

Cartan - A Program for Interactive Model Fitting, Error Analysis and Visualisation

K.A. Hawick

Institute of Information and Mathematical Sciences, Massey University
 Albany, North Shore 102-904, Auckland, New Zealand

Email: k.a.hawick@massey.ac.nz

Tel: +64 9 414 0800 Fax: +64 9 441 8181

July 2006, Revised October 2010

Abstract

Although many excellent data plotting programs such as Gnuplot exist, as do programs for data fitting and parameter estimation, it is valuable to combine these two capabilities in a single extensible program framework. The Cartan program makes use of interactive data-loading or data-synthesis, data-fitting and data-presentation ideas and employs Java and Swing software technologies to support many of the x-y (CARTesian-ANalysis) data manipulation techniques familiar to experimental scientists. A particular feature of the program is its inbuilt and interactive use of first-order calculus methods to track the propagation of data uncertainties alongside the independent and dependent variables themselves. The program was originally intended as a teaching aid but can be used as a science productivity tool. A discussion of the code design and architecture is given alongwith a review of ideas on data formats, data parsing and the management and parameter and model estimation issues. Some ideas on how to extend such a framework to handle project-specific data models and formats are also presented.

Keywords: x-y data; model fitting; plotting; error analysis; data uncertainties; interactive models.

Contents

1	Introduction	2	3	Model Fitting and Reduction	3
			3.1	Lin-Lin Plots	4
			3.2	Log-Log Plots	4
			3.3	Log-Lin Plots	4
			3.4	General Linear Model Fits and Non-Linear Fits	5
4	Cartan Implementation	5			
			4.1	Data Manager Window	5
			4.2	Mouse & Keyboard Controls	6
			4.3	Data Models	6
			4.4	Data File Formats	7
			4.5	File Menus	7
			4.6	XY Viewing Controls	7
			4.7	Display Control Pane	8
			4.8	Table View	9
			4.9	Render Window	9
			4.10	Log Window	9
			4.11	Synthesis Window	10
			4.12	Model Manager Window	11
			4.13	Help Window	12
5	Discussion	12			
6	Conclusions	13			

1 Introduction

Like many scientists I was taught how to plot experimental data in first year university laboratory sessions. In those days (early 1980s) we still used graph paper, but home computers were starting to make chunky character plotting of data possible – as undergraduates we did not obtain access to the expensive mainframe and pen plotters until later. Plotting some measured property against an independent variable is therefore ingrained in many of us as the single most important act of experimental data analysis that it is hard to understand why more effort has not gone into software tools available to do this. One can plot charts using Microsoft Excel – and contrary to apparent wide spread practice it is possible to plot error bars with that tool. Later as a postgraduate I was introduced to Unix plot and variants such as xplot [1] and psplot. In the usual Unix style of a simple tool that does one specific thing very well, they read a plain text file or files of data and make a perfectly acceptable x-y plot, including error bars and a variety of symbols. You had to write your own programs to do fitting, or to generate model data, separately.

The Gnuplot program [2] is a pretty good replacement for those tools and it does a degree of other model generation too, but it seemed to me that fitting and analysing experimental data ought to be more interactive and furthermore that modern software technology and desktops ought to be quite capable of doing a least squares fit in real time – at least for the size of x-y data set that one can understand in one’s head. Some other excellent tools such as Matlab [3] or Mathematica [4] provide wholistic problem solving environments that include data-plotting, although those tools are not necessarily cheap and in fact do not do quite the interactive sort of data manipulation that I find useful for numerical simulation experiments. Other tools like R [5] provide good statistical analysis mechanisms [6], but again, not quite in the interactive form I tend to use.

The Cartan program was designed as a wholly interactive tool, that would allow the user to load around a dozen or so x-y data sets from plain ordinary text files, with or without error bar uncertainty data and support all the usual transforms and fitting operations that I was trained to do manually. I was particularly motivated to do this as I moved a plastic ruler across a printout for the umpteenth time to convince a colleague some plotted data lay on a straight line. In addition, I was trying to analyse Binder cumulants from phase transitions in Ising model [7] simulations and I realised that my colleagues and I had already invested

a huge amount of computer time generating the raw data and that therefore it would really be worthwhile investing a bit of effort into an interactive analysis tool that might guide us where to look in further numerical experiments.

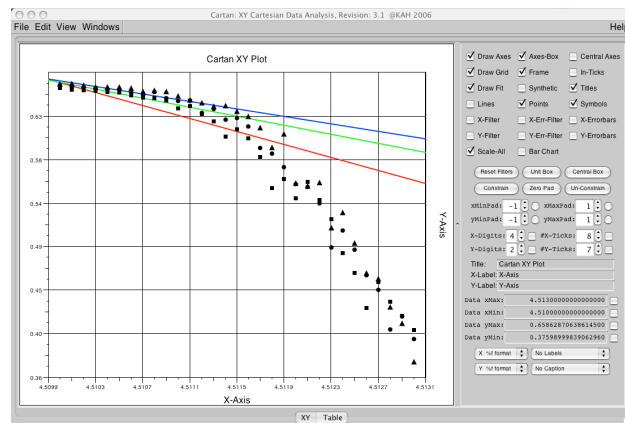


Figure 1: The Main Cartan viewing DisplayPanel and ControlPanel showing a straight line fit to part of some Binder Cumulant data loaded from three separate files.

Figure 1 shows three curves of the Binder cumulant – each point has been calculated from a simulation taking many days – and the point of interest for this finite size scaling analysis is the intersection point (and its uncertainty) between the three cumulant curves. Cartan has built into it the reporting capability to do least-squares fits and compute this intersection point.

Many experimental data sets have problem data points in them. In older times these were often due to incorrect human reading of instruments or just plain writing data down wrongly. Even with modern numerical simulations, sometimes a job crashes, a file gets corrupted, and the automatic data collection you thought was foolproof turns out not to be. At any rate, there is still a degree of human judgement often required to decide what data a points to include or reject for subsequent more careful examination. An incredibly important aspect of data analysis is of course to make ongoing judgements during an experiment of where to look - what parameter space needs refining, what area requires more statistical investigation with bigger system sizes or more independent measurements to average over.

The Cartan Program is a rather personal tool developed for my own use but was also heavily used by my graduate students in “playing around” with data - particularly incomplete data as it comes out of sim-

ulation programs - to guide the experiments. It is not intended for wide release, but I believe the ideas documented here may be of use to other tool makers.

Cartan has model fitting capabilities and data selection and editing and generation facilities as well as basic data analysis and visualisation tools built in. It supports quite a lot of the ordinary operations an experimental scientist uses in trying to understand what to make of data. Cartan does allow reasonable output in the form of a nicely laid out and annotated plot with output options including the better known graphics image formats like PNG or PostScript. It is capable of managing as many data sets as will fit on the screen - typically up to a couple of dozen, and it can manage individual data set sizes of up to around 10,000 data points. It was not designed to handle larger sets than these - and in practice larger sets would typically be pre-processed or reduced down by some other analysis program anyway.

Cartan can be used in combination with Gnuplot - which can often do more complicated renderings and annotations for a final presentation plot. Cartan's power is in its quick and easy interactive use and it will likely be maintained and extended by me for continued use in this niche. The main purpose of this present document is as an aide memoire to me but also as a sort of design philosophy and user manual for students using it or just doing quantitative analysis of experimental data, particularly as generated by simulations.

This article is structured as follows: Section 2 describes the design philosophy and need for the Cartan tool. In Section 3 a description of fitting and error combination formulae using first-order calculus are given. A detailed presentation of the Cartan architecture and its components is given in Section 4 with some discussion of the implementation in Section 5 and overall conclusions and areas for further development given in Section 6.

2 Data Plotting

Generally when I do a numerical experiment my simulation program spits out a series of measurements which might be a time series or of some property averaged over different systems sizes or with different model parameters. At any rate I will usually be trying to see how some dependent variable (y) varies with a dependent control parameter (x) and hopefully I will have some definite means of estimating the uncertainty in y and possibly also an uncertainty in x . I

might want to see my data visually and decide what points look "clean" and which might indicate (from a large uncertainty) that I need more measurements, or to see where the gaps I need to fill in in my experimental procedure are.

The Cartan design philosophy is to support this interactive comparison of data points and data sets and to easily allow filtering by range and uncertainty size, as well as to support a number of quick and easily applied transforms such as logarithms, exponentiation, sign reversal and so forth.

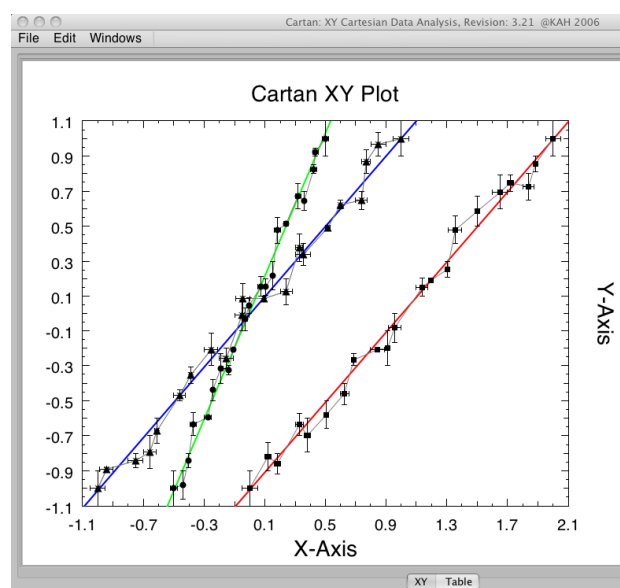


Figure 2: Screen-dump of the Display Window showing three test data sets with error bars.

Figure 2 shows a typical display of the plotting area as rendered by Cartan. Three data sets are plotted with error bars in both x and y and straight line have been automatically fitted using a least-squares fit. Individual data points can be excluded or data can be filtered by range in x , y , error-in- x or error-in- y and the fit is instantaneously recalculated and redrawn. It is thus possible to obtain immediate feedback as to the importance or role of the data. The variation and thus the uncertainty in the fitted parameters can thus be judged much more effectively than just from a computed statistic.

3 Model Fitting and Reduction

Oftentimes a model is used to reduce a set of data to one or more fitted coefficients or parameters in a fitted

model. Although the simplest of these is a simple linear fit of a straight line (lin-lin), there are two other important cases - log-log and log-lin - that crop up a lot in scaling analysis and other applications a lot.

3.1 Lin-Lin Plots

The most well known form of this is in a regression or straight-line fit whereby x-y data are modelled as a line of the form $y = a_0 + a_1x$ where the experimental data may have uncertainty estimates in the y values, or both the x and y values or in neither. The two best known algorithms for fitting a straight line involve finding the least square sum of departures from the fitted line and the experimental data or finding the least absolute value of the summed departures. In either case any available uncertainties can be exploited to inverse weight the data so that good data is treated as more important than data with a large uncertainty.

While these algorithms are usually fairly robust and will generally come up with a sensible fit for sensible input data it is still valuable to be able to perform a fit interactively and visually judge the relative importance of outliers or data points that may be anomalous. Data anomalies can creep into experimental measurements for all sorts of reasons including incorrect transcription, human idiocy or faulty sensors and other mechanical problems. In any case, supporting an interactive fit allows a human to make a sensible judgement whether the fit is justified and indeed to what extent the fitted parameter can be trusted.

Supporting tools to temporarily adjust the data inclusion filters and instantly see the response to the fitted line and its parameters gives a very sound way of estimating uncertainties in the fitted parameters themselves. Interactive comparison of fits to different data sets all viewed together can also give important visual clues to interpreting a set of measurements.

3.2 Log-Log Plots

If we have a straight line fit to a log-log plot, we obtain:

$$\log y = a_0 + a_1 \log x \quad (1)$$

$$= a_0 + \log x^{a_1} \quad (2)$$

and taking the exp of both sides:

$$y = \exp(a_0 + \log x^{a_1}) \quad (3)$$

which becomes:

$$y = \exp(a_0) \times \exp(\log x^{a_1}) \quad (4)$$

$$= \exp(a_0) \times x^{a_1} \quad (5)$$

So that the slope of the straight line fit is the exponent of a power law relationship and the exp function of the intercept gives the scaling multiplier.

This works for logs of other bases too, so for example a \log_2 fit would give the multiplier as 2^{a_0} , or a \log_{10} fit would have a multiplier of 10^{a_0} .

Usually a fit will yield a standard deviation or other uncertainty measure in the coefficients $\delta a_0, \delta a_1$. In a log-log plot we end up using and quoting the exponent a_1 directly. However for the multiplier, we can use the relationship:

$$\frac{d}{da_0} \exp(a_0) = \exp(a_0) \quad (6)$$

which we can use for a first-order approximation for the uncertainty in the multiplier as

$$\delta = \exp(a_0) \delta a_0 \quad (7)$$

This is obviously sensitive to errors in a_0 which become magnified for $a_0 > 0$.

3.3 Log-Lin Plots

If we have a straight line fit to a log-lin plot, we obtain:

$$\log y = a_0 + a_1 x \quad (8)$$

and taking the exp of both sides:

$$y = \exp(a_0) \exp(a_1 x) \quad (9)$$

Similarly if we have a straight line fit to a lin-log plot, we obtain:

$$y = a_0 + a_1 \log x \quad (10)$$

$$= a_0 + \log x^{a_1} \quad (11)$$

or:

$$\exp(y) = \exp(a_0) x^{a_1} \quad (12)$$

3.4 General Linear Model Fits and Non-Linear Fits

A full discussion of fitting more general linear model and the even more general case of fitting non-linear models is beyond the scope of this present note. Some prototype implementations have been incorporated into Cartan, and the algorithms are discussed in detail in [8]

4 Cartan Implementation

The Cartan display is managed in both data units and pixel units. When the mouse is pressed its location in data units is shown in black. If a data point or control feature is highlighted by the press, then the distance (in data units) from the mouse to that feature is shown in red. The data set to which the data point belongs is also shown. Selecting a point makes its set the selected data set as displayed in the Data Manager.

The program manages a list of data sets. Each set can contain an arbitrary list of x-y-dy data. Each set is numbered. The zero'th set is the "scratch" set. Opened xy-files go into set numbers 1,2,... and so forth.

4.1 Data Manager Window

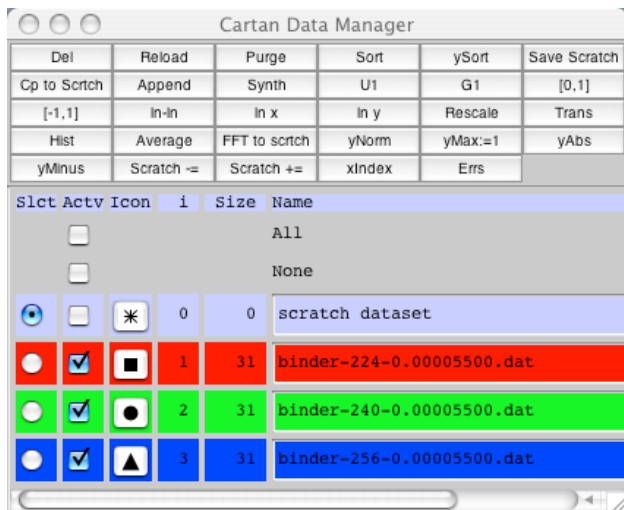


Figure 3: Data manager window for Binder cumulant example.

Figure 3 shows the data Manager window for the Binder cumulant example. The three data sets have

been loaded from file and show up as a coloured line record.

The Data Manager Window allows data sets to be made active or inactive by tick-boxes. It also reports their set number and name. The "Selected data Set," and there can be at most one, is shown by small round radio buttons, and also the toolbar background colour.

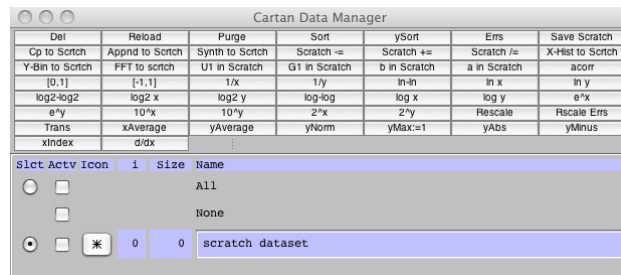


Figure 4: Screen-dump of the Data Manager Window showing a single scratch data set - generatable internally for test purposes - as well as the data transform tools widgets and selection filters.

Figure 4 shows how the data manager can also be used to generate simple test data sets, and transform them, save them or combine them with other loaded data sets.

The plotting symbols used are chosen by default from the data set index. This can be customised by clicking on the plot symbol icon shown next to the data set in the Data Manger dialog. A blank symbol is also available. The size of (all) plot symbols can be changed in the render dialog.

Tools available on the DataManager include:

- **Delete** the selected dataset, removing it entirely from the data manager.
- **Purge** the selected dataset, removing any data points that are outwith the present data filter
- **Sort** the data set in ascending x point order, descending y order - mostly useful to resolve appearance problems when drawing lines joining points or when making a bar-chart.
- **Save** the scratch set to a "proper data set" at the end of the list.
- **Copy** the selected data set to scratch
- **Append** a copy of the selected data set to scratch.

- Make a copy of the **current synthesised data set** in scratch
- Create a **unit test set** (in scratch) of the line $y = x + 0.1 * \text{uniform random noise}$, on $x : [-10, 10]$
- Create a **Gaussian distributed test data set** in scratch
- **Normalise** the selected data set so that all its data are on $x:[0,1]; y:[0,1]$
- **Central-Normalise** the selected data set so that all its data are on $x : [-1, 1]; y : [-1, 1]$
- **log-log**, log x and log-y data transforms - these all work on the selected data set (natural logs, base-e)
- **Rescale** the selected data set, transforming x to $a + bx$ and y to $c + dy$.
- **Transpose** the data set so that x becomes y and vice versa. Likewise the x and y errors
- **Histogram** the data sets x values to make a histogram in scratch
- **Average** points to reduce the size of the data set. This does a simple averaging (in place) of a specified number of points in both the x and y values.
- **FFT** - perform a real Fast Fourier transform of the selected data set.
- **y-Normalisation** so that the sum of all y values are unity
- **y-Max** set equal to unity by scaling all values in the set
- **y-Abs** - sets the y values to their absolute values
- **y-Minus** reverses the sign of the y values
- **scratch** -= and scratch += subtract from or add to the scratch set - assuming the scratch and selected sets have the same order and values of abscissae.
- **xIndex** sets the x values of the set to their point ordinal index.
- **Errs** - offers a linear way to construct errors in the data set as a proportion of the x or y values and also with a fixed amount.

4.2 Mouse & Keyboard Controls

The mouse controls are still under development - some choices still to be ratified but at present:

- a press selects a nearby point
- a ALT (**after** a press is started) followed by a drag changes the x value of the point
- a SHIFT (**after** a press is started) followed by a drag changes the y value of the point
- a CONTROL (**after** a press is started) followed by a drag changes the y-error value of the point
- an 'x' during a press deletes the selected point.
- an 'a' during a press annotates the selected point.
- an 'h' during a press hides (de activates) the selected point.
- an 'c' during a press copies the selected point to scratch (unless it is already part of scratch.)
- an 't' during a press transfers the selected point to scratch (unless it is already in scratch.)
- a SHIFT-click adds a point at an arbitrary location. It is added to the selected data set.

(**After-press** issues seem to be related to Macs without a 3 button mouse, where the CTRL keys etc mean something special in terms of the virtual mouse. Plug in a 3 button mouse and all is well.)

There are also associated **Keyboard Controls** that go with mouse selection of points. Selecting a point by holding down the mouse button near to it allows you to change the x and y values using the arrow keys (left, right, down, up) in the obvious manner.

4.3 Data Models

A linear-regression fit ($y = a + bx$) can be done dynamically as the data are viewed/edited. The resulting fitted line is drawn in red and will be especially thick to indicate a poor data fit. Results of the fit can also be logged to the ongoing textual log. The log can be viewed in the log Window and can be saved to file. Moments and elementary statistics on the y-values only can also be logged to the log. The fit can be a least squares fit, or a "least-abs" or "robust" fit. Least-squares fits can include the use of y-errors as (inverse) weights for the points, or not. Fit choices are controlled from the Model Manager.

4.4 Data File Formats

Data files are assumed to be of the multi-column forms (x-y-dx-dy; x-y-dy; x-y; x):

```
# a comment
1.0 2.0 0.0001 0.01 # a point label
2.0 4.0 0.0001 0.01
3.0 6.0 0.0001 0.01 # another label
```

or:

```
# a comment
1.0 2.0 0.0001 0.01
2.0 4.0 0.0001 0.01
3.0 6.0 0.0001 0.01
```

or:

```
# a comment
1.0 2.0 0.001
2.0 4.0 0.001
3.0 6.0 0.01
```

or:

```
# a comment
1.0 2.0
2.0 4.0
3.0 6.0
```

or:

```
# a comment
1.0
2.0
3.0
```

in this last case, the data are treated as y-values, and corresponding x values are made from the index.

In the case of files with more than 4 columns, a modal dialog inquires which columns are to be treated as x, y, dy etc. The 4 column case indicates the presence of dx error values (in the abscissae) as well as dy errors. The canonical order is therefore x-y-dx-dy.

4.5 File Menus

The **File** menu offers choices to:

- **open** a new data file - multiple file selection is supported

- **save** the selected data set to file
- **import** imports xy data from a multi column format
- **export** submenu offers options to extract a single field eg x, or y or dx or dy as well as exporting the whole (selected) data set as comma separated or tab separated formats.
- **save** and **reload** *preferences* information (stored as plists on OS X)
- **import / export XML** versions of your *preference* information
- **print** the current data viewer
- **export image** submenu offers choices including **ppm** which saves the current view as a PPM image file
- **save** the text *log*
- **exit** the program

The **Edit** Menu is still under design and at present just supports deletion of all data sets (except the scratch set). A recent addition is the capability to restore **all** individually deactivated data points.

The **View** Menu is currently empty.

The **Window** Menu is still under design but currently allows toggling visibility of the: Help Window, the Data Manager window, the Log Window, the Model Window and the Render Manager Window.

The **Help** Menu offers viewing of this help information in a separate Help window.

4.6 XY Viewing Controls

A number of switch setting allow adjustment of the plots appearance. These include the following, which can be saved as preferences:

- **Draw Axes** to draw axes lines with numerical graduation scales and tick marks
- **Axes-Box** to draw an enclosing box around the plot area
- **Central Axes** to use axes centred on the data origin (0,0) rather than at left(y-axis) and bottom(x-axis)
- **Draw Grid** to draw a grid or mesh over the plot area with grid-lines at the major tick marks

- **Frame** to draw an enclosing frame or box around the entire page
- **In-Ticks** to use ticks that go up from the x-axis and right from the y-axis instead of the defaults - down and left respectively.
- **Draw Fit** to automatically draw fitted lines according to the present data model
- **X-Errorbars** to show x-error bars for data point y-errors
- **Y-Errorbars** to show y-error bars for data point x-errors
- **Scale-All** to automatically adjust the plot scale range for all data (even for de-selected data sets)
- **Lines** to draw lines connecting points in the order in which they appear in their respective data sets
- **Points** to draw each data point
- **Symbols** to draw symbols rather than just dots, if we draw the data points
- **Titles** to draw the x-axis label, the y-axis label and a title for the plot
- **Labels** to draw any textual point labels found in the file (following a # after each data point in the file)
- **X-Filter** to activate the x-data filter, so only data points between the low and high values remain and are used in the currently selected model fit - the limits are drawn as vertical lines
- **Y-Filter** to activate the y-data filter, so only data points between the low and high values remain and are used in the currently selected model fit - the limits are drawn as horizontal lines
- **Y-Err-Filter** to activate the dy-data y-errors data filter, so only data points between the low and high values remain and are used in the currently selected model fit - the limits are drawn as an error bar symbol to the left of the plot area. It can be clicked and dragged to change this limit.
- **X-Err-Filter** is not yet implemented, but is intended to be similar to the Y-Err-Filter.
- **Synthetic** to draw a representation of the synthesised data set
- **Barchart** gives the option to draw a barchart plot format

The control area offers various (some experimental) controls to change the plot area appearance. Most are passive and are read by the display area when it re-renders the plot. Some are active tools that change the underlying data model. One such is the button to reset the filters. There is also the capability to constrain the axes min/max values or to set them to specific defaults such as ± 1 .

4.7 Display Control Pane

Figure 5 shows the main display and associated display controls. Three data sets are being selectively fitted in x and straight lines are being fitted to the filtered data.

The control area offers various (some experimental) controls to change the plot area appearance. Most are passive and are read by the display area when it re-renders the plot. The plot area is computed automatically with (by default) a pad area of 1 tick unit at both min and max x and y. The pad area can be explicitly set and constrained. Likewise the number of tick units for each axis is computed automatically but can be explicitly set and constrained. The actual max and min values for x and y are also estimated from the data set(s) but the inclusive choice can be over ridden and constrained.

The Constrained Controls generally have a small square checkbox to their immediate right. Checking it fixes the value to what is in the box - although the user can still edit it, the program will not try to recalculate it. Some controls such as the tick-pad spinners have a small round button accompanying them that resets their value to the default.

Some buttons are active tools that change the underlying data model. One such is the button to reset the filters. There is also the capability to constrain the axes min/max values or to set them to specific defaults such as ± 1 . Unit Box and Central Box set the plot minima and maxima to specific values. Zero Pad removes any padding that was automatically calculated to keep the data points strictly inside the plot area.

Combo boxes offer some other choices. The axes tick numbers can be formatted in various "printf" format styles, the point labels can be shown at various relative positions to the data points, and a caption can be generated as a key for explaining the symbols of multiple data files/sets.

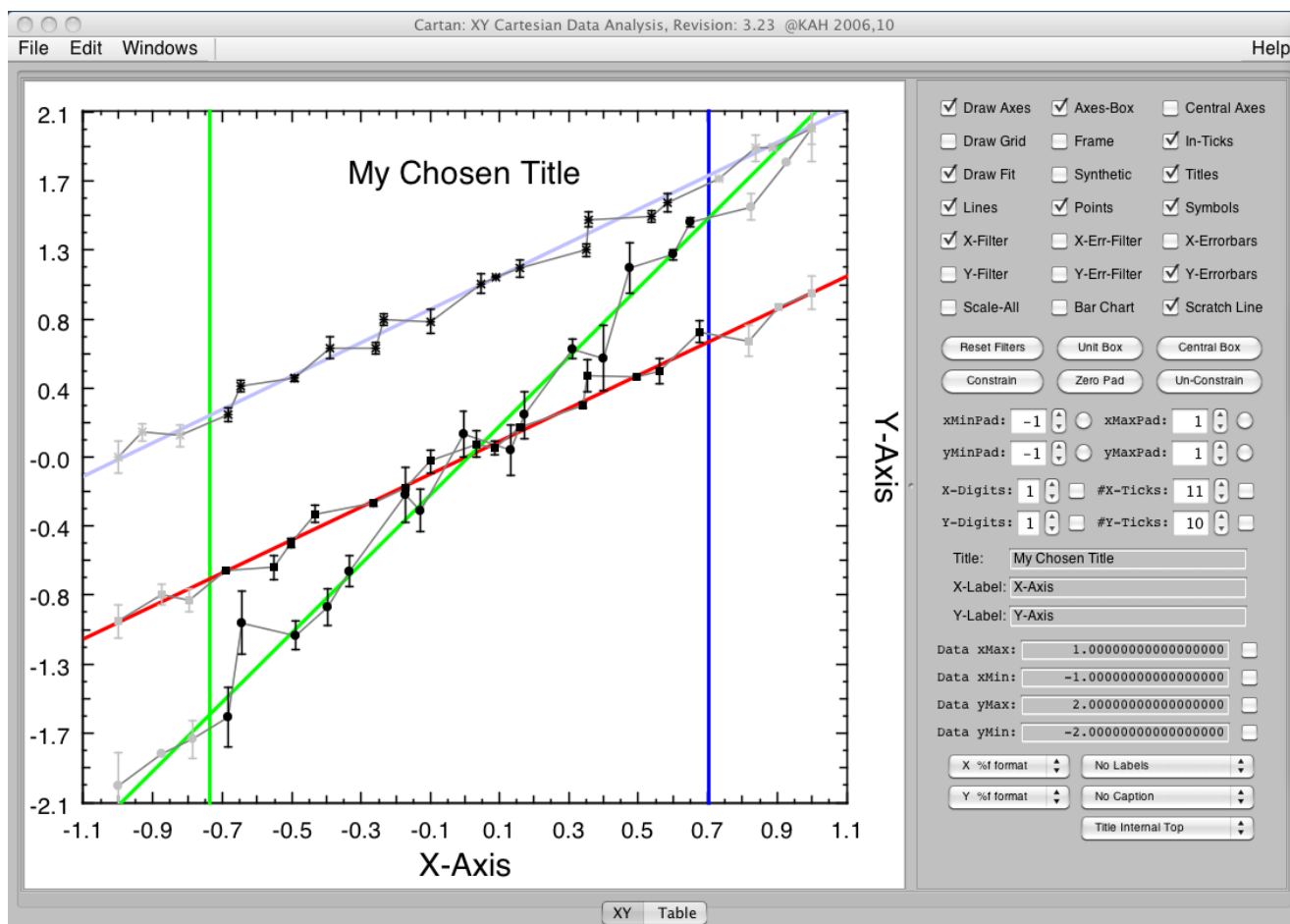


Figure 5: Screen-dump of the Main Display Window with three data sets shown being fitted by straight lines and with the x data filters activated. The display appearance controls and widgets are in a pane to the right.

4.8 Table View

The Table view provides a simple spreadsheet style view of the individual point data in the selected data set. Each point is shown with its index in the set, its x, y, dx and dy values, an annotation string label if any is present, and its activation status. The table controls offer selection of floating point formats and font.

Figure 6 shows the table view for the Binder cumulant data set example. Individual points can be labelled or de-activated directly by editing the table view.

4.9 Render Window

Some rendering controls such as those determining the font size, border length, stroke thickness and symbol size have been relocated to the Render Win-

dow. These parameters are common to all viewers and so are not appropriate for the Control pane that is matched with the XY display. Figure 7 shows the rendering controls available in the render window.

4.10 Log Window

The textual log is maintained in a separate window. It can be viewed and edited and saved to file. Its toolbars also allow generation of fit information and statistics on the data sets. Other information can be logged from any part of the program.

Figure 8 shows the log window as it might appear during analysis of the Binder cumulant data sets.

Figure 9 shows an alternative screen-dump of the log manager window.

#	X-Value	Y-Value	X-Error	Y-Error	Label
0	4.5110000	0.4542045	0.0010000	0.0016159	
1	4.5110000	0.4527724	0.0010000	0.0016044	
2	4.5110000	0.4515728	0.0010000	0.0016219	
3	4.5110000	0.4512424	0.0010000	0.0017953	
4	4.5110000	0.4505049	0.0010000	0.0018479	
5	4.5110000	0.4497725	0.0010000	0.0018972	
6	4.5110000	0.4489881	0.0010000	0.0019457	
7	4.5110000	0.4482010	0.0010000	0.0019913	
8	4.5110000	0.4474402	0.0010000	0.0020349	
9	4.5110000	0.4467022	0.0010000	0.0020764	
10	4.5110000	0.4459822	0.0010000	0.0021159	
11	4.5110000	0.4452857	0.0010000	0.0021532	
12	4.5110000	0.4446184	0.0010000	0.0021882	
13	4.5110000	0.4439752	0.0010000	0.0022214	
14	4.5110000	0.4433514	0.0010000	0.0022529	
15	4.5110000	0.4427442	0.0010000	0.0022829	
16	4.5110000	0.4421582	0.0010000	0.0023114	
17	4.5110000	0.4415982	0.0010000	0.0023382	
18	4.5110000	0.4410684	0.0010000	0.0023632	
19	4.5110000	0.4405644	0.0010000	0.0023862	
20	4.5110000	0.4400814	0.0010000	0.0024072	
21	4.5110000	0.4396154	0.0010000	0.0024262	
22	4.5110000	0.4391614	0.0010000	0.0024432	
23	4.5110000	0.4387154	0.0010000	0.0024582	
24	4.5110000	0.4382814	0.0010000	0.0024712	
25	4.5110000	0.4378454	0.0010000	0.0024822	
26	4.5110000	0.4374154	0.0010000	0.0024912	
27	4.5110000	0.4369884	0.0010000	0.0024982	
28	4.5110000	0.4365614	0.0010000	0.0025032	
29	4.5110000	0.4361384	0.0010000	0.0025072	
30	4.5110000	0.4357184	0.0010000	0.0025102	
31	4.5110000	0.4353014	0.0010000	0.0025122	

Figure 6: Table view for Binder cumulant example.

```

Cartan Log
-----
Cartan Log Started at Tue Oct 03 14:02:54 NZDT 2006 by kahawick

Data Set 1 (binder-224-0.00005500.dat)
Statistical Summary:
  Number of data: 31
  Average: 0.5738962895153527
  Absolute Deviation: 0.06954463086270067
  Standard Deviation: 0.0818081949821232
  Standard Variance: 0.006692574076956853
  Skewness: -0.7282396880372921
  Kurtosis: -2.8134539002987966
  Minimum: 0.407786318173235
  Maximum: 0.6542685436040686
  Sum: 17.79078497497593
  Median: 0.6047815658729154
  Mode: 0.407786318173235
  ModalFrequency: 1

Data Set 2 (binder-240-0.00005500.dat)
Statistical Summary:
  Number of data: 31
  Average: 0.5792508060722354
  Absolute Deviation: 0.0708099031792613
  Standard Deviation: 0.08399186216006062
  Standard Variance: 0.007054632909114623
  Skewness: -0.8479390709774205
  Kurtosis: -2.8103899774428567
  Minimum: 0.3989723492570864
  Maximum: 0.6573425030994435
  Sum: 17.9567749882393
  Median: 0.6221737450750195
  Mode: 0.3989723492570864
  ModalFrequency: 1

Data Set 3 (binder-256-0.00005500.dat)
Statistical Summary:
  Number of data: 31
  Average: 0.5862679111929138
  Absolute Deviation: 0.0705527368632723
  Standard Deviation: 0.08444238781030204
  Standard Variance: 0.007130514689105464
  Skewness: -1.0204699870069274
  Kurtosis: -2.7726092466620598
  Minimum: 0.3759899983906296
  Maximum: 0.658628706386145
  Sum: 18.174305246980328
  Median: 0.6307830960176456
  Mode: 0.3759899983906296
  ModalFrequency: 1
    
```

Figure 8: Log window for Binder cumulant example.

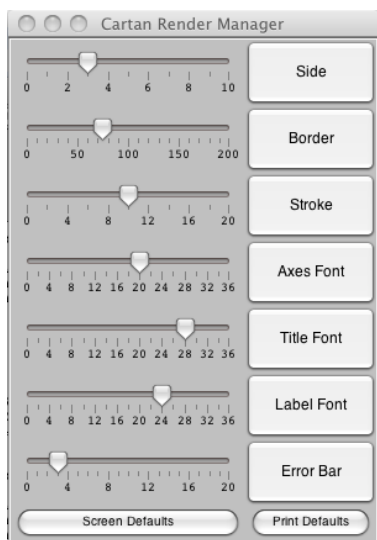


Figure 7: Render manager Window showing some secondary sliders and controls to adjust the visual appearance of the plotting window - primary for use when printing or saving as an image for subsequent inclusion in a document.

4.11 Synthesis Window

The Synthesis window shows controls for adjusting parameters for synthetic data models such as polynomials or sin/cos or Fourier spectrums. Sliders allow parameters to be adjusted in real time. The synthetic data set can be viewed in the display area, and can be copied to the scratch set and used as a normal data set.

Figure 10 shows a screen-dump of the synthesis control window with slider controls for generating (in this

case) synthetic polynomial data.

Figure 11 shows a polynomial being synthesised and plotted alongside some experimental data loaded from file. This gives a powerful yet flexible way to hypothesise a model for some data and obtain immediate feedback whether it is at all reasonable compared with experimental data.

When a synthetic model is initiated by pressing a selection from the synthesis tool bar, the user is prompted for a number of coefficients for some models such as arbitrary polynomials or spectra. This information is used to rebuild the synthesis panel with the appropriate widget set.

The synthetic data set can be saved to the scratch data set - which in turn can be promoted to a full proper data set and/or saved to file. This is controlled from the Data Manager Toolbar.

The Synthetic data set has its abscissae computed once on model initiation. If the data window is changed or filters changed it can become inconsistent. Double clicking the Draw Synthetic checkbox in the plot control area will re-initiate the abscissae.

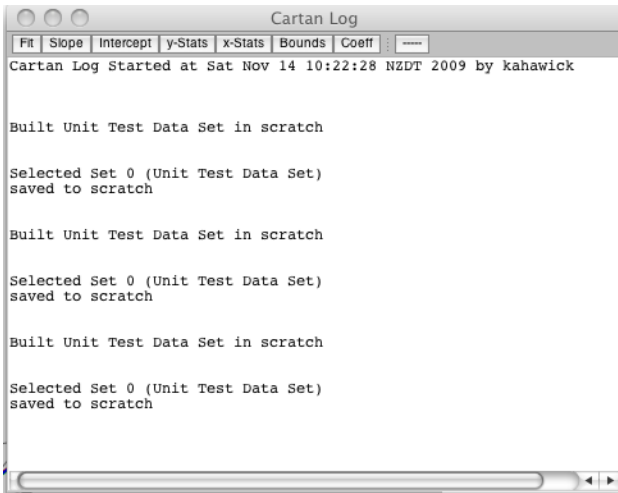


Figure 9: Screen-dump of typical output in the Log Window, which the user can type into, or cut and paste from or save to file.

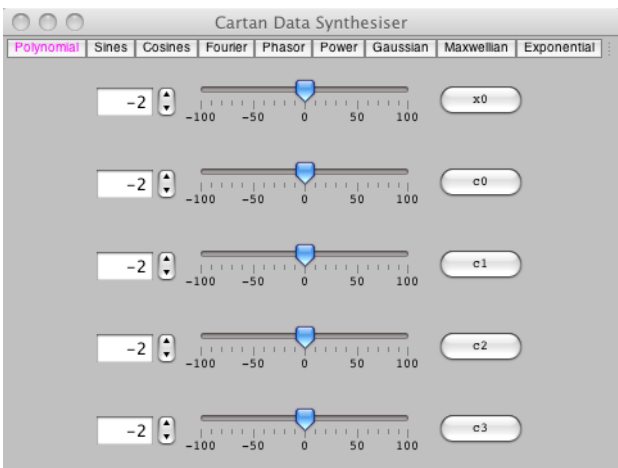


Figure 10: Synthesis Manager Window showing slider controls for generating synthetic polynomial data.

4.12 Model Manager Window

The Model Manager allows various models to be fitted to one or more data sets. The principle models available are: Linear Least-Squares fit of a Straight line with no errors; with y-errors only and with both x- and y-errors. Linear Least-Absolutes "robust" fit of a Straight line with no errors used. Linear Singular-Value-Decomposition fit of an arbitrary order polynomial with y-errors used and required. Non-Linear Levenberg-Marquard fit of an arbitrary number of Gaussian Peaks with y-errors used and required.

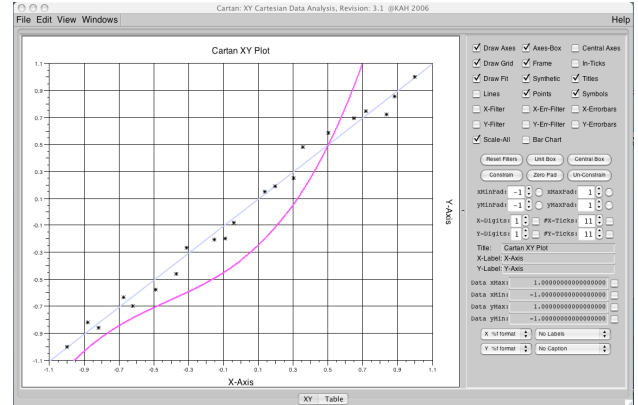


Figure 11: Synthesised data plotted alongside experimental data.

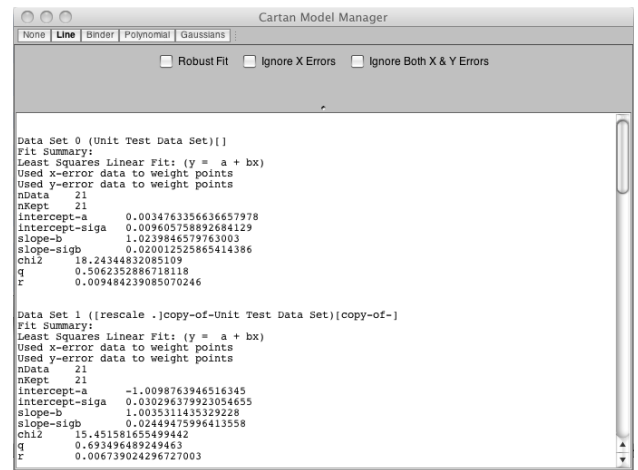


Figure 12: Screen-dump of the Model Manager Window showing controls for selecting a model to fit and some verbatim text of the fitted coefficients and statistics.

In addition a compound fit model for Binder Cumulants computes the intercept of several straight line fits to separate data sets.

Each model is chosen from the Model Manager Toolbar, and can be recomputed dynamically and displayed in the plotting DisplayPanel if its Draw Fit checkbox is set. Otherwise a fit can be initiated as a once off from the compute Fit tool in the LogManager window. In either case both are subservient to what model choice is made in the Model Manager. In the case of models such as an arbitrary Polynomial or Gaussian Peaks, the Model Manager offers Constrained Text fields for entering constrained parameter values. If constrained a coefficient will yield a zero

uncertainty. Otherwise uncertainties in each parameter are given from the computed covariance matrix during the fit.

Model fits will generally present fitted coefficients with their uncertainties as well as a Chi-Squared goodness of fit metric and others such as regression coefficients and probabilities as appropriate to the algorithms and models involved. The fit information is displayed dynamically by the model manager, and can also be logged by the log manager.

Figure 12 shows a screen-dump of the model manager window with controls for selecting which model to fit, and verbatim text of the fitted coefficients and statistics. This information can also be requested in the log window which can be saved to a text file.

While a fit is attempted for each active data set, only the selected data set can have its fit parameters constrained. The straight line fits will attempt to use error data if available. If the Robust Fit check box is set in the Model Manager, then the least absolutes (robust fit) algorithm is used, and it ignores all errors in x and y .

Generally straight line fits and polynomials are easily recomputed interactively and several data sets can be dynamically fitted "each time the mouse moves" or each time a data point is added. This appears to perform adequately even for data sets of several thousand data points. However the non-linear fit uses an iterative matrix algorithm, and it is likely to be slow to refit dynamically other than for relatively few and relatively small data sets.

4.13 Help Window

The in built web browser window capability of Java Swing was used to provide help information in the form of an HTML file, that can be bundled with the Cartan Jar file installation.

Figure 13 shows a screen-dump of the typical help file, with hypertext links to sections within.

5 Discussion

Cartan was packaged up as a self-contained Java jar file. While I tends to launch it from a command line using a shell alias and giving the names of data files as command line arguments, I have noticed my students feel more comfortable launching it as an iconic graphical program and using the mouse to select data

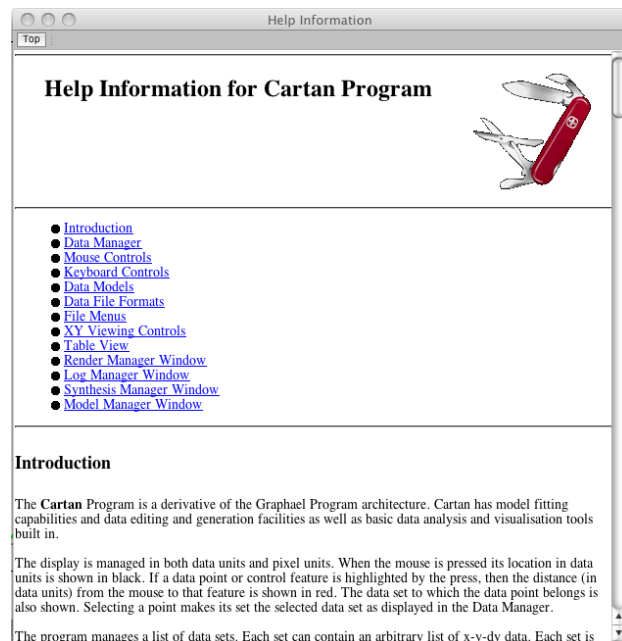


Figure 13: Help Window which allows appropriate rendering of help information written as HTML including internal cross-referencing hyperlinks.

files once it is launched. The jar file with shell alias supports both models of operation rather well and it is not particularly difficult to compile, and install new releases of Cartan.

Testing of Cartan was not difficult in its conception phase and early development phase when all its workings were clear in my head. The program has now grown to approximately 17,000 lines of Java. While I have managed to successfully use a component structure that means upgrades to the tool set can be made with relatively clean and clear file modifications, in hindsight I wish I had planned for some sort of Unit testing at an earlier stage of development.

The internal data structures have proved easy to experiment with although for with the design more fixed, they could be improved for speed and response-time performance. I used linked lists to store data sets. It would be faster to use plain ordinary arrays with accompanying boolean arrays to specify whether data points are active or not. On modern desktops and JVMs, memory is unlikely to be a performance limitation.

The general layout of the controls is somewhat esoteric and would no doubt benefit from some HCI considerations, although like any complex scientific or technical tool, it is straightforward to operate when the user

has a feeling for its purpose and a good grasp of its working vocabulary, and its internal model of operation but quite difficult to use if this is not the case.

6 Conclusions

Java is a good prototype technology for a complex visual program like Cartan, and I see no reason why it is not also useful for developing production tools too. The general interactive loop of having a graphics thread recompute and re-render as one interacts is very powerful. Java on a typical desktop is adequate for this. In general as long as one does not try to manipulate more than around 20 data sets each of more than 10,000 data points, then the Cartan data structures, design and interactive performance are entirely adequate. Beyond this - which is probably outside my normal likely interactive requirements as a human being anyway, the program and its graphics thread do clearly struggle and the interactivity effectiveness corrodes.

Generally as a prototype this project has been well worthwhile. A production prototype could be designed and built based upon the Cartan ideas. Surprisingly there are still no obvious alternative tools available commercially or publicly, that would support the sort of analysis customisation and experimentation that I require. In terms of rendering, for many purposes Gnuplot does produce a better rendered production plot picture for inclusion in documents, but it is still time consuming to set these up, “post-thinking process” as it were. Cartan produces acceptable plots as one goes along without the need to stop and focus on the presentation. Writing up as one goes along is – I believe – still an absolutely vital part of science productivity.

There are some areas for further development of Cartan or a tool like it. There are new data transforms that could readily be incorporated into the Data Manager. Additional models to fit such as positive coefficient polynomials for big-oh analysis of performance and timing data would be useful. A 3D surface plot capability, that could work similarly to gnuplot’s splot capability is attractive and would use x-y-z data is attractive, but might be better written as a separate tool.

Generally the Cartan program implementation has proved very useful in a number of computational science experiments [9] and has aided the Complex Systems and Simulations Group to produce a number of publishable quality data plots and analyses.

References

- [1] Kubat, K.: xplot simple x-y column data plotter for X. Unix Man page/Web Site (1995) Last Visited October 2010.
- [2] Williams, T., Kelley, C., Lang, R., Kotz, D., Campbell, J., Elber, G., Woo, A.: Gnuplot command-driven interactive function plotting program. (2008)
- [3] The MathWorks: Matlab. available at <http://www.mathworks.com> (2007)
- [4] Wolfram Research: Mathematica. available at <http://www.wolfram.com> (2007)
- [5] Ihaka, R., Gentleman, R.: R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* **5** (1996) 299–314
- [6] Lyons, L.: *Statistics for nuclear and particle physicists*. Cambridge University Press (1986)
- [7] Hawick, K.A., James, H.A.: Ising model scaling behaviour on z-preserving small-world networks. Technical Report arXiv.org Condensed Matter: cond-mat/0611763, Information and Mathematical Sciences, Massey University (2006)
- [8] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: *Numerical Recipes - The Art of Scientific Computing*. Third edn. Cambridge (2007) ISBN 978-0-521-88407-5.
- [9] Hawick, K.: Computational science technical notes - the first one hundred - a review of the cstn series. Technical Report CSTN-100, Computer Science, Massey University (2009)