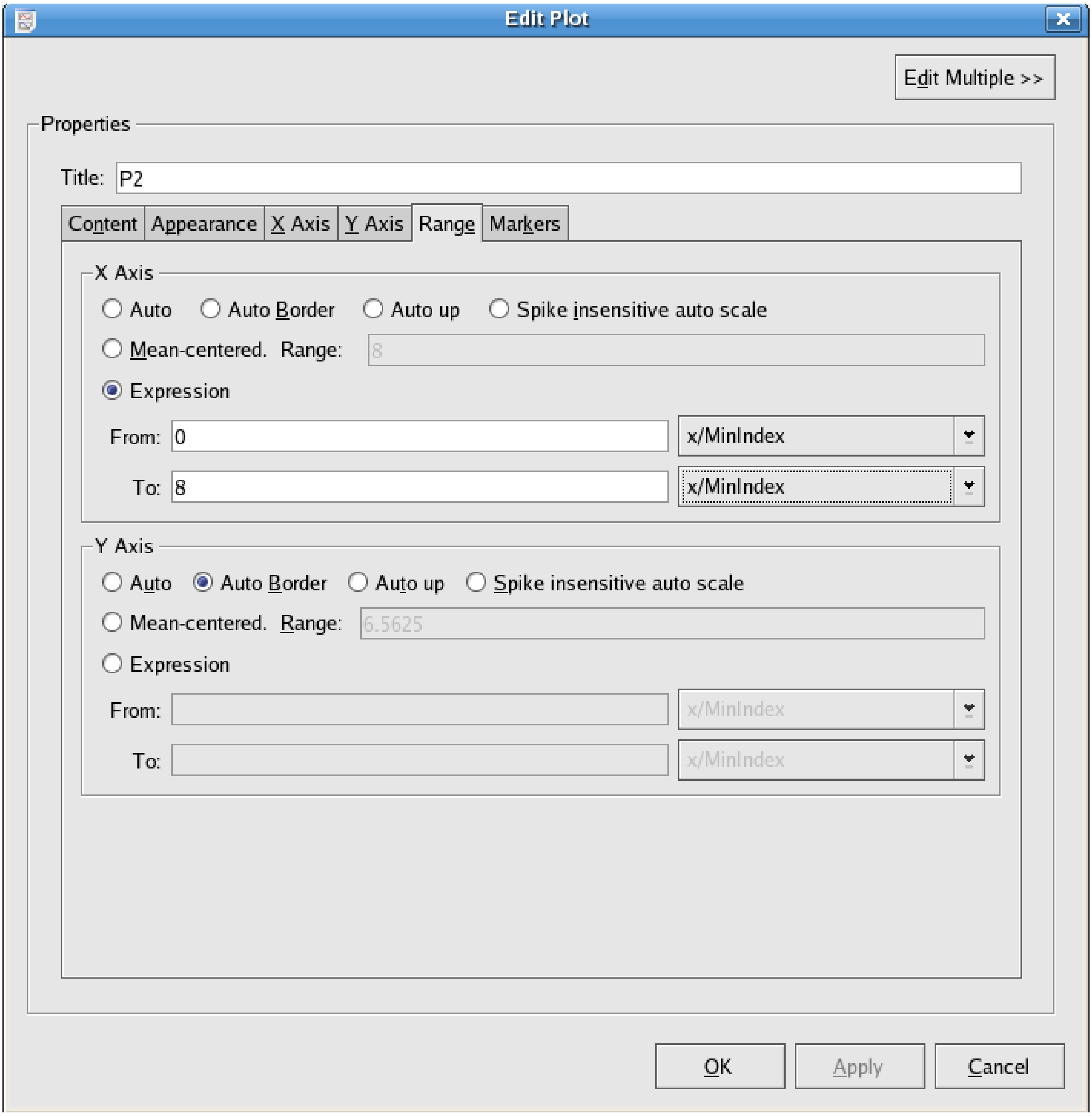
Grid Lines

☒ Show major lines ☒ Show minor lines

Change the minor ticks of the X axis to be 1, and choose to show both the major and minor gridlines for the X axis.

For the Y axis, for which no screenshot is shown, select Reverse for the Scale, change the Minor ticks to be 1.

To change the range of X or Y axis, choose the Range tab.



Edit Plot

Edit Multiple >>

Properties

Title: P2

Content | Appearance | X Axis | Y Axis | Range | Markers

X Axis

○ Auto ○ Auto Border ○ Auto up ○ Spike insensitive auto scale

○ Mean-centered. Range: 8

● Expression

From: 0 x/MinIndex

To: 8 x/MinIndex

Y Axis

○ Auto ● Auto Border ○ Auto up ○ Spike insensitive auto scale

○ Mean-centered. Range: 6.5625

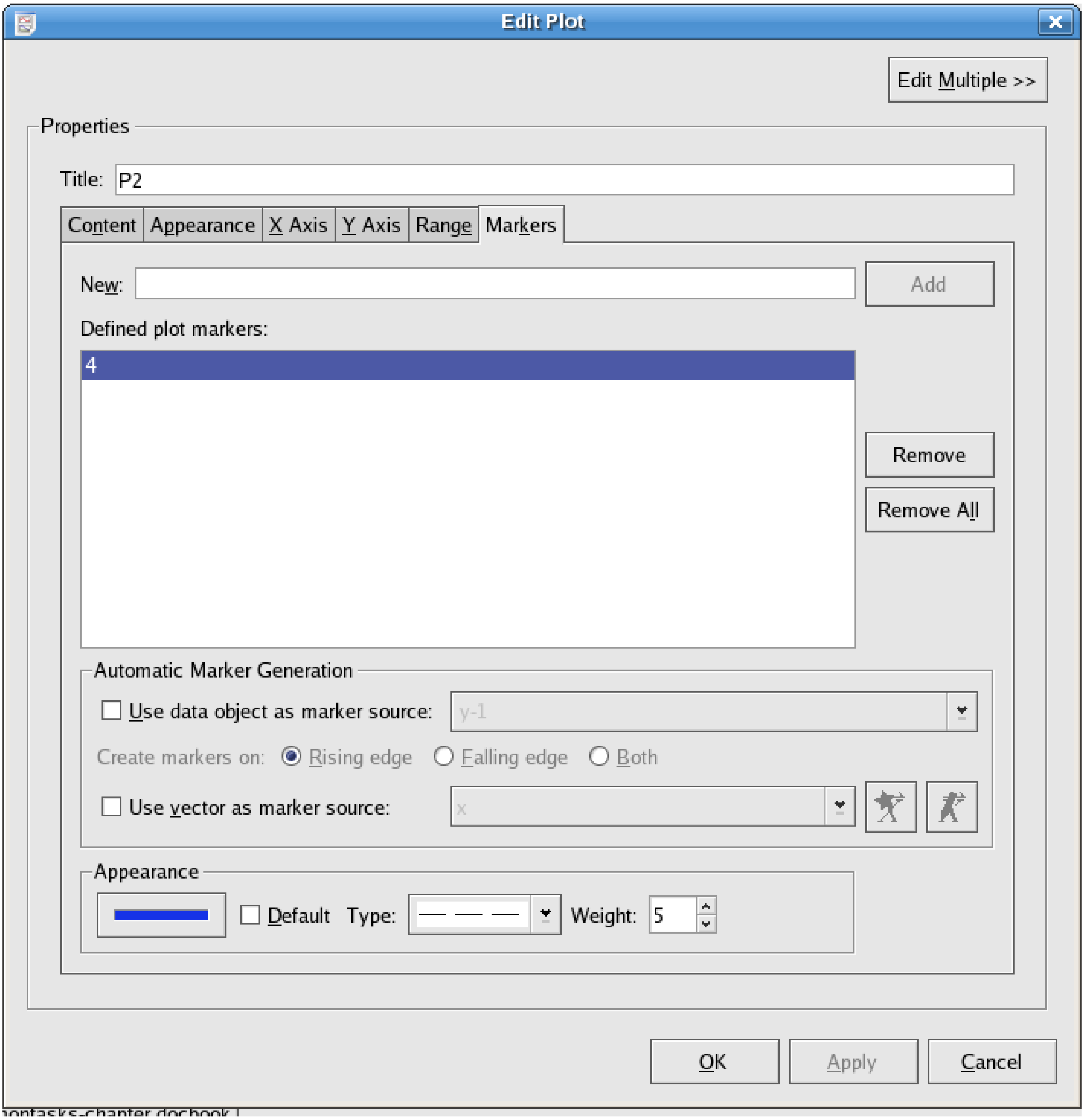
○ Expression

From: x/MinIndex

To: x/MinIndex

Change the range of x axis to be From 0 To 8.

Finally, you can highlight some important part of the plot by using the Markers tab.



Edit Plot

Edit Multiple >>

Properties

Title: P2

New:

Defined plot markers:

4

Automatic Marker Generation

□ Use data object as marker source: y-1

Create markers on: ● Rising edge ○ Falling edge ○ Both

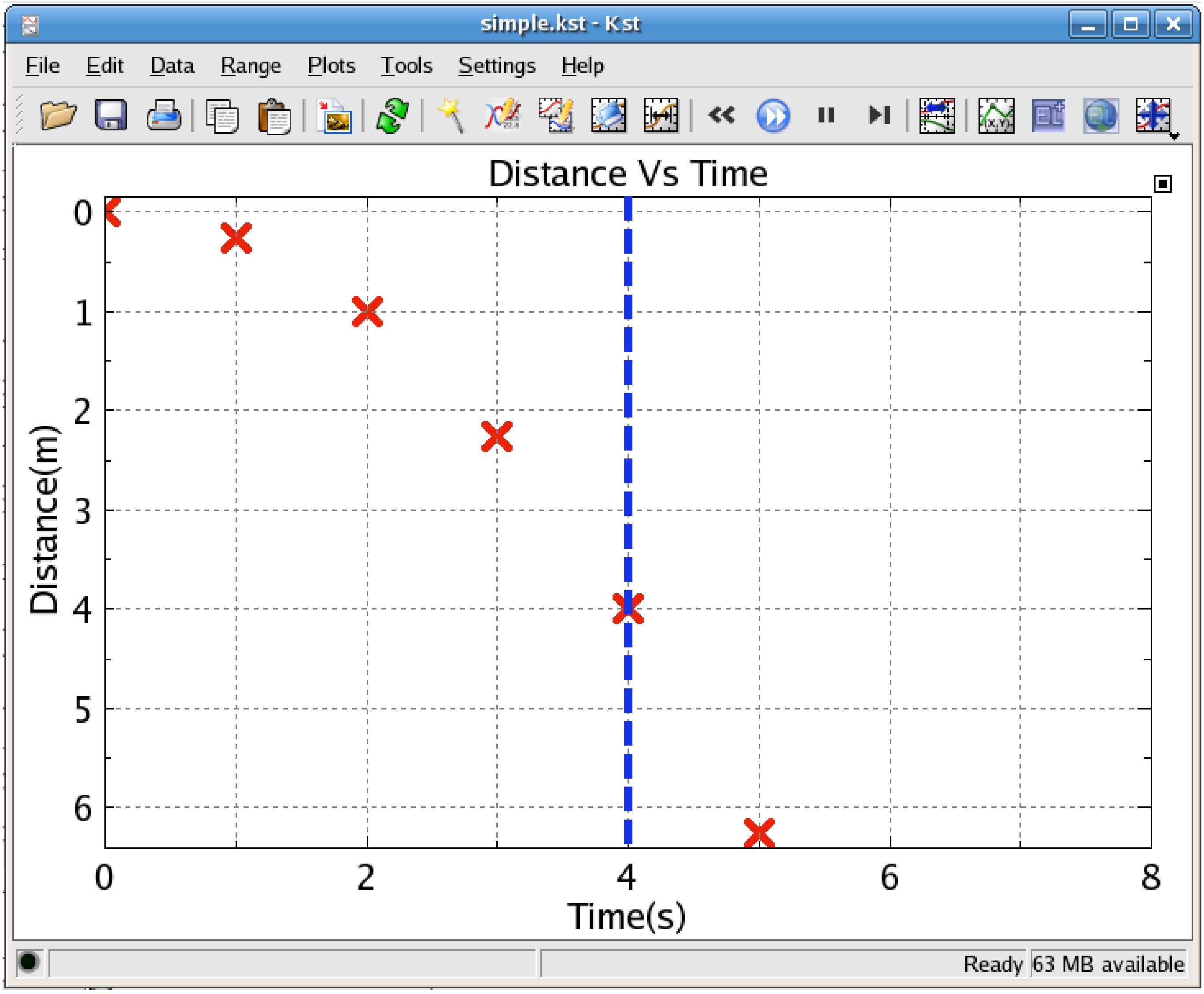
□ Use vector as marker source: x

Appearance

□ Default Type: — — — Weight: 5

For example, if time = 4 s is important, enter ‘4’ in the New textbox and click on the Add button. A marker line will appear at the position of time = 4. You can customize the marker line in Appearance. In this example, choose ‘blue’ for the line color, ‘broken line’ for the Type and ‘5’ for the Weight.

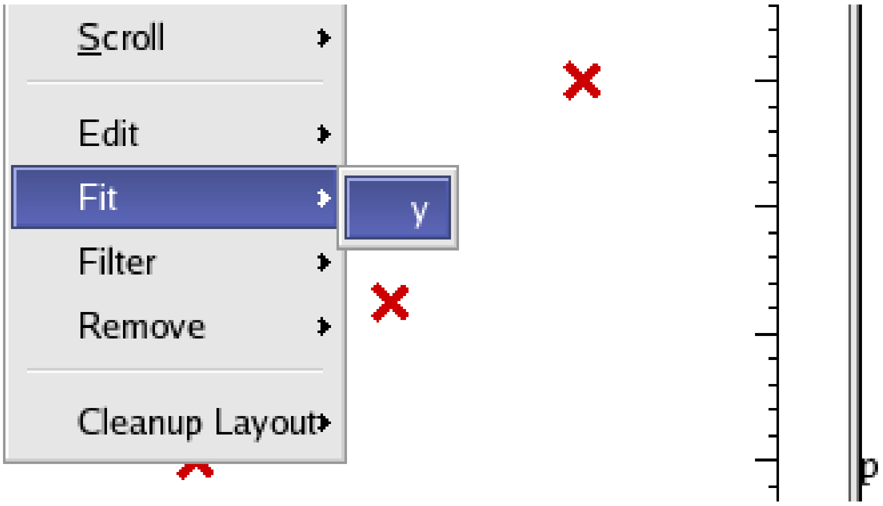
The customized plot will look like this:



2.3 Curve Fitting

Kst also provides functions for many different types of curve fitting: linear, quadratic, sinusoidal, and more. As an example, we'll do a simple polynomial fit to the curve we plotted previously, in the Plotting Simple Graphs section.

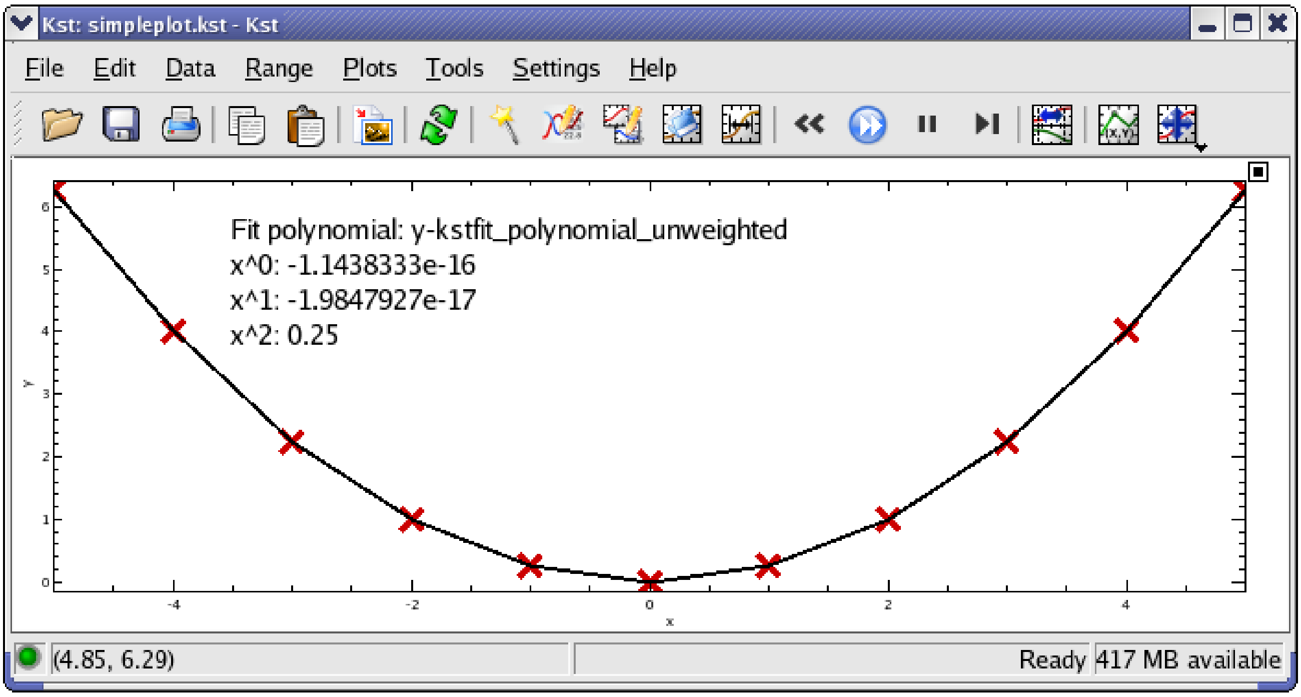
1. Right click on the 'simple graphs' plot which you've created and select 'y' from the Fit submenu in the context menu which appears.



2. Now a ‘New Plugin’ dialog will appear which allows you to configure the fit. Since the example data here are quadratic, we’ll use the Fit polynomial plugin. Choose this from the Plugin Selection drop-list. As it happens, the Order of the polynomial is set to 2 by default, which is what we want. You can set the Order to any integer, one of the predefined scalar quantities, or some statistical value based on one of the available vectors (e.g. the Standard Deviation of x, [x/Sigma]).

   There are several Output fields where you can enter text to be used as the unique name for the vectors and scalars generated by this fit. For more information, you can read the section on the Fit Polynomial in the Plugins chapter.

3. Hit OK to perform the fit. A new curve showing the best fit polynomial will be automatically added to the plot containing the original curve. A new label object is also added to the plot containing the fitted coefficients for xˆ2, xˆ1 and xˆ0 (the constant).


Kst: simpleplot.kst - Kst

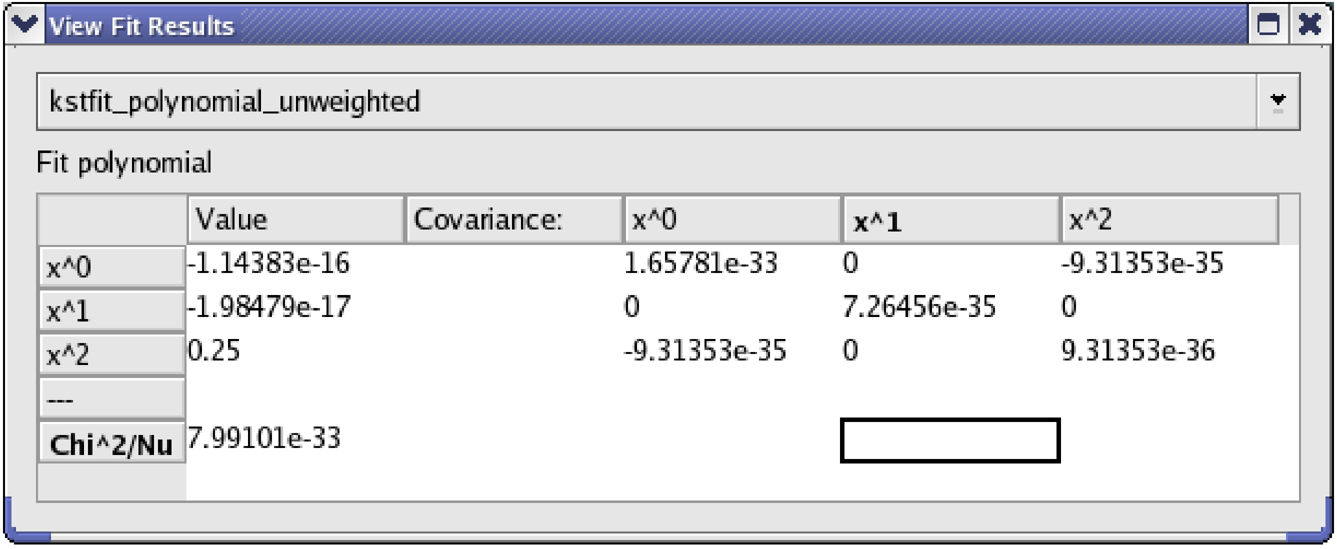
Fit polynomial: y-kstfit_polynomial_unweighted
x^0: -1.1438333e-16
x^1: -1.9847927e-17
x^2: 0.25

(4.85, 6.29)

Ready 417 MB available

Tip

Not all of the fit results are shown in the new label. The covariances for the fit coefficients, as well as the Chi-Square value for the fit can be accessed from the View Fit Results dialog. Select Data → View Fit Results from the main menu, and choose the fit which you are interested in from the drop down list. The results are shown in a table:



Fit polynomial

	Value	Covariance:	x^0	x^1	x^2
x^0	-1.14383e-16		1.65781e-33	0	-9.31353e-35
x^1	-1.98479e-17		0	7.26456e-35	0
x^2	0.25		-9.31353e-35	0	9.31353e-36

Chi^2/Nu	7.99101e-33				

2.4 Generating a Histogram

Kst contains built in routines to generate histograms from data. For this tutorial, we’ll look at the histogram of a vector containing normally distributed numbers.

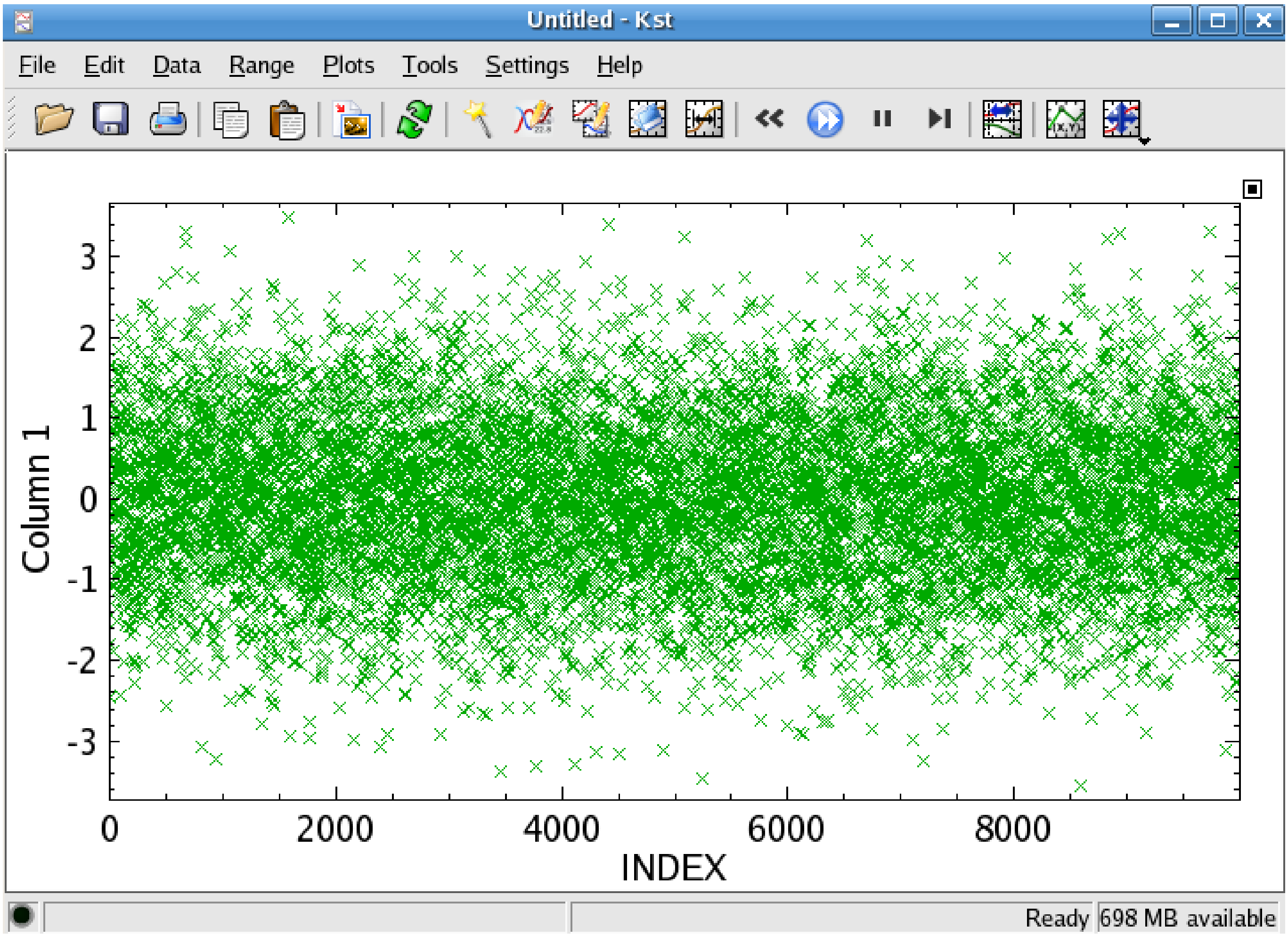
We’ll generate this vector using a perl script. Copy the following code into a file called generate_normal.pl:

```
sub gaussian_rand { # from the Perl Cookbook. (O’Reilly, 1998)
    my ($u1, $u2);  # uniformly distributed random numbers
    my $w;          # variance, then a weight
    my ($g1, $g2);  # gaussian-distributed numbers

    do {
        $u1 = 2 * rand() - 1;
        $u2 = 2 * rand() - 1;
        $w = $u1*$u1 + $u2*$u2;
    } while ( $w >= 1 );

    $w = sqrt( (-2 * log($w))  / $w );
    $g2 = $u1 * $w;
    $g1 = $u2 * $w;
    # return both if wanted, else just one
    return wantarray ? ($g1, $g2) : $g1;
}

srand(time() ^($$ + ($$ <<15))) ; # initialize the RNG.
for ($x=0;$x < 10000;$x++) {
        print gaussian_rand() . "\n";
}
```

Now execute the script with the command:

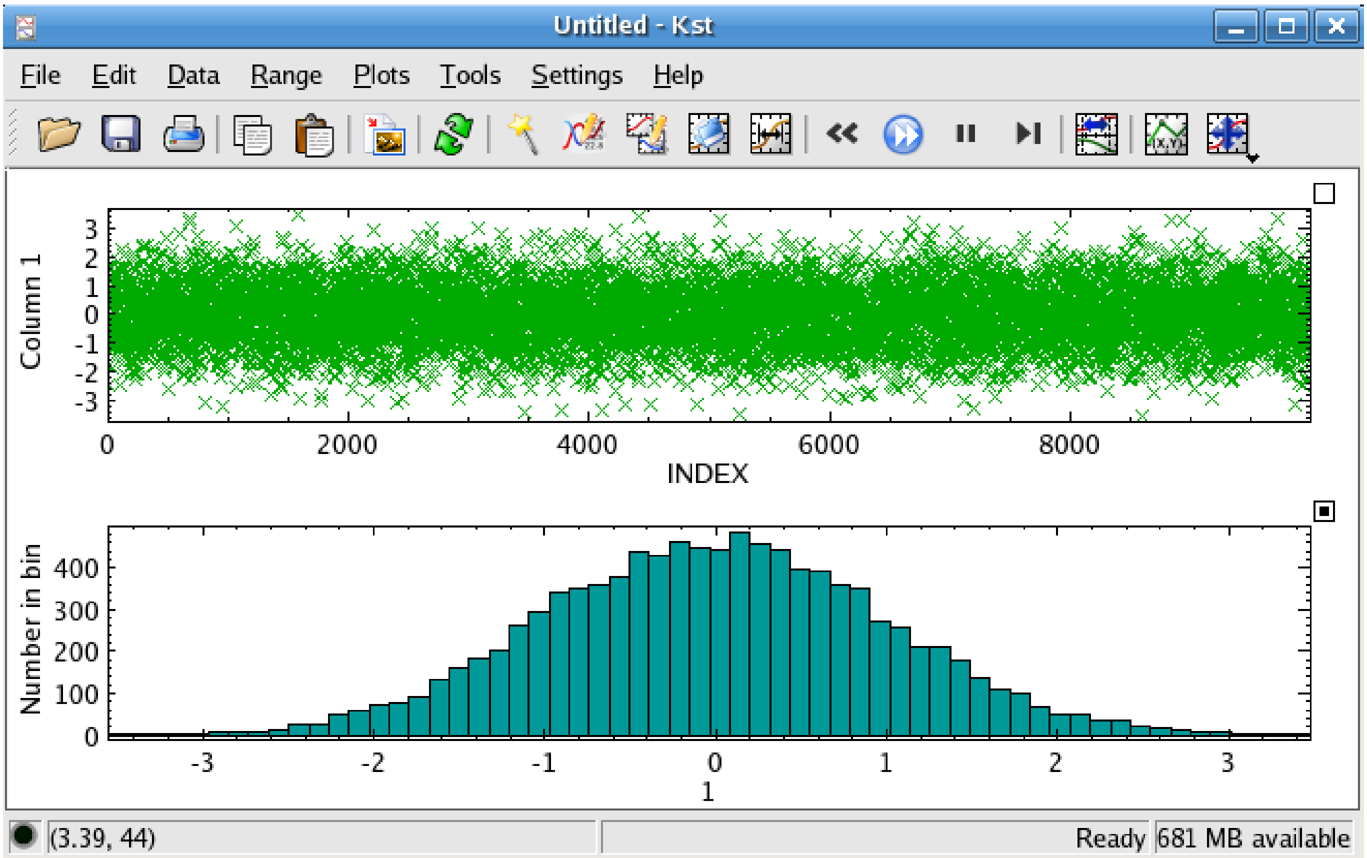
```
perl generate_normal.pl > normal.dat
```

This will generate an ASCII file `normal.dat` containing random numbers with a gaussian distribution centered at 0 with a standard deviation of 1. Import this data into Kst, and plot it as points with INDEX as the x-axis vector (if this gives you trouble, you may want to see the Getting Started section on Importing Data. You should see something which resembles the following:



Now we are ready to generate the histogram itself. Select Data → New Histogram.... Select the Data Vector which has just been created from `normal.dat`. There are several options relating to bin choices. For quick generation of histograms, it is usually easiest to just press the Auto-Bin feature. This will select an optimal bin size and range for the given data.

Click the OK button to generate the histogram.



2.5 Creating an Event Monitor

Kst has many features which are oriented toward real-time data. One of the most useful of these features is the "Event Monitor" data object. Event monitors check conditions on data vectors. A common use for them is to provide warnings about unusual features in incoming data. They can provide feedback in a number of ways:

- Messages in the Kst debug log.
- Automated Electronic Logbook (ELOG) entries.
- Email notification.

In this tutorial, we will create an event monitor to provide debug log notification about statistically unlikely spikes in a data stream. More information about Event Monitor data objects can be found here

First we’ll need a script to generate some simulated realtime data. Copy the following into a file called `simrealtime.pl`

```
#!/usr/bin/perl

open(OUTPUT, ">$ARGV[0]") || die "Can’t open $ARGV[0]: $!";

srand(1); # (badly) initialize the RNG.

for ($x=0;;$x++) {
        open(OUTPUT, ">>$ARGV[0]") || die "Can’t open $ARGV[0]: $!";
        $rn = rand();
        if ($rn > .99) {
                $rn = $rn + 5;
        } else {
                $rn = $rn - .5;
        }
        print OUTPUT $rn . "\n";
        close(OUTPUT);
        print "Created " . $x . " entries. Press Enter to Continue ...";
        $dummy=<STDIN>
}
```

This perl script will generate random data, uniformly distributed in the range [-0.5, 0.5], with occasional spikes. Make the script executable and run it using the commands...

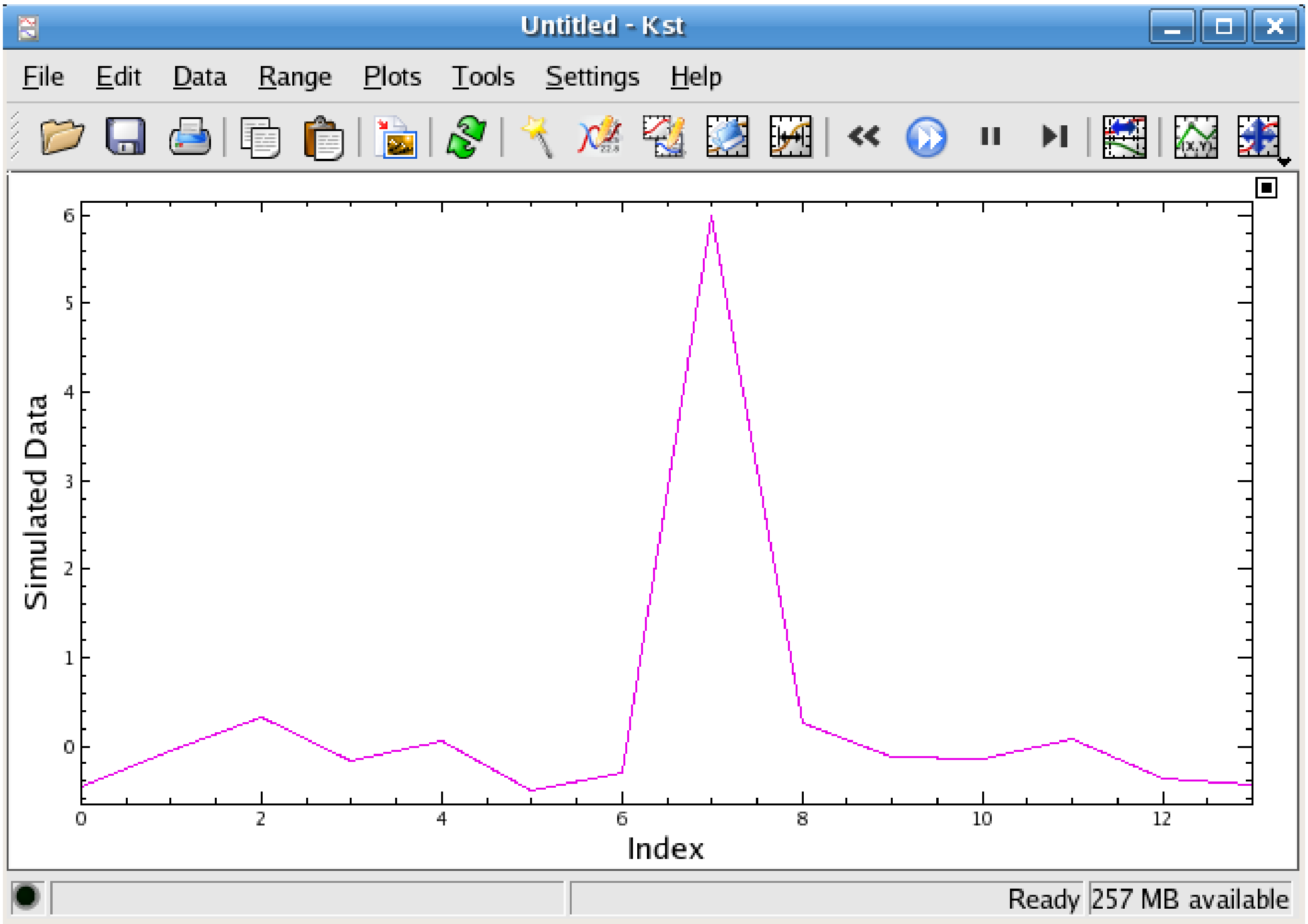
`chmod a+x simrealtime.pl`

`./simrealtime.pl simulated.dat`

In order to simulate ‘realtime’ data, the script will not generate all of its output at once, but will prompt the user before it outputs each entry. Press the enter key several times to get the data set started. Now, in a separate terminal, instruct Kst to plot the data using the command...

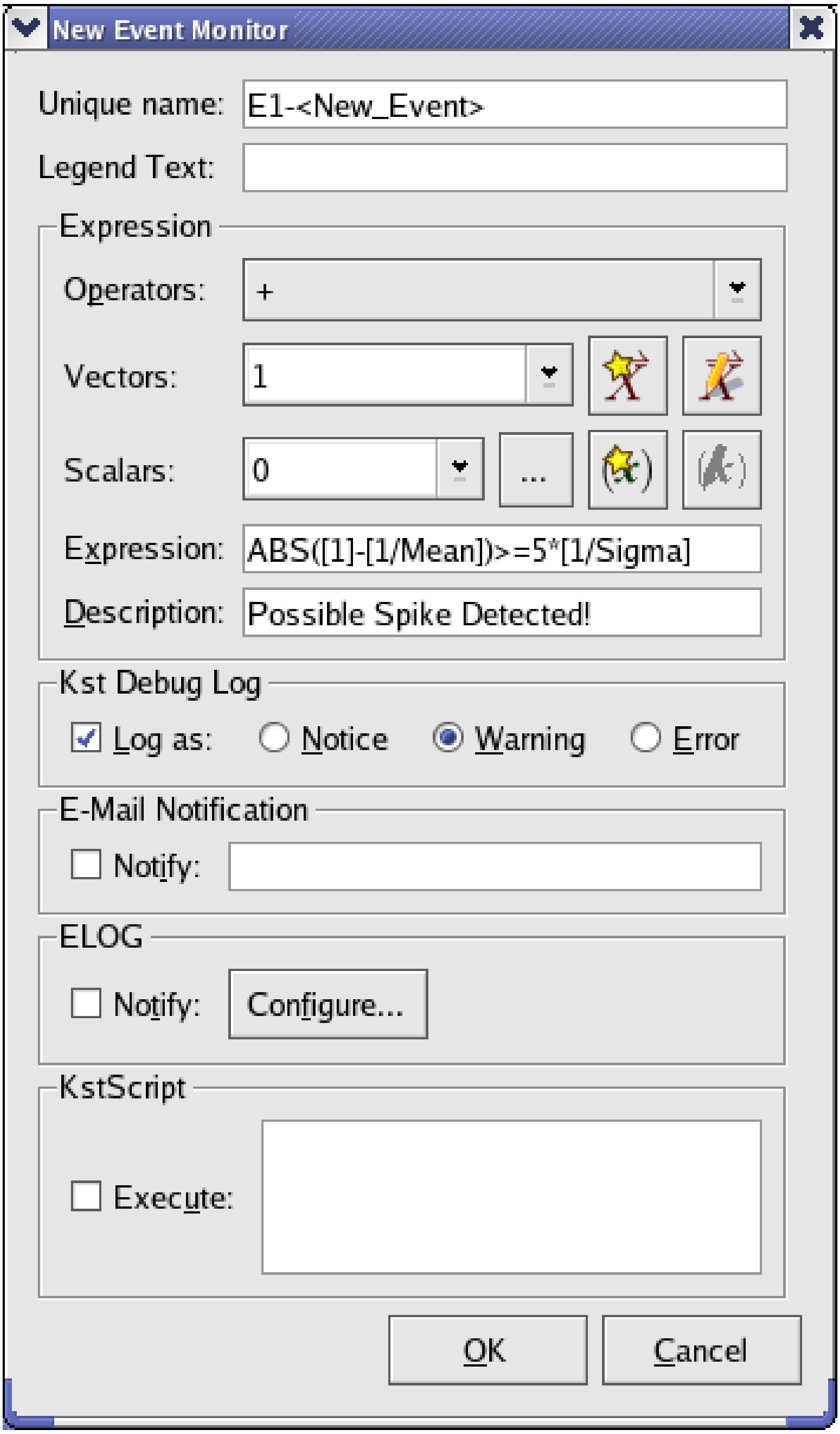
`kst -y 1 simulated.dat`

Depending on the number of data points which you have generated so far, your plot should look something like the graph below, possibly without the large spike in the middle.



Now, return to the terminal where the `simrealtime.pl` script is running and press the enter key a few more times. You should see the plot you’ve made in Kst update automatically as new data is added to the `simulated.dat` stream.

Suppose that we would like to receive notification every time a large spike occurs in our incoming data stream. This can be done using an event monitor. Select the menu-item Data → New Event Monitor..., and configure the new event monitor’s settings as shown below.



New Event Monitor

Unique name: E1-<New_Event>

Legend Text:

Expression

Operators: +

Vectors: 1

Scalars: 0

Expression: ABS([1]-[1/Mean])>=5*[1/Sigma]

Description: Possible Spike Detected!

Kst Debug Log

☑ Log as: ○ Notice ● Warning ○ Error

E-Mail Notification

□ Notify:

ELOG

□ Notify: Configure...

KstScript

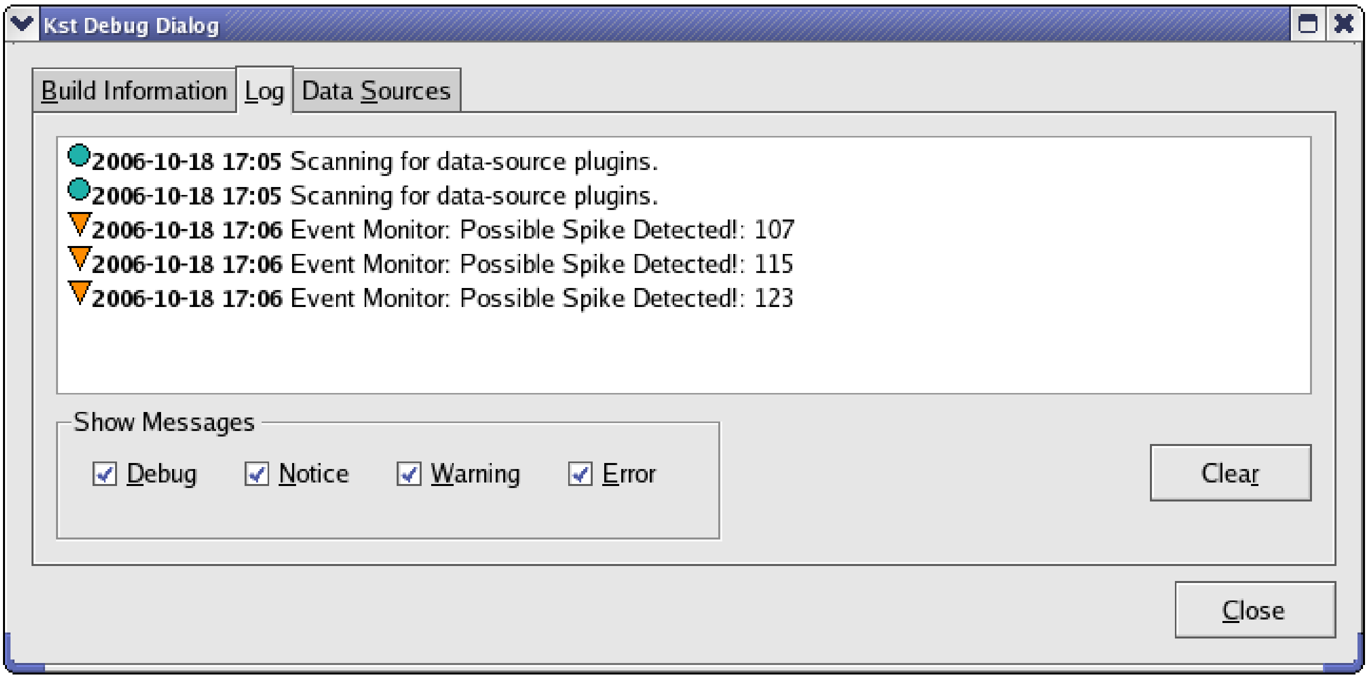
□ Execute:

OK Cancel

The settings in the Expression text box configure the event monitor to report entries in the vector ‘1’ which are more than 5 standard deviations from its mean. This is a fairly simple event monitor. Much more powerful event monitors can be created using the large collection of scalars which Kst maintains, and the many operators which can make comparisons between vectors.

We will have our events logged as ‘Warnings’ in the Kst Debug Log. As you can see, there are several other possible event reporting methods. These are explained in more detail in the documentation for Event Monitors.

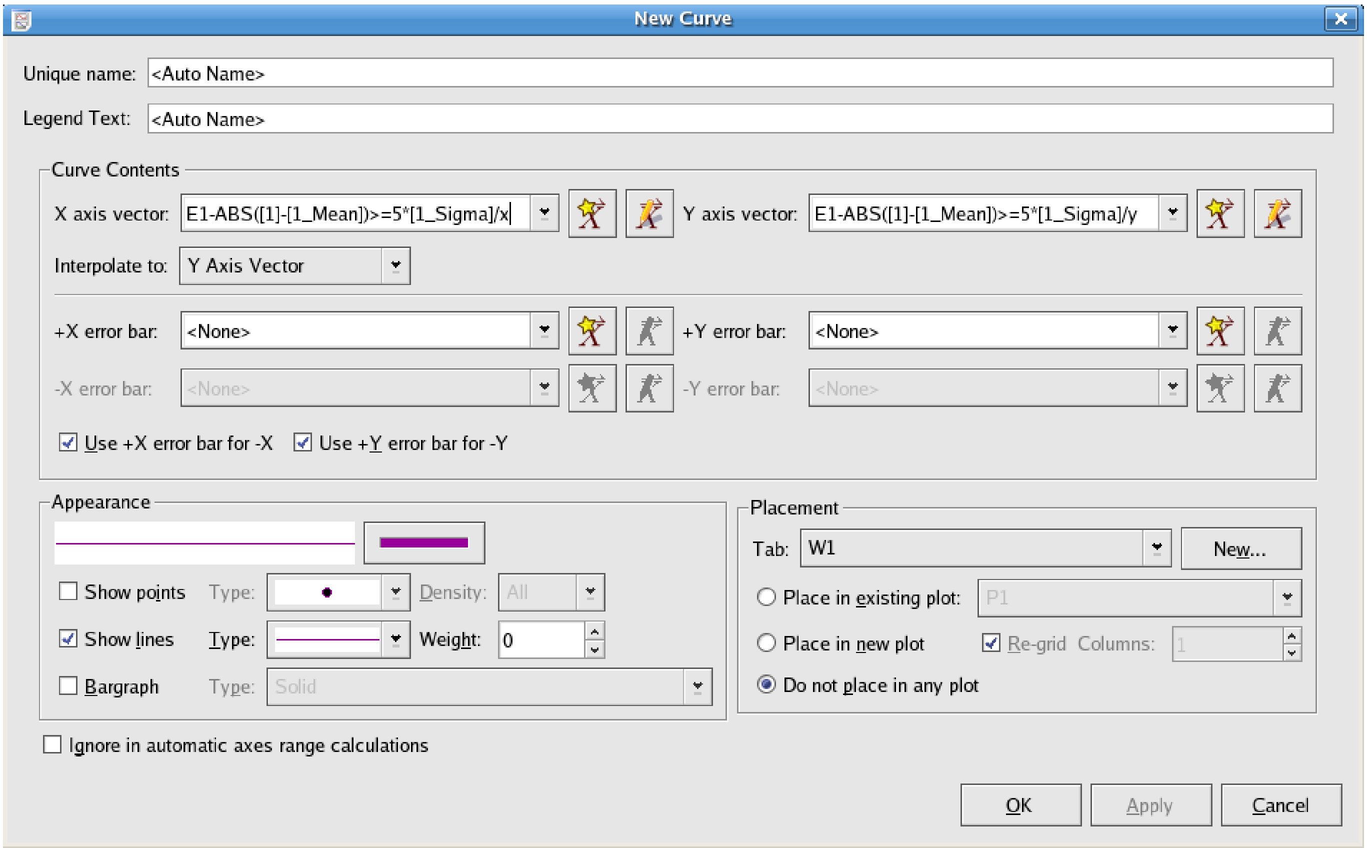
Click the OK button to create the new event monitor. Now, access the Kst Debug Log by selecting the Help → Debug Kst... menu-item and clicking the Log tab. You may see that a spike has been detected. If not, return to the terminal where you are running the `simrealtime.pl` script and continue generating new data until a number of spikes are visible in your plot. About 1% of the data should be spikes. Spikes which occur near the beginning of the data stream may not have been detected, because Kst didn't yet have enough data to establish a reliable estimate of the standard deviation of the vector '1'. After generating a few hundred data points, your Log may look something like this:



```
●2006-10-18 17:05 Scanning for data-source plugins.
●2006-10-18 17:05 Scanning for data-source plugins.
▼2006-10-18 17:06 Event Monitor: Possible Spike Detected!: 107
▼2006-10-18 17:06 Event Monitor: Possible Spike Detected!: 115
▼2006-10-18 17:06 Event Monitor: Possible Spike Detected!: 123
```

Show Messages
☑ Debug ☑ Notice ☑ Warning ☑ Error

Once you have an event monitor up and running, it's sometimes useful to have Kst indicate on your plots where events have occurred. This is best done using objects called 'Plot Markers'. To create plot markers for events, we need to have a Curve Data Object for the events in question. Create a curve data object now by selecting Data → New Curve... and configure the new curve with the options shown below.



New Curve

Unique name: <Auto Name>

Legend Text: <Auto Name>

Curve Contents

X axis vector: E1-ABS([1]-[1_Mean])>=5*[1_Sigma]/x

Y axis vector: E1-ABS([1]-[1_Mean])>=5*[1_Sigma]/y

Interpolate to: Y Axis Vector

+X error bar: <None>

+Y error bar: <None>

-X error bar: <None>

-Y error bar: <None>

☑ Use +X error bar for -X ☑ Use +Y error bar for -Y

Appearance

□ Show points Type: ● Density: All

☑ Show lines Type: Weight: 0

□ Bargraph Type: Solid

Placement

Tab: W1 New...

○ Place in existing plot: P1

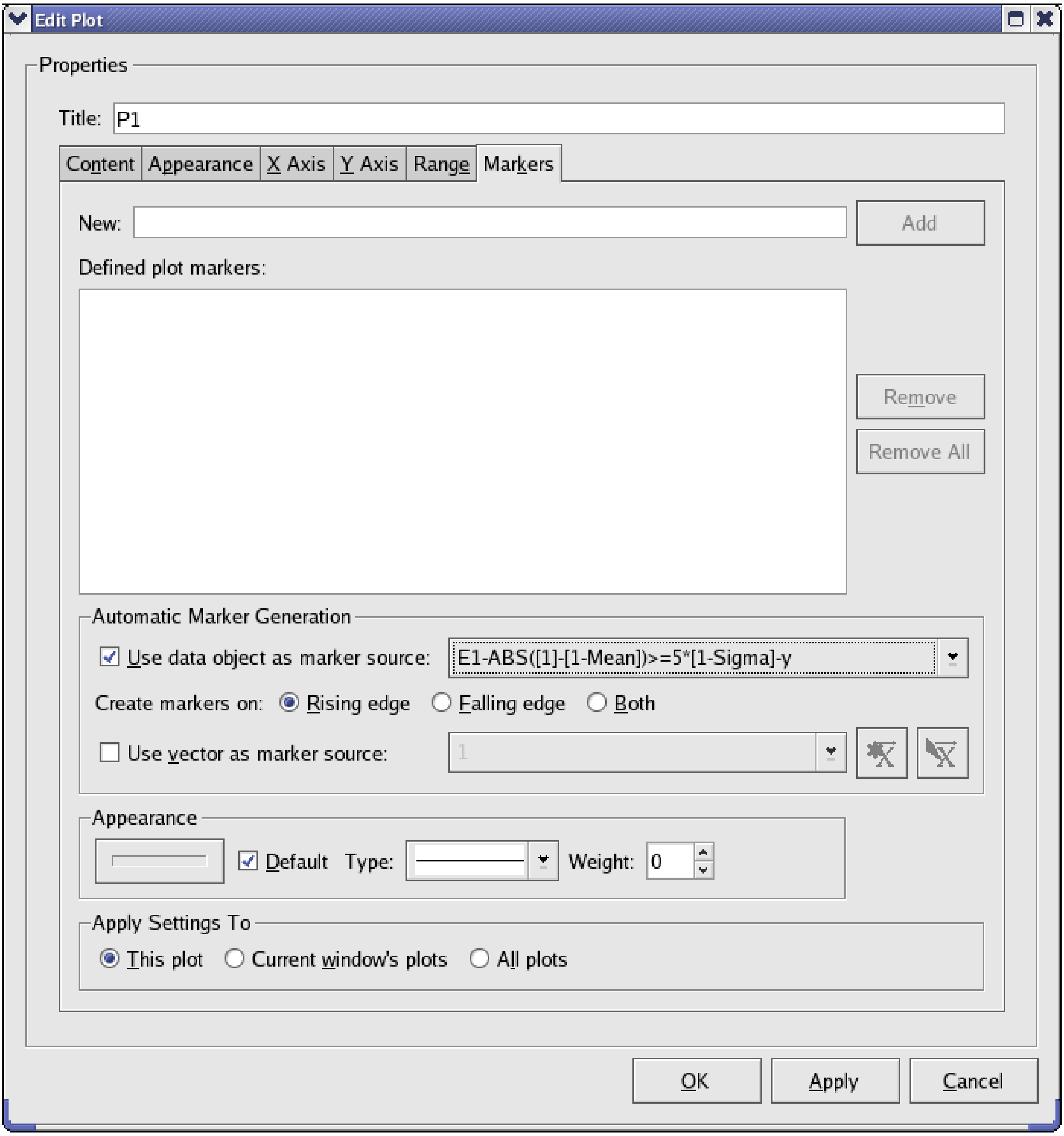
○ Place in new plot ☑ Re-grid Columns: 1

● Do not place in any plot

□ Ignore in automatic axes range calculations

OK Apply Cancel

Now we are ready to create plot markers. Right click on the data plot, and on the context menu which appears select Edit.... Then select the ‘Markers’ tab and configure it to use our event monitor curve as shown below:



Title: P1

New:

☑ Use data object as marker source: E1-ABS([1]-[1-Mean])>=5*[1-Sigma]-y

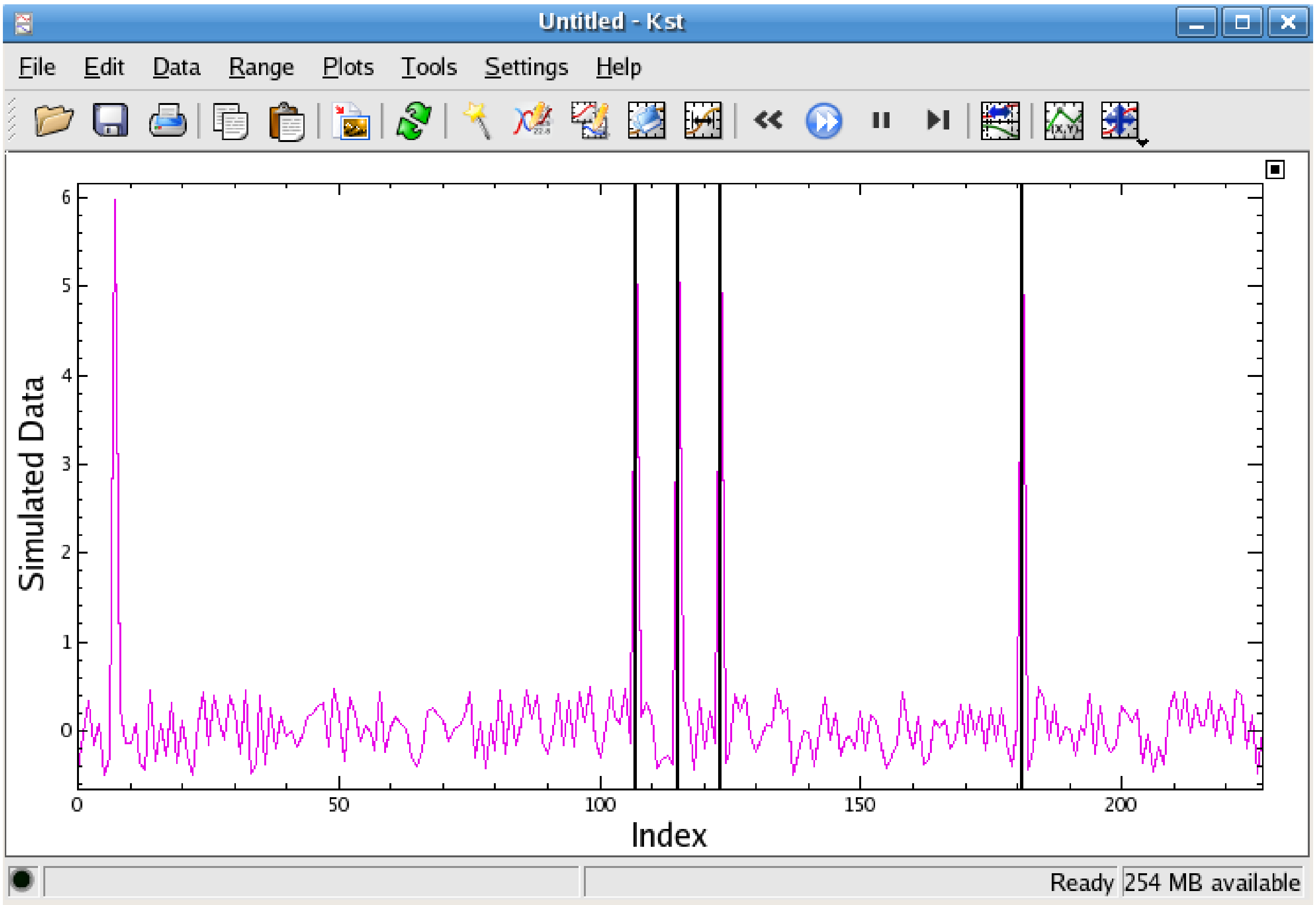
Create markers on: ● Rising edge ○ Falling edge ○ Both

☐ Use vector as marker source: 1

☑ Default Type: Weight: 0

● This plot ○ Current window's plots ○ All plots

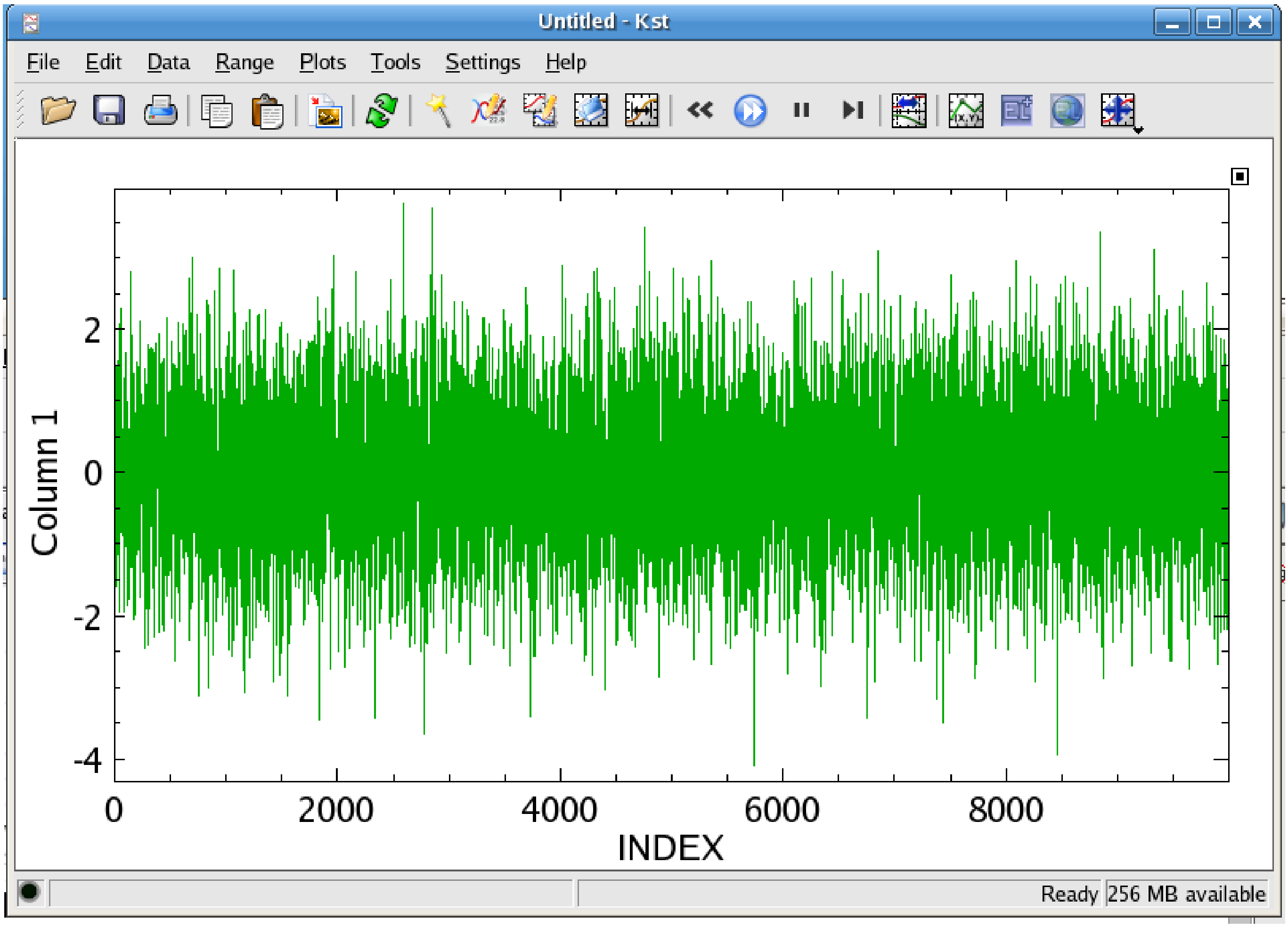
Click the OK button to create the markers. Vertical lines will now appear on the data plot, indicating the position of the spikes. The usefulness of plot markers is that they will always stretch across the entire y-axis, regardless of the zoom level, as shown in the screenshot below.



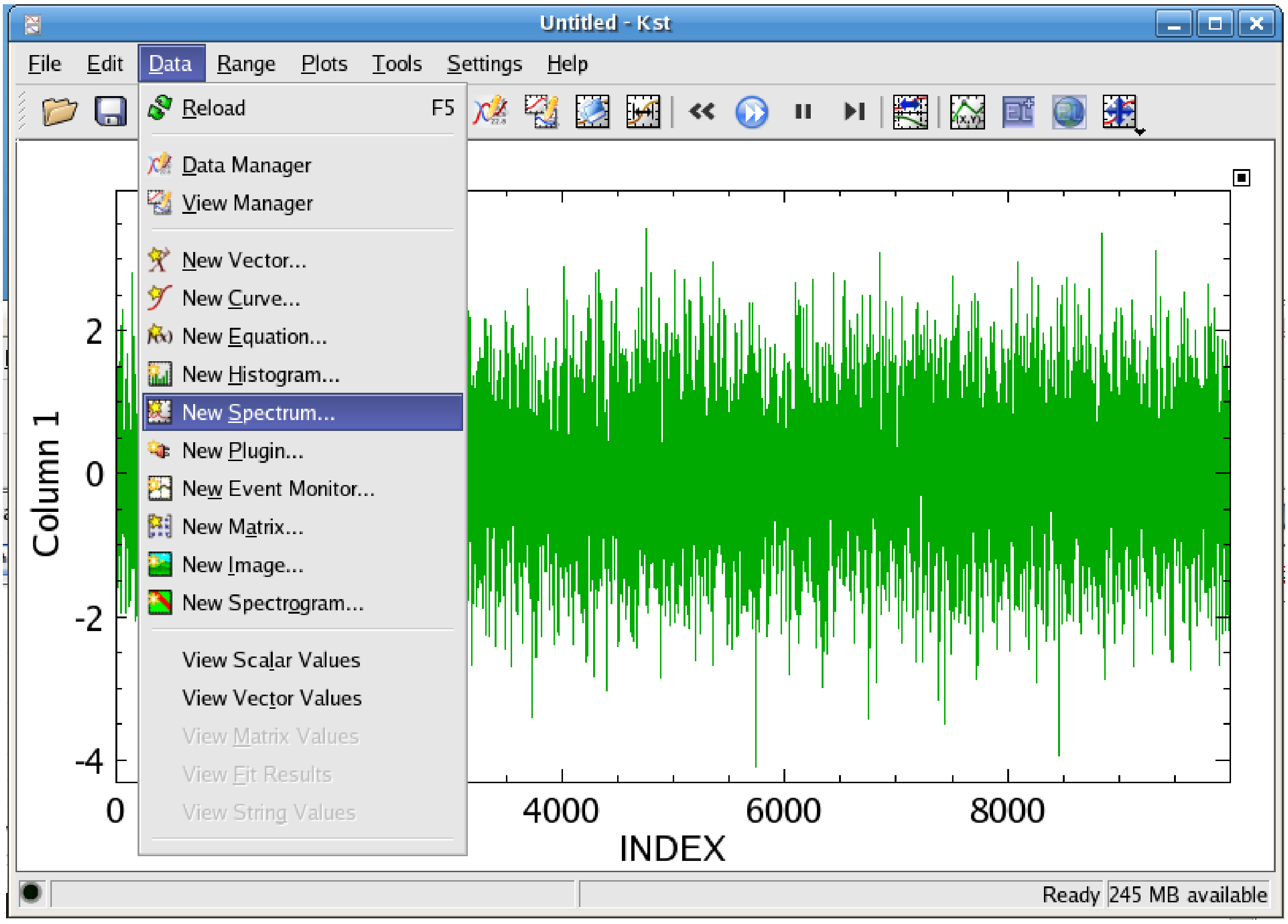
2.6 Creating a Power Spectrum

Kst contains built-in routines to generate power spectrum from data. The following example shows how to create a power spectrum.

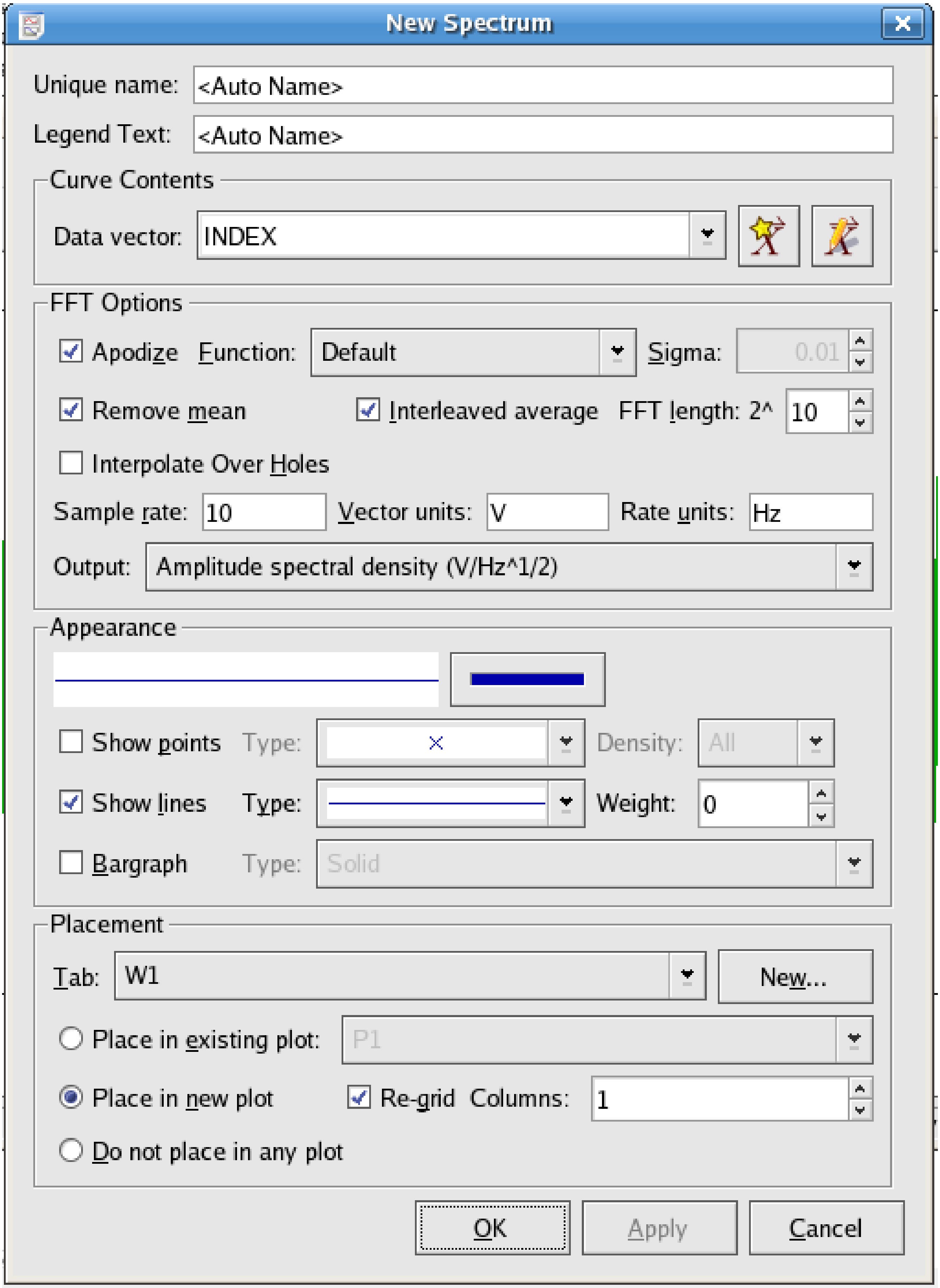
First, plot the normal distributed data vector generated by the perl script: generate_normal.pl in the histogram tutorial. The plot of the normal distributed data is shown below:



To create a power spectrum from this data vector, choose Data → New Spectrum...



Then, the dialog for creating a new specturm appears:



Unique name: <Auto Name>

Legend Text: <Auto Name>

Curve Contents

Data vector: INDEX

FFT Options

☑ Apodize Function: Default Sigma: 0.01

☑ Remove mean ☑ Interleaved average FFT length: 2^ 10

☐ Interpolate Over Holes

Sample rate: 10 Vector units: V Rate units: Hz

Output: Amplitude spectral density (V/Hz^1/2)

Appearance

☐ Show points Type: × Density: All

☑ Show lines Type: Weight: 0

☐ Bargraph Type: Solid

Placement

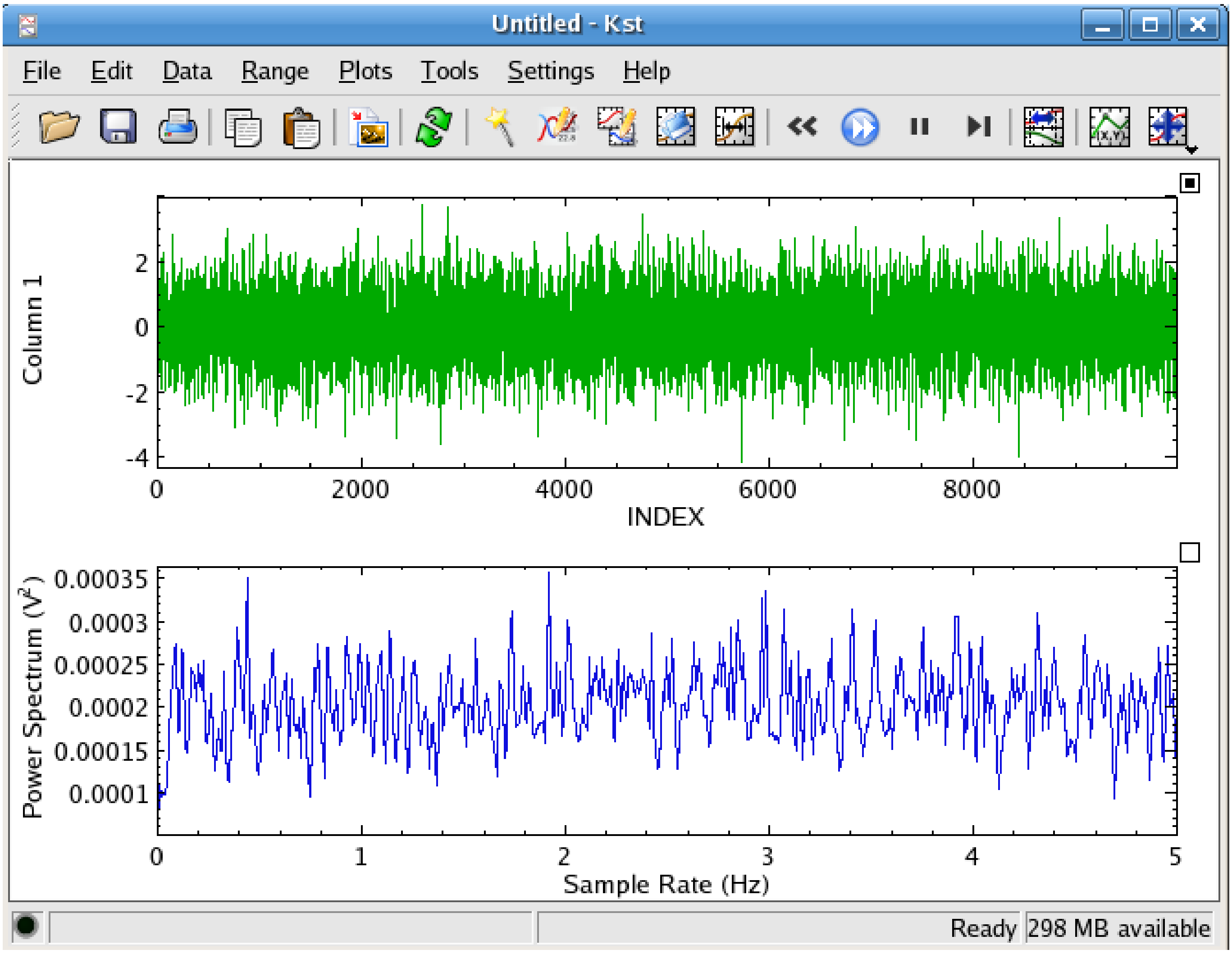
Tab: W1 New...

○ Place in existing plot: P1

● Place in new plot ☑ Re-grid Columns: 1

○ Do not place in any plot

In the text box beside Data vector: select the vector that you want to create a power spectrum from. The default is INDEX, but you can change it by selecting from the dropdown textbox. You can change the name of the specturm in the Unique name text box. FFT option will be explained in detail later in the spectrum section of the data chapter. For a quick look of the resulting spectrum, just use the default choices for the FFT option and click OK. The following spectrum is generated:

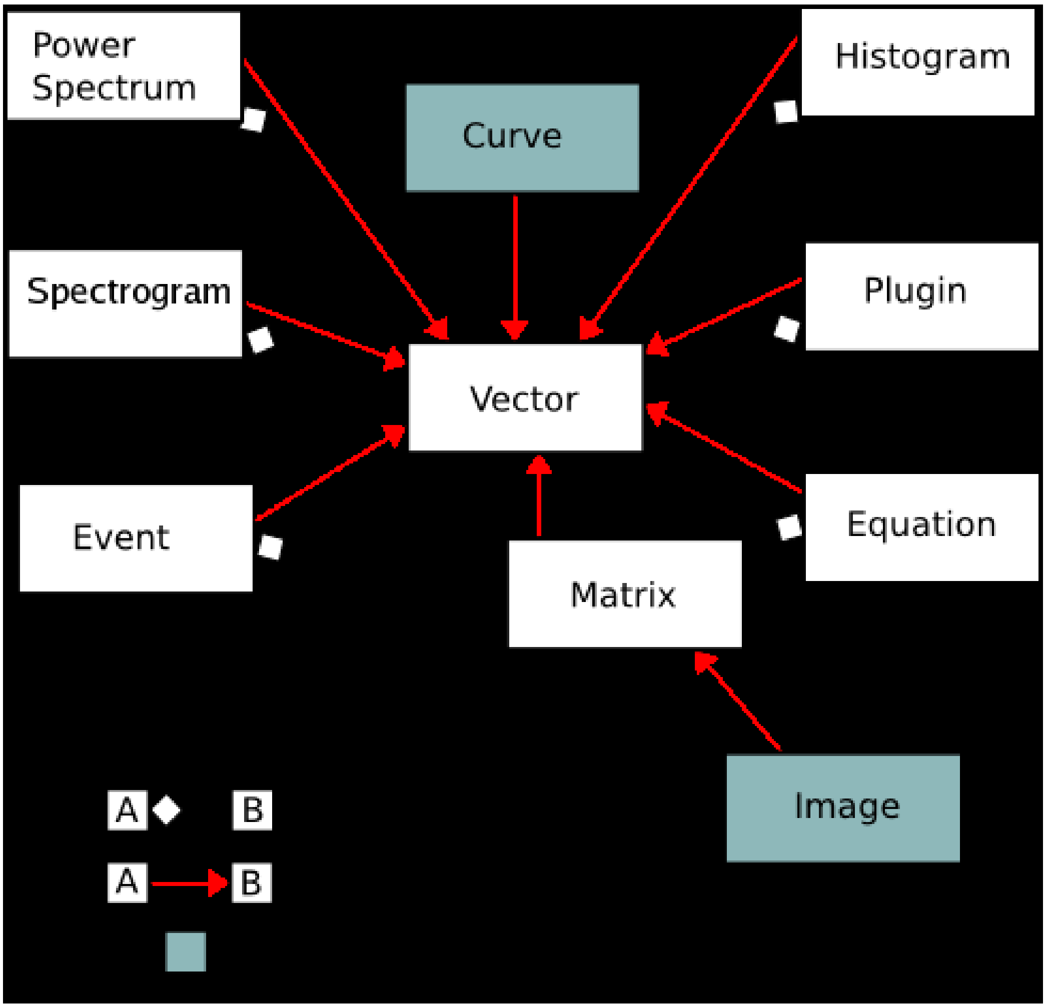


Chapter 3

Working With Data

3.1 Data Objects Overview

We will call Data Objects in Kst all objects which appear in the Data Manager. There are ten main kinds of data object in Kst. The following diagram illustrates the relationships between the different types:

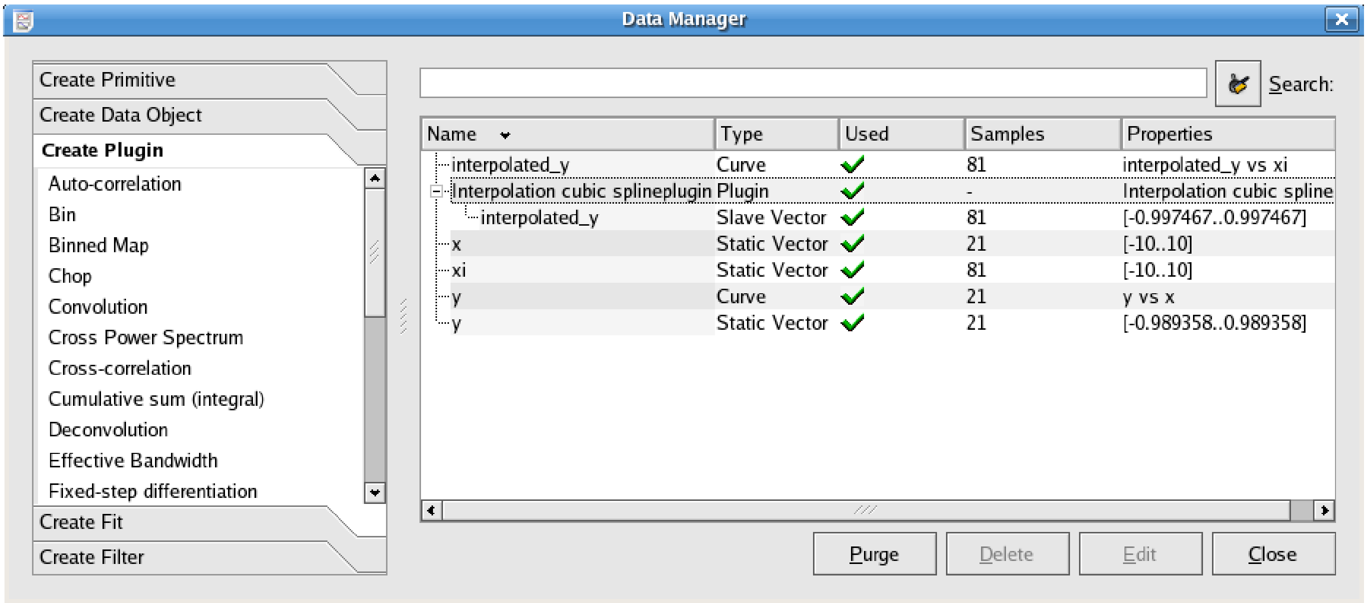


As can be seen in the diagram, the only truly ‘plottable’ data objects are curves and images. Many data objects contain slave vectors which can be plotted using curves, however — so these data objects are effectively plottable as well.

The usefulness of the Data Objects concept in Kst is that they can be tied together to create pipelines. An Event Monitor object, for instance, can take the output of another object as its input. Updates to objects in the pipeline propagate automatically. This is one of the key features which makes Kst powerful for realtime plotting.

3.1.1 The Data Manager

The Data Manager provides a central location for adding, deleting, and modifying all of the data objects used in Kst. It can be accessed from Data → Data Manager.



Name	Type	Used	Samples	Properties
interpolated_y	Curve	✓	81	interpolated_y vs xi
Interpolation cubic splineplugin	Plugin	✓	-	Interpolation cubic spline
interpolated_y	Slave Vector	✓	81	[-0.997467..0.997467]
x	Static Vector	✓	21	[-10..10]
xi	Static Vector	✓	81	[-10..10]
y	Curve	✓	21	y vs x
y	Static Vector	✓	21	[-0.989358..0.989358]

The panel on the left side of the Data Manager can be used to quickly create new data objects. The Purge button automatically deletes Data Objects which are not Used Data Objects. The Edit and Delete buttons allow you to selectively edit or remove particular objects. The Close button exits the Data Manager.

Tip

You can also create new data objects which are based on vectors by right clicking on the corresponding vector and choosing one of the Make ... options. If you select a curve, on the other hand, then you have the option to add it to an existing plot.

For its list of Data Objects, the Data Manager displays several pieces of information. These are described below:

Name The name of the data object, unique among the set of data objects with the same type.

Type The type of the data object determines how it is created and what its options are. Type can be one of: Data Vector, Curve, Equation, Histogram, Spectrum, Plugin, Event Monitor, Matrix, Image, or Spectrogram.

Used This field gives the status of the data object. If there is a check mark in the Used column of a data object, then some other object in Kst is dependent on it.

Samples The number of elements in the data object.

Properties A summary of the data object's key parameters, dependent on its type.

Tip

If you want to quickly find a vector among many items in the data manager, you may just type part of its name in the search field. Then, the data manager will respond by listing only those items which contain the entered text within their names.

3.1.2 Creating and Deleting Data Objects

You can create new Data Objects using either the left panel in the Data Manager, or the Data menu.

When you are creating a new data object, you may enter a unique name to identify the object. If you do not enter a custom name then a unique name will be automatically generated.

To delete a data object you must use the Data Manager. Note that if the Used column for a data object has a check mark then some other data or view object in Kst has a dependency on it. Depending on the strength of these dependencies, Kst will prompt before an object is deleted. If, for example, other objects have critical dependencies on the one which you are attempting to delete, Kst will ask if you would like to delete these other objects as well. Some dependencies are listed below:

- Plots are not critically dependent on the plottable objects which they contain, so if a plotted object is deleted Kst will automatically remove it from all plots, without prompting.
- All data objects which use a particular data vector must be deleted before the data vector itself can be deleted.
- All children of a parent data object must be unused before the parent data object can be deleted.

After a sequence of deletions and removals of plottable data objects from plots, you may find that there are numerous unused data objects displayed in the Data Manager. To quickly remove these objects, you can click the Purge button.

3.2 Data Types

3.2.1 Vectors

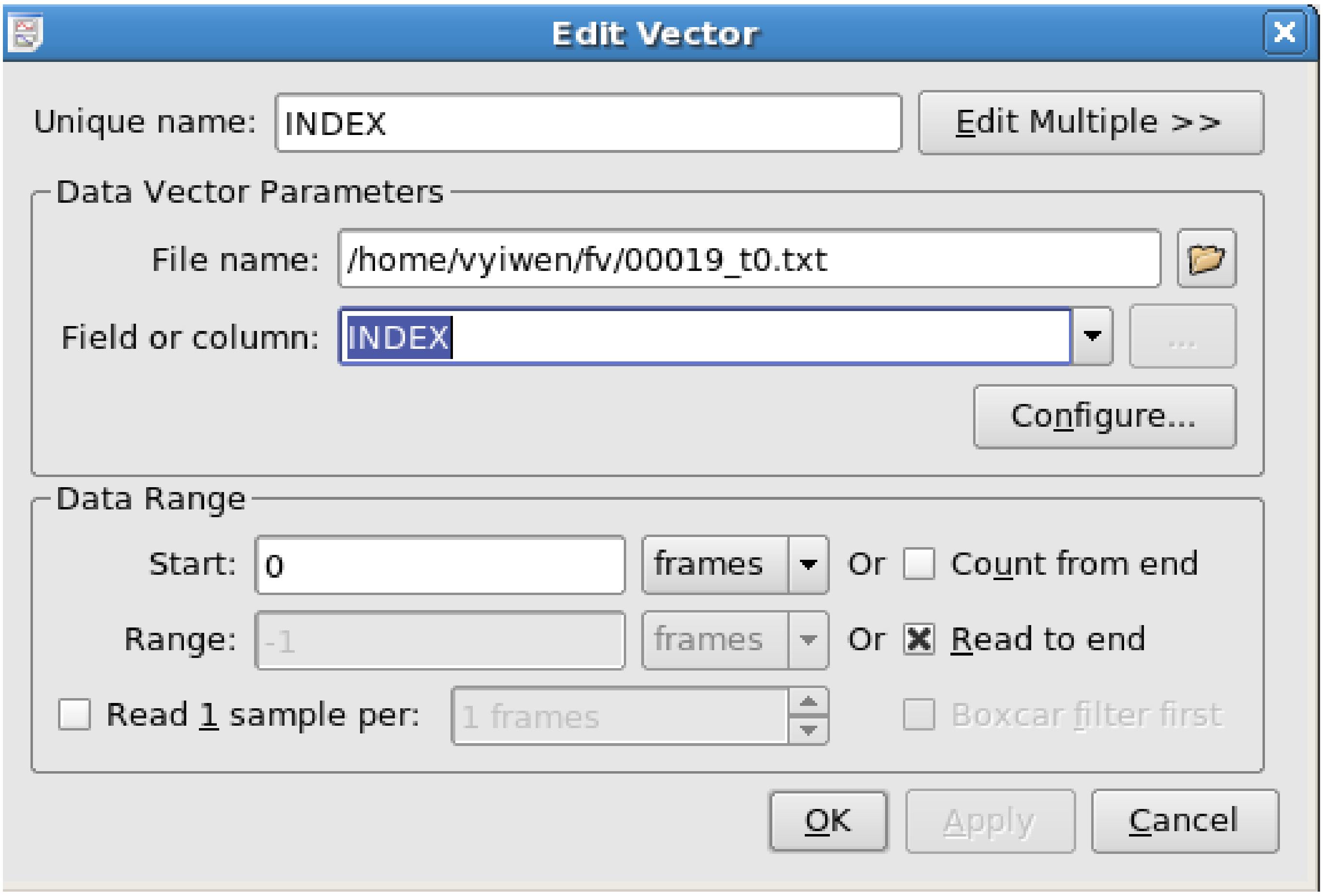
Vectors are one of the most often-used data objects in Kst. As their name implies, vectors are simply ordered lists of numbers. Usually they contain the x or y coordinates of a set of data points.

As vectors can potentially be quite large, it is a good idea to be aware of the amount of memory Kst has available for use. The current available memory is displayed in the lower right corner of the status bar of the main Kst window. If the status bar is not available, select Settings → Show Statusbar to display it.

There are two kinds of vector — data vectors and static vectors. Data vectors are read into Kst through a data source. Static vectors, on the other hand, are generated by Kst. They represent an evenly spaced list of numbers, and can be useful, for example, for generating an x-axis for a plot.

3.2.1.1 Data Vectors

The following is a screenshot of the window displayed when editing data vectors. A new data vector is created if you choose the Read from data source radio button in the New Vector dialog. Explanations of the settings are listed below.



Edit Vector

Unique name: INDEX

Data Vector Parameters

File name: /home/vyiwen/fv/00019_t0.txt

Field or column: INDEX

Data Range

Start: 0 frames Or □ Count from end

Range: -1 frames Or ☒ Read to end

□ Read 1 sample per: 1 frames □ Boxcar filter first

File name The path to the desired data file. Clicking the button to the right displays a file browser window that can be used to graphically browse for the file.

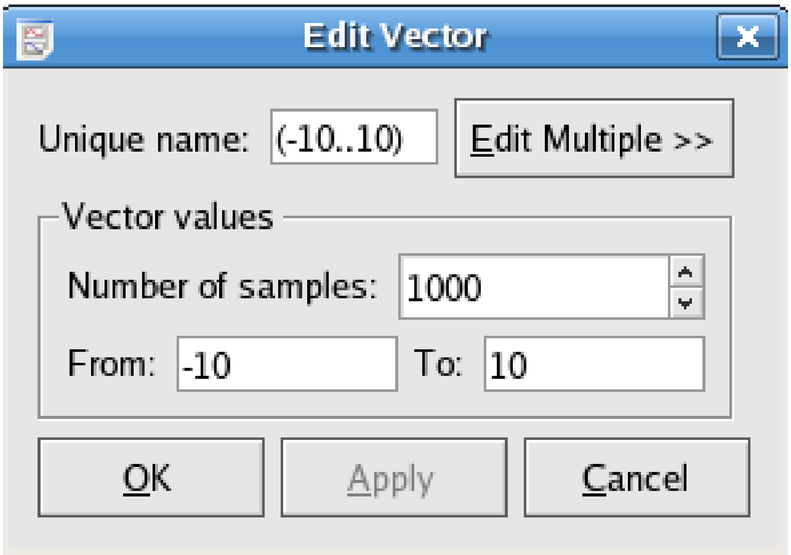
Field or column The field or column to use for the vector.

Start frame, Count from end, Range, and Read to end This section specifies the range of data to use from the selected field for the data vector. This discussion assumes a knowledge of the Frames concept. Using these four settings, the lower and upper boundaries of the data range can be set. For example, to read from frame 10 to frame 900, enter `10` for Start frame and `890` for Range. To read from frame 500 to the end of the file, enter `500` for Start frame and select the Read to end option. To read only the last 450 frames from the file, select the Count from end option and enter `450` for Range. The combinations used in the previous two examples are often useful for reading data from a file that is being updated in real time. Subsequent additions to the file are read, causing associated vectors to be updated as well.

Read 1 sample per and Boxcar filter first In addition to the lower and upper range boundaries, the samples to read from the selected range can be set. If Read 1 sample per is not selected, all samples in the selected range will be read. If Read 1 sample per is selected, only 1 sample per `N` frames will be read, where `N` is the number entered in selection box to the right. The value of the 1 sample that is used to represent a frame depends on whether or not Boxcar filter first is selected. If Boxcar filter first is not selected, the value is the same as the value of the 1st sample in the frame. If Boxcar filter first is selected, the value is the average of all the samples in that particular frame.

3.2.1.2 Static Vectors

The following is a screenshot of the window displayed when editing static vectors. A new static vector is created if you choose the Generate radio button in the New Vector dialog. Explanations of the settings for static vectors are listed below.



Unique name: (-10..10)

Number of samples: 1000

From: -10 To: 10

The entries of the vector will be From + (To - From)*(i-1)/(Number of samples - 1) for i = 1,...,Number of samples.

3.2.1.3 Generating Vectors from KstScript

You can also create and initialize vectors by using KstScript. See details in the KstScript of the Extensions Chapter, and in the Vector and DataVector Classes of the Script Chapter.

Examples are provided below:

Example 3.1 Example of Using KstScript to Construct a Static Vector

```
v = new Vector(); //construct a new vector
v.tagName = "v1"; //give it a name in the Data Manager
v.resize(100); //give it a length of 100
for(var i=0; i<100; i++){v[i]=Math.random()};//intialize this vector
```

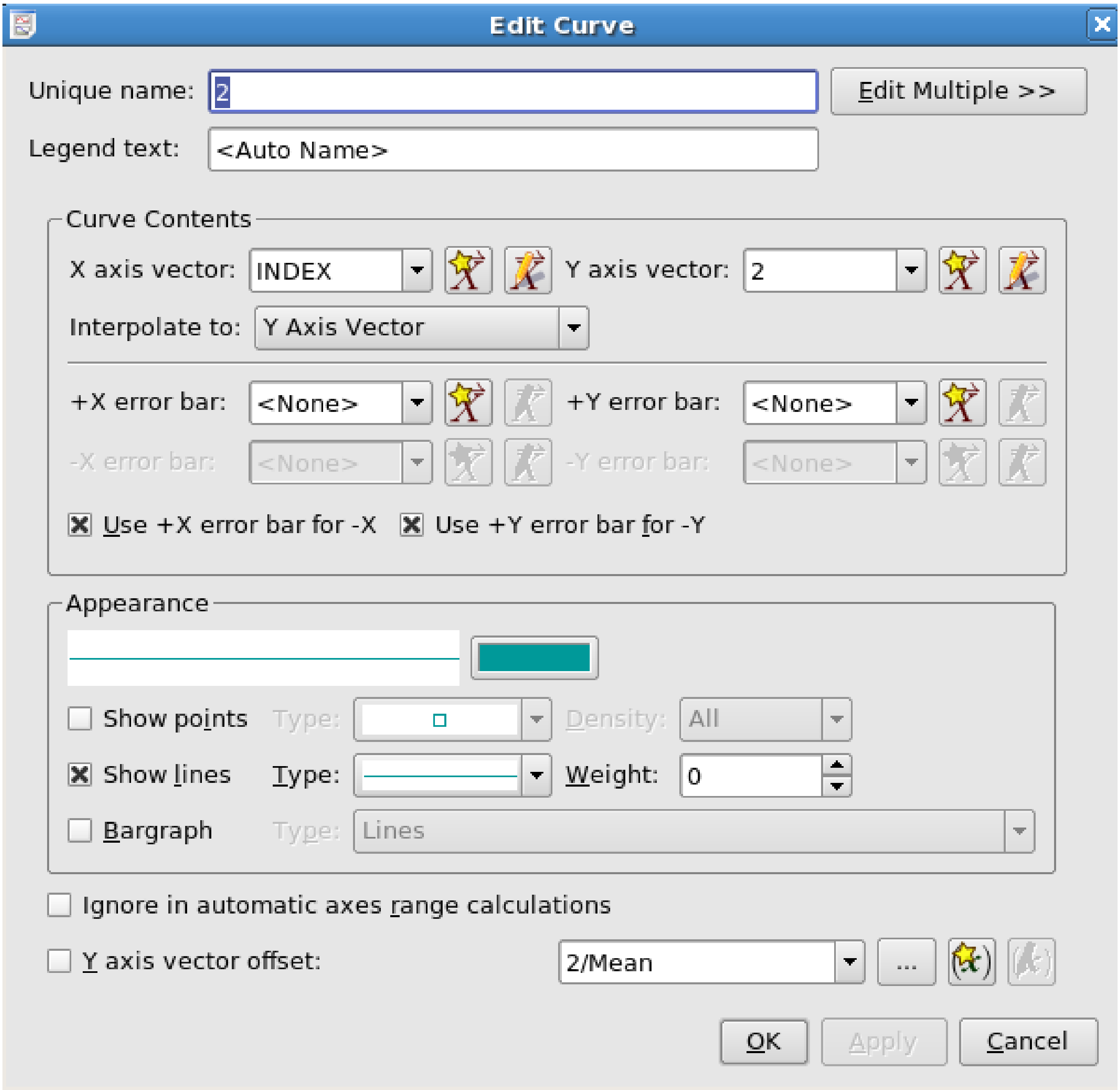
Example 3.2 Example of Using KstScript to Load a Vector from Source Files

```
source = new DataSource("rand.dat");//specify a data source file system. See the DataSource ←↩
     Class for details
v = new DataVector(s,"1");//construct a new vector v and set its value from field 1 of the  ←↩
    source file
v.tagName = "vector" //give the vector a name in the Data Manager
```

3.2.2 Curves

Curves are primarily used to create plottable objects from vectors. Curves are created from two vectors - an ‘X axis vector’ and a ‘Y axis vector’, that provide a set of data points. Thus, a curve can be thought of as a set of data points and the lines that connect them (even though the points or lines may not be visible on plots).

The following is a screenshot of the window displayed when creating or editing curves. Explanations of the settings are listed below.



Unique name: 2

Legend text: <Auto Name>

X axis vector: INDEX

Y axis vector: 2

Interpolate to: Y Axis Vector

+X error bar: <None>

+Y error bar: <None>

-X error bar: <None>

-Y error bar: <None>

☒ Use +X error bar for -X

☒ Use +Y error bar for -Y

☐ Show points Type: □ Density: All

☒ Show lines Type: Weight: 0

☐ Bargraph Type: Lines

☐ Ignore in automatic axes range calculations

☐ Y axis vector offset: 2/Mean

Legend Text The string to be used in Plot Legends to describe this curve.

X axis vector The vector to use for the independent (horizontal) axis.

Y axis vector The vector to use for the dependent (vertical) axis.

+X error bar and +Y error bar The vectors containing error values corresponding to the X axis vector and Y axis vector, respectively. The vectors should contain the sizes of the error bars in the same order as the data points.

Use +`X/Y` error bar for -`X/Y` If this item is checked, the error bars are drawn symmetrically about the data point. To draw asymmetric error bars in the x or y direction, uncheck the box, and choose a vector for the length of the error bar below the data point in the -*X/Y* error bar combo box.

-`X/Y` error bar The vector to use for the error bar below the data point. See the previous entry for more detail on using asymmetric error bars.

Color Determines the color of the data points of the curve. Clicking the button displays the standard KDE color chooser.

Show points Determines whether or not the data points will be indicated by the selected type of marker.

Density If Show points is enabled, Density allows you to have Kst show only occasional points if they would otherwise clutter the plot.

Show lines Determine whether or not the data points will be connected by lines

Weight If Show lines is enabled, this gives the thickness in pixels of the curve.

Bargraph Draws lines or rectangles from the data points to the y = 0 line in the plot.

Ignore in automatic axes range calculations If this option is checked, the curve will be ignored in determining the automatic y-axis range; if it is unchecked the curve will be used in determining the automatic y-axis range. By default, this option is unchecked.

Y axis vector offset If this option is checked, the corresponding selected scalar will be subtracted from the y values of the curve.

3.2.2.1 Using the Curves Class in KstScript

Plotting a curve can be done by using KstScript. See details in the Curve Class of the Script chapter.

Example 3.3 Example of Using KstScript to plot a curve

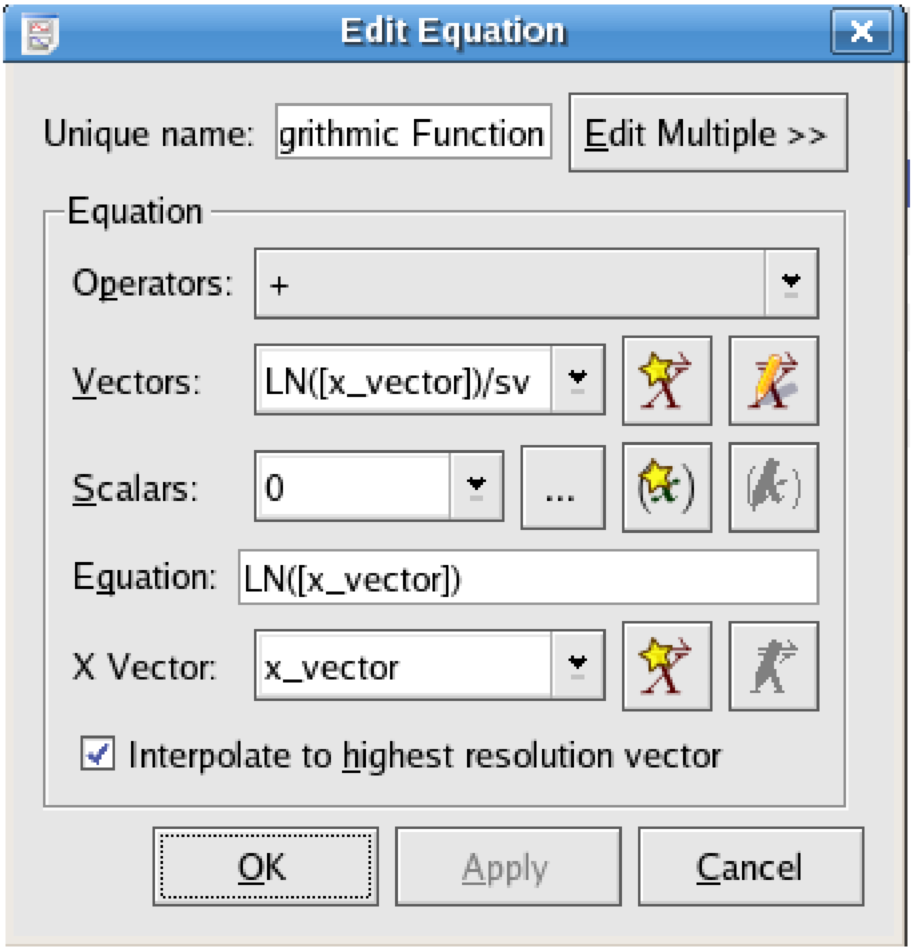
```
//xv and yv are names of two existing vectors in the Data Manager.
//See example 3.1 and 3.2 for constructing vectors using KstScript.
c = new Curve("xv", "yv") // to construct a new curve using xv and yv
c.tagName = "xv vs yv"; //define the name for the curve in the data manager
 //to view the curve, a window object and a plot object need to be constructed.
w = new Window();
p = new Plot(w);
// using the append method in the CurveCollection Class to show the curve in plot
p.cuves.append(c);
```

There is a detailed example of vectors, plots and curves in the Extensions Chapter. Click here to view it.

3.2.3 Equations

An equation data object consists of a mathematical expression and an independent variable. The expression is built using a combination of scalars, vectors, and operators, and usually represents the values of the dependent variable. The independent variable is generated from an existing vector, and stored as a slave vector along with the evaluated results of the equation.

The following is a screenshot of the window displayed when creating or editing equations. Explanations of the settings are listed below.



Unique name: grithmic Function

Equation

Operators: +

Vectors: LN([x_vector])/sv

Scalars: 0

Equation: LN([x_vector])

X Vector: x_vector

☑ Interpolate to highest resolution vector

Operators A list of available operators. Choosing an operator from the list inserts the operator at the current insertion point in the Equation text box.

Vectors A list of the current vector objects in Kst. Choosing a vector from the list inserts the vector at the current insertion point in the Equation text box.

Scalars A list of the available scalar values. This list is composed of both the scalar values in the current Kst session as well as some built-in constants. Choosing a scalar from the list inserts the scalar at the current insertion point in the Equation text box.

Equation The mathematical expression representing the independent variable. You may manually type in this text box or you may select items to insert using the above drop-down lists. When manually typing equation expressions, remember to *put vector and scalar names into square brackets*

For a full list of the valid Kst operators, functions and constants to express equations, see the Equation Expression in Kst.

X Vector Select a vector to use as the independent vector. Select a vector from the drop-down list, or quickly create a new one by clicking the wizard button to the right.

Interpolate to highest resolution vector Select this option to interpolate the selected X vector to the greatest number of samples possible, given the data used in the equation expression. For example, if the data expression acts on a vector Y which contains twice as many variables as the selected X vector, the equation object will create a slave vector of interpolated X values with the same number of points as Y.

3.2.3.1 Using the Equation Class in KstScript

The Equation Class in KstScript provides methods to manipulate Equation objects. An example is shown below.

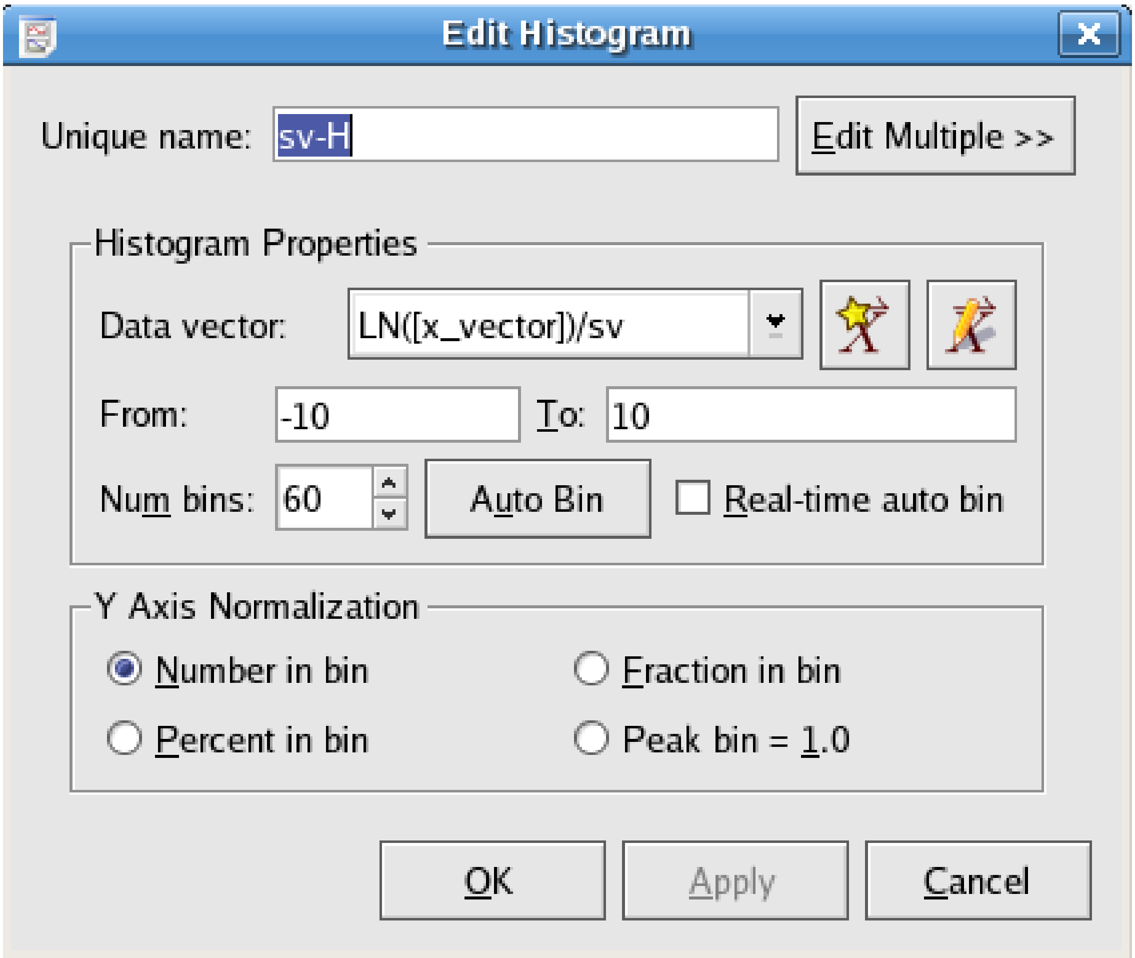
Example 3.4 Example of Using the Equation Class

```
//construct a vector object
x = new Vector();
x.tagName = "x_vector";//call it x_vector in the Data Manager
x.resize(100);

//initialize x_vector
for(var i=0; i<100; i++){x[i]=i;}

//construct an equation object
eq = new Equation("LN([x_vector])",x);//specify the equation expressions
                                      //check equation appendix to see
                                      //what expressions are supported by Kst.
eq.tagName = "Natural Logarithmic Function";

//to check if the equation is valid
eq.valid
//this should return true

//to construct a curve using the resulting vectors of applying this equation
c = new Curve(eq.xVector, eq.yVector);
c.tagName = "eq_curve";
//to view the curve, see Example 3.3.
```

3.2.4 Histograms

This object calculates the histogram of a chosen vector.

The following is a screenshot of the histogram specific options when creating or editing histograms. Explanations of the settings are listed below.



Edit Histogram

Unique name: sv-H

Histogram Properties

Data vector: LN([x_vector])/sv

From: -10 To: 10

Num bins: 60

☐ Real-time auto bin

Y Axis Normalization

● Number in bin

○ Fraction in bin

○ Percent in bin

○ Peak bin = 1.0

Data vector The data vector to create the histogram from. Although a vector is needed to create a histogram, the vector is treated as an unordered set for the purposes of creating a histogram.

From and To The From field contains the left bound for the leftmost bin in the histogram. The To field contains the right bound for the rightmost bin in the histogram.

Num bins Enter the total number of bins to use in this field.

Auto Bin Instead of specifying values for From, To, and Num bins, you can click Auto Bin to automatically generate values for all three fields based on the length and lowest/highest values of the chosen data vector.

Number in bin Selecting this option causes the y axis to represent the number of elements in each bin.

Fraction in bin Selecting this option causes the y axis to represent the fraction of elements (out of the total number of elements) in each bin.

Percent in bin Selecting this option causes the y axis to represent the percentage of elements (out of the total number of elements) in each bin.

Peak bin = 1.0 Selecting this option causes the y axis to represent the number of elements in each bin divided by the number of elements in the largest bin (the bin with the greatest number of elements).

3.2.4.1 Using the Histogram Class in KstScript

Histograms can also be constructed using KstScript. See details in the Histogram Class of the Working with KstScript appendix.

Example 3.5 Example of Constructing a Histogram through KstScript

The example below uses the data file `normal.dat` generated by the perl script in Generating a Histogram of Chapter 2

```
//load the data vector from normal.dat; see details in example 3.2
s = new DataSource("normal.dat");
v = new DataVector(s, "1");

//construct a histogram object using vector v.
hs = new Histogram(v);

//construct a histogram curve; see details in the Curve Class
c = new Curve(hs);

//view the histogram; see example 3.3
w = new Window();
p = new Plot(w);
p.curves.append(c);
```

3.2.5 Spectrum

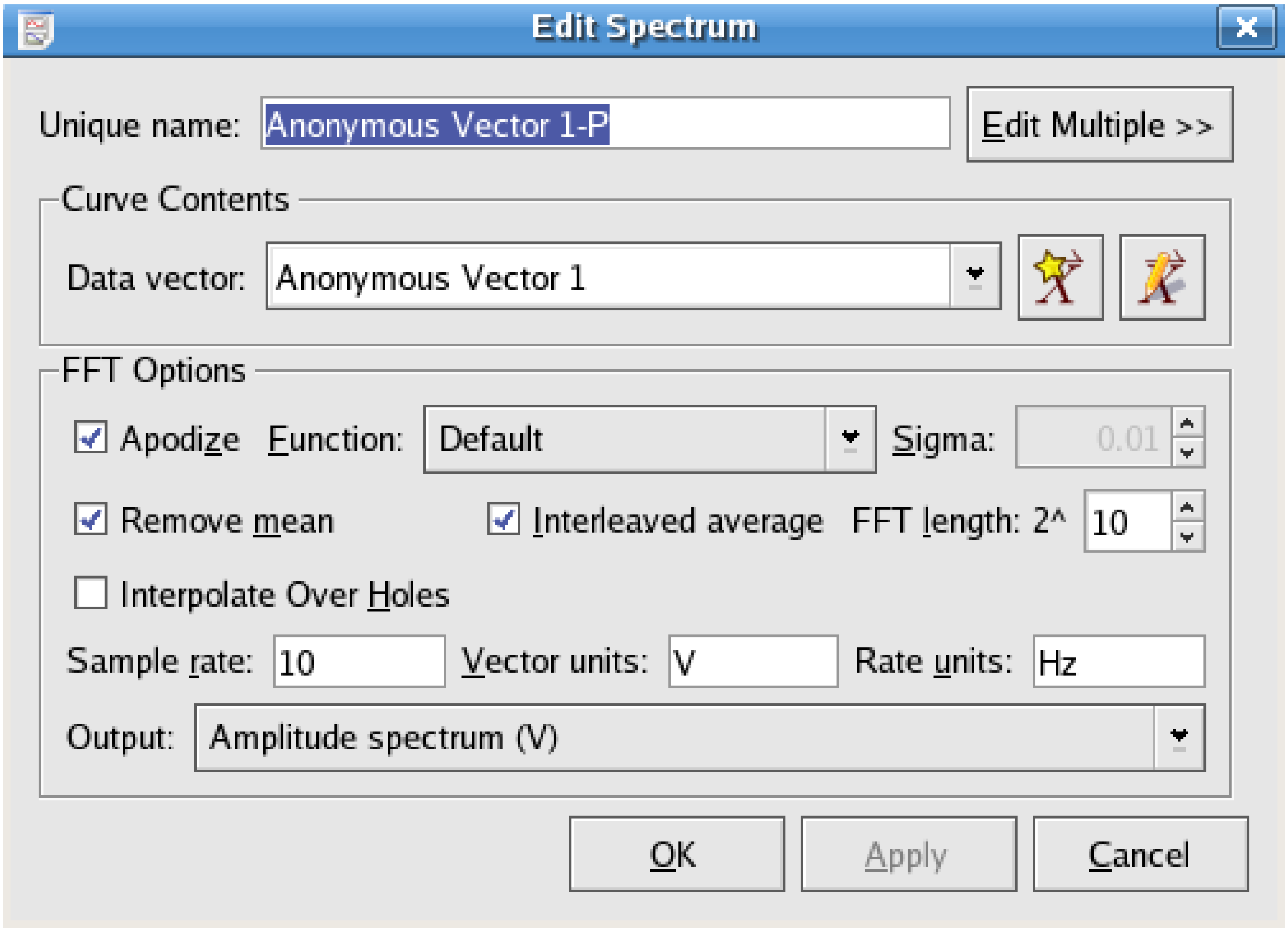
This data object calculates the Discrete Fourier Transform (DFT) of an input signal using a Fast Fourier Transform (FFT) algorithm, and outputs a spectrum of the signal.

The mathematical definition of the DFT is:

$$X(k)=\sum_{n=0}^{N-1} x(n)*e^{-j2\pi nk/N}, \quad k=0, 1, ..., N-1$$

where `x(n)` is the input signal, and `N` is the number of samples.

The following definitions assume basic knowledge of power spectra — we use the conventions and terminology of ‘Numerical Recipes in C: The Art of Scientific Computing’, published by Cambridge University Press.



Edit Spectrum

Unique name: Anonymous Vector 1-P

Curve Contents

Data vector: Anonymous Vector 1

FFT Options

☑ Apodize Function: Default Sigma: 0.01

☑ Remove mean ☑ Interleaved average FFT length: 2^ 10

□ Interpolate Over Holes

Sample rate: 10 Vector units: V Rate units: Hz

Output: Amplitude spectrum (V)

Data vector The data vector to create the spectrum from.

Apodize, Function, and Sigma An Apodization Function is a function used to reduce leakage.

In the drop-down menu of Function, the apodization function applied to the spectrum can be chosen from:

- Default
- Bartlett
- Blackman
- Connes
- Cosins
- Gaussian (custom sigma)
- Hamming
- Hann
- Welch

If the Apodize option is selected, the data is windowed using the function chosen from the drop-down menu at right. Apodization can be used to reduce bin-to-bin leakage but the resolution of the plot will be decreased. If the Gaussian apodization function is chosen, then a sigma value may also be entered to define its effective width.

Remove Mean Select this option to remove the mean from the data while calculating the transform.

Interleaved average and FFT Length Selecting Interleaved average calculates the FFT over segments of the data, which increases the FFT speed and improves accuracy, at the result of less information about low frequency components. The length of the interleaved segments is specified as a power of 2. If Interleaved average is unchecked, Kst will choose the smallest possible FFT length which is greater than the data vector length.

Interpolate Over Holes If the data vector contains NaN values, then selecting this option will cause Kst to interpolate through them when calculating the power spectrum.

Sample rate This is the number of data vector samples per unit time. The sample rate is used to generate the X axis of the spectrum for plotting.

Vector units and Rate units The units specified in these textboxes are used for the purpose of auto-generating axes labels for the plots.

Output The output spectrum can be chosen from

- Amplitude spectral density (V/Hz^1/2)
- Power spectral density (V^2/Hz)
- Amplitude spectrum (V)
- Power spectrum (V^2)

The option selected from this drop-down menu determines which variant of the power spectrum should be computed.

The power spectrum is the plot of |X(k)|^2 vs frequency vector generated by the sample rate. The amplitude spectrum is calculated by taking the square root of the power spectrum. Spectral densities are calculated by multiplying the Power or Amplitude Spectra by the factor: N/sample rate, where N is the number of data samples.

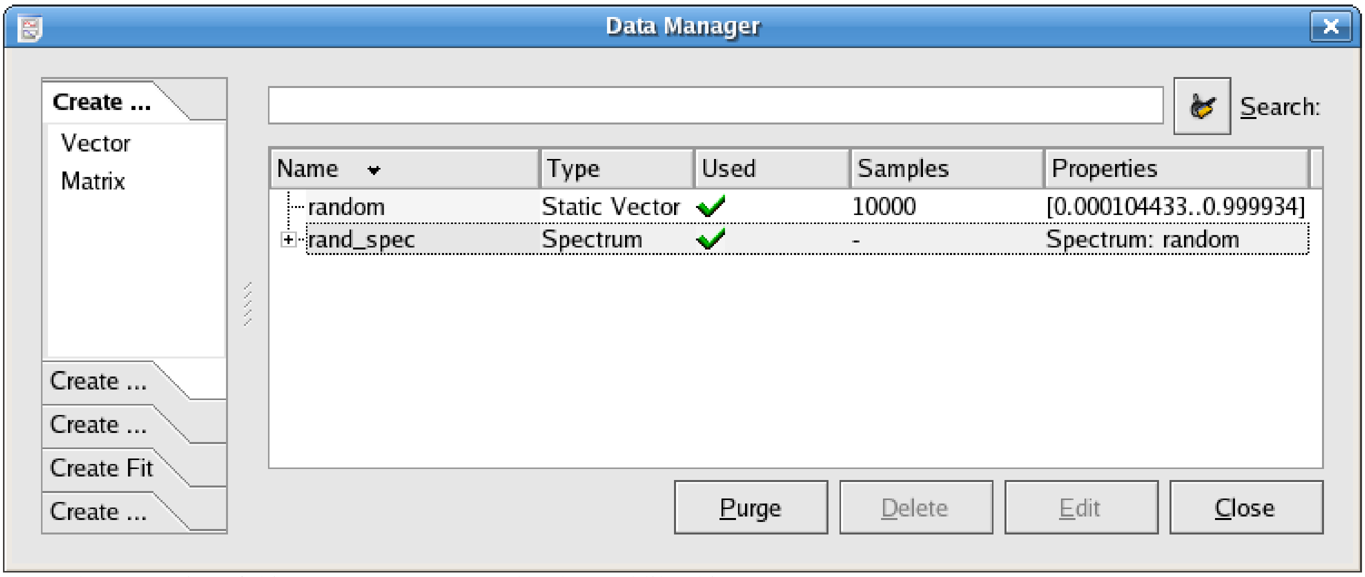
3.2.5.1 Using the PowerSpectrum Class in Kst

The PowerSpectrum Class in KstScript can also be used to construct power spectra.

Example 3.6 Example of Using PowerSpectrum Class

```
//construct a random vector whose value is between 0 and 1 with 10000 samples
x = new Vector();
x.resize(10000);
x.tagName = "random" //give it a name: random in Data Manager
for(var i=0; i <10000; i++){x[i]=Math.random()}

//construct a Power Spectrum according to the constructor:
//PowerSpectrum(vector, freq, [,average[,len[,apodize[,removeMean]]])
ps = new PowerSpectrum(x, 10, true, 10, true, false)
ps.tagName ="rand_Spec";
```

After executing the above scripts, the Data Manager should look like this:



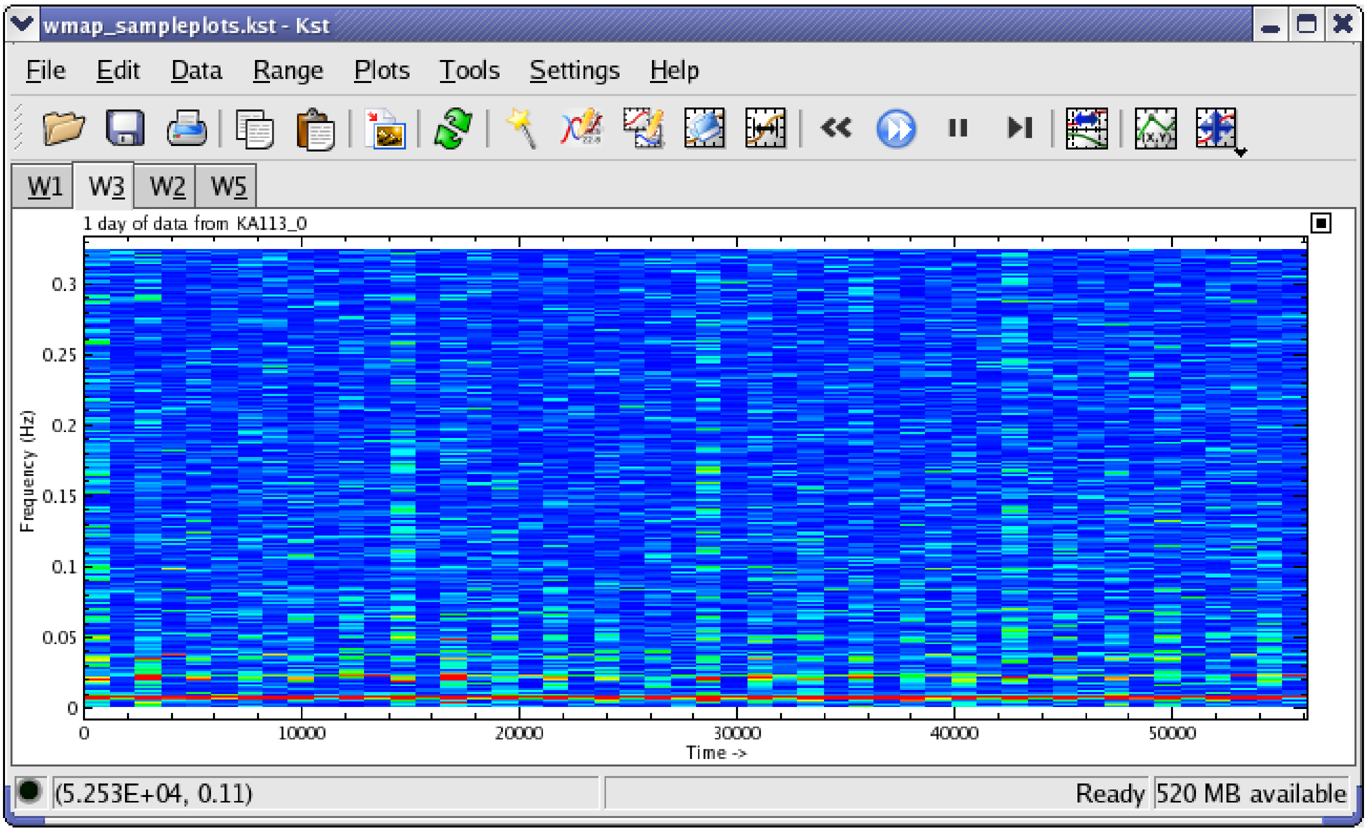
Name	Type	Used	Samples	Properties
random	Static Vector	✓	10000	[0.000104433..0.999934]
rand_spec	Spectrum	✓	-	Spectrum: random

Later, the properties of this power spectrum can be changed like this:

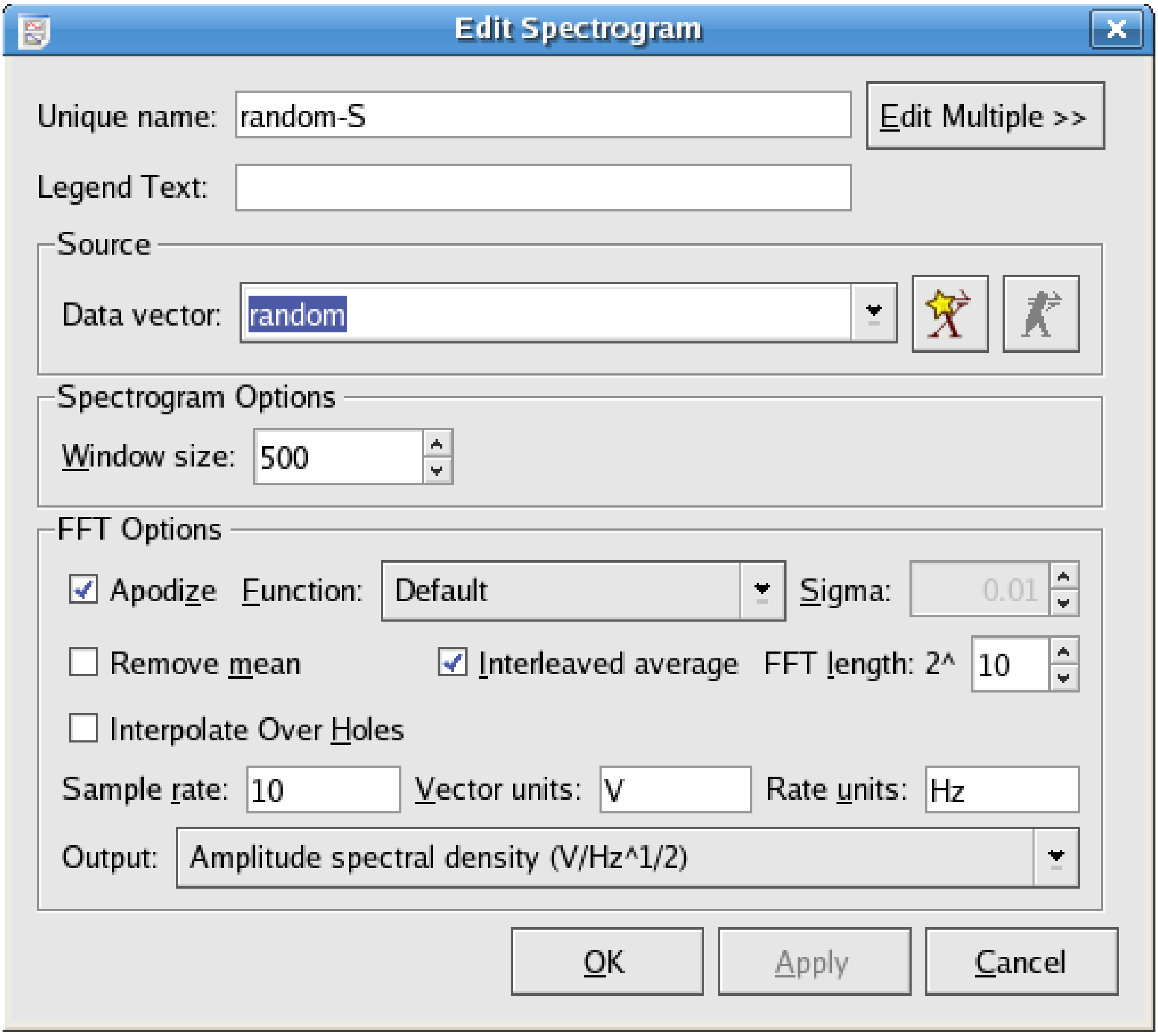
```
ps.removeMean = true; //choose the remove mean option
ps.length = 13; //change the FFT length to be 2^13
```

3.2.6 Spectrogram

A spectrogram is a way to plot the frequency spectrum of a data vector as a function of position within the data. It allows you to visualize, for example, how the frequency distribution of a signal changes with time. In Kst the spectrogram data object takes a data vector as an input and produces an image object which can be displayed as a ‘waterfall’ plot. The waterfall plot is used to show how two-dimensional information changes over time; here, it indicates how power spectrum vs frequency of a data set changes over time. An example is shown in the screenshot below. This is a spectrogram from one day of data from a channel of the WMAP satellite. The size of the intervals over which the component spectra are calculated has been chosen so that each interval corresponds to 1/2 hour of data. The 1h periodic precession of the satellite about its spin axis can be clearly seen. Also, at ~0.007Hz the red lines at the bottom of the image show the much faster (0.5rpm) spin frequency of the satellite.



Here is the edit dialog for the spectrogram object.



Unique name: random-S

Legend Text:

Data vector: random

Window size: 500

☑ Apodize Function: Default Sigma: 0.01

□ Remove mean ☑ Interleaved average FFT length: 2^ 10

□ Interpolate Over Holes

Sample rate: 10 Vector units: V Rate units: Hz

Output: Amplitude spectral density (V/Hz^1/2)

An explanation of the spectrogram specific options is given below:

Window Size The length of the data over which to take the spectra. The data vector will be subdivided into intervals of this length, and a spectrum will be computed for each one.

FFT Options These are the options for the individual spectra. See the Spectrum data object for more information. The output quantities are indicated by color scalar.

Color Palette This option should be accessed from the Edit Image dialog of the image created from the spectrogram.

Here you can choose the color scheme to use for the spectrogram. The amplitudes of the spectra are color-coded according to the selected scheme.

3.2.6.1 Using the Spectrogram Class in KstScript

The spectrogram can also be generated by using the Spectrogram class in KstScript

Example 3.7 Example of Using the Spectrogram Class

The vector: ‘random’ generated in Example 3.6 will continue to be used in this example

```
s = new Spectrogram("random", 10)//generate a spectrogram using vector random
                               //and specify the frequency to be 10 Hz
s.windowsize = 500; // specify the window size to be 500
s.tagName = "rand_Spectrogram";
```

Now, you can check that ‘rand_Spectrogram’ is in the Data Manager. You can view the waterfall plot by making an image from the slave matrix

3.2.7 Plugins

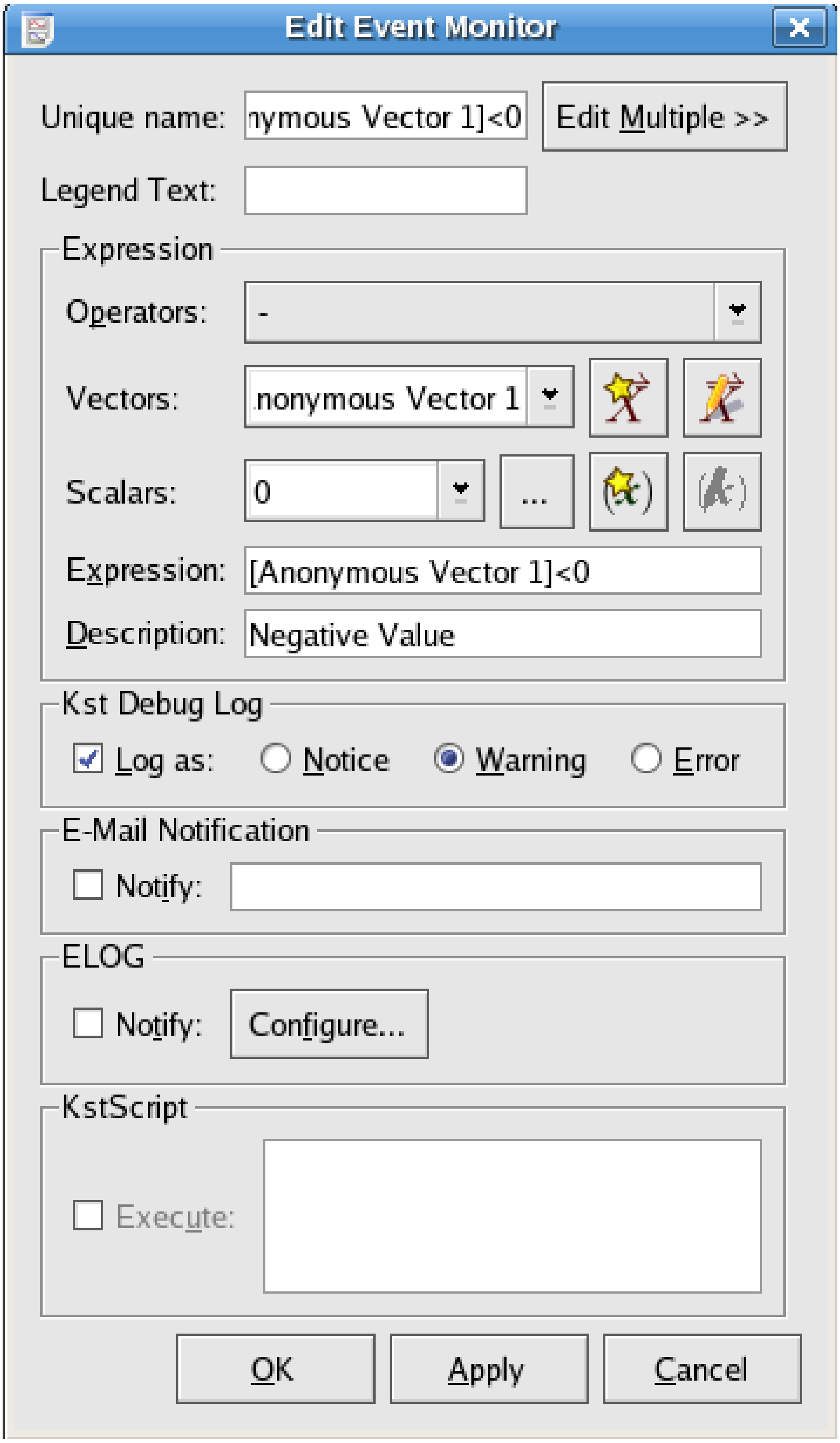
A plugin data object represents a Kst plugin. All plugins have a common format, and show up as type ‘Plugin’ in the Data Manager. For more information about the available plugins and their options, please see the Plugins chapter.

3.2.8 Event Monitors

An event monitor data object essentially keeps track of one or more vectors or scalars, and performs an action when a specified condition involving the vectors and scalars is true. Event monitors are usually used in conjunction with ‘live’, or streaming data. For example, an event monitor could be created to monitor whether or not elements of a data vector representing temperature exceed a predefined value.

You can create a new event monitor from the Data → New Event Monitor..., or by using the Data Manager.

The following is a screenshot of the window displayed when creating or editing events. Explanations of the settings are given below.



Unique name: nymous Vector 1]<0

Legend Text:

Expression

Operators: -

Vectors: nonymous Vector 1

Scalars: 0

Expression: [Anonymous Vector 1]<0

Description: Negative Value

Kst Debug Log

☑ Log as: ○ Notice ● Warning ○ Error

E-Mail Notification

□ Notify:

ELOG

□ Notify:

KstScript

□ Execute:

Expression The expression to monitor. You may type directly in this textbox, or select items to insert using the operator, vector, and scalar drop-down lists. Ensure that the expression entered in this textbox is a boolean expression (i.e. it evaluates to either true or false). This usually entails using an equality or inequality in the expression. Note that vectors entered in the textbox will be monitored according to their individual elements.

Whenever this expression is true (evaluates to non-zero), the event will be triggered. The action taken upon an event trigger depends on the settings specified in the other settings.

Description This textbox is used to store a short description of the event. The description is primarily available as a method for keeping track of multiple events. You can enter any text you wish in this textbox.

Log as: Enable this checkbox to have entries added to the Kst debug log when this event is triggered. There are three types of entries in the debug log — notices, warnings, and errors. Select the desired type of entry using the corresponding radio button.

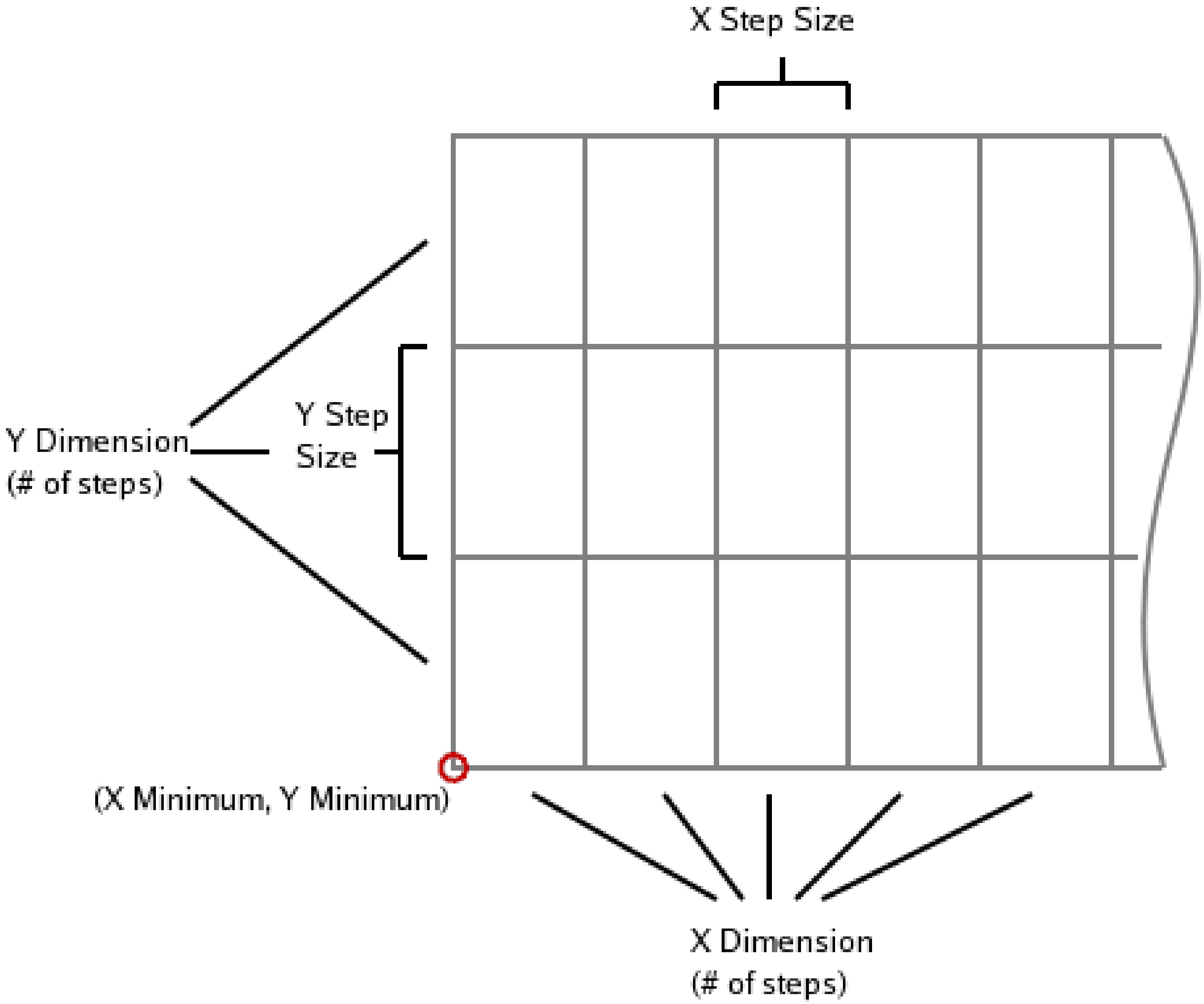
E-Mail Notify Enable this checkbox to have Kst send e-mail notifications to the specified address when this event is triggered. General E-mail configuration options are available in the Kst Settings dialog.

ELOG If the ELOG Extension is activated, then then Event Monitor can provide notification through ELOG. This can be useful for remotely monitoring live data, or for obtaining summaries of event activity. Please see the ELOG section for more information.

KstScript If this option is activated, then the Event Monitor can execute a KstScript when it triggers. Please see the KstScript documentation for more information.

3.2.9 Matrix

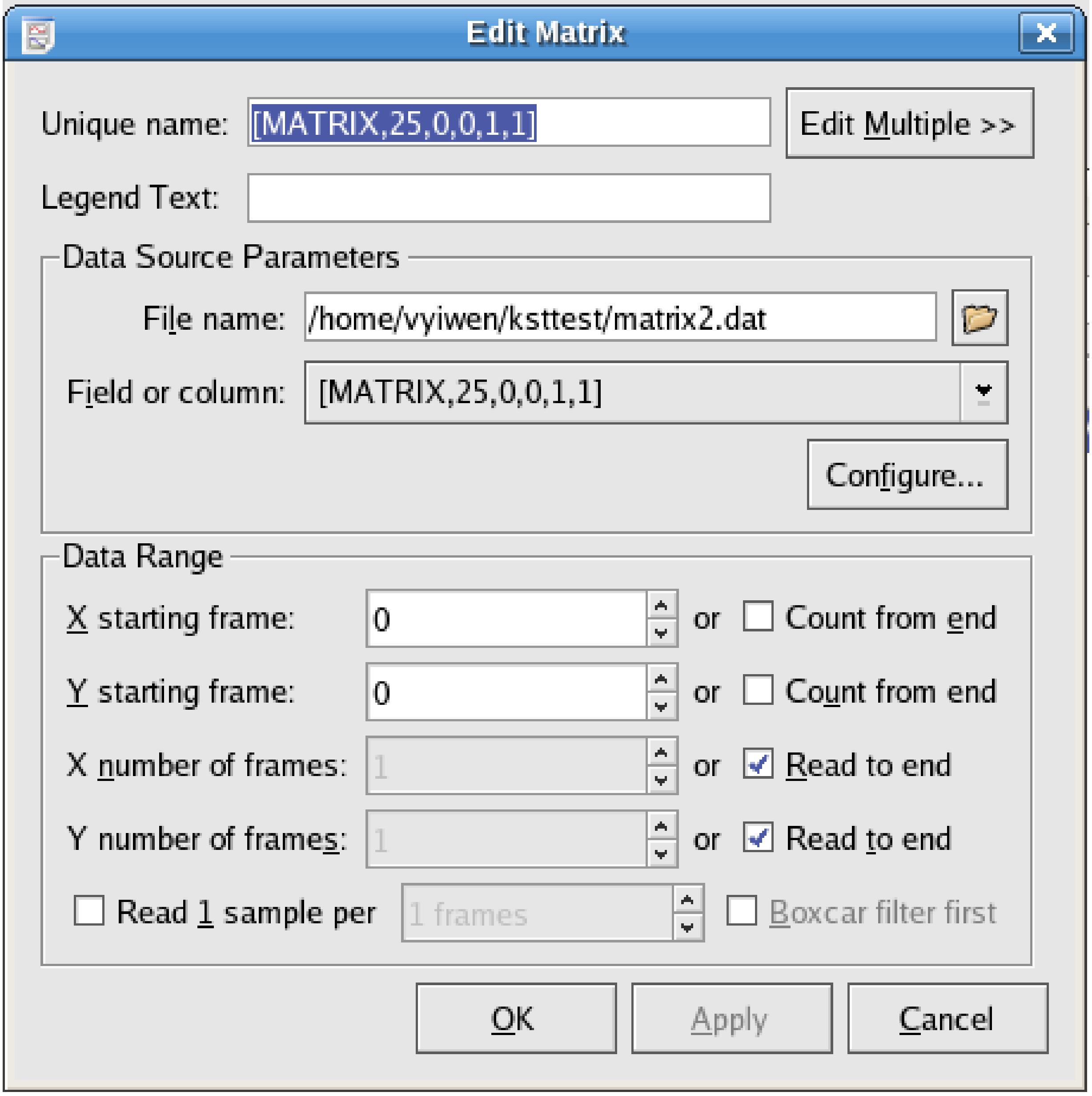
A matrix represents a set of three-dimensional data points (x, y, z) arranged in a two-dimensional grid.



Just as is the case with the vector datatype, there are two kinds of matrix: data and static. Data matrices are read in from data sources, and static matrices are generated by Kst.

3.2.9.1 Data Matrix

The following is a screenshot of the window displayed when editing a data matrix. A new data matrix is created if you choose the Read from data source radio button in the New Matrix dialog.



Unique name: [MATRIX,25,0,0,1,1]

Legend Text:

File name: /home/vyiwen/ksttest/matrix2.dat

Field or column: [MATRIX,25,0,0,1,1]

X starting frame: 0 or ☐ Count from end

Y starting frame: 0 or ☐ Count from end

X number of frames: 1 or ☑ Read to end

Y number of frames: 1 or ☑ Read to end

☐ Read 1 sample per 1 frames ☐ Boxcar filter first

All of the options which are available for data matrices are analogous to the options for Data Vectors.

Data Source Parameters Any field in ascii file that is going to be used as data source for matrix must have its field name in this format: [MATRIX, `# of rows, x min, y min, x step size, y step size`]. A very simple example of a matrix source file is provided below.

Data Range This option allows you to use only a portion of the matrix by specifying the range of data in both the x and y directions.

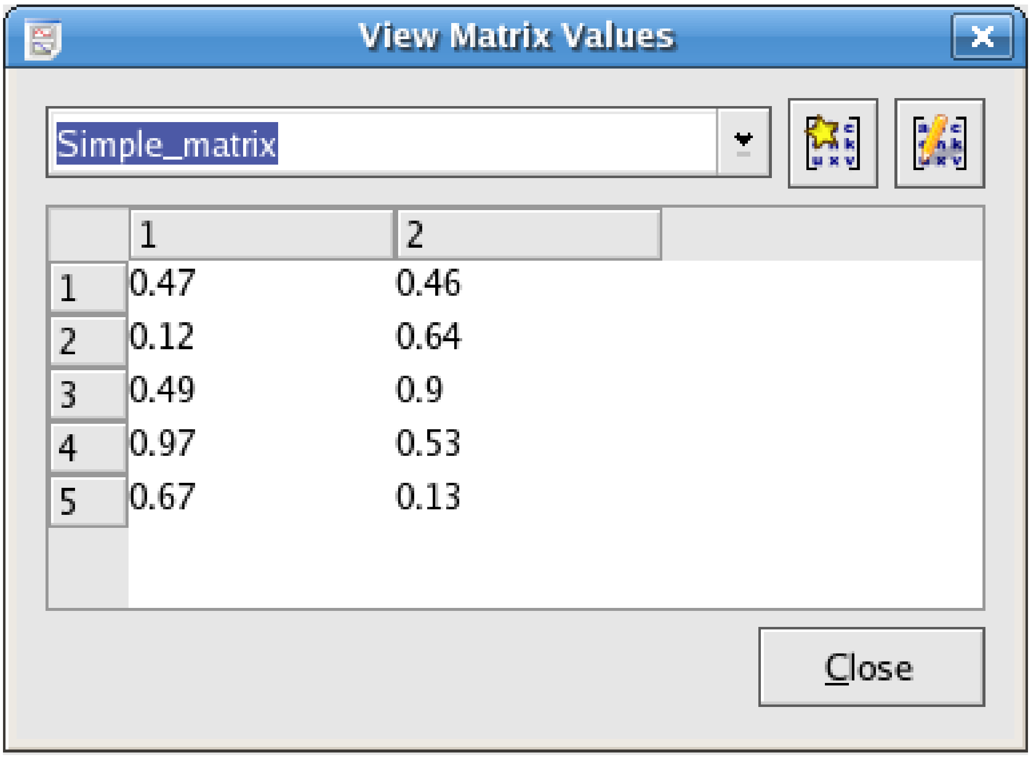
- X starting frame: the starting index in x dimension of the matrix
- Y starting frame: the starting index in y dimension of the matrix
- X number of frames: the number of data points in x dimension
- Y number of frames: the number of data points in y dimension

The simple example below shows how to use this option.

3.2.9.1.1 Example of a Simple Matrix Source File

```
[MATRIX,5,0,0,1,1]
0.47
0.12
0.49
0.97
0.67
0.46
0.64
0.9
0.53
0.13
```

The resulting matrix would be a 2×5 matrix, and the 10 samples are all included:



	1	2
1	0.47	0.46
2	0.12	0.64
3	0.49	0.9
4	0.97	0.53
5	0.67	0.13

Note
Notice that only the number of rows needs to be specified since Kst can figure out the number of columns automatically in order to have the loaded matrix use the largest number of samples in the original data field.
If the field name of the matrix is [MATRIX,6,0,0,1,1], the resulting matrix would have dimension 1×6; only 6 samples would be included.

3.2.9.1.2 Example of Specifying a portion of matrix

For the 2×5 matrix in the Screenshot above, if only part of the matrix:from index(1,2) to index(2,4), is needed; that is

0.12 0.64

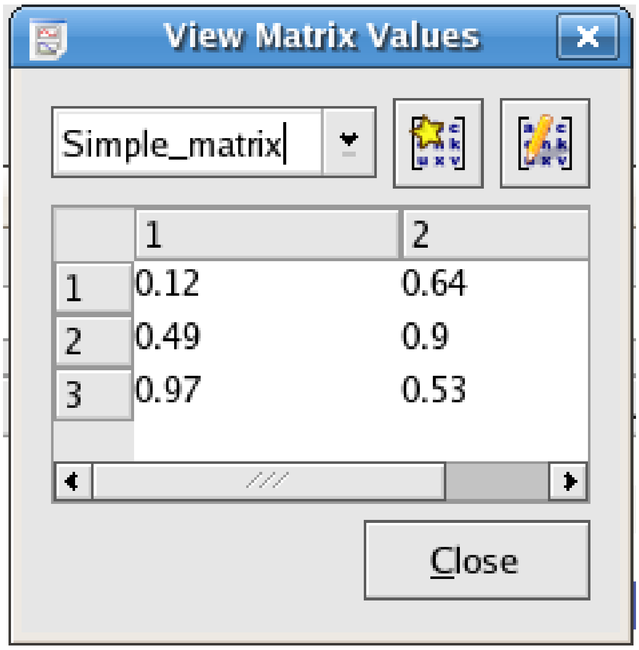
0.49 0.9

0.97 0.53

The Data Range needs to be specified in the following way:

- X starting frame: 0
- Y starting frame: 1
- X number of frames: 2
- Y number of frames: 3

The result is this:

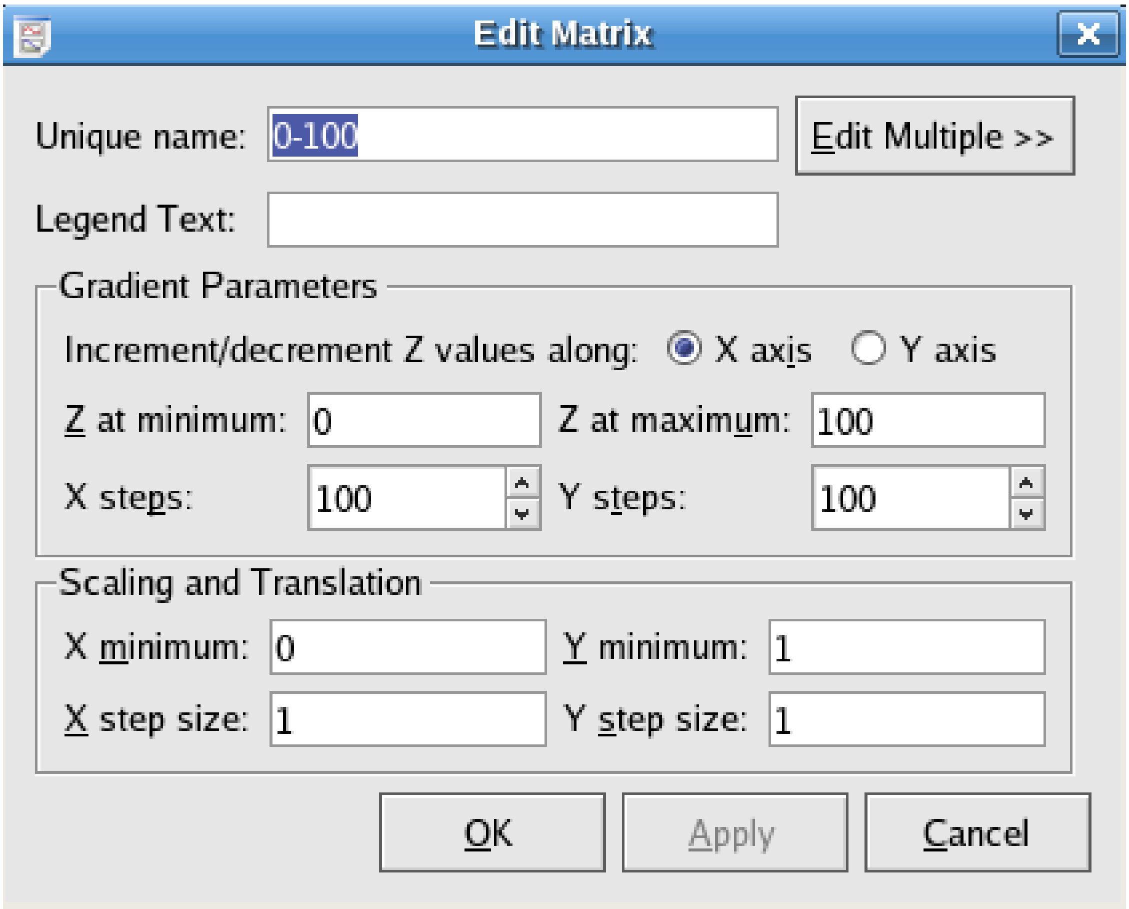


Simple_matrix

	1	2
1	0.12	0.64
2	0.49	0.9
3	0.97	0.53

3.2.9.2 Static Matrix

The following is a screenshot of the window displayed when editing static matrices. Static matrices allow you to create gradient matrices such as the one shown in the background of the annotation object example. A new static matrix is created if you choose the Generate gradient radio button in the New Matrix dialog.



Unique name: 0-100

Legend Text:

Increment/decrement Z values along: ● X axis ○ Y axis

Z at minimum: 0

Z at maximum: 100

X steps: 100

Y steps: 100

X minimum: 0

Y minimum: 1

X step size: 1

Y step size: 1

These options allow you to construct a matrix which has a linear gradient in Z along either the X or Y axis.

The steps options allow you to set the size of the matrix.

- X steps indicates column numbers
- Y steps indicates row numbers

The X/Y minimum and X/Y step size options allow you to set the X and Y ranges.

3.2.9.3 Using Matrices through KstScript

Matrices can also be constructed by KstScript. See details in the Matrix class and the DataMatrix Class of the Script Chapter.

Example 3.8 Example of Using the DataMatrix Class in KstScript

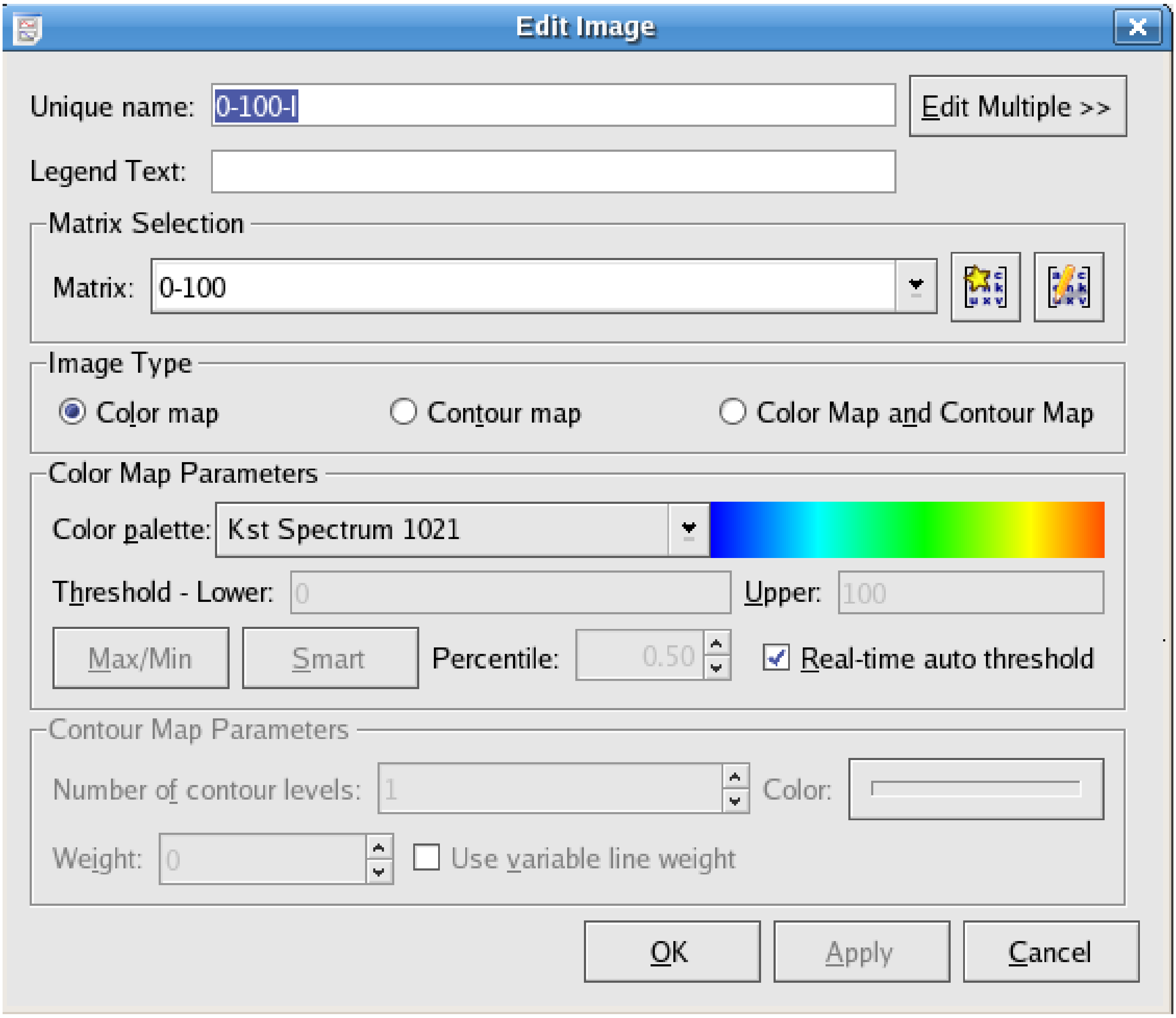
```
s = new DataSource("home/vyiwen/mx.dat")//mx.dat is the same matrix source file in the  ↩
    example source file above
m = new DataMatrix(s, "[MATRIX,5,0,0,1,1]")//use mx.dat to construct a matrix
m.tagName = "Simple_matrix";//Name the matrix Simple_matrix and you should be able to see  ↩
    it in the Data Manager.

//if you want to specify a portion of matrix by specifying the data range in the x, y  ↩
    dimension,
//use the method: change(xStart, yStart, xFrames, yFrames)

m.change(0,1,2,3)//you can see the change in Simple_matrix
                 // it should be the same 2 by 3 matrix
                 // as in the example of specifying a portion of matrix above.
```

Example 3.9 Example of Using the Matrix Class to construct a Static Matrix

```
y = new Matrix(); // to construct a new matrix
y.resize(3,5);  //define the size of the matrix 3 by 5 matrix
                //i.e 3 rows and 5 columns
for(var i=0; i<5;i++){for(var j=0; j<3;j++){y.setValue(i,j,i+j)}}; //to initialize the  ↩
    matrix
y.tagName="mx"; //you should see matrix mx in the Data Manager now
```

3.2.10 Images

Images are graphical representations of Matrices. Images can be plotted as color maps, contour maps, or both.



Edit Image

Unique name: 0-100-I

Legend Text:

Matrix Selection

Matrix: 0-100

Image Type

● Color map ○ Contour map ○ Color Map and Contour Map

Color Map Parameters

Color palette: Kst Spectrum 1021

Threshold - Lower: 0 Upper: 100

Percentile: 0.50 ☑ Real-time auto threshold

Contour Map Parameters

Number of contour levels: 1 Color:

Weight: 0 □ Use variable line weight

Matrix Selection Select the matrix to use for this image. New matrices can be created or edited by clicking buttons to the right.

Color Map, Contour Map, and Color Map and Contour Map Select the type of image to be plotted. Changing this selection enables or disables sections of the image dialog as appropriate. A color map represents the Z value of each cell in the matrix using a color. A contour map plots curves which follow lines of constant Z.

Color palette Select the color palette to use for the color map.

Threshold - Lower and Upper Enter the lower and upper thresholds to use for the mapping of colors. Palette colors are evenly distributed between the lower and upper thresholds. Any cells in the selected matrix with z values less than the lower threshold are mapped to the first color in the palette. Any cells in the selected matrix with z values greater than the upper threshold are mapped to the last color in the palette.

Max/Min Clicking this button causes the lower and upper threshold values to be set to the current minimum and maximum z values of the selected matrix.

Smart and Percentile Clicking the Smart button causes the lower and upper threshold values to be set such that the percentage of z values in the selected matrix contained between them is equal to the value in the Percentile numberbox.

Real-time auto threshold Selecting this option causes the lower and upper threshold values to be always set to the minimum and maximum of the z values for the selected matrix, even when the matrix updates.

Number of contour levels Select the number of contour levels to use. The contour levels will be distributed evenly between the lowest and highest z values found in the matrix.

Color Select the color to use for the contour lines. Clicking this button displays a standard KDE color chooser dialog.

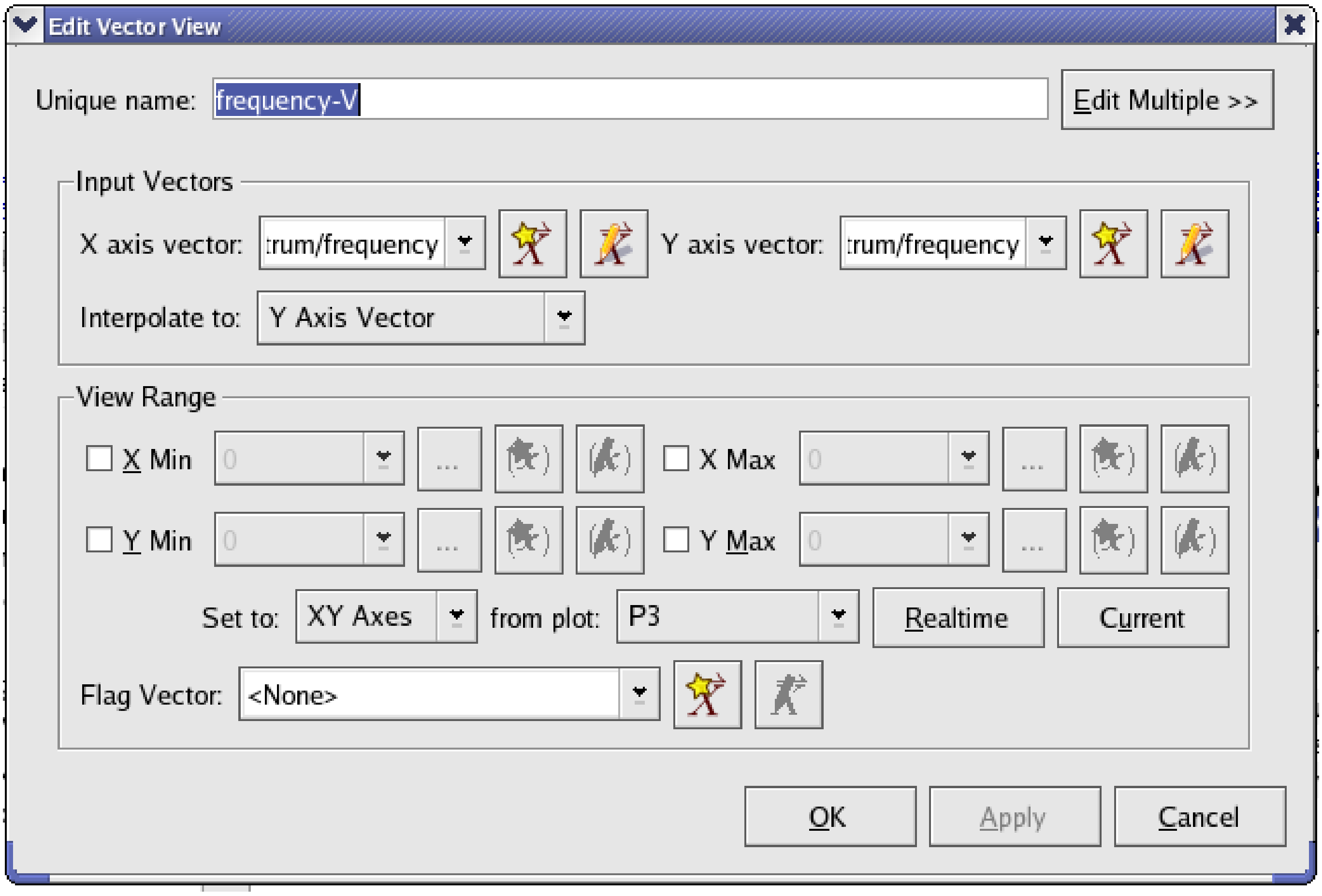
Weight and Use variable line weight Select the Weight, or ‘thickness’ of the contour lines. If Use variable line weight is selected, contour lines representing higher elevations are drawn thicker than those representing lower elevations. Use discretion when selecting this option, as images with high contour line densities may become unreadable.

3.2.10.1 The Image Class in KstScript

See the Image Class of the Script Chapter for detail.

3.2.11 Vector View

Vector view provides a way to view a portion of the data plot. It excludes any data point which is not in the range of the X/Y limits specified by the user.



Edit Vector View

Unique name: frequency-V

Input Vectors

X axis vector: :rum/frequency

Y axis vector: :rum/frequency

Interpolate to: Y Axis Vector

View Range

□ X Min 0

□ X Max 0

□ Y Min 0

□ Y Max 0

Set to: XY Axes from plot: P3

Flag Vector: <None>

Input Vectors Select an input X/Y axis vector from existing vectors or create one by using the shortcut icons.

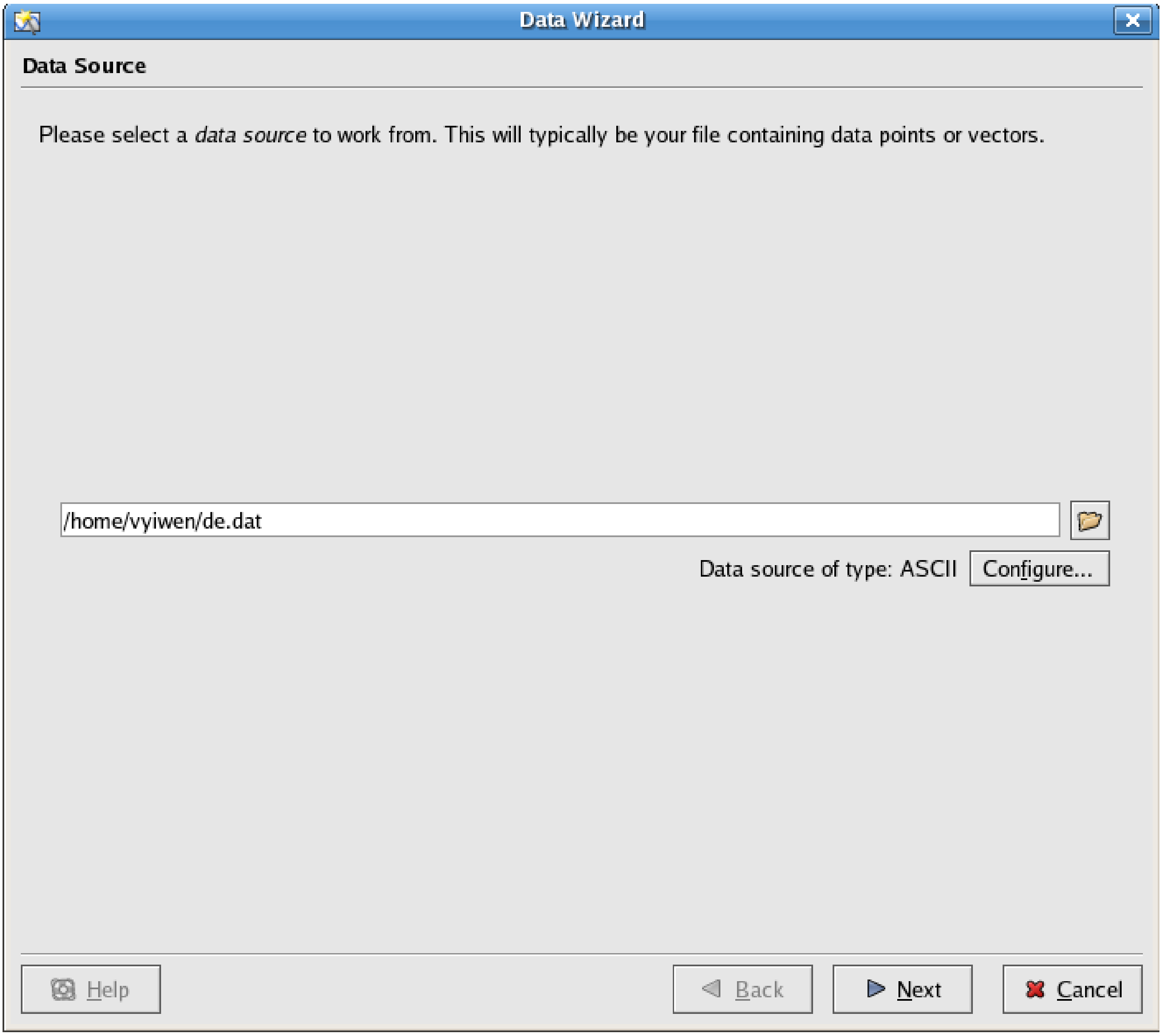
View Ranges Adjust the view range of the plot by selecting X min/Y min and X max/Y max values for the input vectors.

Flag Vector This option is used to exclude specific points in the input vector for the vector view. The indices of non-zero points in the flag vector indicate that the data points of the input vector with the same indices should be filtered out for its vector view.

3.3 Data Tools

3.3.1 The Data Wizard

The Data Wizard provides a graphical quick and easy way of importing data into Kst- automatically creating vectors, curves, power spectra, and plots. To launch the wizard, select Tools → Data Wizard or click on the icon in the toolbar. The Data Source pane will appear as following and you will need to select a data source file.



Please select a *data source* to work from. This will typically be your file containing data points or vectors.

/home/vyiwen/de.dat

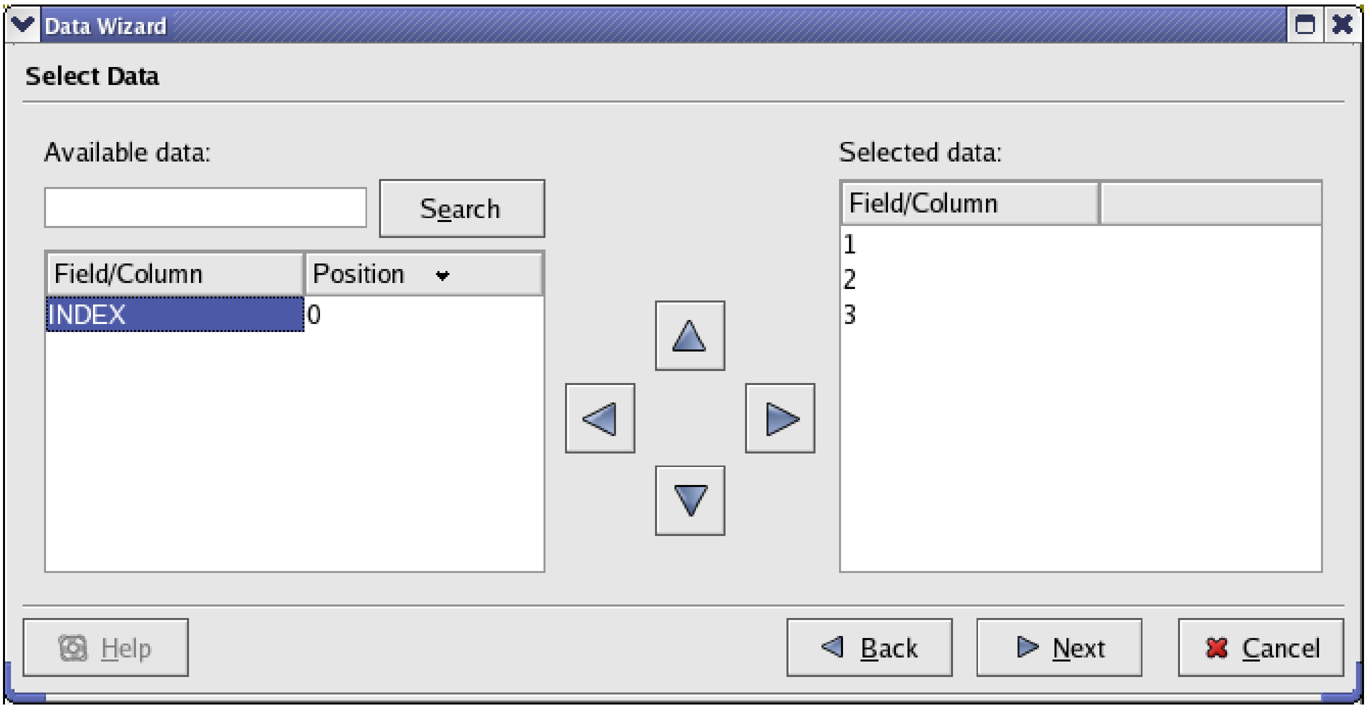
Data source of type: ASCII

3.3.1.1 Configuring Data Source

After selecting the data source file, you may need to configure the file by clicking on the Configure... button. For information on how to use the Configure Data Source dialog, see the ASCII datasource section.

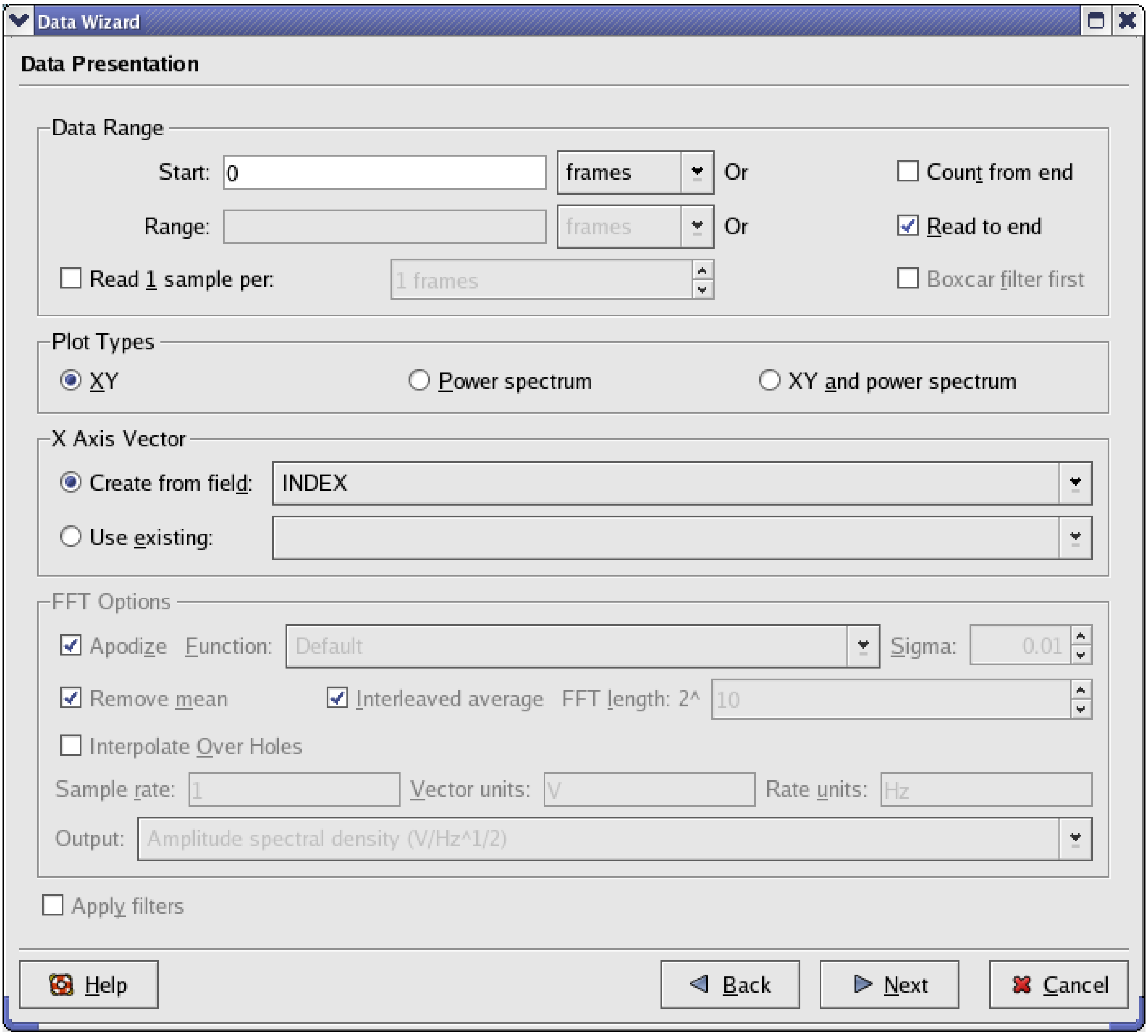
After configuring the data source file, click on the OK button to close the Configure Data Source dialog, and click on the Next button in the Data Source pane. The Select Data pane will appear as shown below.

3.3.1.2 Selecting Data



Here you can select the fields you wish to import into Kst. The right and left arrow buttons allow you to move data vectors back and forth between the Available data and the Selected data lists. The up and down arrow buttons allow you to move the positions of the selected vectors up and down in the Selected data list. You may filter the list of fields by entering a string to match (wildcards such as $\star$ are supported) into the text box above the list. Click the Next button to advance to the Data Presentation pane.

3.3.1.3 Presentation



Data Wizard

Data Presentation

Data Range

Start: 0 frames Or □ Count from end

Range: frames Or ☑ Read to end

□ Read 1 sample per: 1 frames □ Boxcar filter first

Plot Types

● XY ○ Power spectrum ○ XY and power spectrum

X Axis Vector

● Create from field: INDEX

○ Use existing:

FFT Options

☑ Apodize Function: Default Sigma: 0.01

☑ Remove mean ☑ Interleaved average FFT length: 2^ 10

□ Interpolate Over Holes

Sample rate: 1 Vector units: V Rate units: Hz

Output: Amplitude spectral density (V/Hz^1/2)

□ Apply filters

The Data Range section is used to specify the range of data to read from the selected vectors in the input file. For information about these parameters, read the description of the Vector data object.

Power Spectrum and X axis settings can be specified within the Plot Types section. These settings are described below.

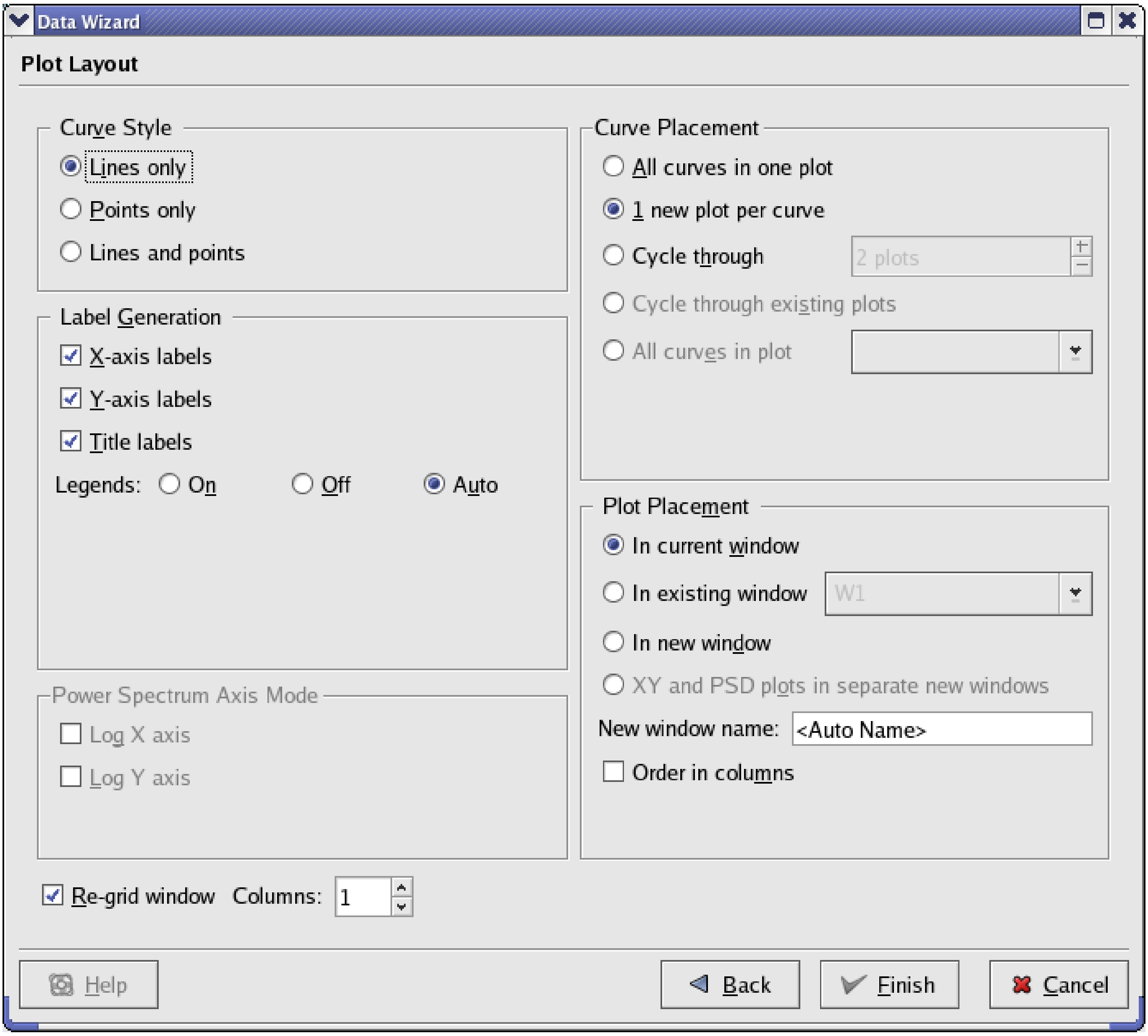
XY, Power Spectrum, and XY and Power Spectrum Select whether to plot the data (XY) only, Power spectrum only, or both. If the power spectrum is selected for plotting, additional settings in this section become available.

X Axis Vector The vector to be used as the independent vector for the plots. You may select a field from your data file, or the INDEX vector. The INDEX vector is simply a vector containing elements from 0 to N-1, where N is the number of frames in the data file.

The FFT Options subsection in the Plot Types section is available only if a power spectrum is to be plotted. Detailed information on these options can be found in the Spectrum section.

Once you are satisfied with all the settings, click the Next button to advance to the final window.

3.3.1.4 Configure Plot Layout



Plot Layout

Curve Style
● Lines only
○ Points only
○ Lines and points

Label Generation
☑ X-axis labels
☑ Y-axis labels
☑ Title labels
Legends: ○ On ○ Off ● Auto

Power Spectrum Axis Mode
☐ Log X axis
☐ Log Y axis

Curve Placement
○ All curves in one plot
● 1 new plot per curve
○ Cycle through 2 plots
○ Cycle through existing plots
○ All curves in plot

Plot Placement
● In current window
○ In existing window W1
○ In new window
○ XY and PSD plots in separate new windows
New window name: <Auto Name>
☐ Order in columns

☑ Re-grid window Columns: 1

From here, you can change general plot settings. Most of the settings are self-explanatory. A brief overview of each section is provided below.

Curve Style Select whether to plot data points only, lines connecting the data points only, or both. By default, lines are continuous with weight 0, and data points are marked using crosses. Line and data point colors are chosen so that curves with identical colors are minimized. Note that the curve style settings apply to both Power Spectra and XY plots.

Curve Placement Select the plots to place the new curves on. Cycle through distributes the curves on the plots by cycling through the plots.

For example, if you have 4 curves, and you choose to cycle through 2 plots. The result will be that curve 1 is placed in plot 1; curve 2 is in plot 2, and this is the first plot cycle. Curve 3 is back to plot 1; curve 4 goes to plot 2, and this is the second cycle. Note that the curve placement settings apply to both Power Spectra and XY plots.

Label Generation Select the desired labels and legends to be placed on the plots.

Plot Placement Select the desired window(s) to place the new plots in. New windows can be created for the plots by selecting In new window.

Power Spectrum Axis Mode Select these check-boxes if you would like the automatically created power spectra to have logarithmic X or Y axes.

Once you are satisfied with all the settings, click the Finish button and the plots will be generated.

3.3.2 The Data Menu

The Data menu provides quick access to many features related to data objects in Kst. Most of the menu functions duplicate functions found elsewhere, so only brief descriptions are provided below.

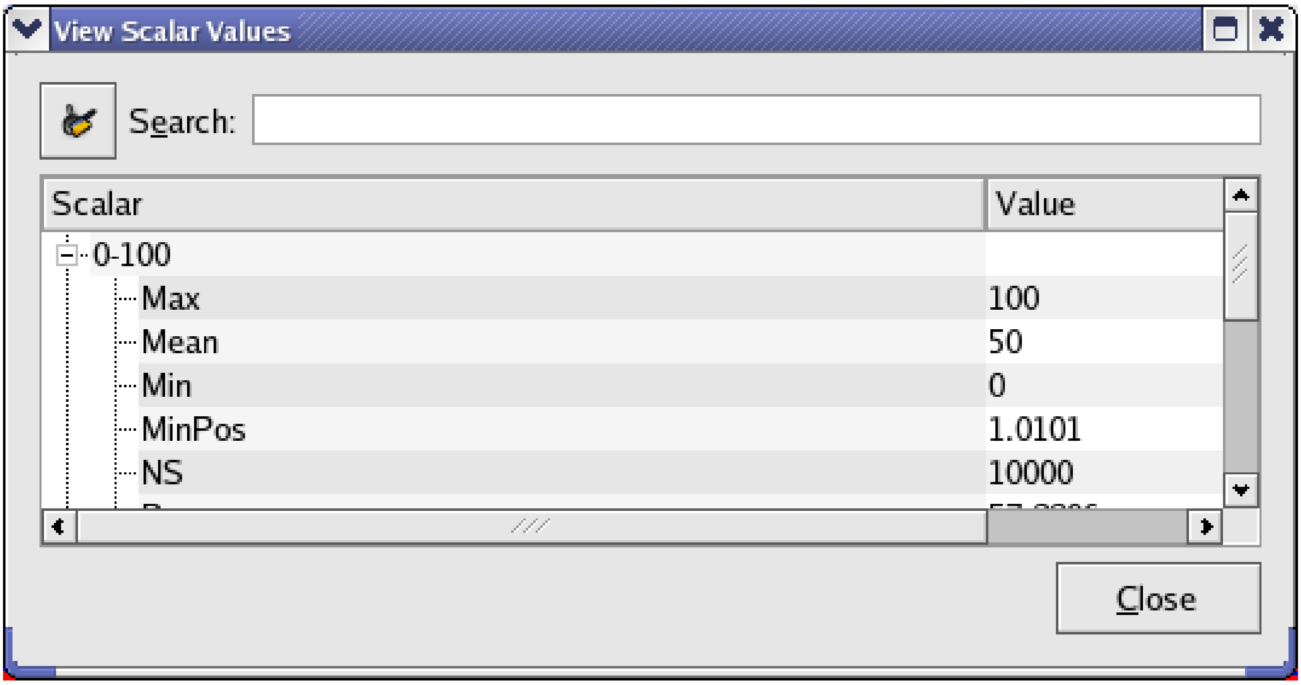
Reload Reloads all data vectors from their source files.

Data Manager Displays the Data Manager.

View Manager Displays the View Manager.

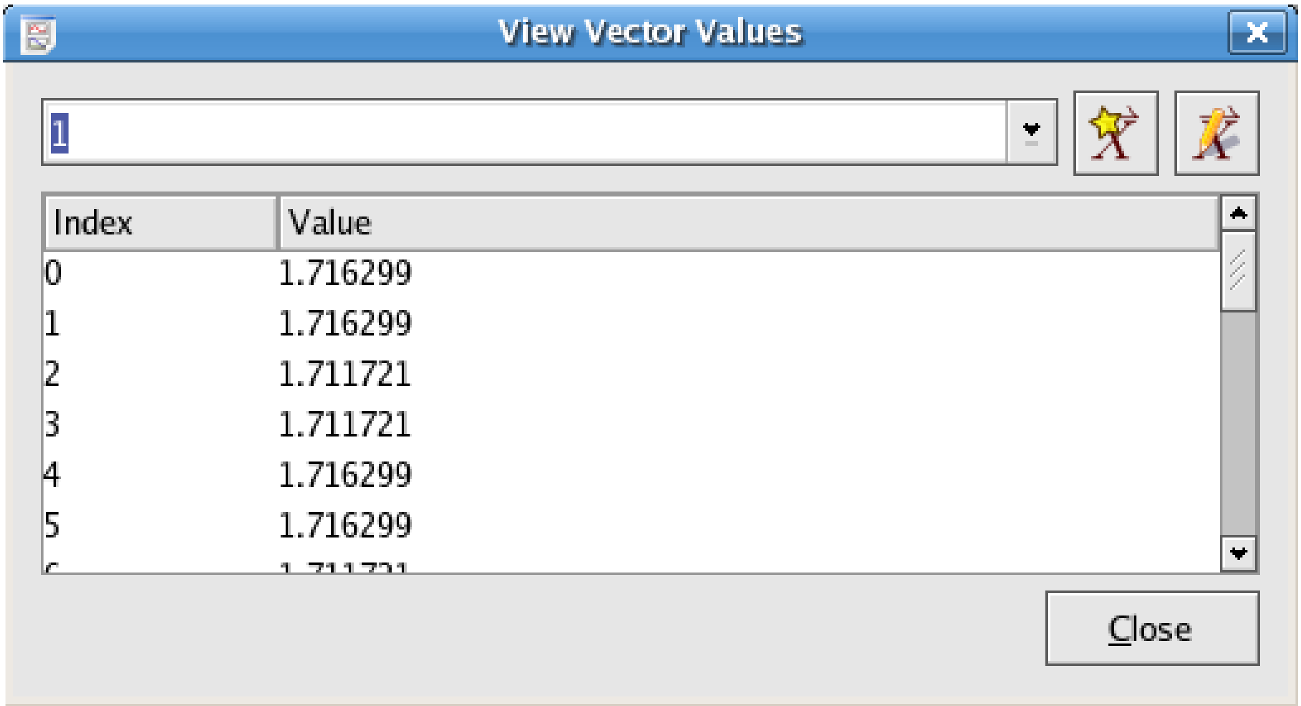
New [data object] Displays the corresponding dialog for creating the data object. Refer to Data Types for descriptions of each dialog.

View Scalar Values Displays a dialog from which the values of all the scalars in the current Kst session can be viewed. The values are displayed hierarchically, based on which higher level data objects the scalars are determined for. The dialog is updated dynamically if the values change.



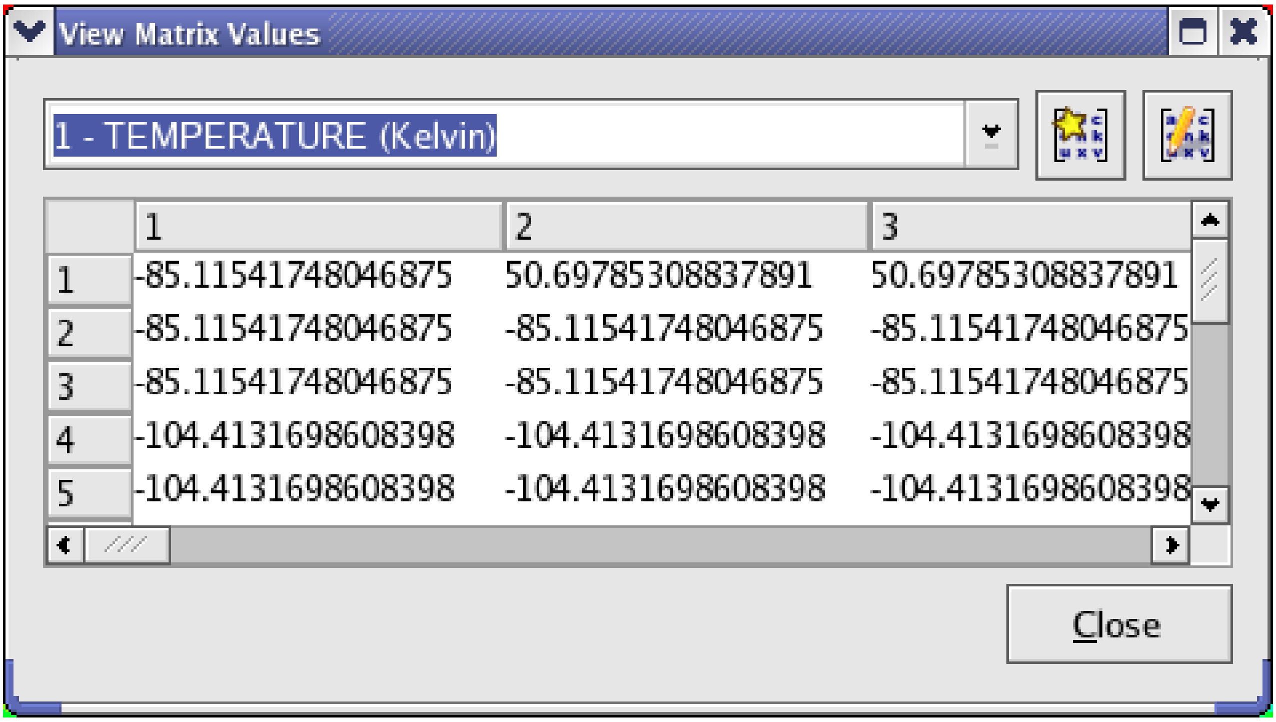
Scalar	Value
0-100	
Max	100
Mean	50
Min	0
MinPos	1.0101
NS	10000

View Vector Values Displays a dialog from which the values in all the current vectors can be viewed. Select a vector to view using the drop-down list. The dialog is updated dynamically if the values change.



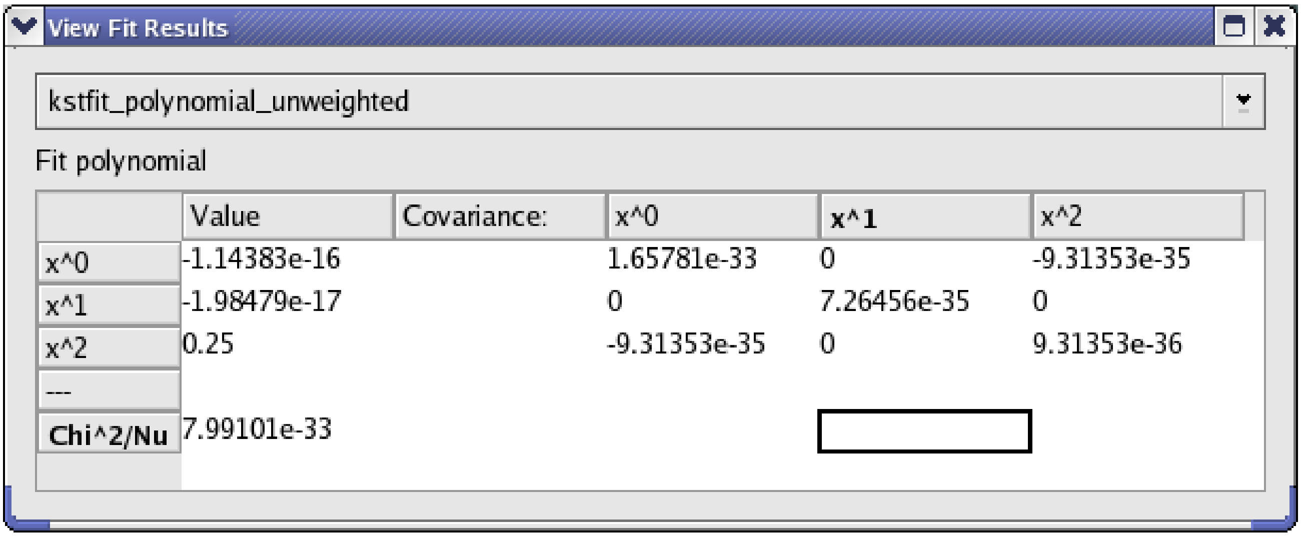
Index	Value
0	1.716299
1	1.716299
2	1.711721
3	1.711721
4	1.716299
5	1.716299

View Matrix Values Displays a dialog from which the values in all the current matrices can be viewed. Select a matrix to view using the drop-down list. The dialog is updated dynamically if the values change.



	1	2	3
1	-85.11541748046875	50.69785308837891	50.69785308837891
2	-85.11541748046875	-85.11541748046875	-85.11541748046875
3	-85.11541748046875	-85.11541748046875	-85.11541748046875
4	-104.4131698608398	-104.4131698608398	-104.4131698608398
5	-104.4131698608398	-104.4131698608398	-104.4131698608398

View Fit Results Displays a dialog which shows all the resulting values from fit plugins. Select a fit result to view using the drop-down list. The dialog is updated dynamically if the values change.



kstfit_polynomial_unweighted

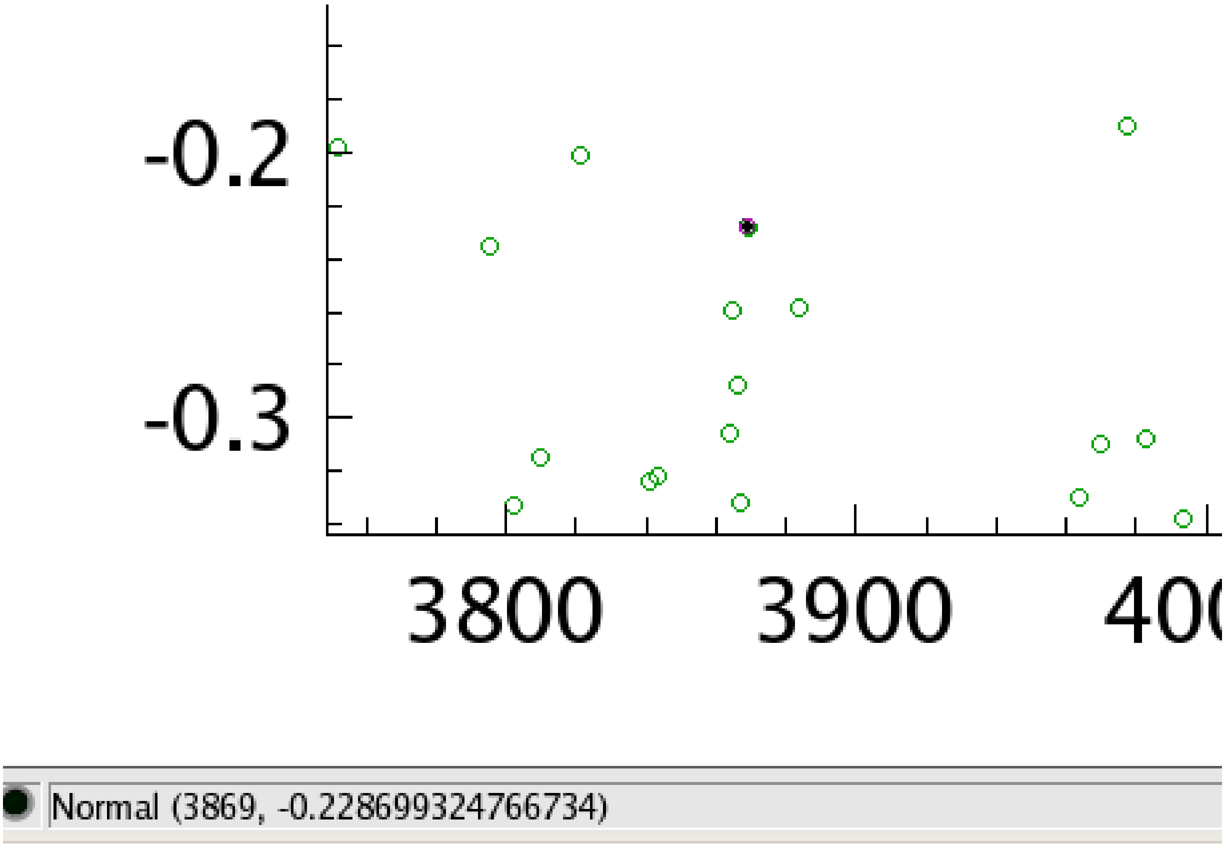
Fit polynomial

	Value	Covariance:	x^0	x^1	x^2
x^0	-1.14383e-16		1.65781e-33	0	-9.31353e-35
x^1	-1.98479e-17		0	7.26456e-35	0
x^2	0.25		-9.31353e-35	0	9.31353e-36

Chi^2/Nu	7.99101e-33				

3.3.3 Data View Mode

Data mode allows precise viewing of the data points used in a plotted data object. To toggle data mode, select Plots → Data Mode, or click the Data Mode icon on the Toolbar. Now, when the cursor is moved over a plot, a colored dot will indicate the closest data point to the cursor, as shown in the screenshot below. The status bar will display the coordinates of the data point (in terms of the x and y vectors used to plot the data object) in the status bar at the lower left corner of the Kst window. The status bar will also display the x, y, and z coordinates of any visible image. If images overlap, only the coordinates of the topmost image will be displayed. Note that all zooming functions are still available while in data mode.



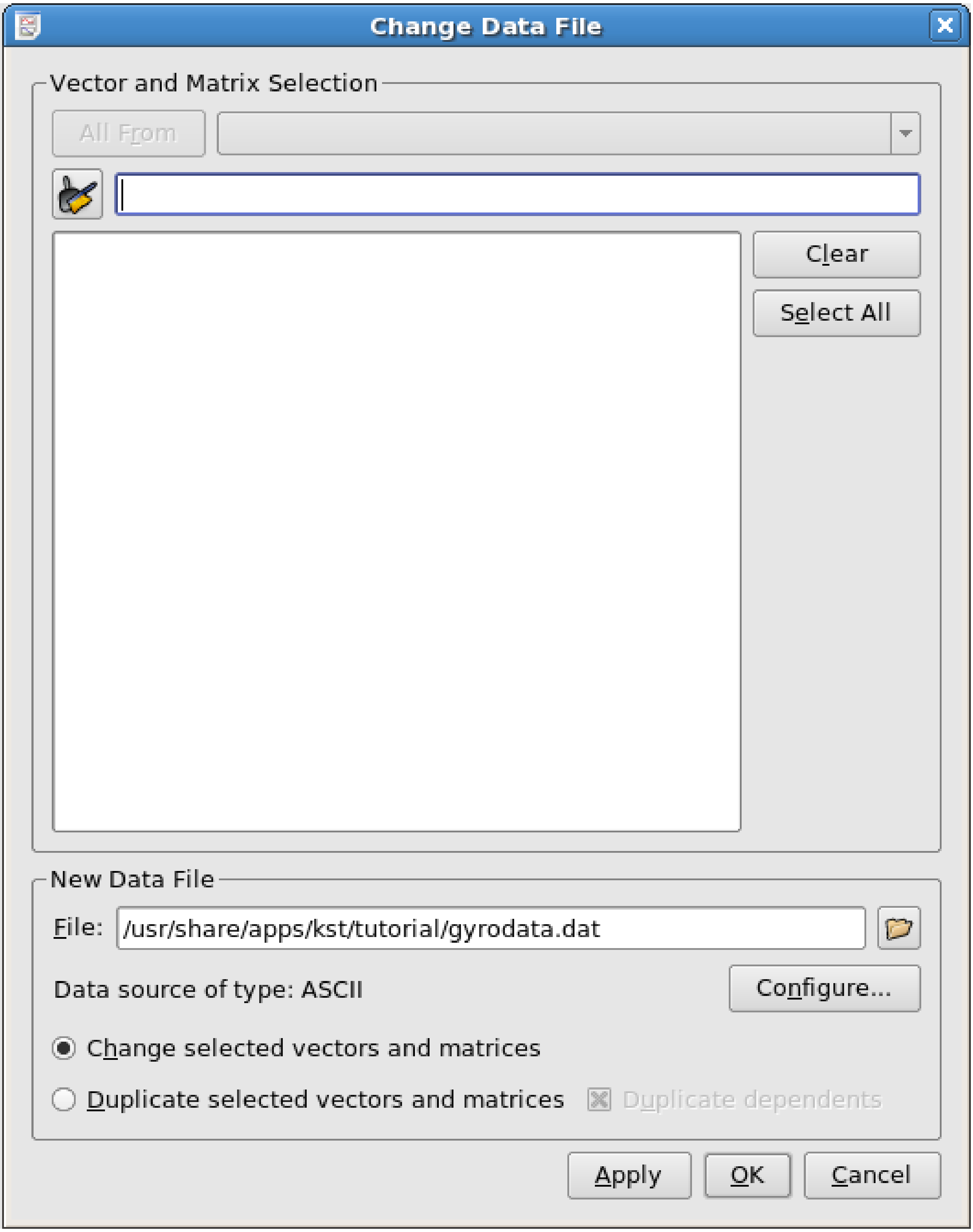
Normal (3869, -0.228699324766734)

Tip
If the status bar is not visible, select Settings → Show Statusbar

3.3.4 Change Data File

One common mode of using Kst is to create a ‘session’ of plots and analysis for a particular type of data. Then, the same plots and analysis can be performed easily on another set of data simply by changing the data source file, provided that the two data files are similar. For example, for ASCII data files, similar data files need to have same number of fields.

The Change Data File dialog can be accessed by choosing Tools → Change Data File or clicking on the icon in the toolbar. The dialog which appears is shown below:



File: /usr/share/apps/kst/tutorial/gyrodata.dat

Data source of type: ASCII

● Change selected vectors and matrices

○ Duplicate selected vectors and matrices ☒ Duplicate dependents

To use this dialog, select the vectors which you would like to change the data file of using the list in the middle of the dialog.

The All From, Clear, Select All, and search field can help you to do this. The use of wildcards is supported in the search field.

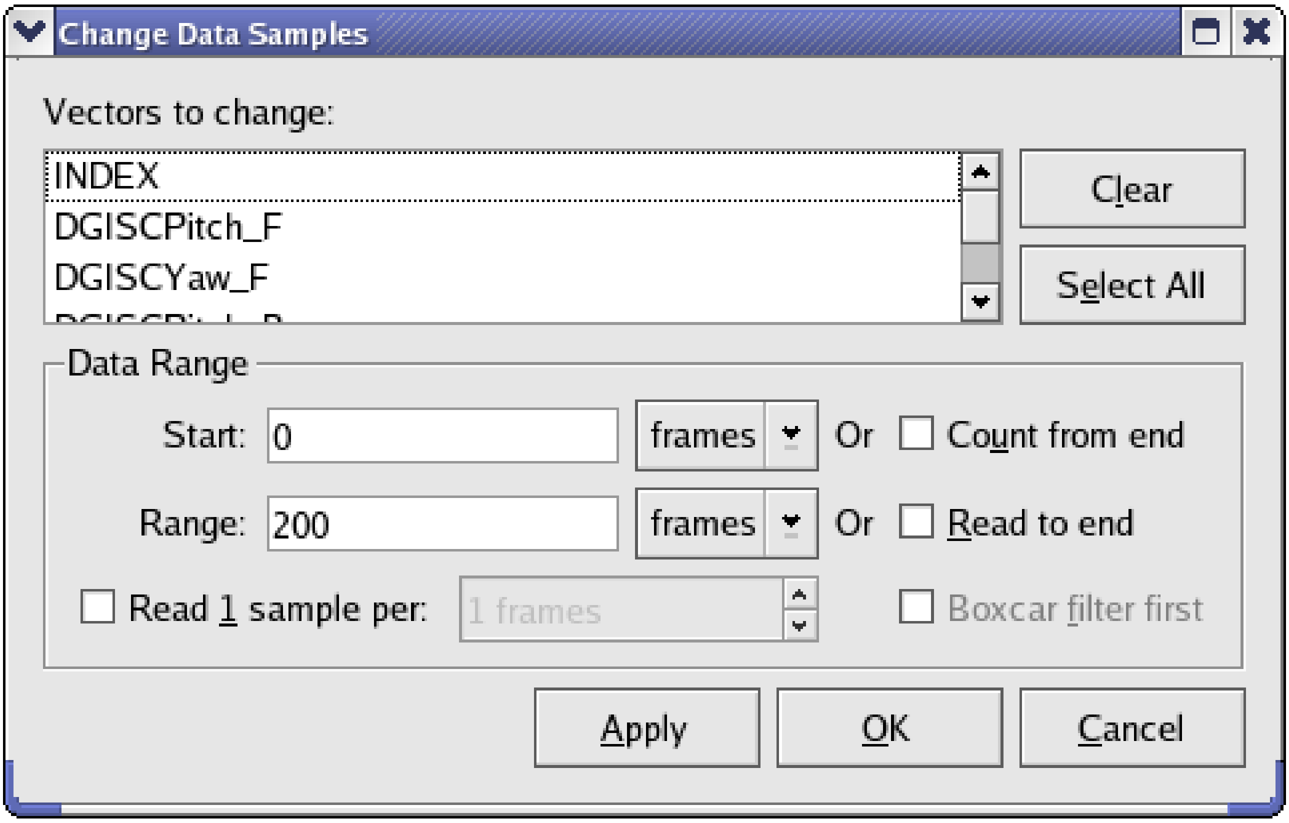
Then, select a new data file and press Configure button to configure data source. If the Change selected vectors and matrices radio button is selected, then when you press the Apply or OK button, the plots of the selected data vectors will be replaced by the data vectors loaded from the same fields of the new data file that you specified.

If the Duplicate selected vectors and matrices radio button is selected, vectors coming from the new file will be plotted together with the existing vectors in exactly the same way.

If some of the selected vectors have dependents (curves, spectra, histograms, etc.) then the Duplicate dependents option allows you to duplicate these objects as well.

3.3.5 Change Data Sample Ranges

The Change Data Sample Ranges dialog can be accessed through the Tools menu or the icon in the toolbar. It allows you to modify how a set of vectors are read in from their associated files. The options are detailed in the Data Range option of the Data Vectors documentation.



Change Data Samples

Vectors to change:

INDEX
DGISCPitch_F
DGISCYaw_F

Data Range

Start: 0 frames Or □ Count from end

Range: 200 frames Or □ Read to end

□ Read 1 sample per: 1 frames □ Boxcar filter first

3.3.6 Common Icons

As mentioned above, most data tools have corresponding icons in the toolbar. There are also some useful icons commonly seen in various dialogs.

This icon opens the New Vector dialog, the same as choosing Data → New Vector...

This icon opens the Edit Vector dialog where you can edit the properties of the currently selected vector.

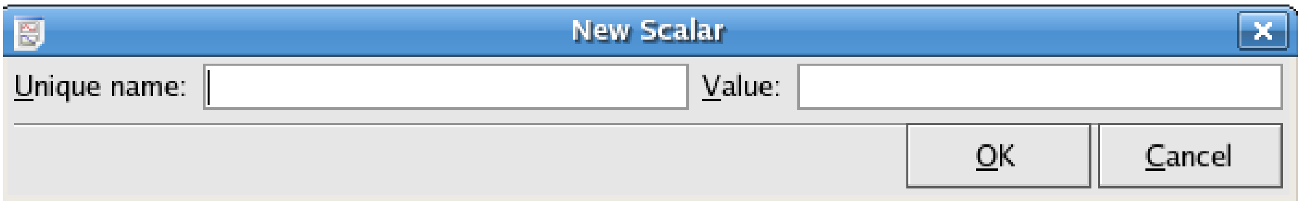
This icon opens the New Matrix dialog, the same as choosing Data → New Matrix...

This icon opens the Edit Matrix dialog where you can edit the properties of the currently selected matrix.

This icon is commonly seen in the search field of many dialogs used to list data objects (vectors, curves, etc). For example, it appears in the Data Manager, View Scalar Values, and Change Data File, etc.

It is used to quickly clear the textbox of the Search field and restore the original list of data objects.

This icon opens the New Scalar dialog as shown below:



Here, you can define your own scalar by typing its name and associated value.

This icon is used to change values of user-defined scalars. Essentially, it opens the same dialog as shown above.

3.4 Datasources

Kst uses externally linked datasources to read from data files. This means that new data formats can be transparently added to the Kst repertoire. Currently, the Kst distribution includes the following default datasources:

- ASCII
- Dirfiles
- HEALPix
- QImage
- Indirect
- CDF
- netCDF
- LFIIO
- SCUBA

If you have a data format which you would like Kst to read, it's easy to create your own.

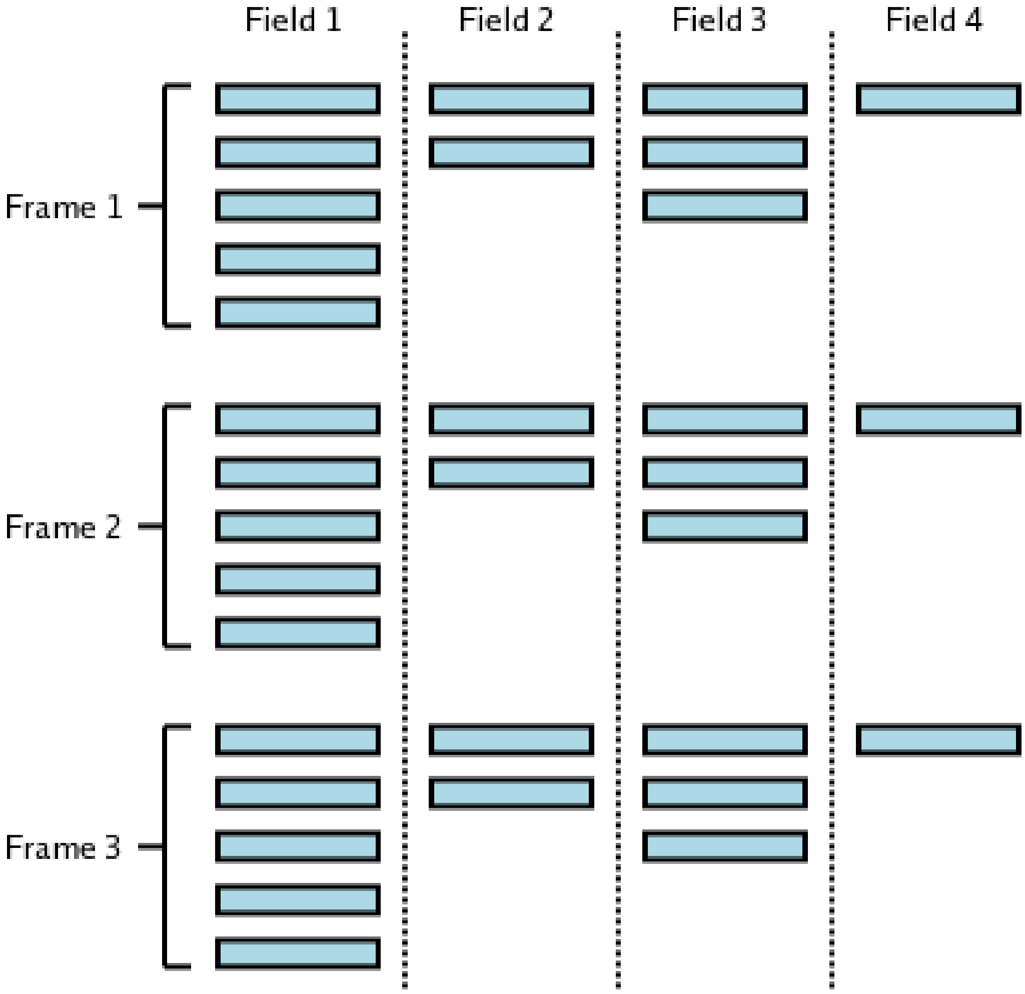
All of the data sources read from files using KIO, which is usually able to load data through common file compression formats (e.g. BZIP2, GZIP, ZIP, TAR) and most popular networking protocols (e.g. HTTP, FTP, SFTP, SMB). For more information on which formats are supported on your system type **`kinfocenter`** at the command line and inspect the protocols tab. If Kst detects that you have typed a URL into the filename of a vector or matrix, for example, a 'Connect' button will appear. Clicking the Connect button instructs Kst to load the indicated data, and channel it to a suitable data source.

The following concepts are important in understanding how Kst works with data sources. Some terminology is also introduced in this section.

Samples A sample is considered to be the fundamental unit with respect to data files. Each sample consists of one data value in the file. Note, however, that one sample may not correspond to one value in a data vector in Kst, due to the concept of frames.

Fields A field usually corresponds to one vector in Kst. For example, a column in an ASCII data file is considered to be one field. Each field can have an explicit or implicit name. Datasource readers provide functions for reading and obtaining fields and field names.

Frames A frame corresponds to a set number of samples, and each field in a data file can have its own ratio of samples per frame. The size of a data file is measured by the number of frames it contains. For ASCII data files, the number of samples per frame is 1, but for some data files, there may be multiple samples per frame. In the below illustration, the first 3 frames of an imaginary data file are shown. In this particular data file, Field 1 has a ratio of 5 samples per frame, Field 2 has a ratio of 2 samples per frame, Field 3 has a ratio of 3 samples per frame, and Field 4 has a ratio of 1 sample per frame.



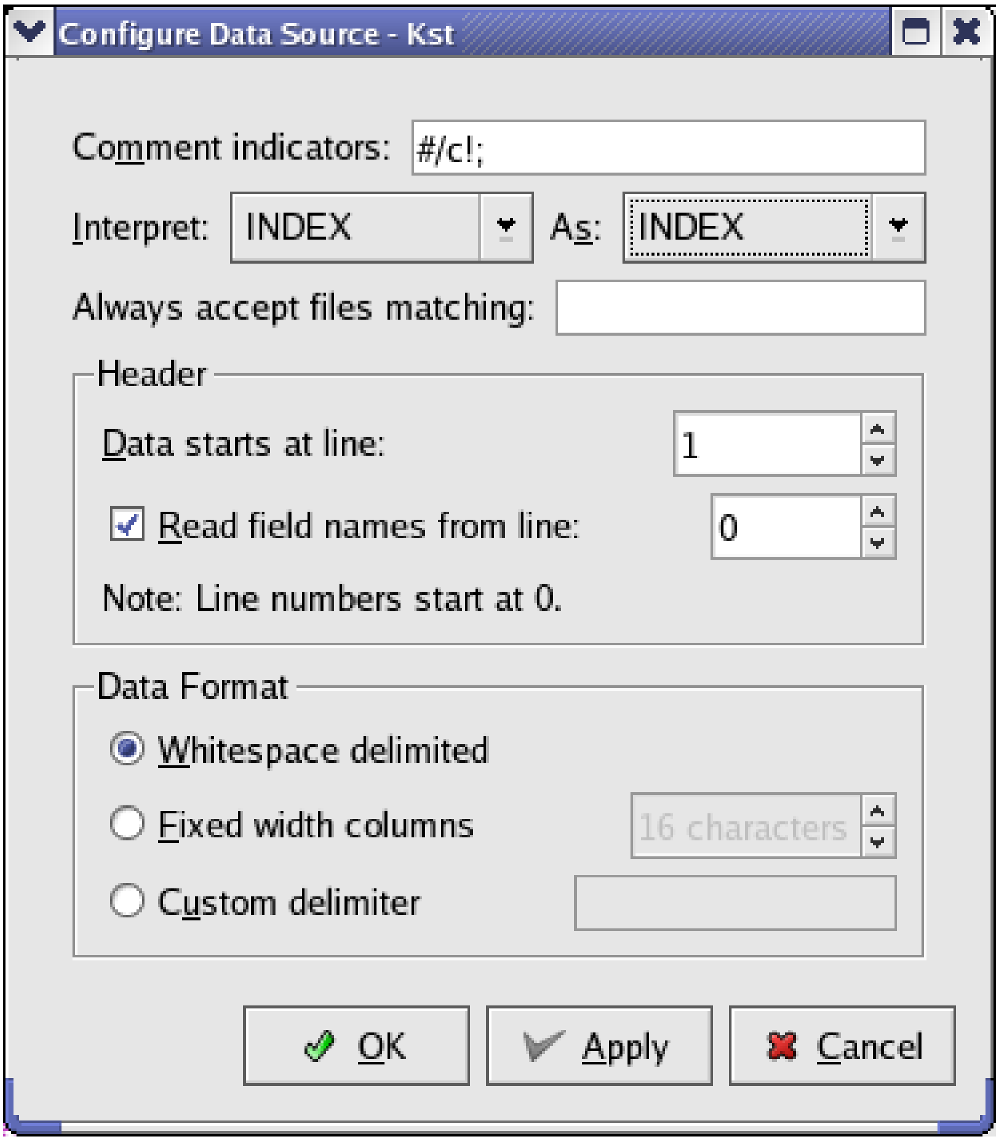
Depending on the specific data vector settings in Kst, data from files may be read as frames instead of samples, with either the first sample in a frame or the mean of all the samples in a frame representing the value of the frame. This capability can be useful for reducing extremely large data-sets to a more manageable size.

INDEX field Some data files may not have a field that represents the x axis of a plot. However, Kst implicitly creates an INDEX field for all data sources. The INDEX field simply contains integers from 0 to N-1, where N is the number of frames in the data file.

3.4.1 ASCII

The simplest input file format is the ASCII text file. These files are usually human-readable and can be easily created by hand or simple scripts if desired. The following is an example of an ASCII input file.

```
112.5 3776 428
187.5 5380 429
262.5 5245 345
337.5 2942 184
412.5 1861 119
487.5 2424 138
567.5 2520 162
637.5 1868 144
712.5 1736 211
787.5 1736 211
862.5 2172 292
937.5 1174 377
1000.5 499 623
```



Comment indicators: #/c!;

Interpret: INDEX As: INDEX

Always accept files matching:

Header

Data starts at line: 1

☑ Read field names from line: 0

Note: Line numbers start at 0.

Data Format

● Whitespace delimited

○ Fixed width columns 16 characters

○ Custom delimiter

Comment Indicators By default, commented lines in ASCII files start with one of the characters in this set `{#, !, /, ;, c}`. All commented lines and empty lines are ignored by Kst. Valid numbers include those with decimals, negative signs, or `e`, to indicate scientific notation. Invalid numbers (such as English words) are replaced with 0 by Kst.

Interpret ... As ... See the corresponding section for the Plot Dialog.

Always accept files matching: Here you can enter an expression which you want to designate ASCII data files, in case they are being claimed by another data source.

Data starts at line: Data before this line will be ignored. Setting this value to zero ensures that all of the data will be read.

Read field names from line: If this option is selected, the field names for the file's vectors will be read from the specified line. If the columns do not have labels, field names are assigned by Kst based on the order of the columns (with the leftmost column having a field name of `1`).

Data format Choose the option which describes how the data is delimited (how the columns and rows are separated). Each column of this file represents a field, while each row represents one frame. Columns are typically separated by tabs or spaces, and rows are separated by carriage returns.

3.4.2 Dirfiles

Dirfiles are a very simple data source designed to be read efficiently by Kst. The data source is a directory which contains each raw field in its own binary file. Because the data for a given field are stored sequentially on disk in a binary format, reading is very efficient, typically limited by hard disk speed. Fixed and floating point binary formats are supported, as well as calibrations, bit fields and linear combinations of channels.

The directory contains one file for each field. These files contain only the raw samples, written sequentially. As in the rest of Kst, each field in a datasource must have the same number of frames, but each field may have a different (fixed) number of samples per frame. Formatting and calibration information is contained in a file in the Dirfile called `format`, which lists each field and its properties. Below is example of such a file: In this file, a '#' specifies a comment, and for all fields, the first column specifies the field name.

```
##### Raw fields ####
# data types:
# c: 8 bit signed
# u: 16 bit unsigned.  U: 32 bit unsigned
# s: 16 bit signed     S: 32 bit signed
# f: 32 bit float      d: 64 bit float
# The fieldname is also the name of the raw binary file
# holding the data.
# Fieldname  RAW    datatype   samples/frame
sensor1      RAW       s         1
sensor2      RAW       U         20
sensor3      RAW       c         1
#
#### derived fields ####
# LINCOM fields: (F_out = F1_in*m+b [+ F2_in*m+b [+ ...]])
# Fieldname  LINCOM  N  Field1_in       gain        offset
S1V          LINCOM  1  sensor1    1.52587890625E-4 0.00
#
# LINTERP Fields: Calibrate using an ascii lookup table.
# lookup table format:
#     2 whitespace separated columns
#     The first column is raw values, and the second is
#     a corresponding calibrated value.  The file must be
#     sorted by column 1.
# The table is linearly interpolated to the value of
# Field_in.
# Fieldname  LINTERP  Field_in  Calibration_file
S2K          LINTERP  sensor2   /data/etc/calibration/C2K.cal
#
# BIT values: the value of 1 bit of raw data: 0 or 1
# Fieldname  BIT  Field_in  bit_num (0 to 7 for chars)
RELAY1       BIT  sensor3   0
RELAY2       BIT  sensor3   1
#
# The next line includes the contents of another format file
# into this one.  Any fields referred to in this file will be
# looked for in ../work/, not in ./, so this is essentially
# including an entire other datasource.
```

```
INCLUDE ../work/format
```

The following code fragment (which foolishly ignores all pretense at error checking) could be used to create this data source...

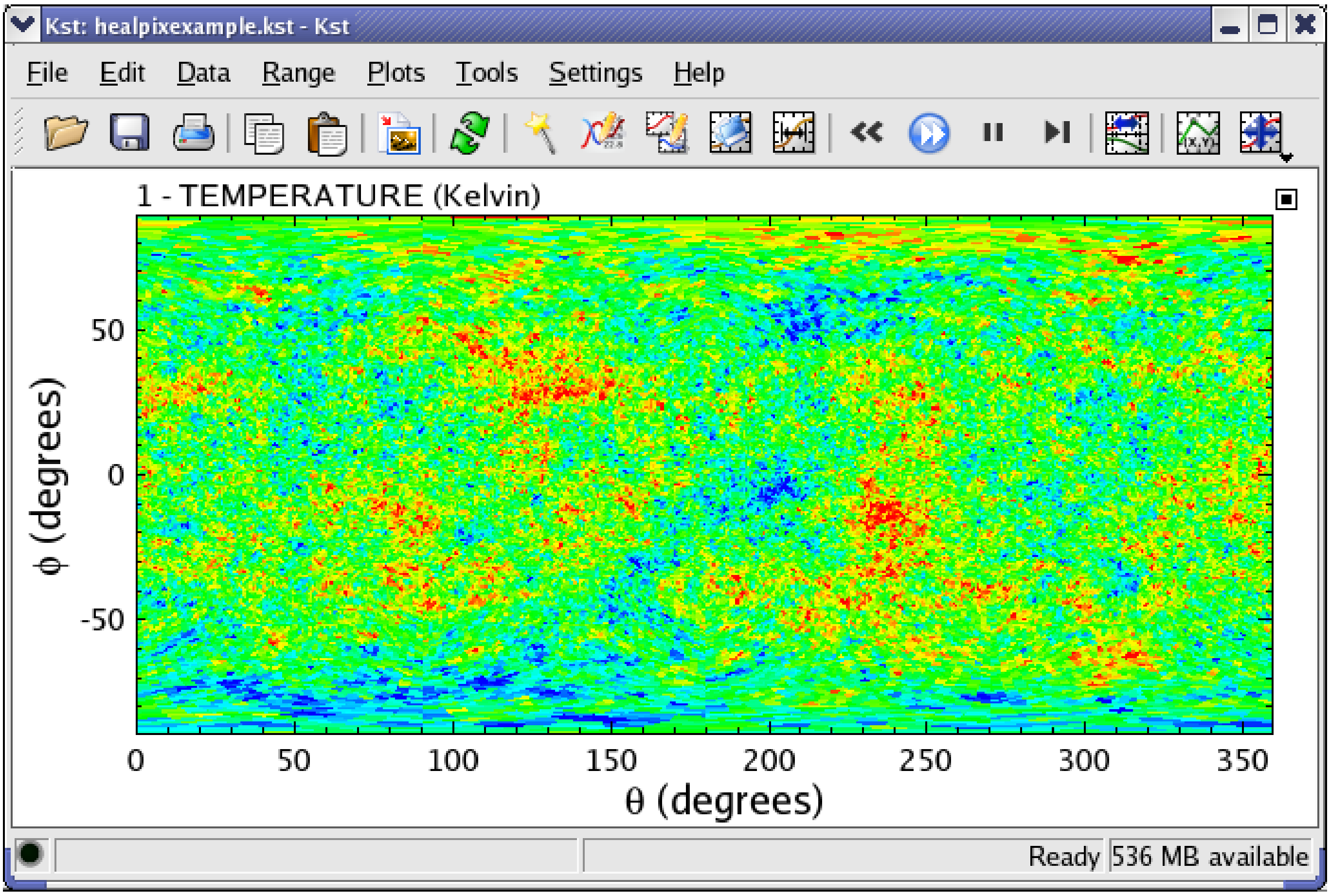
```
  char bits[1000];
  short s1[1000];
  unsigned int s2[20000];
  int fp;

  ......
  /* fill bits, s1, and s2 with data...*/
  ......

  fp = open("/data/example/sensor3",  O_WRONLY|O_CREAT, 00644);
  write(fp, bits, 1000*sizeof(char));
  close(fp);

  fp = open("/data/example/sensor2",  O_WRONLY|O_CREAT, 00644);
  write(fp, bits, 1000*20*sizeof(unsigned int));
  close(fp);

  fp = open("/data/example/sensor1",  O_WRONLY|O_CREAT, 00644);
  write(fp, bits, 1000*sizeof(short));
  close(fp);

  /* create the ascii file /data/example/format, listed above */
  /* create the cal file, /data/etc/calibration/C2K.cal,
     described above. */

  ......
```

Dirfiles can be used for real time data, but as each field is written separately, this requires some extra care. First of all, the writing application should avoid all buffering or caching - raw writes as in the above example work fine. Secondly, the order in which the fields are written need to be considered: Kst determines the number of frames available in the data source from the size of the file containing the first RAW field encountered in the format file. So the first RAW field in `format` needs to be the last one written, as in the above example. If the data source is being read over NFS, attribute caching needs to be turned off.

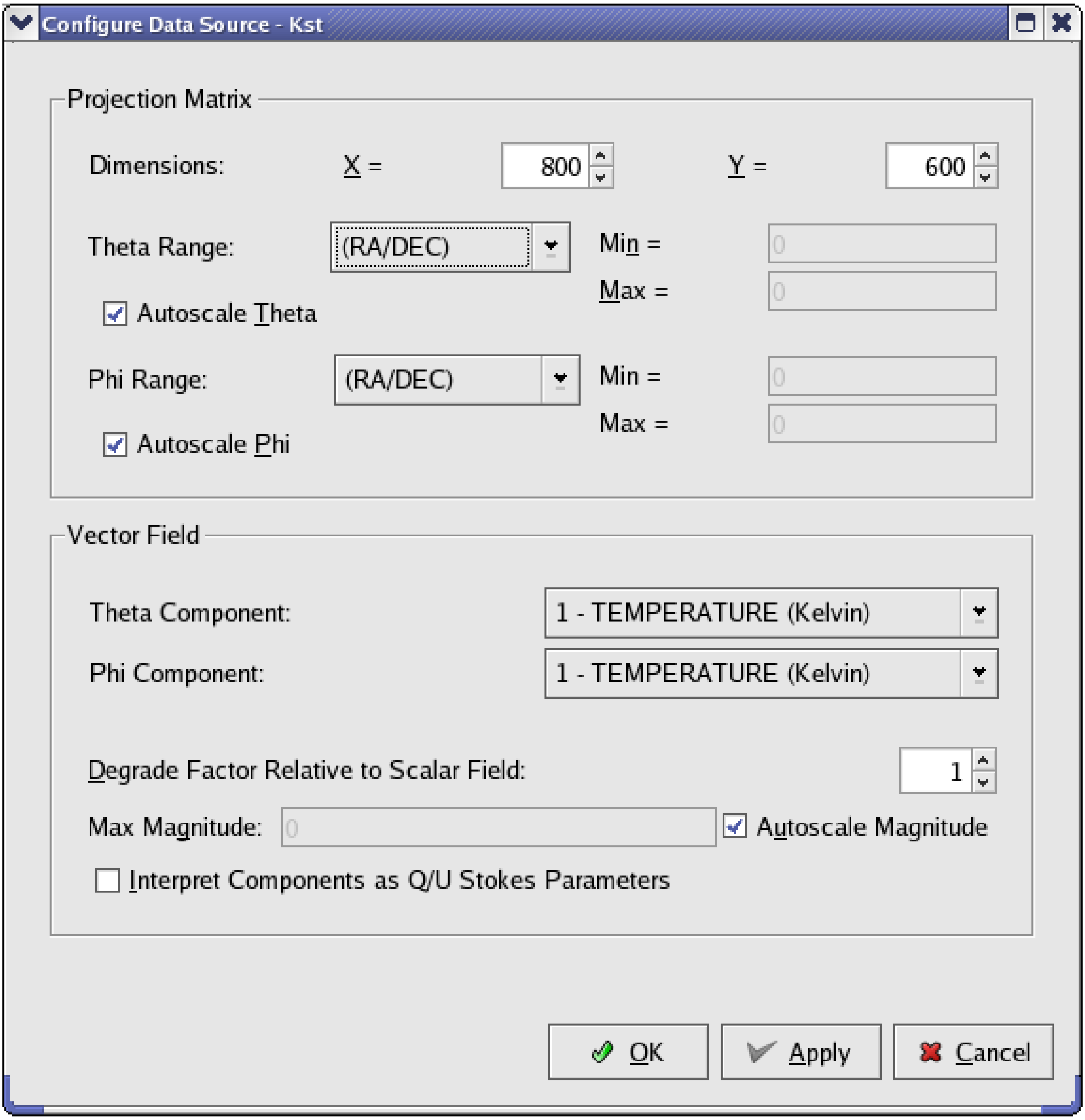
When selecting a dirfile for use in Kst, the directory containing the field files should be selected. Kst will then automatically look for a `format` file, if it exists, to determine the fields and their properties.

3.4.3 HEALPix FITS files

HEALPix is a pixelisation scheme for the sphere. More information can be found at http://healpix.jpl.nasa.gov/. HEALPix data is typically be loaded into Kst as a matrix. For this data source to work you must have the CFITSIO libraries installed. An image of a sample HEALPix matrix is shown below:



Here is the configuration dialog for the HEALPix data source, an explanation of the possible options is given underneath:



Configure Data Source - Kst

Projection Matrix

Dimensions: X = 800 Y = 600

Theta Range: (RA/DEC) Min = 0 Max = 0

☑ Autoscale Theta

Phi Range: (RA/DEC) Min = 0 Max = 0

☑ Autoscale Phi

Vector Field

Theta Component: 1 - TEMPERATURE (Kelvin)

Phi Component: 1 - TEMPERATURE (Kelvin)

Degrade Factor Relative to Scalar Field: 1

Max Magnitude: 0 ☑ Autoscale Magnitude

□ Interpret Components as Q/U Stokes Parameters

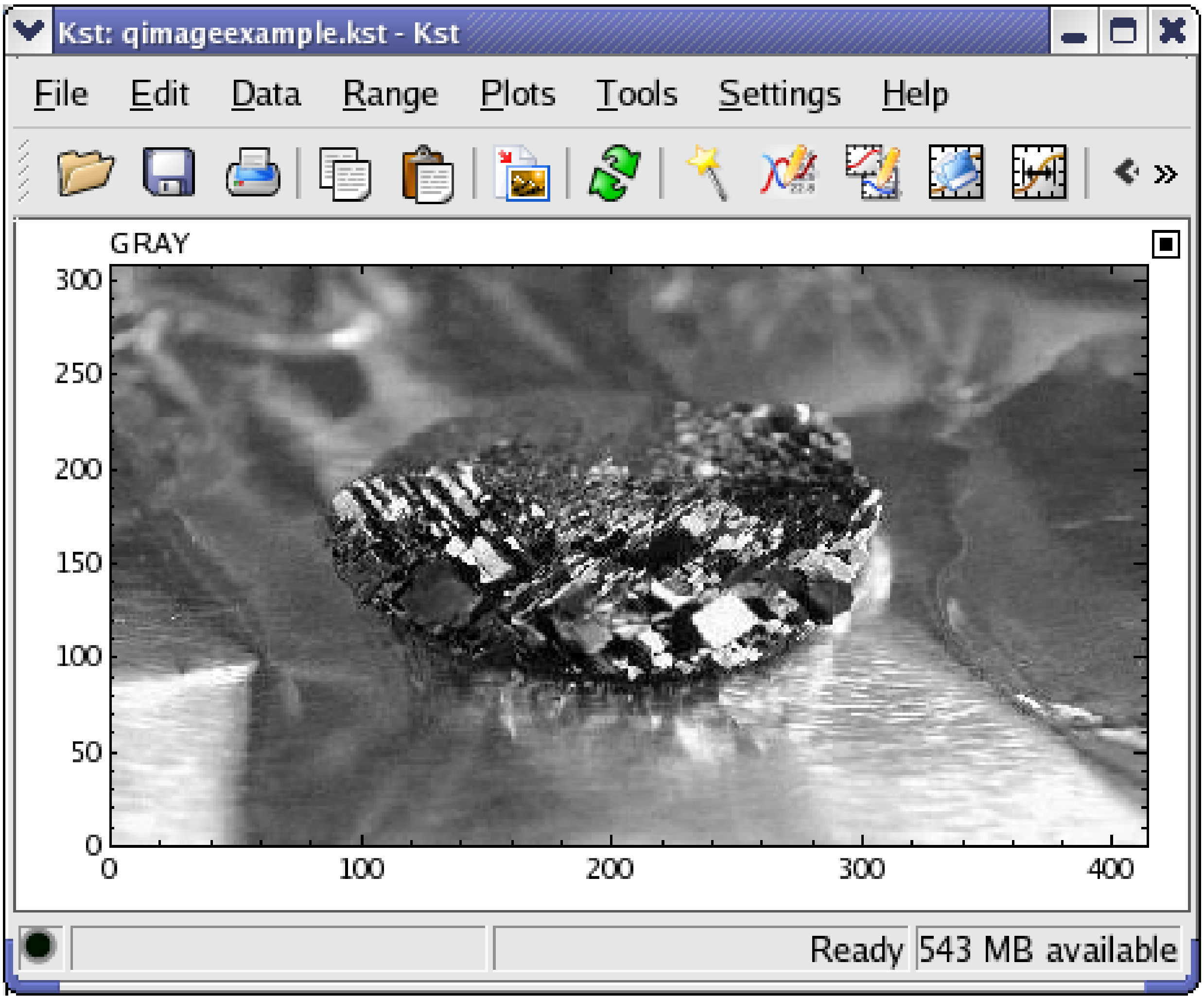
Dimensions The size of the matrix to create based on the HEALPix data. The larger the matrix, the higher the resolution. If you choose dimensions which are larger than the resolution of the data in the HEALPix file, you will be able to see the shape of the HEALPix pixels in the image of the matrix.

Theta / Phi Range The colatitude / azimuth range of data to extract.

Theta / Phi Range The colatitude / azimuth range of data to extract.

3.4.4 QImage

The QImage datasource allows you to read image data into Kst matrices from a variety of formats. The collection of supported formats depends on the libraries installed on your system, but most popular formats are generally readable. If you'd just like to add a picutre to a plot, then the Picture annotation object is a better choice to use. An example of an image plotted from a QImage sourced matrix is shown below:


Kst: qimageexample.kst - Kst
File Edit Data Range Plots Tools Settings Help
GRAY
300
250
200
150
100
50
0
0
100
200
300
400
Ready
543 MB available

3.4.5 Indirect

This is a ‘helper’ data source, which allows you to access data through a file pointer. If you put the URL or path of a data file on the first line of a text file with the extension `.cur`, then the Indirect data source is used to load the referenced file. The use of this is that the contents of the `.cur` file can then be externally modified to change the data file which is loaded by Kst.

3.4.6 CDF

The Common Data Format (CDF) was developed by NASA. It is described as ‘a self-describing data format for the storage and manipulation of scalar and multidimensional data in a platform- and discipline-independent fashion.’. This means that it is a generalized file format for storing and retrieving data, similar to FITS. For this datasource to work you must have the CDF libraries installed.

3.4.7 netCDF

The network Common Data Form (netCDF) was developed by Unidata. They describe it as ‘a set of interfaces for array-oriented data access and a freely-distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The netCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.’. In practice, it is similar to the FITS and CDF formats. For this datasource to work you must have the netCDF libraries installed.

3.4.8 LFIIO

The LFIIO datasource reads FITS format data files created by the LFI component of the Planck satellite. It is currently the default reader for FITS files in Kst. For this datasource to work you must have CFITSIO installed.

3.4.9 WMAP Reader

This datasource allows you to load vectors from WMAP Time Ordered Data (TOD) files. For more information, see http://lambda.gsfc.na
. For this datasource to work you must have CFITSIO installed.

3.4.10 SCUBA File Reader

This datasource allows you to read the datafiles produced for SCUBA, an instrument on the JCMT telescope in Hawaii.

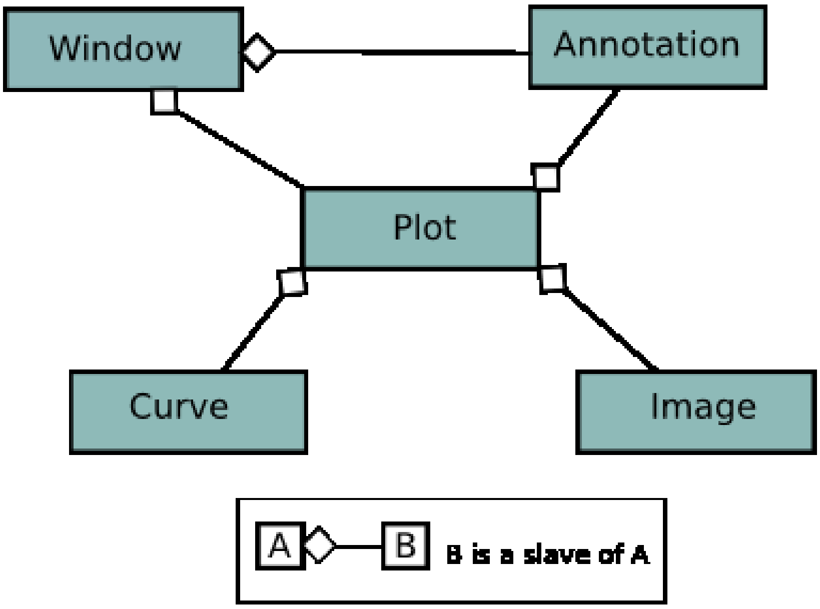
Chapter 4

Working with Views

4.1 View Objects Overview

4.1.1 View Object Concepts

We will call Kst objects which are painted onto the screen ‘View Objects’. These include windows, annotations, plots, curves, and images. A diagram of the relationship between these objects is displayed below:



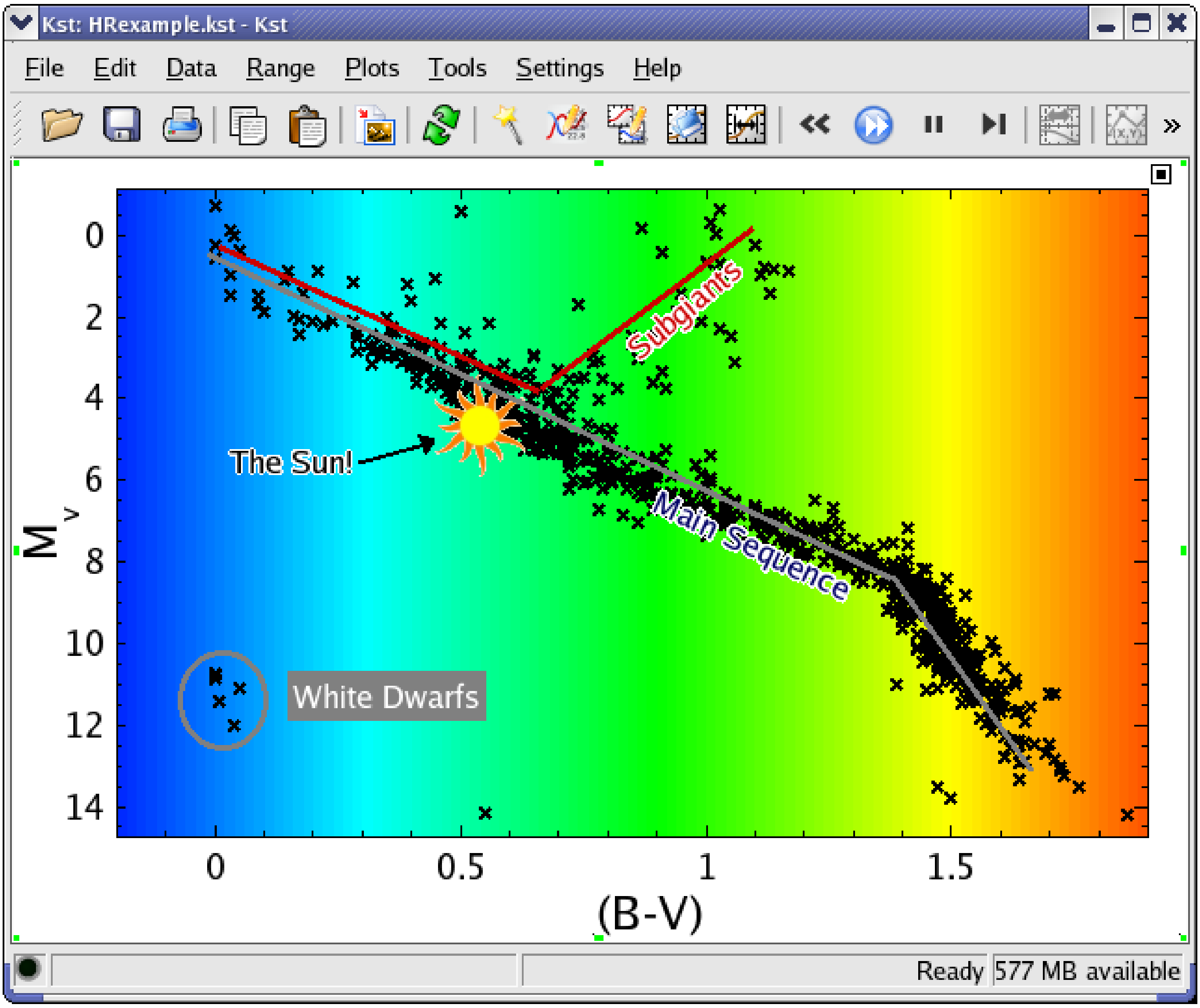
As can be seen from the diagram, the most fundamental views are windows, which act as containers for plots and annotation objects. Plots, however, are the most important view objects of Kst as they perform its chief function of displaying data.

Annotations are objects like labels, arrows, etc. which can be used to display information about plots, or highlight areas of interest. The annotation objects currently implemented in Kst are Labels, Boxes, Ellipses, Lines, Arrows, Pictures and Legends. They are very useful when preparing plots for presentation or publication.

Instances of the various view objects in Kst are hierarchically grouped - annotations, for example, are often deemed ‘children’ of a specific plot based on their location. Moving and resizing a plot then resizes the annotation also. A good way to work directly with the view object hierarchy is to use the View Manager. It can be accessed by selecting Data → View Manager.

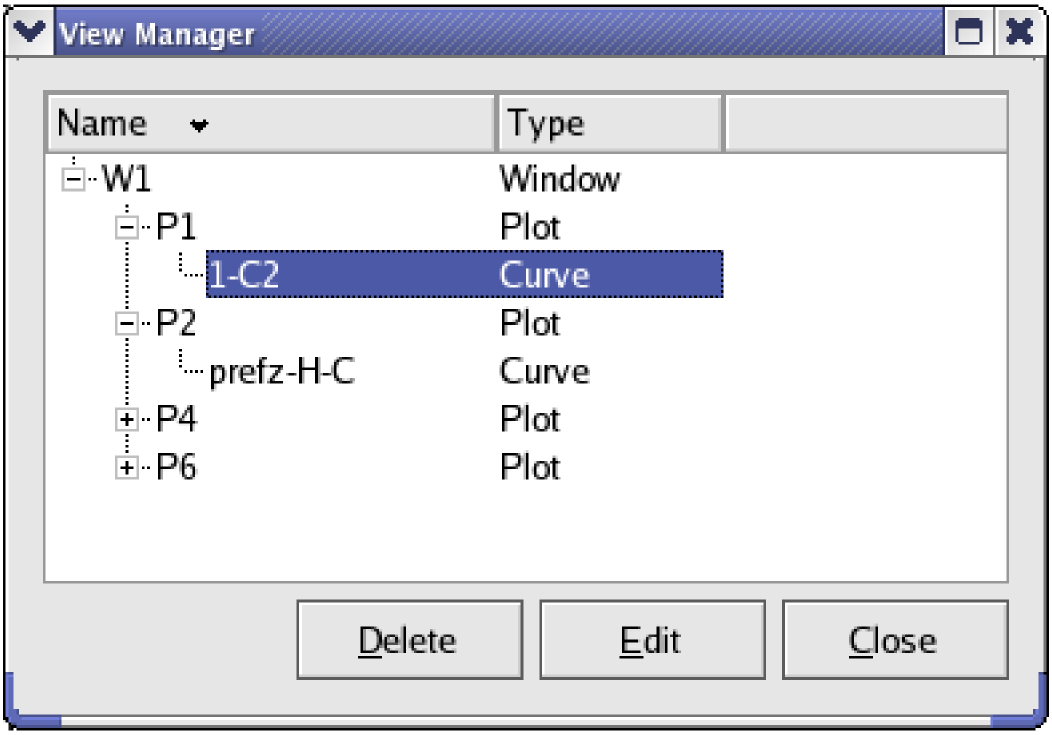
All view objects can be created by selecting their corresponding draw tools. You can edit their options by double clicking on them in Layout mode.

Here is a sample use of view objects in Kst, where we have used a Plot, Labels, Arrows, Lines, and a transparent PNG Picture to mark up an astronomical 'color magnitude diagram'.



4.1.2 The View Manager

The View Manager is a tree list view which allows you to copy and delete view objects. It also allows you to access their edit dialogs.



Name	Type	
W1	Window	
P1	Plot	
1-C2	Curve	
P2	Plot	
prefz-H-C	Curve	
P4	Plot	
P6	Plot	

- Edit/Delete View Object: Select a view object (window, plot, curve, etc.) to edit and click the Edit button, or right click on the view object's entry and select the Edit... option in the context menu which appears.
- Copy view objects: Select the view object which you would like to copy and click-drag it over the name of the parent which you would like to copy it into.

Tip

The view manager can be accessed by clicking on the toolbar.

4.1.3 Context Menus

When working with plots and other view objects, right-clicking will bring up a context menu which provides commonly used functions. In general, the context menu which is presented depends on which mouse mode you are in and which object you are centered on, but there are four basic options common to all view objects. The title of the context menu which appears gives the name of the view object which you have clicked on.

Edit... Brings up the edit dialog for the underlying view object. See the View Object Types sections for more information on these settings.

Delete Deletes the view object.

Rename Presents a dialog to modify the unique name assigned to the view object.

Cleanup Layout Arranges the plots in the current window in a tile type pattern, with no overlap between plots. Selecting Default Tile from this sub-menu rearranges the plots according to an aesthetically pleasing algorithm. Selecting the Custom option allows you to manually specify how many columns to tile the plots into. This action applies to all plots in the current window.

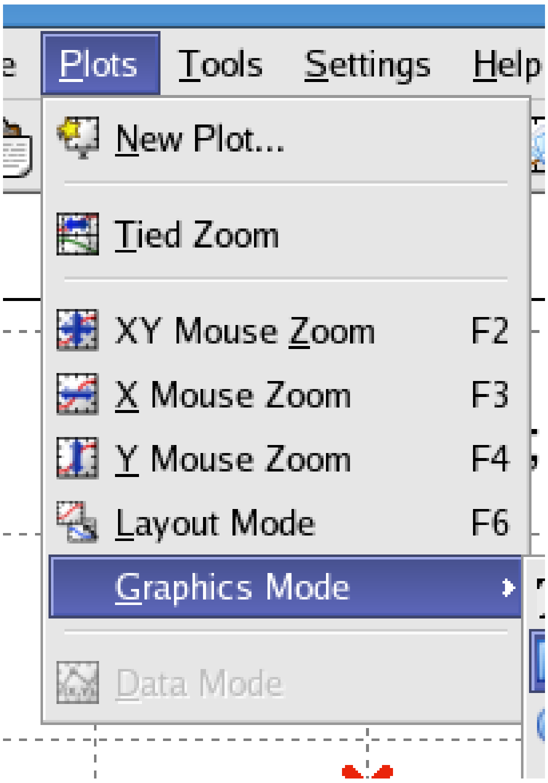
There are additional context menu options specific to the Zoom mouse modes and the Data Layout mode.

4.1.4 Switching among Different Modes

Kst has various mode choices. Each mode has it own specific usages. The following are available modes in Kst:

- Mouse Zoom Mode
- Graphics Mode
- Layout Mode
- Data Mode

You can enable any of the modes from the Plot menu.

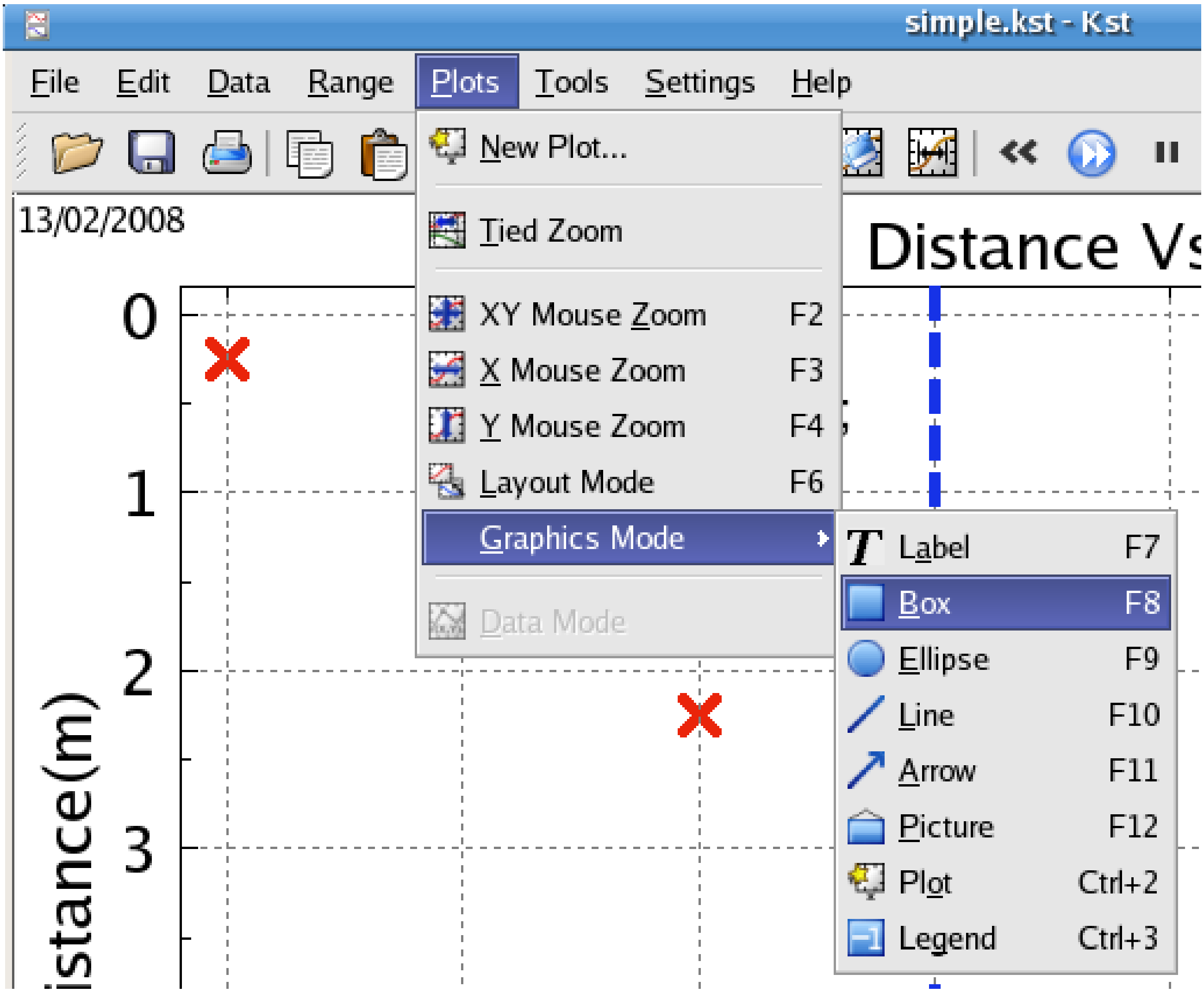


Note that when you open a new session of Kst, it will be in XY Mouse Zoom mode by default. The following summarizes major usages for each mode.

Mouse Zoom Modes The primary usage of mouse zoom modes is to access various zooming functions provided in the context menu. See the section: Zoom Mode Context Menu for detail. There are three mouse zoom modes available in Kst.

- XY Mouse Zoom
- X Mouse Zoom
- Y Mouse Zoom

Graphics Mode Graphics Mode is used to insert various view objects including all of the annotation objects, plots and legends. To enable graphics mode, choose the object you want to insert from the Graphics Mode menu in the Plots menu. See the image below.



Once a view object has been selected from the Graphics Mode menu, you can insert it anywhere on the current window. Also, you can insert this type of object as many times as you want until you choose another view object or switch to a different mode. The context menu for graphics mode is the same as for data mode.

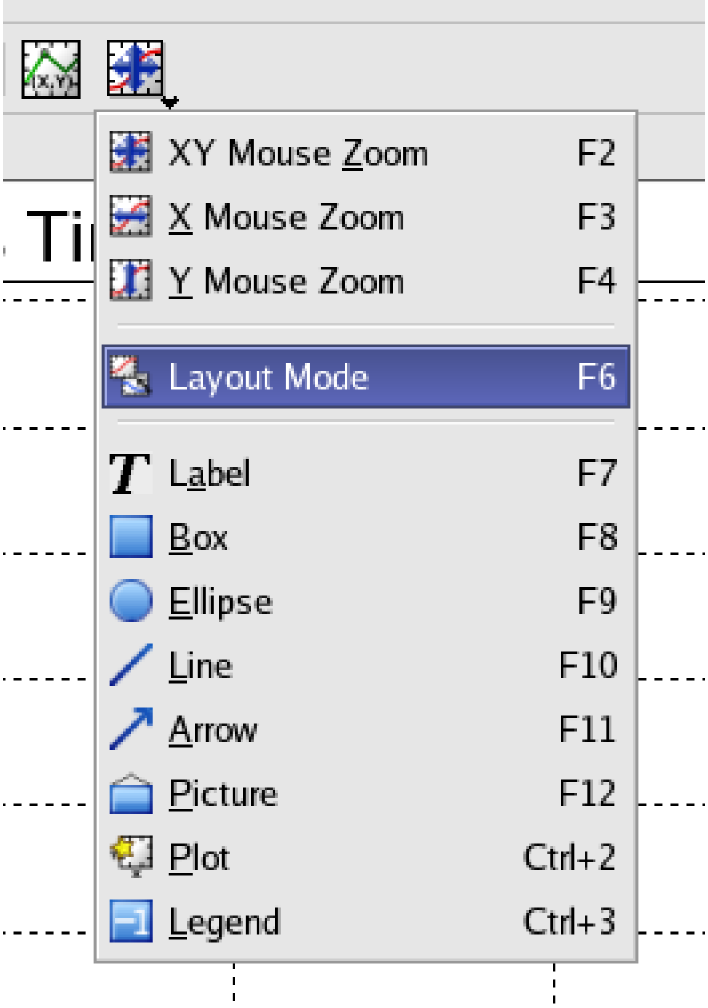
Layout Mode Layout Mode is chosen whenever you want to change the size of or move an object. See Moving and Resizing View Objects.

You can also change the layer of an object or group multiple objects together through the context menu under layout mode. See the Layout Mode Context Menu for detail.

Data Mode Data Mode has already been introduced in Chapter 3. See details in the corresponding section. Note that data mode can only be enabled when Kst is in one of the mouse zoom modes.

Tip

The rightmost icon in the toolbar indicates which mode the current Kst session is in. To quickly switch to another mode, click on that icon; the menu below will appear, and you can choose the mode you want.



In the above example, Kst is in XY Mouse Zoom mode, and it is going to change into Layout Mode.

4.2 View Object Types

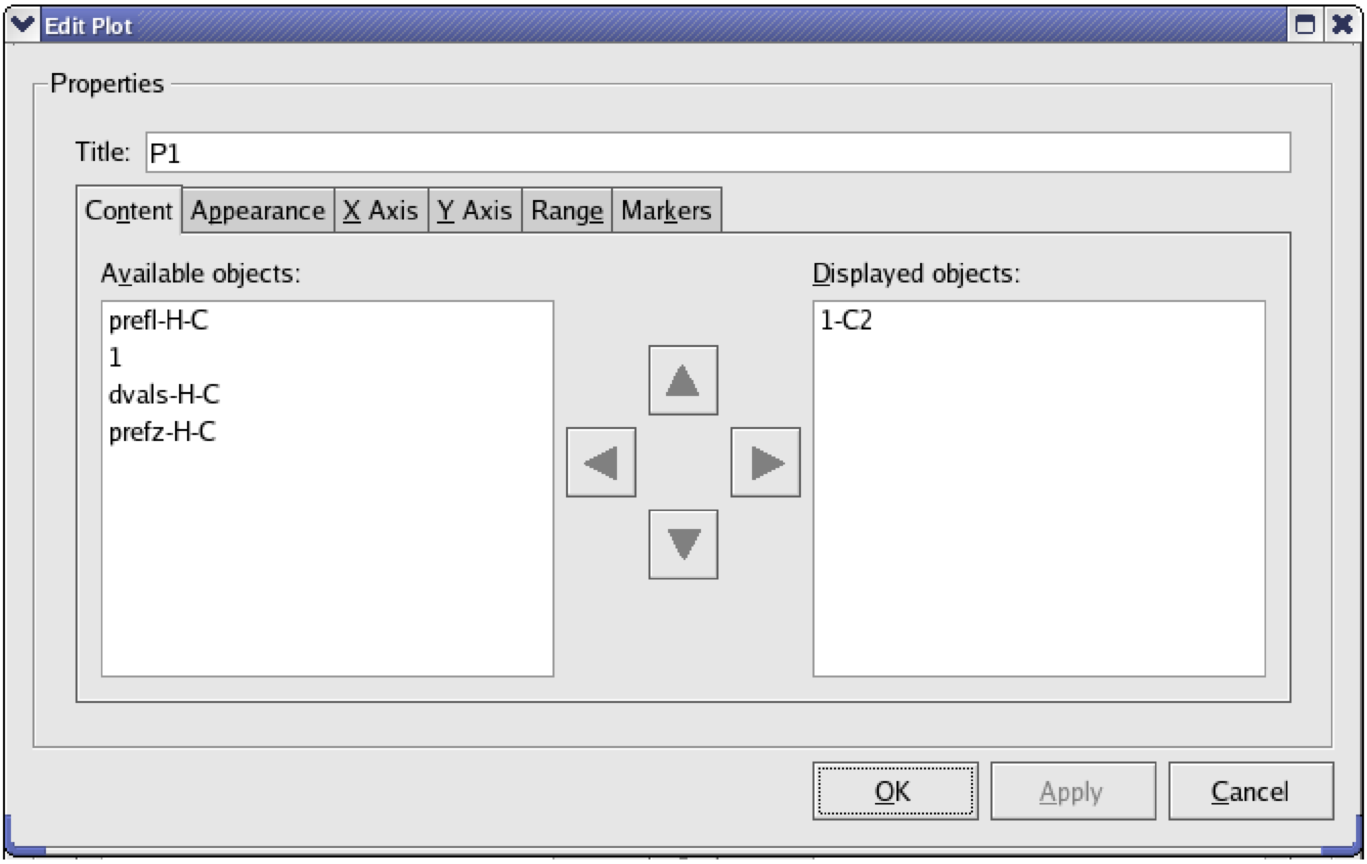
4.2.1 Plots

Plots allow you to display curves and images in Kst. The Edit Plot dialog is used to change a plot's content, appearance, axes, and markers. To access it, right click on a plot and select the Edit... menu item from the context menu, or use the View Manager. There are several tabs:

- Content
- Appearance
- X/Y Axis
- Range
- Markers

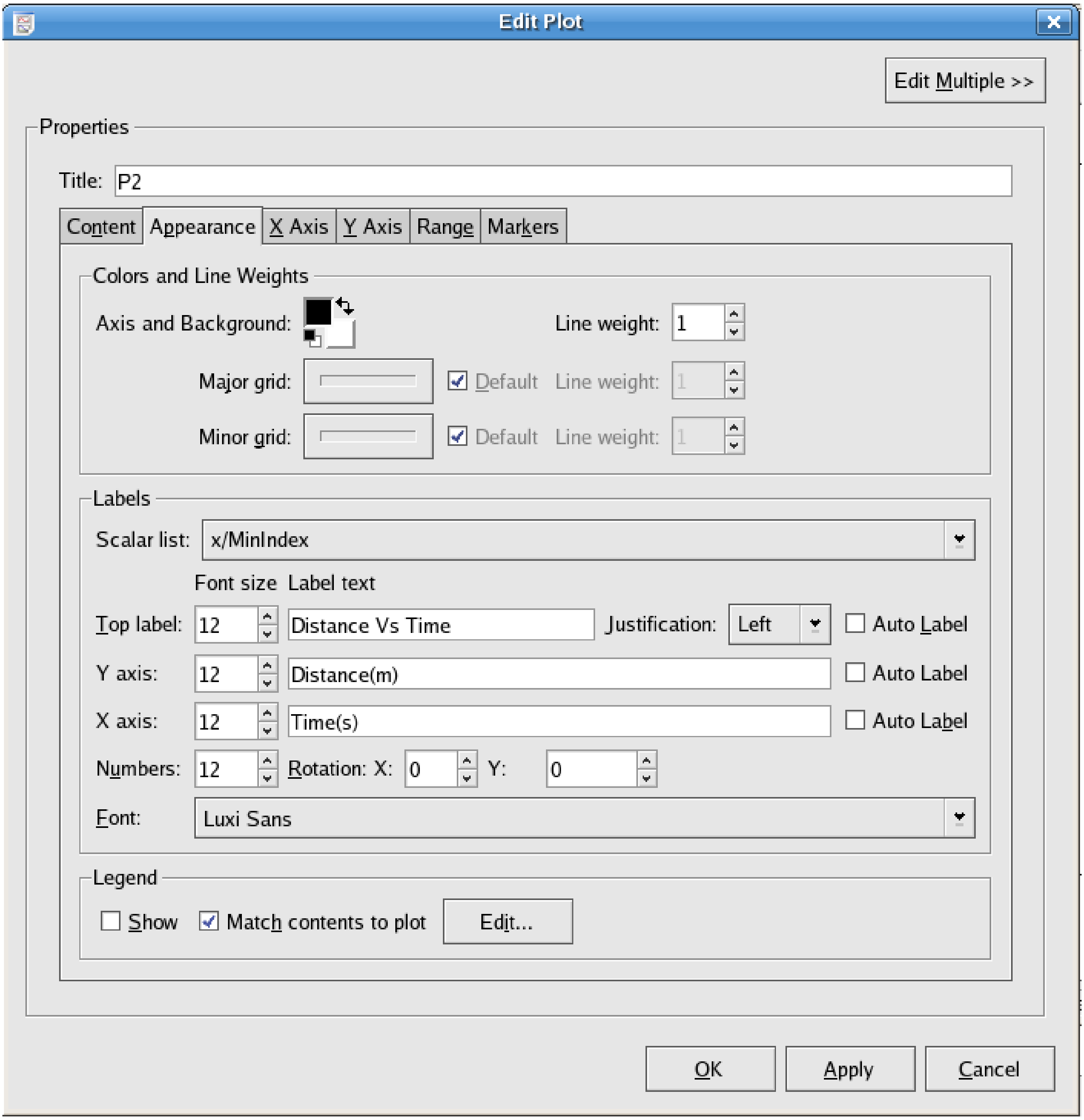
4.2.1.1 Content Tab

This tab allows you to select which of the available plot objects (curves, images, etc.) will be displayed in the plot. Move objects between the Available and Displayed object lists using the left and right arrow buttons. Change the order in which the objects are plotted with the up and down arrow buttons.



4.2.1.2 Appearance Tab

This tab allows you to adjust the colors, fonts, legend, and labels of a plot.



Edit Plot

Edit Multiple >>

Properties

Title: P2

Content | Appearance | X Axis | Y Axis | Range | Markers

Colors and Line Weights

Axis and Background: ■ Line weight: 1

Major grid: ☑ Default Line weight: 1

Minor grid: ☑ Default Line weight: 1

Labels

Scalar list: x/MinIndex

Font size Label text

Top label: 12 Distance Vs Time Justification: Left □ Auto Label

Y axis: 12 Distance(m) □ Auto Label

X axis: 12 Time(s) □ Auto Label

Numbers: 12 Rotation: X: 0 Y: 0

Font: Luxi Sans

Legend

□ Show ☑ Match contents to plot Edit...

OK Apply Cancel

4.2.1.2.1 Colors and Line Weights

Axis and Background The colors for the axes (foreground) and background of the plot.

Line Weight The line width in pixels.

Major/Minor Grid The line width and colour of the lines marking out the major/minor grid lines. To enable these grid lines for the axes, use the X Axis/Y Axis tabs.

4.2.1.2.2 Labels

Scalar list A list of the current scalars defined in Kst. This list is used to quickly insert values in the label texts. Choosing an item from this list will insert the item at the current text cursor position.

Top label The label located at the top of the plot. Select a font size and enter the text for the label using the controls in this row. These labels can use all of the formatting options of ‘true’ label objects.

Justification This drop-box controls whether the top label will be located at the Left, Right, or Center of the plot’s title space. The justification for the X and Y axis labels of a plot is always the center.

X/Y axis The label located vertically next to the X/Y axis of the plot. Select a font size and enter the text for the label using the controls in this row. These labels can use all of the formatting options of ‘true’ label objects.

Numbers, Rotation X and Rotation Y The numbers used to label both the X axis and the Y axis of the plot. The size and angles of the numbers can be specified. Positive angles rotate the text clockwise, while negative angles rotate the text counter-clockwise.

Auto Label Click this button to automatically generate label texts for all labels on the plot. The text for Top label will be a list of paths to the data files used in the plot. The text for Y axis will be a list of the dependent variable descriptions (e.g. ‘Number in Bin’ for a histogram). The text for X axis will be the name of the vector used as the independent variable.

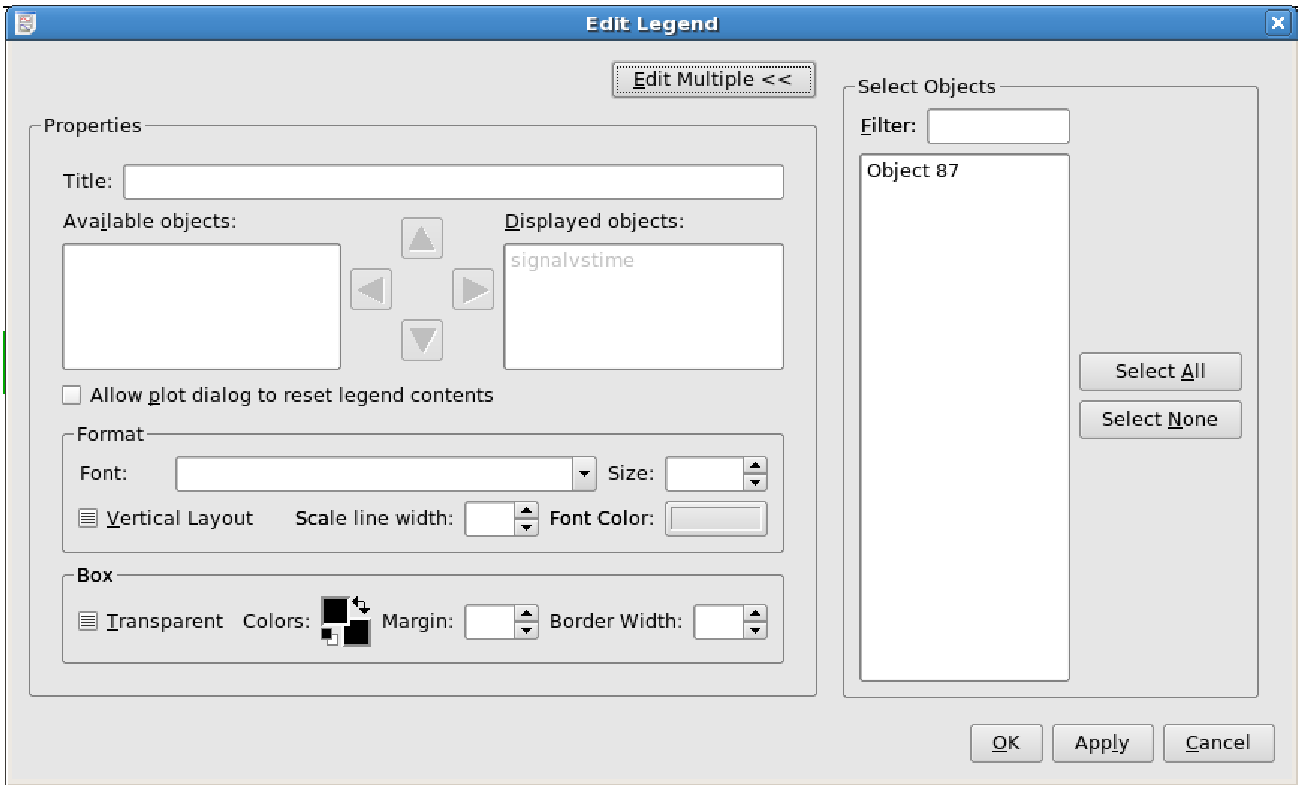
Font The font used for all labels of the plot. Select an available font using the drop-down list.

4.2.1.2.3 Legend

Show Check this box if you would like the plot to display a legend.

Match Contents to Plot Check this box if you would like the plot to automatically adjust its legend to reflect its contents.

Edit... This button displays the Edit Legend dialog, shown below.



The available options are:

Title The text for the top of the legend.

Available / Displayed Objects This allows you to choose a set of objects to display in the legend which differs from the ones displayed on the plot.

Allow plot dialog to reset legend contents If Match Contents to Plot was not selected in the Edit Plot Dialog, and this option is selected, then the Plot can enforce that its legend matches its true contents.

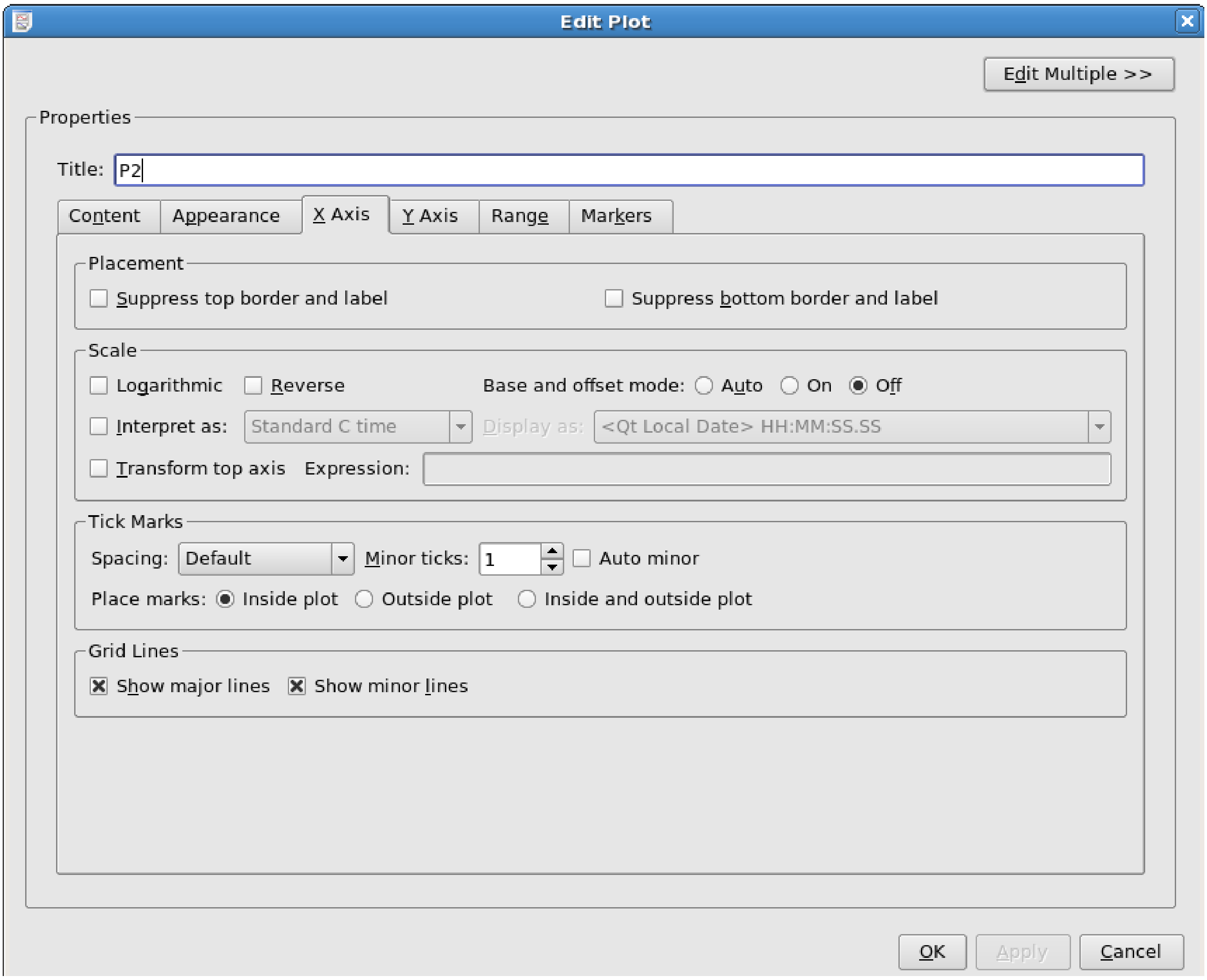
Font / Size / Color The formatting options for the legend text.

Box (Transparent / Colors / Margin / Border Width) The properties of the legend outline and fill.

4.2.1.2.4 Edit Multiple

Click on the Edit Multiple button, the dialog allows you to apply modifications of the selected plot to other plots as well. Besides manually choosing the objects you want to edit in the dialog, you can type objects' name in the Filter textbox. The use of wildcards is supported.

4.2.1.3 X/Y Axis Tabs



Title: P2

Placement

□ Suppress top border and label □ Suppress bottom border and label

Scale

□ Logarithmic □ Reverse Base and offset mode: ○ Auto ○ On ● Off

□ Interpret as: Standard C time Display as: <Qt Local Date> HH:MM:SS.SS

□ Transform top axis Expression:

Tick Marks

Spacing: Default Minor ticks: 1 □ Auto minor

Place marks: ● Inside plot ○ Outside plot ○ Inside and outside plot

Grid Lines

☒ Show major lines ☒ Show minor lines

4.2.1.3.1 Placement

Suppress top/bottom border and label Check this box to remove the white space at the top/bottom of the plot.

4.2.1.3.2 Scale

Logarithmic Enable this checkbox if you wish to use a logarithmic scale for the axis.

Base and offset mode Enabling this option causes the axis to be labelled by displacements from an initial position, where the initial position is displayed in the lower left of the plot. The displacement labels are printed inside square brackets to avoid confusion.

Reverse Causes the axis direction to be reversed.

Interpret as: / Display as: If your axis contains date/time information encoded in a difficult to read form (e.g. Julian Date), this option allows you to have Kst automatically convert it to something more legible. Note: The time data has to be in one of the formats listed in Table 4.1.

Date Format	Description	Online Reference
Julian Year	a time interval of exactly 365.25 days in astronomy	See definition in wikipedia
Standard C time	The amount of time that has elapsed since midnight at the beginning of January 1st 1970	
JD	Julian Date	See definition for Julian Date
MJD	Modified Julian Day: the number of days that have elapsed since midnight at the beginning of Wednesday November 17th 1858	See more details inwikipedia
RJD	Reduced Julian Day	See the details in wikipedia
TAI	Temps Atomique International (International Atomic Time)	For more details, see this webpage.
KDE Short Date and Time	It has the following format: YYYY-MM-DD HH:MM:SS[seconds]. e.g. 1958-01-29 10:05:58[seconds]	
KDE Long Date and Time	It has the following format: day date Month Year HH:MM:SS[seconds]. e.g. Wednesday 29 January 1958 10:05:58[seconds]	

Table 4.1: Date Format in Kst

Transform top axis Checking this box causes the top axis of the plot to be labelled following the contents of the expression field. The variable x can be used in the expression field to refer to the value on the lower X axis. To plot the top axis as the lower axis scaled by a factor of 2, for example, enter `2*x` in the expression field.

4.2.1.3.3 Tick Marks

Spacing Select Course, Default, Fine, or Very Fine from this drop down menu to determine which algorithm Kst will use to determine the number of Major and Minor tick marks which will be plotted.

Minor Ticks / Auto Minor Unchecking Auto Minor allows you to tell Kst how many minor ticks to place between each pair of major ticks. The number of major ticks cannot be set manually.

Place marks: Select Inside plot, Outside plot, or Inside and outside plot to determine the location of the tick marks with respect to the axes.

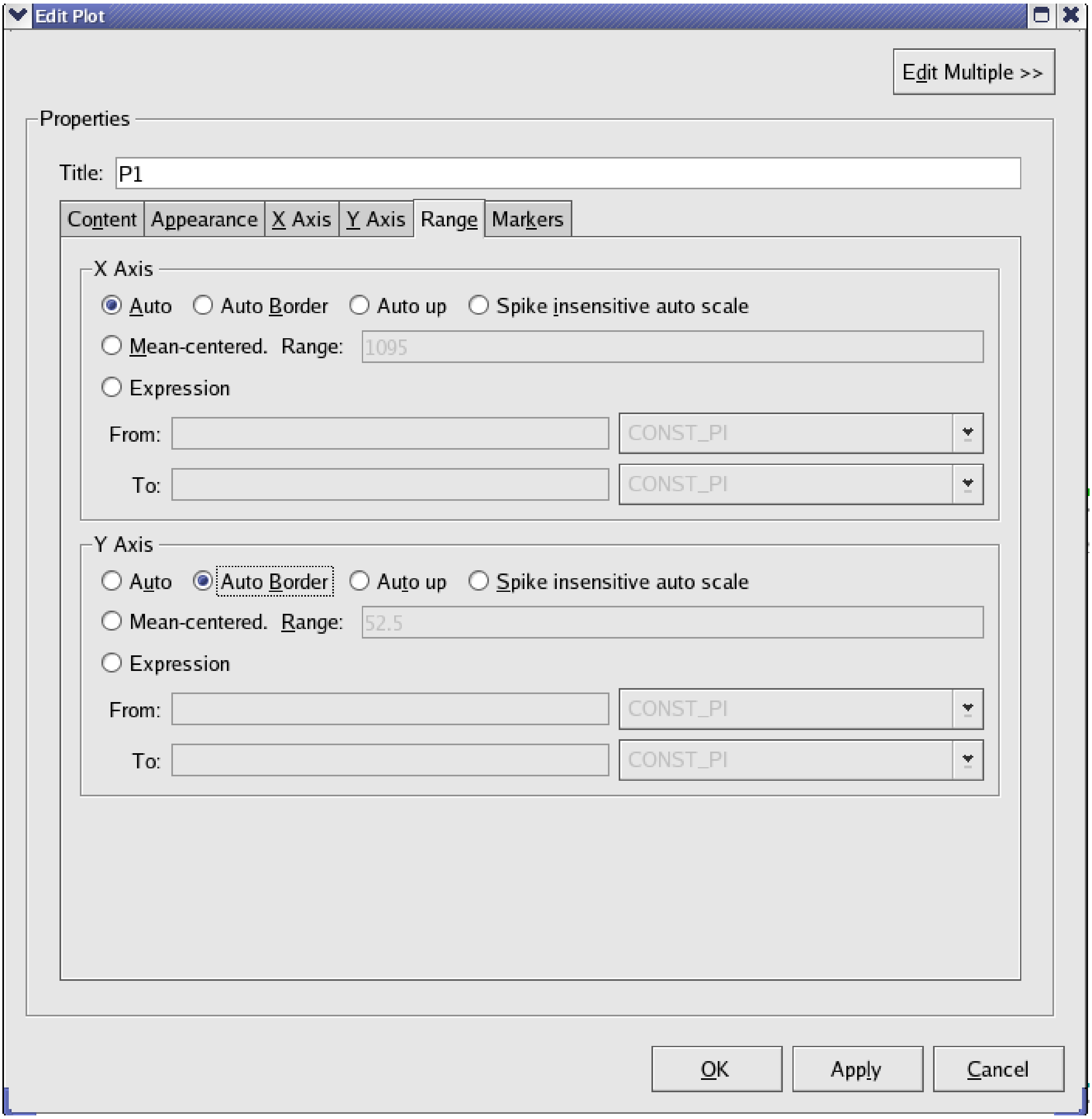
4.2.1.3.4 Grid Lines

Show major/minor lines Select these boxes if you would like lines to be drawn through the plot connecting the major/minor tick marks on opposing axes.

4.2.1.3.5 Edit Multiple

4.2.1.4 Range Tab

The settings for the plot axes are specified on this tab. The axis settings are split into two identical sections — an X axis section and a Y axis section.



Title: P1

X Axis

● Auto ○ Auto Border ○ Auto up ○ Spike insensitive auto scale

○ Mean-centered. Range: 1095

○ Expression

From: CONST_PI

To: CONST_PI

Y Axis

○ Auto ● Auto Border ○ Auto up ○ Spike insensitive auto scale

○ Mean-centered. Range: 52.5

○ Expression

From: CONST_PI

To: CONST_PI

4.2.1.4.1 X/Y Axis

Auto Select this option to let Kst automatically choose a scale for this axis based on the highest and lowest values for this axis found in the plotted objects.

Auto border This option behaves similarly to Auto, but the axis is padded with a small amount of extra space. This can sometimes improve the appearance of plots, and make it easier to see features near the range limits.

Auto up This option behaves similarly to Auto, but the upper axis limit will only increase, and the lower axis limit will only decrease.

Spike insensitive auto scale Select this option to let Kst automatically choose a scale for this axis that is based on the values found in the plotted objects. In general, ‘spikes’, or sudden short increases or decreases in the value, will be ignored when determining the scale. This option is usually for the dependent (Y) axis.

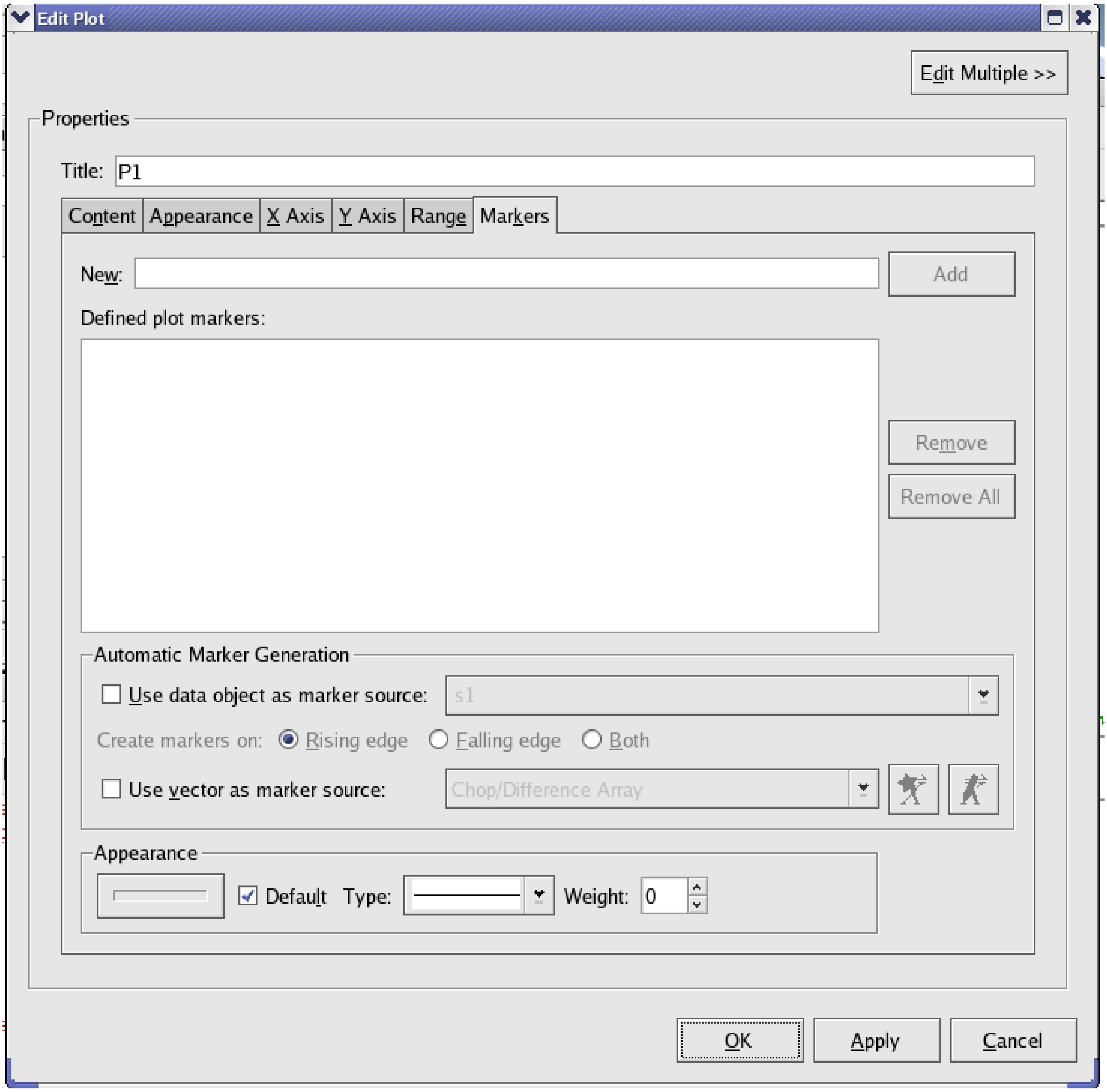
Mean-centered Select this option to center the axis around the mean of the values found in the plot (X values are used to calculate the mean for the X axis, and Y values are used to calculate the mean for the Y axis). Specify the length of the axis in the Range text box.

Expression Manually specify lower and upper limits for the axis. Enter the expression for the limits in the From: and To: textboxes. These can be numerical values, or equations based on the scalars maintained by Kst. These scalars can be quickly added to the expressions by selecting them from the pull-down menus to the right of the textboxes.

4.2.1.4.2 Edit multiple

4.2.1.5 Markers

Below is a screenshot of the Markers tab.



Title: P1

New:

☐ Use data object as marker source: s1

Create markers on: ● Rising edge ○ Falling edge ○ Both

☐ Use vector as marker source: Chop/Difference Array

☑ Default Type: Weight: 0

Plot markers provide an easy means of flagging areas of interest on a plot. The Markers tab provides full access to plot marker settings and options. Navigation functions for plot markers are described in the Scroll Menu section.

Tip
To quickly create plot markers on a plot, place the mouse cursor over the plot and press the **Insert** key. A vertical line should appear indicating the position of the plot marker.

New and Add Use the New text field to manually enter a plot marker value. Values can be entered in floating point or scientific notation. Values in scientific notation must be entered in the form `mEx` or `mex`, where `m` is the mantissa and `x` is the exponent. For example, one way to enter the value 15000 in scientific notation is by typing `1.5e4`.

Click the Add button to create a new plot marker with the specified value. You will be warned if duplicate plot markers already exist, or if the entered value is not valid. Note that plot marker additions or removals made from the Markers tab are not applied unless the plot dialog settings are applied.

Defined plot markers, Remove, and Remove All The list of existing plot markers for this plot are listed under Defined plot markers. To remove plot markers, highlight the desired markers in the list and click the Remove button. Multiple markers can be selected for removal by clicking and dragging within the list, or by holding down the **Shift** or **Ctrl** key and clicking on items in the list. To remove all plot markers for this plot, click the Remove All button. Note that plot marker additions or removals made from the Markers tab are not applied unless the plot dialog settings are applied.

Automatic Marker Generation In addition to individually defined plot markers, it is also possible to generate plot markers based on a curve. To use a curve object for plot marker generation, check the Use data object as marker source checkbox and select a curve from the drop down list. Then select the criteria for creating plot markers. Selecting Rising edge causes plot markers to be created when the Y value of the selected curve changes from zero to non-zero. Selecting Falling edge causes plot markers to be created when the Y value of the selected data object changes from non-zero to zero. Selecting Both causes plot markers to be created on both rising edges and falling edges of the data object. In all cases, the X value of the point in the data object with the non-zero X value is used to create the new plot marker.

If the selected curve is subsequently updated, additional plot markers will be created if necessary. However, plot markers will never be removed as a consequence of data object changes, so manually created plot markers will never be affected.

It is also possible to create plot markers from a vector. Enable the Use vector as marker source checkbox and select a data vector from the drop down menu. The values of this vector will now be used as markers

Appearance These options allow you to select the colour, line type, and weight (thickness) of the plot markers.

4.2.1.6 Using the Plot Class in KstScript

Plots in Kst can also be manipulated by KstScript. You can change the view of a plot by modifying its properties defined in the Plot Class, such as legend, topLabel, x/yAxis, etc.

See the Plot Class of the Script Chapter for detail. In order to change the view of a plot, you often need to use the Plot Class with other classes, such as the Axis Class, or the Curve Class.

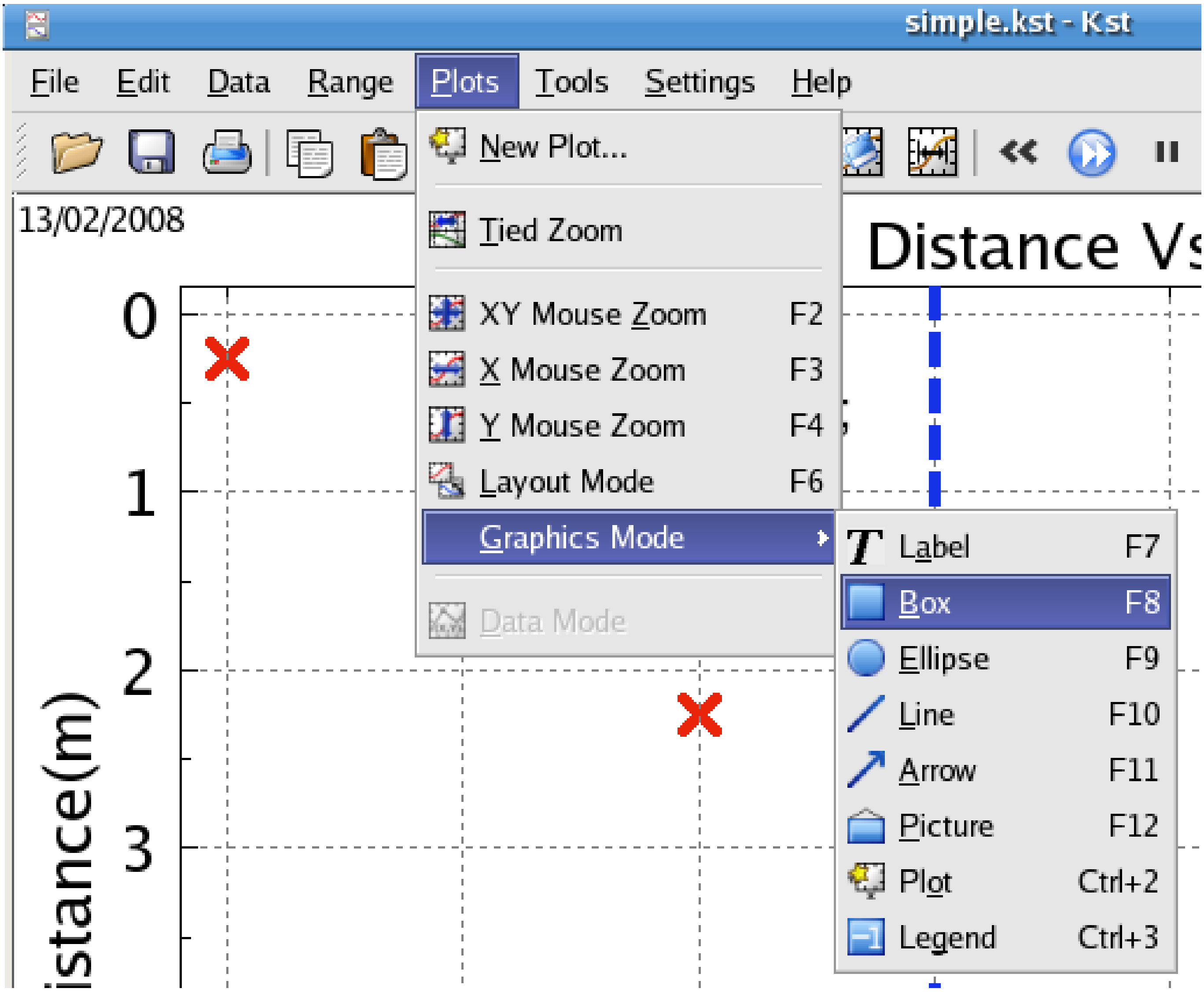
The example below is to show how to use the Plot Class and the TimeInterpretation Class to interpret various date formats for a plot.

Example 4.1 Example of Using the Plot Class to interpret date format for x Axis

```
w = new Window();     //construct a new window
p = new Plot(w);              // place a plot on the window
c = new Curve("time","Y");  // use vector time and Y to construct a new curve
c.tagName = "time vs Y";
p.curves.append(c);         //to show the curve on the plot
p.xAxis.interpretation.input = 2;   // the input time vector is in Julian Date.
           //2 stands for Julian Date in the TimeInterpretation Class.
p.xAxis.interpretation.output = 0;  //  the time vector needs to be shown in
            //the format: DD/MM/YY HH:MM:SS.

p.xAxis.interpretation.active = true;   //Active the interpretation
```

4.2.2 Annotation Objects

Annotation objects are Labels, Boxes, Ellipses, Lines, Arrows and Pictures. They can be easily created by choosing the Graphic mode in the Plots menu.

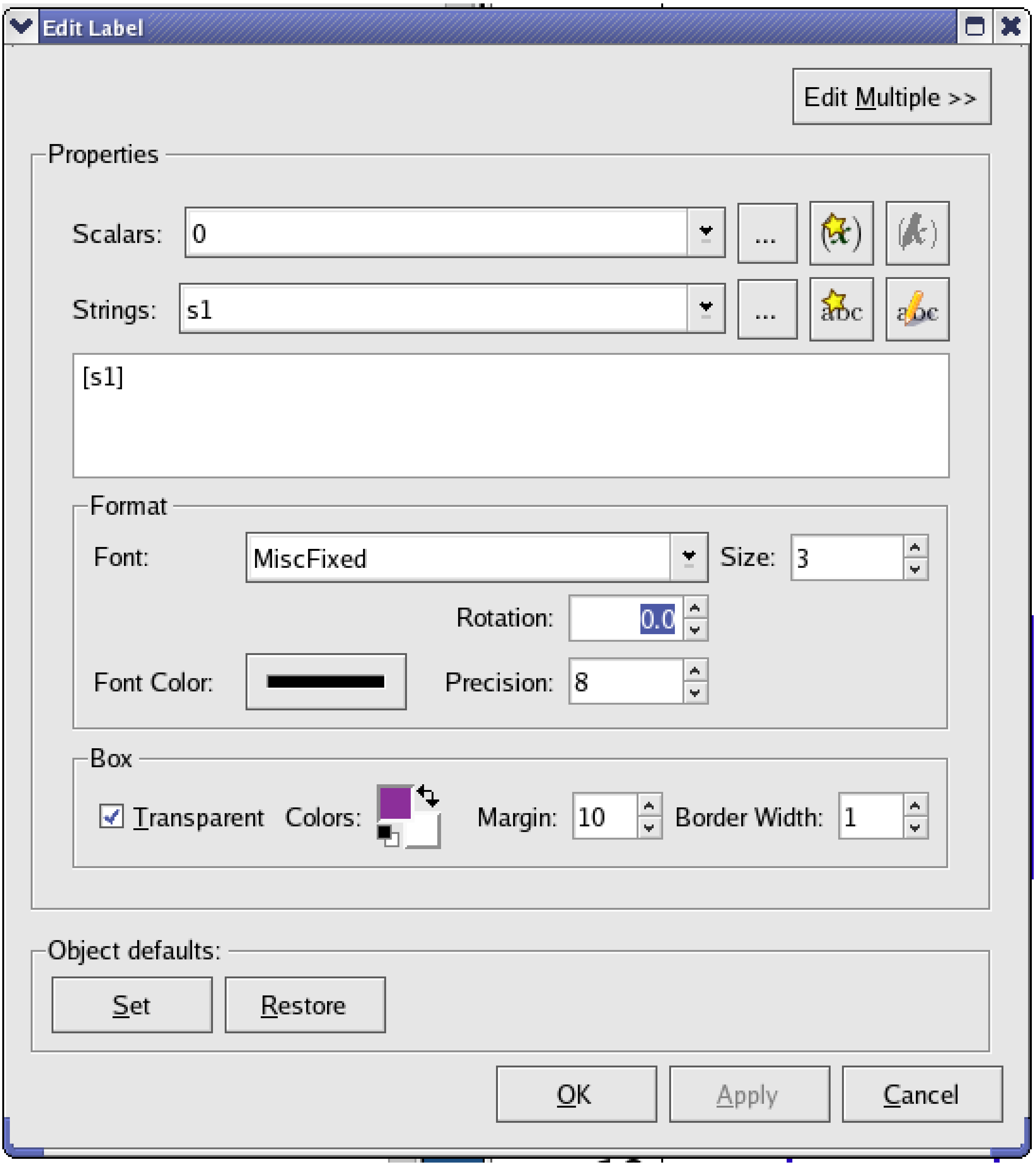


To edit an annotation object, right click on the object and choose the Edit... menu item in the context menu.

To move or resize an annotation object, you must be in Layout mode.

4.2.3 Labels

Custom labels can be displayed in arbitrary locations in Kst plot windows.



Scalars: 0

Strings: s1

[s1]

Font: MiscFixed

Size: 3

Rotation: 0.0

Precision: 8

☑ Transparent

Margin: 10

Border Width: 1

The following are explanations of the dialog box elements.

Scalars A list of scalars currently defined in Kst. Selecting an item from the dropdown list inserts a reference to the scalar's value at the current cursor position in the label text.

Strings A list of strings currently defined in Kst. Selecting an item from the dropdown list inserts a reference to the string's contents at the current cursor position in the label text.

Label text The text displayed by the label. You may enter text manually in this dialog, or insert references into it using the scalar and string drop down menus. Kst labels support many LaTeX formatting and symbol commands, prefixed with the usual '\'. A list of supported commands is given in the table below:

Font The font to use for the label text. Select a font from the dropdown list.

Size The size of the label text. 0 (the default value) roughly corresponds to a 12pt size.

Rotation The number of degrees to rotate the label text. Positive values rotate the label clockwise, while negative values rotate the text counter-clockwise.

Font Color Select the color to use for the label text. Clicking the button displays the standard KDE color chooser.

Precision When scalar variables are included in labels, Precision dictates the number of digits which are displayed.

Box Select the color to use for the label text. Clicking the button displays the standard KDE color chooser.

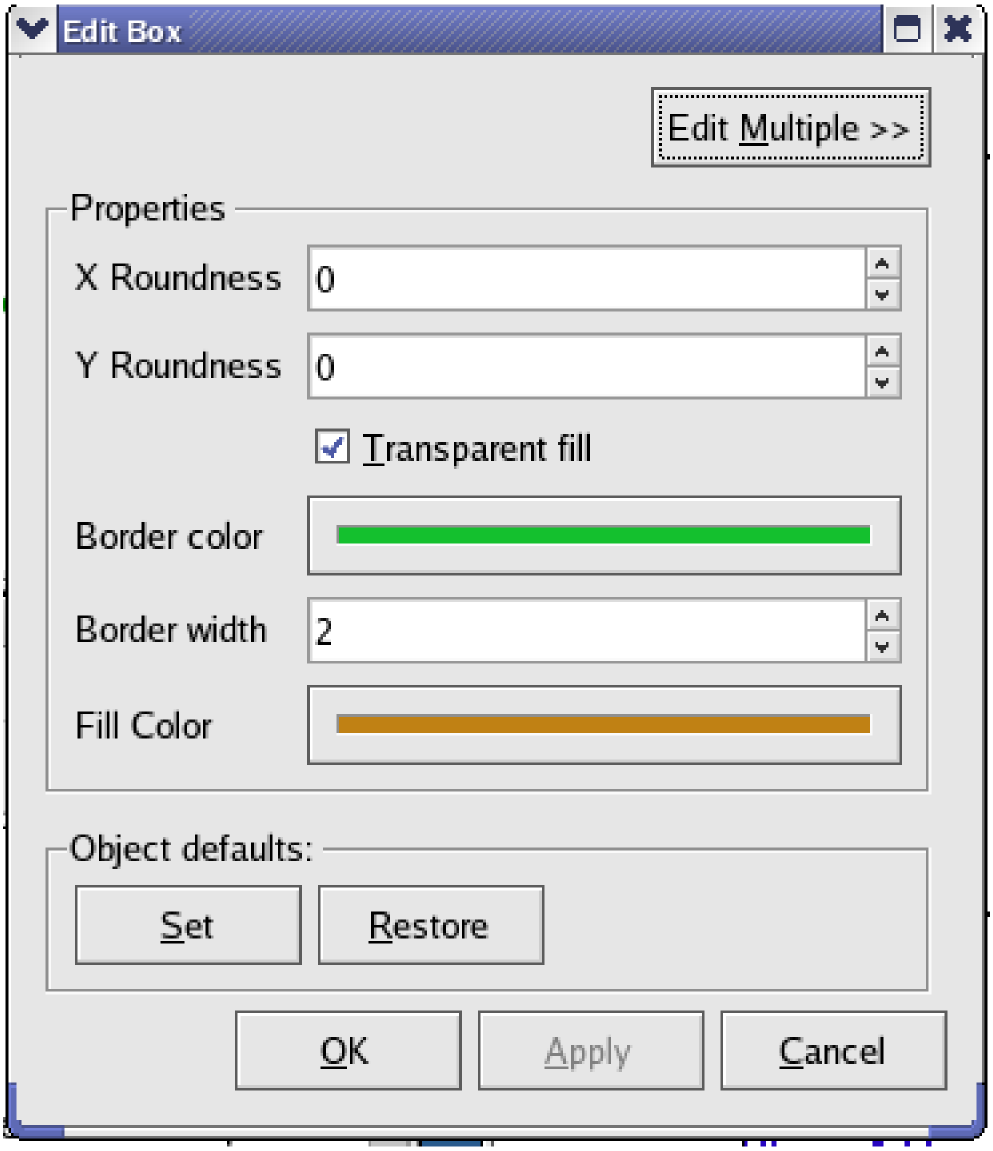
Once you are satisfied with the label settings, click the Apply button to view the label settings without closing the Edit Label dialog. Click the OK button to apply the label settings and close the dialog. Alternatively, you can click the Close button to close the Edit Label dialog without applying any label settings.

4.2.4 Boxes / Ellipses

Box and ellipse annotation objects are usually used to outline important features on a plot.

Command	
\Greekletternamne	Γ
\greekletternamne	γ δ
\approx	≈
\cdot	·
\geq, \ge	≥
\leq, \le	≤
\t	
\n	
\partial	∂
\sqrt	√
\inf	∞
\int	∫
\pm	±
\[, \]	[ ]
\sum	Σ
\prod	Π
y^{x}	y^x
y_{x}	y_x
\textit{’text’}	*italic*
\textbf{’text’}	**bold**
\underline{’text’}	under
\textcolor{’color’}{’text’}	color

Table 4.2: LaTeX Commands Supported in Label Text



X Roundness 0

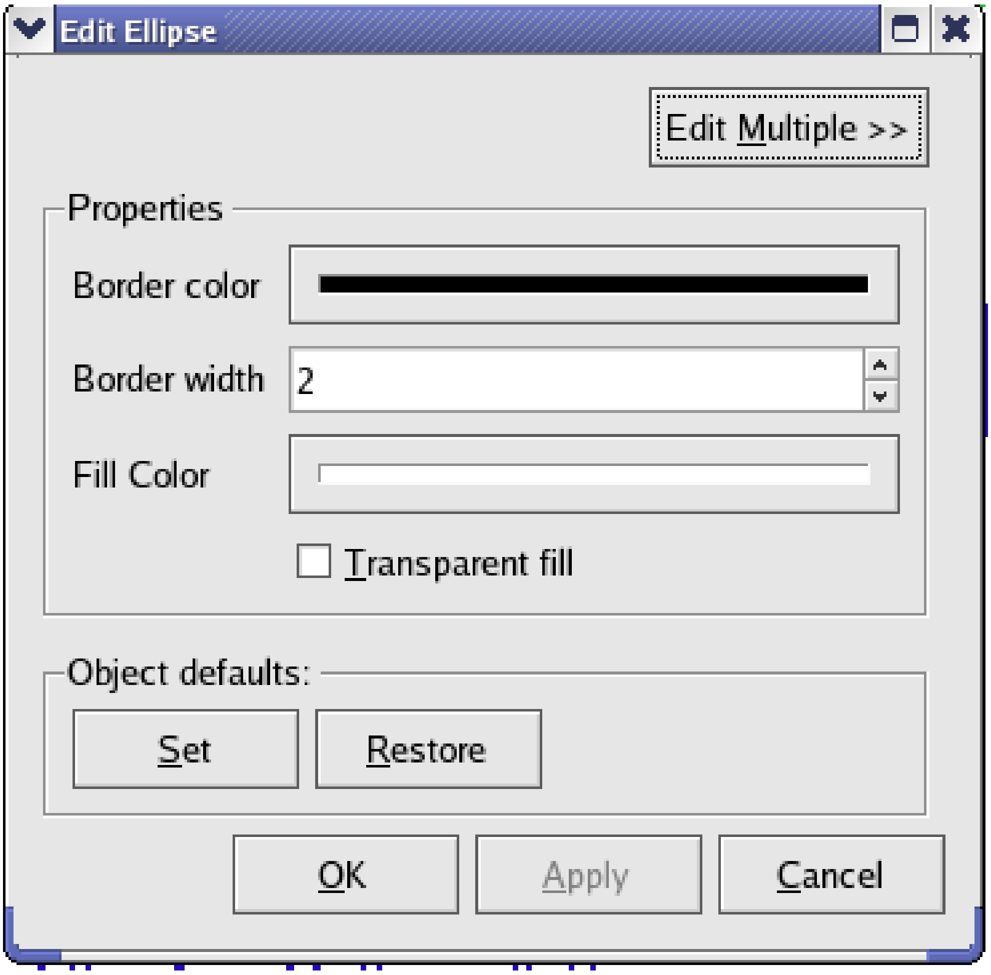
Y Roundness 0

☑ Transparent fill

Border color 🟩

Border width 2

Fill Color 🟧



Edit Ellipse

Edit Multiple >>

Properties

Border color

Border width 2

Fill Color

□ Transparent fill

Object defaults:

Set

Restore

OK

Apply

Cancel

The following are explanations of the dialog box elements.

Border Color The color of the outline of the box/ellipse object. Clicking the ■ button displays the standard KDE color chooser.

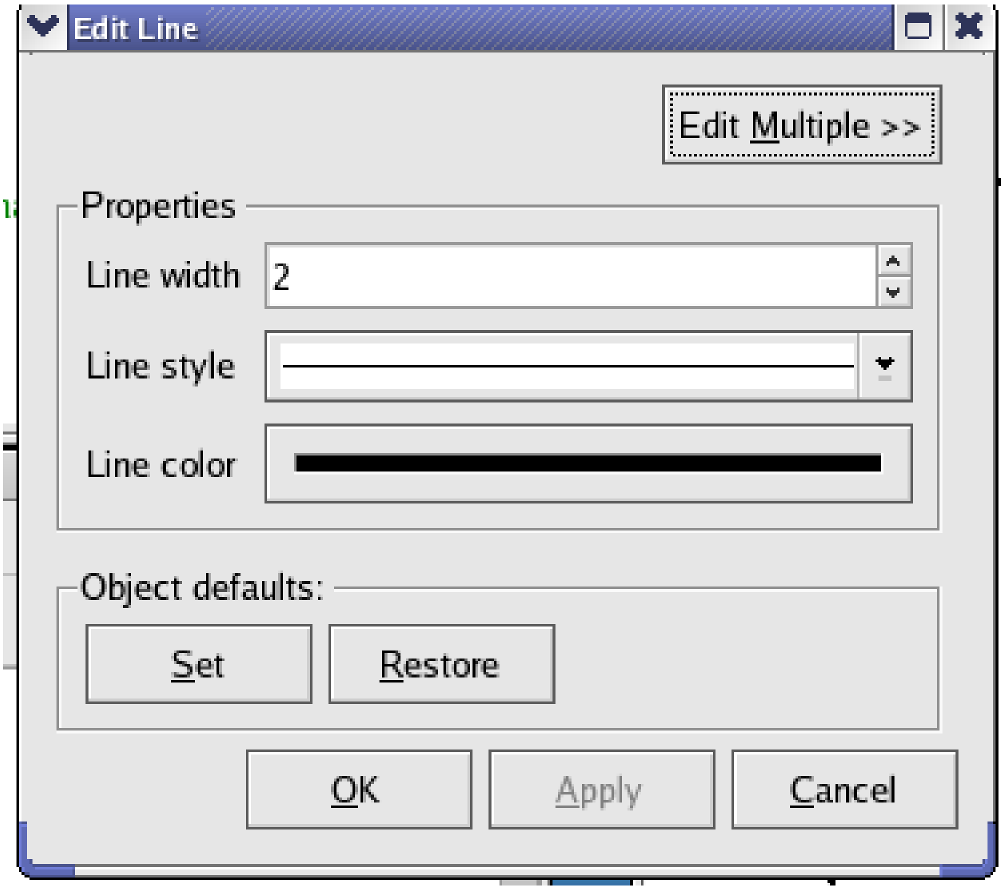
Border Width The width of the annotation's line/border (in pixels).

Fill Color / Transparent Fill The color of the contents of the box/ellipse. Clicking the ■ button displays the standard KDE color chooser. If you select Transparent Fill then the box/ellipse will be see-through.

X/Y Roundness These options are for boxes only. They determine the fractional rounding of the box's corners. Values of 0 result in standard corners (right angles), while values of 99 result in a maximally rounded box (an ellipse). For more information on these options, see the Qt documentation on QPainter::drawRoundRect.

4.2.5 Lines / Arrows

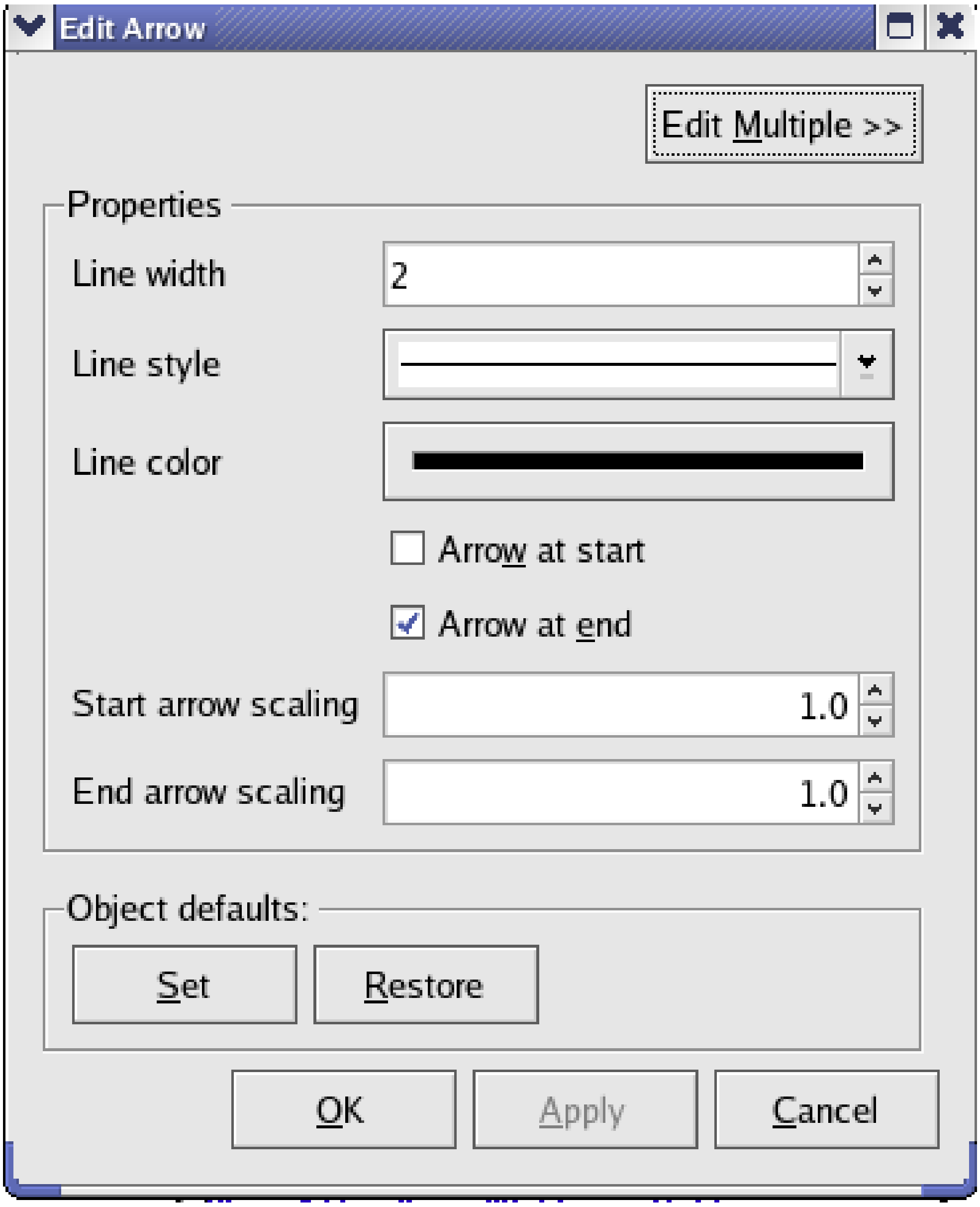
Line and Arrow annotation objects are usually used to highlight important features on a plot (by pointing to a particular peak, for example.).



Line width: 2

Line style: ———

Line color: ■



Line width 2

Line style

Line color

□ Arrow at start

☑ Arrow at end

Start arrow scaling 1.0

End arrow scaling 1.0

The following are explanations of the dialog box elements.

Line Color The color of the line/arrow object. Clicking the ■ button displays the standard KDE color chooser.

Line Style Here you can chose to make the line solid, dashed, dot-dashed, etc.

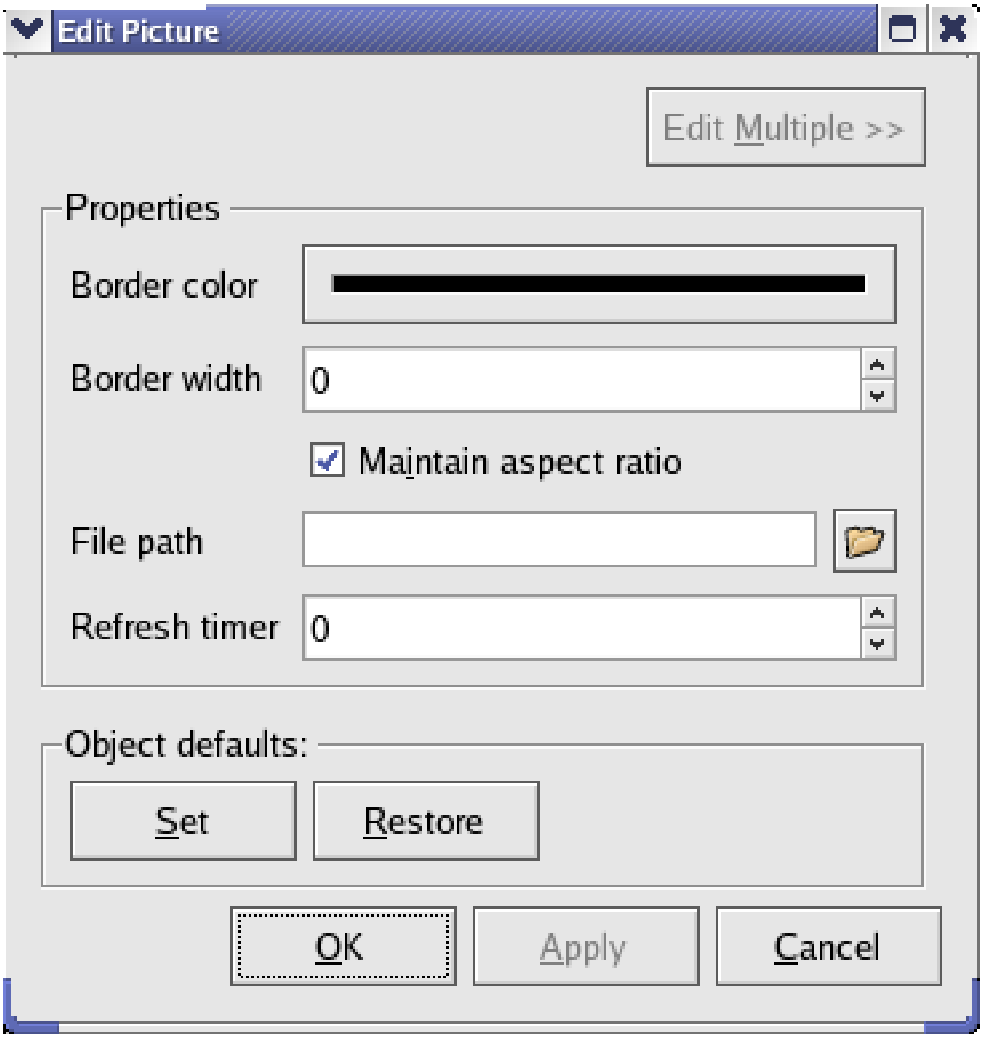
Line Width The width of the line (in pixels). This will also change the head size for an arrow object.

Arrow at start/end Determines where to place the arrow heads. The heads will always point away from the line.

Start arrow scaling A scaling factor for the arrow head sizes. This can be useful if your line width leads to a head size which is larger than you would like.

4.2.6 Pictures

Picture objects can be used to easily add an external graphic to your plot. More popular image formats are supported, depending on the libraries installed on your system.



Properties

Border color

Border width 0

☑ Maintain aspect ratio

File path

Refresh timer 0

Object defaults:

The following are explanations of the dialog box elements.

Border Color The color of the outline of the box/ellipse object. Clicking the ■ button displays the standard KDE color chooser.

Border Width The width of the annotation’s line/border (in pixels).

Maintain Aspect Ratio If this box is checked, then when resizing the picture object Kst will maintain the ratio between the width and height of the picture. You can temporarily disable (or enable) this option by holding down the **Shift** key as you resize the picture.

File Path The location of the image you would like to display.

Refresh Timer Set this to the period (in seconds) with which you would like Kst to check the image file for updates, and reload it if necessary. If the refresh timer is set to 0 then the image will only be reloaded after restarting Kst or opening and closing the picture object’s edit dialog.

4.2.7 Using KstScript to manipulate Annotation objects

You can construct and edit annotation objects from the corresponding classes in KstScript. See details in

- Label Class
- Box Class

- Ellipse Class
- Line Class
- Arrow Class
- Picture Class

To change a general property of an annotation object, such as moving or resizing an object, you may use methods and properties defined in the ViewObject Class, the super class of all annotation objects.

Example 4.2 Function for Creating and Resizing Annotation Objects

If you use the code below to create an annotation object, for example, a box, in tab W1:

```
b = new Box("W1");
```

You may notice that there is nothing in W1. However, the new box is actually created. It has a very small size and is sitting in the position (0,0), the upper left corner of the Kst window; therefore, it is necessary to move and resize it in order to make it visible. Notice that you will need to change the default settings for every annotation object you create, so it may be more convenient to use a KstScript function to do that. The following is an example function which will create a user specified annotation object, and then resize and move this object to the position (250,250) on the user specified window.

```
//annotation.js
//A(string) is the name of an annotation object
//W(string) is the name of a window
function createAnot(A, W)
{
//declare strings to hold names of possible annotation objects
lb = new String();
lb.value = "Label";
b  = new String();
b.value = "Box";
e = new String();
e.value = "Ellipse";
l = new String();
l.value = "Line";
a = new String();
a.value = "Arrow";
input = new String();//declare a string to hold the name of the user specified annotation ←
    object
input.value = A;
//create the user specified annotation object and resize it to let it become visible
if(input.value==lb.value)
{
anot = new Label(W);
anot.tagName = "new Label";
anot.resize(new Size(100,100));
anot.text = "Enter Text Here";
}
else if(input.value==b.value)
{
anot = new Box(W);
anot.tagName = "new Box";
anot.resize(new Size(100,100));
}
else if(input.value==e.value)
{
anot = new Ellipse(W);
anot.tagName = "new Ellipse";
anot.resize(new Size(100,100));
}
else if(input.value==l.value)
{
anot = new Line(W);
anot.tagName = "new Line";
anot.to = new Point(100,100);
anot.width = 2;
}
else
{
anot = new Arrow(W);
anot.tagName = "new Arrow";
anot.toArrow = true;
anot.to= new Point(100,100);
}
//move this annotation object to the position (250,250)
anot.move(new Point(250,250));
 }
```

After loading this function script in Kst, you can create an edited annotation object like this:

```
createAnot("Label","W1");
```

4.3 Zooming and Scrolling

Zooming and scrolling in plots is easy and intuitive with Kst. The following sections explain the different zooming and scrolling modes.

4.3.1 Mouse Zoom Modes

To access the different zoom modes, choose one of XY Mouse Zoom, X Mouse Zoom, or Y Mouse Zoom from the Plots menu. The different modes are described below.

4.3.1.1 XY Mouse Zoom

In XY Mouse Zoom mode, you can zoom into a specific rectangular area of the plot by simply clicking and dragging to draw a rectangle where desired. The x and y axes of the plot will change to reflect the new scale. This mode is often useful for quickly looking at an interesting area of the plot without having to specify exact axis scales.

4.3.1.2 X Mouse Zoom

In X Mouse Zoom mode, the y axis is fixed. Zooming in is performed by clicking and dragging a rectangular area; however, the upper and lower limits of the rectangle will always be equal to the current upper and lower limits of the y axis. This mode is often useful for looking at a certain time range, if the x axis happens to represent a time vector.

Tip
You can quickly switch to X Mouse Zoom mode by holding down **Ctrl**. The mouse cursor will change to indicate the new mode. Releasing **Ctrl** will return you to the previous mouse zoom mode.

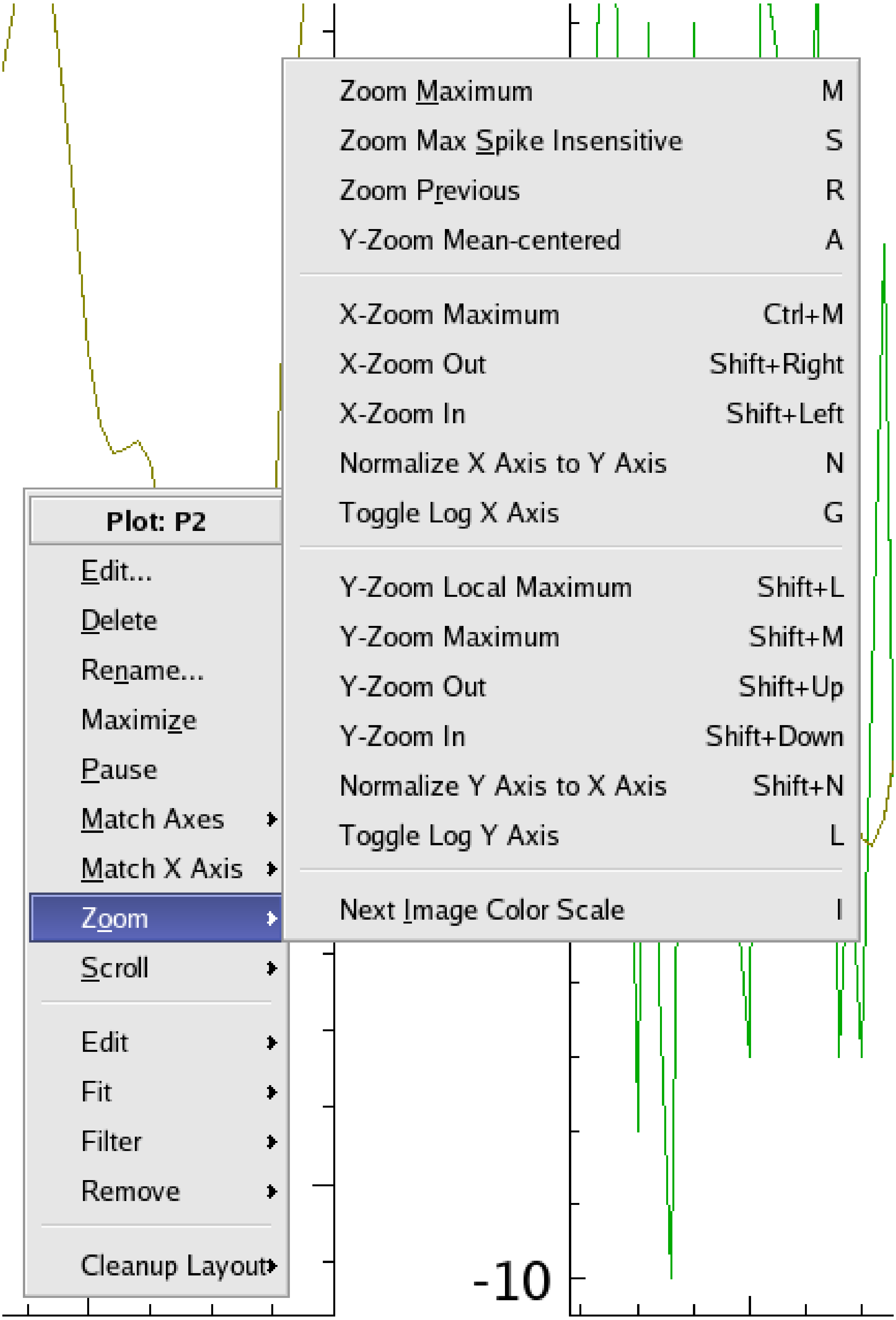
4.3.1.3 Y Mouse Zoom

In Y Mouse Zoom mode, the x axis is fixed. Zooming in is performed by clicking and dragging a rectangular area; however, the left and right limits of the rectangle will always be equal to the current left and right limits of the x axis. This mode is often useful for zooming in on data that is concentrated around a horizontal line.

Tip
You can quickly switch to Y Mouse Zoom mode by holding down **Shift**. The mouse cursor will change to indicate the new mode. Releasing **Shift** will return you to the previous mouse zoom mode.

4.3.2 Zoom Mode Context Menu

When in any of the mouse zoom modes, right clicking on a plot brings up some special zoom options, in addition to the basic context menu options.



Maximize Temporarily expands the plot so that it takes up the entire area of the window. Uncheck this option to return the plot to its previous size.

Pause Pauses automatic update of live data. This menu item duplicates the functionality of Pause from the Range menu.

Match Axes These options allow you to quickly change the axes of a plot to match those of another plot. Both the X and Y axes scales of the current plot will change to match those of plot which you choose from the submenu. Note that this does not permanently tie the axes scales together; changing the zoom on either plot will unmatch the axes scales again. To tie the axis scales of two or more plots together, use the Tied Zoom feature.

Match X Axis This option allows you to quickly change the X axis of a plot to match that of another plot, similarly to the Match Axes submenu.

Zoom (submenu) The Zoom submenu is described in the section entitled The Zoom Menu.

Scroll (submenu) This menu is described in the section entitled The Scroll Menu.

Edit (submenu) This submenu displays a list of objects available for editing in the underlying plot. Clicking on an object name from the submenu displays the Edit dialog for the object.

Fit (submenu) This submenu displays a list of curves which can be fit. Selecting a name from the submenu displays a New Plugin dialog which allows you to choose a fit plugin. The input vectors are prechosen based on the curve selected.

Filter (submenu) This submenu displays a list of data objects that can be filtered. Selecting a name from the submenu displays a New Plugin dialog which allows you to choose a filter plugin. The input vector to be filtered is selected based on the curve. In addition, curve appearance properties can be changed, as the Filter function creates vectors, and curves based on the vectors.

Remove (submenu) This submenu displays a list of curves currently in the plot. Clicking on a curve name from the submenu removes the curve from the plot (the curve itself, however, is not removed as a data object).

4.3.3 The Zoom Menu

The Zoom menu can be accessed by right-clicking on a plot and selecting the Zoom submenu from the context menu. A list of zoom actions and their corresponding keyboard shortcuts will be displayed. These actions are described below.

Zoom Action	Keyboard Shortcut	Description
Zoom Maximum	**M**	Sets both the x axis and the y axis scales so that all data points are displayed. This is equivalent to the AutoScale setting of the Plot Dialog.
Zoom Max Spike Insensitive	**S**	Sets both the x axis and the y axis scales so that most data points are displayed. Spikes, or sudden increases or decreases in x or y values, are excluded from the plot display.
Zoom Previous	**R**	Returns to the most recent zoom setting used.
Y-Zoom Mean-centered	**A**	Sets the Y axis so that the mean of the y values in the plot is centered vertically. The actual zoom level is not changed.
X-Zoom Maximum	Ctrl+M	Sets the x axis scale such that the x values for all data points are between the minimum and maximum of the x axis. The y axis scale is unaltered.
X-Zoom Out	Shift+Right	For a non-logarithmic x axis, increases the length of the x axis by a factor of approximately 0.5, without changing the midpoint of the x axis. The y axis scale is unaltered.

Zoom Action	Keyboard Shortcut	Description
X-Zoom In	Shift+Left	For a non-logarithmic x axis, decreases the length of the x axis by a factor of approximately 0.5, without changing the midpoint of the x axis. The y axis scale is unaltered.
Normalize X Axis to Y Axis	**N**	Sets the length of the X Axis to be equal to that of the Y Axis, maintaining the current center.
Toggle Log X Axis	**G**	Enables or disables using a logarithmic scale for the x axis.
Y-Zoom Maximum	Shift+M	Sets the y axis scale such that the y values for all data points are between the minimum and maximum of the y axis. The x axis scale is unaltered.
Y-Zoom Out	Shift+Up	For a non-logarithmic y axis, increases the length of the y axis by a factor of approximately 0.5, without changing the midpoint of the y axis. The x axis scale is unaltered.
Y-Zoom In	Shift+Down	For a non-logarithmic y axis, decreases the length of the y axis by a factor of approximately 0.5, without changing the midpoint of the y axis. The x axis scale is unaltered.
Normalize Y Axis to X Axis	**Shift+N**	Sets the height of the Y Axis to be equal to that of the X Axis, maintaining the current center.
Toggle Log Y Axis	**L**	Enables or disables using a logarithmic scale for the y axis.
Scale	**I**	Only has any effect on an image; cycle through the extreme values of the image matrix to redefine the threshold values of the image scale every time; its purpose is to let the user better view the intermediate values of an image

Many of the zoom actions are best used in conjunction with the various mouse zoom modes.

4.3.4 The Scroll Menu

Functions for scrolling a plot are available by right-clicking a plot and selecting the Scroll submenu from the context menu. Most scrolling functions and their keyboard shortcuts should be self-explanatory. To quickly jump to a plot marker, select Next Plot Marker (to jump right) or Previous Plot Marker (to jump left). For more information on plot markers, see the documentation on the Markers section of the Edit Plot Dialog.

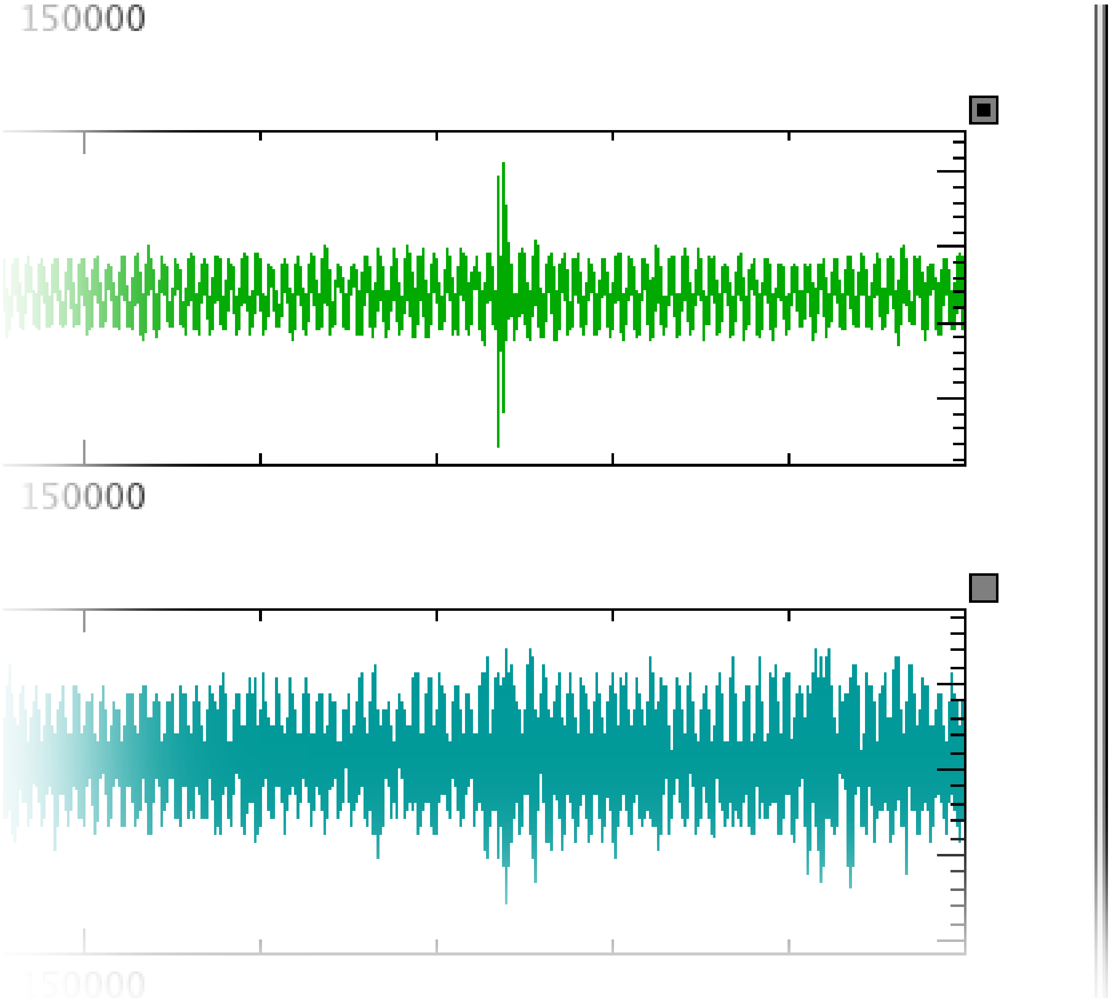
Assuming non-logarithmic axis scales are used, each function scrolls the plot in the indicated direction by 0.1 of the current length of the x axis (when scrolling left or right), or 0.25 of the current length of the y axis (when scrolling up or down).

Tip
You can also scroll left or right in a plot by using the mouse wheel (if available).

Tip
To quickly go forwards or backwards along the x axis, select Back 1 Screen or Advance 1 Screen from the Range menu. The keyboard shortcuts, Ctrl+Left and Ctrl+Right respectively, can be used as well.

4.3.5 Tied Zoom

When looking at two or more related plots (for example, two curves on separate plots both dependent on the same time vector), it can sometimes be useful to zoom or scroll the plots simultaneously. This is possible using Kst's tied zoom feature. To activate tied zoom, click the small square at the top-right corner of the plots you wish to tie together. The squares will change colour to indicate the plots are tied, as shown below.



Zooming and scrolling actions performed on one plot in a group of tied plots will be performed on all the plots in the group. To remove a plot from the tied group, simply click on the small square at the top-right corner of the plot again. The square will become transparent to indicate that the plot is not tied.

Tip

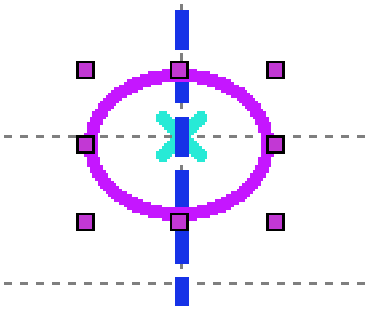
To quickly tie or untie all plots (including plots in other windows), select Tied Zoom from the Plots menu, or the corresponding icon on the Ksttoolbar.

4.4 Manipulating Plot Layout

View objects in a Kst window are arranged in layers. Each object is positioned on one layer, and each layer contains only one object. Thus, view objects can overlap, with plots in higher layers taking precedence in visibility over those in lower ones. To change the layout of plots in Kst, layout mode must be activated. Layout mode can be toggled by selecting Layout Mode from the Plots menu.

4.4.1 Moving and Resizing View Objects

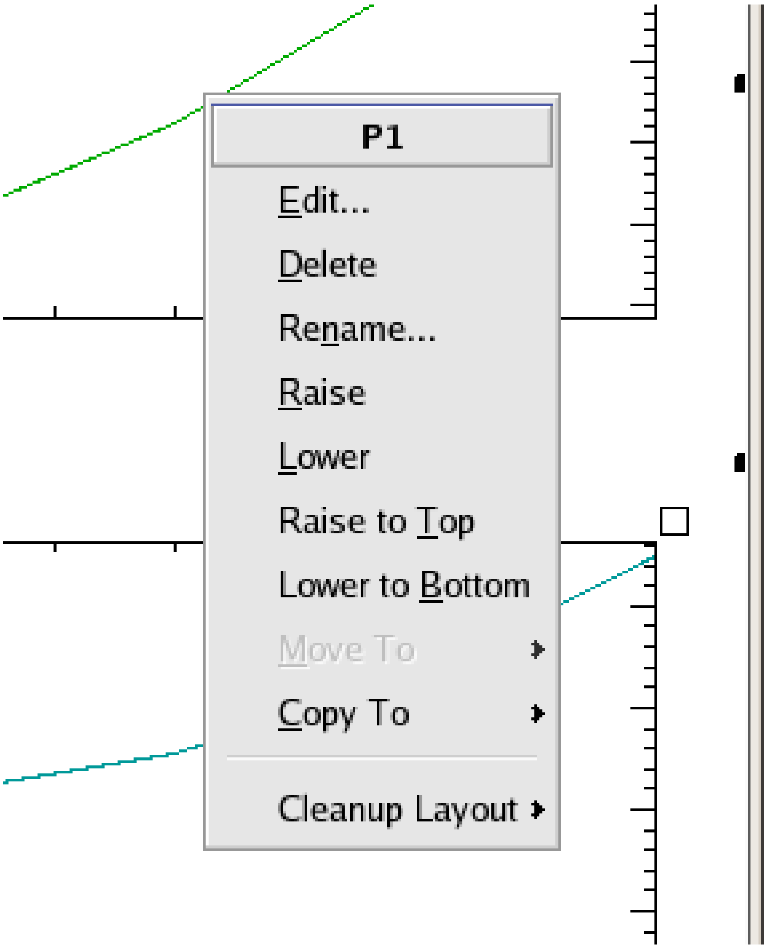
Moving and resizing view objects in layout mode is analogous to moving and resizing regular windows. To move a plot, simply click anywhere on the desired plot and drag the plot. An outline of the plot will appear, indicating where the plot will be placed, as shown in the following image.



You can drag plots anywhere within the current window. To resize a plot, move the mouse cursor to any edge or corner of the plot. The cursor will change to indicate that the plot can be resized. Click and drag the outline of the plot until the plot is the desired shape and size.

4.4.2 Layout Mode Context Menu

While layout mode is activated, right-clicking on any view object displays the basic context menu with several additional options. The selections available in this menu are listed below.



Raise Raises the plot one layer up.

Lower Lowers the plot one layer down.

Raise to Top Raises the plot to the top-most layer, guaranteeing that it will be visible.

Lower to Bottom Lowers the plot to the bottom-most layer.

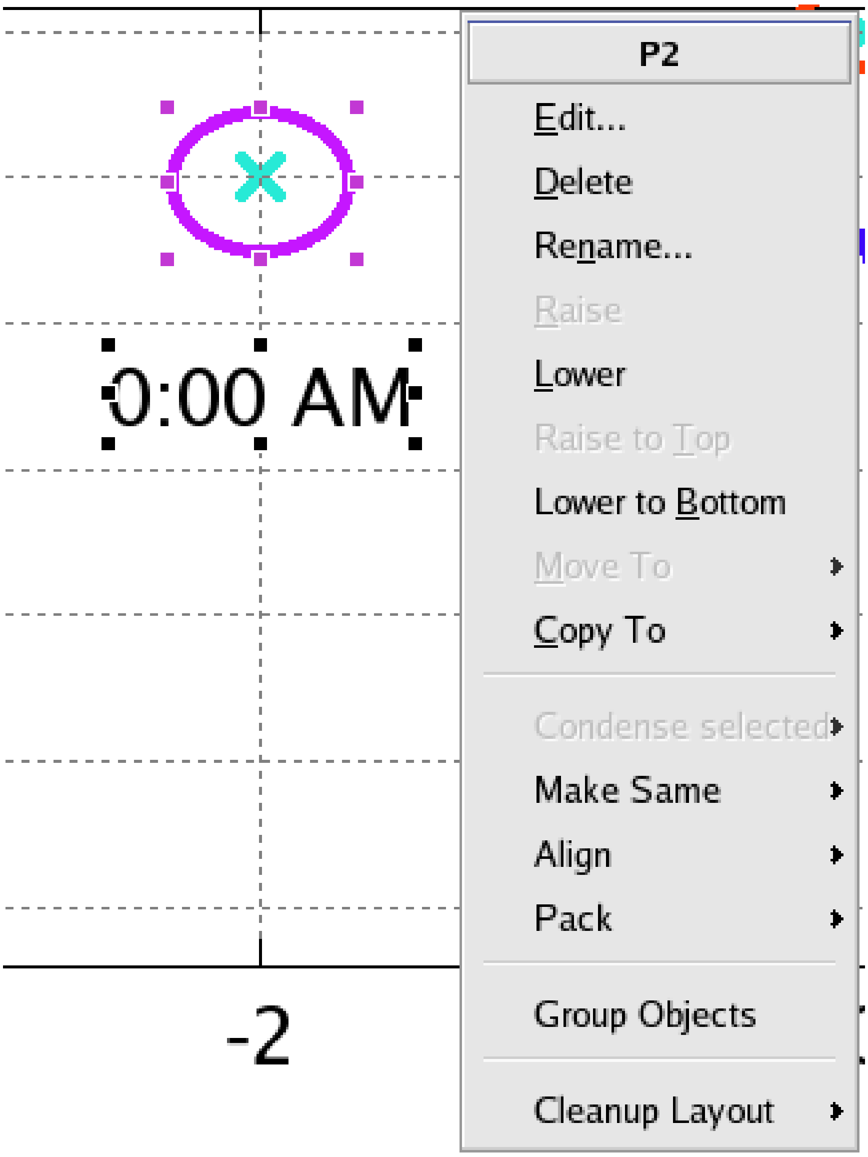
Move To Selecting a window name from this submenu moves the plot to the window with the corresponding name.

Copy To Selecting a window name from this submenu copies the plot to the window with the corresponding name.

If multiple view objects are selected in data mode, additional context menu items become available.

4.4.3 Selecting Multiple View Objects and Grouping

Two or more view objects can be selected together in layout mode. To select the objects, hold down the **Shift** key and either sequentially click on each you wish to select, or drag to draw a dotted rectangle around a group of view objects. The selected objects will be indicated by corner highlights, as shown below. Multiple selection allows you to move several view objects simultaneously by dragging any single object that is part of the selection. In addition, the layout mode context menu contains three additional items when multiple objects are selected. These items are described below.



Condense Selected This option is available if the view objects which are selected are all plots. Selecting the X-axis or Y-axis options from this submenu removes either the X axis of the upper plot(s) or the Y axis of the rightmost plot(s) respectively. Note: this option does not rescale the plot data, so make sure that the original axes scales for all the plots are identical.

Make Same Resizes all the selected objects in the chosen dimension(s) to match the dimension(s) of the view object from which the context menu was invoked. For example, right-clicking on a plot in the group and selecting Make Same → Width will match the width of all the plots in the group to the width of the first plot. If Size is chosen, both the width and height of each object in the selection will change.

Align Aligns all objects in the selection to the object directly underneath the context menu. Select from Left, Right, Top, or Bottom.

Pack (submenu) The Pack submenu contains two functions. Pack Horizontally divides the entire width of the object group evenly between the individual object widths, while Pack Vertically divides the entire height of the group evenly between the individual heights. Packing plots usually results in a staggered or ‘checkered’ pattern. To align the plots in a grid, use the Cleanup Layout function of the layout mode context menu.

Group Objects Groups all the selected objects together. A group of view objects can be resized or moved as a whole in the same way the layout of a object can be changed. Grouped plots remain grouped even when not in layout mode, so the Maximize or Delete functions of the plot context menu can be used on an entire group of plots.

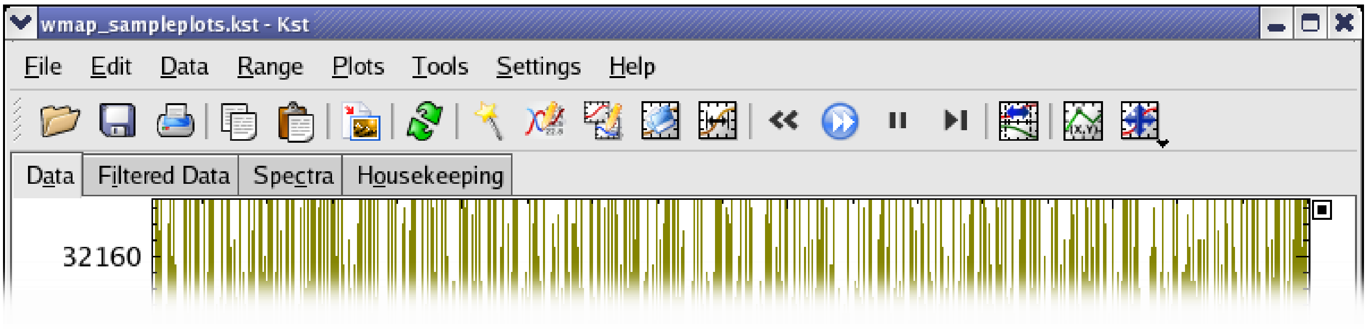
To dissociate a group, simply right-click on the group and select Ungroup.

4.5 View Tools

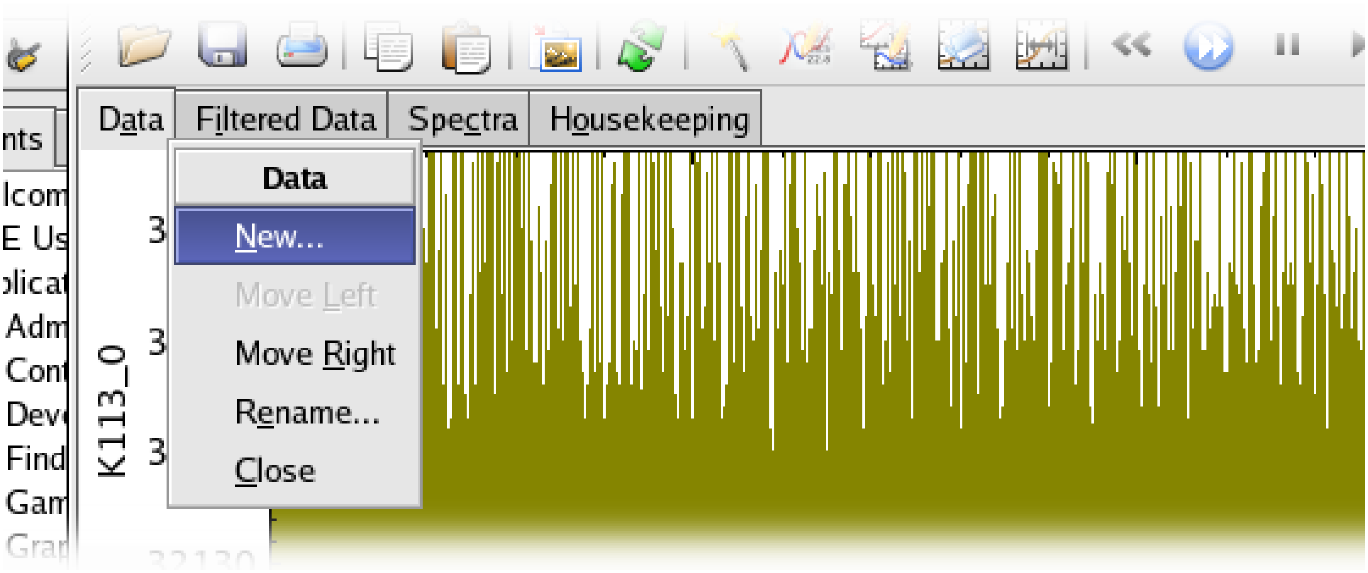
4.5.1 Window Tabs

Kst allows the use of tabs to organize plots into separate windows for efficient viewing and manipulation.

To switch between plot windows in tab page mode, you can click on the tabs corresponding to each window, as shown below.

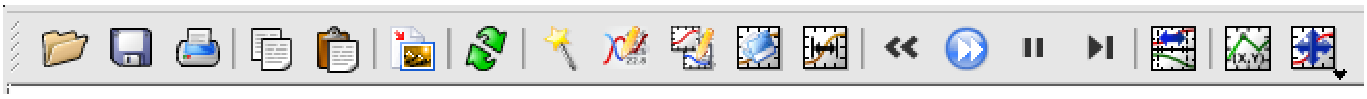


- To create a new tab, select File → New Tab, or right click on the tab bar and select New... from the tab context menu which appears.
- The tab context menu allows you to create, delete and rename tabs, as well as shift their order.

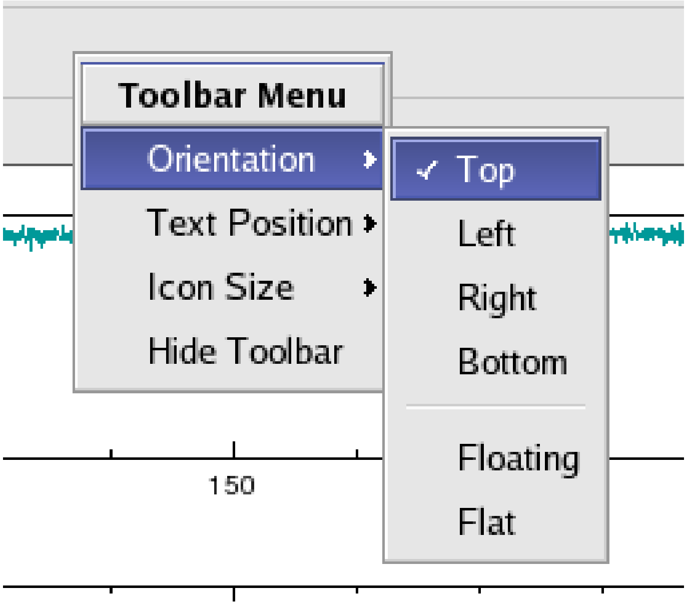


4.5.2 The Toolbar

The toolbar provides quick access to commonly used menu functions in Kst. To toggle the toolbar, select Settings → Show Toolbar.

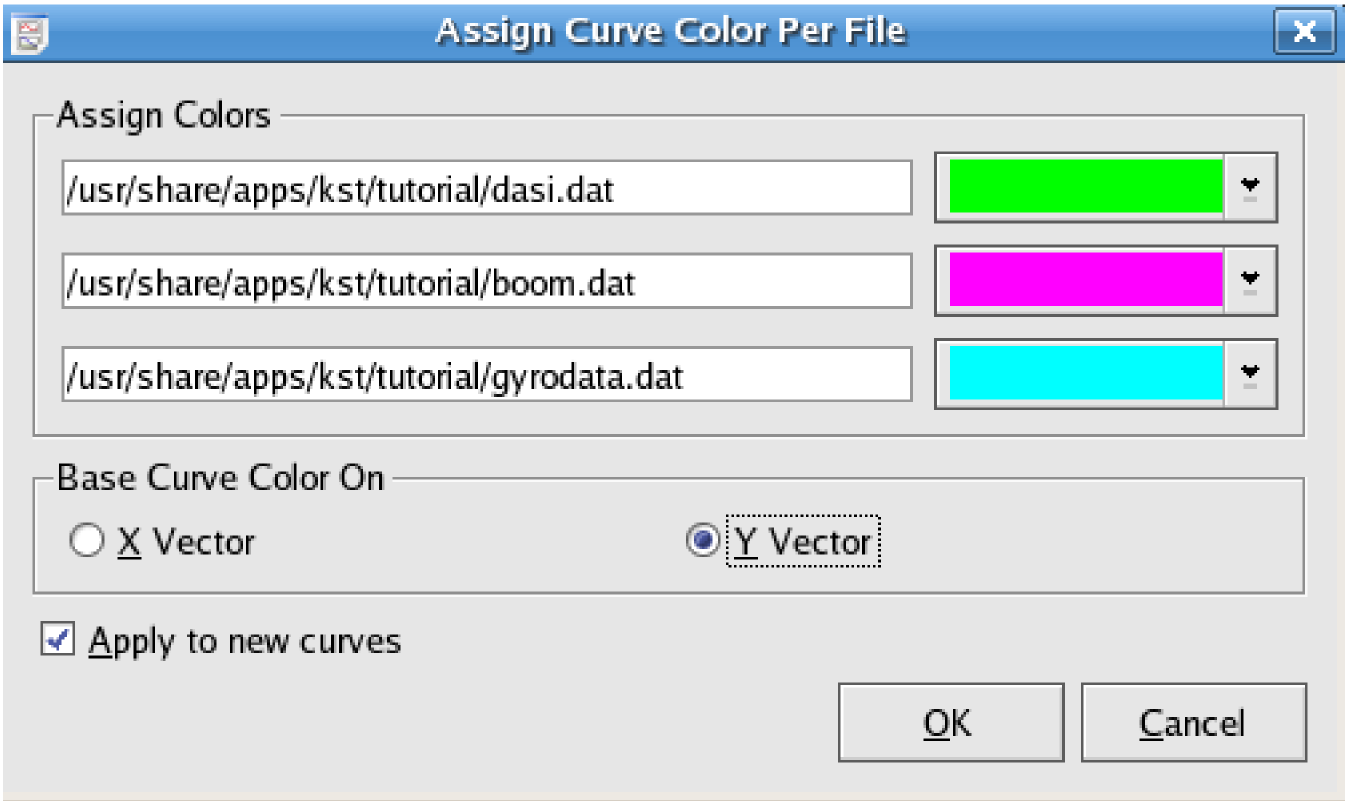


The toolbar menu, accessible by right-clicking anywhere on the toolbar, contains options for modifying the appearance and location of the toolbar.



4.5.3 Assign Curve Color From File

This dialog allows you to assign colors to curves based on the file from which their data originates. You can access it by selecting the Tools → Assign Curve Color From File...



Assign Colors

/usr/share/apps/kst/tutorial/dasi.dat 🟩

/usr/share/apps/kst/tutorial/boom.dat 🟪

/usr/share/apps/kst/tutorial/gyrodata.dat 🟦

Base Curve Color On

○ X Vector ● Y Vector

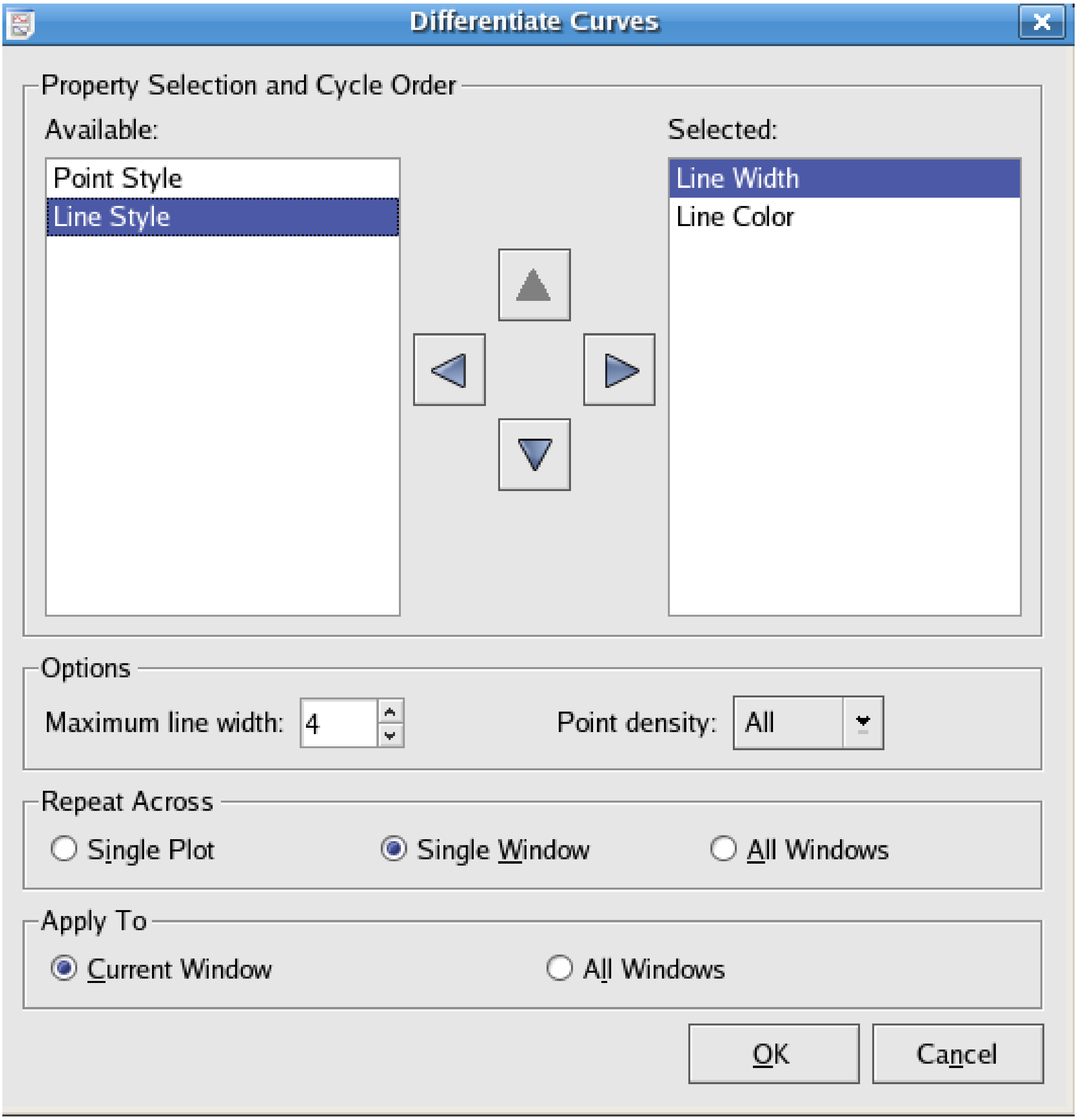
☑ Apply to new curves

The dialog will present a list of currently used data files and colour choices. Select the colours you would like, and choose the X Vector/Y Vector radio button based on whether you would like the curves colored based on the data file of their X or Y vector if the X and Y vectors for a curve come from two different data files. Then click OK.

If Apply to new curves is checked, then the chosen color pattern will also apply to curves that will be constructed later form those data files.

4.5.4 Differentiate Between Curves

This dialog allows you to easily make plotted curves visually differentiable by cycling through their style, color, and width properties. You can access it by selecting the Tools → Differentiate Between Curves...



Differentiate Curves

Property Selection and Cycle Order

Available:

- Point Style
- Line Style

Selected:

- Line Width
- Line Color

Options

Maximum line width: 4

Point density: All

Repeat Across

○ Single Plot ● Single Window ○ All Windows

Apply To

● Current Window ○ All Windows

To use this dialog, move some subset of the curve properties into the selected list. They will be cycled through in the order they appear in the selected list.

When cycling through Line Width, a maximum Line width must be set. The curves will then cycle through width 1 and this maximum width.

The Repeat Across options allows you to have the differentiation performed separately for each plot (Single Plot), separately for each window (Single Window), or continuously through All Windows.

The Apply to options allow you to determine whether the differentiation will be performed in All Windows, or only the Current Window.

Click OK or Apply to perform the differentiation, or Cancel to abort.

Chapter 5

Saving and Printing

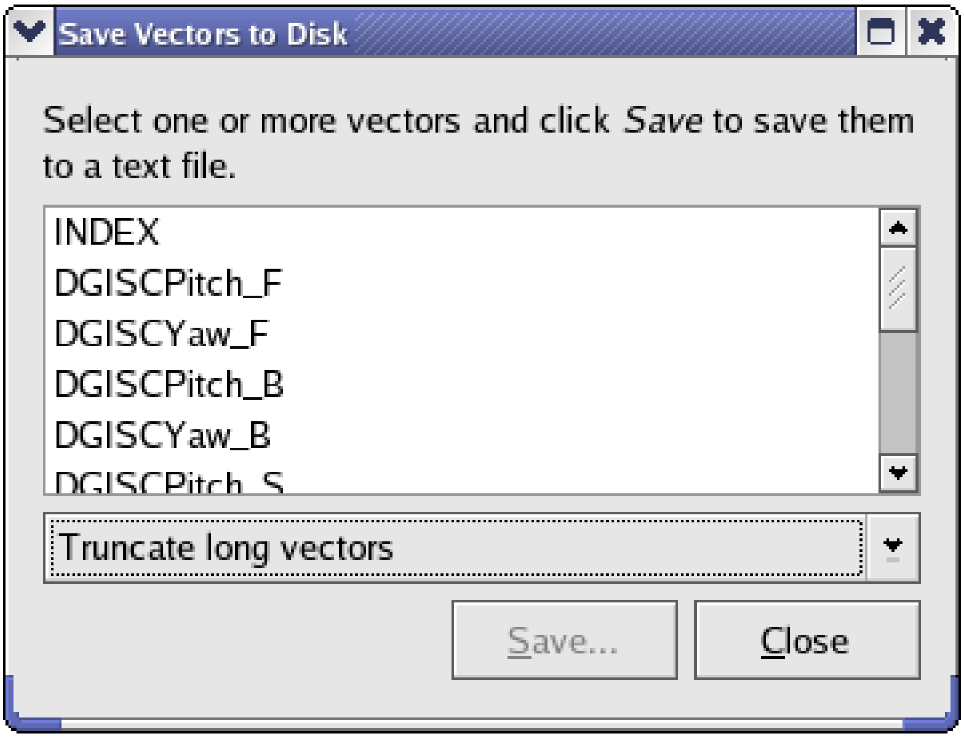
5.1 Saving Sessions

Selecting Save or Save As... (depending on whether or not you wish to save to a new file) from the File menu allows you to save a Kst session. This contains all of the plot and layout information which Kst requires to reconstruct your session (as long as the data files used in the sessions are still present when Kst is reopened!). Browse to the location you wish to save the plot definition, enter a filename, and click Save. The plot definition will be saved as a `*.kst` file.

5.2 Exporting Vectors

Vectors in Kst can be exported. This can be useful, for example, if you would like to use a vector generated by Kst in another application.

To export or save vectors to a file, select Save Vectors to Disk... from the File menu. A dialog will be displayed allowing you to choose which vectors to export.



Select one or more vectors and click *Save* to save them to a text file.

INDEX
DGISCPitch_F
DGISCYaw_F
DGISCPitch_B
DGISCYaw_B
DGISCPitch_S

You can select one or more vectors from the list, and click Save... to export them to an ASCII file. This file will contain the data in rows, with one column for each vector exported. The first two lines of this file are commented out with ';' symbols. The first line contains the names of the exported vectors, and the second row is a visual separator.

Here is an example, for two 5 element vectors named 'HWYD' and 'PWMY'.

```
; HWYD PWMY
; ---------
-2 4
-1 1
0 0
1 1
2 4
```

If you have selected multiple vectors to export, a drop down menu at the bottom of the Save Vectors to Disk dialog becomes enabled. There are three options available...

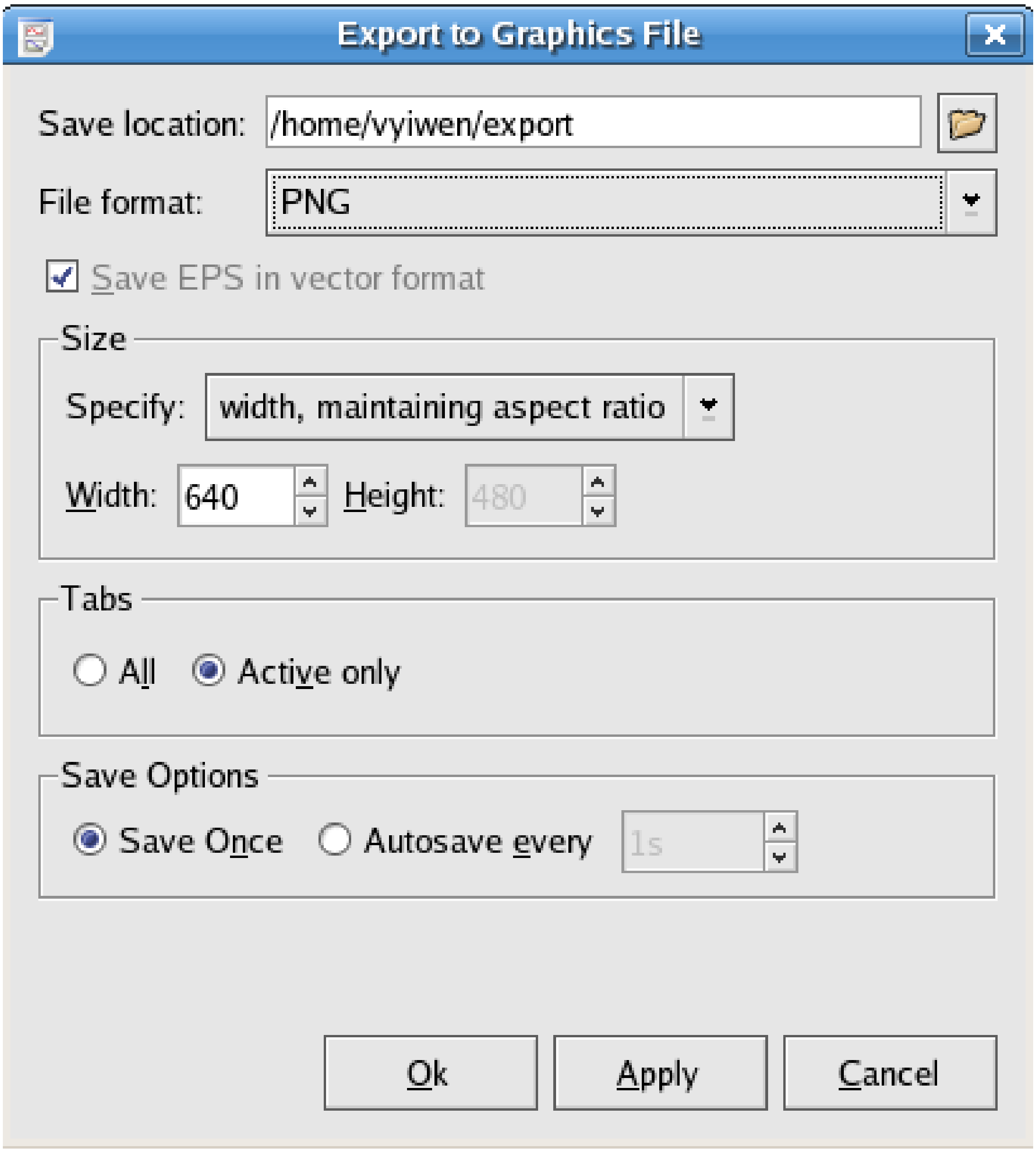
Truncate long vectors. All of the exported vectors will be truncated to match the length of the shortest vector selected.

Interpolate short vectors. All of the exported vectors will be interpolated to match the length of the longest vector selected.

Save in separate files. Each vector will be saved in a separate file. After hitting the Save... button you will be promted to enter a base file name, and then Kst will save each selected vector in a separate file named by appending `.1`, `.2`, `.3`, ... to this base.

5.3 Exporting Plot Images

To export a Kst plot to a graphics file, select File → Export to Graphics File... The following dialog box will appear.



Export to Graphics File

Save location: /home/vyiwen/export

File format: PNG

☑ Save EPS in vector format

Size

Specify: width, maintaining aspect ratio

Width: 640 Height: 480

Tabs

○ All ● Active only

Save Options

● Save Once ○ Autosave every 1s

The following settings and options are available when saving:

Save location The path to the graphics file. Enter a location manually, or click the button to right of the text box to browse to a location. The default name of graphic file is export.

File format The format to use for the saved graphics file. The drop down list shows you all the available formats. PNG is suggested if you are not sure which format to use. If you select EPS format, the Save EPS in vector format option becomes available, which allows you to save the image as a vector graphic which can be cleanly scaled later.

Size Select the dimensions, in pixels, of the graphic. The drop down list gives several self-explanatory options for specifying this size.

Tabs Select whether to export only the plots in the current window, or all plots. If All is selected, one graphics file will be created for each window, each with filename `file_n.fmt`, where `file` is the filename specified in Save location, `n` is an index number assigned to each window, and fmt is an extension representing the selected file format.

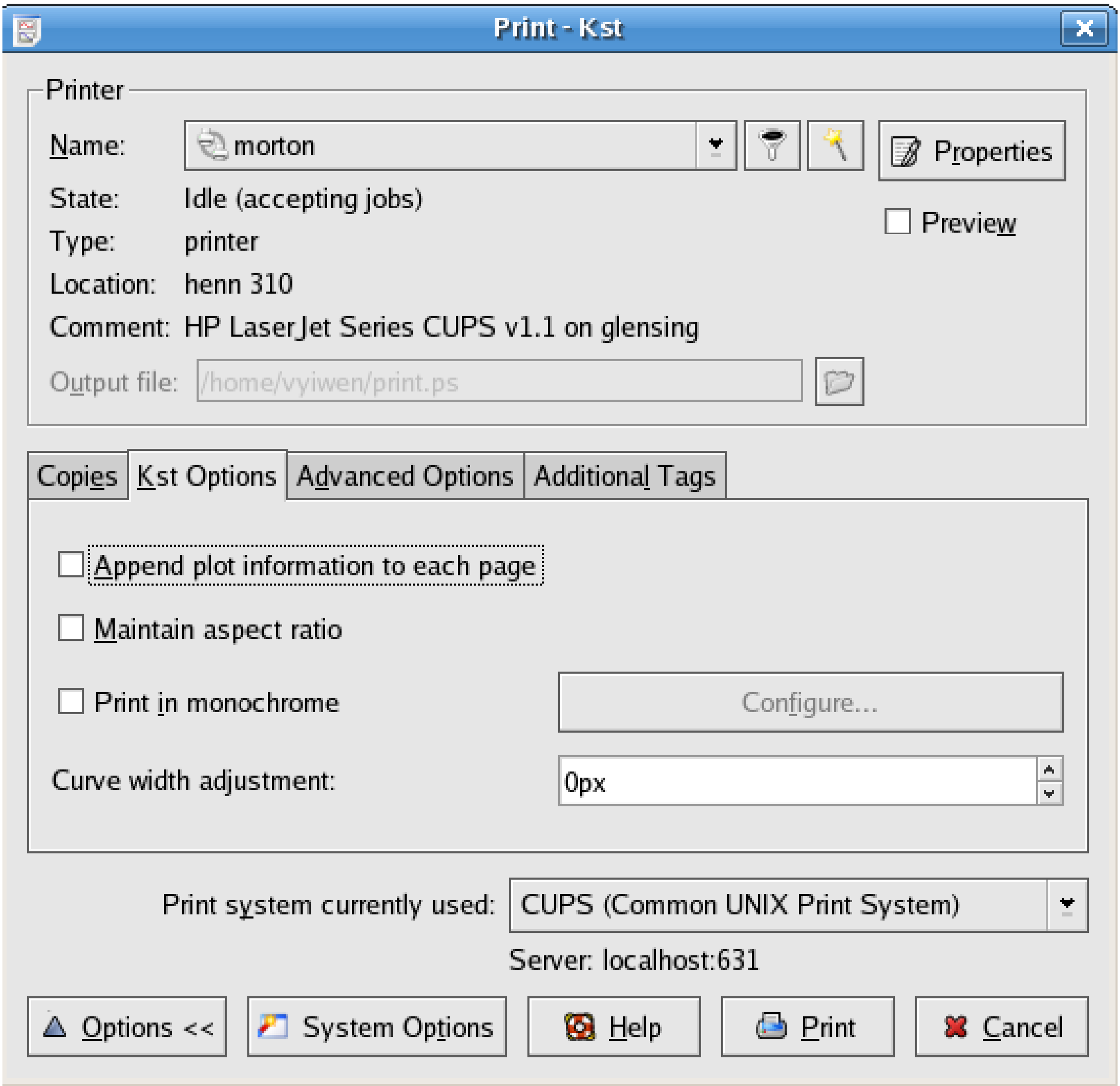
Autosave Check this option if you want Kst to automatically save the plot to the specified file on a specified time interval.

Save interval The time interval used to perform autosave, in seconds.

Click Save Once to immediately perform a save with the specified options and settings. Click Close to close the dialog box (autosave will still be performed at the specified intervals, if selected).

5.4 Printing

To print all the plots in the current window, select Print... from the File menu. A standard KDE print dialog will be displayed. Some Kst-specific options can be set by clicking on the Options » button, and selecting the Kst Options tab. The options are:



Print - Kst

Printer

Name: morton

State: Idle (accepting jobs)

Type: printer

Location: henn 310

Comment: HP LaserJet Series CUPS v1.1 on glensing

Output file: /home/vyiwen/print.ps

□ Preview

Copies | Kst Options | Advanced Options | Additional Tags

□ Append plot information to each page

□ Maintain aspect ratio

□ Print in monochrome

Curve width adjustment: 0px

Print system currently used: CUPS (Common UNIX Print System)

Server: localhost:631

Append plot information to each page If selected, a footer showing page number, plot name and the date and time is printed on each page.

Maintain aspect ratio If selected, the aspect ratio of the plots in the Kst window is retained in the print output. If unselected, the plots are resized to fill the printed page in both directions.

Print in monochrome If selected, the plot is printed in monochrome (black-and-white). If not, the colors of the on-screen plot are used. If the monochrome options is selected the Configure... becomes enabled. Clicking this button allows you to tell Kst how you would like curve appearance properties to be cycled through. This can compensate for the colour information which is lost in monochrome printing.

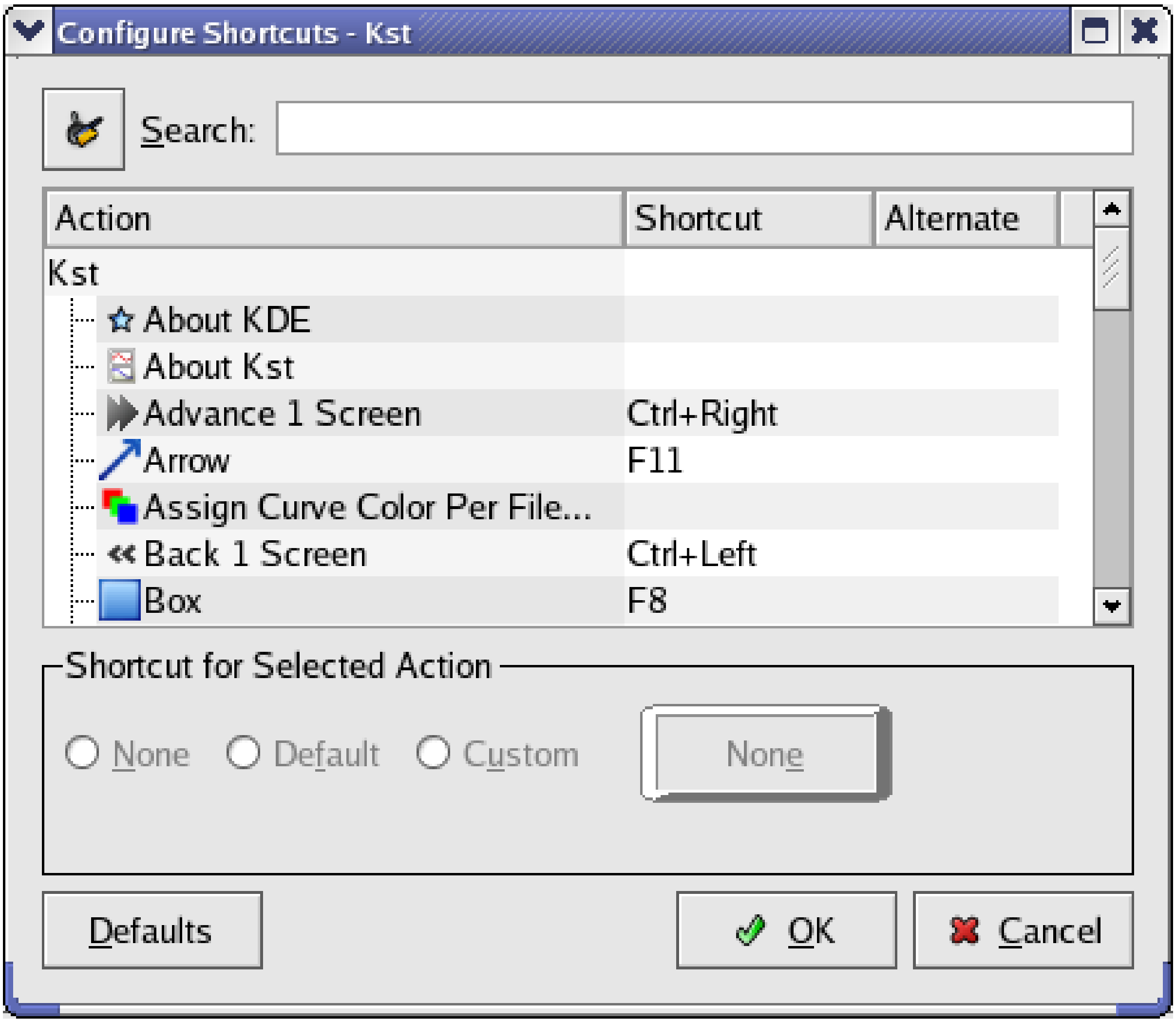
Curve width adjustment Adjusts the width of all curves in the printed output by this number of pixels relative to the width of the curve in the plot window. You can set a negative value to reduce the width of the curves in the print output.

Chapter 6

Settings and Logs

6.1 Shortcuts

You can create custom shortcut keys to make working in Kst more efficient. To modify the shortcut keys, select Settings → Configure Shortcuts.... The following dialog will be displayed.


Configure Shortcuts - Kst
Search:

Action	Shortcut	Alternate
Kst		
About KDE		
About Kst		
Advance 1 Screen	Ctrl+Right	
Arrow	F11	
Assign Curve Color Per File...		
Back 1 Screen	Ctrl+Left	
Box	F8	

Shortcut for Selected Action
○ None ○ Default ○ Custom
None
Defaults
OK
Cancel

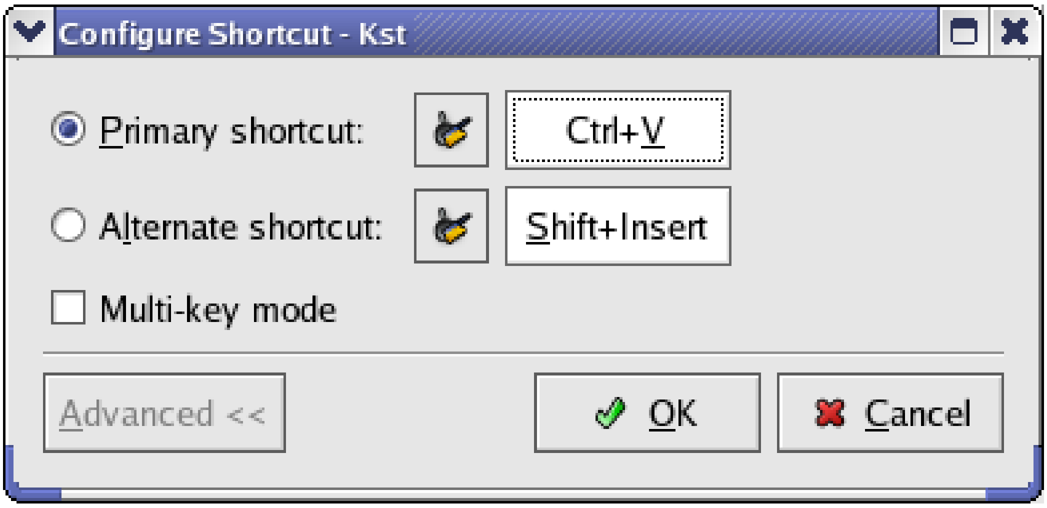
A list of available actions and their associated shortcuts are shown in the dialog. Each action is allowed two shortcuts—a primary shortcut and an alternate shortcut. Either shortcut can be used at any time. To define or change a shortcut for an action, simply highlight the appropriate row in the shortcut list. Under Shortcut for Selected Action, there are three choices:

None Select this option to disable any shortcut keys for the selected action.

Default Select this option to use the default shortcut key(s) for the selected action. Note that not all actions have default shortcuts defined.

Custom Select this option to define a custom shortcut for the selected action.

If you select the Custom option, double click on a shortcut, or press the key-shaped button then the Define Shortcut dialog will appear.



Configure Shortcut - Kst

● Primary shortcut: Ctrl+V

○ Alternate shortcut: Shift+Insert

□ Multi-key mode

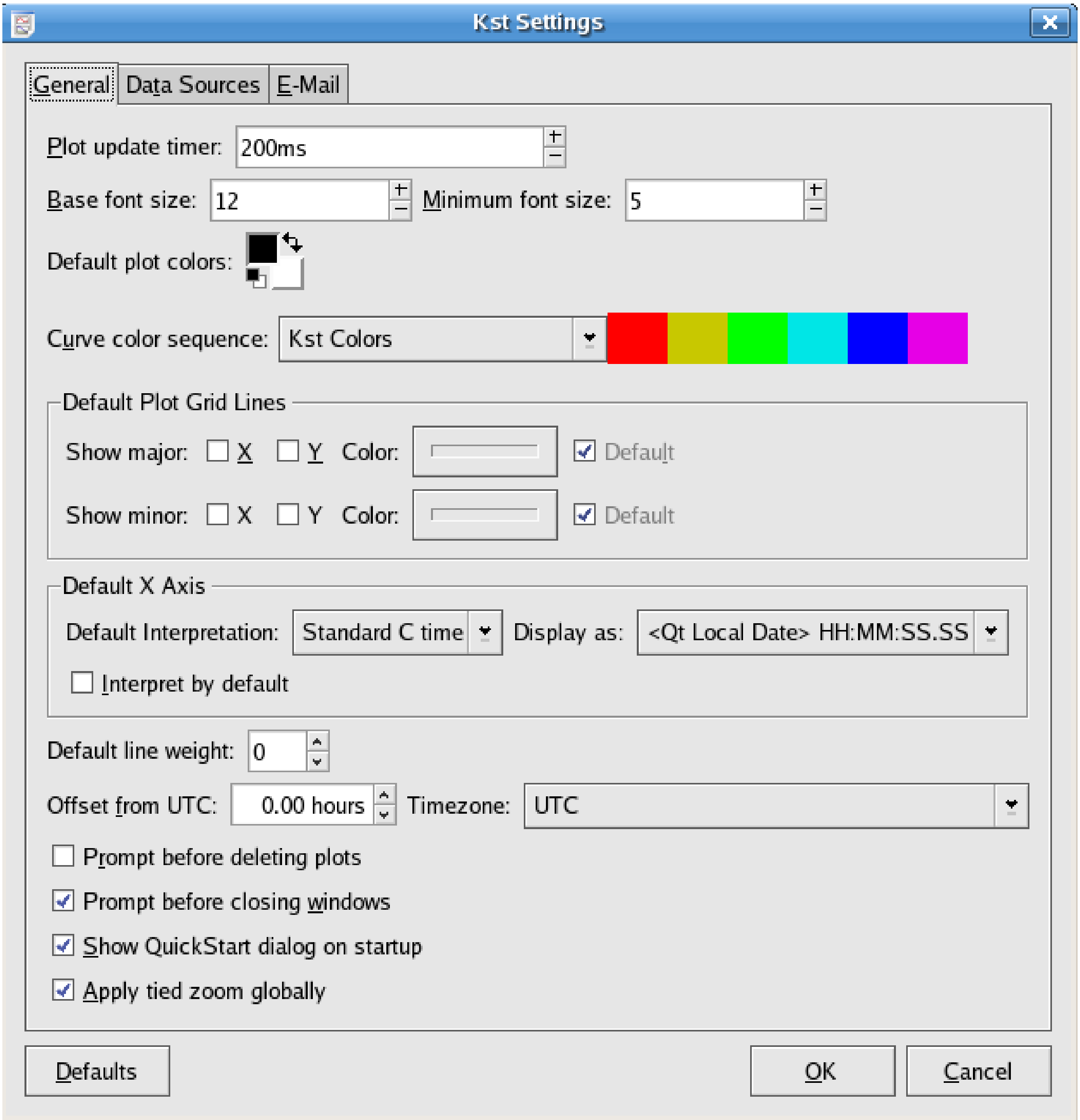
To define a key combination, select Primary or Alternate. If you wish to define a key *sequence*, check the Multi-key option. Now enter the key combination or sequence to be used for this action. Click OK when you are done.

Note

The action associated with a multi-key shortcut is performed as soon as only one action matches the keys pressed so far. For example, if a multi-key shortcut of A B C is defined, and no other multi-key shortcuts are defined, the action associated with the shortcut will be performed as soon as **A** is pressed.

6.2 Global Settings

Kst's global settings can be accessed by selecting the Settings → Configure Kst.... The following dialog will be displayed.



Plot update timer: 200ms

Base font size: 12 Minimum font size: 5

Default plot colors:

Curve color sequence: Kst Colors 🟥🟨🟩🟦🟦🟪

Default Plot Grid Lines

Show major: □ X □ Y Color: ☑ Default

Show minor: □ X □ Y Color: ☑ Default

Default X Axis

Default Interpretation: Standard C time Display as: <Qt Local Date> HH:MM:SS.SS

□ Interpret by default

Default line weight: 0

Offset from UTC: 0.00 hours Timezone: UTC

□ Prompt before deleting plots

☑ Prompt before closing windows

☑ Show QuickStart dialog on startup

☑ Apply tied zoom globally

There are three available tabs. General, Data Sources, and E-Mail.

6.2.1 General

The "General tab" contains most of Kst's global settings.

Plot update timer: The period of Kst's updates. These will check, for example, for changes in the data files associated with curves and update the plots accordingly.

Base font size: The base font size for newly created plots.

Minimum font size: The smallest font size which will be allowed in newly created plots (even if they are squeezed for space).

Default plot colors: The default foreground and background colours for plots.

Curve color sequence: The colours to cycle through as new curves are added to a plot. This should have high contrast so that the curves can be differentiated easily.

Default Plot Grid Lines See the Colour and Line Weights section for Plots.

Default X Axis See the X/Y Axis Scale section for Plots.

Default Line Weight See the Colour and Line Weights section for Plots.

Offset from UTC: / Timezone: This allows Kst to calculate the local time for data given in UTC.

Prompt before closing windows Kst will require confirmation before closing a window (and deleting the plots which it contains!).

Show QuickStart dialog on startup Enable if you would like the QuickStart Dialog (see Getting Started - Importing Data for a screenshot) to appear every time Kst starts up.

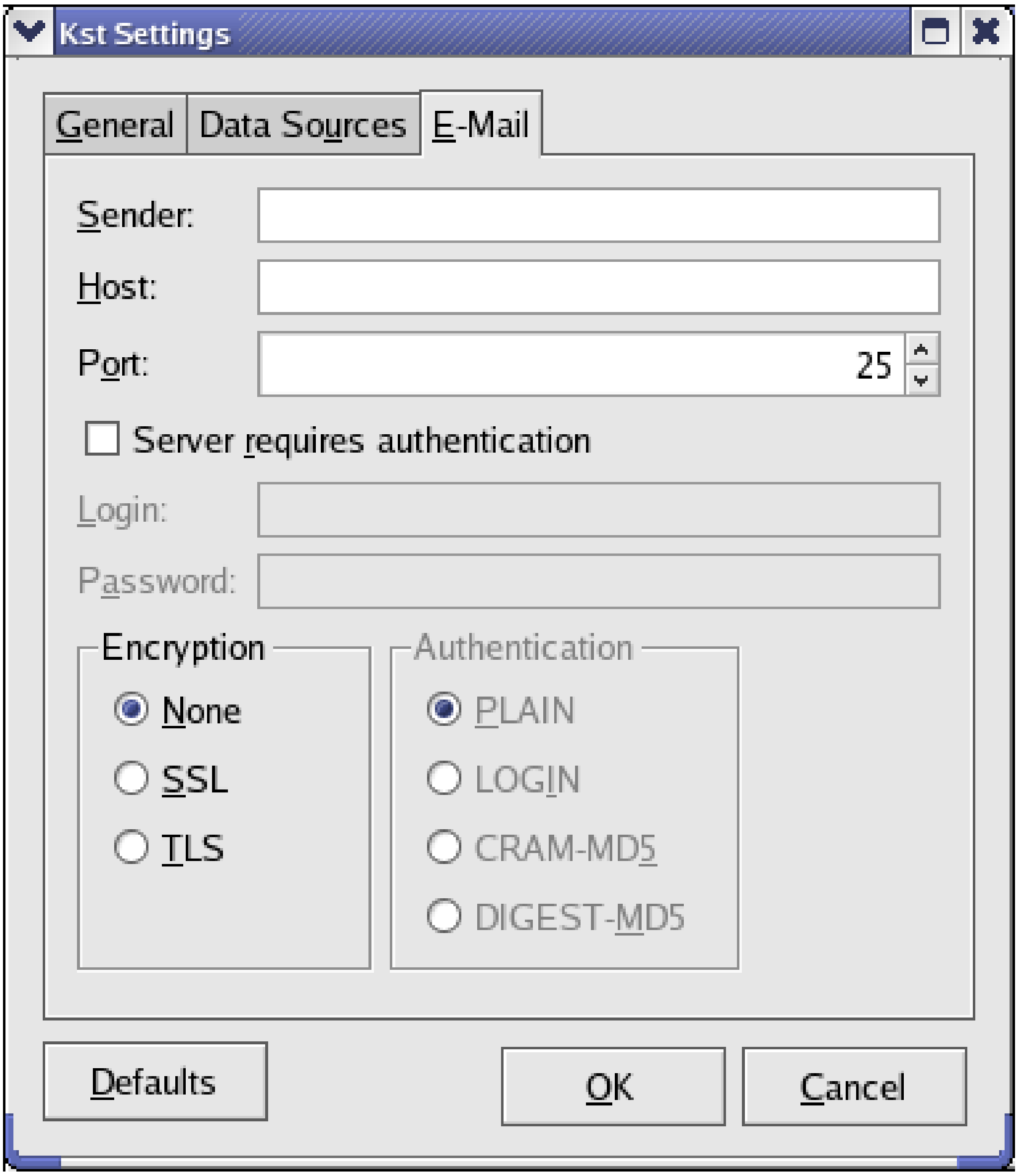
Apply tied zoom globally If this setting is enabled, tied zoom is applied to all windows, rather than just the current one.

6.2.2 Data Sources

This pane allows you to choose the default options for Kst's data sources. An explanation of the options available for the set of data sources included with Kst can be found in the Data Sources documentation.

6.2.3 E-Mail

Kst can send e-mails for debugging and logging purposes. In this tab, you give Kst the settings for the SMTP server which you would like to use to send mail.



Sender:

Host:

Port: 25

☐ Server requires authentication

Login:

Password:

Encryption

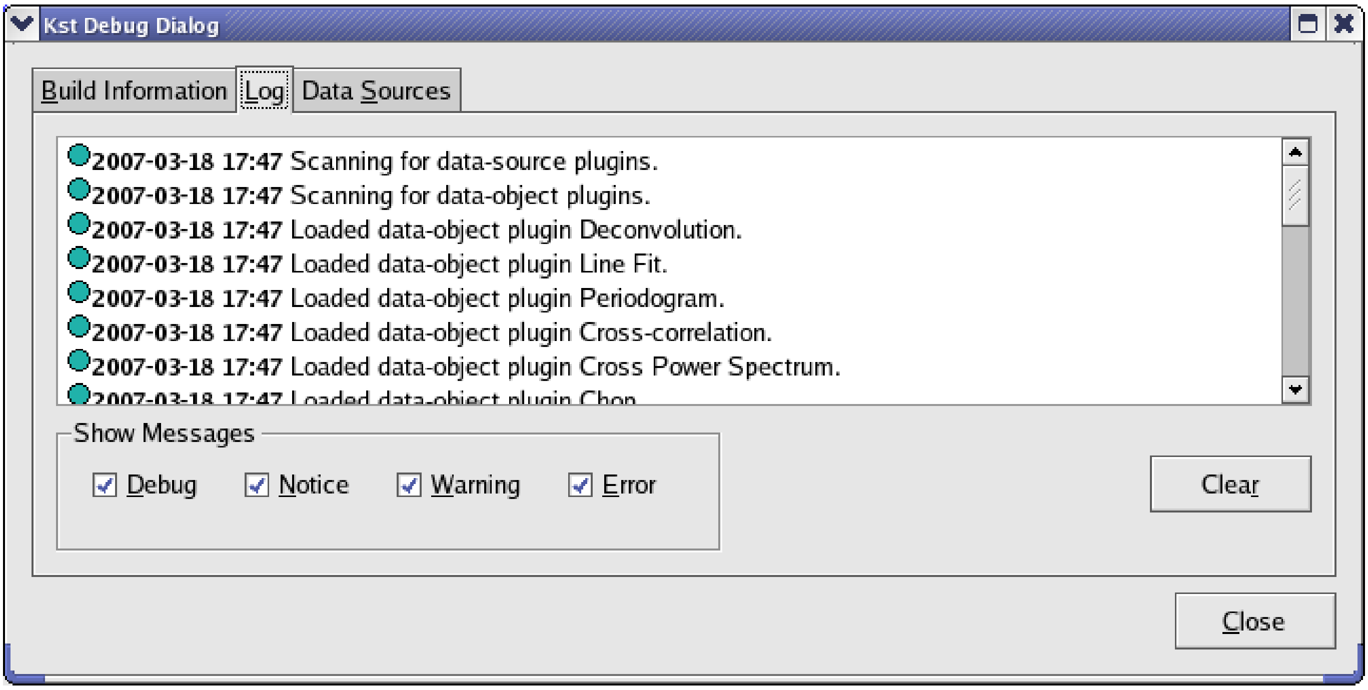
● None
○ SSL
○ TLS

Authentication

● PLAIN
○ LOGIN
○ CRAM-MD5
○ DIGEST-MD5

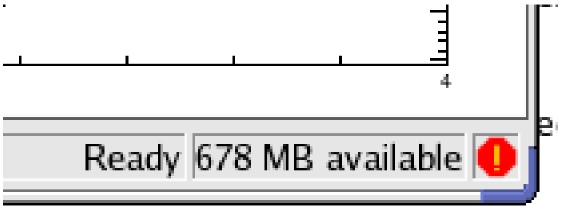
6.3 The Debug Dialog

The Debug Dialog contains the Kst message log, and information about loaded data sources. To access the debug dialog, select Help → Debug Kst...



```
2007-03-18 17:47 Scanning for data-source plugins.
2007-03-18 17:47 Scanning for data-object plugins.
2007-03-18 17:47 Loaded data-object plugin Deconvolution.
2007-03-18 17:47 Loaded data-object plugin Line Fit.
2007-03-18 17:47 Loaded data-object plugin Periodogram.
2007-03-18 17:47 Loaded data-object plugin Cross-correlation.
2007-03-18 17:47 Loaded data-object plugin Cross Power Spectrum.
2007-03-18 17:47 Loaded data-object plugin Chop
```

The Build Information tab contains the Kst version number. The Data Sources tab contains a list of the data sources loaded in Kst and the files which they are being used to read. The Log tab contains a list of messages which Kst has recorded, and can be useful for debugging. If an error message has been posted, but not read, then Kst will notify you by placing an excited stop sign in the lower right corner of the main window.



The following lists the different messages that can appear in the debug log, with brief explanations and suggested actions (where appropriate).

Type	Message	Explanation/Suggested Action
Unknown	Kst Extension %1 loaded	None.
Unknown	Error trying to load Kst extension %1. Code=%2, \"%3\"	Make sure extension exists, meets the requirements for a Kst extension, and is in the correct location.
Unknown	Unable to find required vector [%1] for data object %2.	Ensure the indicated vector exists in the Data Manager. If so, try recreating the data object.
Unknown	Unable to find required scalar [%1] for data object %2.	Ensure that the indicated scalar exists (try searching in View Scalars of the Data menu.
Unknown	Error loading data-source plugin [%1]: %2	A datasource reader (KDE plugin) could not be loaded. Make sure the data-source reader meets requirements described in Creating Datasource Readers and is compiled correctly. If it is a built-in datasource reader, try recompiling and installing it again from the source package.
Unknown	Scanning for data-source plugins.	Kst is scanning for datasource readers. No action required.

Type	Message	Explanation/Suggested Action
Unknown	No data in file %1	Make sure file is supported by one of the datasource readers, and that the data is formatted correctly and is not corrupt.
Unknown	Unsupported element '%1' in file %2	Ensure that the file is not corrupt, and meets the specifications for the file format.
Unknown	Unable to find X vector for %2: [%1]	???
Unknown	Unable to find Y vector for %2: [%1]	???
Unknown	Undefined justification %1 in label \"%2\"	This should not happen. Most likely it is the result of a bug.
Unknown	Internal error: No X scale type checked in %1.	None.
Unknown	Internal error: No Y scale type checked in %1.	None.
Unknown	Unable to load plugin %1 for \"%2\"	Make sure that the plugin meets the requirements described in Creating Additional Plugins. In particular, make sure the shared object and `*.xml` file are in the same directory. If the plugin is a built-in plugin, try re-installing the plugin from the source package.
Unknown	Input vector [%1] for plugin %2 not found. Unable to continue.	Ensure that the `*.xml` file for the plugin specifies all the inputs correctly and in the correct order. Check the source code for the shared object to ensure that it is using the correct arrays from the input array. Also ensure that the specified input vector still exists (check in the Data Manager).
Unknown	Output vector [%1] for plugin %2 not found. Unable to continue.	Ensure that the `*.xml` file for the plugin specifies all the outputs correctly and in the correct order. Check the source code for the shared object to ensure that it is outputting the correct arrays in the output array.
Unknown	Data file for vector %1 was not opened.	???
Unknown	Error parsing XML file '%1'; skipping.	Check the XML file to make sure it has correct syntax and form for the intended use. If is an XML file for a plugin, check Creating Additional Plugins for more information.
Unknown	Detected disappearance of '%1'.	None.
Unknown	Could not parse xml file '%1'.	Check the XML file to make sure it has correct syntax and form for the intended use. If is an XML file for a plugin, check Creating Additional Plugins for more information.
Unknown	Could not find library '%1' while loading plugin.	Make sure the library exists and is in the correct location. If this is a built-in plugin, try recompiling and installing the plugin again.

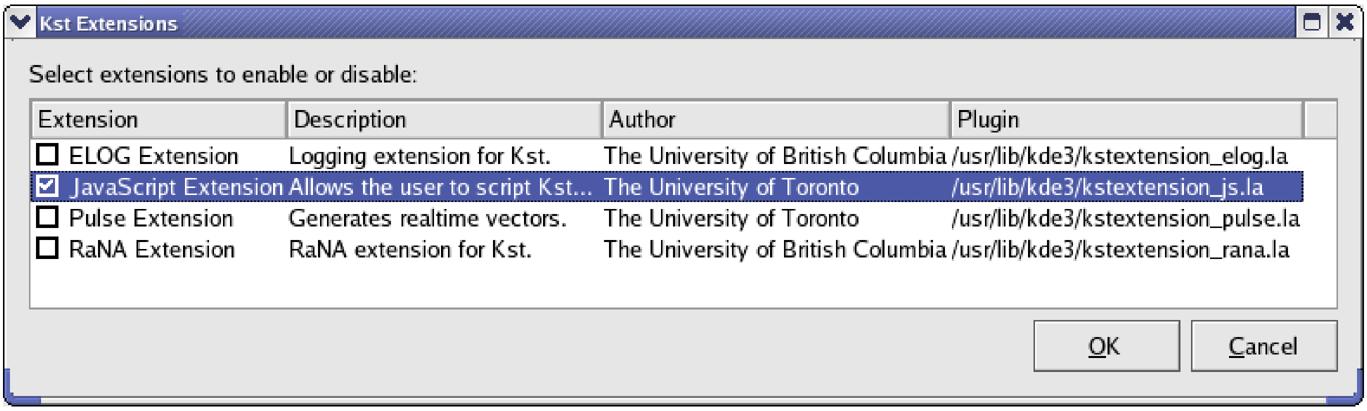
Type	Message	Explanation/Suggested Action
Unknown	Could not find symbol '%1' in plugin %2.	A required C symbol could not be found in the plugin. Make sure the plugin meets the requirements described in Creating Additional Plugins. If this is a built-in plugin, try recompiling and re-installing the plugin.
Unknown	The %1 font was not found and was replaced by the %2 font; as a result, some labels may not display correctly."	If this causes problems, install the missing font.

Chapter 7

Kst Extensions

Kst extensions are a powerful way to add functionality to Kst. They can, for example, add elements to the Kst user interface. They are usually designed to accomplish specific tasks which aren't of general use to the Kst community.

To load or unload extensions in Kst, select Settings → Extensions.... A dialog will be presented which allows you to load or unload installed plugins.



Select extensions to enable or disable:

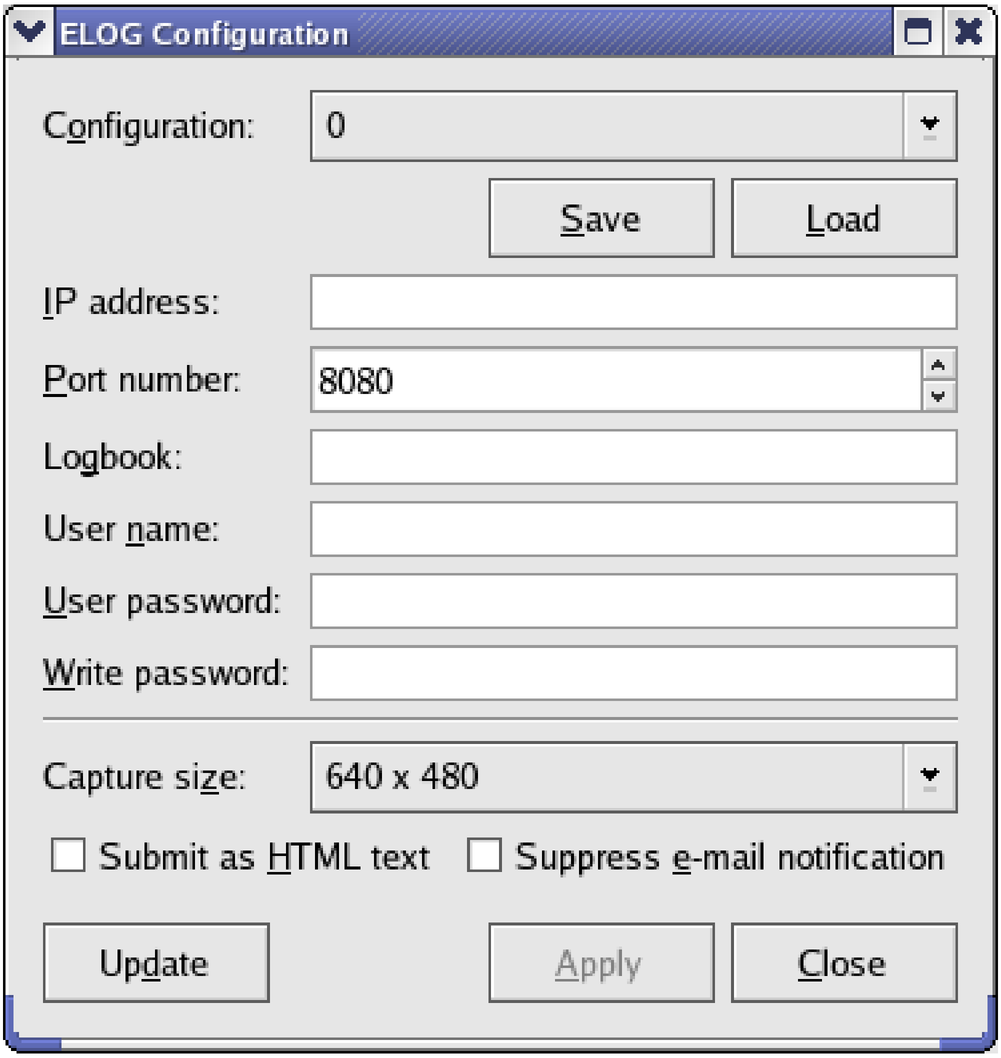
Extension	Description	Author	Plugin
☐ ELOG Extension	Logging extension for Kst.	The University of British Columbia	/usr/lib/kde3/kstextension_elog.la
☑ JavaScript Extension	Allows the user to script Kst...	The University of Toronto	/usr/lib/kde3/kstextension_js.la
☐ Pulse Extension	Generates realtime vectors.	The University of Toronto	/usr/lib/kde3/kstextension_pulse.la
☐ RaNA Extension	RaNA extension for Kst.	The University of British Columbia	/usr/lib/kde3/kstextension_rana.la

7.1 The ELOG Extension

The ELOG extension allows event monitors to create logbook entries on a server running the Electronic Logbook (ELOG) application. The created logbook entries can be used, for example, to remotely monitor Kst events or to produce periodic data reports. For information on configuring and maintaining an ELOG server, please see the official ELOG Homepage. To enable the ELOG extension in Kst, select Settings → Extensions... and ensure that ELOG Extension is checked.

7.1.1 Server and Logging Settings

Server and logging settings should be configured before an entry is added to a logbook. To configure logging settings, select Settings → ELOG... or click the Configure... button in the ELOG Entry dialog accessed from either File → Add ELOG Entry... or the Event Monitor dialog if ELOG functionality is enabled for an event monitor object. The following dialog should be displayed:



Configuration: 0

IP address:

Port number: 8080

Logbook:

User name:

User password:

Write password:

Capture size: 640 x 480

☐ Submit as HTML text ☐ Suppress e-mail notification

Descriptions of the settings in this dialog follow.

Configuration, Save, and Load The Configuration drop down list contains a list of saved ELOG server configurations. To load a set of configuration settings, select an item from the list and click the Load button. To save the current configuration settings in this dialog into a slot, select the desired slot and click the Save button.

> **Caution**
> Saving settings into a slot overwrites any existing settings in that slot.

IP address and Port number Enter the IP address of the machine running the ELOG server application. The port number that the ELOG server listens on is configurable through the server, but by default it is 8080, so if you are unsure, enter 8080 as the port number.

Logbook Each ELOG server can contain one or more logbooks, so the name of the logbook to write entries to must be specified. Ensure that the spelling of the logbook name is correct.

User name, User password, and Write password Depending on the how the ELOG server application is configured, these settings may or may not be necessary. If they are, enter the correct login credentials in the appropriate textboxes.

Capture size Use this list box to select the size, in pixels, of the Kst screen capture to attach (if specified).

Submit as HTML text If this checkbox is selected, HTML tags may be used in the logbook entry.

Suppress e-mail notification If the logbook is configured to send e-mail notifications of new entries, selecting this checkbox will disable the notifications. Note that this option is not related to Kst's own e-mail notification setting for event monitors.

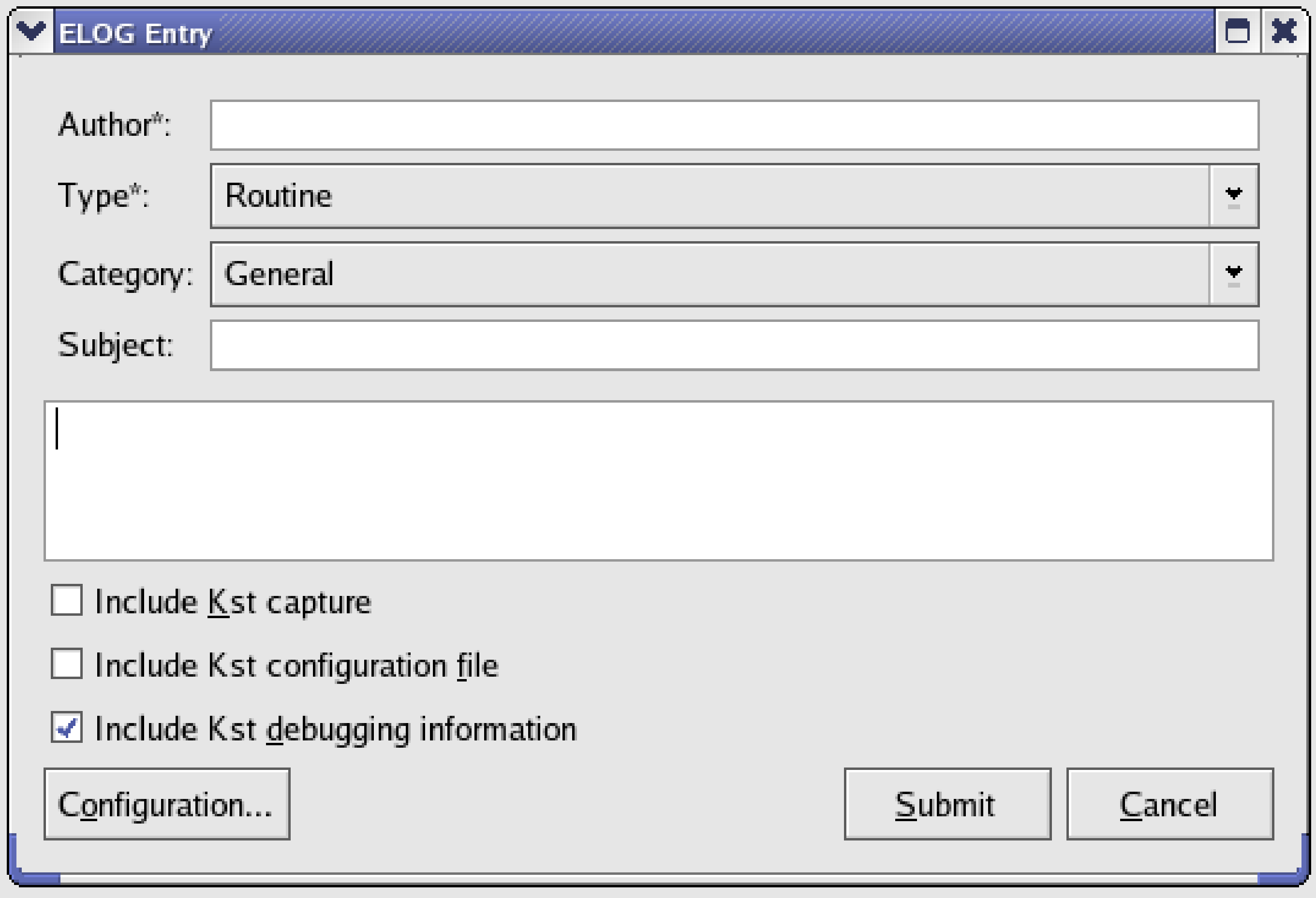
Apply Click the Apply button to apply the entered settings.

Update Click Update to update the ELOG Event Entry dialog.

Close Click Close to exit the dialog.

7.1.2 Add an ELOG Entry

To add an entry to the logbook, choose File → Add ELOG Entry... This brings a dialog allowing the user to provide information for attributes as well as entering the contents for the log message. The types of attributes depend on the logbook you specified. The screenshot below shows one example of possible required attributes.



ELOG Entry

Author*:

Type*: Routine

Category: General

Subject:

☐ Include Kst capture

☐ Include Kst configuration file

☑ Include Kst debugging information

Logged Information Use the three checkboxes at the bottom to select the type of information to be logged.

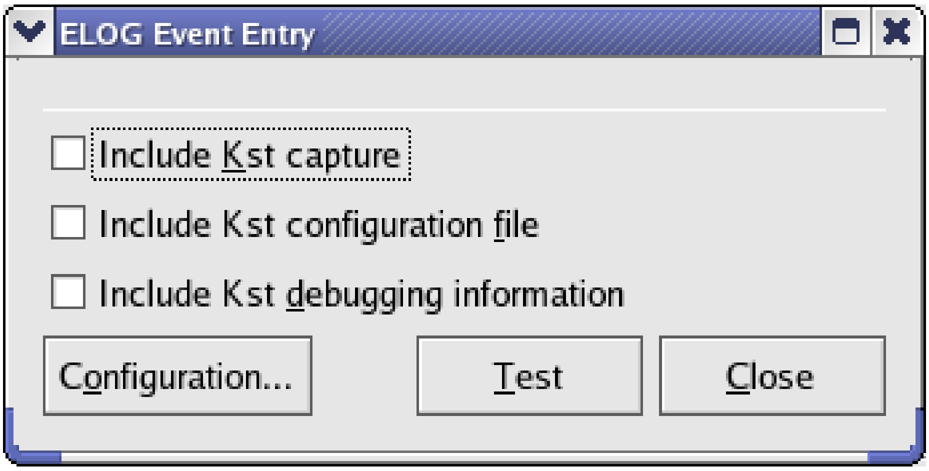
- Include Kst capture specifies that a screen capture of the currently active window should be included in the entry.
- Include Kst configuration file specifies that `*.kst` save file should be attached to the entry.
- Include Kst debugging information is checked, a copy of the debug log will be attached to the entry as well.

Configuration... Clicking Configuration... brings the ELOG Configuration dialog. See the Server and Logging Settings section for details.

Submit Click Submit to submit the log entry.

Cancel Click Cancel to close the dialog without posting the log entry.

When the ELOG functionality is enabled for an Event Monitor object, clicking the Configure... button in the New Event Monitor dialog will bring the ELOG Event Entry dialog. In this dialog, the user can enter information for logbook attributes; again, the types of attributes depend on the logbook configured in the ELOG configuration dialog.



☐ Include Kst capture

☐ Include Kst configuration file

☐ Include Kst debugging information

Logged Information See explanation above.

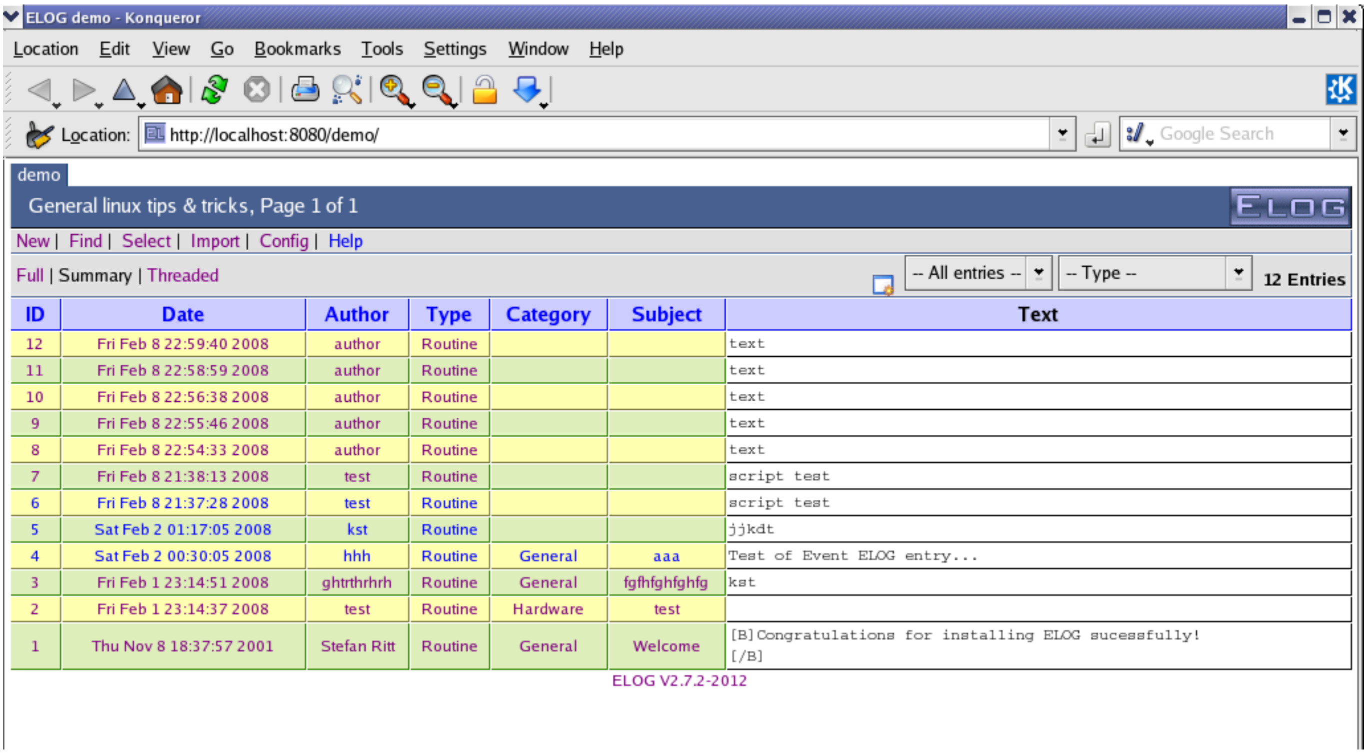
Configuration... See explanation above.

Test This button is used to test the ELOG configuration. If the user-specified logbook is configured correctly, clicking Test will post a test message:'Test of Event ELOG Entry' in the logbook.

Close Click Close to close the dialog.

7.1.3 ELOG Browser

To view existing logbooks through Kst, choose File → Launch ELOG Browser....


ELOG demo - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Google Search
demo
ELOG
New | Find | Select | Import | Config | Help
Full | Summary | Threaded
-- All entries --
-- Type --

Location: http://localhost:8080/demo/

General linux tips & tricks, Page 1 of 1

12 Entries

ID	Date	Author	Type	Category	Subject	Text
12	Fri Feb 8 22:59:40 2008	author	Routine			text
11	Fri Feb 8 22:58:59 2008	author	Routine			text
10	Fri Feb 8 22:56:38 2008	author	Routine			text
9	Fri Feb 8 22:55:46 2008	author	Routine			text
8	Fri Feb 8 22:54:33 2008	author	Routine			text
7	Fri Feb 8 21:38:13 2008	test	Routine			script test
6	Fri Feb 8 21:37:28 2008	test	Routine			script test
5	Sat Feb 2 01:17:05 2008	kst	Routine			jjkdt
4	Sat Feb 2 00:30:05 2008	hhh	Routine	General	aaa	Test of Event ELOG entry...
3	Fri Feb 1 23:14:51 2008	ghtrthrhrh	Routine	General	fgfhfghfghfg	kst
2	Fri Feb 1 23:14:37 2008	test	Routine	Hardware	test	
1	Thu Nov 8 18:37:57 2001	Stefan Ritt	Routine	General	Welcome	[B]Congratulations for installing ELOG sucessfully! [/B]

ELOG V2.7.2-2012

Tip

To quickly view the ELOG browser and add an entry to a logbook, just click the in the toolbar to launch the ELOG browser and to add an ELOG entry.

7.1.4 Using the ELOG Class in KstScript

Alternatively, you can use the ELOG class provided in KstScript to enter and submit messages in a logbook and change the server and logging settings. See details in the ELOG class documentation.

Example 7.1 Adding an ELOG entry through KstScript

```
elog = new ELOG();       //construct a new ELOG object
elog.port = 8080;        //specify the port
elog.hostname = "localhost";    //specify the hostname;
elog.logbook = "demo";      // specify the logbook;
elog.text = "message contents";   //enter the message;
elog.addAttribute("Author", "Kst");
elog.addAttribute("Type", "Routine");

elog.submit();
```

7.2 KstScript

KstScript is a JavaScript binding for Kst. It aims to offer full control over all aspects of Kst. To enable KstScript, select Settings → Extensions... and ensure that JavaScript Extension is checked.

7.2.1 Running a KstScript

There are three ways to to run a KstScript.

- from the command line when starting Kst:

  ```
  kst -E js:"loadScript('myscript.js')"
  ```

- using kstcmd. kstcmd is a JavaScript interpreter which connects to your Kst session. It can be run from an external terminal, or from within Kst itself by selecting Tools → Show JavaScript Console.

  ```
  kstcmd Attached to Kst session kst-17704
  kst> loadScript('myscript.js')
  ```

- from the menu once Kst is loaded: File → LoadJavaScript

Note that with the first two methods, any KstScript command, or collection of KstScript commands can be substituted for the loadscript command.

7.2.2 KstScript Examples

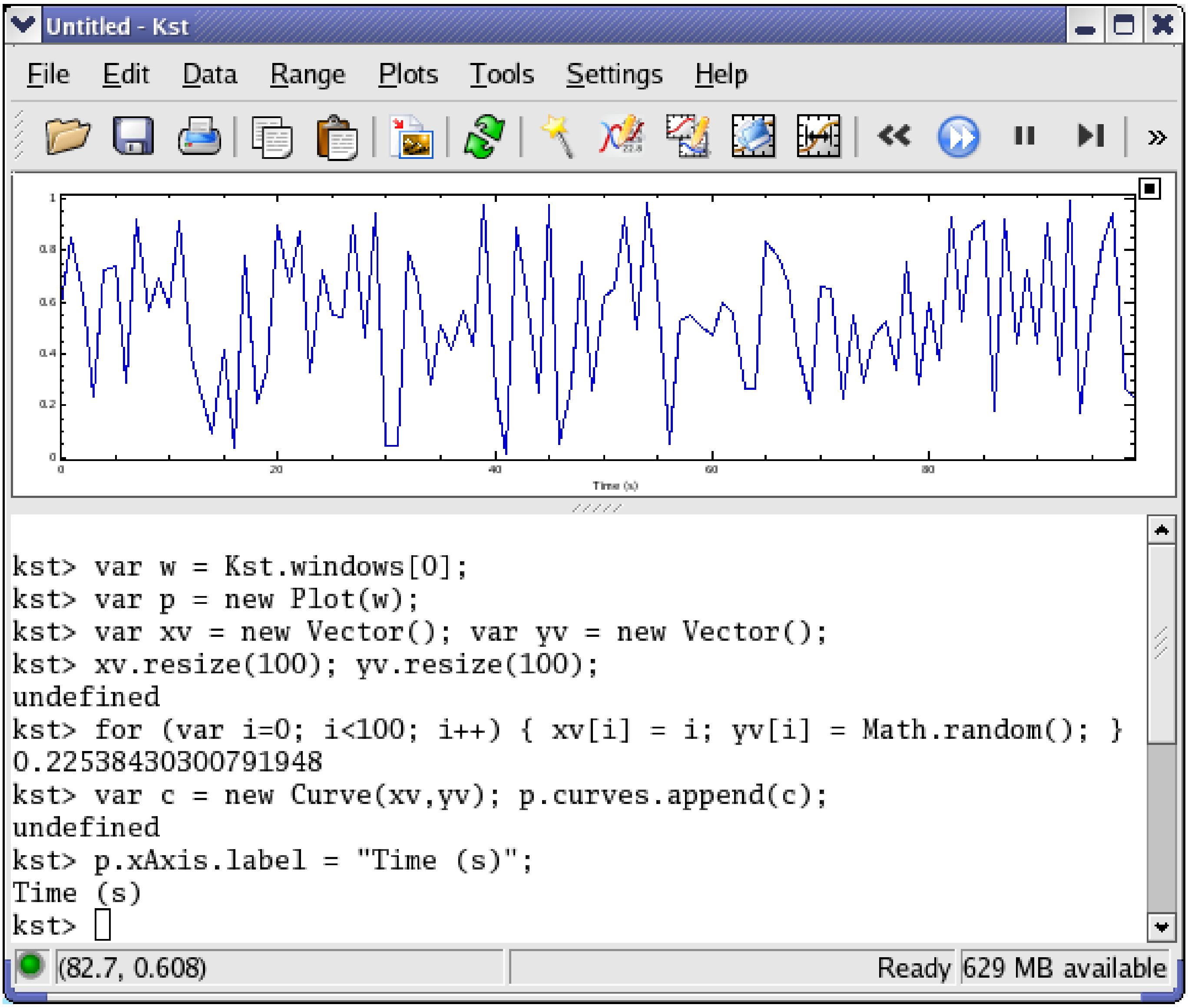
KstScript gives Kst a powerful programming interface. Chiefly, it can be used to construct 'smart' macro functions for common tasks. We will first look at a basic example illustrating how to control Kst through KstScript, and then we will look at an example of a macro function.

7.2.2.1 A simple KstScript

Here we have the ‘hello world’ of KstScripts: plot a curve of random numbers.

```
// simple.js: A simple KstScript demo.
// Generates a new plot with a curve of random numbers.

// grab the window
var w = Kst.windows[0];

// add a plot to the window
var p = new Plot(w);

// create new (editable) vectors.
// these could also be loaded from files.
var xv = new Vector(); var yv = new Vector();

xv.tagName = "time"; yv.tagName = "signal";

// make their length 100.
xv.resize(100); yv.resize(100);

// fill in their values.
for (var i = 0; i < 100; i++) {
        xv[i] = i; //index
        yv[i] = Math.random(); //uniformly distributed between 0 and 1.
}

// create a curve from x and y
var c = new Curve(xv,yv);
c.tagName = "signalvstime";

// put the curve in our new plot
p.curves.append(c);

// label the x-axis of our plot
p.xAxis.label = "Time (s)"
```

This script can be run using any of the methods from Section 7.2.1. Here is a screenshot of Kst after entering the commands using the embedded console method.



```
kst> var w = Kst.windows[0];
kst> var p = new Plot(w);
kst> var xv = new Vector(); var yv = new Vector();
kst> xv.resize(100); yv.resize(100);
undefined
kst> for (var i=0; i<100; i++) { xv[i] = i; yv[i] = Math.random(); }
0.22538430300791948
kst> var c = new Curve(xv,yv); p.curves.append(c);
undefined
kst> p.xAxis.label = "Time (s)";
Time (s)
kst>
```

In the Kst UI, objects are not labelled by the variable names which we use in our KstScript, but are are referred to by their Tag Name (or "Unique Name"). If you do not manually set the tagName attribute of new objects, then when you look things up in the data manager you will find cryptic names like ‘anonymous vector 1’, or, ‘’. This is the reason why we received several ‘undefined’ messages in the screenshot above. If you ever plan to use the UI to work with objects you have created with a KstScript, you will want to set their tagName property. You can also use tag names within KstScript itself. For example, after setting the tagName of our new curve we could have called

```
p.curves.append("signalvstime");
```

7.2.2.2 Writing, Including, and Calling Functions

As with any JavaScript environment, you can write and call functions with KstScript. Additionally, KstScript possesses a means for a KstScript to load functions from disk. Consider the following example:

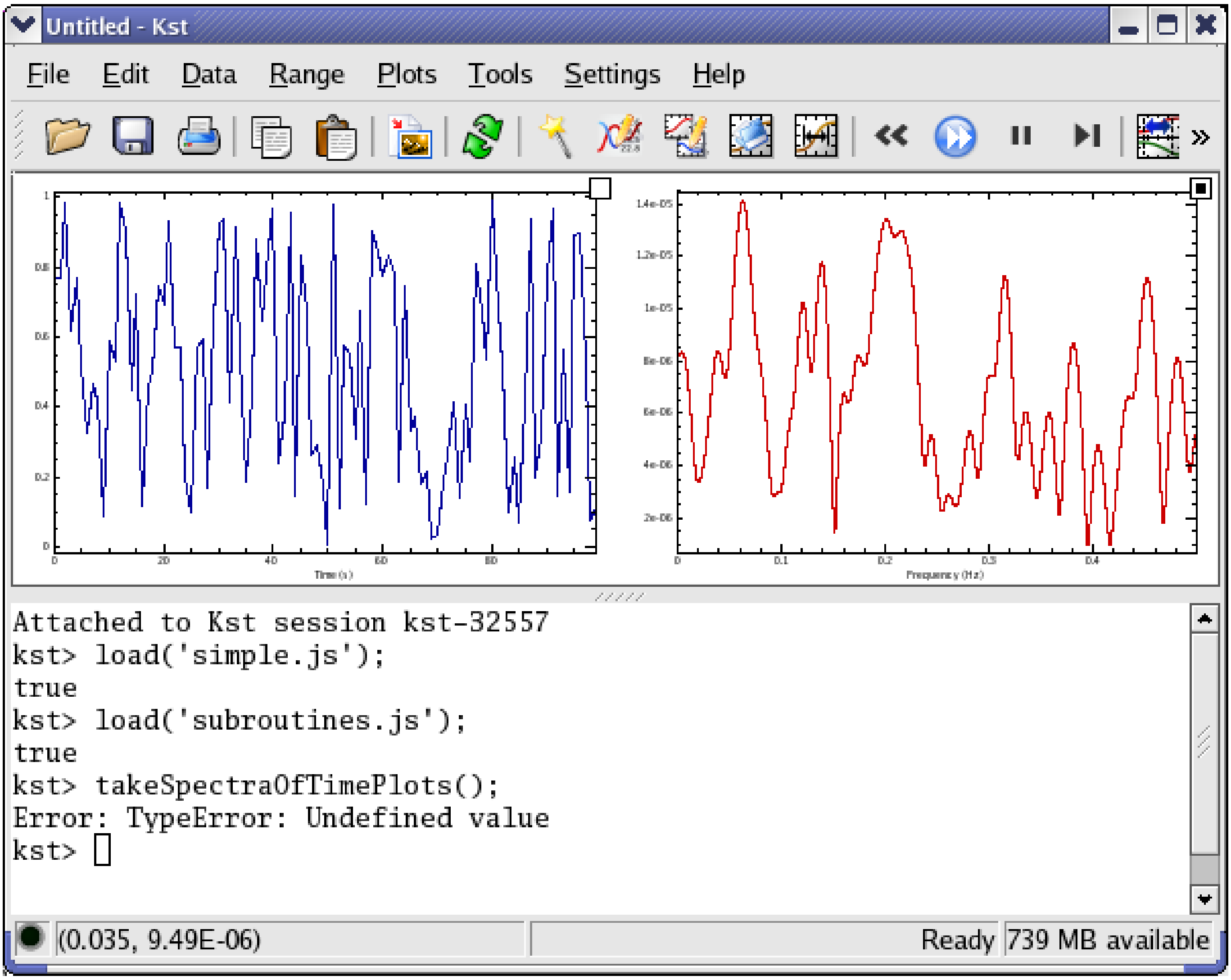
```
// subroutines.js

function takeSpectraOfTimePlots() {
//takes all of the plots in the first window which have an x-axis of "Time (s)"
// and generates power spectra for their curves, which it dumps into a single new
// plot.
        var w = Kst.windows[0];
        var p = new Plot(w);
        p.xAxis.label = "Frequency (Hz)"

        for (var ip = 0; ip < w.plots.length; ip++ ) {
                cp = w.plots[ip]; //current plot
                if (cp.xAxis.label == "Time (s)") {
                        for (ic = 0; ip < cp.curves.length; ic++) {
                                plotSpectrum(p,cp.curves[ic].yVector)
                        }
                }
        }
}

function plotSpectrum(p,v) {
//creates a power spectrum for the vector v and displays it in plot p. Assumes
//the vector is data sampled at 1Hz.
        ps = new PowerSpectrum(v, 1);
        ps.tagName = v.tagName + "-PS";
        c = new Curve(ps.xVector,ps.yVector);
        c.tagName = ps.tagName + "-C";
        p.curves.append(c);
}
```

This example contains two functions: **takeSpectraOfTimePlots()** and **plotSpectrum(p,v)**. takeSpectraOfTimePlots() is a macro function which will generate spectra for all of the plots in the first window which have "Time (s)" as their x-axis. In this example, we can see the power of KstScript for such tasks.

The following script loads these routines from the `subroutines.js`. It uses the `simple.js` of the previous section to generate a noise plot with a time axis, and then calls takeSpectraOfTimePlots() to find its spectrum. Notice that `subroutines.js` is loaded using **loadScript()**, the same call that we used from the command line in Section 7.2.1.

```
// callsubs.js
// demonstrate calling subroutines

loadScript('subroutines.js');
loadScript('simple.js');

takeSpectraOfTimePlots();
```

A screenshot of Kst after executing these commands is shown below.



```
Attached to Kst session kst-32557
kst> load('simple.js');
true
kst> load('subroutines.js');
true
kst> takeSpectraOfTimePlots();
Error: TypeError: Undefined value
kst> 
```

7.2.3 More KstScript Documentation

Complete documentation on the KstScript classes is given in Appendix D.

Chapter 8

Plugins, Fits, and Filters

8.1 Managing Plugins

Plugins provide added functionality to Kst. By default, Kst comes packaged with an extensive selection of built-in plugins. In addition, a simple plugin API allows for easy creation of third-party plugins.

Kst plugins are usually comprised of an object file and a .desktop file. For plugins which are available system-wide the object and .desktop files should be stored in `/usr/lib/kde3/` and `/usr/share/services/kst/` respectively (this may vary depending on how your system is configured).

8.2 Built-in Plugins

8.2.1 Overview

Kst plugins are powerful tools which can perform a lot of tasks, such as data fitting, interpolation, statistics, and signal processing including periodogram, correlations, convolution and filters, etc. There are over 25 built-in plugins available in Kst currently.

8.2.2 Using Built-in Plugins

All plugins can be created from Data → New Plugins... or from the Data Manager, and all data objects resulting from plugins can be seen in the Data Manager.

To quickly apply fit or filter plugins to a set of data, you can right click on the data plot under any mouse modes, and choose the Fit or Filter submenu in the context menu.

The following sections describe the purpose, key algorithms or formulae used to perform calculations, and inputs and outputs for each plugin.

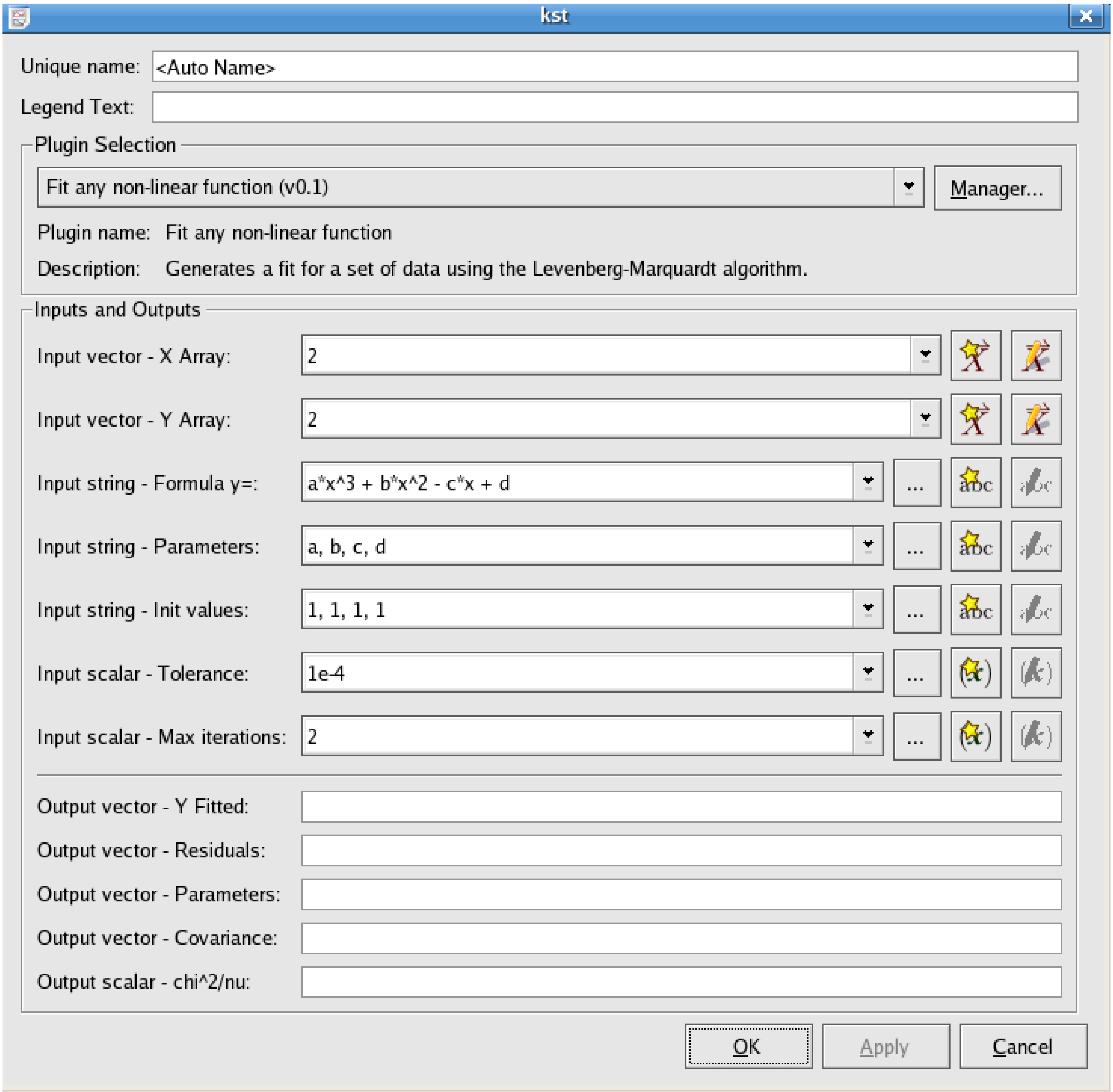
8.3 Fit Plugins

Kst provides lots of algorithms to perform data fitting. All fitting plugins output a vector of best fitted parameters, covariances, and Chi^2/Nu. These values can be seen by selecting the Data → View Fit Results

Note: The weight is not the same as the error. It is given as 1/yError^2.

8.3.1 Fit any non-linear function

This plugin is used to fit a set of data using the Levenberg-Marquardt algorithm.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit any non-linear function (v0.1)

Plugin name: Fit any non-linear function

Description: Generates a fit for a set of data using the Levenberg-Marquardt algorithm.

Inputs and Outputs

Input vector - X Array: 2

Input vector - Y Array: 2

Input string - Formula y=: a*x^3 + b*x^2 - c*x + d

Input string - Parameters: a, b, c, d

Input string - Init values: 1, 1, 1, 1

Input scalar - Tolerance: 1e-4

Input scalar - Max iterations: 2

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.1.1 Inputs

X Array (vector) The x array of the input data set

Y Array (vector) The y array of the input data set

Formula y= (string) The formula of the curve used to fit the data set

The default string is `a*x^3 + b*x^2 - c*x +d`

Parameters (string) Use a string to specify the parameter representation letters in the fitting formula. The plugin will later output the fitted values for these parameters.

The default string is `a, b, c, d` corresponding to the default formula string.

Init values (string) Use a string to specify the initial guess

The default string is `1, 1, 1, 1`

Tolerance (scalar) Tolerance is a number to indicate the precision of fitting; if the precision reaches the tolerance value, the curve fitting will stop.

The default tolerance is `1e-4`.

Max iterations (scalar) Specify the maximum number of iterations to indicate when the curve fitting should stop.

The default max iterations is `2`.

8.3.1.2 Outputs

Y Fitted (vector) The Y values of the fitted curve

Residuals (vector) The array of residuals

Parameters (vector) The best fitted values of the model parameters specified in the inputs.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

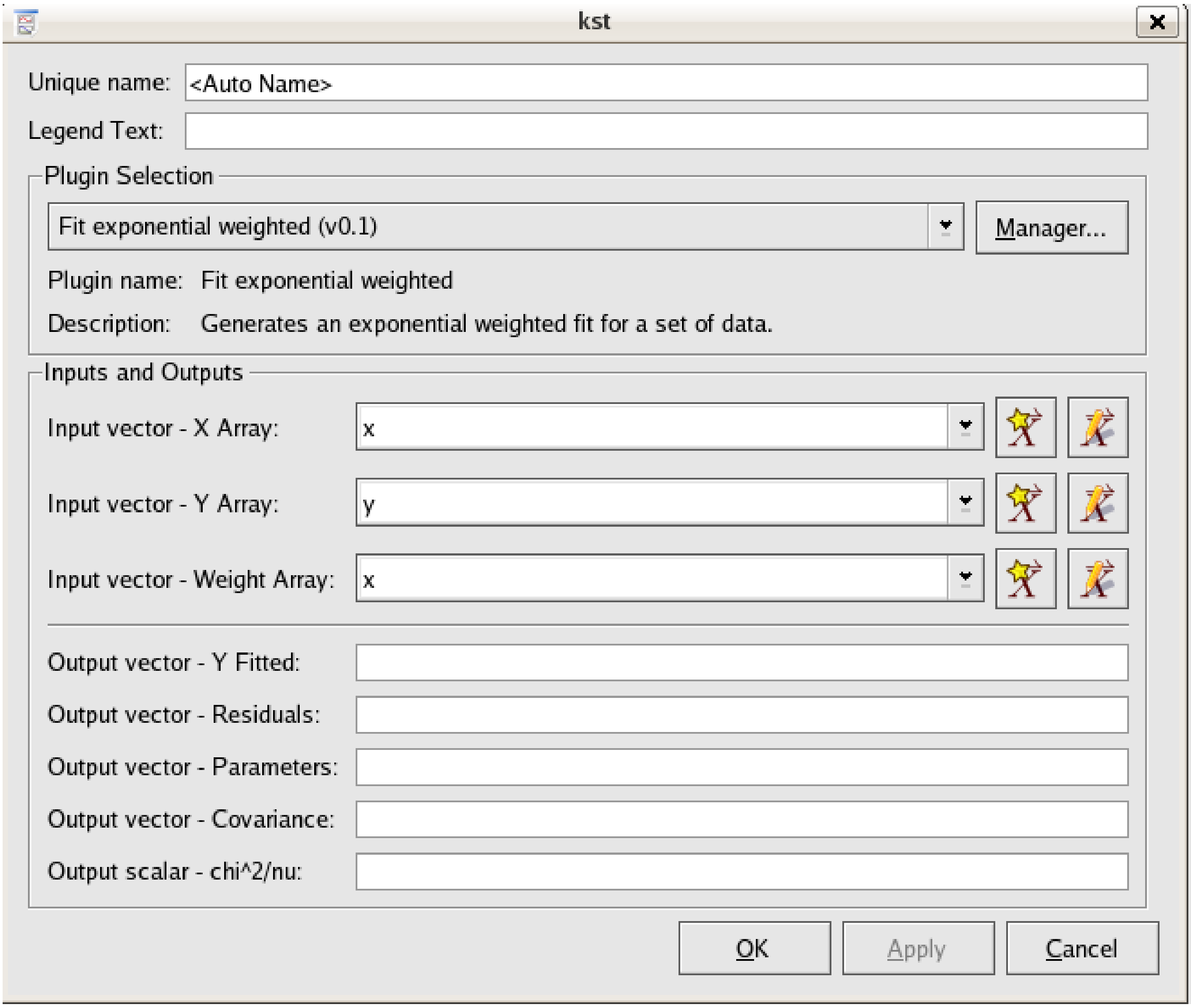
chi^2/nu (scalar) The sum of squares of residuals, divided by the degrees of freedom

8.3.2 Fit exponential weighted

The Fit exponential weighted plugin performs a weighted non-linear least-squares fit to an exponential model

$$y=ae^{-\lambda x}+b$$

An initial estimate of `a=1.0`, λ `=0`, and `b=0` is used. The plugin subsequently iterates to the solution until a precision of `1.0e-4` is reached or 500 iterations have been performed.



kst

Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit exponential weighted (v0.1)

Plugin name: Fit exponential weighted

Description: Generates an exponential weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weight Array: x

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.2.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array of weights to use for the fit.

8.3.2.2 Outputs

Y Fitted (vector) The array of fitted y values.

Residuals (vector) The array of residuals.

Parameters (vector) The best fit parameters a, λ , and b.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

chiˆ2/nu (scalar) The weighted sum of squares of the residuals, divided by the degrees of freedom.

8.3.3 Fit exponential

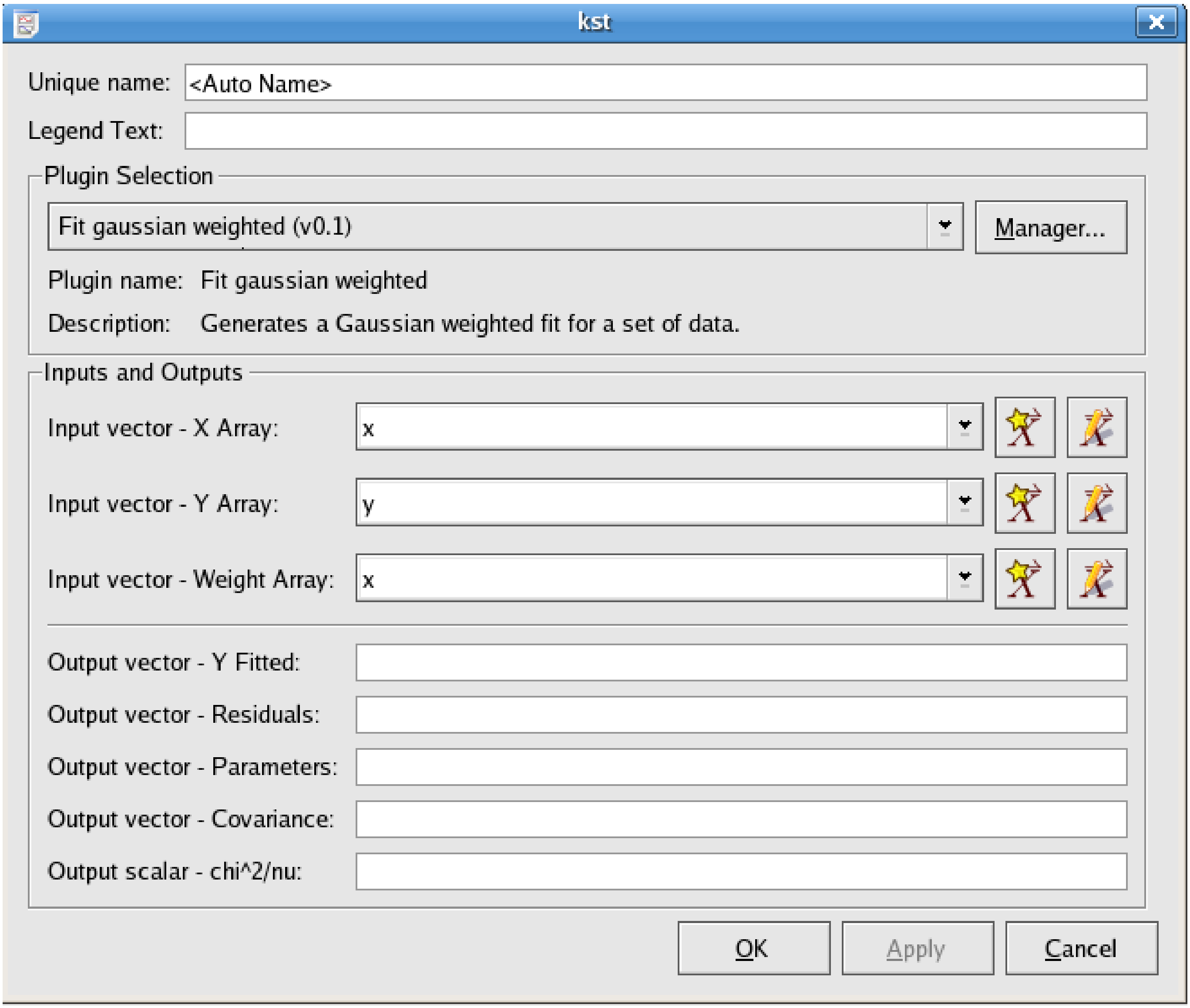
The Fit exponential plugin is identical in function to the Fit exponential weighted plugin with the exception that the weight value w_i is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.3.4 Fit Gaussian weighted

The Fit Gaussian weighted plugin performs a weighted non-linear least-squares fit to a Gaussian model:

$$y=\frac{a}{\sigma\sqrt{2\pi}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

An initial estimate of `a`=(maximum of the y values), μ =(mean of the x values), and σ =(the midpoint of the x values) is used. The plugin subsequently iterates to the solution until a precision of `1.0e-4` is reached or 500 iterations have been performed.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit gaussian weighted (v0.1)

Plugin name: Fit gaussian weighted

Description: Generates a Gaussian weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weight Array: x

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.4.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array of weights to use for the fit.

8.3.4.2 Outputs

Y Fitted (vector) The array of fitted y values.

Residuals (vector) The array of residuals.

Parameters (vector) The best fit parameters μ , σ , and `a`.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

chi^2/nu (scalar) The weighted sum of squares of the residuals, divided by the degrees of freedom.

8.3.5 Fit Gaussian

The Fit Gaussian plugin is identical in function to the Fit Gaussian weighted plugin with the exception that the weight value w_i is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.3.6 Fit gradient weighted

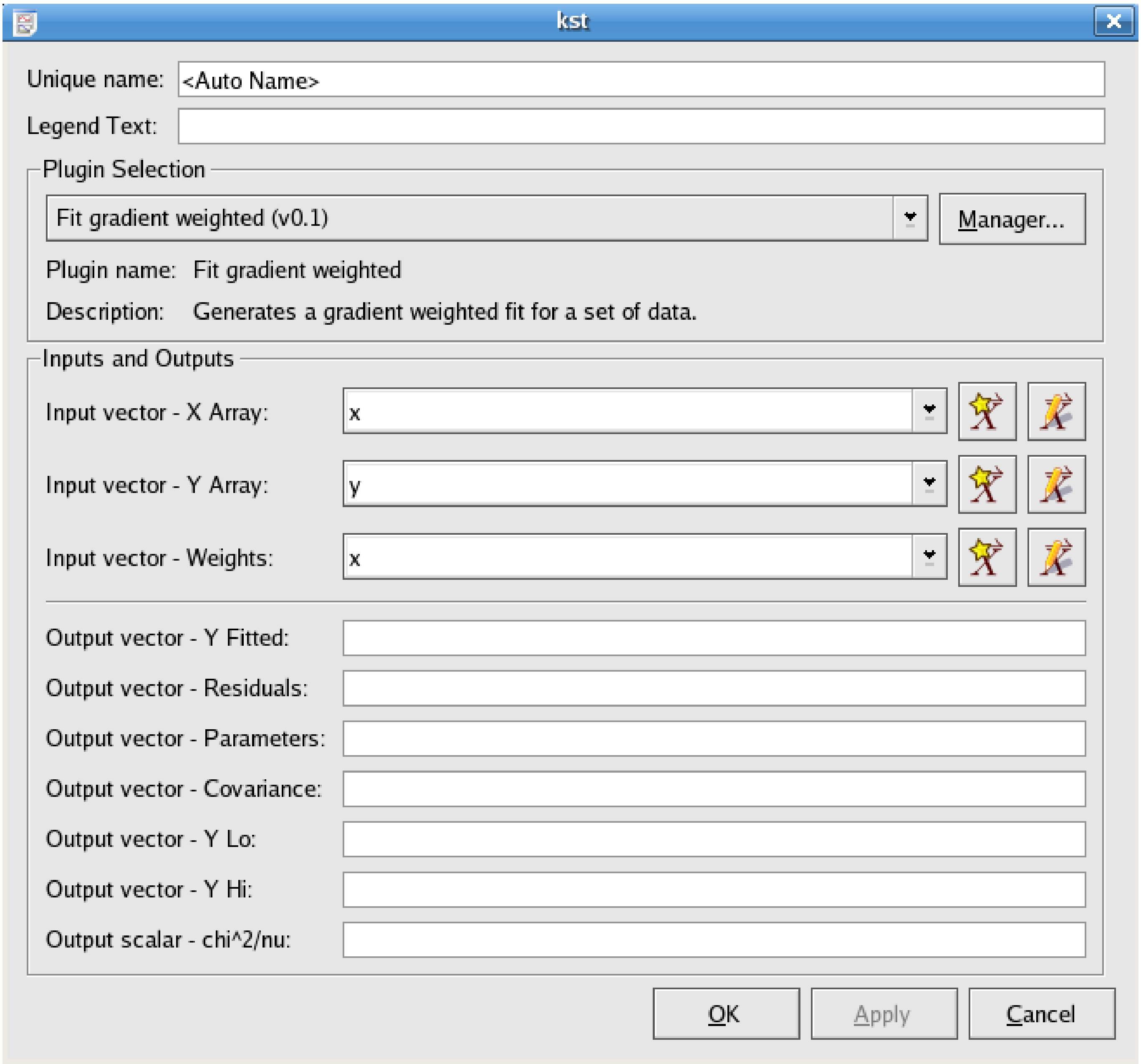
The gradient weighted plugin performs a weighted least-squares fit to a straight line model without a constant term:

$$Y = bX$$

The best-fit is found by minimizing the weighted sum of squared residuals:

$$\chi^2 = \sum_i w_i (y_i - bx_i)^2$$

for `b`, where w_i is the weight at index `i`.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit gradient weighted (v0.1)

Plugin name: Fit gradient weighted

Description: Generates a gradient weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weights: x

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output vector - Y Lo:

Output vector - Y Hi:

Output scalar - chi^2/nu:

8.3.6.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array containing weights to be used for the fit.

8.3.6.2 Outputs

Y Fitted (vector) The array of y values for the points representing the best-fit line.

Residuals (vector) The array of residuals, or differences between the y values of the best-fit line and the y values of the data points.

Parameters (vector) The parameter b of the best-fit.

Covariance (vector) The estimated covariance matrix, returned row after row, starting with row 0.

Y Lo (vector) The corresponding value in Y Fitted minus the standard deviation of the best-fit function at the corresponding x value.

Y Hi (vector) The corresponding value in Y Fitted plus the standard deviation of the best-fit function at the corresponding x value.

chiˆ2/nu (scalar) The value of the sum of squares of the residuals, divided by the degrees of freedom.

8.3.7 Fit gradient

The Fit linear plugin is identical in function to the Fit gradient weighted plugin with the exception that the weight value w_i is equal to 1 for all index values i. As a result, the Weights (vector) input does not exist.

8.3.8 Fit linear weighted

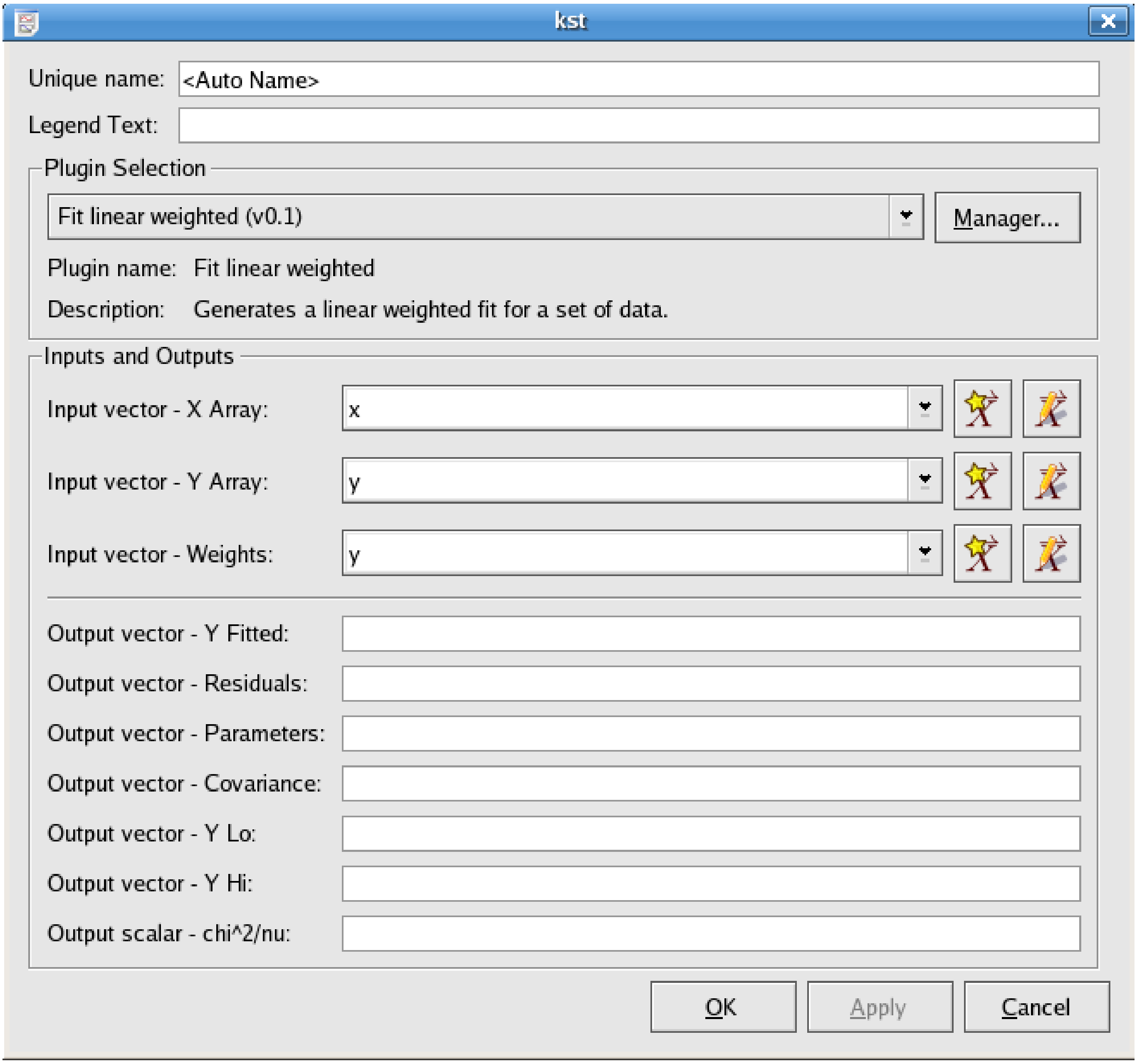
The Fit linear weighted plugin performs a weighted least-squares fit to a straight line model:

$$Y=a+bX$$

The best-fit is found by minimizing the weighted sum of squared residuals:

$$X^2=\sum_i w_i(y_i-(a+bx_i))^2$$

for a and b, where w_i is the weight at index i.



kst

Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit linear weighted (v0.1)

Plugin name: Fit linear weighted

Description: Generates a linear weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weights: y

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output vector - Y Lo:

Output vector - Y Hi:

Output scalar - chi^2/nu:

8.3.8.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array containing weights to be used for the fit.

8.3.8.2 Outputs

Y Fitted (vector) The array of y values for the points representing the best-fit line.

Residuals (vector) The array of residuals, or differences between the y values of the best-fit line and the y values of the data points.

Parameters (vector) The parameters `a` and `b` of the best-fit.

Covariance (vector) The estimated covariance matrix, returned row after row, starting with row 0.

Y Lo (vector) The corresponding value in Y Fitted minus the standard deviation of the best-fit function at the corresponding x value.

Y Hi (vector) The corresponding value in Y Fitted plus the standard deviation of the best-fit function at the corresponding x value.

chiˆ2/nu (scalar) The value of the sum of squares of the residuals, divided by the degrees of freedom.

8.3.9 Fit linear

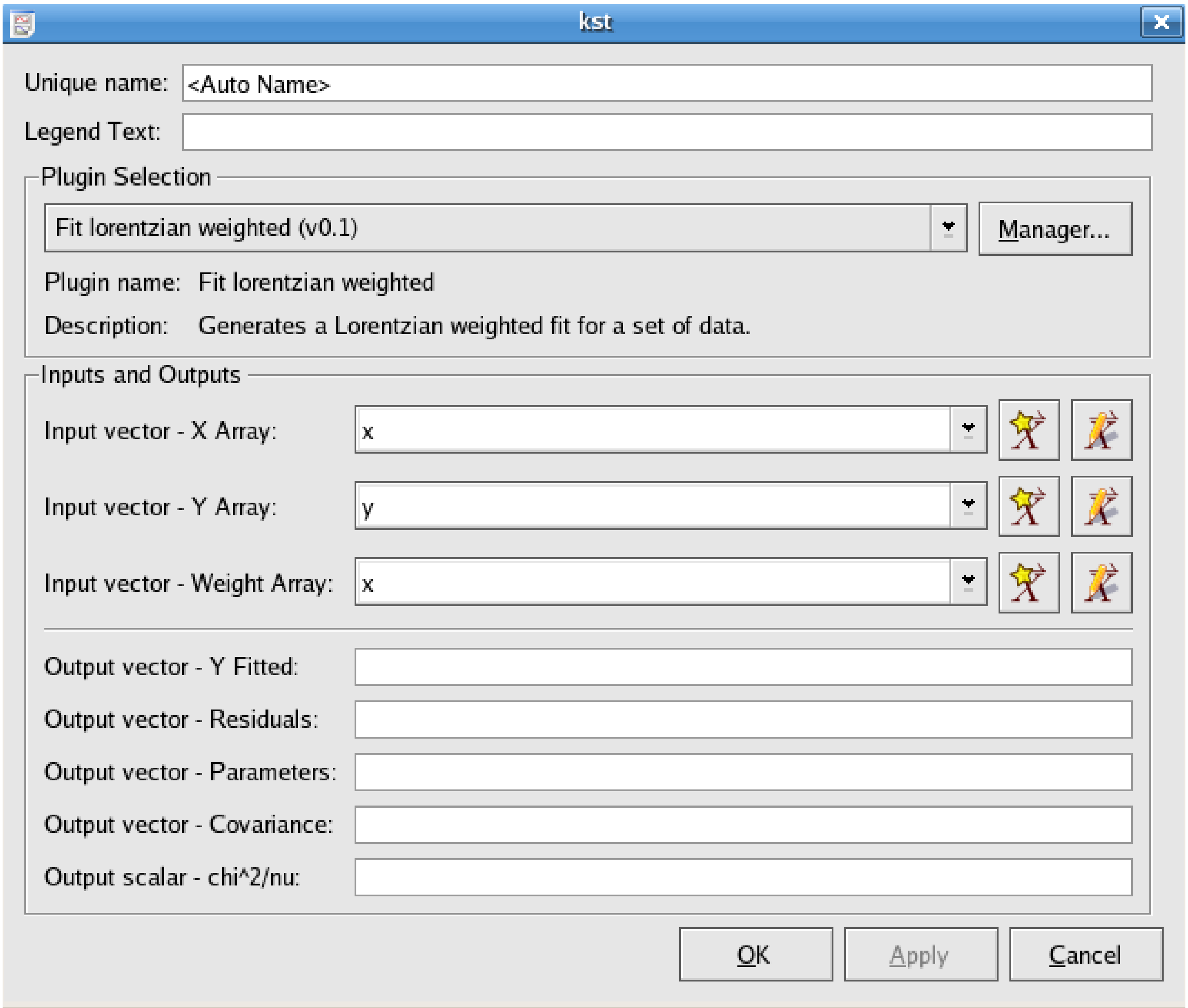
The Fit linear plugin is identical in function to the Fit linear weighted plugin with the exception that the weight value w_i is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.3.10 Fit Lorentz weighted

The Fit Lorentz weighted plugin performs a weighted non-linear least-squares fit to a Lorentzian model:

$$y = \frac{a}{\pi} \frac{\frac{1}{2}\Gamma}{(x - x_0)^2 + (\frac{1}{2}\Gamma)^2}$$

An initial estimate of `a`=(maximum of the y values), x_0=(mean of the x values), and Γ =(the midpoint of the x values) is used. The plugin subsequently iterates to the solution until a precision of `1.0e-4` is reached or 500 iterations have been performed.



Unique name: <Auto Name>

Legend Text:

Fit lorentzian weighted (v0.1)

Plugin name: Fit lorentzian weighted

Description: Generates a Lorentzian weighted fit for a set of data.

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weight Array: x

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.10.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array of weights to use for the fit.

8.3.10.2 Outputs

Y Fitted (vector) The array of fitted y values.

Residuals (vector) The array of residuals.

Parameters (vector) The best fit parameters x_0, Γ , and a.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

chiˆ2/nu (scalar) The weighted sum of squares of the residuals, divided by the degrees of freedom.

8.3.11 Fit Lorentz

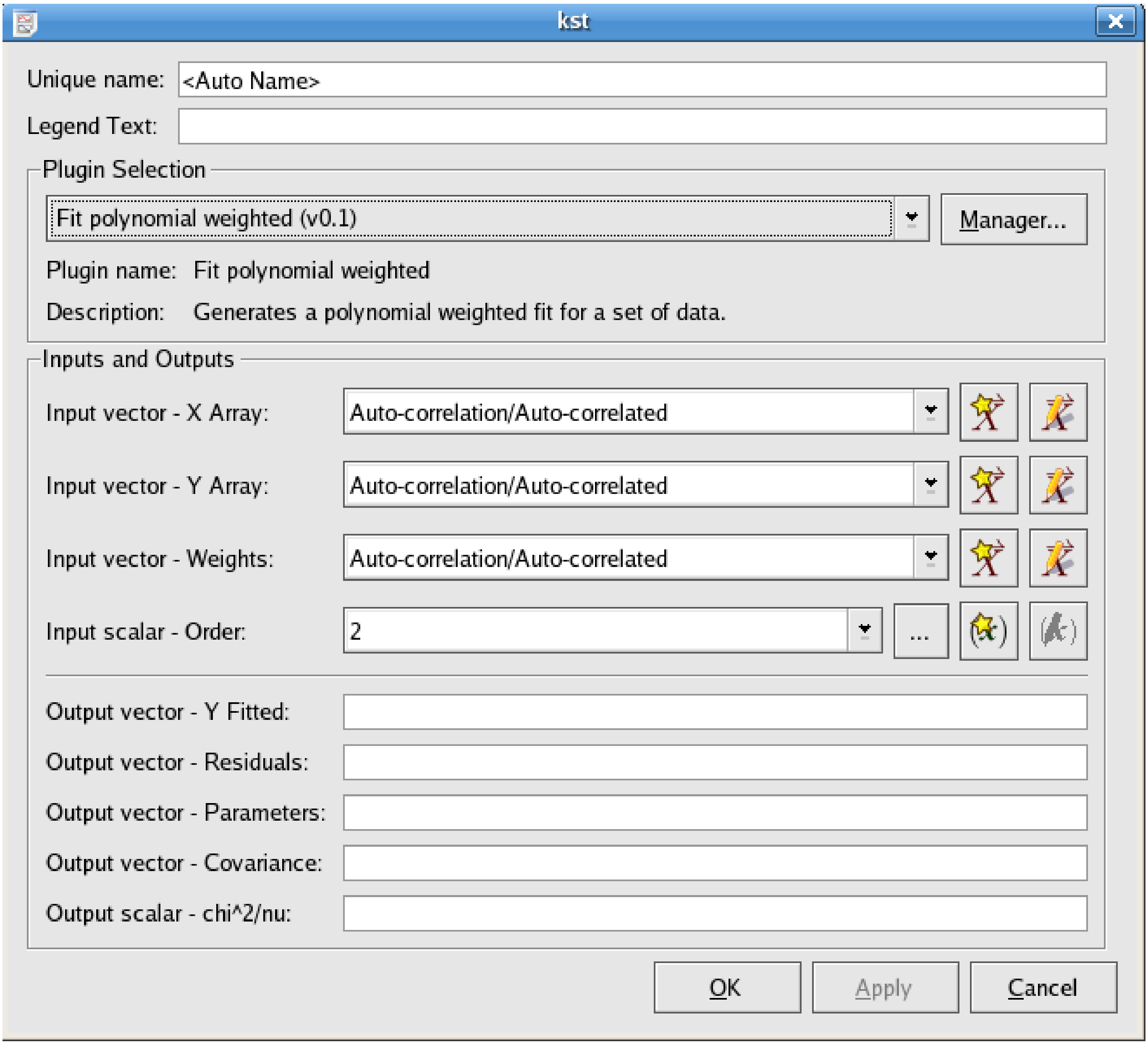
The Fit Lorentz plugin is identical in function to the Fit Lorentz weighted plugin with the exception that the weight value `w_i` is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.3.12 Fit polynomial weighted

The Fit polynomial weighted plugin performs a weighted least-squares fit to a polynomial model:

$$y=c_0 x^n+c_1 x^{n-1}+c_3 x^{n-2}+\ldots+c_n$$

where `n` is the degree of the polynomial model.



kst

Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit polynomial weighted (v0.1)

Plugin name: Fit polynomial weighted

Description: Generates a polynomial weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: Auto-correlation/Auto-correlated

Input vector - Y Array: Auto-correlation/Auto-correlated

Input vector - Weights: Auto-correlation/Auto-correlated

Input scalar - Order: 2

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.12.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights Array (vector) The array of weights to use for the fit.

Order (scalar) The order, or degree, of the polynomial model to use.

8.3.12.2 Outputs

Y Fitted (vector) The array of fitted y values.

Residuals (vector) The array of residuals.

Parameters (vector) The best fit parameters c_0, c_1,..., c_n.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

chiˆ2/nu (scalar) The weighted sum of squares of the residuals, divided by the degrees of freedom.

8.3.13 Fit polynomial

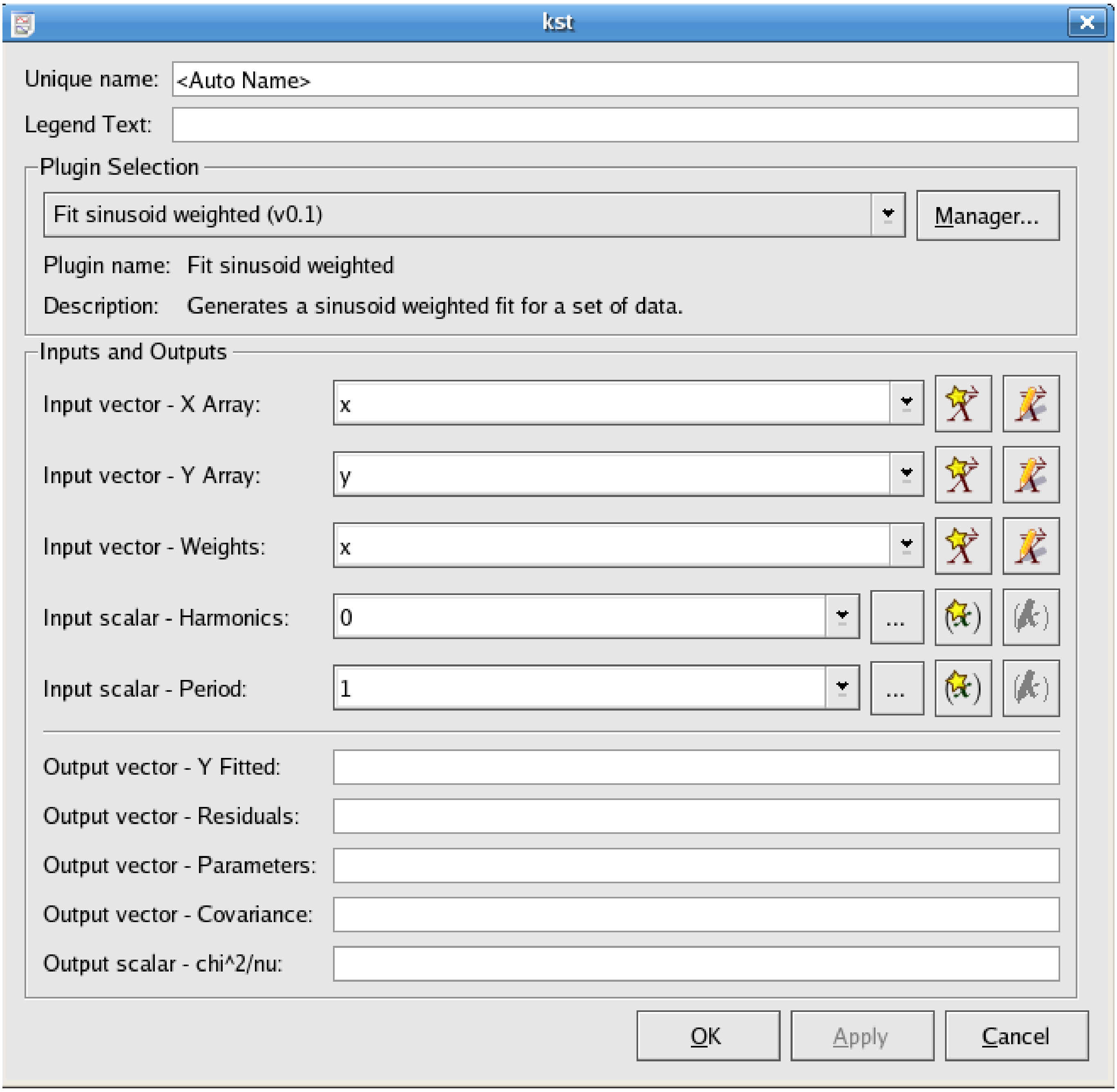
The Fit polynomial plugin is identical in function to the Fit polynomial weighted plugin with the exception that the weight value w_i is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.3.14 Fit sinusoid weighted

The Fit sinusoid weighted plugin performs a least-squares fit to a sinusoid model:

$$y=c_0+c_1\cos\left(\frac{1+1}{2}2\pi\frac{x}{T}\right)-c_2\sin\left(\frac{2}{2}2\pi\frac{x}{T}\right)+...+c_{n-1}\cos\left(\frac{(n-1)+1}{2}2\pi\frac{x}{T}\right)-c_n\sin\left(\frac{n}{2}2\pi\frac{x}{T}\right)$$

where `T` is the specified period, and `n=2+2H`, where `H` is the specified number of harmonics.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

Fit sinusoid weighted (v0.1)

Plugin name: Fit sinusoid weighted

Description: Generates a sinusoid weighted fit for a set of data.

Inputs and Outputs

Input vector - X Array: x

Input vector - Y Array: y

Input vector - Weights: x

Input scalar - Harmonics: 0

Input scalar - Period: 1

Output vector - Y Fitted:

Output vector - Residuals:

Output vector - Parameters:

Output vector - Covariance:

Output scalar - chi^2/nu:

8.3.14.1 Inputs

X Array (vector) The array of x values for the data points to be fitted.

Y Array (vector) The array of y values for the data points to be fitted.

Weights (vector) The array of weights to use for the fit.

Harmonics (scalar) The number of harmonics of the sinusoid to fit.

Period (scalar) The period of the sinusoid to fit.

8.3.14.2 Outputs

Y Fitted (vector) The array of fitted y values.

Residuals (vector) The array of residuals.

Parameters (vector) The best fit parameters c_0, c_1,..., c_n.

Covariance (vector) The covariance matrix of the model parameters, returned row after row in the vector.

chiˆ2/nu (scalar) The weighted sum of squares of the residuals, divided by the degrees of freedom.

8.3.15 Fit sinusoid

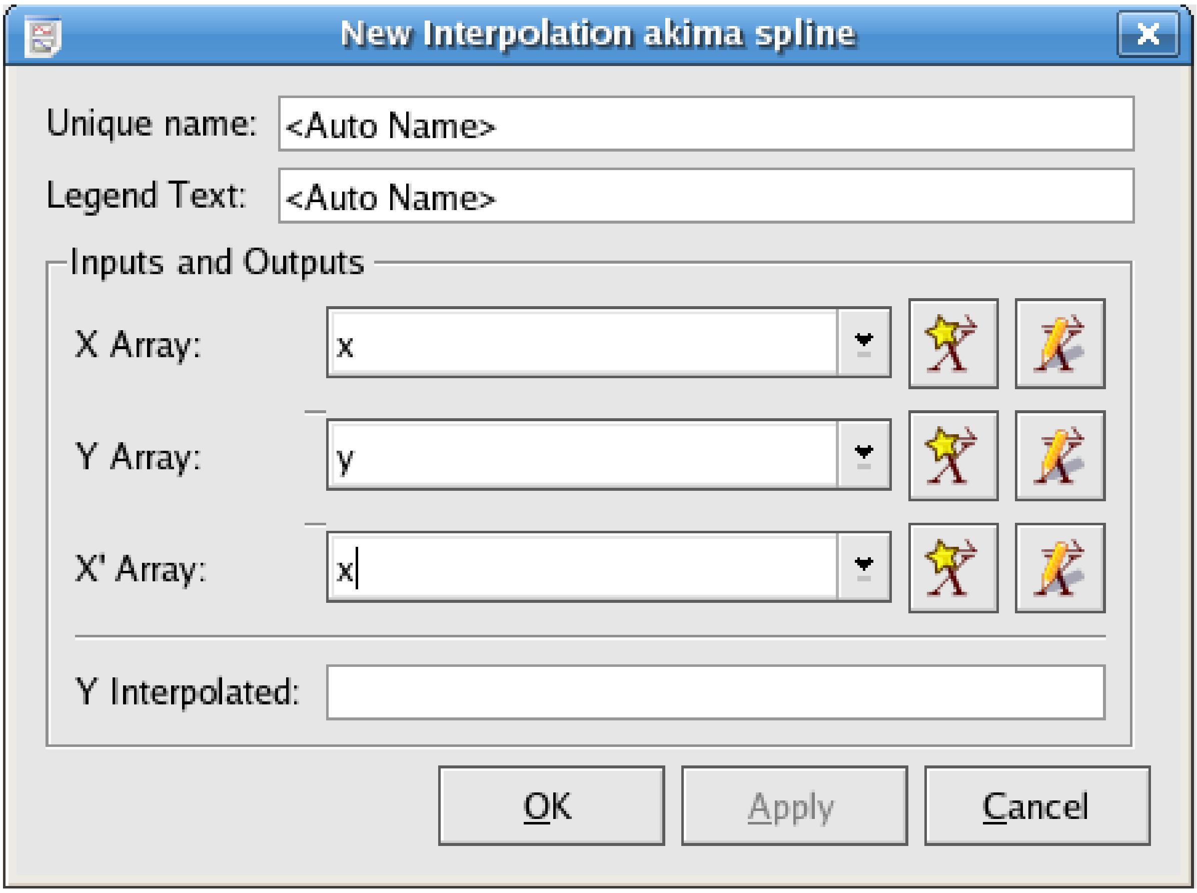
The Fit sinusoid plugin is identical in function to the Fit sinusoid weighted plugin with the exception that the weight value w_i is equal to `1` for all index values `i`. As a result, the Weights (vector) input does not exist.

8.4 Interpolation Plugins

Interpolation is a way to construct new data points within a data set. All Kst interpolation plugins take two vectors x and y, and a desired values x', then find the interpolation functions which fit x and y, and evaluate functions to get the corresponding y' value (interpolated y) for each x' value.

8.4.1 Interpolation Akima spline

The Interpolation Akima spline plugin generates a non-rounded Akima spline interpolation for the supplied data set, using natural boundary conditions.



New Interpolation akima spline

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

X Array: x

Y Array: y

X' Array: x

Y Interpolated:

8.4.1.1 Inputs

X Array (vector) The array of x values of the data points to generate the interpolation for.

Y Array (vector) The array of y values of the data points to generate the interpolation for.

X' Array (vector) The array of x values for which interpolated y values are desired.

8.4.1.2 Outputs

Y Interpolated (vector) The interpolated y values.

8.4.2 Interpolation Akima spline periodic

The interpolation akima periodic plugin generates a non-rounded Akima spline interpolation for the supplied data set, using periodic boundary conditions. The input and output options are the same as those for Interpolation Akima spline.

8.4.3 Interpolation cubic spline

The Interpolation cubic spline plugin generates a cubic spline interpolation for the supplied data set, using natural boundary conditions. The input and output options are the same as those for Interpolation Akima spline.

8.4.4 Interpolation cubic spline periodic

The Interpolation cubic spline periodic plugin generates a cubic spline interpolation for the supplied data set, using periodic boundary conditions. The input and output options are the same as those for Interpolation Akima spline.

8.4.5 Interpolation linear

The Interpolation linear plugin generates a linear interpolation for the supplied data set. The input and output options are the same as those for Interpolation Akima spline.

8.4.6 Interpolation polynomial

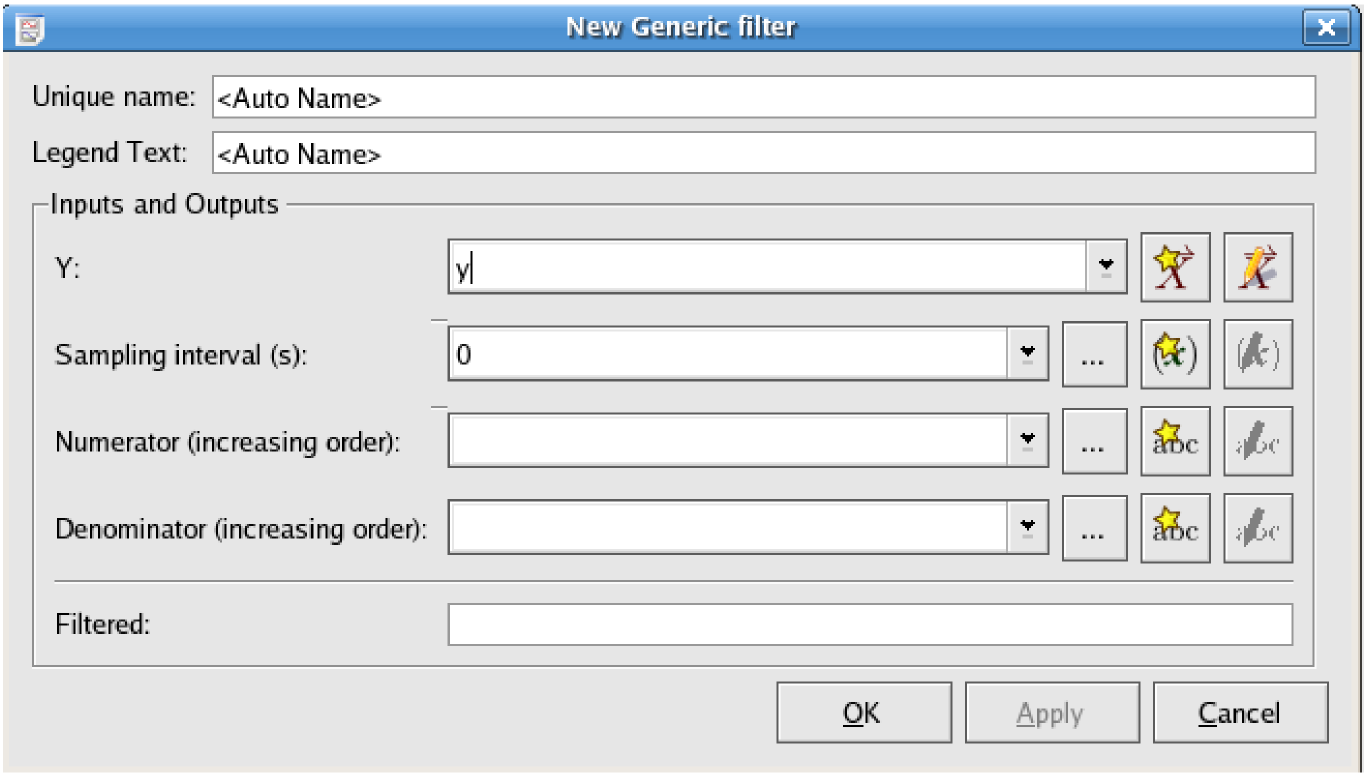
The Interpolation polynomial plugin generates a polynomial interpolation for the supplied data set. The number of terms in the polynomial used is equal to the number of points in the supplied data set. The input and output options are the same as those for Interpolation Akima spline.

8.5 Filter Plugins

Kst has a number of built-in filters.

8.5.1 Generic Filter

This plugin filters a set of data by using thebilinear transformation formula.



New Generic filter

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

Y: y

Sampling interval (s): 0

Numerator (increasing order):

Denominator (increasing order):

Filtered:

8.5.1.1 Inputs

Y (vector) The array of data to filter

Sampling interval(s) (scalar) The time periods for sampling of the input vector, reciprocal of the sampling frequency

Numerator/Denominator(increasing order) (string) A string of numbers separated by commas to indicate the coefficients of the polynomial as numerator/denominator in the transform formula. For example, if you have a polynomial: 2*zˆ2 + 3*z + 5, you will need to specify the string: 2,3,5. To input the string, click on the icon abc to construct a string with value 2,3,5

8.5.1.2 Outputs

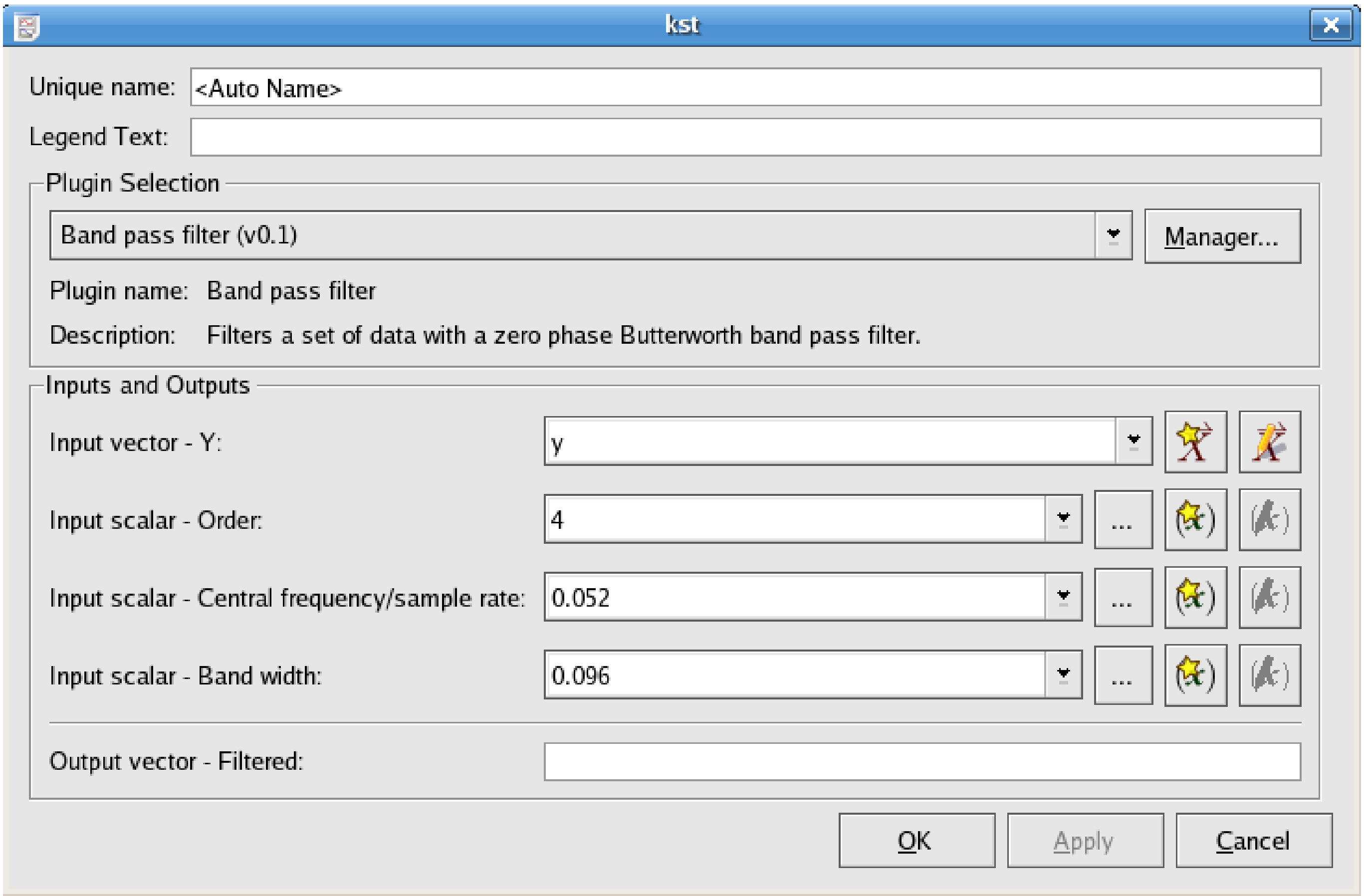
Filtered (vector) an array of filtered data

8.5.2 Band-pass filter

The Butterworth band-pass plugin filters a set of data by calculating the Fourier transform of the data and scaling the spectral amplitudes using the following function G(f)

$$G=\frac{1}{1+\left(\frac{f^2-f_c^2}{fb}\right)^{2n}}$$

where `f` is the frequency, f_c is the central frequency to filter around, `b` is the bandwidth to pass, and `n` is the order of the Butterworth filter. The inverse Fourier transform is then calculated using the scaled spectral amplitudes to give a filtered dataset.



kst

Unique name: <Auto Name>

Legend Text:

Plugin Selection

Band pass filter (v0.1)

Plugin name: Band pass filter

Description: Filters a set of data with a zero phase Butterworth band pass filter.

Inputs and Outputs

Input vector - Y: y

Input scalar - Order: 4

Input scalar - Central frequency/sample rate: 0.052

Input scalar - Band width: 0.096

Output vector - Filtered:

8.5.2.1 Inputs

Y (vector) The array of values to filter.

Order (scalar) The order of the Butterworth filter to use.

Central frequency/sample rate (scalar) The frequency to filter around.

Band width (scalar) The width of the band to pass. This should be the difference between the desired high cutoff frequency and the low cutoff frequency.

8.5.2.2 Outputs

Filtered (vector) The array of filtered data values.

8.5.3 Band-stop filter

The Butterworth band-stop effectively removes signals in a given frequency range from an input vector. The options are the same as for the Band-pass filter except that the scaling function G(f) which is used is now given by:

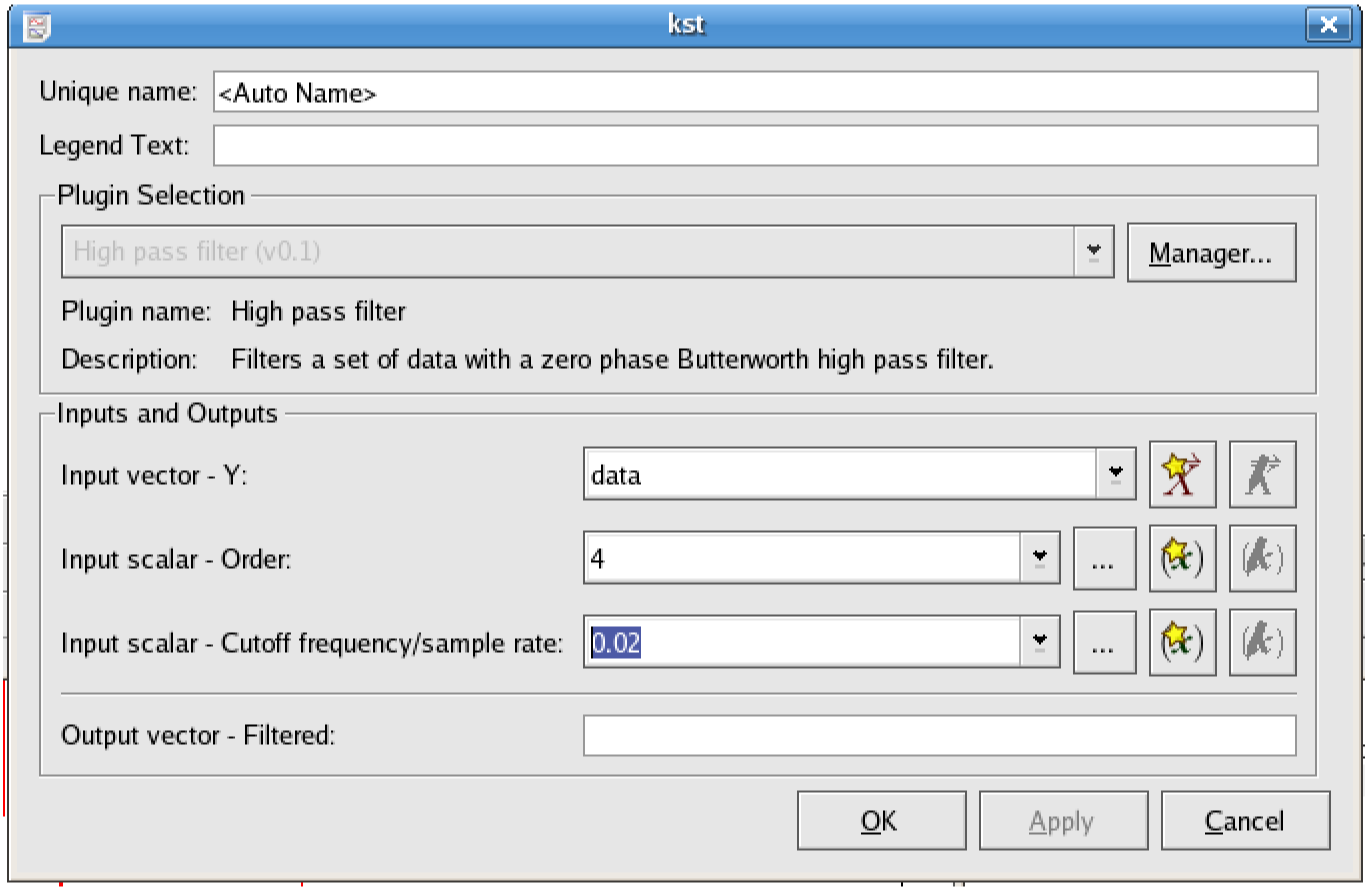
$$G=\frac{1}{1+\left(\frac{fb}{f^2-f_c^2}\right)^{2n}}$$

8.5.4 High-pass filter

The Butterworth high-pass plugin effectively removes signals below a given cutoff frequency from a data vector. It does this by by calculating the Fourier transform of the data and rescaling the the spectral amplitudes using the following following function G(f)

$$G=\frac{1}{1+\left(\frac{f_c}{f}\right)^{2n}}$$

where `f` is the frequency, f_c is the high frequency cutoff, and `n` is the order of the Butterworth filter. The inverse Fourier transform is then calculated using the new filtered frequency responses.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

High pass filter (v0.1)

Plugin name: High pass filter

Description: Filters a set of data with a zero phase Butterworth high pass filter.

Inputs and Outputs

Input vector - Y: data

Input scalar - Order: 4

Input scalar - Cutoff frequency/sample rate: 0.02

Output vector - Filtered:

8.5.4.1 Inputs

Y (vector) The array of values to filter.

Order (scalar) The order of the Butterworth filter to use.

Cutoff frequency/sample rate (scalar) The cutoff frequency (`f`$_c$).

8.5.4.2 Outputs

Filtered (vector) The array of filtered data values.

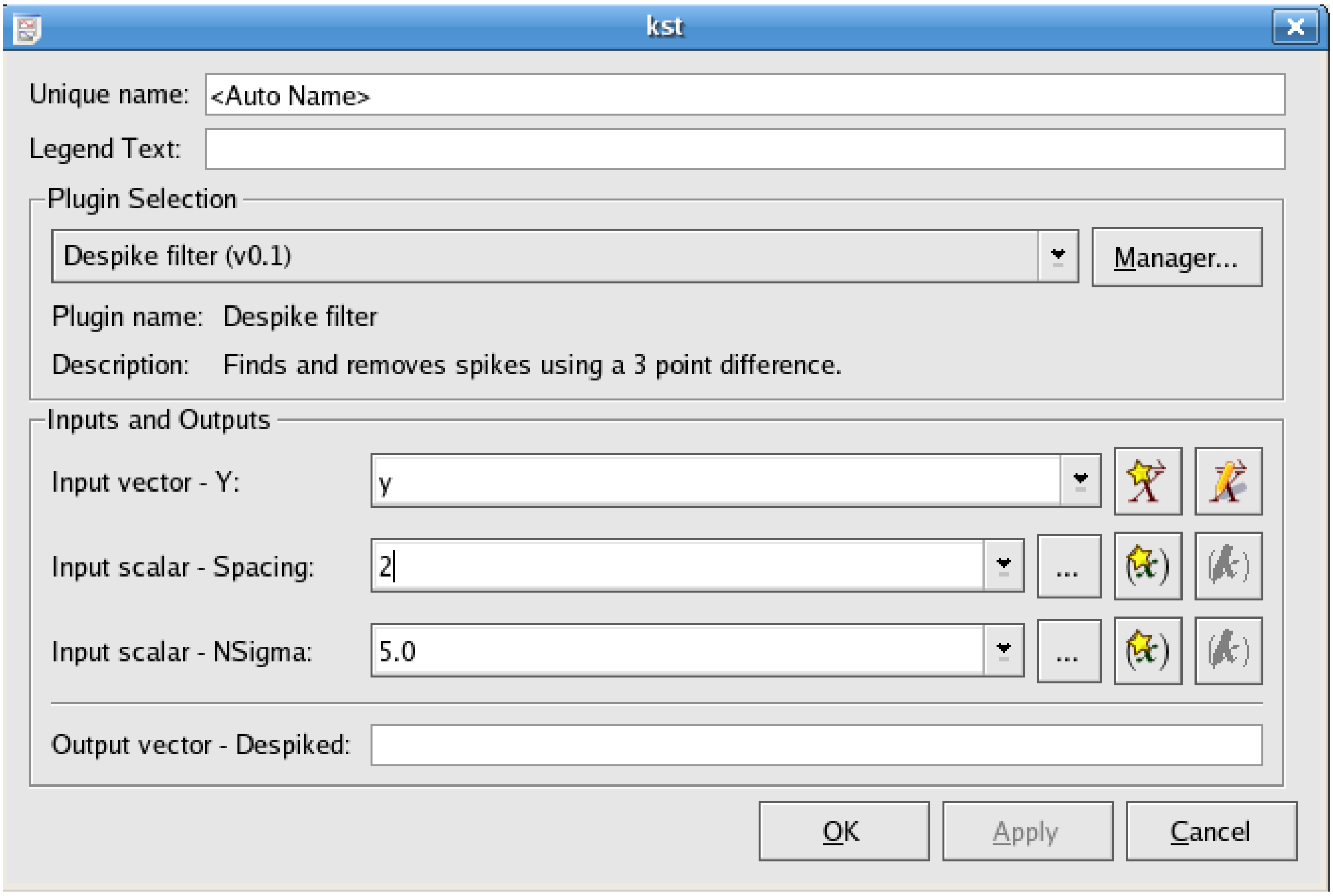
8.5.5 Low-pass filter

The Butterworth low-pass filter effectively removes signals above a given frequency cutoff from an input vector. The options are the same as for the High-pass filter except that the scaling function G(f) which is used is now given by:

$$G = \frac{1}{1 + \left(\frac{f}{f_c}\right)^{2n}}$$

8.5.6 Despike Filter

This plugin is used to find and remove spikes in a data set.



Unique name: <Auto Name>

Legend Text:

Plugin Selection

Despike filter (v0.1)

Plugin name: Despike filter

Description: Finds and removes spikes using a 3 point difference.

Inputs and Outputs

Input vector - Y: y

Input scalar - Spacing: 2

Input scalar - NSigma: 5.0

Output vector - Despiked:

8.5.6.1 Inputs

Y (vector) Select the data vector needed to be smoothed.

Spacing (scalar) Specify the width of the spikes.

NSigma (scalar) If a data point value varies from its neighboring data points by `NSigma` times more, it will be counted as a spike.

8.5.6.2 Outputs

Despiked (vector) This plugin outputs a smoothed data set with all specified spikes removed.

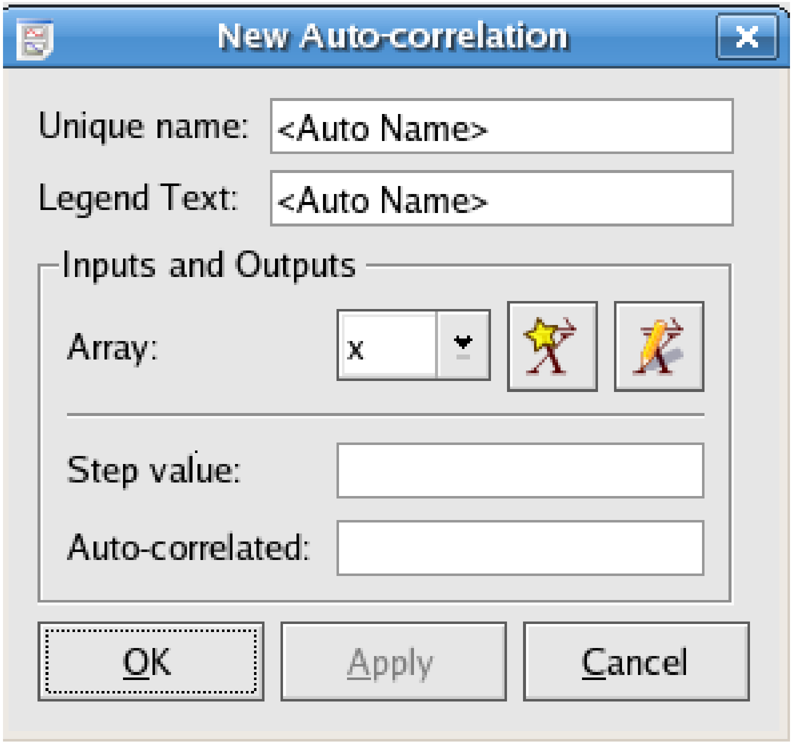
8.6 Signal Processing Plugins

Kst provides various plugins to help you easily process signals.

8.6.1 Autocorrelation

The autocorrelation plugin calculates correlation values between a series (vector) and a lagged version of itself, using lag values from `floor(-(N-1)/2)` to `floor((N-1)/2)`, where `N` is the number of points in the data set. The time vector is not an input as it is assumed that the data is sampled at equal time intervals. The correlation value `r` at lag `k` is:

$$r_k = \sum_{i=1}^{N-k} ((x_i - \bar{x})(y_{i-k} - \bar{y}))$$



8.6.1.1 Inputs

Array (vector) The array x of values to calculate correlation values for.

8.6.1.2 Outputs

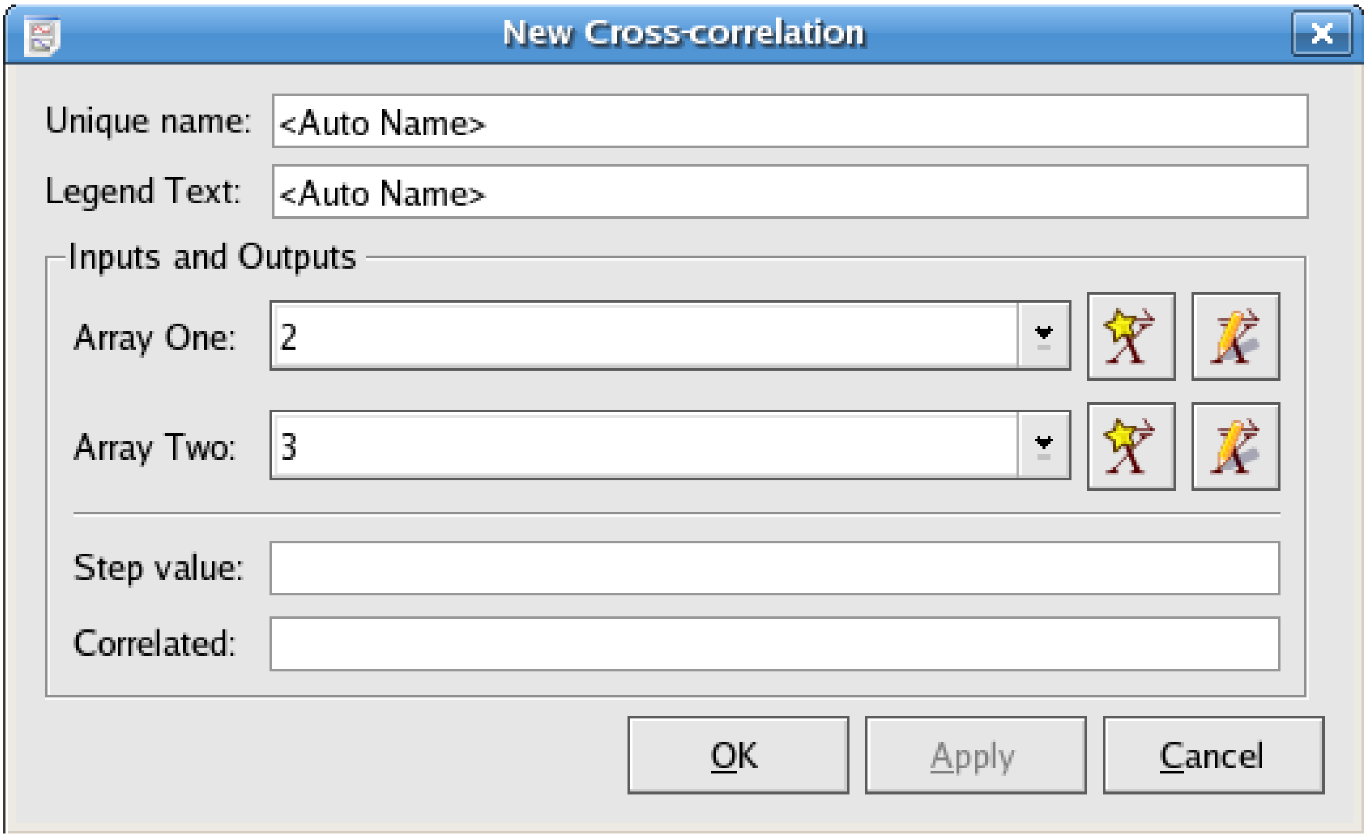
Step Value (vector) The array of step, or lag values.

Auto-correlated (vector) The array of correlation values calculated using the corresponding step number in the Step Number vector.

8.6.2 Crosscorrelation

The crosscorrelation plugin calculates correlation values between two series (vectors) x and y, using lag values from `floor(--(N-1)/2)` to `floor((N-1)/2)`, where `N` is the number of elements in the longer vector. The shorter vector is padded to the length of the longer vector using `0`s. The time vector is not an input as it is assumed that the data is sampled at equal time intervals. The correlation value `r` at lag `k` is:

$$r_k = \sum_{i=1}^{N-k} ((x_i - \bar{x})(y_{i-k} - \bar{y}))$$



Unique name: <Auto Name>

Legend Text: <Auto Name>

Array One: 2

Array Two: 3

Step value:

Correlated:

8.6.2.1 Inputs

Array One (vector) The array x used in the cross-correlation formula.

Array Two (vector) The array y used in the cross-correlation formula.

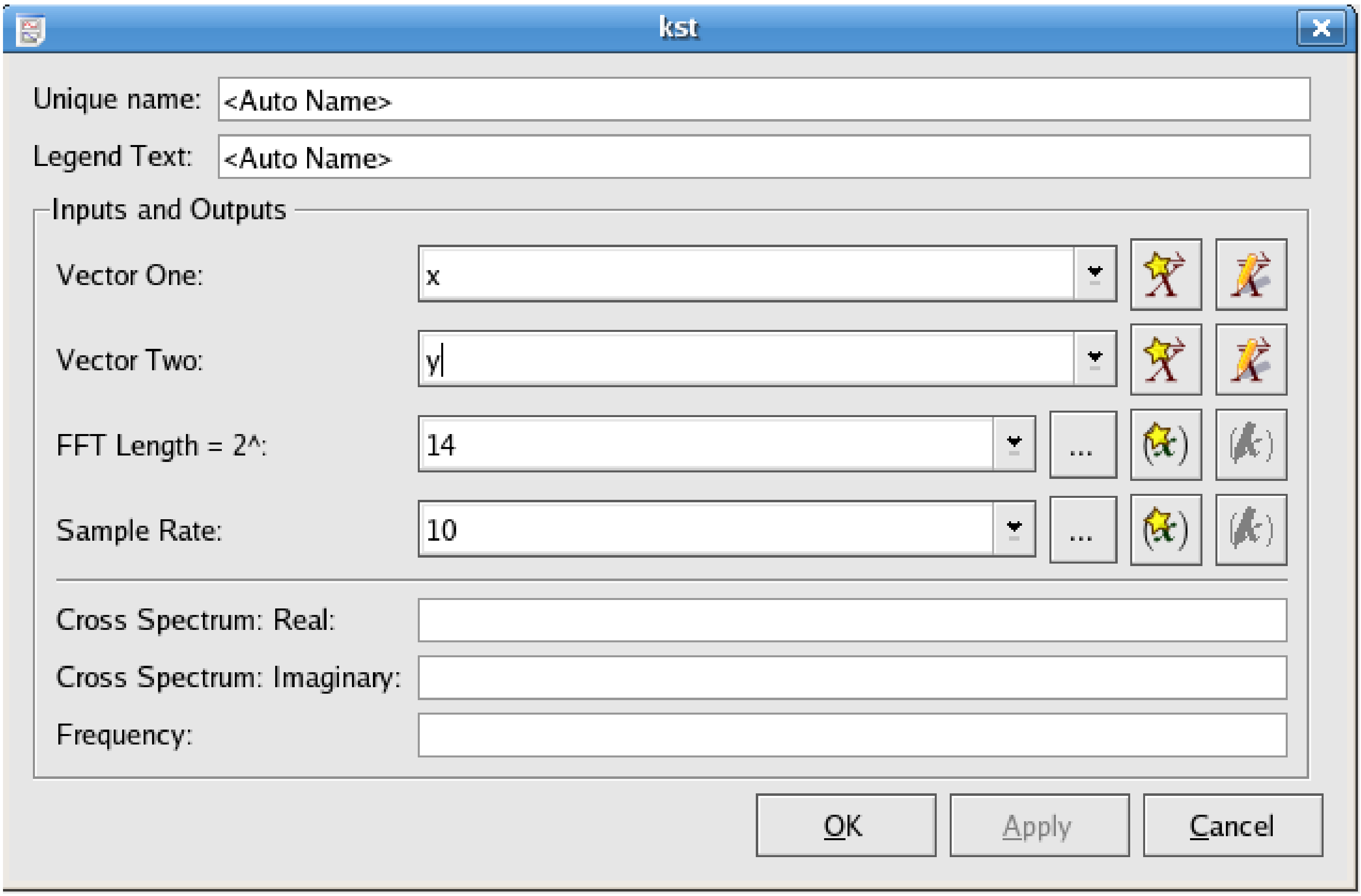
8.6.2.2 Outputs

Step value (vector) The array of step, or lag values.

Correlation (vector) The array of correlation values calculated using the corresponding step number in the Step value vector.

8.6.3 Cross Power Spectrum

The cross power spectrum plugin takes two vectors and calculates the FFT of their cross-correlation values.



Unique name: <Auto Name>

Legend Text: <Auto Name>

Vector One: x

Vector Two: y

FFT Length = 2^: 14

Sample Rate: 10

Cross Spectrum: Real:

Cross Spectrum: Imaginary:

Frequency:

8.6.3.1 Inputs

Vector One/Two The two vectors used to calculate the cross power spectrum

FFT Length= 2ˆ (scalar) Refers to FFT options in Spectrum

Sample rate (scalar) Refers to FFT options in Spectrum

8.6.3.2 Outputs

Cross Spectrum: Real(vector) The real part of the calculated cross power spectrum.

Cross Spectrum: Imaginary (vector) The imaginary part of the calculated cross power spectrum.

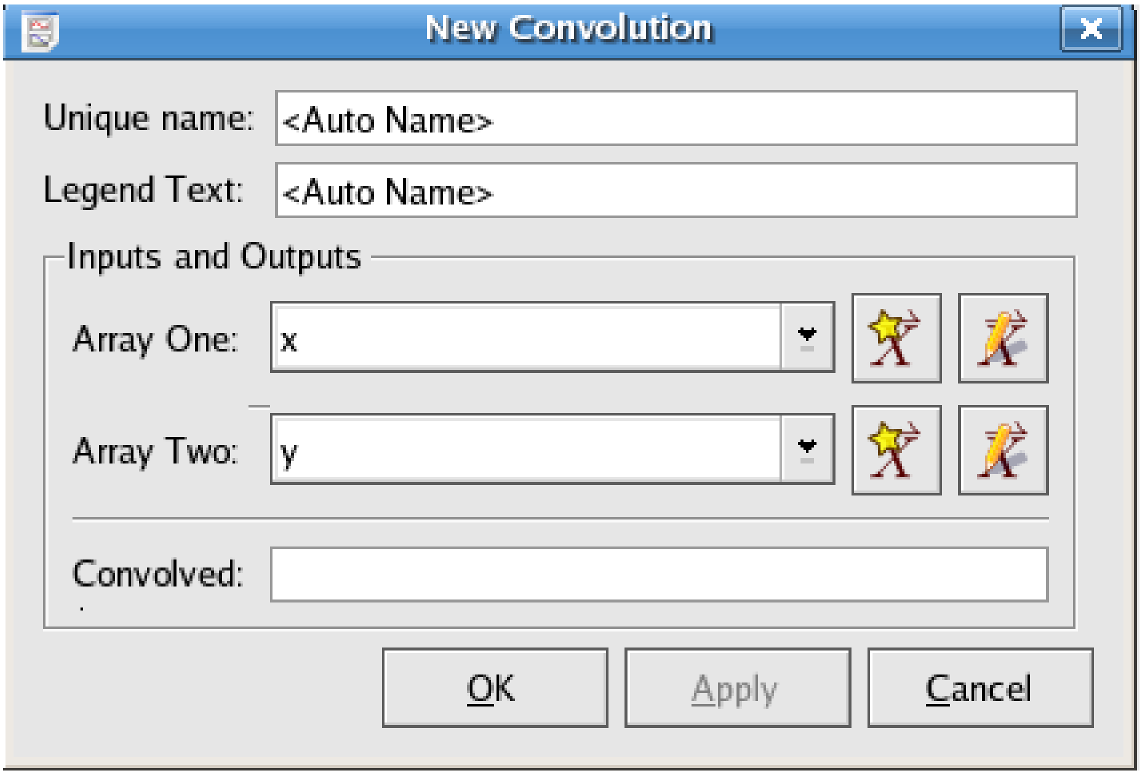
Frequency (vector) The frequency vector of the output cross power spectrum.

8.6.4 Convolution

The convolution plugin generates the convolution of one vector with another. The convolution of two functions `f` and `g` is given by:

$$f * g = \int_{-\infty}^{\infty} f(x) g(t-x) dx$$

The order of the vectors does not matter, since f*g=g*f. In addition, the vectors do not need to be of the same size, as the plugin will automatically interpolate smallest vector to the larger length.



Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

Array One: x

Array Two: y

Convolved:

8.6.4.1 Inputs

Array One (vector) One of the pair of arrays to take the convolution of.

Array Two (vector) One of the pair of arrays to take the convolution of.

8.6.4.2 Outputs

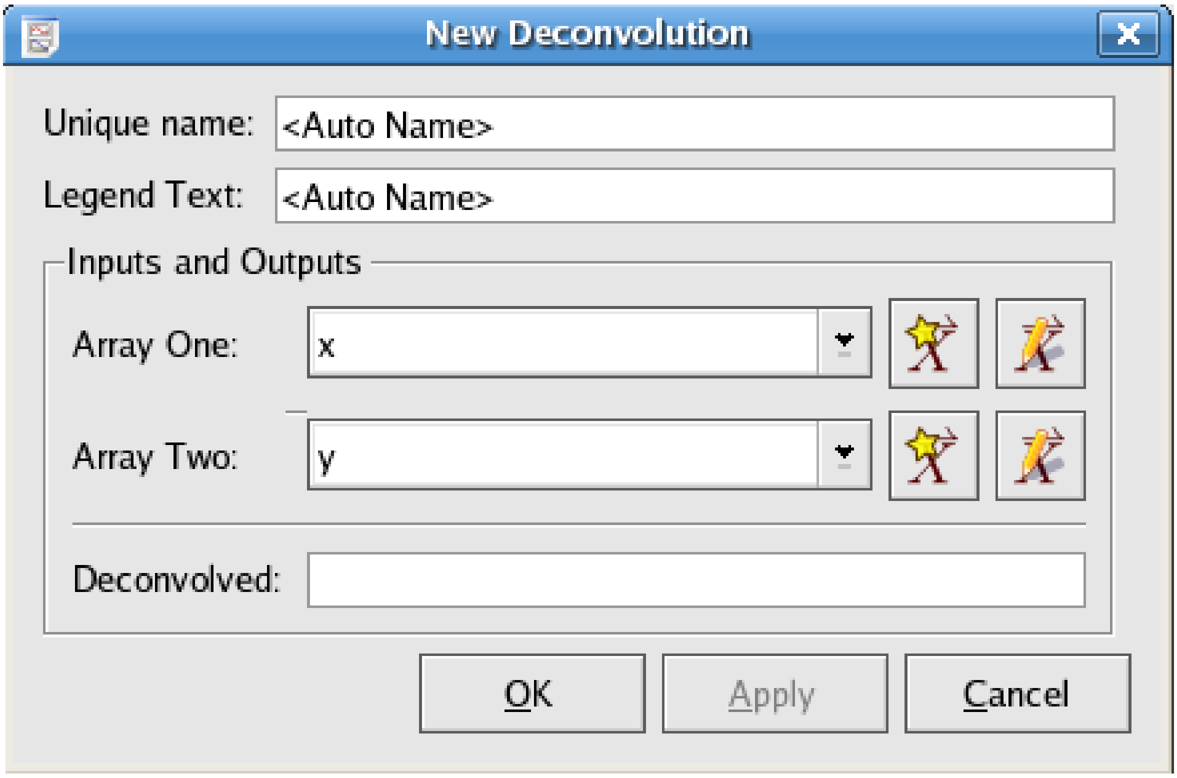
Convolved (vector) The convolution of the two input vectors.

8.6.5 Deconvolution

The deconvolution plugin generates the deconvolution of one vector with another. Deconvolution is the inverse of convolution. Given the convolved vector h and another vector g, the deconvolution f is given by:

$$f*g=h$$

The vectors do not need to be of the same size, as the plugin will automatically extrapolate the shorter vector. The shorter vector is assumed to be the response function g.



New Deconvolution

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

Array One: x

Array Two: y

Deconvolved:

8.6.5.1 Inputs

Array One (vector) One of the pair of arrays to take the deconvolution of.

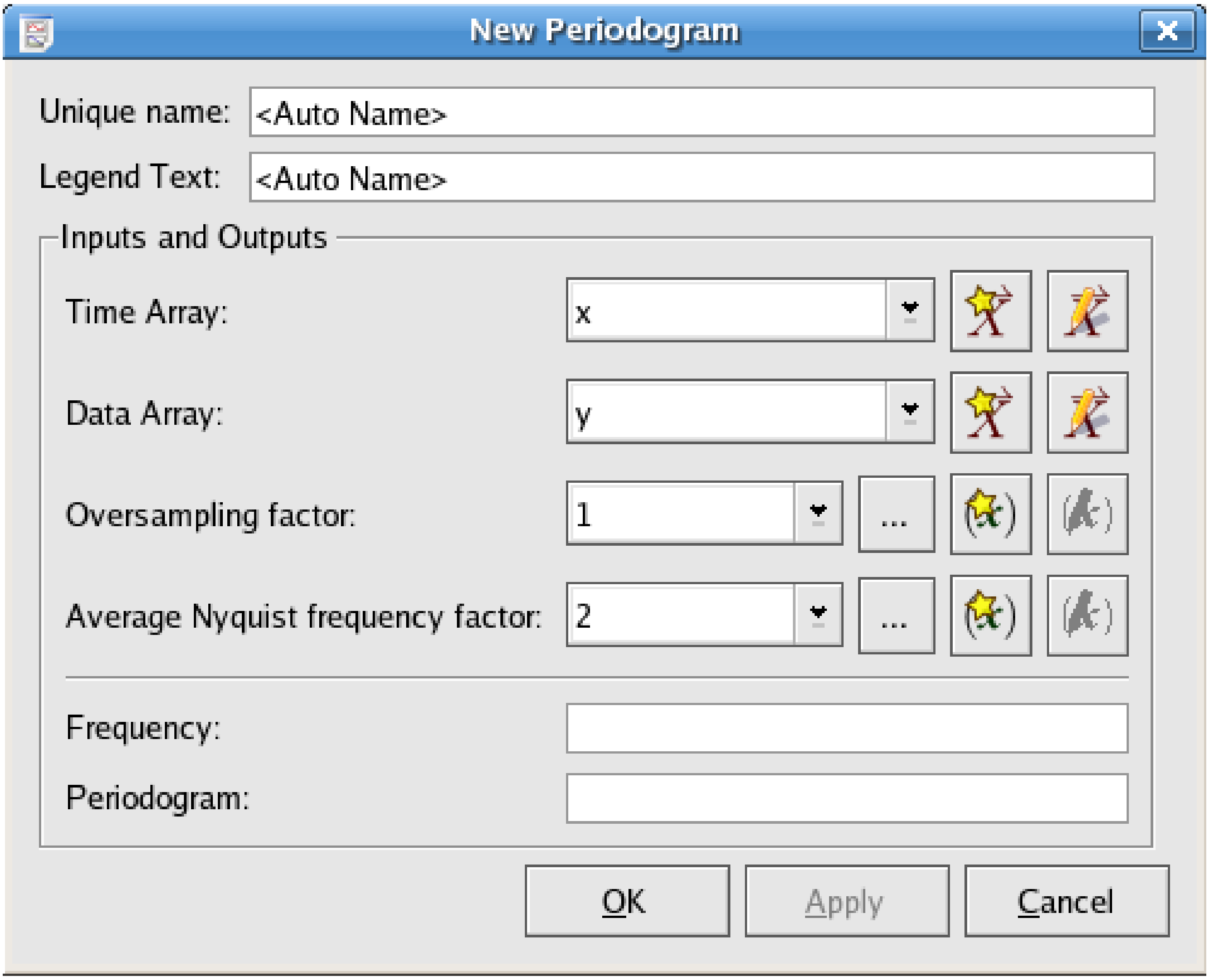
Array Two (vector) One of the pair of arrays to take the deconvolution of.

8.6.5.2 Outputs

Deconvolved (vector) The deconvolution of the two input vectors.

8.6.6 Periodogram

The periodogram plugin produces the periodogram of a given data set. Periodogram is an estimate of the spectral density of a signal.



New Periodogram

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

Time Array: x

Data Array: y

Oversampling factor: 1

Average Nyquist frequency factor: 2

Frequency:

Periodogram:

8.6.6.1 Inputs

Time Array (vector) The array of time values of the data points to generate the interpolation for.

Data Array (vector) The array of data values, dependent on the time values of the data points to generate the interpolation for.

Oversampling factor (scalar) The factor to oversample by.

Average Nyquist frequency factor (scalar) The average Nyquist frequency factor.

8.6.6.2 Outputs

Frequency (vector) The frequency vector.

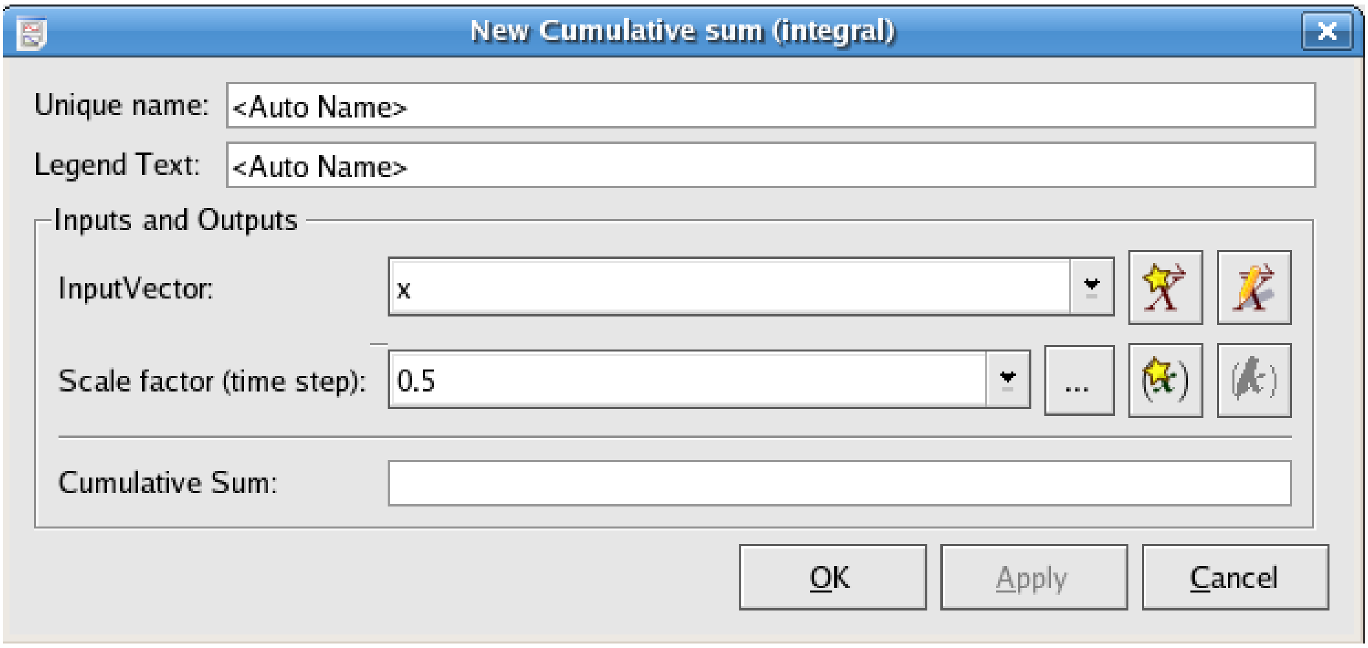
Periodogram (vector) The frequency response vector for the periodogram.

8.7 Calculus Plugins

There are some plugins which can be used to do basic calculus.

8.7.1 Cumulative Sum (integral)

This plugin sums up the input vector cumulatively over a chosen time step which is the time interval between two neighboring data points of the input vector. As the time step becomes smaller, the calculated sum will approach this integral: $\int x(t)dt$,where x is the input data vector.



New Cumulative sum (integral)

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

InputVector: x

Scale factor (time step): 0.5

Cumulative Sum:

8.7.1.1 Inputs

InputVector The vector used to calculate the cumulative sum.

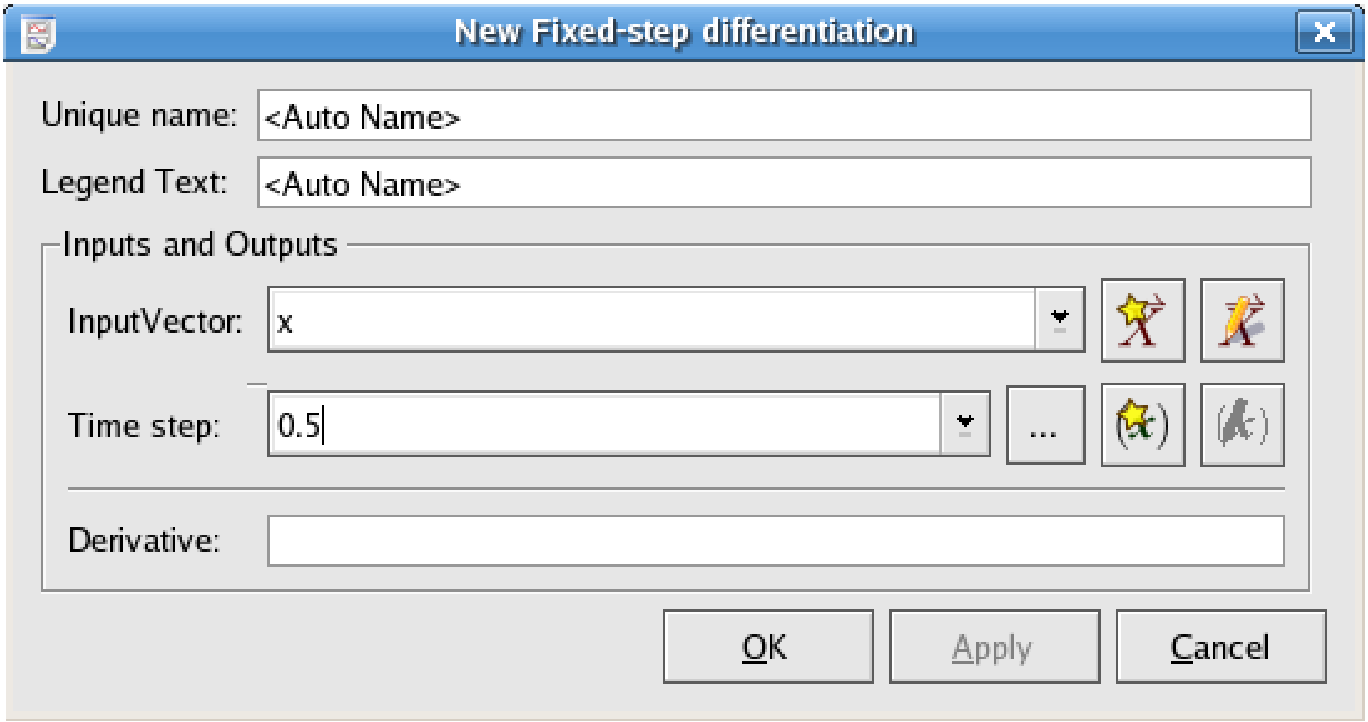
Scale factor(time step) (scalar) The interval size for each step of calculating the cumulative sum

8.7.1.2 Outputs

Cumulative Sum (vector) This array stores values of the cumulative sum for each step.

8.7.2 Fixed Step Differentiation

This plugin calculates derivative values with respect to time for an input vector. As time step becomes smaller, the output vector will reach the values of dx/dt, where x is the input vector.


New Fixed-step differentiation

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

InputVector: x

Time step: 0.5

Derivative:

8.7.2.1 Inputs

inputVector the data vector to take derivative of

Time step (scalar) Specify the size of the time interval between any two neighbor values in the input vector

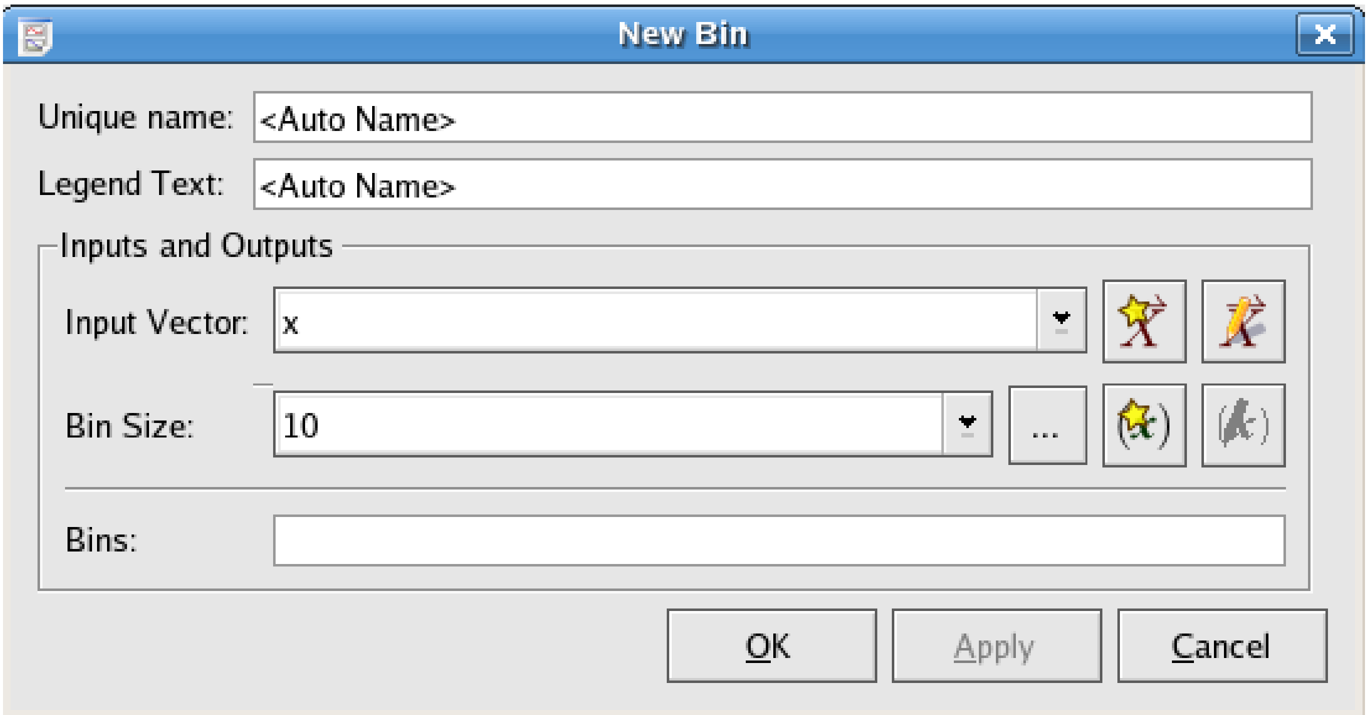
8.7.2.2 Outputs

Derivative (vector) This plugin outputs the derivative values for every time step.

8.8 Miscellaneous Plugins

8.8.1 Bin

The bin plugin group elements of a single data vector into bins of a specified size. The value of each bin is the mean of the elements belonging to the bin. For example, if the bin size is `3`, and the input vector is `[9,2,7,3,4,74,5,322,444,2,-1]`, then the outputted bins would be `[6,27,257]`. Note that any elements remaining at the end of the input vector that do not form a complete bin (in this case, elements `2` and `1`), are simply discarded.



Unique name: <Auto Name>

Legend Text: <Auto Name>

Input Vector: x

Bin Size: 10

Bins:

8.8.1.1 Inputs

Input Vector (vector) The vector to bin.

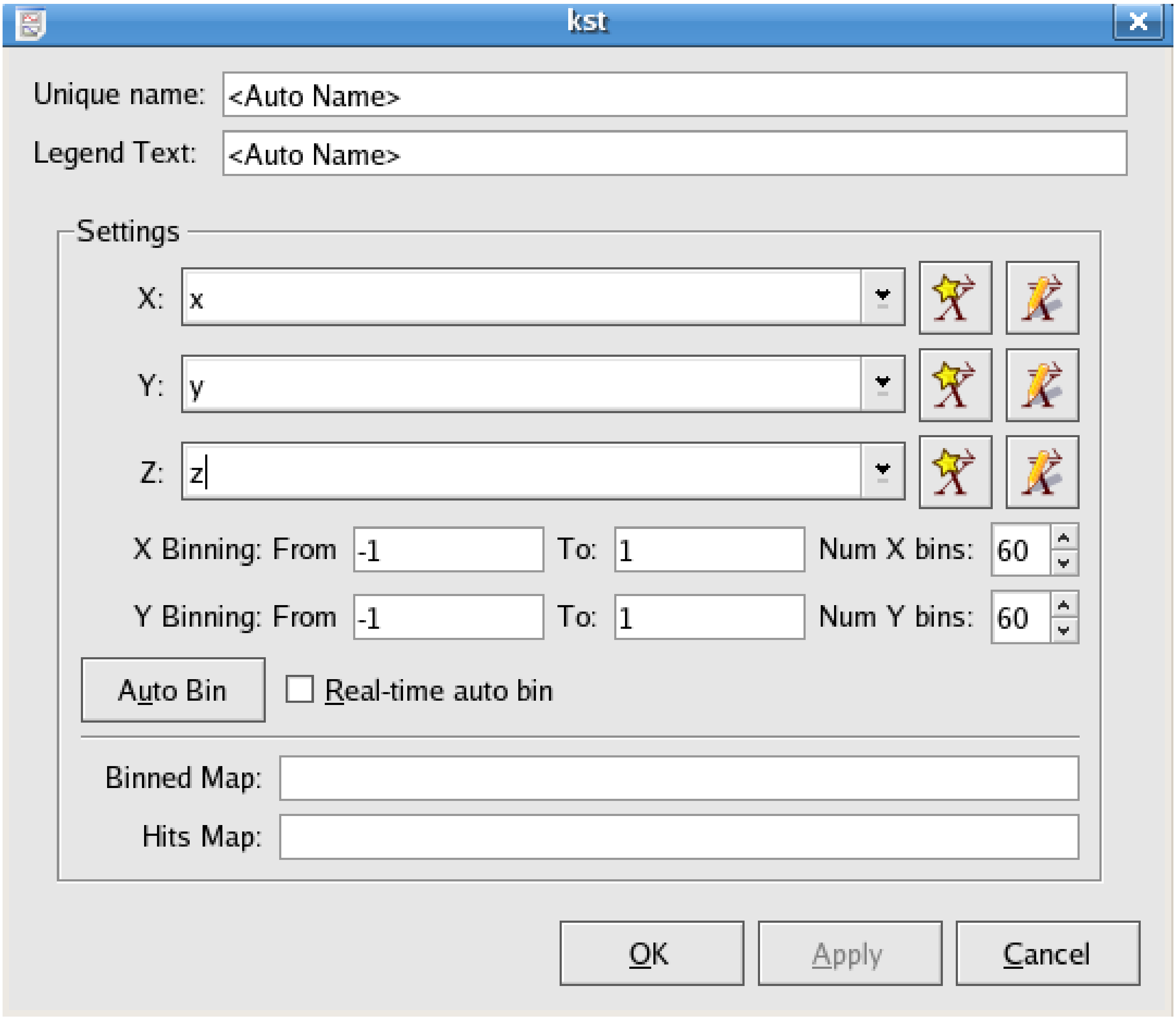
Bin Size (scalar) The size (# of elements) to have in each bin.

8.8.1.2 Outputs

Bins (vector) The array of means for each bin.

8.8.2 Binned Map

The binned map can generate a surface count or plot of a two-variable function.



Unique name: <Auto Name>

Legend Text: <Auto Name>

X: x

Y: y

Z: z

X Binning: From -1 To: 1 Num X bins: 60

Y Binning: From -1 To: 1 Num Y bins: 60

☐ Real-time auto bin

Binned Map:

Hits Map:

8.8.2.1 Settings

X,Y,Z (vector) X/Y vectors are used to specify values of the two independent variables in a function, and Z vector specifies the values of the dependent variable.

X/Y Binning: From/To (scalar) Specify the range of X/Y values.

Num X/Y bins (scalar) Specify the number of resolution grid for the output binned map.

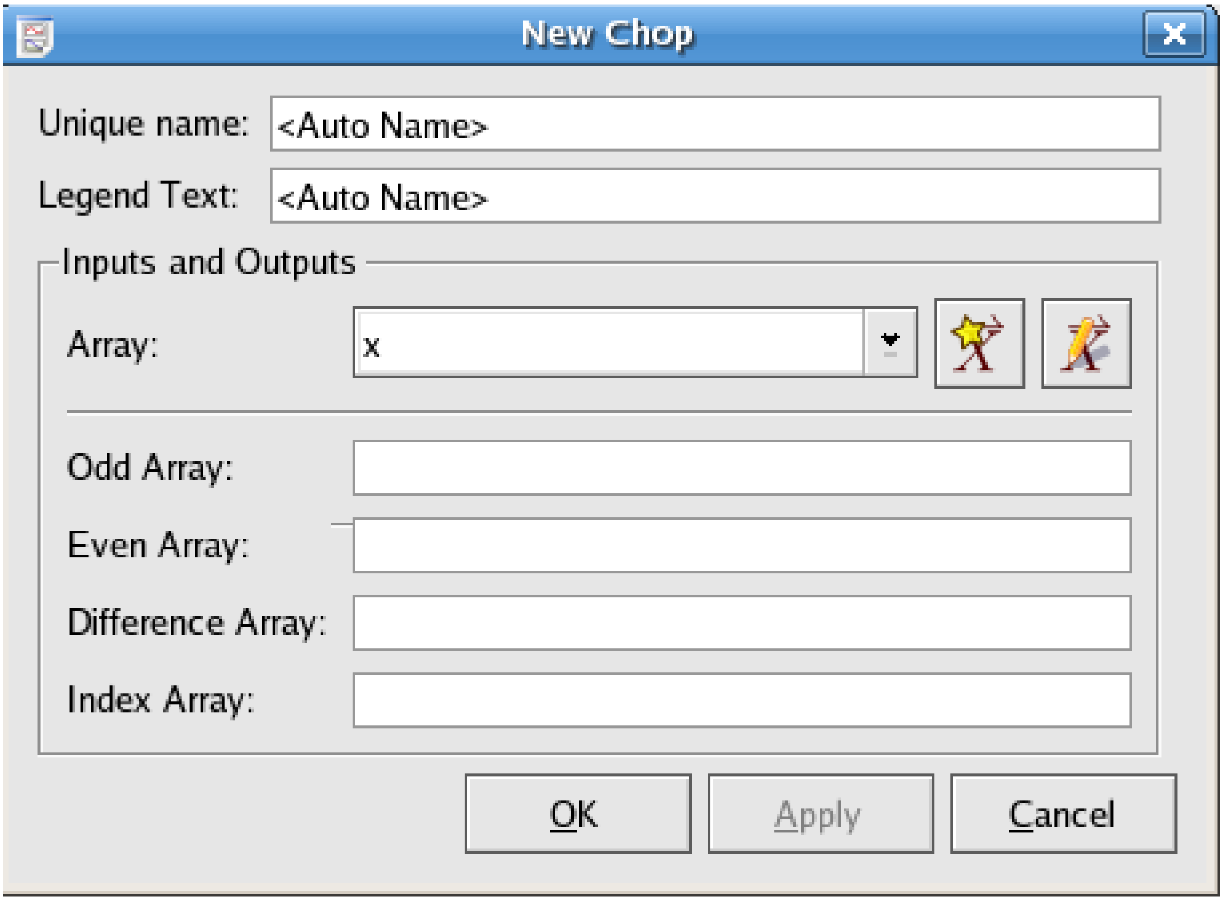
8.8.2.2 Outputs

Binned Map (matrix) Binned Map is a matrix whose entries can be used to generate the surface contour plot of the Z vector.

Hits Map (matrix) Hits Map is a matrix whose values are specified by the number of points in each X/Y bin, and the image of Hits Map indicates the positions where Z values have been located in the X/Y plane.

8.8.3 Chop

The chop plugin takes an input vector and divides it into two vectors. Every second element in the input vector is placed in one output vector, while all other elements from the input vector are placed in another output vector.



Unique name: <Auto Name>

Legend Text: <Auto Name>

Array: x

8.8.3.1 Inputs

Array (vector) The array of values to perform the chop on.

8.8.3.2 Outputs

Odd Array (vector) The array containing the odd part of the input array (i.e. it contains the first element of the input array).

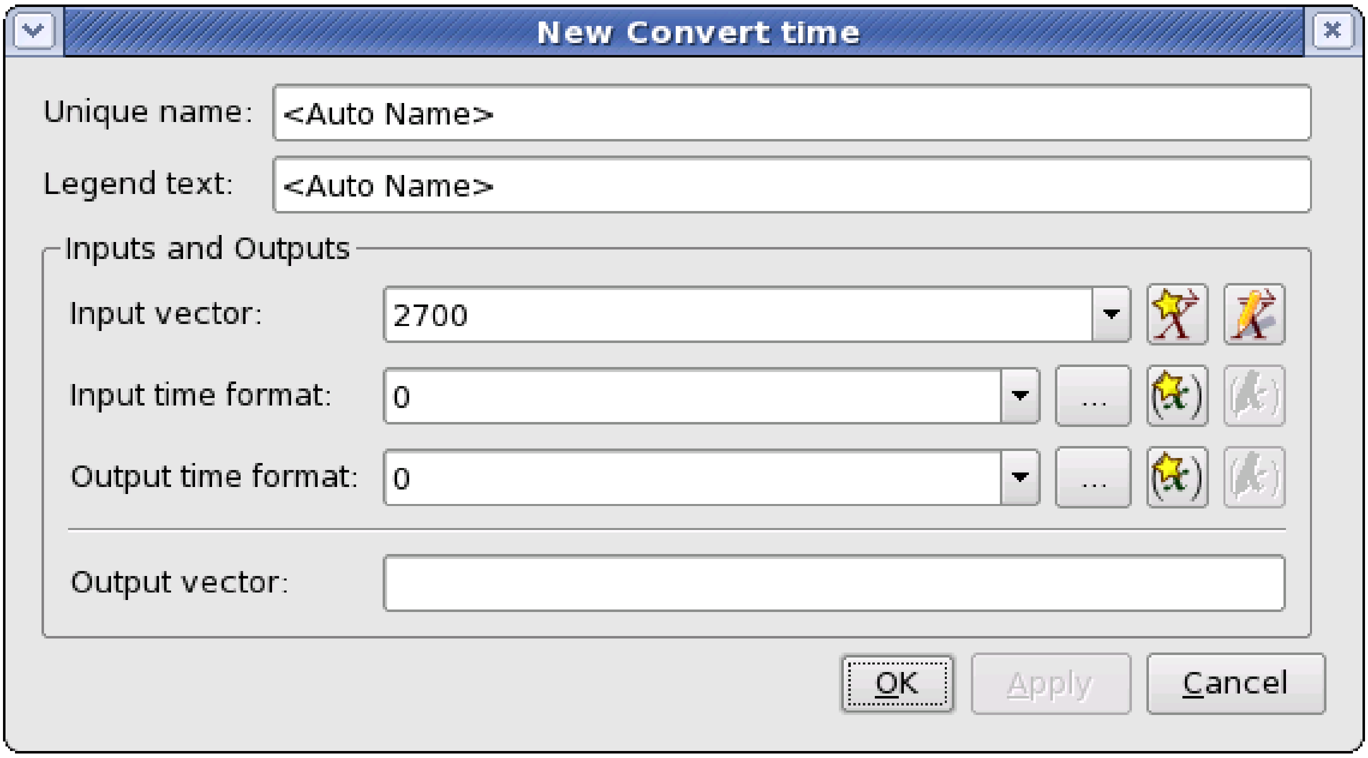
Even Array (vector) The array containing the even part of the input array (i.e. it does not contain the first element of the input array).

Difference Array (vector) The array containing the elements of the odd array minus the respective elements of the even array.

Index Array (vector) An index array has the same length as the other three output arrays.

8.8.4 Convert time

The Convert time plugin converts a vector from one time format to another:



8.8.4.1 Inputs

Input vector The vector holding the time values to be converted.

Input time format (scalar) The time format of the values in the input vector. The formats are:

0: standard C time
1: TAI
2: JD
3: MJD
4: RJD
5: JY
6: TAI (ns)
7: TAI (2ˆ-16 sec)

Output time format (scalar) The time format for the values in the output vector. The formats are the same as for the input time format.

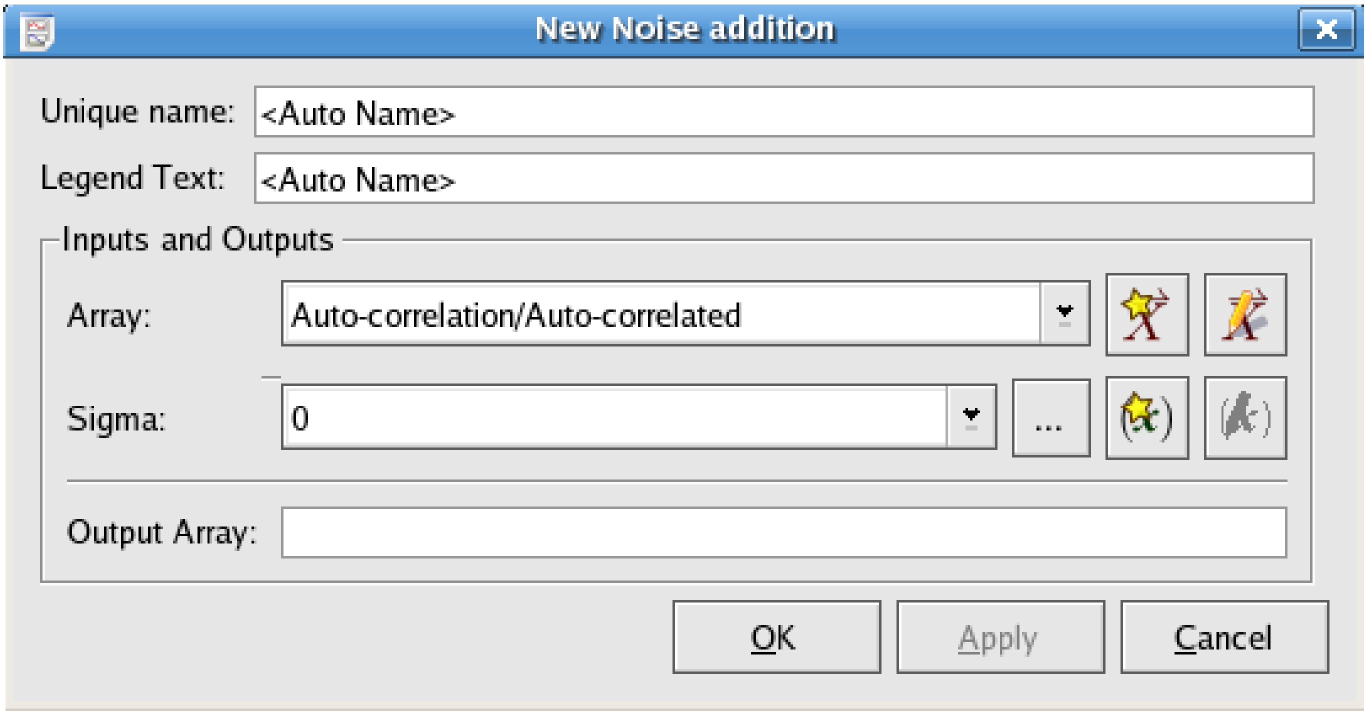
8.8.4.2 Outputs

Output vector The vector holding the converted time values, which will be the same size as the input vector.

8.8.5 Noise Addition

The Noise addition plugin adds a Gaussian random variable to each element of the input vector. The Gaussian distribution used has a mean of 0 and the specified standard deviation. The probability density function of a Gaussian random variable is:

$$p(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right)$$



New Noise addition

Unique name: <Auto Name>

Legend Text: <Auto Name>

Inputs and Outputs

Array: Auto-correlation/Auto-correlated

Sigma: 0

Output Array:

8.8.5.1 Inputs

Array (vector) The array of elements to which random noise is to be added.

Sigma (scalar) The standard deviation to use for the Gaussian distribution.

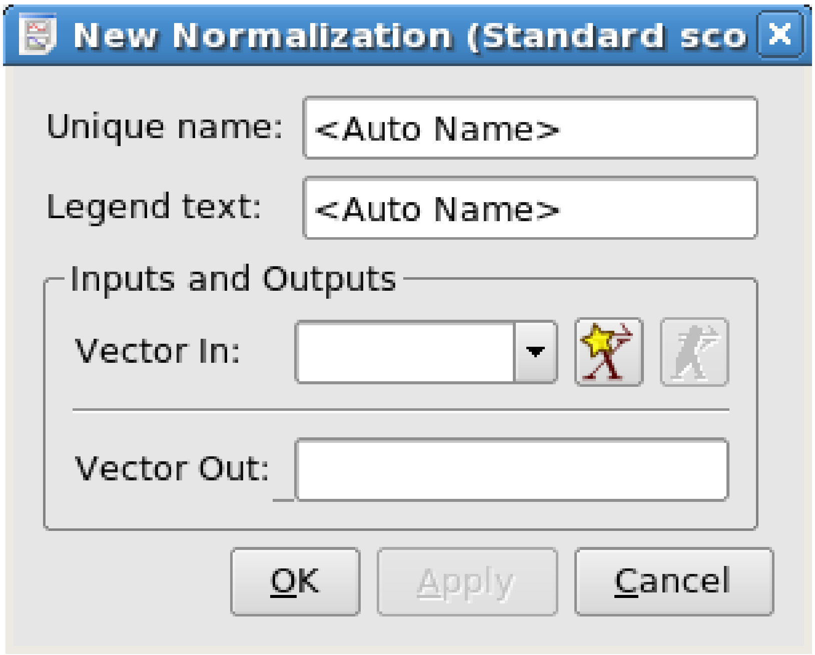
8.8.5.2 Outputs

Output Array (vector) The array of elements with Gaussian noise added.

8.8.6 Normalization (Standard score)

This plugin subtracts the mean from the input vector and divides by the standard deviation.

$$z = \frac{x - \mu}{\sigma}$$



New Normalization (Standard sco

Unique name: <Auto Name>

Legend text: <Auto Name>

Inputs and Outputs

Vector In:

Vector Out:

8.8.6.1 Inputs

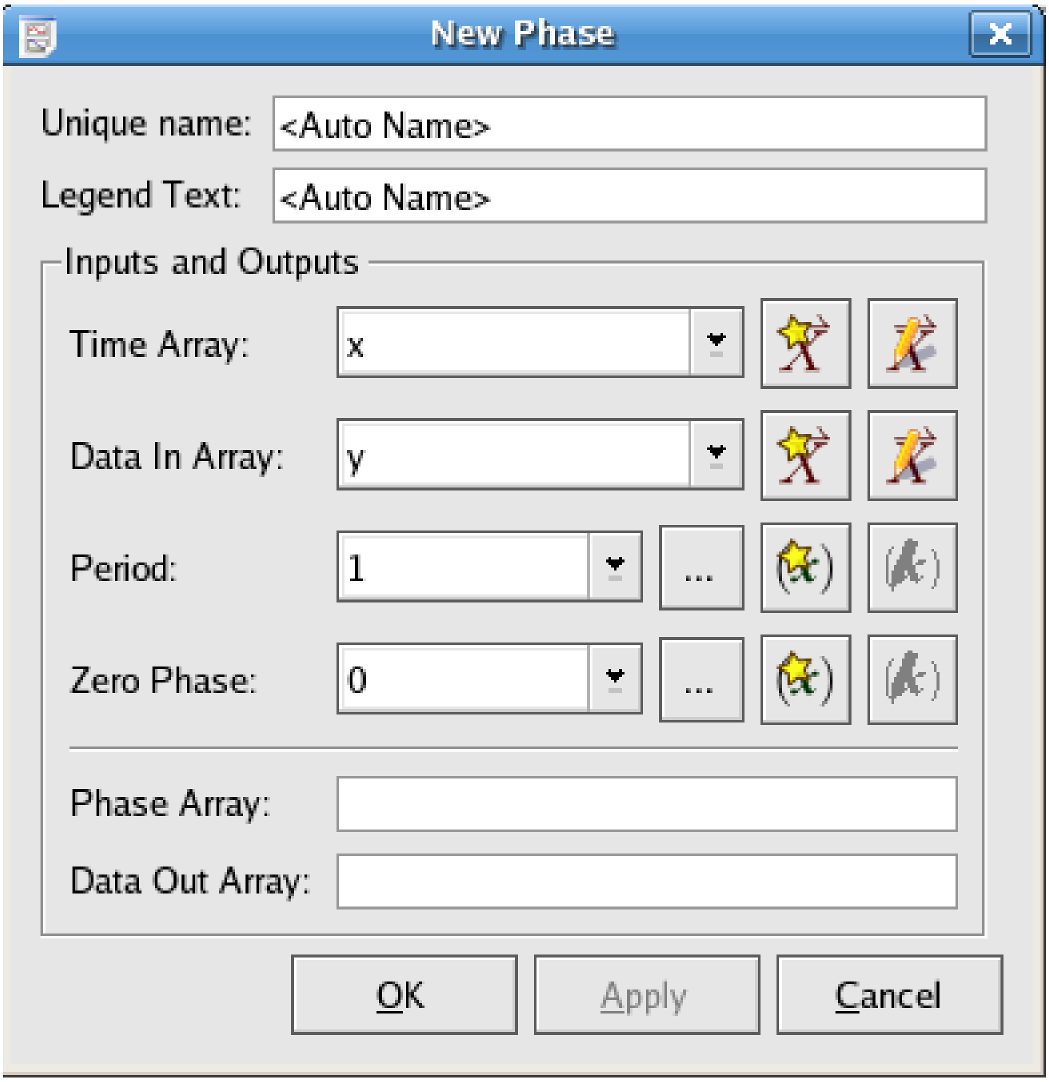
Vector In The vector to normalize

8.8.6.2 Outputs

Vector Out Output the original vector after normalizing.

8.8.7 Phase

This plugin calculates the phase value for each data point of an input time array.



Unique name: <Auto Name>

Legend Text: <Auto Name>

Time Array: x

Data In Array: y

Period: 1

Zero Phase: 0

Phase Array:

Data Out Array:

8.8.7.1 Inputs

Time Array (vector) An array of time values

Data In Array (vector) Input data array corresponding to the time array

Period (scalar) Set the period of the input time array.

Zero Phase (scalar) Set the time for the zero phase.

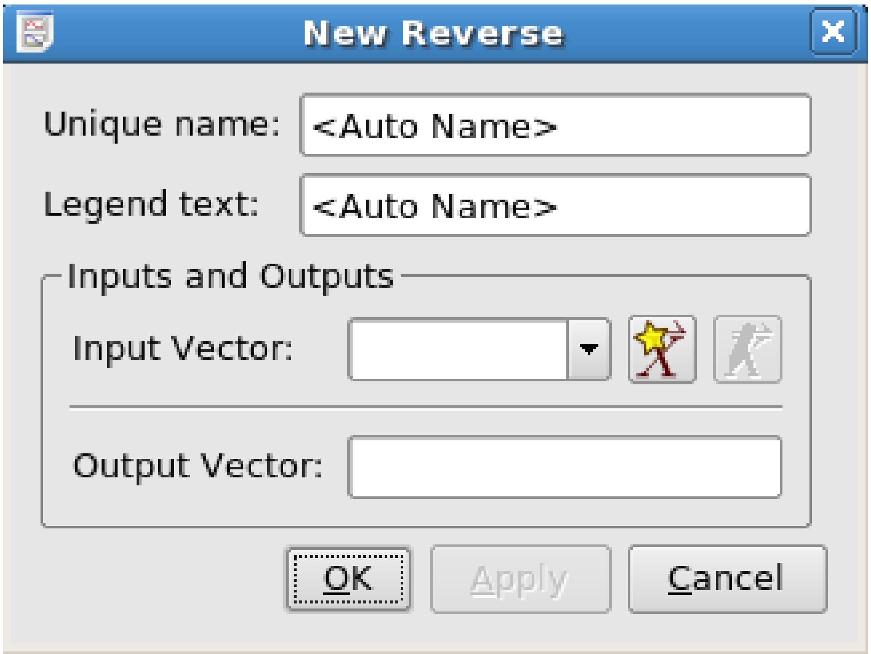
8.8.7.2 Outputs

Phase Array (vector) An array of phase values corresponding to the input data values.

Data Out Array (vector) Output the original data set with values sorted by the phase array.

8.8.8 Reverse

This plugin reverses the input vector.



Unique name: <Auto Name>

Legend text: <Auto Name>

8.8.8.1 Inputs

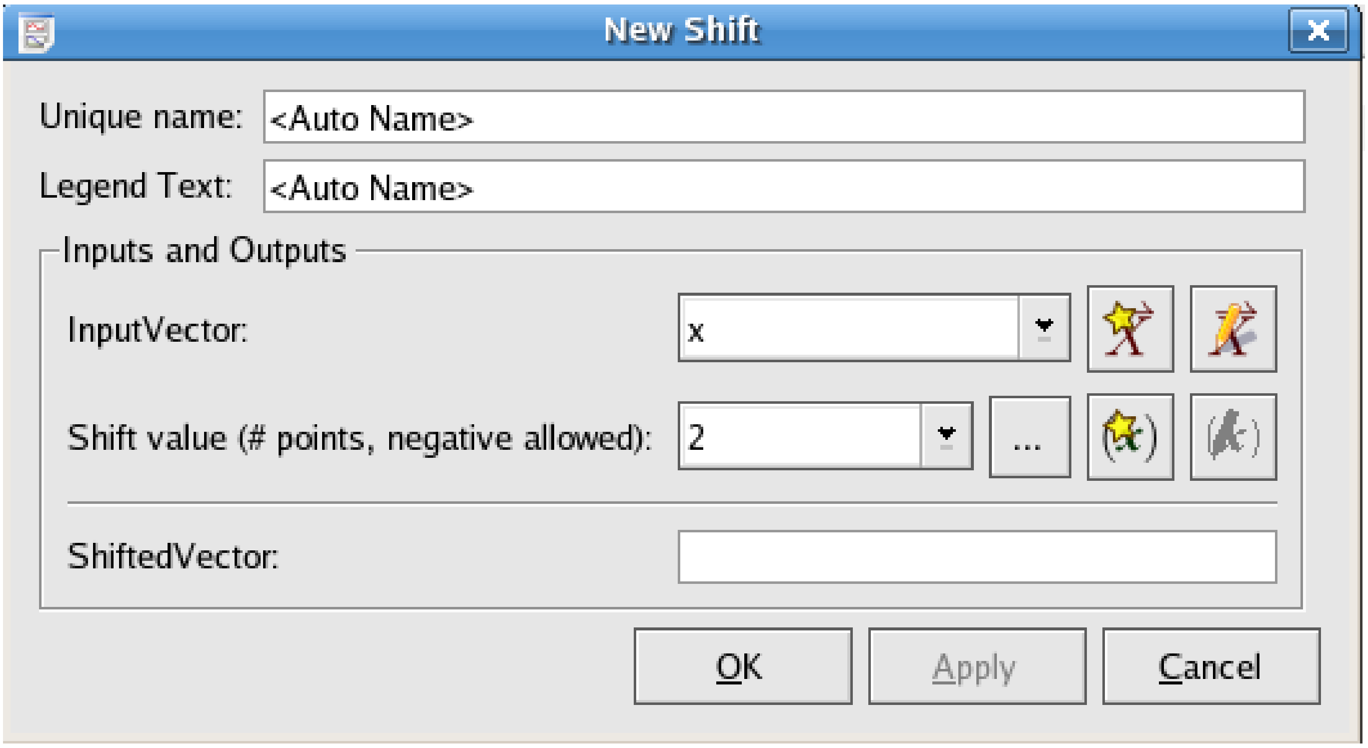
InputVector The vector to reverse.

8.8.8.2 Outputs

Output Vector The reversed vector.

8.8.9 Shift

This plugin shifts the plot of an input vector forwards or backwards on the X axis.



8.8.9.1 Inputs

InputVector The vector to shift.

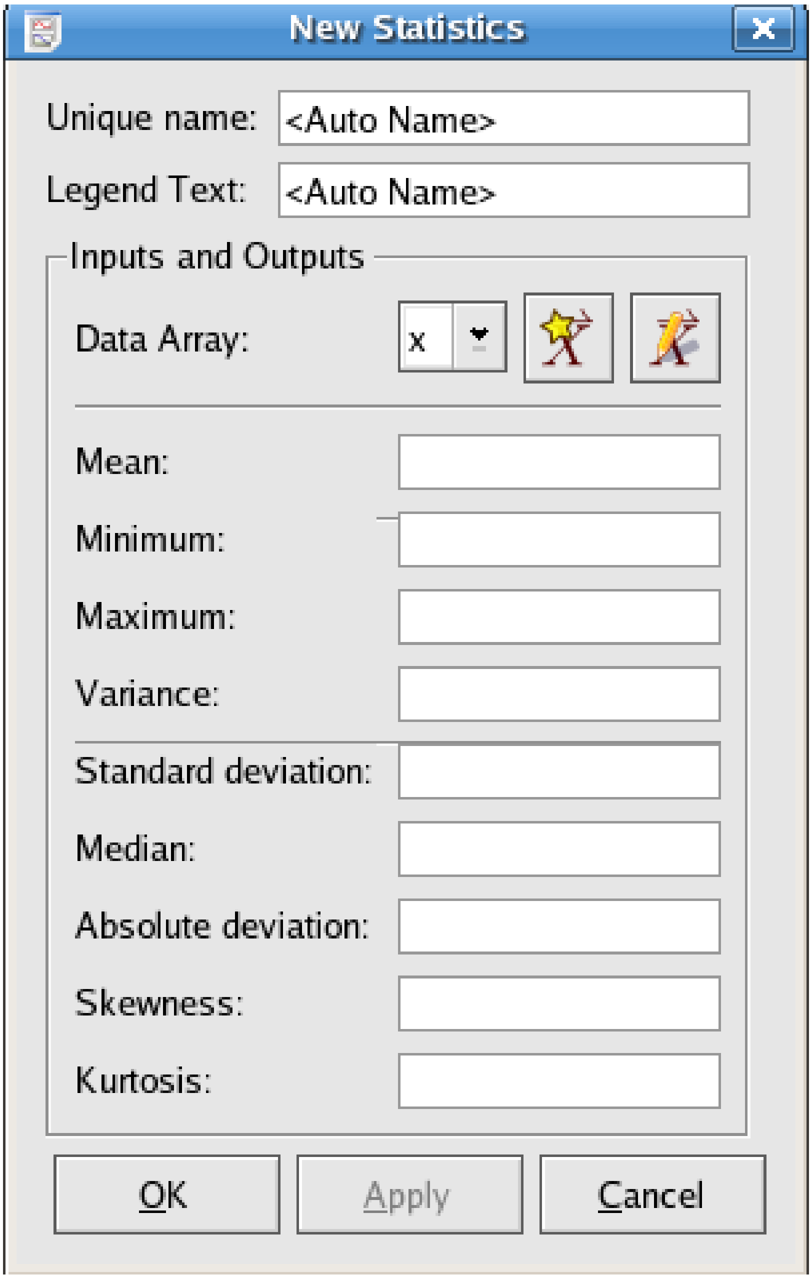
Shift value(#points, negative allowed) (scalar) Specify the units to be shifted. Positive shifted value results right or forward shift. Negative shifted value results left or backwards shift.

8.8.9.2 Outputs

ShiftedVector Output the original vector after shifting by the units indicated above.

8.8.10 Statistics

The statistics plugin calculates statistics for a given data set beyond those automatically calculated by Kst. Most of the output scalars are named such that the values they represent should be apparent. Standard formulae are used to calculate the statistical values.



New Statistics

Unique name: <Auto Name>

Legend Text: <Auto Name>

Data Array: x

8.8.10.1 Inputs

Data Array (vector) The array of data whose statistic values needed to be calculated.

8.8.10.2 Outputs

Mean (scalar) The mean of the data values.

Minimum (scalar) The minimum value found in the data array.

Maximum (scalar) The maximum value found in the data array.

Variance (scalar) The variance of the data set.

Standard deviation (scalar) The standard deviation of the data set.

Median (scalar) The median of the data set.

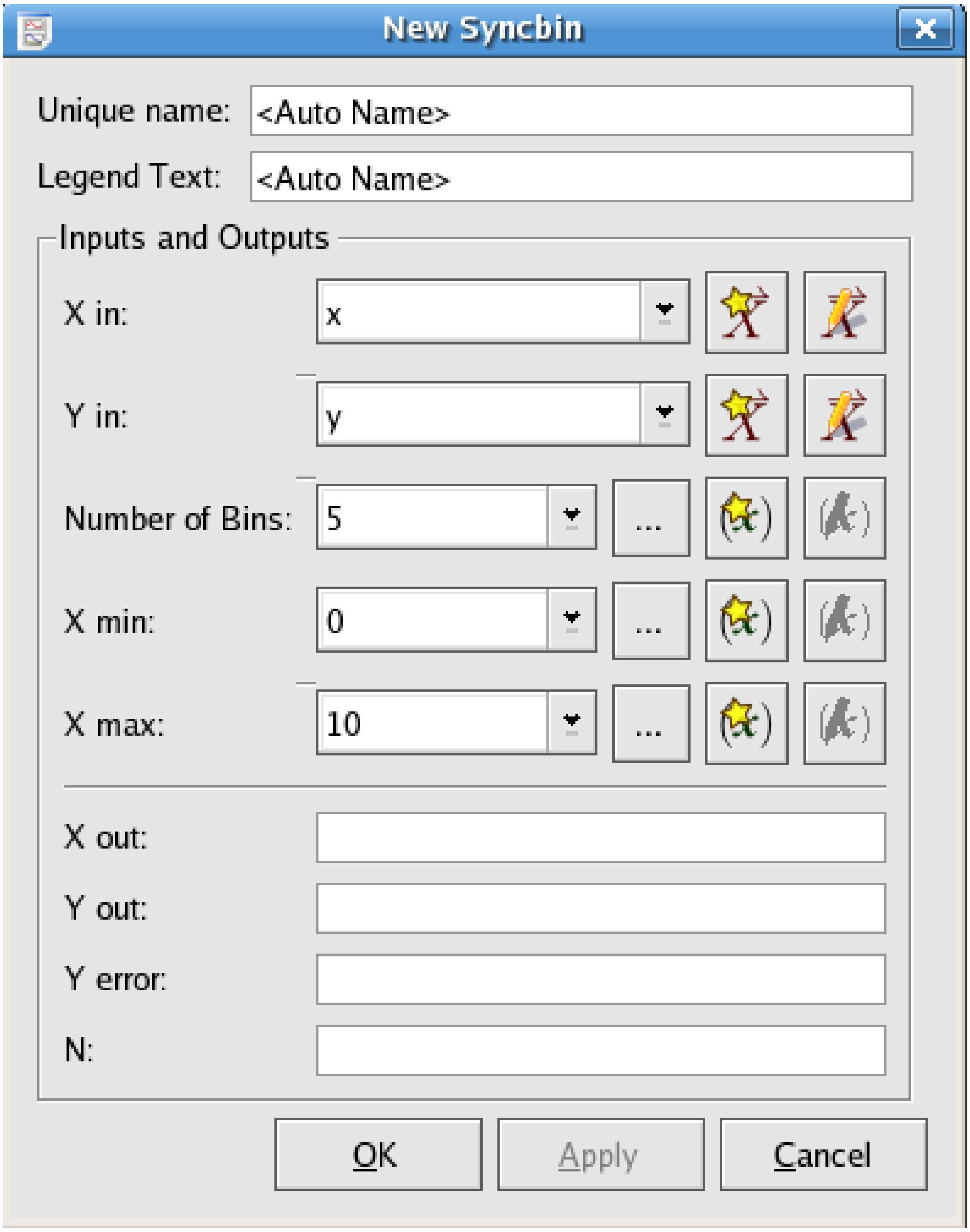
Absolute deviation (scalar) The absolute deviation of the data set.

Skewness (scalar) The skewness of the data set.

Kurtosis (scalar) The kurtosis of the data set.

8.8.11 Syncbin

This plugin groups y values of a data set into bins defined by x values.



Unique name: <Auto Name>

Legend Text: <Auto Name>

X in: x

Y in: y

Number of Bins: 5

X min: 0

X max: 10

X out:

Y out:

Y error:

N:

8.8.11.1 Inputs

X in (vector) The X values of a set of data points

Y in (vector) The Y values of a set of data points

Number of Bins (scalar) Specify the number of bins used to group the data points

X min/X max (scalar) Specify min/max X values of a data set to indicate the range of data points needed to be grouped.

8.8.11.2 Outputs

X out (vector) The X values after binning. The X out vector is composed of the median values of the X elements in each bin

Y out (vector) The Y values after binning. The Y out vector is composed of the mean values of the Y elements in each bin

Y error (vector) This vector is composed of the standard deviations of the Y values in each bin.

N (vector) The N vector is composed of the number of data points in each bin.

8.9 Using Plugins through KstScript

8.9.1 Overview

Plugins can also be generated and edited through KstScript. See details in the Plugin class in the Working With KstScript chapter.

To create a new plugin, using the following syntax:

```
plugin = new Plugin(Kst.pluginManager.modules["Plugin Name"])
//the "Plugin Name" should be typed exactly as shown in the Create Plugin tab in the Data  ↩
    Manager.
```

To set input values of this plugin

```
plugin.setInput(index of input, value of input);
```

You must validate the plugin before you can get its outputs. To validate the plugin, use the following method:

```
plugin.validate();
```

The output values stored in Kst as an objectcollection. To get an output of this plugin, you need to specify its index:

```
v = plugin.outputs[index of output];
```

To know the index of inputs and outputs, refer to their appearing orders in dialogs of plugins UI.

See two examples below of using plugins through KstScript

8.9.2 Examples

Example 8.1 KstScript of Statistics Plugin

```
p = new Plugin(Kst.pluginManager.modules["Statistics"]);//create a new statistics plugin
p.tagName = "Stat";       // this will be the plugin name in the data manager
p.setInput(0, "x");      //x is the name of a vector whose various
                         //statistical values are needed
p.validate();           //validate the plugin
s = new Scalar();
s = p.outputs[0];       //to get the first output of this plugin, i.e the mean of x
s.value         //the value of s will be printed on the screen, e.g. -4.12232621329268
```

Example 8.2 KstScript Function of Using Interpolation Plugins

All Kst interpolation plugins have the same inputs and outputs; therefore, instead of writing lots of similar code to set inputs and get outputs for different interpolation schemes, we can write a kstScript function to do that.

The following is a KstScript function which needs 3 vectors: x, y, ix, and a string as inputs. The function will use the string to specify an interpolation plugin, and apply the specified interpolation scheme to x, y vectors in order to get the interpolated y values for ix vector. Then the function will plot y vs x, and interpolated y vs ix.

```
//intplFuction.js
function intpl(x,y,ix,intpl){
//x is a vector used as X Array in interpolation plugins
//y is a vector used as Y Array
//ix is a vector used as X’ Array
//intpl is a string used to specify the name
//of a specific interpolation plugin, such as "Interpolation akima spline".
//construct a new interpolation plugin
 pl = new Plugin(Kst.pluginManager.modules[intpl]);
 pl.tagName = intpl+"plugin";
//set the Inputs of the plugin
 pl.setInput(0,x);
 pl.setInput(1,y);
 pl.setInput(2,ix);
 pl.validate(); //validated the plugin

iy = pl.outputs[0]; //gets the output vector from this plugin
iy.tagName = "interpolated_y"; //call it interpolated_y vector in the data_manager
//open a new window to plot x vs y and ix vs interpolated_y;
w=new Window();
p=new Plot(w);
c1 = new Curve(x,y);
c1.tagName = "y";
c2 = new Curve(ix,iy);
c2.tagName = "interpolated_y";
p.curves.append(c1);
p.curves.append(c2);
p.xAxis.label = "X";
p.yAxis.label = "Y";
//edit the settings of the curves, such as line style, point style, etc
//in order to make the two curves more distinguishable
c1.hasPoints = true;
c1.pointStyle = 3;
c1.lineStyle = 1;
c1.lineWidth = 2;
c2.hasPoints = true;
c2.pointStyle = 0;
c2.lineWidth = 1;
//add an legend
l = new Legend(w);
l.fontSize = 5;
l.addCurve(c1);
l.addCurve(c2);
   }
```

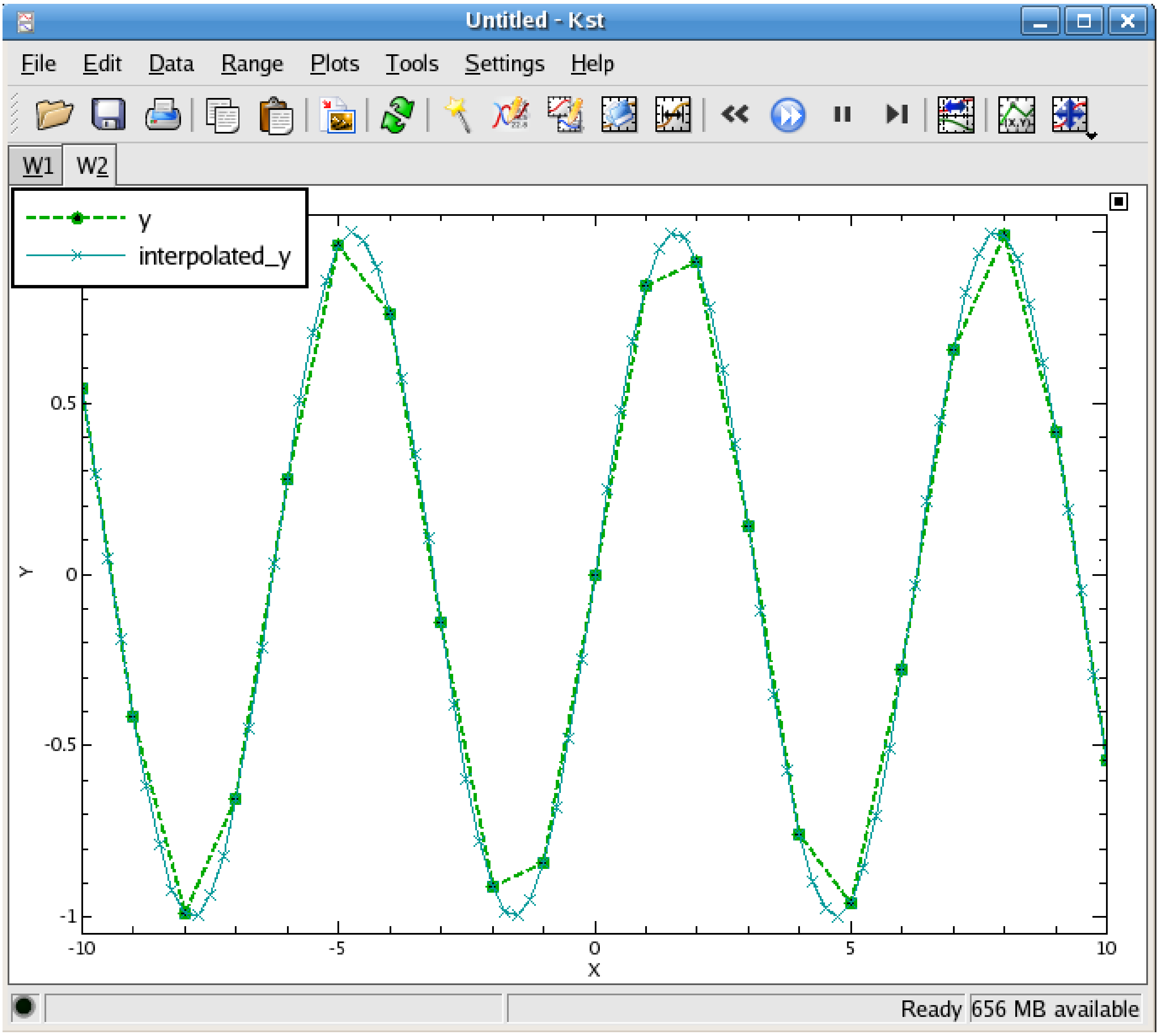
Let us see an example of using the above function. First, save the above script in a file called `intplFunction.js` in your home folder, and open a new text file to copy the following lines, then save this file as `intplTest.js` in your home folder.

```
//intplTest.js
//construct 2 vectors x and y, with y = sin(x) and x = {-10,-9,-8...0...8,9,10}
x = new Vector();
x.tagName = "x";
y = new Vector();
```

```
y.tagName = "y";
x.resize(21);
y.resize(21);
var k=-10;
for(var i=0; i<21; i++){
x[i]=k;
y[i]=Math.sin(x[i]);
k++;}
//construct the ix vector; ix={-10,-9.75,-9.25,-9,...0...9,9.25,9,75,10}
//The y values of ix will be obtained by interpolation.
ix=new Vector();
ix.tagName = "ix";
ix.resize(81);
var k=-10;
for(var i=0; i<81; i++){
ix[i]=k;
k = k+0.25;}
//load intpl function script
loadScript('/home/vyiwen/intplFunction.js');
//call intpl function
intpl(x,y,ix,"Interpolation cubic spline")
```

Open Kst, and load `intplTest.js` in the Java Console like this: **loadScript('/home/vyiwen/intplTest.js');**

After loading the script, you should see the plots below:


Untitled - Kst
File
Edit
Data
Range
Plots
Tools
Settings
Help
W1
W2
y
interpolated_y
0.5
0
-0.5
-1
Y
-10
-5
0
5
10
X
Ready
656 MB available

Chapter 9

Licensing

This documentation is licensed under the terms of the GNU Free Documentation License.

Appendix A

Command Line Usage and Examples

A typical use of kst is from the command line to make X-Y plots or Z images from data files. kst can read ascii data, readdata/dirfile compatible binary files, and optionally (with external libraries) FITS images, PIOLib files, netCDF files, and HEALPix FITS files.

A.0.2.0.0.1 Kst Command Line Synopsis

Name

kst – A plotting and data viewing program

Synopsis

The options are:

`kst` [Qt-options] [KDE-options] [options] [*file*...]

Options

file...] A .kst file, or one or more data files. Supported formats are ASCII columns, BOOMERANG frame files, BLAST dirfile files, and other optionally supported types. A .kst files stores all options that can be set by other flags. The following flags can be used to override the options set in the .kst file: `-F datafile`, `-n NS`, `-s NS`, `-f F0`, `-a`. The rest can not be overridden. If an override flag is given, it is applied to all vectors in the plot.

ASCII data from stdin can be plotted by including "stdin" in the list `[file...]`.

-y Y The source for the Y axis data. For ASCII files, this is the column. For binary files, this is the field name. To plot multiple curves, this may be specified multiple times. The same source file, sample ranges, and X axis data are assumed.

--ye equation The values for the Y axis data are calculated from the equation specified. Multiple equations are allowed.

--xe X Use equation `x0:x1:n` to specify the X vector (INDEX). x0:x1 specifies the range of the vector, and n specifies the the number of values in X vector.

e.g. -10:10:21 creates this static vector: X = {-10,-9,-8,...0...8,9,10}

-E text Pass argument to extension. `text` is of format `extensionname:argumentlist`

e.g. **kst -E js:"loadScript('intplTest.js')"**

-e E Error bars for Y axis data. For ASCII file, refer to the column holding the data. For binary files use the field name. If multiple `-y Y` options are given, this may also be used multiple times. Individual Y entries are associated in order with multiple E entries.

-x X The source for the X axis data. For ASCII files, this is the column. For readata files, this is the field name. If this option is not specified, the row number in the file is used. This option can only be given once.

-z Z The source for the Z matrix data (which gets displayed as an image). For ASCII files, this is the column containing the z data. For other optional formats (HEALPix, FITS image, etc), this is the name of the desired matrix field, as displayed in the matrix dialog. Some matrix-compatible datasources also allow using an alias which is the number of the desired matrix field. To plot multiple matrices, this may be specified multiple times.

-p P The source for power spectra. For ASCII files, this is the column. For binary files, this is the field name. To plot power spectra of multiple curves, this may be specified multiple times. The same source file, sample ranges and fft lengths are used for all Power Spectra requested from the command line.

-l P The length of the FFT used in power specra estimation is 2ˆP.

-r f Specify sample rate: `f` for power spectrum

--ru U Specify the units `U` for psd sample rate

--yu U Specify the units `U` for psd sample rate

-h H The source for histograms. For ASCII files, this is the column. For binary files, this is the field name. Multiple histograms can be defined from the command line.

-m NC Used when multiple curves have been defined. Rather than plotting all curves in the same plot, plot each in its own plot,

-f F0 The starting frame number (for readdata files) or row (for ASCII files) to read.

-n The number of frames (for readdata files) or rows (for ASCII files) to read.

-s NS The number of frames or rows to skip each read. This is useful when working with very large data files, to speed up response and conserve memory, in the case that the data are slowly varying.

-g Provide a legend box

-w file Display the data wizard

--nq Bypass the quickstart dialog

-a Used in with the `-s NS`: rather than skipping each NS frames, average them. This provides a combination of very basic high pass filtering, and memory conservation.

-F Datafile Override the file to read the data from for all vectors listed in the .kst file. Can only be used in conjunction with a kst file.

--print psfile Rather than displaying the plot, export the image to a postscript file and exit. *BUG note: even though no windows are opened on screen, access to the X server is still required.*

--png pngfile Rather than displaying the plot, export the image to a png file of dimensions 640x480 and exit. *BUG note: even though no windows are opened on screen, access to the X server is still required.*

Command Line Examples

Several examples of typical use follow.

To plot column 1 a file (tmp.dat) of ASCII data:

```


To plot column 2, 3, and 4 vs. column 1 of an ASCII file, in 1 plot:

\begin{lstlisting}[firstnumber=1,escapeinside={:>,]<:kst -x 1 -y 2 -y 3 -y 4 tmp.dat:>
```

To plot column 2, 3, and 4 vs. column 1 of an ASCII file, in 3 plots, arranged in 1 column:

```
kst -x 1 -y 2 -y 3 -y 4 -m 1 tmp.dat
```

To plot 500 rows of column 2 of an ASCII file, starting at line 100:

```
kst -y 2 -f 100 -n 500 tmp.dat
```

To plot the first 100,000 rows of column 2 of an ASCII file, averaging every 100 rows:

```
kst -y 2 -f 0 -n 100000 -s 100 -a tmp.dat
```

Using command to plot a spectrum from a data field of an ASCII data file with sample rate = 10HZ and FFT length = 2ˆ14

```
 kst -p 2 /usr/share/apps/kst/tutorial/gyrodata.dat -r 10 -l 14
```

(Using optional HEALPix Data Source) To plot the first 3 maps in a HEALPix FITS file in a 2x2 grid:

```
kst -z 1 -z 2 -z 3 -m 2 healpix_example_sm.fits
```

(Using optional HEALPix Data Source) To plot a map with a specific field name in a HEALPix FITS file:

```
kst -z "1 - TEMPERATURE (Kelvin)" healpix_example_sm.fits
```

Appendix B

Creating Additional Plugins

A plugin can be used to extend the functionality of Kst, without the need to recompile Kst. A properly configured plugin, once installed, will be automatically detected by Kst when it is next run and the associated functionality made available to the user.

There are three flavors of plugin and the one you select will depend on what you are trying to achieve. The ‘C’ plugin should now be considered as deprecated by the Basic plugin.

- Basic Plugin - written in C++ and derived from `KstBasicPlugin`.
- Data Object Plugin - written in C++ and derived from `KstDataObject`.
- ‘C’ Plugin - written in C.

B.1 Creating a Basic Plugin

The purpose of a KstBasicPlugin plugin is to provide an implementation of the virtual class `KstDataObject` via the abstract class `KstBasicPlugin`. Plugin writers need to provide a class that inherits `KstBasicPlugin` and a `.desktop` file.

Here is an example of the `.desktop` file named `kstobject_myplugin.desktop`:

```
[Desktop Entry]
Encoding=UTF-8
Type=Service
ServiceTypes=Kst Data Object
X-KDE-ModuleType=Plugin
X-KDE-Library=kstobject_fooplugin
X-Kst-Plugin-Author=Your Name
X-Kst-Plugin-Version=0.1
Name=Foo
Comment=A plugin that provides Foo algorithm.
```

Your C++ class should inherit `KstBasicPlugin` and provide implementations of the pure virtual methods found in `KstBasicPlugin`:

```
//The implementation of the algorithm the plugin provides.
//Operates on the inputVectors, inputScalars, and inputStrings
//to produce the outputVectors, outputScalars, and outputStrings.
virtual bool algorithm() = 0;

//String lists of the names of the expected inputs.
virtual QStringList inputVectorList() const = 0;
virtual QStringList inputScalarList() const = 0;
virtual QStringList inputStringList() const = 0;
```

```
    //String lists of the names of the expected outputs.
    virtual QStringList outputVectorList() const = 0;
    virtual QStringList outputScalarList() const = 0;
    virtual QStringList outputStringList() const = 0;
```

Here is an example of a plugins header file:

```
#ifndef FOOPLUGIN_H
#define FOOPLUGIN_H

#include <kstbasicplugin.h>

class FooPlugin : public KstBasicPlugin {
  Q_OBJECT
  public:
    FooPlugin(QObject *parent, const char *name, const QStringList &args);
    virtual ~FooPlugin();

    virtual bool algorithm();

    virtual QStringList inputVectorList() const;
    virtual QStringList inputScalarList() const;
    virtual QStringList inputStringList() const;
    virtual QStringList outputVectorList() const;
    virtual QStringList outputScalarList() const;
    virtual QStringList outputStringList() const;
};
```

And here is an example of a plugin’s `.cpp` file:

```
#include "fooplugin.h"

#include <kgenericfactory.h>

static const QString& VECTOR_IN = KGlobal::staticQString("Vector In");
static const QString& SCALAR_IN = KGlobal::staticQString("Scalar In");
static const QString& STRING_IN = KGlobal::staticQString("String In");
static const QString& VECTOR_OUT = KGlobal::staticQString("Vector Out");
static const QString& SCALAR_OUT = KGlobal::staticQString("Scalar Out");
static const QString& STRING_OUT = KGlobal::staticQString("String Out");

K_EXPORT_COMPONENT_FACTORY( kstobject_fooplugin,
    KGenericFactory<FooPlugin>( "kstobject_fooplugin" ) )

FooPlugin::FooPlugin( QObject */*parent*/, const char */*name*/, const QStringList &/*args ↩
    */ )
    : KstBasicPlugin() {
}


FooPlugin::~FooPlugin() {
}


bool FooPlugin::algorithm() {
  //Do something...
  return true;
}


QStringList FooPlugin::inputVectorList() const {
  return QStringList( VECTOR_IN );
}


QStringList FooPlugin::inputScalarList() const {
  return QStringList( SCALAR_IN );
}


QStringList FooPlugin::inputStringList() const {
  return QStringList( STRING_IN );
}


QStringList FooPlugin::outputVectorList() const {
  return QStringList( VECTOR_OUT );
}


QStringList FooPlugin::outputScalarList() const {
  return QStringList( SCALAR_OUT );
}


QStringList FooPlugin::outputStringList() const {
  return QStringList( STRING_OUT );
}

#include "fooplugin.moc"
```

B.1.1 Default Scalar Values

Default values for input scalars can be specified in the constructor of the basic plugin. In the above example the constructor could be modified as follows to set a default value of 1.0 for the input scalar as follows:

```
FooPlugin::FooPlugin( QObject */*parent*/, const char */*name*/, const QStringList &/*args ←
    */ )
    : KstBasicPlugin() {
  _inputScalarDefaults.insert(SCALAR_IN, 1.0);
}
```

B.1.2

The KstBasicPlugin takes care of providing almost everything, including the configuration widget for your plugin. It does not provide the actual algorithm or the names of the inputs/outputs.

See the bin plugin for an example implementation.

B.2 Creating a Data Object Plugin

The purpose of a KstDataObject plugin is to provide an implementation of the virtual class KstDataObject. Plugin writers need to provide a class that inherits KstDataObject directly. You will also need to provide a .desktop file and a configuration widget for this plugin. The benefit of this type of plugin over KstBasicPlugin plugins is the degree of control you'll have over the implementation. Most plugins should probably use KstBasicPlugin, but if you need a custom dialog for instance, this is the type of plugin for you.

Here is an example of the .desktop file named kstobject_myplugin.desktop:

```
    [Desktop Entry]
    Encoding=UTF-8
    Type=Service
    ServiceTypes=Kst Data Object
    X-KDE-ModuleType=Plugin
    X-KDE-Library=kstobject_fooplugin
    X-Kst-Plugin-Author=Your Name
    X-Kst-Plugin-Version=0.1
    Name=Foo
    Comment=A plugin that provides Foo algorithm.
```

You C++ class should inherit `KstDataObject` and provide implementations of the pure virtual methods found in `KstDataObject`:

```
    virtual UpdateType update(int updateCounter = -1) = 0;
    virtual QString propertyString() const = 0;
    virtual void showNewDialog() = 0;
    virtual void showEditDialog() = 0;
```

There is also a number of regular virtual methods that can be reimplemented as needed.

See the Cross Power Spectrum plugin for an example implementation and the `KstDataObject` header file for API declarations.

B.3 Creating a ‘C’ Plugin

A Kst plugin consists of two files — an XML file and a shared object file.

B.3.1 The XML File

The XML file provide information about the plugin and describes its inputs and outputs. The following is an example of an XML file for a Kst plugin:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Module SYSTEM "file:/Repository/Level2/Soft/ProC/moduledef.dtd">

<module>

<intro>
<modulename name="testplugin"/>    <!-- The name of the module -->
<author name="Rick Chern"/> <!-- The name of the author -->
<description text="A test plugin for me"/> <!-- A description of the module -->
<version minor="1" major="0"/>  <!-- The version number of the module -->
<state devstate="release"/>     <!-- The development state of the module (optional)-->
</intro>

<interface>

<!--inputs here-->
<input>
<table type="float" name="Input Vector 1" descr="The first input vector" />
</input>

<input>
<float name="Input Scalar 1" descr="The first input scalar" />
</input>

<!--outputs here-->
<output>
```

```
<table type="float" name="Output Vector 1" descr="The first output vector" />
</output>

<output>
<float name="Output Scalar 1" descr="The first output scalar" />
</output>

</interface>

</module>
```

Generally, you can use the example above as a template and modify sections to fit your needs. As can be seen from the example, the XML file consists of one `module` node. The `module` node has an `intro` node and an `interface` node. You should modify the `intro` node appropriately using the comments in the above XML file as guides. It is important that `modulename` has the `name` attribute set to the same name that your shared object file will use.

The `interface` node describes the actual inputs and outputs of the plugin. Each input is described by an `input` node, and each output is described by an `output` node. Each input or output should have either a `table` or a `float` node as a child. The `type` attribute of a `table` must be set to `"float"`. Note that the order of the inputs and outputs matters — the order is used to determine the index values for each input and output array of the object file, and is the same order used to display the input and output fields in the Kst plugin interface.

Once you have completed the XML file, save it as `[modulename].xml`, where *`[modulename]`* is the value of the `name` attribute of the `modulename` node.

B.3.2 The Shared Object File

The shared object file contains the actual functionality of the plugin. In other words, it determines how to derive the outputs from the given inputs. The following are the requirements for the shared object file:

- The object must export the following C linkage symbol:

  ```
  int symbol(const double *const inArrays[],
             const int inArrayLens[],
             const double inScalars[],
             double *outArrays[],
             int outArrayLens[],
             double outScalars[])
  ```

  where *`symbol`* must be the value of the `name` attribute of the `modulename` node in the XML file. This is the only function that will be called by Kst (although you may have other functions). The following describes each argument of this function:

  `const double *const inArrays[]` The array of input arrays. Each input array corresponds to an input vector. The arrays are in the same order as the vectors are listed in the XML file, so `inArrays[0]` is the array representing the first input vector, `inArrays[1]` is the array representing the second input vector, and so on.

  `const int inArraysLens[]` The array containing array lengths for each input array. The lengths are stored in the same order as the arrays in `inArrays[]` (e.g. `inArrayLens[0]` is the length of `inArrays[0]`).

  `const double inScalars[]` The array of input scalars. The scalars are stored in the same order as they are listed in the XML file.

  `double *outArrays[]` The array of output arrays. Each output array corresponds to an output vector, and the arrays should be in the same order as the output vectors are listed in the XML file.

  `int outArrayLens[]` The array that should contain lengths of the output arrays. The lengths should be stored in the same order as the arrays in `outArrays[]`.

  `double outScalars[]` The array of output scalars. The scalars should be in the same order they are listed in the XML file.

- The function must return `0` if it executed successfully, and `-1` otherwise.
- The code for the object file must handle unexpected inputs, such as empty input arrays (in most cases a return value of `-1` would be sufficient when such situations are encountered).
- The number and type of outputs must be exactly as specified by the XML file.
- You will probably need to resize the arrays in `outArrays[]`. To do so, use the `realloc()` function. E.g.,

  ```
  outArrays[0]=(double*)realloc(outArrays[0], 5*sizeof(double));
  ```

  will allocate space for 5 doubles for `outArrays[0]`. Do not use any memory allocator other than `realloc()`.
- The input arguments must remain constant. Do not cast them to non-constant types.

The following is an example of the shared object file source code for a simple plugin:

```
#include <stdlib.h>

extern "C" int testplugin(const double *const inArrays[], const int inArrayLens[],
                const double is[],
                double *outArrays[], int outArrayLens[],
                double outScalars[]);

int testplugin(const double *const inArrays[], const int inArrayLens[],
                const double is[],
                double *outArrays[], int outArrayLens[],
                double outScalars[])

//Accept 1 vector and 1 scalar. Multiply all elements of the vector by the
//scalar, and output the resulting vector. Also output the original scalar.
{
  //Set the outputs
  outArrayLens[0]=inArrayLens[0];

  //resize the output arrays
  outArrays[0]=(double*)realloc(outArrays[0], inArrayLens[0]*sizeof(double));

  //multiply each element of the input vector
  //by the scalar
  for (int i=0; i<inArrayLens[0]; i++)
  {
    outArrays[0][i]=inArrays[0][i] * is[0];
  }

  //now set the output scalar
  outScalars[0]=is[0];

  return 0;
}
```

B.3.3 Compiling the Plugin

If you are using **gcc** to compile your plugin, simply compile the object file:

```
cc -Wall -c -o myplugin.o myplugin.c -fPIC -DPIC
```

and then create the shared library:

```
ld -o myplugin.so -shared myplugin.o
```

The resulting `*.so` file and `*.xml` file must be put in the same directory. When you use Kst's Plugin Manager to load the XML file, it will automatically look for the shared object file in the same directory.

B.4 Creating Linear Fit Plugins

To create a linear fit plugin, you could implement your own fitting algorithms and output the appropriate vectors. However, Kst already comes with header files that make it easy for you to implement linear least-squares fit plugins by just providing a few functions. This section will describe how to take advantage of these files.

B.4.1 Header Files

Two header files are provided for performing linear fits, `linear.h` (for unweighted linear fits) and `linear_weighted.h` (for weighted linear fits). They are both located under `kst/plugins/fits/` in the Kst source tarball. To use these files, include only one of them in the source code for your plugin:

```
#include <../linear.h>
```

or

```
#include <../linear_weighted.h>
```

(by convention, we will place the source code for the plugin one directory below where the header files are).

B.4.2 Implementing Required Functions

Given a general linear model:

$$y = Xc$$

where `y` is a vector of `n` observations, `X` is an `n` by `p` matrix of predictor variables, and `c` is the vector of `p` best-fit parameters that are to be estimated, the header files provide functions for estimating `c` for a given `y` and `X`. To provide `X`, the following function needs to be implemented in the source code for the plugin:

```
double calculate_matrix_entry( double dX, int iPos )
```

This function should return the value of the entry in column `iPos` of the matrix of predictor variables, using `x` value `dX`. This function will be called by linear.h or linear_weighted.h. The implementation of this function depends on the model you wish to use for the fit, and is unique to each linear fit plugin. For example, to fit to a polynomial model, `calculate_matrix_entry` could be implemented as follows:

```
double calculate_matrix_entry( double dX, int iPos ) {
  double dY;
  dY = pow( dX, (double)iPos );
  return dY;
}
```

B.4.3 Calling the Fitting Functions

Once the appropriate header file has been included and `calculate_matrix_entry` has been implemented, call the appropriate fitting function included from the header file:

```
kstfit_linear_unweighted( inArrays, inArrayLens,
                          outArrays, outArrayLens,
                          outScalars, iNumParams );
```

or

```
kstfit_linear_weighted( inArrays, inArrayLens,
                        outArrays, outArrayLens,
                        outScalars, iNumParams );
```

Each function will return `0` on success, or `-1` on error, so it is a good idea to set the return value of the exported C function to be equal to the return value of the fitting function. To maintain simplicity, the code for the plugin can simply pass the arguments given to the exported C function to the fitting function. Note, however, that inArrays must be structured as follows:

- `inArrays[0]` must contain the array of x coordinates of the data points
- `inArrays[1]` must contain the array of y coordinates of the data points
- `inArrays[2]` only exists if `kstfit_linear_weighted` is being called, and must contain the array of weights to use for the fit.

The easiest way to ensure that inArrays is structured correctly is to specify the correct order of input vectors in the XML file for the plugin.

`iNumParams` is the number of parameters in the fitting model used, which should be equal to the number of columns in the matrix `X` of predictor variables. `iNumParams` must be set correctly before the fitting function is called.

After `kstfit_linear_unweighted` or `kstfit_linear_weighted` is called, `outArrays` and `outScalars` will be set as follows:

- `outArrays[0]` will contain the array of fitted y values.
- `outArrays[1]` will contain the array of residuals.
- `outArrays[2]` will contain the array of best-fit parameters that were estimated.
- `outArrays[3]` will contain the covariance matrix, returned row after row in an array.
- `outScalars[0]` will contain chi^2/nu, where chi^2 is the weighted sum of squares of the residuals, and nu is the degrees of freedom.

`outArrayLens` will be correctly set to indicate the length of each output array.

Ensure that the specified outputs in the XML file match those that the exported C function returns (which in most cases will simply be the outputs returned by the fitting function).

B.4.4 Example

The following is an example of the source code for a linear fit plugin.

```
/*
 *  Polynomial fitting plugin for KST.
 *  Copyright 2004, The University of British Columbia
 *  Released under the terms of the GPL.
 */

#include "../linear.h"

double calculate_matrix_entry( double dX, int iPos ) {
  double dY;

  dY = pow( dX, (double)iPos );

  return dY;
}
```

```
extern "C" int kstfit_polynomial_unweighted(
  const double *const inArrays[],
  const int inArrayLens[],
  const double inScalars[],
  double *outArrays[], int outArrayLens[],
  double outScalars[]);

int kstfit_polynomial_unweighted(
  const double *const inArrays[],
  const int inArrayLens[],
  const double inScalars[],
  double *outArrays[], int outArrayLens[],
  double outScalars[])
{
  int iRetVal = -1;
  int iNumParams;

  iNumParams = 1 + (int)floor( inScalars[0] );
  if( iNumParams > 0 ) {
    iRetVal = kstfit_linear_unweighted( inArrays, inArrayLens,
                                        outArrays, outArrayLens,
                                        outScalars, iNumParams );
  }

  return iRetVal;
}
```

B.5 Creating Non-linear Fit Plugins

Kst provides header files designed to simplify the creation of non-linear least-squares fit plugins. The following sections detail the use of the header files.

B.5.1 Header Files and Definitions

The non-linear fit header files are located in `kst/plugins/fits_nonlinear/` of the Kst source tarball. The files are named `non_linear.h` and `non_linear_weighted.h` for unweighted and weighted fits, respectively. To use these files, include only one of them in the source code for your plugin:

```
#include <../non_linear.h>
```

or

```
#include <../non_linear_weighted.h>
```

(by convention, we will place the source code for the plugin one directory below where the header files are).

As non-linear fitting is an iterative process, you must also define the maximum number of iterations that should be performed. The non-linear fitting algorithm will stop when at least one of the following conditions is true:

- The maximum number of iterations has been reached.
- A precision of 10^{-4} has been reached.

In addition, you need to define the number of parameters in the model, as it is not passed to the fitting function explicitly. To define these two values, include the following at the top of your source code:

```
#define NUM_PARAMS [num1]
#define MAX_NUM_ITERATIONS [num2]
```

replacing `[num1]` with the number of parameters in the model, and `[num2]` with the maximum number of iterations to perform.

B.5.2 Implementing Required Functions

To use the header files for non-linear fits, you must provide the function to use as the model, the partial derivatives of the function with respect to each parameter, and initial estimates of the best-fit parameters. To do this, three functions must be implemented. These functions are described below.

double(function_calculate(double *dX*, , double* *pdParameters*)) This function calculates the y value of the fitting model for a given x value dX, using the supplied array of parameters pdParameters. The order of parameters in pdParameters is arbitrary, but should be consistent with the other two implemented functions. For example, for an exponential model, function_calculate could be implemented as follows:

```
double function_calculate( double dX, double* pdParameters ) {
  double dScale  = pdParameters[0];
  double dLambda = pdParameters[1];
  double dOffset = pdParameters[2];
  double dY;

  dY  = ( dScale * exp( -dLambda * dX ) ) + dOffset;

  return dY;
}
```

void(function_derivative(double *dX*, , double* *pdParameters*, , double* *pdDerivatives*)) This function calculates the partial derivatives of the model function for a give value of x dX. The partial derivatives should be returned in pdDerivatives. The order of the partial derivatives in the array pdDerivatives should correspond to the order of the parameters in pdParameters (i.e. if pdParameters[0] contains the parameter lambda for an exponential model, pdDerivatives[0] should contain the derivative of the model with respect to lambda).

void(function_initial_estimate(const double* *pdX*, , const double* *pdY*, , int *iLength*, , double* *pdParameterEstimates*)) This function provides an initial estimate of the best-fit parameters to the fitting function. The array of x values and y values of the data points are provided in pdX and pdY respectively, and the number of data points is provided by iLength. You can use any or none of these parameters at your discretion to calculate the initial estimate. The function should put the calculated initial estimates in pdParameterEstimates, with the order of the estimates corresponding to the order of the parameters in pdParameters of function_calculate and function_derivative. Keep in mind that the initial estimate is important in determining whether or not the fitting function converges to a solution.

B.5.3 Calling the Fitting Functions

Once all the required functions have been implemented, the fitting function from the included header file can be called:

```
kstfit_nonlinear( inArrays, inArrayLens,
      inScalars, outArrays,
      outArrayLens, outScalars );
```

or

```
kstfit_nonlinear_weighted( inArrays, inArrayLens,
                           inScalars, outArrays,
                           outArrayLens, outScalars );
```

depending on whether you are implementing a non-weighted fit or a weighted fit.

The function will return 0 on success, or -1 on error, so it is simplest to set the return value of the exported C function to be equal to the return value of the fitting function. To maintain simplicity, the code for the plugin can simply pass the arguments given to the exported C function to the fitting function. Note, however, that inArrays must be structured as follows:

- inArrays[0] must contain the array of x coordinates of the data points
- inArrays[1] must contain the array of y coordinates of the data points

- inArrays[2] only exists if kstfit_linear_weighted is being called, and must contain the array of weights to use for the fit.

The easiest way to ensure that inArrays is structured correctly is to specify the correct order of input vectors in the XML file for the plugin.

After kstfit_linear_unweighted or kstfit_linear_weighted is called, outArrays and outScalars will be set as follows:

- outArrays[0] will contain the array of fitted y values.
- outArrays[1] will contain the array of residuals.
- outArrays[2] will contain the array of best-fit parameters that were estimated.
- outArrays[3] will contain the covariance matrix, returned row after row in an array.
- outScalars[0] will contain chi^2/nu, where chi^2 is the weighted sum of squares of the residuals, and nu is the degrees of freedom.

outArrayLens will be correctly set to indicate the length of each output array.

Ensure that the specified outputs in the XML file match those that the exported C function returns (which in most cases will simply be the outputs returned by the fitting function).

B.5.4 Example

The following is an example of a non-linear fit plugin that performs a fit to an exponential model.

```
/*
 *  Exponential fit plugin for KST.
 *  Copyright 2004, The University of British Columbia
 *  Released under the terms of the GPL.
 */

#define NUM_PARAMS 3
#define MAX_NUM_ITERATIONS 500

#include "../non_linear.h"

void function_initial_estimate( const double* pdX, const double* pdY,
                                int iLength, double* pdParameterEstimates ) {
  KST_UNUSED( pdX )
  KST_UNUSED( pdY )
  KST_UNUSED( iLength )

  pdParameterEstimates[0] =  1.0;
  pdParameterEstimates[1] =  0.0;
  pdParameterEstimates[2] =  0.0;
}

double function_calculate( double dX, double* pdParameters ) {
  double dScale  = pdParameters[0];
  double dLambda = pdParameters[1];
  double dOffset = pdParameters[2];
  double dY;

  dY  = ( dScale * exp( -dLambda * dX ) ) + dOffset;

  return dY;
}
```

```
void function_derivative( double dX, double* pdParameters, double* pdDerivatives ) {
  double dScale  = pdParameters[0];
  double dLambda = pdParameters[1];
  double dExp;
  double ddScale;
  double ddLambda;
  double ddOffset;

  dExp     = exp( -dLambda * dX );
  ddScale  = dExp;
  ddLambda = -dX * dScale * dExp;
  ddOffset = 1.0;

  pdDerivatives[0] = ddScale;
  pdDerivatives[1] = ddLambda;
  pdDerivatives[2] = ddOffset;
}

extern "C" int kstfit_exponential(const double *const inArrays[], const int inArrayLens[],
    const double inScalars[],
    double *outArrays[], int outArrayLens[],
    double outScalars[]);

int kstfit_exponential(const double *const inArrays[], const int inArrayLens[],
    const double inScalars[],
    double *outArrays[], int outArrayLens[],
    double outScalars[])
{
  return kstfit_nonlinear( inArrays, inArrayLens,
                           inScalars, outArrays,
         outArrayLens, outScalars );
}
```

B.6 Creating Pass Filter Plugins

Kst provides header files to simplify the implementation of pass filter plugins. The use of these header files is described below.

B.6.1 Header Files

The pass filter header file is located in `kst/plugins/pass_filters` of the Kst source tarball. The file is named `filters.h` To use this file, include it in the source code for your plugin:

```
#include <../filters.h>
```

(by convention, we will place the source code for the plugin one directory below where the header files are).

B.6.2 Required Functions

The `filters.h` header file contains a single function that calculates the Fourier transform of a supplied function, applies the supplied filter to the Fourier transform, and then calculates the inverse Fourier transform of the filtered Fourier transform. To supply the filter, the following function needs to be implemented in the source code for your plugin:

`double`(filter_calculate(double *`dFreqValue`*, , const double *`inScalars[]`*))

This function should calculate the filtered amplitude for the frequency `dFreqValue`. `inScalars[]` will contain the unaltered input scalars for the plugin, specified in the XML file. Most likely `inScalars[]` will contain cutoff frequencies or other properties of the filter. For example, to implement a Butterworth high-pass filter, `filter_calculate` could be implemented as follows:

```
double filter_calculate( double dFreqValue, const double inScalars[] ) {
  double dValue;
  if( dFreqValue > 0.0 ) {
    dValue = 1.0 / ( 1.0 +
        pow( inScalars[1] / dFreqValue, 2.0 * (double)inScalars[0] ) );
  } else {
    dValue = 0.0;
  }
  return dValue;
}
```

B.6.3 Calling the Filter Function

Once the required `filter_calculate` has been implemented, the filter function from the header file can be called:

```
kst_pass_filter( inArrays,
                 inArrayLens,
                 inScalars,
                 outArrays,
                 outArrayLens,
                 outScalars );
```

The arguments supplied to the exported C function can usually be passed to `kst_pass_filter` without modification. However, there are a few restrictions on the arguments:

- `inArrays[0]` must contain the array of data to filter.
- `inScalars` should contain the filter-specific parameters to be used by the `filter_calculate` function.

After the function call, `outArrays[0]` will contain the filtered array of data, and `outArrayLens` will be set appropriately. The `kst_pass_filter` function does not use `outScalars`.

B.6.4 Example

The following is an example of a pass filter plugin that implements the Butterworth high-pass filter.

```
/*
 *  Butterworth low pass filter plugin for KST.
 *  Copyright 2004, The University of British Columbia
 *  Released under the terms of the GPL.
 */

#include <stdlib.h>
#include <math.h>
#include "../filters.h"

extern "C" int butterworth_highpass(const double *const inArrays[], const int inArrayLens ←
    [],
    const double inScalars[],
    double *outArrays[], int outArrayLens[],
    double outScalars[]);

int butterworth_highpass(const double *const inArrays[], const int inArrayLens[],
    const double inScalars[],
    double *outArrays[], int outArrayLens[],
    double outScalars[])
{
  int iReturn;

  iReturn = kst_pass_filter( inArrays,
                             inArrayLens,
                             inScalars,
                             outArrays,
                             outArrayLens,
                             outScalars );

  return iReturn;
}

double filter_calculate( double dFreqValue, const double inScalars[] ) {
  double dValue;

  if( dFreqValue > 0.0 ) {
    dValue = 1.0 / ( 1.0 + pow( inScalars[1] / dFreqValue, 2.0 * (double)inScalars[0] ) );
  } else {
    dValue = 0.0;
  }

  return dValue;
}
```

Appendix C

Supporting Additional File Formats

This section describes how to create additional readers for currently unsupported file formats. If you are not already familiar with data source concepts, please read Data Source Concepts

C.1 Creating Datasource Readers

If you wish to use a file format other than those currently supported, you might choose to write a custom datasource reader.

All Kst datasource readers are regular KDE plugins. Like all KDE plugins, each datasource reader must have a shared object file and a `.desktop` file. Before writing the reader, the library name of the plugin must be decided. This name must be a valid variable name in C, as it will be used in the function names of the shared object. For example, the library name of the reader for ASCII files is ‘ascii’.

Note
KDE plugins are *not* the same as the Kst plugins used to manipulate data. All references to plugins in this section are to KDE plugins.

C.1.1 The Shared Object

A datasource reader should be a subclass of the abstract `KstDataSource` class. Ensure that you include the header file for `KstDataSource` in the source code for your datasource reader:

```
#include <kstdatasource.h>
```

There are certain requirements that a datasource reader must meet. One requirement concerns the presence of exported ‘C’ functions. Other requirements are consequences of the fact that datasource readers inherit from `KstDataSource`. Both sets of requirements, along with suggestions and general explanations, are provided in the following sections.

C.1.1.1 Exported ‘C’ Functions

The exported ‘C’ functions are listed below. In the following examples, all instances of *`libname`* should be replaced with the actual library name of the plugin.

`KstDataSource *`(create_*`libname`*, (KConfig* *`cfg`*, , const QString& *`filename`*, , const QString& *`type`*)) This function should create a new datasource reader of type *`libname`*, where *`libname`* is the library name of the plugin. A pointer of type `KstDataSource` to the new reader should be returned.

[OPTIONAL] `KstDataSource *`(load_*`libname`*, (KConfig* *`cfg`*, , const QString& *`filename`*, , const QString& *`type`*, , cons
This function should create a new datasource reader of type *`libname`*, where *`libname`* is the library name of the plugin. A pointer of type `KstDataSource` to the new reader should be returned.

`int`(understands_*`libname`*, (const QString& *`filename`*)) This function should return an integer from 0 to 100. A value of 0 indicates a complete inability of the datasource to read the file specified by `filename`, while a value of 100 indicates that the datasource is completely confident that it is best able to understand the file specified by `filename`. The function should check the contents of the file for validity, and not simply rely on any filename extensions.

[OPTIONAL] `QStringList`(fieldList_*`libname`*, (KConfig* *`cfg`*, , const QString& *`filename`*, , const QString& *`type`*, , QStri
This function should, if it is able to understand the file specified by `filename`, return a list of the fields that it found within the file, else it should return an empty `QStringList`. The value of `typeSuggestion` should be set to the file type and the value of `complete` should be set to false if no fields were found, else it should be set to true.

[OPTIONAL] `QStringList`(matrixList_*`libname`*, (KConfig* *`cfg`*, , const QString& *`filename`*, , const QString& *`type`*, , QS
This function should, if it is able to understand the file specified by `filename`, return a list of the matrices that it found within the file, else it should return an empty `QStringList`. The value of `typeSuggestion` should be set to the file type and the value of `complete` should be set to false if no matrices were found, else it should be set to true.

[OPTIONAL] `QWidget*`(widget_*`libname`*( )) If the datasource wants to provide a custom configuration dialog it should implement this function and return the newly created `QWidget` derived class here.

`QStringList`(provides_*`libname`*( )) This function should return a `QStringList` of the file types supported by this reader. The strings returned are arbitrary, but should be descriptive of and appropriate for the actual file types.

[OPTIONAL] `bool`(supportsHierarchy_*`libname`*( )) This function should return true if the datasource supports hierarchical field names. The hierarchy is denoted by the '`/`' character.

C.1.1.2 Protected Member Variables

`KstDataSource` contains various protected member variables that the custom datasource reader can use. These variables are described below.

`bool _valid` This variable should be `true` if the custom datasource reader is valid. Most likely the reader will be valid, unless there is an error condition (such as the data file being unreadable by the reader). This variable is used by the `isValid()` function of `KstDataSource`, which is usually not reimplemented in subclasses.

`QStringList _fieldList` This variable should hold a list of the field names in the data source.

`QString _filename` This variable should hold the name of the data file this datasource reader is associated with.

`QString _source` This variable should hold the type name of the source.

C.1.1.3 Virtual Functions

The `KstDataSource` class contains many virtual functions that can be overridden in the custom datasource reader. It is only necessary to override those functions where the default implementation is insufficient. These functions are in the template files `template.h` and `template.cpp`, listed in the Example Templates section. Descriptions of the functions follow.

`TemplateSource(const QString&amp;` (*`filename`*, , const QString&amp; *`type`*)) The constructor for the datasource reader. `filename` is the name of the file to read data from, and `type` is the type of data the file contains. This constructor should most likely invoke the `KstDataSource` constructor in the constructor initializer list, and probably call the `update` function listed below to initialize member variables. In particular, the `_valid` variable should be set appropriately.

`virtual ~TemplateSource()` The destructor for the datasource reader. Any dynamically allocated memory should be freed.

`virtual` (KstObject::UpdateType update(*`int = -1`*)) This function should see if new data exists in the data file since the last time `update` was called, and update the frame count and other member variables appropriately. The function should return `KstObject::UPDATE` if the file contained changes, or `KstObject::NO_CHANGE` otherwise.

`virtual` (int readField(double **`v`***, , const QString &***`field`***, , int* ***`s`***, , int* ***`n`*****))** This function should read `n` frames of data, starting at frame `s` from the field specified by the field name `field`, and return the contents in the array `v`. If `n` is less than 0, the function should instead read 1 sample from the start of frame `s`. The number of samples that were read should be returned.

`virtual` (bool isValidField(const QString &*`field`*) const) This function should return true if the field specified by the field name `field` is a valid field in the current data source, or false if the field is not a valid field.

`virtual` (int readMatrix(KstMatrixData **`data`***, , const QString &***`matrix`***, , int* ***`xStart`***, , int* ***`yStart`***, , int* ***`xNumSteps`***, , int* This function should read the specified sub-range of the matrix, reading `xNumSteps` starting at `xStart`, and `yNumSteps` starting at `yStart` from the matrix specified by the matrix name `matrix`, and return the contents in the `KstMatrixData data`. If `xNumSteps` is less than 0, the function should instead read 1 sample starting from `xStart`. If `yNumSteps` is less than 0, the function should instead read 1 sample starting from `yStart`.

`virtual` (bool isValidMatrix(const QString &*`matrix`*) const) This function should return true if the matrix specified by the matrix name `matrix` is a valid matrix in the current data source, or false if the matrix is not a valid matrix.

`virtual` (bool matrixDimensions(const QString &*`matrix`*, , int* *`xDim`*, , int* *`yDim`*)) This function should return true if the matrix specified by the matrix name `matrix` is a valid matrix in the current data source, or false if the matrix is not a valid matrix. If the matrix is a valid matrix then the values of `xDim` and `yDim` should be set to the x-dimension and y-dimension of the matrix, respectively.

`virtual` (int samplesPerFrame(const QString& *`field`*)) This function should return the ratio of samples per frame for the field specified by the field name `field`. For data sources that do not make use of this concept, the number of samples per frame can be set to `1`.

`virtual int frameCount() const` This function should return the total number of frames in the data source, as of the last time `update` was called.

`virtual` (QString fileType() const) This function should return the file type of the data file currently being used, usually the same as the `type` parameter that was passed to the constructor. Alternatively, it can contain an error message (to indicate, for example, that the file is not valid).

`virtual` (void save(QTextStream &*`ts`*)) This function should save the file description information to `ts`. In most cases the implementation provided by `KstDataSource` should be sufficient.

C.1.1.4 Example Templates

In general, the following two template files can be used to create new shared object files. Simply modify the function bodies as appropriate for your particular data source.

```
/***************************************************************************
                  template.h  -  data source plugin template
                             -------------------
    begin                : Fri Oct 17 2003
    copyright            : (C) 2003 The University of Toronto
    email                :
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifndef TEMPLATE_H
#define TEMPLATE_H
```

```
#include <kstdatasource.h>

class TemplateSource : public KstDataSource {
public:
  TemplateSource(const QString& filename, const QString& type);
  virtual ~TemplateSource();

  virtual KstObject::UpdateType update(int = -1);
  virtual int readField(double *v, const QString &field, int s, int n);
  virtual bool isValidField(const QString &field) const;
  virtual int samplesPerFrame(const QString &field);
  virtual int frameCount() const;
  virtual QString fileType() const;
  virtual void save(QTextStream &ts);
};

#endif
```

```
/***************************************************************************
                    template.cpp  -  data source template
                             -------------------
    begin                : Fri Oct 17 2003
    copyright            : (C) 2003 The University of Toronto
    email                :
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include "template.h"

TemplateSource::TemplateSource(const QString& filename, const QString& type)
: KstDataSource(filename, type) {
}

TemplateSource::~TemplateSource() {
}

KstObject::UpdateType TemplateSource::update(int u) {
  Q_UNUSED(u)
  return KstObject::NO_CHANGE;
}

int TemplateSource::readField(double *v, const QString& field, int s, int n) {
  Q_UNUSED(v)
  Q_UNUSED(field)
  Q_UNUSED(s)
  Q_UNUSED(n)
  return -1;
}

bool TemplateSource::isValidField(const QString& field) const {
  Q_UNUSED(field)
  return false;
}
```

```
int TemplateSource::samplesPerFrame(const QString &field) {
  Q_UNUSED(field)
  return 0;
}

int TemplateSource::frameCount() const {
  return 0;
}

QString TemplateSource::fileType() const {
  return QString::null;
}

void TemplateSource::save(QTextStream &ts) {
  KstDataSource::save(ts);
}

extern "C" {
KstDataSource *create_template(const QString& filename, const QString& type) {
  return new TemplateSource(filename, type);
}

QStringList provides_template() {
  QStringList rc;
  // create the stringlist
  return rc;
}

bool understands_template(const QString& filename) {
  // determine if it is an X file
  Q_UNUSED(filename)
  return false;
}
}
```

C.1.2 The `.desktop` File

The following is an example of a `.desktop` file for the template plugin:

```
[Desktop Entry]
Encoding=UTF-8
Type=Service
ServiceTypes=Kst Data Source
X-KDE-ModuleType=Plugin
X-Kst-Plugin-Library=template
X-Kst-Plugin-Author=The University of British Columbia
Name=File Reader Template
Comment=Long description of the file reader template.
```

You should add translations in additional languages for the Name and Comment fields by adding additional Name and Comment fields with `[xx]` appended to the end of the field names, where `xx` is the two-letter language code. For example, to add Spanish translations, the following lines would need to be added:

```
Name[es]=Plantilla de lectura de ficheros
Comment[es]=Plantilla de código para hacer un complemento de lectura de ficheros.
```

The field `X-Kst-Plugin-Library` should be exactly the same as the decided library name for the plugin.

C.1.3 Compiling and Copying

To compile and install the new custom datasource reader, create a new directory under `kst/datasources` of the source package. Place the source files for the object, along with the `.desktop` file in the new directory. Then, edit `kst/datasources/Makefile.am` so that `SUBDIRS` contains the name of the new subdirectory. For example, if the new subdirectory is called `template`, `SUBDIRS` might be changed to the following:

```
SUBDIRS=ascii dirfile frame indirect template $(PIOLIB_SUBDIR) $(FITSIO_SUBDIR)
```

After the required files are in the newly created subdirectory, a `Makefile.am` needs to be created in there as well. Use the following sample as a template, replacing all instances of ‘template’ with your custom library name in `kde_module_LTLIBRARIES`, `kstdata_template_la_SOURCES` and `services_DATA`.

```
INCLUDES=-I$(srcdir)/../.. $(all_includes)

kde_module_LTLIBRARIES=kstdata_template.la

kstdata_template_la_LDFLAGS=$(all_libraries) -module -avoid-version
kstdata_template_la_SOURCES=template.cpp

METASOURCES=AUTO

services_DATA=kstdata_template.desktop
servicesdir=$(kde_servicesdir)/kst
```

Once this is done, you can compile and re-install Kst from the modified source package, or alternatively, only install the new libraries, like follows (using the ‘template’ subdirectory as an example). First change to the root directory of the Kst source package. Then,

```
./configure --prefix=`kde-config --prefix`
cd ./kst/datasources/template
make
make install
```

Restart Kst and the new datasource reader should be automatically loaded.

Appendix D

Working with KstScript

D.1 KstScript Classes:

Arrow	Axis	AxisLabel
AxisTickLabel	BinnedMap	BorderedViewObject
Box	Collection	ColorSequence
CrossPowerSpectrum	Curve	CurveCollection
DataMatrix	DataObject	DataObjectCollection
DataSource	DataSourceCollection	DataVector
Debug	DebugLog	DebugLogEntry
Dir	Document	Ellipse
ELOG	Equation	EquationCollection
Extension	ExtensionCollection	File
Group	Histogram	HistogramCollection
Image	Kst	Label
Legend	Line	Matrix
MatrixCollection	Object	ObjectCollection
Picture	Plot	PlotCollection
PlotLabel	Plugin	PluginCollection
PluginIO	PluginIOCollection	PluginManager
PluginModule	PluginModuleCollection	Point
PowerSpectrum	PowerSpectrumCollection	Scalar
ScalarCollection	Size	Spectrogram
SpectrogramCollection	String	StringCollection
TimeInterpretation	Vector	VectorCollection
VectorView	ViewObject	ViewObjectCollection
Window	WindowCollection	

D.2 Class: Arrow

A customizable arrow graphic.

Inherits: Line

Collection class: ViewObjectCollection

D.2.1 Constructors:

- Arrow (parent)

- Arrow (window)

D.2.2 Properties:

- fromArrow
- toArrow
- fromArrowScaling
- toArrowScaling

D.2.3

D.2.3.1 Arrow (ViewObject parent)

- ViewObject parent - The parent to place the new arrow in. May also be a string containing the name of an existing ViewObject.

Creates a new arrow and places it in the ViewObject *parent*.

D.2.3.2 Arrow (Window window)

- Window window - The window to place the new arrow in. May also be a string containing the name of an existing Window.

Creates a new arrow and places it in the Window *window*.

D.2.3.3 boolean fromArrow

True if the arrow has an arrow at the start point.

D.2.3.4 boolean toArrow

True if the arrow has an arrow at the end point.

D.2.3.5 number fromArrowScaling

Scale size of the arrow at the start point.

D.2.3.6 number toArrowScaling

Scale size of the arrow at the end point.

D.3 Class: Axis

A class representing a plot axis.

D.3.1 Methods:

- scaleAuto ( )
- scaleAutoSpikeInsensitive ( )
- scaleExpression (minExp, maxExp)
- scaleRange (min, max)

D.3.2 Properties:

- log
- suppressed
- oppositeSuppressed
- offsetMode
- reversed
- majorGridLines
- minorGridLines
- transformation
- innerTicks
- outerTicks
- label
- type
- majorGridColor
- minorGridColor
- minorTickCount
- majorTickDensity
- scaleMode
- interpretation
- title
- tickLabel

D.3.3

D.3.3.1 void scaleAuto ( )

Sets the scale mode to Auto.

D.3.3.2 void scaleAutoSpikeInsensitive ( )

Sets the scale mode to Auto Spike Insensitive.

D.3.3.3 void scaleExpression (string minExp, string maxExp)

- string minExp - The expression for the minimum of the scale.
- string maxExp - The expression for the maximum of the scale.

Sets the scale mode to Expression.

D.3.3.4 void scaleRange (number min, number max)

- number min - The value for the minimum of the scale.
- number max - The value for the maximum of the scale.

Sets the scale mode to Fixed and sets the range.

D.3.3.5 boolean log

True if the axis is in logarithm mode.

D.3.3.6 boolean suppressed

True if this axis is suppressed.

D.3.3.7 boolean oppositeSuppressed

True if the opposite axis is suppressed. (Right X or top Y)

D.3.3.8 number offsetMode

The base + offset mode. Value must be one of:

- 0 - Automatic
- 1 - On
- 2 - Off

For backwards compatibility the value can also be set as a boolean, in which case true corresponds to On and false corresponds to Automatic.

D.3.3.9 boolean reversed

True if this axis is reversed.

D.3.3.10 boolean majorGridLines

True if this axis shows major grid lines.

D.3.3.11 boolean minorGridLines

True if this axis shows minor grid lines.

D.3.3.12 string transformation

The transformation expression for this axis.

D.3.3.13 boolean innerTicks

True if tick marks are displayed inside the plot.

D.3.3.14 boolean outerTicks

True if tick marks are displayed outside the plot.

D.3.3.15 string label

The label for this axis.

D.3.3.16 string type [Read-Only]

The type of axis - X or Y presently.

D.3.3.17 string majorGridColor

The color for the major grid lines.

D.3.3.18 string minorGridColor

The color for the minor grid lines.

D.3.3.19 number minorTickCount

The number of minor ticks between two major ticks. The special value -1 forces Kst to determine this automatically.

D.3.3.20 number majorTickDensity

The density of major tick markers along this axis. Value must be one of:

- 0 - Coarse
- 1 - Default
- 2 - Fine
- 3 - Very Fine

D.3.3.21 number scaleMode [Read-Only]

The type of scaling done on the axis. Value is one of:

- 0 - Auto
- 1 - All Curves (By Midpoint)
- 2 - Fixed
- 3 - Auto Up (Only Scales Upward)
- 4 - Auto (Spike Insensitive)
- 5 - Auto (obsolete, same as 0)
- 6 - Expression Based

D.3.3.22 TimeInterpretation interpretation [Read-Only]

The method with which the axis is interpreted and times are displayed.

D.3.3.23 AxisLabel title

The label for this axis.

D.3.3.24 AxisTickLabel tickLabel

The tick label for this axis.

D.4 Class: AxisLabel

A class representing a plot axis label.

D.4.1 Properties:

- text
- font
- fontSize
- type

D.4.2

D.4.2.1 string text

Contains the text contents of the label. This may include scalar references of the form *[scalar_name]* and some basic LaTeX.

D.4.2.2 string font

Used to set or get the current font used for the label.

D.4.2.3 number fontSize

Contains the size of the font used to draw the label.

D.4.2.4 string type [Read-Only]

The type of axis - X or Y presently.

D.5 Class: AxisTickLabel

A class representing a plot axis tick label.

D.5.1 Properties:

- font
- fontSize
- rotation
- type

D.5.2

D.5.2.1 string font

Used to set or get the current font used for the label.

D.5.2.2 number fontSize

Contains the size of the font used to draw the label.

D.5.2.3 number rotation

Contains the rotation angle of the label. Rotation is clockwise from *normal* and is stored in degrees.

D.5.2.4 string type [Read-Only]

The type of axis - X or Y presently.

D.6 Class: BinnedMap

This class represents a binned map object in Kst.

Inherits: DataObject

D.6.1 Constructors:

- BinnedMap ( )

D.6.2 Methods:

- validate ( )

D.6.3 Properties:

- x
- y
- z
- binnedMap
- hitsMap
- xFrom
- xTo
- yFrom
- yTo
- nX
- nY
- autobin
- valid

D.6.4

D.6.4.1 BinnedMap ( )

Main constructor for the BinnedMap class.

D.6.4.2 void validate ( )

Validates the plugin.

D.6.4.3 Vector x

Contains the x vector.

D.6.4.4 Vector y

Contains the y vector.

D.6.4.5 Vector z

Contains the z vector.

D.6.4.6 string binnedMap

Contains the name of binned map matrix.

D.6.4.7 string hitsMap

Contains the name of the hits map matrix.

D.6.4.8 Scalar xFrom

Contains the from value of the x vector.

D.6.4.9 Scalar xTo

Contains the to value of the x vector.

D.6.4.10 Scalar yFrom

Contains the from value of the y vector.

D.6.4.11 Scalar yTo

Contains the to value of the y vector.

D.6.4.12 Scalar nX

Contains the number of x bins.

D.6.4.13 Scalar nY

Contains the number of y bins.

D.6.4.14 Scalar autobin

Contains the autobin setting. Zero for no auto-binning and non-zero for auto-binning.

D.6.4.15 boolean valid [Read-Only]

Returns true if all the inputs and outputs of the binned map have been set.

D.7 Class: BorderedViewObject

Represents some objects in a Kst window. This is an abstract object and may not be instantiated directly. It has several properties not in regular ViewObjects including margins, padding, and a border.

Inherits: ViewObject

Collection class: ViewObjectCollection

D.7.1 Properties:

- padding
- margin
- borderWidth
- borderColor

D.7.2

D.7.2.1 number padding

The number of pixels between the border and the contents.

D.7.2.2 number margin

The number of pixels between the edge of the object and the border.

D.7.2.3 number borderWidth

The width in pixels of the border.

D.7.2.4 string borderColor

The border color for this object.

D.8 Class: Box

A customizable box graphic.

Inherits: ViewObject

Collection class: ViewObjectCollection

D.8.1 Constructors:

- Box (parent)
- Box (window)

D.8.2 Properties:

- xRound
- yRound
- borderWidth
- borderColor

D.8.3

D.8.3.1 Box (ViewObject parent)

- ViewObject parent - The parent to place the new box in. May also be a string containing the name of an existing ViewObject.

Creates a new box and places it in the ViewObject *parent*.

D.8.3.2 Box (Window window)

- Window window - The window to place the new box in. May also be a string containing the name of an existing Window.

Creates a new box and places it in the Window *window*.

D.8.3.3 number xRound

The roundness of the corners in the X dimension. Must be an integer between 0 and 99.

D.8.3.4 number yRound

The roundness of the corners in the Y dimension. Must be an integer between 0 and 99.

D.8.3.5 number borderWidth

The width in pixels of the border.

D.8.3.6 string borderColor

The border color for this object.

D.9 Class: Collection

This is the generic collection class used by Kst to store lists of objects. It behaves mostly like an array in JavaScript. At this time indices are read-only. In addition, if a collection is read-only, mutator methods will throw exceptions. Iteration looks something like this:

```
for (i = 0; i < collection.length; ++i) {
  // do something with collection[i]
}
```

D.9.1 Methods:

- append (newObj)
- prepend (newObj)
- remove (n)
- remove (obj)
- clear ( )

D.9.2 Properties:

- length
- readOnly

D.9.3

D.9.3.1 void append (Object newObj)

- Object newObj - The new object to append to the collection.

Appends a new object to the end of the collection.

D.9.3.2 void prepend (Object newObj)

- Object newObj - The new object to prepend to the collection.

Prepends a new object to the start of the collection.

D.9.3.3 void remove (number n)

- number n - The index of the entry to remove.

Removes an entry from the collection.

D.9.3.4 void remove (Object obj)

- Object obj - The object to remove.

Removes an object from the collection.

D.9.3.5 void clear ( )

Clears the collection, removing all entries.

D.9.3.6 number length [Read-Only]

The number of items in the collection. Items are ordered 0..(length - 1) so it is easy to iterate.

D.9.3.7 boolean readOnly [Read-Only]

True if this is a read-only collection.

D.10 Class: ColorSequence

Provides access to the Kst color sequence used for coloring curves.

D.10.1 Methods:

- next ([previous])
- tooClose (firstColor, secondColor)

D.10.2

D.10.2.1 string next ([string previous])

- string previous - The previous color in the color sequence you are following. Not necessary. [OPTIONAL]

Returns the next color in the color sequence.

D.10.2.2 boolean tooClose (string firstColor, string secondColor)

- string firstColor - The first of two colors to compare.
- string secondColor - The second of two colors to compare.

Returns true if colors are too close.

D.11 Class: CrossPowerSpectrum

This class represents a cross power spectrum object in Kst.

Inherits: DataObject

D.11.1 Methods:

- validate ( )

D.11.2 Properties:

- v1
- v2
- length
- sample
- real
- imaginary
- frequency
- valid

D.11.3

D.11.3.1 void validate ( )

Validates the plugin.

D.11.3.2 Vector v1

Contains the first vector.

D.11.3.3 Vector v2

Contains the second vector.

D.11.3.4 Scalar length

Contains the base 2 logarithm of the length of the cross power spectrum. Should be an integer >= 2.

D.11.3.5 Scalar sample

Contains the sample rate.

D.11.3.6 string real

Contains the name of the output vector holding the real part of the cross power spectrum.

D.11.3.7 string imaginary

Contains the name of the output vector holding the imaginary part of the cross power spectrum.

D.11.3.8 string frequency

Contains the name of the output vector holding the frequencies of the cross power spectrum.

D.11.3.9 boolean valid [Read-Only]

Returns true if all the inputs and outputs of the cross power spectrum have been set.

D.12 Class: Curve

Represents a curve formed from an X and a Y vector.

Inherits: DataObject

Collection class: CurveCollection

D.12.1 Constructors:

- Curve (x, y [, xError [, yError [, xMinusError [, yMinusError]]]])
- Curve (hs)

D.12.2 Methods:

- point (index)
- xErrorPoint (index)
- yErrorPoint (index)
- xMinusErrorPoint (index)
- yMinusErrorPoint (index)

D.12.3 Properties:

- color
- xVector
- yVector
- xErrorVector
- yErrorVector
- xMinusErrorVector
- xMinusErrorVector
- yVectorOffset
- samplesPerFrame
- ignoreAutoScale
- hasPoints
- hasLines
- hasBars
- lineWidth
- pointStyle
- lineStyle
- barStyle
- pointDensity
- topLabel
- xLabel
- yLabel
- legendText

D.12.4

D.12.4.1 Curve (Vector x, Vector y [, Vector xError [, Vector yError [, Vector xMinusError [, Vector yMinusError]]]])

- Vector x - The X vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object.
- Vector y - The Y vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object.
- Vector xError - The X error bar vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object. [OPTIONAL]
- Vector yError - The Y error bar vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object. [OPTIONAL]
- Vector xMinusError - The X minus error bar vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object. [OPTIONAL]
- Vector yMinusError - The Y minus error bar vector for the curve. Can be specified as a string containing the tag name of an existing vector, or as a vector object. [OPTIONAL]

Main constructor for Curve class. Constructing a new curve automatically adds it to the data collection of Kst.

D.12.4.2 Curve (Histogram hs)

- Histogram hs - Creates a curve from a histogram.

Constructor for Curve class. Allows a curve to be simply created from a Histogram class.

D.12.4.3 Point point (number index)

- number index - The index of the point to retrieve the coordinates for. Starts at 0.

Retrieve the coordinates for a given point in the curve. Returns an invalid point if that index does not exist.

D.12.4.4 number xErrorPoint (number index)

- number index - The index of the point to retrieve the coordinates for. Starts at 0.

Retrieve the value for a given point in the X error bars. Returns an invalid point if that index does not exist.

D.12.4.5 number yErrorPoint (number index)

- number index - The index of the point to retrieve the coordinates for. Starts at 0.

Retrieve the value for a given point in the Y error bars. Returns an invalid point if that index does not exist.

D.12.4.6 number xMinusErrorPoint (number index)

- number index - The index of the point to retrieve the coordinates for. Starts at 0.

Retrieve the value for a given point in the X minus error bars. Returns an invalid point if that index does not exist.

D.12.4.7 number yMinusErrorPoint (number index)

- number index - The index of the point to retrieve the coordinates for. Starts at 0.

Retrieve the value for a given point in the Y minus error bars. Returns an invalid point if that index does not exist.

D.12.4.8 string color

The color of the curve. Stored in the form "#RRGGBB" as hex values. This property can also be set with English strings such as "blue".

D.12.4.9 Vector xVector

The X axis vector for the curve.

D.12.4.10 Vector yVector

The Y axis vector for the curve.

D.12.4.11 Vector xErrorVector

The X axis error vector for the curve.

D.12.4.12 Vector yErrorVector

The Y axis error vector for the curve.

D.12.4.13 Vector xMinusErrorVector

The X axis negative error vector for the curve.

D.12.4.14 Vector xMinusErrorVector

The X axis negative error vector for the curve.

D.12.4.15 Scalar yVectorOffset

The offset to be applied to the y-axis vector.

D.12.4.16 number samplesPerFrame [Read-Only]

The number of samples per frame for the curve.

D.12.4.17 boolean ignoreAutoScale

If true, this curve ignores auto scale.

D.12.4.18 boolean hasPoints

If true, points are plotted for this curve.

D.12.4.19 boolean hasLines

If true, lines are drawn for this curve.

D.12.4.20 boolean hasBars

If true, bars are drawn for this curve.

D.12.4.21 number lineWidth

Sets the line width for this curve when lines are drawn.

D.12.4.22 number pointStyle

Sets the point style for this curve when points are drawn.

- 0 - Cross
- 1 - Hollow box
- 2 - Hollow circle
- 3 - Filled circle
- 4 - Inverted hollow triangle
- 5 - Hollow triangle
- 6 - Filled box
- 7 - Plus
- 8 - Asterisk
- 9 - Inverted filled triangle
- 10 - Filled triangle
- 11 - Hollow diamond
- 12 - Filled diamond
- 13 - Point

D.12.4.23 number lineStyle

Sets the line style for this curve when lines are drawn.

- 0 - Solid
- 1 - Dash
- 2 - Dot
- 3 - Dash Dot
- 4 - Dash Dot Dot

D.12.4.24 number barStyle

Sets the bar style for this curve when bars are drawn.

- 0 - Lines
- 1 - Solid

D.12.4.25 number pointDensity

Sets the point density for this curve when points are plotted.

- 0 - Draw all points
- 1 - High density
- 2 - Medium density
- 3 - Low density

D.12.4.26 string topLabel [Read-Only]

The top label suggestion for this curve.

D.12.4.27 string xLabel [Read-Only]

The X-axis label suggestion for this curve.

D.12.4.28 string yLabel [Read-Only]

The Y-axis label suggestion for this curve.

D.12.4.29 string legendText

The legend text associated with a curve.

D.13 Class: CurveCollection

An array of curves.

Inherits: Collection

D.13.1

D.14 Class: DataMatrix

A matrix object loaded from a data source.

Inherits: Matrix

D.14.1 Constructors:

- DataMatrix (source, field [, xStart, yStart, xN, yN [, skip [, ave]]])

D.14.2 Methods:

- reload ( )
- changeFile (source)
- change (xStart, yStart, xCount, yCount [, skip [, ave]])

D.14.3 Properties:

- valid
- skip
- boxcar
- xReadToEnd

- yReadToEnd
- xCountFromEnd
- yCountFromEnd
- skipLength
- field
- dataSource

D.14.4

D.14.4.1 DataMatrix (DataSource source, string field [, number xStart, number yStart, number xN, number yN [, number skip [, boolean ave]]])

- DataSource source - The DataSource to load data from. This may be a string containing the url of the DataSource to load. If so, it will attempt to reuse an existing DataSource instance, or fall back to a new DataSource object. It may also be specified as a DataSource object.
- string field - The field to load from the source.
- number xStart - The x frame to start reading from. [OPTIONAL]
- number yStart - The y frame to start reading from.
- number xN - The number of x frames to read.
- number yN - The number of y frames to read.
- number skip - The number of samples to skip by. [OPTIONAL]
- boolean ave - Set to true to do boxcar filtering. [OPTIONAL]

D.14.4.2 void reload ( )

D.14.4.3 void changeFile (DataSource source)

- DataSource source - The DataSource to switch to.

Switches this DataMatrix to a different DataSource.

D.14.4.4 void change (number xStart, number yStart, number xCount, number yCount [, number skip [, boolean ave]])

- number xStart - The x starting frame.
- number yStart - The y starting frame.
- number xCount - The number of x frames to read.
- number yCount - The number of y frames to read.
- number skip - The number of frames to skip by. [OPTIONAL]
- boolean ave - True to boxcar filter. [OPTIONAL]

Changes the sequence of data that is read in by this matrix.

D.14.4.5 boolean valid [Read-Only]

True if the data matrix is valid.

D.14.4.6 boolean skip [Read-Only]

True if the matrix should skip by *skipLength* samples while reading from the data source.

D.14.4.7 boolean boxcar [Read-Only]

True if the matrix should be boxcar filtered.

D.14.4.8 boolean xReadToEnd [Read-Only]

If true, the matrix is read to the end of the source in the x dimension.

D.14.4.9 boolean yReadToEnd [Read-Only]

If true, the matrix is read to the end of the source in the y dimension.

D.14.4.10 boolean xCountFromEnd [Read-Only]

If true, the matrix reads only a maximum number of samples from the end of the source in the x dimension.

D.14.4.11 boolean yCountFromEnd [Read-Only]

If true, the matrix reads only a maximum number of samples from the end of the source in the y dimension.

D.14.4.12 number skipLength [Read-Only]

The number of samples to skip by.

D.14.4.13 string field [Read-Only]

The field being read from the data source to create this matrix.

D.14.4.14 DataSource dataSource [Read-Only]

The data source object used by this DataMatrix.

D.15 Class: DataObject

The abstract base class for all Kst data objects.

Inherits: Object

Collection class: DataObjectCollection

D.15.1 Methods:

- convertTo (type)

D.15.2 Properties:

- type

D.15.3

D.15.3.1 DataObject convertTo (string type)

- string type - The type to attempt to convert this object to.

Attempts to convert this DataObject to a different type. The object must be derived from this type at some level. Null is returned if it is not possible to convert to the requested type.

D.15.3.2 string type [Read-Only]

The type of data object this is.

D.16 Class: DataObjectCollection

An array of DataObjects.

Inherits: Collection

D.16.1

D.17 Class: DataSource

An object that represents a data file that is understood by Kst.

Inherits: Object

Collection class: DataSourceCollection

D.17.1 Constructors:

- DataSource (url [, type])

D.17.2 Methods:

- isValidField (field)
- fieldList ( )
- reset ( )
- matrixList ( )
- samplesPerFrame (field)

- frameCount ([field])
- setConfiguration (setting, setting)
- configuration (setting)
- units (field)

D.17.3 Properties:

- valid
- empty
- completeFieldList
- fileName
- fileType
- source
- metaData

D.17.4

D.17.4.1 DataSource (string url [, string type])

- string url - The filename or URL to load.
- string type - The name of the data source plugin to use. [OPTIONAL]

Creates a new DataSource object. If the url cannot be opened and read, *isValid* will be false.

D.17.4.2 boolean isValidField (string field)

- string field - A field name to check for.

Returns if the field is valid or not.

D.17.4.3 StringArray fieldList ( )

Generates a list of the fields available from this source.

D.17.4.4 void reset ( )

Resets the data source.

D.17.4.5 StringArray matrixList ( )

Generates a list of the matrices available from this source.

D.17.4.6 number samplesPerFrame (string field)

- string field - A field name to get the number of samples per frame for. This is the same for every field in some sources, but different in others.

The number of samples per frame for this field or source.

D.17.4.7 number frameCount ([string field])

- string field - An optional field name to get the number of frames for. This is the same for every field in some sources, but different in others. [OPTIONAL]

Gets the number of frames for the source, or the given field if it is specified.

D.17.4.8 boolean setConfiguration (string setting, string setting)

- string setting - Gives the setting name to set the value of.
- string setting - Gives the value of the setting.

Sets a configuration setting of the data source.

D.17.4.9 string configuration (string setting)

- string setting - Gives the setting name to retrive the value of.

Gets a configuration setting of the data source.

D.17.4.10 string units (string field)

- string field - Gives the field name to retrieve the units of.

Gets the units associated with a field of the data source.

D.17.4.11 boolean valid [Read-Only]

True if the data source is valid.

D.17.4.12 boolean empty [Read-Only]

True if the data source is empty.

D.17.4.13 boolean completeFieldList [Read-Only]

True if the field list is complete.

D.17.4.14 string fileName [Read-Only]

Name of the file.

D.17.4.15 string fileType [Read-Only]

The type (format) of the file, such as ASCII.

D.17.4.16 string source [Read-Only]

The name of the data source.

D.17.4.17 StringArray metaData [Read-Only]

Any metadata associated with the data source. This is an associative array of form metaData["key"] = "value".

D.18 Class: DataSourceCollection

An array of DataSources.

Inherits: Collection

D.18.1

D.19 Class: DataVector

A vector object loaded from a data source.

Inherits: Vector

D.19.1 Constructors:

- DataVector (source, field [, start [, n [, skip [, ave]]]])
- DataVector (source, field, typeStart, start, typeCount, count [, skip [, ave]])

D.19.2 Methods:

- reload ( )
- changeFile (source)
- changeFrames (start, count [, skip [, ave]])
- changeFramesByTime (typeStart, start, typeCount, count [, skip [, ave]])

D.19.3 Properties:

- valid
- skip
- boxcar
- readToEnd
- countFromEnd

- skipLength
- startFrame
- startFrameRequested
- frames
- framesRequested
- samplesPerFrame
- field
- dataSource

D.19.4

D.19.4.1 DataVector (DataSource source, string field [, number start [, number n [, number skip [, boolean ave]]]])

- DataSource source - The DataSource to load data from. This may be a string containing the url of the DataSource to load. If so, it will attempt to reuse an existing DataSource instance, or fall back to a new DataSource object. It may also be specified as a DataSource object.
- string field - The field to load from the source.
- number start - The frame to start reading from. [OPTIONAL]
- number n - The number of frames to read. [OPTIONAL]
- number skip - The number of samples to skip by. [OPTIONAL]
- boolean ave - Set to true to do boxcar filtering. [OPTIONAL]

D.19.4.2 DataVector (DataSource source, string field, string typeStart, number start, string typeCount, number count [, number skip [, boolean ave]])

- DataSource source - The DataSource to load data from. This may be a string containing the url of the DataSource to load. If so, it will attempt to reuse an existing DataSource instance, or fall back to a new DataSource object. It may also be specified as a DataSource object.
- string field - The field to load from the source.
- string typeStart - The type of the value specifying the start position. The permissible values are as follows:
  - frames
  - date
  - ms (milliseconds)
  - s (seconds)
  - m (minutes)
  - h (hours)
  - days
  - weeks
  - months
  - years

The date entry uses the format yyyy:mm:dd:hh:mm.ss (e.g. 2009:2:13:13:10.12). If any of the leading elements are omitted then the current year, month, day, etc. are assumed. If the year is less than 0, then it is assumed to be given relative to 1970 (e.g. -10 becomes 1980), else if the year is less than 100 it is assumed to be relative to 2000 (e.g. 9 becomes 2009), else the year is taken as given.

- number start - The starting frame, date (a string argument), or time offset.
- string typeCount - The type of the value specifying the count
- number count - The number of frames or time duration to read. The permissible values are as follows:
  - frames
  - ms (milliseconds)
  - s (seconds)
  - m (minutes)
  - h (hours)
  - days
  - weeks
  - months
  - years
- number skip - The number of samples to skip by. [OPTIONAL]
- boolean ave - Set to true to do boxcar filtering. [OPTIONAL]

D.19.4.3 void reload ( )

D.19.4.4 void changeFile (DataSource source)

- DataSource source - The DataSource to switch to.

Switches this DataVector to a different DataSource.

D.19.4.5 void changeFrames (number start, number count [, number skip [, boolean ave]])

- number start - The starting frame.
- number count - The number of frames to read.
- number skip - The number of frames to skip by. [OPTIONAL]
- boolean ave - True to boxcar filter. [OPTIONAL]

Changes the sequence of data that is read in by this vector.

D.19.4.6 void changeFramesByTime (string typeStart, number start, string typeCount, number count [, number skip [, boolean ave]])

- string typeStart - The type of the value specifying the start position. The permissible values are as follows:
  - frames
  - date
  - ms (milliseconds)
  - s (seconds)

  - m (minutes)
  - h (hours)
  - days
  - weeks
  - months
  - years

  The date entry uses the format yyyy:mm:dd:hh:mm.ss (e.g. 2009:2:13:13:10.12). If any of the leading elements are omitted then the current year, month, day, etc. are assumed. If the year is less than 0, then it is assumed to be given relative to 1970 (e.g. -10 becomes 1980), else if the year is less than 100 it is assumed to be relative to 2000 (e.g. 9 becomes 2009), else the year is taken as given.

- number start - The starting frame, date (a string argument), or time offset.
- string typeCount - The type of the value specifying the count
- number count - The number of frames or time duration to read. The permissible values are as follows:
  - frames
  - ms (milliseconds)
  - s (seconds)
  - m (minutes)
  - h (hours)
  - days
  - weeks
  - months
  - years
- number skip - The number of frames to skip by. [OPTIONAL]
- boolean ave - True to boxcar filter. [OPTIONAL]

Changes the sequence of data that is read in by this vector.

D.19.4.7 boolean valid [Read-Only]

True if the data vector is valid.

D.19.4.8 boolean skip [Read-Only]

True if the vector should skip by *skipLength* samples while reading from the data source.

D.19.4.9 boolean boxcar [Read-Only]

True if the vector should be boxcar filtered.

D.19.4.10 boolean readToEnd [Read-Only]

If true, the vector is read to the end of the source.

D.19.4.11 boolean countFromEnd [Read-Only]

If true, the vector reads only a maximum number of samples from the end of the source.

D.19.4.12 number skipLength [Read-Only]

The number of samples to skip by.

D.19.4.13 number startFrame [Read-Only]

The starting frame number.

D.19.4.14 number startFrameRequested [Read-Only]

The requested starting frame number. May not be the actual *startFrame*.

D.19.4.15 number frames [Read-Only]

The number of frames read from the source.

D.19.4.16 number framesRequested [Read-Only]

The number of frames requested to be read from the source. May not be the actual *frames*.

D.19.4.17 number samplesPerFrame [Read-Only]

The number of samples per frame for the associated data source.

D.19.4.18 string field [Read-Only]

The field being read from the data source to create this vector.

D.19.4.19 DataSource dataSource [Read-Only]

The data source object used by this DataVector.

D.20 Class: Debug

A reference to the debug and logging subsystem of Kst.

D.20.1 Methods:

- warning (message)
- error (message)
- notice (message)
- debug (message)
- clear ( )
- clearNewError ( )

D.20.2 Properties:

- log
- newError
- version
- revision

D.20.3

D.20.3.1 void warning (string message)

- string message -

Logs a warning message to the Kst log.

D.20.3.2 void error (string message)

- string message -

Logs an error message to the Kst log.

D.20.3.3 void notice (string message)

- string message -

Logs a notice to the Kst log.

D.20.3.4 void debug (string message)

- string message -

Logs a debug message to the Kst log.

D.20.3.5 void clear ( )

Clears the Kst log.

D.20.3.6 void clearNewError ( )

Clears the new error flag.

D.20.3.7 DebugLog log [Read-Only]

Provides access to the Kst log contents.

D.20.3.8 boolean newError [Read-Only]

Return the value of the new error flag.

D.20.3.9 string version [Read-Only]

Return the version of Kst.

D.20.3.10 string revision [Read-Only]

Return the revision of Kst.

D.21 Class: DebugLog

A reference to the Kst debug log. This object is an array of the entries in the debug log. It cannot be modified. The length of the array is the number of entries, and each entry is a *DebugLogEntry* object.

D.21.1 Properties:

- length
- text
- lengthNotices
- textNotices
- lengthWarnings
- textWarnings
- lengthErrors
- textErrors
- lengthDebugs
- textDebugs

D.21.2

D.21.2.1 number length [Read-Only]

The number of entries in the log.

D.21.2.2 string text [Read-Only]

The text contents of the log as a single large string with carriage returns between entries.

D.21.2.3 number lengthNotices [Read-Only]

The number of notices in the log.

D.21.2.4 string textNotices [Read-Only]

The text contents of the log notices as a single large string with carriage returns between entries.

D.21.2.5 number lengthWarnings [Read-Only]

The number of warnings in the log.

D.21.2.6 string textWarnings [Read-Only]

The text contents of the log warnings as a single large string with carriage returns between entries.

D.21.2.7 number lengthErrors [Read-Only]

The number of errors in the log.

D.21.2.8 string textErrors [Read-Only]

The text contents of the log errors as a single large string with carriage returns between entries.

D.21.2.9 number lengthDebugs [Read-Only]

The number of debug messages in the log.

D.21.2.10 string textDebugs [Read-Only]

The text contents of the log debug messages as a single large string with carriage returns between entries.

D.22 Class: DebugLogEntry

An entry in the Kst debug log.

D.22.1 Properties:

- text
- date
- level

D.22.2

D.22.2.1 string text [Read-Only]

Contains the text of this entry.

D.22.2.2 date date [Read-Only]

Contains the timestamp of this entry.

D.22.2.3 string level [Read-Only]

Contains the log level for this entry.

- N - Notice
- D - Debug
- W - Warning
- E - Error
- A single space ' ' indicates an unknown entry type.

D.23 Class: Dir

An object that represents a directory.

D.23.1 Constructors:

- Dir (name)

D.23.2 Methods:

- fileList (nameFilters)
- dirList (nameFilters)

D.23.3 Properties:

- path

D.23.4

D.23.4.1 Dir (string name)

- string name - Name of the directory.

Constructs a dir in the given directory.

D.23.4.2 StringArray fileList (string nameFilters)

- string nameFilters - Specifies the name filters.

Returns a list of files in the directory that match the given name filters.

D.23.4.3 StringArray dirList (string nameFilters)

- string nameFilters - Specifies the name filters.

Returns a list of sub-directories in the directory that match the given name filters.

D.23.4.4 string path

The path of the folder.

D.24 Class: Document

A pointer to a Kst document. Typically accessed via the global *Kst* object via *Kst.document*.

D.24.1 Methods:

- save ([filename])
- clear ( )
- load (filename)

D.24.2 Properties:

- text
- name
- modified

D.24.3

D.24.3.1 void save ([string filename])

- string filename - An optional filename to save to. If not provided, it will be saved to the current filename or the user will be prompted. [OPTIONAL]

Saves the current Kst session to disk.

D.24.3.2 void clear ( )

Clears the current Kst session and starts a new one.

D.24.3.3 boolean load (string filename)

- string filename - The filename for a Kst file to load.

Loads a new Kst file into Kst.

D.24.3.4 string text [Read-Only]

The XML text of the current session in Kst-file form.

D.24.3.5 string name

The filename for the current Kst session.

D.24.3.6 boolean modified

True if the document has been modified but not saved.

D.25 Class: Ellipse

A customizable ellipse graphic.

Inherits: ViewObject

Collection class: ViewObjectCollection

D.25.1 Constructors:

- Ellipse (parent)
- Ellipse (window)

D.25.2 Properties:

- borderWidth
- borderColor

D.25.3

D.25.3.1 Ellipse (ViewObject parent)

- ViewObject parent - The parent to place the new ellipse in. May also be a string containing the name of an existing ViewObject.

Creates a new ellipse and places it in the ViewObject *parent*.

D.25.3.2 Ellipse (Window window)

- Window window - The window to place the new ellipse in. May also be a string containing the name of an existing Window.

Creates a new ellipse and places it in the Window *window*.

D.25.3.3 number borderWidth

The width of the border of the ellipse.

D.25.3.4 string borderColor

The color of the border of the ellipse.

D.26 Class: ELOG

An object that represents the interface to an ELOG server.

D.26.1 Constructors:

- ELOG ( )

D.26.2 Methods:

- submit ( )
- addAttachment (attachment)
- clearAttachments ( )
- addAttribute (attribute, attributeValue)
- removeAttribute (attribute)
- clearAttributes ( )

D.26.3 Properties:

- hostname
- port
- logbook
- username
- password
- writePassword
- text
- encodedHTML
- suppressEmailNotification
- includeCapture
- captureWidth
- captureHeight
- includeConfiguration
- includeDebugInfo

D.26.4

D.26.4.1 ELOG ( )

Interface to an ELOG server.

D.26.4.2 void submit ( )

Submits the ELOG entry.

D.26.4.3 void addAttachment (string attachment)

- string attachment - The name of the attachment.

Adds an attachment to the ELOG entry.

D.26.4.4 void clearAttachments ( )

Clears the list of attachments in the ELOG entry.

D.26.4.5 void addAttribute (string attribute, string attributeValue)

- string attribute - The name of the attribute to be added to the ELOG entry.
- string attributeValue - The value of the attribute to be added to the ELOG entry.

Adds an attribute to the ELOG entry.

D.26.4.6 void removeAttribute (string attribute)

- string attribute - The name of the attribute to be removed from the ELOG entry.

Removes an attribute from the ELOG entry.

D.26.4.7 void clearAttributes ( )

Clears the list of attributes in the ELOG entry.

D.26.4.8 string hostname

The hostname of the ELOG server.

D.26.4.9 number port

The port number of the ELOG server.

D.26.4.10 string logbook

A weblog made available by the ELOG server.

D.26.4.11 string username

Username.

D.26.4.12 string password

Password.

D.26.4.13 string writePassword

The write password.

D.26.4.14 string text

Text.

D.26.4.15 boolean encodedHTML

The message is HTML encoded.

D.26.4.16 boolean suppressEmailNotification

If true email notification is suppressed.

D.26.4.17 boolean includeCapture

If true an image capture of the Kst session is included in the ELOG entry.

D.26.4.18 number captureWidth

The width of the image capture to be included in the ELOG entry.

D.26.4.19 number captureHeight

The height of the image capture to be included in the ELOG entry.

D.26.4.20 boolean includeConfiguration

If true a capture of the Kst session is included in the ELOG entry.

D.26.4.21 boolean includeDebugInfo

If true a capture of the Kst debug information is included in the ELOG entry.

D.27 Class: Equation

This object represents an equation object in Kst. It is interpreted using Kst's internal equation interpreter, and not the JavaScript interpreter. It should be considerably faster than the JavaScript interpreter.

Inherits: DataObject

Collection class: EquationCollection

D.27.1 Constructors:

- Equation (equation, xVector [, interpolate])
- Equation (equation, x0, x1, n)

D.27.2 Properties:

- equation
- valid
- xVector
- yVector

D.27.3

D.27.3.1 Equation (string equation, Vector xVector [, boolean interpolate])

- string equation - The text of the equation to interpret.
- Vector xVector - The X axis (input) vector to use. The symbolic variable *x*; represents the value at the current index in this vector. A text string representing the name of an existing vector may also be used here.
- boolean interpolate - If true, the equation interpreter will interoplate to the highest resolution vector. This is the default behavior. [OPTIONAL]

The default constructor to create an equation object.

D.27.3.2 Equation (string equation, number x0, number x1, number n)

- string equation - The text of the equation to interpret.
- number x0 - The first value of the generated X (input) vector.
- number x1 - The last value of the generate X (input) vector.
- number n - The number of values in the generated input vector.

A utility constructor which generates an X vector implicitly.

D.27.3.3 string equation

The text of the equation. See the Kst documentation for more information on what the internal equation interpreter supports.

D.27.3.4 boolean valid [Read-Only]

True if the equation object is valid. If false, any values obtained from the yVector should be considered meaningless.

D.27.3.5 Vector xVector [Read-Only]

The X vector (input vector) for the equation.

D.27.3.6 Vector yVector [Read-Only]

The Y vector (output vector) for the equation.

D.28 Class: EquationCollection

An array of Equations.

Inherits: Collection

D.28.1

D.29 Class: Extension

A class representing an extension in Kst.

Collection class: ExtensionCollection

D.29.1 Methods:

- load ( )
- unload ( )

D.29.2 Properties:

- name
- loaded

D.29.3

D.29.3.1 boolean load ( )

Loads this extension into Kst.

D.29.3.2 void unload ( )

Unloads this extension if it is loaded.

D.29.3.3 string name [Read-Only]

The name of this extension.

D.29.3.4 boolean loaded [Read-Only]

True if this extension is presently loaded.

D.30 Class: ExtensionCollection

An array of Kst Extension objects.

Inherits: Collection

D.30.1

D.31 Class: File

An object that represents a file.

D.31.1 Constructors:

- File (name)

D.31.2 Methods:

- open ( )
- close ( )
- readLine ([length])

D.31.3 Properties:

- name
- size
- exists

D.31.4

D.31.4.1 File (string name)

- string name - Name of the file to be opened.

Constructs a file with the given name.

D.31.4.2 void open ( )

Opens the file.

D.31.4.3 void close ( )

Closes the file.

D.31.4.4 string readLine ([number length])

- number length - Maximum length of string to return [OPTIONAL]

Reads a line from the file.

D.31.4.5 string name

The name of the file.

D.31.4.6 number size

The size of the file.

D.31.4.7 boolean exists

True if the underlying file exists.

D.32 Class: Group

A customizable group object.

Inherits: ViewObject

Collection class: ViewObjectCollection

D.32.1 Constructors:

- Group (parent)
- Group (window)

D.32.2 Methods:

- append (newObj)
- prepend (newObj)
- ungroup ( )

D.32.3

D.32.3.1 Group (ViewObject parent)

- ViewObject parent - The parent to place the new arrow in. May also be a string containing the name of an existing ViewObject.

Creates a new arrow and places it in the ViewObject *parent*.

D.32.3.2 Group (Window window)

- Window window - The window to place the new arrow in. May also be a string containing the name of an existing Window.

Creates a new arrow and places it in the Window *window*.

D.32.3.3 void append (Object newObj)

- Object newObj - The new object to append to the collection.

Appends a new object to the end of the collection.

D.32.3.4 void prepend (Object newObj)

- Object newObj - The new object to prepend to the collection.

Prepends a new object to the start of the collection.

D.32.3.5 void ungroup ( )

Removes all objects from the collection.

D.33 Class: Histogram

A class representing a histogram object in Kst.

Inherits: DataObject

Collection class: HistogramCollection

D.33.1 Constructors:

- Histogram (vector [, xMin [, xMax [, bins]]])

D.33.2 Methods:

- setVector (vector)
- setRange (from, to)

D.33.3 Properties:

- realTimeAutoBin
- normalization
- bins
- xMin
- xMax
- xVector
- yVector

D.33.4

D.33.4.1 Histogram (Vector vector [, number xMin [, number xMax [, number bins]]])

- Vector vector - The input vector for the histogram. May also be specified by the tagName for an existing vector.
- number xMin - The minimum X value. Default is -10. [OPTIONAL]
- number xMax - The maximum X value. Default is 10. [OPTIONAL]
- number bins - The number of bins to use. Default is 60. [OPTIONAL]

Creates a new histogram with the defined properties.

D.33.4.2 void setVector (Vector vector)

- Vector vector - The vector for the histogram to operate on.

Sets the input vector for the histogram.

D.33.4.3 void setRange (number from, number to)

- number from - The starting X value.
- number to - The ending X value.

Sets the X range for the histogram.

D.33.4.4 boolean realTimeAutoBin

If true, the histogram automatically bins in real-time.

D.33.4.5 number normalization

Set the y-axis normalization.

- 0 - Number in bin
- 1 - Percent in bin
- 2 - Fraction in bin
- 3 - Peak bin = 1.0

D.33.4.6 number bins

The number of bins for the histogram.

D.33.4.7 number xMin [Read-Only]

The minimum X value for the histogram.

D.33.4.8 number xMax [Read-Only]

The maximum X value for the histogram.

D.33.4.9 Vector xVector [Read-Only]

The X-axis vector generated by the histogram.

D.33.4.10 Vector yVector [Read-Only]

The Y-axis vector generated by the histogram.

D.34 Class: HistogramCollection

An array of Histograms.

Inherits: Collection

D.34.1

D.35 Class: Image

An image object created from a matrix.

Inherits: DataObject

D.35.1 Constructors:

- Image (matrix)

D.35.2 Methods:

- minMaxThreshold ( )
- smartThreshold ( )

D.35.3 Properties:

- matrix
- map
- palette
- lowerThreshold
- upperThreshold
- autoThreshold
- numContours
- contourWeight
- color

D.35.4

D.35.4.1 Image (Matrix matrix)

- Matrix matrix - The matrix for the image. Can be specified as a string containing the tag name of an existing matrix, or as a matrix object.

Main constructor for Image class. Constructing a new image automatically adds it to the data collection of Kst.

D.35.4.2 void minMaxThreshold ( )

Set the thresholding to the current minimum and maximum of the associated matrix.

D.35.4.3 void smartThreshold ( )

Set the thresholding to automatically calculated smart values based on the current values of the the associated matrix.

D.35.4.4 Matrix matrix

The matrix associated with the image.

D.35.4.5 number map

The map type of the image.

- 0 - Color map
- 1 - Contour map
- 2 - Color map and contour map

D.35.4.6 string palette

The name of the palette associated with the image color map.

D.35.4.7 number lowerThreshold

The lower threshold associated with the map.

D.35.4.8 number upperThreshold

The upper threshold associated with the map.

D.35.4.9 boolean autoThreshold

If true, automatic thresholding is enabled.

D.35.4.10 number numContours

The number of contour levels if the contour map is enabled.

D.35.4.11 number contourWeight

The weight of the lines used to draw the contour map, if the contour map is enabled. A value of -1 can be used for a variable contour weight.

D.35.4.12 string color

The color of the contours. Stored in the form "#RRGGBB" as hex values. This property can also be set with English strings such as "blue".

D.36 Class: Kst

The global Kst object. Accessed as *Kst*, this cannot be reinstantiated.

D.36.1 Methods:

- loadScript (fileName)
- resetInterpreter ( )
- waitForUpdate ( )
- purge ( )
- advance ( )
- back ( )
- writeHistory (filename)
- clearHistory ( )
- addToOutput (output)
- autoWriteHistory (filename [, append])
- forceRepaint ( )

D.36.2 Properties:

- dataSources
- scalars
- strings
- vectors
- matrices
- windows
- objects
- colors
- extensions
- document
- pluginManager
- gui
- repaintEnabled

D.36.3

D.36.3.1 boolean loadScript (string fileName)

- string fileName - The file to load the script from.

Loads a script file from disk and runs it.

D.36.3.2 void resetInterpreter ( )

Resets the KstScript interpreter. All variables will be lost.

D.36.3.3 void waitForUpdate ( )

Waits for the next update loop to complete.

D.36.3.4 void purge ( )

Purges unused objects in Kst. The has the same behavior as the purge button in the data manager.

D.36.3.5 void advance ( )

Advance one screen.

D.36.3.6 void back ( )

Back one screen.

D.36.3.7 void writeHistory (string filename)

- string filename - The name of the file to save the command history to.

Saves the command history to a file.

D.36.3.8 void clearHistory ( )

Clears the command history.

D.36.3.9 void addToOutput (string output)

- string output - String to add to the output.

Adds to the output generated from a function call.

D.36.3.10 void autoWriteHistory (string filename [, boolean append])

- string filename - The name of the file to save commands to automatically. Entering a blank filename will stop the saving of commands.
- boolean append - If true and the file already exists then commands will be appended to the existing file. [OPTIONAL]

Saves commands automatically to a file.

D.36.3.11 void forceRepaint ( )

Forces a repaint regardless of the value of repaintEnabled.

D.36.3.12 DataSourceCollection dataSources [Read-Only]

The list of all loaded data sources.

D.36.3.13 ScalarCollection scalars [Read-Only]

The list of all scalars in Kst.

D.36.3.14 StringCollection strings [Read-Only]

The list of all strings in Kst.

D.36.3.15 VectorCollection vectors [Read-Only]

The list of all vectors in Kst.

D.36.3.16 MatrixCollection matrices [Read-Only]

The list of all matrices in Kst.

D.36.3.17 WindowCollection windows [Read-Only]

The list of all Kst windows in this process.

D.36.3.18 DataObjectCollection objects [Read-Only]

The list of all data objects in Kst.

D.36.3.19 ColorSequence colors [Read-Only]

A reference to the Kst color sequence in its current state.

D.36.3.20 ExtensionCollection extensions [Read-Only]

The list of all extensions Kst can find installed on the system.

D.36.3.21 Document document [Read-Only]

An object that provides access to the current Kst document along with various utility functions.

D.36.3.22 PluginManager pluginManager [Read-Only]

A reference to the plugin management subsystem of Kst.

D.36.3.23 QWidget gui [Read-Only]

A reference to the Kst GUI. This is implemented using KJSEmbed. The GUI is dynamic and may change between releases or even while Kst is running.

D.36.3.24 boolean repaintEnabled

Determines whether or not manipulation of the Kst interface through scripting will trigger a repaint. Setting this to false before issuing a long series of commands (that each triggers a repaint), and then setting it back to true can greatly enhance the response time of Kst.

D.37 Class: Label

A text label.

Inherits: BorderedViewObject

Collection class: ViewObjectCollection

D.37.1 Constructors:

- Label (parent)
- Label (window)

D.37.2 Methods:

- adjustSizeForText ( )

D.37.3 Properties:

- text
- font
- fontSize
- rotation
- dataPrecision
- interpreted
- scalarReplacement
- autoResize [Obsolete]

D.37.4

D.37.4.1 Label (ViewObject parent)

- ViewObject parent - The parent to place the new label in. May also be a string containing the name of an existing ViewObject.

Creates a new label and places it in the ViewObject *parent*.

D.37.4.2 Label (Window window)

- Window window - The window to place the new label in. May also be a string containing the name of an existing Window.

Creates a new label and places it in the Window *window*.

D.37.4.3 void adjustSizeForText ( )

Adjusts the size of the label to fit the text.

D.37.4.4 string text

Contains the text contents of the label. This may include carriage returns (\n), scalar references of the form *[scalar_name]*, and some basic LaTeX.

D.37.4.5 string font

Used to set or get the current font used for the label.

D.37.4.6 number fontSize

Contains the size of the font used to draw the label.

D.37.4.7 number rotation

Contains the rotation angle of the label. Rotation is clockwise from *normal* and is stored in degrees.

D.37.4.8 number dataPrecision

Contains the number of digits of precision to display data values in the label. Restricted to a value between 0 and 16 inclusive.

D.37.4.9 boolean interpreted

Determines if the contents of the label should be interpreted in order to substitute LaTeX-like constructs. Default is true.

D.37.4.10 boolean scalarReplacement

Determines if the contents of the label should be interpreted in order to substitute variable reference constructs. Default is true.

D.37.4.11 boolean autoResize [Obsolete]

Automatically resizes the label to fit the text exactly. May trigger a slight move of the label. It will not automatically resize if the user resizes it manually via the size property or layout mode.

D.38 Class: Legend

A plot legend.

Inherits: BorderedViewObject

Collection class: ViewObjectCollection

D.38.1 Constructors:

- Legend (parent)
- Legend (window)

D.38.2 Methods:

- addCurve (curve)
- removeCurve (curve)

D.38.3 Properties:

- font
- fontSize
- textColor
- vertical
- curves
- title
- scaleLineWidth

D.38.4

D.38.4.1 Legend (ViewObject parent)

- ViewObject parent - The parent to place the new legend in. May also be a string containing the name of an existing ViewObject.

Creates a new legend and places it in the ViewObject *parent*.

D.38.4.2 Legend (Window window)

- Window window - The window to place the new legend in. May also be a string containing the name of an existing Window.

Creates a new legend and places it in the Window *window*.

D.38.4.3 void addCurve (string curve)

- string curve - The curve to be added. This can be either the name of the curve or a reference to the curve.

Adds the curve to the legend.

- Throws: TypeError - Throws this exception if the curve could not be found.

D.38.4.4 void removeCurve (string curve)

- string curve - The curve to be removed. This can be either the name of the curve or a reference to the curve.

Removes the curve from the legend.

- Throws: TypeError - Throws this exception if the curve could not be found.

D.38.4.5 string font

Used to set or get the current font used for the legend.

D.38.4.6 number fontSize

Contains the size of the font used to draw the legend.

D.38.4.7 string textColor

Contains the color of the text used to draw the legend.

D.38.4.8 boolean vertical

True if the legend entries are stacked vertically.

D.38.4.9 CurveCollection curves [Read-Only]

The list of curves shown in the legend.

D.38.4.10 string title

The title of the legend.

D.38.4.11 number scaleLineWidth

Contains the scale factor for the width of the lines in the legend, relative to the width of the lines in the actual plot.

D.39 Class: Line

A customizable line graphic.

Inherits: ViewObject

Collection class: ViewObjectCollection

D.39.1 Constructors:

- Line (parent)
- Line (window)

D.39.2 Properties:

- from
- to
- width
- capStyle
- lineStyle

D.39.3

D.39.3.1 Line (ViewObject parent)

- ViewObject parent - The parent to place the new line in. May also be a string containing the name of an existing ViewObject.

Creates a new line and places it in the ViewObject *parent*.

D.39.3.2 Line (Window window)

- Window window - The window to place the new line in. May also be a string containing the name of an existing Window.

Creates a new line and places it in the Window *window*.

D.39.3.3 Point from

The starting point of the line.

D.39.3.4 Point to

The ending point of the line.

D.39.3.5 number width

The width of the line.

D.39.3.6 number capStyle

The cap style for the line.

- 0 - Flat - may not cover the line ends (default)
- 1 - Box - may extend past line ends
- 2 - Rounded

D.39.3.7 number lineStyle

The style for the line.

- 0 - Solid line(default)
- 1 - Dashed line
- 2 - Dotted line
- 3 - Dash - dot line
- 4 - Dash - dot - dot line

D.40 Class: Matrix

Represents a matrix of any type in Kst. Some matrices are editable, while others are not.

Inherits: Object

Collection class: MatrixCollection

D.40.1 Constructors:

- Matrix ( )

D.40.2 Methods:

- resize (columns, rows)
- zero ( )
- value (column, row)
- setValue (column, row, value)
- update ( )

D.40.3 Properties:

- editable
- min
- max
- mean
- numNew
- rows
- columns

D.40.4

D.40.4.1 Matrix ( )

Default constructor creates an empty matrix.

D.40.4.2 void resize (number columns, number rows)

- number columns - The new number of rows in the matrix. Should be >= 1.
- number rows - The new number of columns in the matrix. Should be >= 1.

Resizes the matrix.

- Throws: GeneralError - Throws this exception if the matrix is not editable.

D.40.4.3 void zero ( )

Sets all values in the matrix to zero.

- Throws: GeneralError - Throws this exception if the matrix is not editable.

D.40.4.4 number value (number column, number row)

- number column - The column index of the matrix position to be read.
- number row - The row index of the matrix position to be read.

Gets the value of a matrix at the given column and row indices.

- Throws: GeneralError - Throws this exception if the indices are out of range.

D.40.4.5 void setValue (number column, number row, number value)

- number column - The column index of the matrix position to be set.
- number row - The row index of the matrix position to be set.
- number value - The value to set the specified matrix position to.

Sets the value of a matrix at the given column and row indices.

- Throws: GeneralError - Throws this exception if the matrix is not editable or the indices are out of range.

D.40.4.6 void update ( )

Updates the statistical values associated with a matrix.

- Throws: GeneralError - Throws this exception if the matrix is not editable.

D.40.4.7 number editable [Read-Only]

True if the matrix is editable by the user. This should be checked before attempting to modify a matrix in any way.

D.40.4.8 number min [Read-Only]

The value of the smallest sample in the matrix.

D.40.4.9 number max [Read-Only]

The value of the largest sample in the matrix.

D.40.4.10 number mean [Read-Only]

The mean value of all samples in the matrix.

D.40.4.11 number numNew [Read-Only]

The number of new samples.

D.40.4.12 number rows [Read-Only]

The number of rows in the matrix.

D.40.4.13 number columns [Read-Only]

The number of coluns in the matrix.

D.41 Class: MatrixCollection

An array of Matrices.

Inherits: Collection

D.41.1

D.42 Class: Object

Represents any object in Kst. This is an abstract object and may not be instantiated directly.

Collection class: ObjectCollection

D.42.1 Properties:

- tagName
- fullTagName

D.42.2

D.42.2.1 string tagName

A unique identifier for this data object.

D.42.2.2 string fullTagName [Read-Only]

The unique contextual identifier for this data object, contained with [...].

D.43 Class: ObjectCollection

An array of Objects. This could contain any objects that inherit from Object, but they are only immediately accessible in their pure base class form.

Inherits: Collection

D.43.1

D.44 Class: Picture

A customizable picture object.

Inherits: BorderedViewObject

Collection class: ViewObjectCollection

D.44.1 Constructors:

- Picture (parent)
- Picture (window)

D.44.2 Methods:

- load (url)

D.44.3 Properties:

- image
- url
- refreshTimer

D.44.4

D.44.4.1 Picture (ViewObject parent)

- ViewObject parent - The parent to place the new picture in. May also be a string containing the name of an existing ViewObject.

Creates a new picture and places it in the ViewObject *parent*.

D.44.4.2 Picture (Window window)

- Window window - The window to place the new picture in. May also be a string containing the name of an existing Window.

Creates a new picture and places it in the Window *window*.

D.44.4.3 boolean load (string url)

- string url - The URL to retrieve the image from.

Load an image from a give url. Returns true on success.

D.44.4.4 Image image

The raw image that is being displayed. You can take a copy of this, manipulate it, then set it back to change the image.

D.44.4.5 string url [Read-Only]

The URL of the image if it was loaded from a file and has not been modified.

D.44.4.6 number refreshTimer

A timer to automatically refresh the picture. Value of 0 means the timer is disabled. Units are in seconds.

D.45 Class: Plot

This class represents a plot in Kst.

Inherits: BorderedViewObject

Collection class: PlotCollection

D.45.1 Constructors:

- Plot (window)

D.45.2 Methods:

- createLegend ( )

D.45.3 Properties:

- curves
- legend
- topLabel
- xAxis
- yAxis
- title
- tied

D.45.4

D.45.4.1 Plot (Window window)

- Window window - The window to place the new plot in. May also be a string containing the name of an existing Window.

Creates a new plot and places it in the Window *window*.

D.45.4.2 Legend createLegend ( )

Creates a Legend for this plot. If a legend already exists then that one is returned.

D.45.4.3 CurveCollection curves [Read-Only]

A list of all the Curves used by the plot.

D.45.4.4 Legend legend [Read-Only]

The Legend for this plot. May be null.

D.45.4.5 string topLabel

The top label for this plot.

D.45.4.6 Axis xAxis [Read-Only]

The X-axis for this plot.

D.45.4.7 Axis yAxis [Read-Only]

The Y-axis for this plot.

D.45.4.8 PlotLabel title [Read-Only]

The title for this plot.

D.45.4.9 boolean tied

True if the plot zoom state is tied.

D.46 Class: PlotCollection

An array of Plots.

Inherits: Collection

D.46.1

D.47 Class: PlotLabel

A class representing a plot label.

D.47.1 Properties:

- text
- font
- fontSize
- justification
- dataPrecision

D.47.2

D.47.2.1 string text

Contains the text contents of the label. This may include carriage returns (\n), scalar references of the form *[scalar_name]*, and some basic LaTeX.

D.47.2.2 string font

Used to set or get the current font used for the label.

D.47.2.3 number fontSize

Contains the size of the font used to draw the label.

D.47.2.4 number justification

The horizontal justification for the label. This is a bit field. The values are as follows:

- 0 - Justify None
- 1 - Justify Left / Top
- 2 - Justify Right / Bottom
- 3 - Justify Center

D.47.2.5 number dataPrecision

Contains the number of digits of precision to display data values in the label. Restricted to a value between 0 and 16 inclusive.

D.48 Class: Plugin

This class represents a data plugin in Kst. It can be instantiated to create a new instance of a plugin in the Kst data collection.

Inherits: DataObject

Collection class: PluginCollection

D.48.1 Constructors:

- Plugin ([module])

D.48.2 Methods:

- setInput (index, input)
- validate ( )

D.48.3 Properties:

- inputs
- outputs
- module
- lastError
- valid

D.48.4

D.48.4.1 Plugin ([PluginModule module])

- PluginModule module - The plugin module to use for this plugin. [OPTIONAL]

Main constructor for the Plugin class. New plugins are typically invalid until various properties have been set.

D.48.4.2 void setInput (number index, Object input)

- number index - or name of the input argument to be set.
- Object input - argument.

Sets the input argument at the specified index or name.

D.48.4.3 void validate ( )

Validates the plugin.

D.48.4.4 ObjectCollection inputs

The list of inputs to the plugin.

D.48.4.5 ObjectCollection outputs

The list of outputs to the plugin.

D.48.4.6 PluginModule module

The library or module that is used for data processing by this plugin object.

D.48.4.7 string lastError [Read-Only]

A string containing details of the last error that occurred while running the plugin.

D.48.4.8 boolean valid [Read-Only]

True if this plugin object is valid. If false, there is probably an invalid setting somewhere that needs to be corrected before it can be used. Some errors include invalid inputs or outputs, or an invalid or missing PluginModule.

D.49 Class: PluginCollection

An array of Plugins.

Inherits: Collection

D.49.1

D.50 Class: PluginIO

An object representing an input or output for a given plugin module.

Collection class: PluginIOCollection

D.50.1 Properties:

- name
- type
- subType
- typeObject
- description
- defaultValue

D.50.2

D.50.2.1 string name [Read-Only]

The name of the input or output.

D.50.2.2 string type [Read-Only]

The data type of the input or output.

D.50.2.3 string subType [Read-Only]

The sub-type of the input or output, if needed. For instance, a *table* may have a subtype of *float*.

D.50.2.4 string typeObject [Read-Only]

The object type of the input or output. The result will be one of:

- Vector
- Scalar
- String
- Unknown - error condition

D.50.2.5 string description [Read-Only]

A description of the meaning of the input or output.

D.50.2.6 string defaultValue [Read-Only]

The default value for this input or output, if specified by the plugin.

D.51 Class: PluginIOCollection

An array of Plugin inputs or outputs.

Inherits: Collection

D.51.1

D.52 Class: PluginManager

A reference to Kst's plugin manager subsystem.

D.52.1 Constructors:

- PluginManager ( )

D.52.2 Properties:

- modules

D.52.3

D.52.3.1 PluginManager ( )

Creates a new PluginManager referring to the Kst plugin manager.

D.52.3.2 PluginModuleCollection modules [Read-Only]

A list of all PluginModules available.

D.53 Class: PluginModule

Represents a plugin module in Kst. This is not an instance of a plugin, but a reference to the actual plugin itself. It is typically used to instantiate a plugin object.

Collection class: PluginModuleCollection

D.53.1 Properties:

- usesLocalData
- isFit
- isFilter
- name
- readableName
- author
- description

- version
- inputs
- outputs

D.53.2

D.53.2.1 boolean usesLocalData [Read-Only]

If true, the plugin makes use of persistant local storage.

D.53.2.2 boolean isFit [Read-Only]

If true, this plugin is a *fit* for a curve.

D.53.2.3 boolean isFilter [Read-Only]

If true, this plugin is a *filter* for a vector.

D.53.2.4 string name [Read-Only]

Represents the short name of the plugin.

D.53.2.5 string readableName [Read-Only]

Contains a long name for the plugin suitable for display in a user interface.

D.53.2.6 string author [Read-Only]

Contains the name of the author of the plugin.

D.53.2.7 string description [Read-Only]

Contains a text description of what the plugin is and/or does.

D.53.2.8 string version [Read-Only]

Contains the version number or string of the plugin.

D.53.2.9 PluginIOCollection inputs [Read-Only]

A list of all the inputs for the plugin.

D.53.2.10 PluginIOCollection outputs [Read-Only]

A list of all the outputs of the plugin.

D.54 Class: PluginModuleCollection

An array of PluginModules.

Inherits: Collection

D.54.1

D.55 Class: Point

Represents a cartesian co-ordinate of the form (x,y).

D.55.1 Constructors:

- Point ( )
- Point (x, y)

D.55.2 Properties:

- x
- y

D.55.3

D.55.3.1 Point ( )

Creates a new point object with value (0.0, 0.0).

D.55.3.2 Point (number x, number y)

- number x - The X value.
- number y - The Y value.

Creates a new point object with value (x, y).

D.55.3.3 number x

The value of the x co-ordinate.

D.55.3.4 number y

The value of the y co-ordinate.

D.56 Class: PowerSpectrum

This class represents a power spectrum (PSD) object in Kst.

Inherits: DataObject

Collection class: PowerSpectrumCollection

D.56.1 Constructors:

- PowerSpectrum (vector, freq [, average [, len [, apodize [, removeMean]]]])

D.56.2 Properties:

- vector
- length
- output
- removeMean
- average
- apodize
- apodizeFn
- frequency
- xVector
- yVector
- vUnits
- rUnits
- interpolateHoles

D.56.3

D.56.3.1 PowerSpectrum (Vector vector, number freq [, boolean average [, number len [, boolean apodize [, boolean removeMean]]]])

- Vector vector - The vector to use as input to the power spectrum.
- number freq - The frequency for the power spectrum.
- boolean average - [OPTIONAL]
- number len - The base 2 logarithm of the length of the power spectrum. Should be an integer >= 4. [OPTIONAL]
- boolean apodize - If true, sharp discontinuities are removed. [OPTIONAL]
- boolean removeMean - True if the mean should be removed before performing the transform. [OPTIONAL]

Creates a new power spectrum (PSD) object in Kst. Units are *V* and *Hz* by default.

D.56.3.2 Vector vector

The input vector for the power spectrum.

D.56.3.3 number length

Contains the base 2 logarithm of the length to be used for interleaved averaging. Should be an integer >= 4.

D.56.3.4 number output

the normalization for the transform:

- 0 - amplitude spectral density
- 1 - power spectral density
- 2 - amplitude spectrum
- 3 - power spectrum

D.56.3.5 boolean removeMean

True if the mean should be removed before performing the transform.

D.56.3.6 boolean average

True if the power spectrum should be calculated using an interleaved average.

D.56.3.7 boolean apodize

If true, sharp discontinuities are removed.

D.56.3.8 number apodizeFn

The apodization function to be used in the case that apodize is true:

- 0 - default
- 1 - bartlett
- 2 - blackman
- 3 - connes
- 4 - cosine
- 5 - gaussian
- 6 - hamming
- 7 - hann
- 8 - welch
- 9 - uniform

D.56.3.9 number frequency

Contains the sampling rate of the power spectrum.

D.56.3.10 Vector xVector [Read-Only]

The X-axis vector for the power spectrum.

D.56.3.11 Vector yVector [Read-Only]

The Y-axi vector for the power spectrum.

D.56.3.12 string vUnits

A string containing the units for the vector.

D.56.3.13 string rUnits

A string containing the units for the rate.

D.56.3.14 boolean interpolateHoles

True if the transform should automatically interpolate over missing data values.

D.57 Class: PowerSpectrumCollection

An array of PowerSpectrums.

Inherits: Collection

D.57.1

D.58 Class: Scalar

Represents a scalar value (a number) in Kst.

Inherits: Object

Collection class: ScalarCollection

D.58.1 Constructors:

- Scalar ( )
- Scalar (value)

D.58.2 Properties:

- value

D.58.3

D.58.3.1 Scalar ( )

Default constructor. Creates a new scalar with value 0.0.

D.58.3.2 Scalar (number value)

- number value -

Creates a new scalar with the specified value.

D.58.3.3 number value

Used to access or modify the value of a scalar.

D.59 Class: ScalarCollection

An array of Scalars.

Inherits: Collection

D.59.1

D.60 Class: Size

Represents a two-dimensional size of the form (w, h).

D.60.1 Constructors:

- Size ( )
- Size (w, h)

D.60.2 Properties:

- w
- h

D.60.3

D.60.3.1 Size ( )

Creates a new size object with value (0.0, 0.0).

D.60.3.2 Size (number w, number h)

- number w - The width.
- number h - The height.

Creates a new size object with value (w, h).

D.60.3.3 number w

The width.

D.60.3.4 number h

The height.

D.61 Class: Spectrogram

This class represents a spectrogram (CSD) object in Kst.

Inherits: DataObject

Collection class: SpectrogramCollection

D.61.1 Constructors:

- Spectrogram (vector, freq [, average [, len [, apodize [, removeMean]]]])

D.61.2 Properties:

- vector
- length
- output
- removeMean
- average
- apodize
- apodizeFn
- frequency
- matrix
- vUnits
- rUnits
- windowSize
- interpolateHoles

D.61.3

D.61.3.1 Spectrogram (Vector vector, number freq [, boolean average [, number len [, boolean apodize [, boolean removeMean]]]])

- Vector vector - The vector to use as input to the spectrogram.
- number freq - The frequency for the spectrogram.
- boolean average - [OPTIONAL]
- number len - The base 2 logarithm of the length of the power spectrum. Should be an integer >= 4. [OPTIONAL]
- boolean apodize - If true, sharp discontinuities are removed. [OPTIONAL]
- boolean removeMean - True if the mean should be removed before performing the transform. [OPTIONAL]

Creates a new power spectrum (PSD) object in Kst. Units are *V* and *Hz* by default.

D.61.3.2 Vector vector

The input vector for the spectrogram.

D.61.3.3 number length

Contains the base 2 logarithm of the length of the spectrogram. Should be an integer >= 4.

D.61.3.4 number output

the normalization for the transform:

- 0 - amplitude spectral density
- 1 - power spectral density
- 2 - amplitude spectrum
- 3 - power spectrum

D.61.3.5 boolean removeMean

True if the mean should be removed before performing the transform.

D.61.3.6 number average

D.61.3.7 boolean apodize

If true, sharp discontinuities are removed.

D.61.3.8 number apodizeFn

The apodization function to be used in the case that apodize is true:

- 0 - default
- 1 - bartlett
- 2 - blackman
- 3 - connes
- 4 - cosine
- 5 - gaussian
- 6 - hamming
- 7 - hann
- 8 - welch
- 9 - uniform

D.61.3.9 number frequency

Contains the sampling rate of the spectrogram.

D.61.3.10 Matrix matrix [Read-Only]

The matrix for the spectrogram.

D.61.3.11 string vUnits

A string containing the units for the vector.

D.61.3.12 string rUnits

A string containing the units for the rate.

D.61.3.13 number windowSize

The window size of the spectrogram.

D.61.3.14 boolean interpolateHoles

True if the transform should automatically interpolate over missing data values.

D.62 Class: SpectrogramCollection

An array of Spectrograms.

Inherits: Collection

D.62.1

D.63 Class: String

Represents a string in Kst. Note that this is *not* analagous to a string in JavaScript, and may not be directly interchanged as such. A Kst String is an object internal to Kst and is subject to update scheduling, sharing, and other policies of the Kst core. However the *contents* of a String may be manipulated with JavaScript string manipulation functions.

Inherits: Object

Collection class: StringCollection

D.63.1 Constructors:

- String ( )
- String (value)

D.63.2 Properties:

- value

D.63.3

D.63.3.1 String ( )

Default constructor - creates an empty String.

D.63.3.2 String (string value)

- string value -

Creates a new string with the specified value.

D.63.3.3 string value

The value of the String.

D.64 Class: StringCollection

An array of Kst Strings.

Inherits: Collection

D.64.1

D.65 Class: TimeInterpretation

A class representing a time interpretation. It is used to convert values from one set of units to display in another.

D.65.1 Properties:

- active
- axisType
- input
- output

D.65.2

D.65.2.1 boolean active

True if the axis is being interpreted.

D.65.2.2 string axisType [Read-Only]

The type of axis this interpretation applies to.

D.65.2.3 number input

The format of the input to the interpretation. Must be one of:

- 0 - Standard C Time
- 1 - TAI
- 2 - JD
- 3 - MJD
- 4 - RJD
- 5 - JY
- 6 - TAI (ns)
- 7 - TAI (2^-16s)

D.65.2.4 number output

The format to convert the interpretation to. Must be one of:

- 0 - DD/MM/YY HH:MM:SS
- 1 - YY/MM/DD HH:MM:SS
- 2 - JD
- 3 - MJD
- 4 - RJD
- 5 - JY
- 6 - Localized Short Date
- 7 - Localized Long Date

D.66 Class: Vector

Represents a vector of any type in Kst. Some vectors are editable, while others are not. This class behaves just as a JavaScript array, but has additional properties and methods.

Inherits: Object

Collection class: VectorCollection

D.66.1 Constructors:

- Vector ( )
- Vector (array)

D.66.2 Methods:

- resize (size)
- interpolate (i, n)
- zero ( )
- update ( )
- valueAt (index)

D.66.3 Properties:

- length
- min
- max
- mean
- numNew
- numShifted
- editable
- numNaN
- array

D.66.4

D.66.4.1 Vector ( )

Default constructor creates an empty, editable vector.

D.66.4.2 Vector (Array array)

- Array array - The array to be copied.

Creates a vector that is a copy of array.

D.66.4.3 void resize (number size)

- number size - The new size of the vector. Should be >= 2.

Resizes the vector.

- Throws: GeneralError - Throws this exception if the vector is not editable.

D.66.4.4 number interpolate (number i, number n)

- number i - The sample number to obtain the interpolated value of.
- number n - The number of samples to interpolate to.

Interpolates the vector to *n* samples and returns the interpolated value of sample *i*.

D.66.4.5 void zero ( )

Sets all values in the vector to zero.

- Throws: GeneralError - Throws this exception if the vector is not editable.

D.66.4.6 void update ( )

Updates the statistical values associated with a vector.

- Throws: GeneralError - Throws this exception if the vector is not editable.

D.66.4.7 void valueAt (number index)

- number index - The sample number to obtain the value of. A negative value of -n returns the n'th sample from the end of the vector..

Returns the value at the given index.

D.66.4.8 number length [Read-Only]

The number of samples in the vector.

D.66.4.9 number min [Read-Only]

The value of the smallest sample in the vector.

D.66.4.10 number max [Read-Only]

The value of the largest sample in the vector.

D.66.4.11 number mean [Read-Only]

The mean value of all samples in the vector.

D.66.4.12 number numNew [Read-Only]

D.66.4.13 number numShifted [Read-Only]

D.66.4.14 number editable [Read-Only]

True if the vector is editable by the user. This should be checked before attempting to modify a vector in any way.

D.66.4.15 number numNaN [Read-Only]

The number of NaN values in the vector.

D.66.4.16 Array array [Read-Only]

The vector as an Array.

D.67 Class: VectorCollection

An array of Vectors.

Inherits: Collection

D.67.1

D.68 Class: VectorView

Represents a vectorView.

Inherits: DataObject

D.68.1 Constructors:

- VectorView (x, y)

D.68.2 Properties:

- xVector
- yVector
- flagVector
- xMin
- xMax
- yMin
- yMax
- useXMin
- useXMax
- useYMin
- useYMax
- interpolateTo

D.68.3

D.68.3.1 VectorView (Vector x, Vector y)

- Vector x - The x-vector.
- Vector y - The y-vector.

Creates a new vector view with the given x-vector and y-vector.

D.68.3.2 Vector xVector

the x-vector.

D.68.3.3 Vector yVector

the y-vector.

D.68.3.4 Vector flagVector

the flag-vector. Those points corresponding to non-zero values of the flag vector are removed from the output vectors.

D.68.3.5 Scalar xMin

the value for the minimum x-value for the view range.

D.68.3.6 Scalar xMax

the value for the maximum x-value for the view range.

D.68.3.7 Scalar yMin

the value for the minimum y-value for the view range.

D.68.3.8 Scalar yMax

the value for the maximum y-value for the view range.

D.68.3.9 boolean useXMin

If true, the minimum x-value is respected.

D.68.3.10 boolean useXMax

If true, the maximum x-value is respected.

D.68.3.11 boolean useYMin

If true, the minimum y-value is respected.

D.68.3.12 boolean useYMax

If true, the maximum y-value is respected.

D.68.3.13 number interpolateTo

the value for the maximum y-value for the view range. Must be one of:

- 0 - interpolate to the x-vector
- 1 - interpolate to the y-vector
- 2 - interpolate to the high resolution vector
- 3 - interpolate to the low resolution vector

D.69 Class: ViewObject

Represents any object in a Kst window. This is an abstract object and may not be instantiated directly.

Inherits: Object

Collection class: ViewObjectCollection

D.69.1 Methods:

- resize (size)
- move (pos)
- findChild (pos)
- convertTo (type)
- raiseToTop ( )
- lowerToBottom ( )
- raise ( )
- lower ( )
- remove ( )

D.69.2 Properties:

- position
- size
- transparent
- onGrid
- columns
- color
- backgroundColor
- maximized
- minimumSize
- type
- children

D.69.3

D.69.3.1 ViewObject resize (Size size)

- Size size - The new size for the object.

Resizes the object to the given size, if possible.

D.69.3.2 ViewObject move (Point pos)

- Point pos - The position to move the object to.

Moves the object to a new position, if possible.

D.69.3.3 ViewObject findChild (Point pos)

- Point pos - The relative position to search.

Finds the topmost child of this view object at the given point. Returns null if there is no child there.

D.69.3.4 ViewObject convertTo (string type)

- string type - The type to attempt to convert this object to.

Attempts to convert this ViewObject to a different type. The object must be derived from this type at some level. Null is returned if it is not possible to convert to the requested type.

D.69.3.5 void raiseToTop ( )

Raises the object to the top of the z-order.

D.69.3.6 void lowerToBottom ( )

Lowers the object to the bottom of the z-order.

D.69.3.7 void raise ( )

Raises the object one place in the z-order.

D.69.3.8 void lower ( )

Lowers the object one place in the z-order.

D.69.3.9 void remove ( )

Deletes the object.

D.69.3.10 Point position

The location of the object relative to its parent.

D.69.3.11 Size size

The size of the object in pixels.

D.69.3.12 boolean transparent

True if this object is transparent. Not all objects support transparency.

D.69.3.13 boolean onGrid

True if the children of this object are on a grid.

D.69.3.14 number columns

The number of columns the children are organized into. If this value is modified, *onGrid* is set to true.

D.69.3.15 string color

The foreground color for this object.

D.69.3.16 string backgroundColor

The background color for this object.

D.69.3.17 boolean maximized

If true, this object is maximized relative to its parent. This is a temporary state whereby the plot uses all the space available in the parent but can be restored to *normal* size again.

D.69.3.18 Size minimumSize [Read-Only]

The minimum size of the view object.

D.69.3.19 string type [Read-Only]

A string containing the type of this view object.

D.69.3.20 ViewObjectCollection children [Read-Only]

The list of all children of this view object in z-order from the furthest back to the closest to the top. This is a reference to a collection which may be modified, but cannot be *set*.

D.70 Class: ViewObjectCollection

An array of ViewObjects.

Inherits: Collection

D.70.1

D.71 Class: Window

An object that represents a reference to a Kst window or tab.

Collection class: WindowCollection

D.71.1 Constructors:

- Window ([name])

D.71.2 Methods:

- repaint ( )
- close ( )

D.71.3 Properties:

- name
- plots
- view
- columns

D.71.4

D.71.4.1 Window ([string name])

- string name - A name for the new window. [OPTIONAL]

Creates a new window and object to refer to it.

D.71.4.2 void repaint ( )

Repaints the window.

D.71.4.3 void close ( )

Closes the window.

D.71.4.4 string name

The name of the window this object refers to.

D.71.4.5 PlotCollection plots

The list of plots contained in this window, flattened.

D.71.4.6 ViewObject view

The view object for this window.

D.71.4.7 number columns

The number of columns the children are organized into. If this value is modified, *onGrid* is set to true.

D.72 Class: WindowCollection

An array of Kst Windows.

Inherits: Collection

D.72.1

Appendix E

The Kst DCOP Interface

Kst provides an interface that acts as a client using the the KDE Desktop COmmunications Protocol (DCOP). The set of functions provided in the `KstIface` interface allows for easy control of Kst by other applications or scripts. For more information on DCOP, refer to this webpage.

Appendix F

Installation

The following sections contain detailed instructions for obtaining, compiling, and installing Kst along with any optional libraries or packages. If you use a package management system, you may use it to install Kst instead.

F.1 Obtaining Kst

Kst is part of the extragear/graphics module of KDE. You can also download the latest version separately from the Kst website.

The current version of Kst is also available via SVN. It can be downloaded anonymously with the following command:

```
svn co svn://anonsvn.kde.org/home/kde/branches/extragear/kde3/graphics
```

You can also browse the Kst source code using the KDE WebSVN repository at http://websvn.kde.org/branches/extragear/kde3/graphics/kst/.

More information about accessing KDE SVN repositories can be found at http://developer.kde.org/documentation/tutorials/subversion/.

F.2 Installing Kst from Source

The following sections detail how to compile and install Kst from source code.

F.2.1 Requirements

Kst v1.10.0 requires the KDE 3.4 libraries and Qt™ 3.3. Almost all major Linux® distributions have packages for KDE and Qt™. Since Qt™ is a dependency of KDE, having the required version of KDE should be sufficient. It is recommended that you refer to your particular distribution's documentation for KDE library installation. Note that the entire KDE desktop environment is not required — in most cases, look for packages called `kdebase-*` and `kdelibs-*`. Alternatively, more information is available on the KDE website.

F.2.2 Optional Libraries and Packages

The following libraries and packages are optional, but provide full plugin and data source functionality in Kst if installed. Installation/Download instructions for the packages are given on the linked websites.

Note

You may already have some or all of these libraries and packages installed, in which case there is no need to reinstall them. Also, check first to see if pre-compiled binaries for the libraries and packages exist in your distribution’s package management system, as it may be easier to install these packages.

Library	Description
GSL	The GNU Scientific Library (GSL) provides a variety of mathematical operations for the built-in Kst plugins. Having GSL installed enables a variety of plugins such as linear and non-linear fitting, correlation, and interpolation.
CFITSIO	CFITSIO provides support for reading and writing to the FITS (Flexible Image Transport System) data format. This is required by all of the datasources which use FITS (currently HEALPix, LFIIO, and WMAP)
CDF	The Common Data Format (CDF) library is required to read files with the CDF datasource
netCDF	The network Common Data Form (netCDF) library is required to read files with the netCDF datasource
muParser	The muParser library is required to use the non-linear fitting plugin.

F.2.3 Compiling and Installing

Once all of the prerequisites have been met on your system you can compile and install Kst. It is assumed that you have already downloaded the Kst source package. If not, please see Obtaining Kst.

The simplest way to install Kst is:

```
tar -zxvf kst-version.tar.gz
cd kst-version
./configure --prefix=`kde-config --prefix`
make
make install
```

replacing `version` with the version of Kst you have.

Important

Note that **kde-config --prefix** is surrounded by backquotes. It is very important that the **kde-config** utility returns the correct path (most likely `/usr`), otherwise Kst will *not* install properly. You can double-check by running it directly:

```
kde-config --prefix
```

Ensure that a non-null path is displayed.

You can read through the messages displayed by the **configure** script — if it reports any missing libraries (such as GSL) that should not be missing, please go back and ensure that the libraries and header files have been correctly installed in the appropriate locations.

Appendix G

Equation Expressions in Kst

Kst equations are expressed in terms of *built in operators, functions, constants*, scalars and vectors in the current Kst session.

Some rules should be noticed when you enter equation expressions in either UI textbox or KstScript:

- Vector names must be included in square brackets like this: [`VectorName`].
- Scalar names should also be included in square brackets: [`ScalarName`]
- Numbers don't need to be included in square brackets. PI and e should be counted as numbers, and all other built-in constants listed in table G.2 are scalars. i.e. you should include them in square brackets.
- Values in scientific notation must be entered in the form: `m`E`x` or `m`e`x`, where `m` is the mantissa and `x` is the exponent.

  For example, one way to enter the value 15000 in scientific notation is `1.5E4`.

The following is a list of all built-in *operators, functions and constants* used to express Equations in Kst.

G.1 Operators

G.1.1 Arithmetic Operators

+ Addition operator

- Subtraction operator

***** Multiplication operator

/ Division operator

% Remainder operator

ˆ Raise to power operator

G.1.2 Logical and Relational Operators

Logical operators are used to evaluate expressions to true(1) or false(0) value. In Kst, 0 is false, and all other numbers represent true value.

& Bitwise And operator

It compares the corresponding bits of two operands in their binary representation of equal length. If both values of the pair are 1, the resulting bit value is 1, 0 otherwise.

| Bitwise Or operators.

It compares the corresponding bits of two numbers in their binary representation of equal length. If both values of the pair are 0, the resulting bit is 0, 1 otherwise.

&& Logical operator And.

It takes two numbers as boolean variables, and results 1 (true) if both operands are non-zero, 0 (false) otherwise. See an example below.

Expression	Results
1 && 3	1
3 && 0	0

|| Logical Or operators

It takes two operands and results 0 (false) if both operands are 0, 1 (true) otherwise. See an example below

<table>
<tr><th>Expression</th><th>Results</th></tr>
<tr><td>2 || 0</td><td>1</td></tr>
<tr><td>2 || 1</td><td>1</td></tr>
<tr><td>0 || 0</td><td>0</td></tr>
</table>

! Logical not

< Less than

<= Less than and equal to

== Equal to

> Greater than

>= Greater than and equal to

!= Not equal to

G.2 Built-in Functions

G.3 Built-in Variables

G.4 Built-in Constants

Function	Description
ABS()	returns the absolute value of the argument
SQRT()	returns the square root value of the argument
CBRT()	returns the cubic root value of the argument
SIN()	sine function
COS()	cosine function
TAN()	tangent function
ASIN()	acrsine function
ACOS()	arccosin function
ATAN()	arctangent function
SEC()	secant function
CSC()	cosecant function
COT()	cotangent function
SINH()	hyperbolic sine function
COSH()	hyperbolic cosine function
TANH()	hyperbolic tangent function
EXP()	exponential function
LN()	natural logarithmic function
LOG()	base 10 logarithmic function

Table G.1: Kst Built-in Function

Parent Type	Name	Description
Data Source	frames	Number of frames in the data source
Data Vector	First	First value in the vector
Data Vector	Last	Last value in the vector
Data Vector	Max	Largest value in the vector
Data Vector	MaxIndex	Index of the largest value in the vector
Data Vector	Mean	Mean of all values in the vector
Data Vector	Min	Smallest value in the vector
Data Vector	MinIndex	Index of the smallest value in the vector
Data Vector	MinPos	Smallest positive and non-zero value in the vector
Data Vector	NS	Number of values in the vector
Data Vector	Rms	Quadratic mean of all values in the vector
Data Vector	Sigma	Standard deviation of all values in the vector
Data Vector	Sum	Sum of all values in the vector
Data Vector	SumSquared	Sum of squares of all values in the vector
Plot	XMax	Maximum x-value visible in the plot
Plot	XMin	Minimum x-value visible in the plot
Plot	YMax	Maximum y-value visible in the plot
Plot	YMin	Minimum y-value visible in the plot

Table G.2: Kst Built-in Variables

Name	**Value**	**Description**
PI or C_PI or CONST_PI	3.14159	
e	2.71282	
C_D2R	0.0174533	1 deg = 0.0174533 rad; conversion factor for degrees to radians
C_R2D	57.2958	1 rad = 57.2958 deg; conversion factor for radians to degrees

Table G.3: Kst Built-in Constants