Csvgraph – a graph plotter for csv format files

Manual for Version 3.10 6/4/2024

# Introduction

Csvgraph is designed to allow quick viewing of graphs of potentially very large (GB) csv files (for comparison most spreadsheets are limited to 1,048,576 rows). Csvgraph has no built-in limits, but ultimately it is limited by your available RAM. The 32-bit version will it will use up to 4GB of RAM if its available, while the 64-bit version will use all the available RAM and as much virtual memory as Windows will allow (with 16GB of RAM it’s possible to load a 100GB file which needs more than 32GB of virtual memory). Even with extremely large files loading a file is fast and zooming is normally instantaneous. Note that reading a file that forces Windows to use virtual memory means things do slow down, but they are normally acceptable with an SSD and the progress indicator will give you an idea of how long it will take.

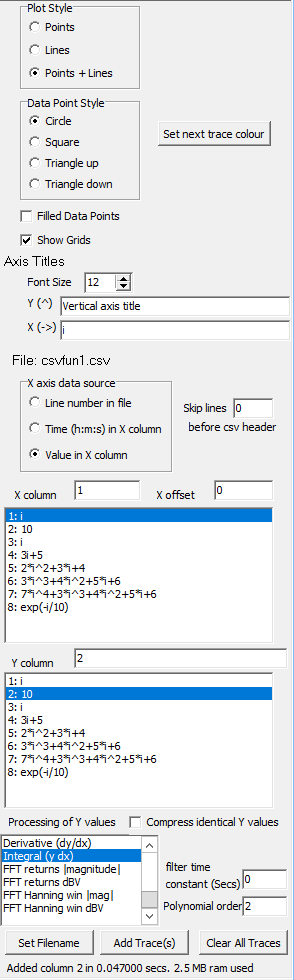
These csv files are assumed to have column headers on their first line so a typical csv file would start:

"Time(sec)","Col-2","Col-3","Col-4","Col-5"

99950,20,0,20,20

99950.1,10,1,11,12

If the header row is not on the first line of the file, then it’s possible to tell csvgraph to skip a number of lines before it reads the header (this must be selected before the file is opened).



Values are read as floating-point numbers so are restricted to numbers between +/-3.4e+38 and the smallest non-zero number is approximately 1.4e-45, with approximately 7 significant digits.

The X values are assumed to be monotonically increasing, if they are not in the csv file then the X values (together with the corresponding Y value) are automatically sorted before they are displayed.



# Use

Run csvgraph.exe by double clicking on it.



Select a suitable csv file (there are few examples in the archive and csvfun2.csv was used for the screenshot) and drag the file onto csvgraph (or select the file by pressing the “Set filename” button bottom right or use the menu, File, Open).

This should populate the Xcolumn (horizontal axis) and Ycolumn (vertical axis) boxes from the header row of the csv file as shown in the screen shot above. To add a trace to the graph, select one x column and one or more y columns and press the “Add Trace(s)” button on the bottom left.

If the Xcolumn and Ycolumn boxes are not populated correctly then it’s likely the csv file does not have a header as its first row. If there are some lines before the header then please see the Introduction for instructions on how to skip the lines before the header. Alternatively, you could add a suitable header line manually using a text editor like Notepad++.

Multiple traces can be added, and you can change the filename between traces if required by pressing the “Set Filename” button again before selecting X and Y columns and pressing the “Add Trace(s)” button.

When a graph is displayed move the mouse over it and press the left mouse button, while keeping this button pressed move the mouse to select an area of the graph – when the mouse button is released the graph will zoom into the selected area. To restore the original view, press the middle mouse button. The right mouse button allows for measurements to be taken from the graph as shown below:



The mouse scroll wheel also allows quick zooming in and out on the graph.

The menu Scales option allows a specific area of the graph to be easily viewed, while the buttons on the bottom left allow the X and Y axes to be moved and zoomed independently.

There is no limit to zooming within the number range specified in the introduction – the graph below shows zooming in on a single point on the graph:



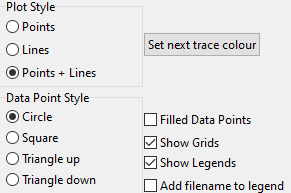
# Adding Legends to a graph

A title can just be typed into the area by default labelled “Title”.

The y and x axis titles are entered into the boxes in the column on the right. Just above these the font size for these legends can be set.

The y (vertical) title can be split into 2 lines by including \n at the end of the first line e.g., “vertical\naxis title”.

# Changing the format of the graph



By default, the graphs consist of points joined by lines; at the top of the right-hand column the plot style can be changed to display just points or just lines and the shape of the points and filled/not selected.

By default, additional traces will be given different colours automatically, but the colour can be set by pressing the “Set next trace colour” button if required.

Graphs legends normally just consist of the Y column name, if the “Add filename to legend” box is ticked then the basename of the filename is added which can be useful for example if the same Y column is being plotted from several different files.

Traces are not visible on the region of the screen where legends are displayed, this is not normally an issue (the Y (vertical) axis can always be rescaled to avoid this). If many traces are added the legends can take up a large area of the screen, in which case unticking the “Show legends” tick-box will remove the legends from the graph.

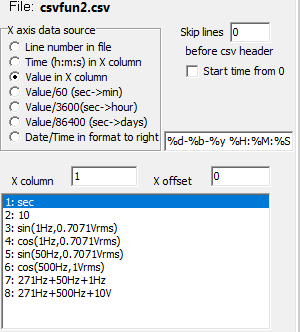
# Saving graphs

Graphs can be copied to the clipboard by pressing the “Plot to Clipboard” button along the bottom of the screen, or by using the menu File/Save/Plot to clipboard.

Graphs can be saved to files by pressing the “Plot saveas” button on the bottom of the screen or using the menu File/Save/Save plot as. Plots can be saved as bmp, jpg, gif or png files. Note the graphs title is not saved (or copied to the clipboard) as in most uses this will be added as a caption to the graphic.

The data within a graph can be saved (which is useful if this has been calculated or filtered – see later for how to do this) by using the menu File/Save/Save Data as csv. In this case the x values from the first trace are used for the resultant csv file – if necessary, the y values for traces 2 and above are interpolated if they don’t have the same x values as the first trace. Normally the complete range of x values data is saved to the csv file, but the “save xrange on screen as CSV” menu selection just saves the range of X values that are currently shown on the screen, so allowing a portion of the currently loaded file to be saved.

# Selecting columns of the csv file



As seen previously the easiest way to select traces is to click on the names of the columns (shown in blue above). Multiple Y columns may be selected using shift and a left click or control and a left click of the mouse.

Alternatively, the columns may be selected by typing numbers into the areas to the right of the X column and Y column legends. Multiple Y column numbers are separated by commas. The 1st column in the file is numbered 1, the second 2 etc – and these can be found from the 2nd csvgraph window as shown below:



It is also possible to select the x axis value to be the line number in the file, or to specify the value in the column is a time (h:m:s with an optional leading date e.g., 1/1/2020). If “time” is selected then time on the x-axis will start at zero and be in seconds (i.e., the first value read will be used as an offset for all future values) if the “start time from 0” tick box is selected otherwise it will start at the first time read .If the time increments past 23:59:59.9999 to 0:0:0 the x axis value will be 86400 (24 hours in seconds) rather than rolling back to zero so times longer than 1 day are automatically supported. Because of this action the actual date [ if present] is ignored (see below for a way to read in dates and times if that’s required). Note that if some lines have dates and some do not, lines without dates will be loaded but will be counted as errors in the error count.

Note that numbers or times can be within double quotes in the csvfile (e.g., “12” will be read as 12).

The y column value may also be described as an expression potentially combining the values from multiple columns e.g., $2-$3 would plot the difference between the 2nd and 3rd columns.

Allowable expressions are described in Appendix A – expressions.

Note as of 3v9 if an expression evaluates to nan (Not A Number) then the line is skipped. If the column referenced does not have a valid number this is treated as a nan and any expression expecting real numbers will evaluate to nan. It is also possible to use an expression that purposely evaluates to nan (using the constant nan) to skip lines. As an example, the expression:

$2==nan?x:nan

will display lines of the input file on the graph if the value in the 2nd column in the csv file is missing (i.e. when it equals nan) and the x value is present (not equal to nan), and skip all rows that have a valid number in the 2nd column.

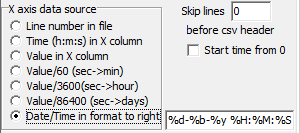
The Xoffset value (to the right of the X column) allows the x values of the most recently added trace to be moved left and right compared to previous traces (+positive numbers move right, negative move left). This can be useful to align traces.

If some lines in the csv file cannot be read (i.e., do not contain a valid number or time), then a count of these is given after the file is loaded and a number of examples of bad lines are given in the 2nd csvgraph window. At least one example for every type of error found will be shown.

If the csv file is too large to view easily, then wmawk2 (see the end of Appendix A – expressions) can be used to view lines from a commands line e.g., the example below shows line 123456 of file.csv

wmawk2 “NR==123456{print $0}” file.csv

From version 3v6 of csvgraph there are more option to read a x column that contains time or a date and a time.



Selecting the “Date/Time in format to right” as shown above and then describing the date/time using a format in the box to the right. The dates/times will be converted into seconds since 1st January 1970. It is recommended that the “Start time from 0” box is also ticked so the first valid time read (normally on the line of the csv file after the header line) will be assigned the time zero and the x axis will show seconds from this time.

The example shown (which in full is %d-%b-%y %H:%M:%S.%f ) is designed to read date/times like

05-Jul-19 11:59:24.037063

Another example of a format is %Y-%m-%d %H:%M:%S.%f which will read dates/times like:

2019-12-31 00:00:01.567000

The full list of formats are in Appendix B, but the ones most commonly used are:

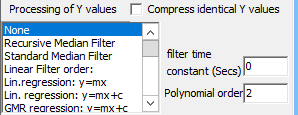
* %b Month in text (either as 3 characters like *Jul* or in full like *December*)
* %d Day of the month as a one- or 2-digit integer number (e.g., 05 in the first example)
* %f Fractional part of the seconds (e.g., 037063 in the first example)
* %H Hours (one- or 2-digit integer number) (e.g., 11 in the first example)
* %m Month as a one- or 2-digit integer number (e.g., 12 in the second example)
* %M Minutes as a 1- or 2-digit integer number (e.g., 59 in the first example)
* %S Seconds as a 1- or 2-digit integer number (e.g., 24 in the first example)
* %y Year as a 2-digit integer number (e.g., 19 in the first example)
* %Y Year as a 4-digit integer number (e.g., 2019 in the second example)
* %% matches a percent sign (%)
* Space matches any “whitespace” in the input (e.g., the gap between the date and the time in the examples).
* Any other character is matched exactly (e.g., the two –‘s, two :’s and a . in the examples). This can also be used to match double quotes (“) if these surround the date/time.

Note if the format does not match what’s actually in the csv file the line will be ignored. If the format is set so no lines match then csvgraph will warn that no values have been read in. Csvgraph’s textual window shows the contents of the first line of data and a summary of errors found reading the csv file so this is useful to see the format required.

If the X column in the csv file is in seconds, then it can be converted to minutes, hours or days by selecting the relevant option in the “X axis data source”. Note that Value/60 can also be used to convert minutes in the csv file to hours. If the graph is saved as a csv file (see earlier) then it will be saved in the format that its displayed, so for example if the original csv file was in seconds and the Value/86400 option is selected the X values saved in the new csv file will be in days.

# Filtering/curve fitting

By default, traces are added without any processing (i.e., exactly as in the csv file), but csvgraph offers a number of options to “filter” the data before displaying it.



The simplest option which is unlikely to be needed unless you need to view extremely large files or have limited RAM is “Compress identical Y values”. If this is ticked then sequences with identical Y values will be compressed (just the first and last point in a sequence kept) – the line graph will be identical with this option ticked. Note this is not true if filtering is applied as the filter values will only be calculated at stored points (csvgraph will remind you if you select “compress” and a filter).

Underneath this a range of filters can be selected and these are described below.

## None

No filtering is done (this is the default) – the data is displayed exactly as it is in the csv file.

Looking at the example file demo1M.csv column 2 scaled with x from 0 to 2500 and Y from -10 to +20 (using menu/Scales to set these limits) gives:



## Recursive Median Filter/Standard Median Filter

These implement median filters.

The output of Standard Median Filter at a point is the median value of the input data inside the window centred at the point.

If {x(k) | 1 ≤ k ≥ L} is the input and {y(k) | 1 ≤ k ≥ L} the output of a standard median filter of window size 2N + 1, then:

y(k) = median{x(k-N),..x(k - 1), x(k),x(k+1),..x(k+N)}

The Recursive median filter is defined as:

y(k) = median{y(k-N),... y(k - 1), x(k), x(k+1),..x(k+N)}

Note that in csvgraph the size the filter window is defined as a time rather than a number of samples as this gives more predictable results if the sampling time in the csv file is not constant.

Using the same example data/scale as using in the “None” example above, with a 2 sec time constant, this gives:



If the spikes on the original were noise, then this filter has completely removed them without impacting any of the other values.

From version 2v6 the standard median filter uses an exact median of points +/- the specified filter time constant if the number of points in the data is <=10,000 while for a larger number of data points it uses an approximation that appears exact if the graph is viewed with its standard scaling (i.e., without any zooming).

From version 2v7 the recursive median filter will normally use an exact algorithm, only swapping to a faster approximate algorithm if the execution speed would be too slow (which is very rare on real datasets). The standard median filter is normally slower than the recursive median filter and both give very similar results for most datasets.

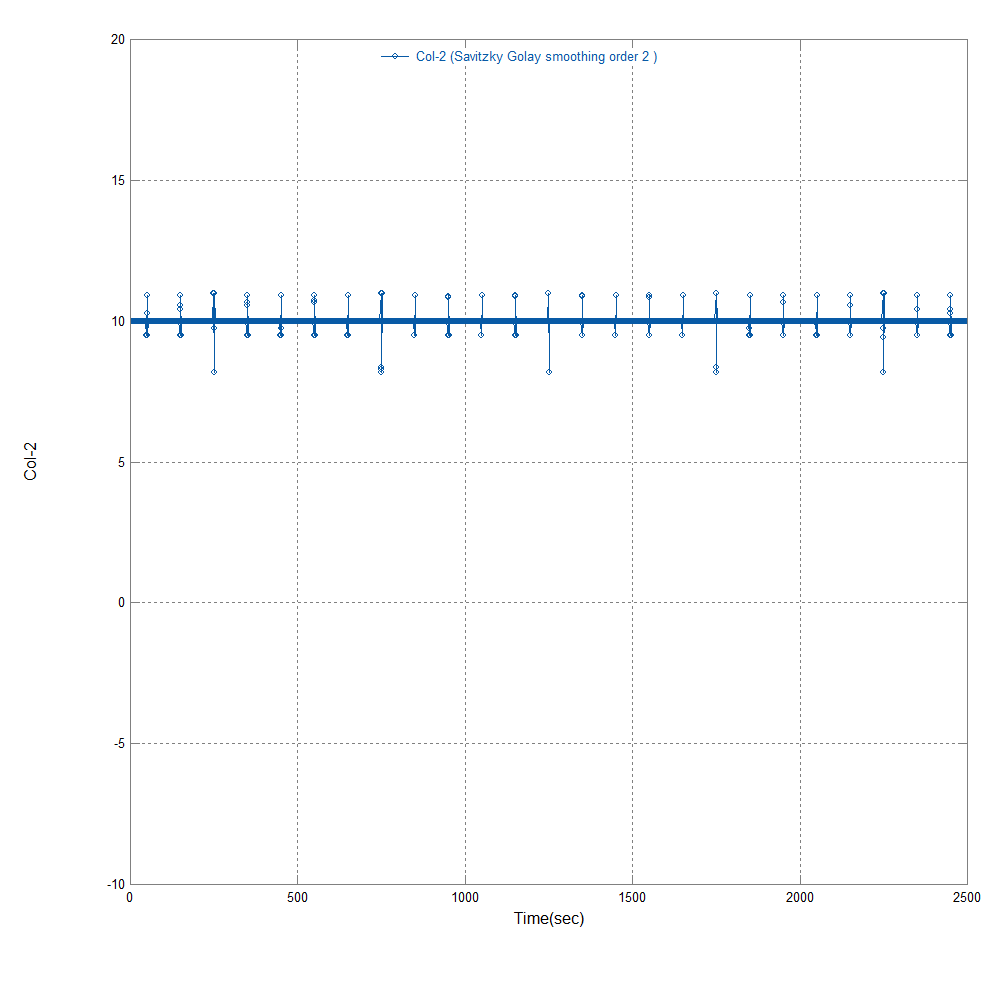
The algorithms used are very efficient so large filter times can be used if required, the (exact) median of the whole data can be found by supplying a very large value for the time constant.

See the paper "MEDIAN FILTERS THEORY AND APPLICATIONS" by Milan STORK ( <https://www.emo.org.tr/ekler/2130c418d4f02c7_ek.pdf> ) for more information on median filters and the pros and cons of a recursive median filter compared to a standard median filter.

## Savitzky Golay smoothing

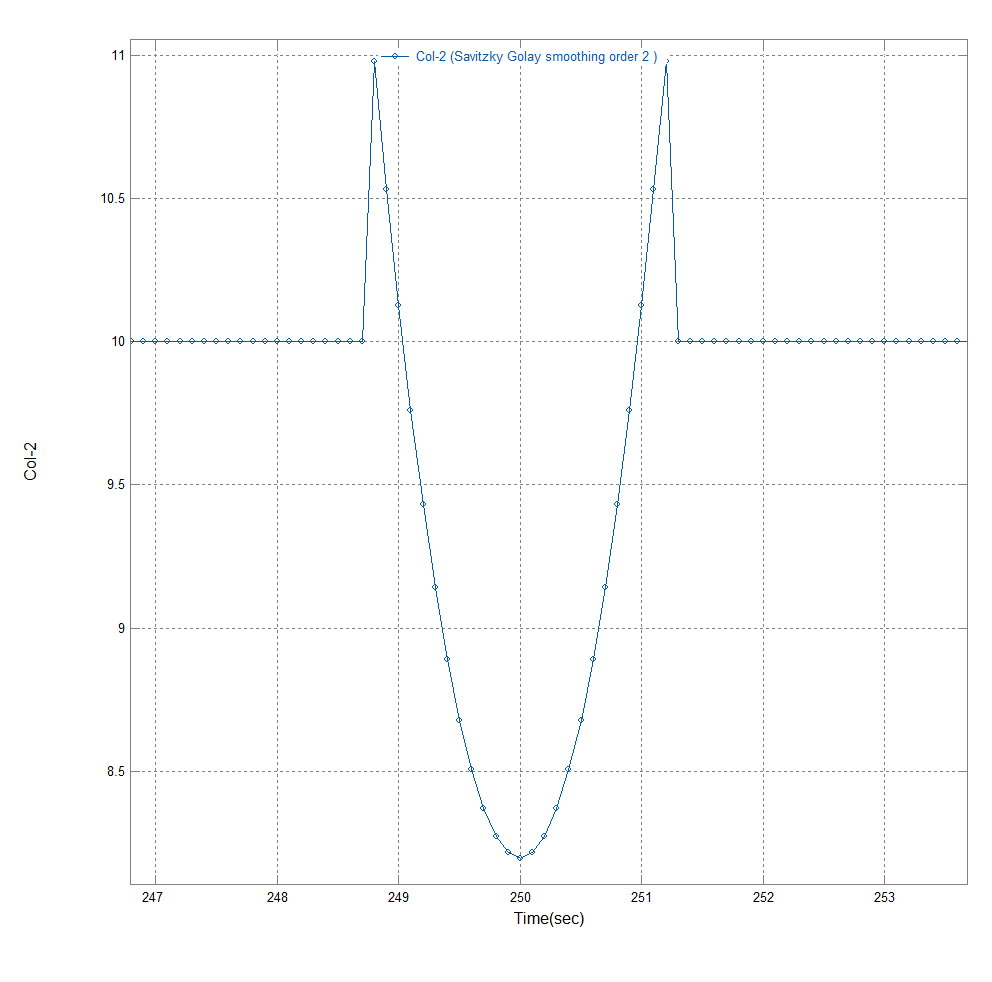
This takes successive subsets of 25 points (+/-12 points around a middle point), fits a polynomial of the specified order (using a least squares method) and replaces the y value of the middle point with the value of the polynomial at that x value. The polynomial order can be set to any value from 1 up, but internally is clipped at 10.

Using the same example data/scale as using in the “None” example above, with a polynomial order 2, this gives:

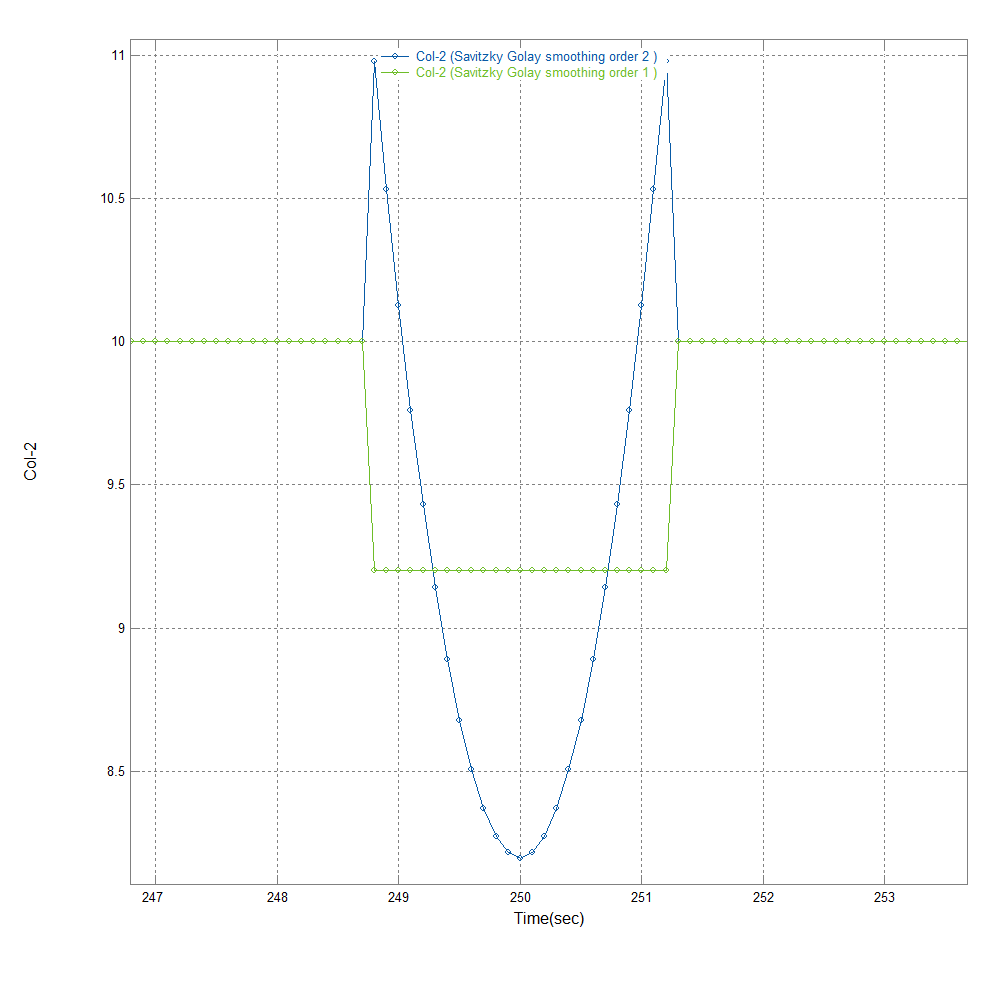


If the spikes on the original were noise, then this filter has reduced their amplitude.

Zooming in on a single (negative) spike we get:



This shape is the result of fitting a 2nd order polynomial to very spiky data – using a 1st order polynomial avoids this issue as seen by the green trace below:



If the data is expected to be smooth using a polynomial order >1 can be very effective at removing noise while keeping the shape of the underlying data – in particular peaks and dips will appear at their correct x value and normally very close to the correct y value.

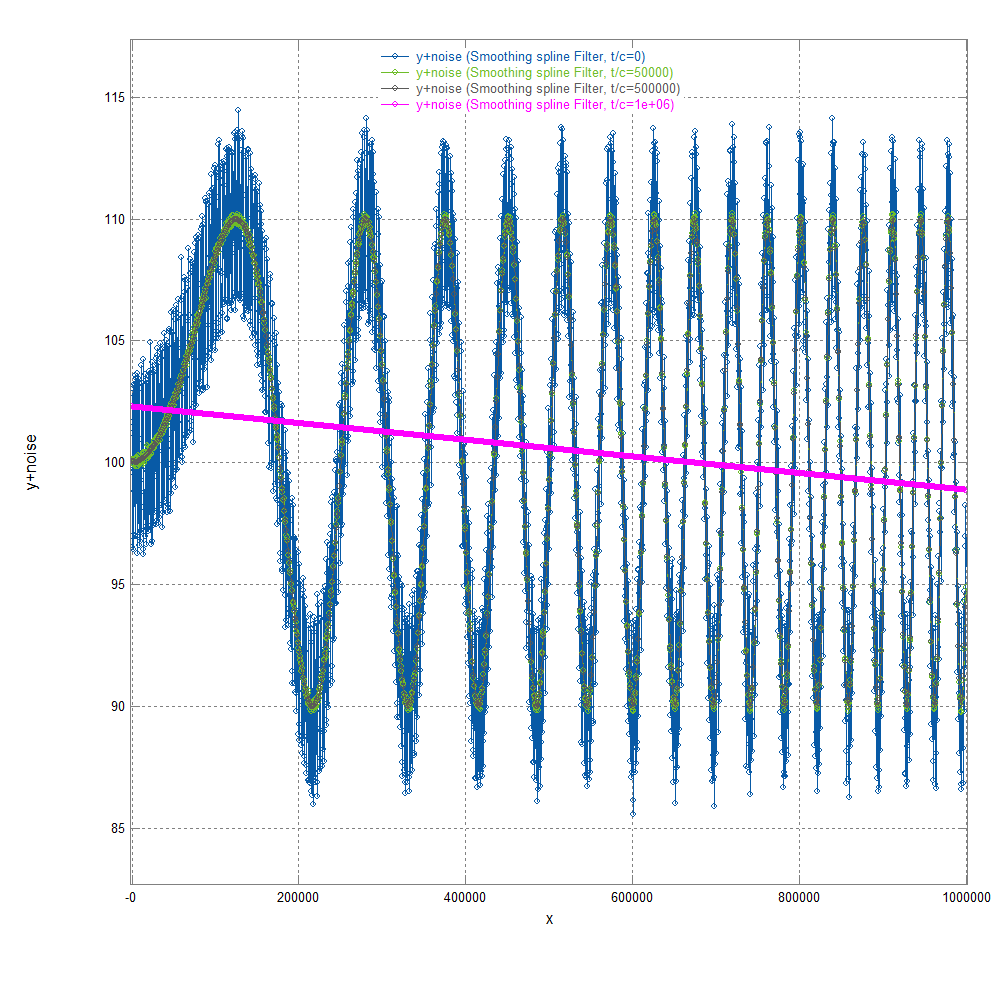
For more information see the paper A. Savitzky, M.J.E. Golay, “Smoothing and differentiation

of data by simpliﬁed least-squares procedures”, Anal. Chem. 36 (8) (1964) 1627–1639.

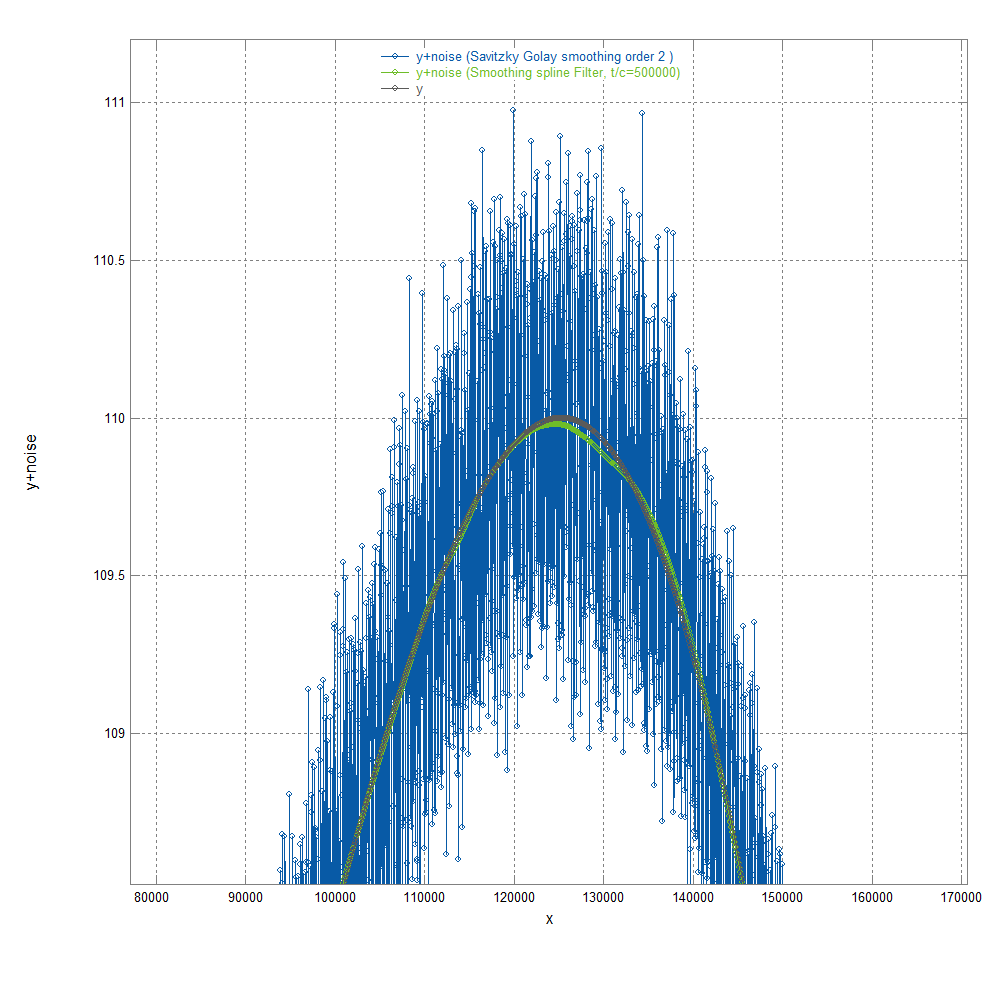
## Smoothing Spline Filter

This filter assumes the underlying data is smooth and allows you to trade-off how closely the input data is followed vs the smoothness of the resulting curve. If the filter time constant is set to zero the output of this filter is identical to its input (it fits exactly but has no smoothing), if the filter time constant is set equal to the total x range, then the result will be a straight line (which has a second derivative of zero and so is considered perfectly smooth). Intermediate values for the filter time constant will give a degree of smoothing (larger values giving more smoothing). Technically this filter works by fitting a cubic spline with a knot at every x value – for more details see for example Carl de Boor, "A Practical Guide to Splines", 1978, in particular chapter XIV. The implementation in Csvgraph is very fast, but does need a moderate amount of spare memory. Note most other implementations of smoothing splines use a value 0->1 or 0->infinity to vary the trade-off between fitting the data and smoothness – as stated above Csvgraph uses a “time constant” for compatibility with its other filters, and this approach normally allows a suitable amount of filtering to be quickly selected.

The image below shows the impact of varying degrees of filtering on a sine wave of increasing frequency with a significant amount of normally distributed noise, and a vertical offset of 100 added:



If we zoom in and also plot the noise free function and a Savitzky Golay smoothing order 2 for comparison, we see that the smoothing spline has very accurately recovered the noise free data from the very noisy input, while the Savitzky Golay smoothing leaves quite a lot of noise:



The time constant used is not normally very critical (you can get a closer fit to the real function with different time-constants in this example – but normally you would not have access to the noisy free data for comparison!).

As noted in the section on Savitzky Golay smoothing, fitting polynomials to rapidly changing waveforms can result in large overshoots and this effect can sometimes be seen with smoothing splines (which fit a series of piecewise 3rd order (cubic) polynomials to the data). In many such cases it is possible to select a suitable filter time-constant that gives an acceptable output using a smoothing spline filter, but smoothing splines give their best results where the underlying data is known to be smoothly varying.

## Linear Filter

This is a low pass filter with the order and time constant specified.

Using the same data as the Median filter above (order 1 and 2 sec time constant) gives:



Zooming into one of the spikes shows that while this filter has reduced the height of the spikes it has also extended their duration:



This is implemented as a ncascaded first order filters (giving 10\*order dB/decade). Order=0 gives no filtering. Order =1 gives same filtering as on versions of csvgraph prior to 2v2. The higher the order the longer the execution time, but this is unlikely to be an issue in practice as order is normally < 10.

In the other (textual) csvgraph window the -3dB frequency of the filter is given.

If more filtering than that provided by order 10 is required one of the other filter options might be more appropriate.

## Linear Regression

Selecting this filter will result in a straight line which is the best (least squares) fit to the input data.

In the example above this gives:



If you look at the other csvgraph window you will see the text:

Adding trace of Col-2 (col 2)

vs Time(sec) (col 1)

1000000 lines read from csv file

Best Least squares straight line is Y=2.38342e-13\*X+10.004 which has an R^2 of 2.95898e-16

Maximum value = 10.004 found on trace 0 (Col-2 (Lin. regression (y=mx+c))) at X=0

Minimum value = 10.004 found on trace 0 (Col-2 (Lin. regression (y=mx+c))) at X=0

Added column 2 in 0.750000 secs. 22.0 MB ram used

This gives the equation for the line (Y=2.38342e-13\*X+10.004) and the R^2 value which varies between zero (meaning a poor match to the underlying data, which is the case here) and 1 (a very good match to the data).

In this case it has removed the spikes in a similar way as the median filter, but the resultant y value is a little different (here it is the average Y value (10.004) rather than the median (10.000)).

In version Csvgraph 1v3 and above there are two versions of Linear Regression, y=mx and y=mx+c.

The first (y=mx) forces the line to pass through the origin (x=0, y=0) the second y=mx+c is the more general version.

Least squares linear regression minimises the sum of squared errors between the measurements and the fitted straight line. So, if the correct value is 10 and the estimated value is 12 the squared error is 4.

In Csvgraph 3v10 and above using a smoothing spline filter with a very large time-constant (at least equal to the total span of x values) gives identical results to a y=mx+c linear regression.

## GMR regression (y=mx+c)

This is another technique to fit a straight line to the supplied data.

Geometric Mean Regression (GMR) is also called Triangular regression. This method is less sensitive to outliers in the data than the (least squares) Linear regression method above.

This method minimises the sum of the areas of the right-angle triangles between the measurements and the fitted straight line.

## Minimum absolute error for y=mx+c

This is another method to fit a straight line to data, it minimises the maximum absolute error and as a secondary function when the minimum error is reached it then minimises the sum of the absolute errors (this means the best line is normally unique; if the maximum absolute error only was minimised then typically a large set of lines would have the same maximum absolute error). For example, if the correct value is 10 and the estimated value is 12 the absolute error is 2.

## Minimum relative error for y=mx+c

This is another method to fit a straight line to data, it minimises the maximum relative error and as a secondary function when the minimum error is reached it then minimises the sum of the absolute relative errors. For example, if the correct value is 10 and the estimated value is 12 the relative error is 2/10 (0.2 or 20%). Relative error may give a better fit if the y values change a lot, for example if y values range from 1 to 1000 then an absolute error of 1 is 100% of the lowest y value (1) but only 0.1% of the largest y value (1000). Using relative errors than a relative error of 1/100 (1%) would be an absolute error 0.01 when y is 1 and an absolute error of 10 when y is 1000.

### Which straight line fit should I use?

Each method has its place which is why csvgraph supports 5 ways to fit a straight line (4 techniques and one option to force the line to pass through the origin).

The plot below shows an example with all 5 options:



The blue curve on the graph above is what the other lines are trying to approximate. The minimum absolute error line (red) normally has its maximum absolute error at both ends and “the middle”, while the minimum relative error line (yellow) will have a steadily increasing absolute error (as it is trying to keep the relative error constant). The GMR line (purple) is normally reasonably similar to the linear regression line (least squares) (black) and that’s true here – and both will follow the curve reasonably closely giving a compromise between the absolute and relative errors. For this example, in terms of reducing maximum absolute errors, the minimum relative error line gives 5.8, the linear regression (y=mx+c) gives 2.05, GMR gives 1.98, linear regression (y=mx) gives 1.17 and the minimum absolute error line gives 0.97.

## Logarithmic Regression

Fits the equation y=m\*log(x)+c using a best (least squares) fit to the input data.

In the same way as linear regression (above), the 2nd csvgraph windows gives the fitted equation and the corresponding R2 value.

The supplied data file csvfun3.csv includes an example, see graph below.



In the 2nd csvgraph window you will see the fit (which in this case is exact):

Best Least squares line is Y=-2\*log(X)+3 which has an R^2 of 1.

## Exponential Regression

Fits the equation y=c\*em\*x using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Power Regression

Fits the equation y=c\*xm using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Reciprocal Regression

Fits the equation y= (m/x)+c using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Inverse Linear Regression

Fits the equation y=1/(m\*x+c) using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Hyperbolic Regression

Fits the equation y=x/(m\*x+c) using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Square Root Regression

Fits the equation y=m\*sqrt(x)+c using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



## Y=a\*x+b\*sqrt(X)+c

This fits the equation y=a\*x+b\*sqrt(x)+c using a best (least squares) fit to the input data.

The built-in maths capabilities of csvgraph allows us to create a suitable example as in csvfun3.csv X is the first column ($1).

Looking in the 2nd csv graph window the fitted equation is found:

Best fit found is Y=0.1\*X-2\*sqrt(X)+5

Max abs error of above curve is 4.76837e-07

The resultant graph is shown below:



## Y=a+b\*sqrt(x)+c\*x+d\*x1.5

This is similar to the function above but has one more term (d\*x1.5).

As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window.

## Polynomial in sqrt(x)

The general case (polynomials in sqrt(x)) is also available by selecting “Poly in sqrt(x) order:” and specifying the required order in the “order” box to the right.



For example, selecting order=2 will fit Y=a\*x+b\*sqrt(X)+c.

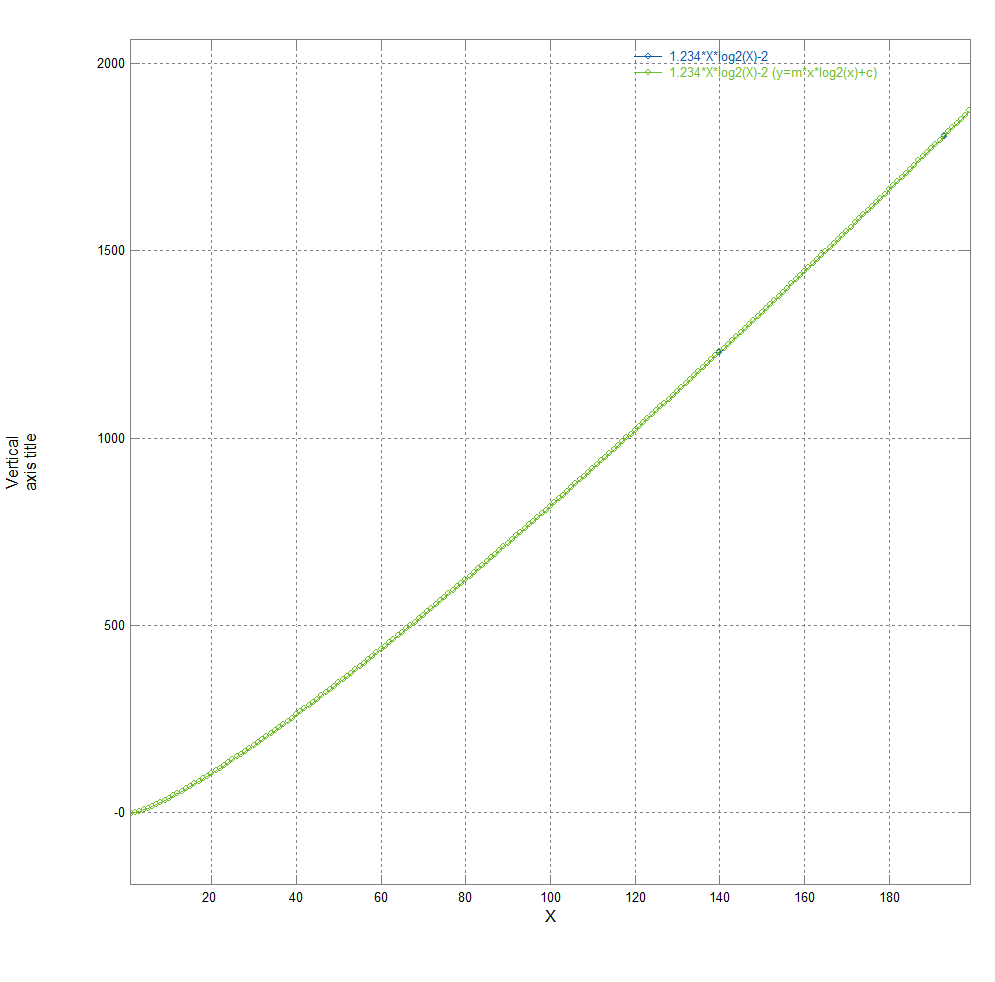
As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window. Not all powers of x may be present (only non-zero coefficients are shown).

Note that with a very large number of data points specifying a large order may result in a long wait for the results (e.g., order 10 will take ~ 0.5 minute on a reasonably modern PC, while order 20 will take about 1 minute both for 15 million data points). For orders of 1,2,3 it is more efficient to select that particular form from the menu rather than using the general fit with an order of 1,2 or 3 but the results should be identical.

## Y=M\*X\*log2(X)+c

This fits the function Y=m\*x\*log2(x)+c using a best (least squares) fit to the input data.

The supplied data file csvfun3.csv includes an example, see graph below.



As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window, in the case of the example the results were:

Best Least squares curve is Y=1.234\*X\*log2(X)-1.99999 which has an R^2 of 1

Max abs error of above curve is 0.00500488

Note that while the coefficient m is given for X\*log2(X) it is simple to calculate the value of m for a logarithm to a different base by divide m by log(2), for example to get m for ln (log base e) you need to divide m by ln(2)= 0.693147. Doing this has no impact on the accuracy of the fit.

Therefore, in the case of the example above this could also be written as:

Best Least squares curve is Y= 1.78029\*X\*log(X)-1.99999

Max abs error of above curve is 0.00500488

Csvgraph does actually supply the equation for y=m\*x\*ln(x) in its 2nd window.

## Y=(a+b\*x)/(1+c\*x)

This fits the rational function Y=(a+b\*x)/(1+c\*x).

As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window.

## Y=(a+bx+cx2)/(1+dx+ex2)

This fits the rational function Y=(a+bx+cx2)/(1+dx+ex2).

As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window.

## Rational (poly/poly)

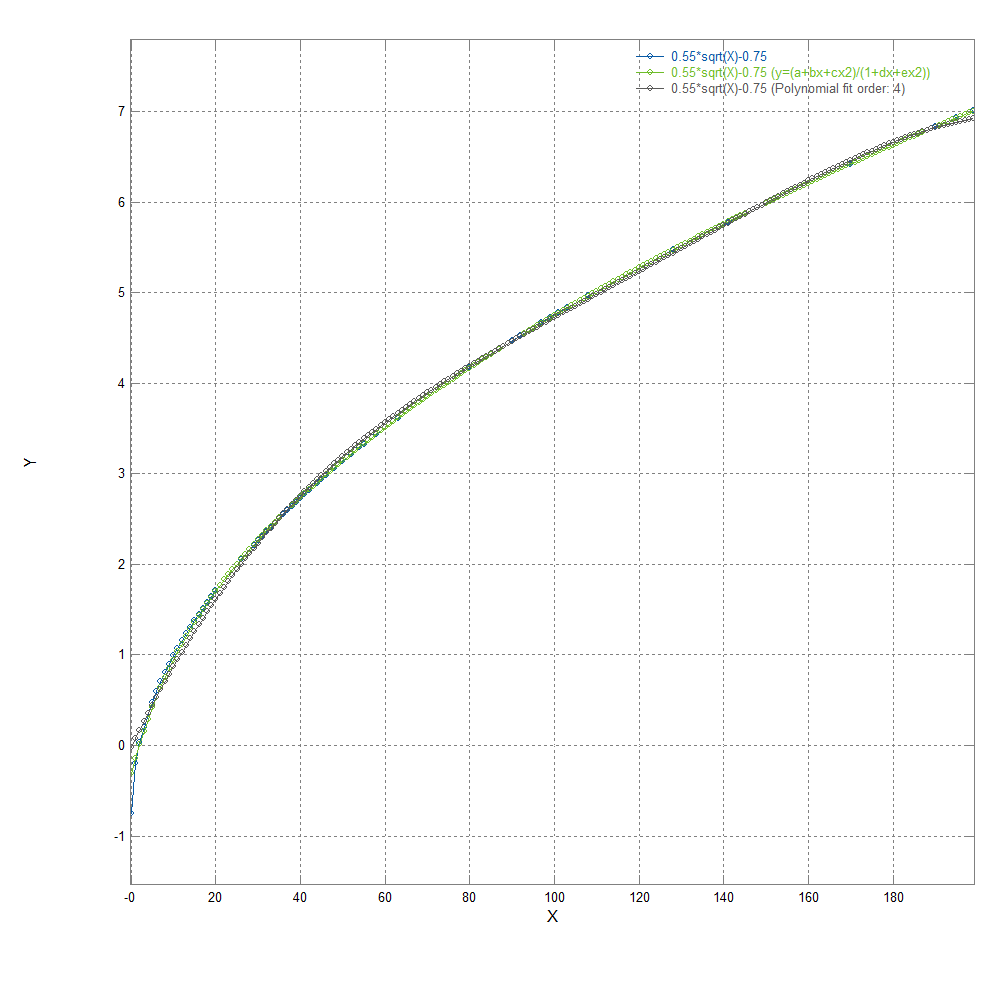
The general case (polynomial/polynomial) is also available by selecting “rational (poly/poly) order:” and specifying the required order in the “order” box to the right.



As usual, the fitted equation (with numerical values for the coefficients) is found in the 2nd csv graph window. Not all powers of x may be present (only non-zero coefficients are shown).

Note that with a very large number of data points specifying a large order may result in a long wait for the results (e.g., order 10 will take ~ 0.5 minute on a reasonably modern PC, while order 20 will take about 1 minute both for 15 million data points). For small values of order, it is more efficient to select the required equation directly from the available list rather than use this general case, but the results should be identical whichever approach is used.

Ratios of polynomials can give a better fit to many functions than using a higher order polynomial e.g.:



The polynomial fit has a maximum error of 0.726, while the rational fit has a maximum error of 0.439 with both using the same number of coefficients. Don’t forget in this case a sqrt(x) fit would give an exact match with effectively zero error with only two coefficients.

## Polynomial fit

This is a more general form than linear regression as the order of the polynomial can be selected.

If the order is 1 this gives identical results to the (least squares) linear regression y=mx+c option, but higher orders can give a better fit.

Using demo1M.csv and a 30th order polynomial gives:



Which is the polynomial with the equation (as before this is given in the 2nd csvgraph window):

Y=(((((((((((((((((((((((((((((6.97951274809e-137\*X-1.04691504709e-130)\*X+7.46518751585e-125)\*X-3.36717955513e-119)\*X+1.07841837595e-113)\*X-2.60977659582e-108)\*X+4.95825937558e-103)\*X-7.585603359e-98)\*X+9.51094773116e-93)\*X-9.8953321991e-88)\*X+8.61839059489e-83)\*X-6.32174846775e-78)\*X+3.92052333073e-73)\*X-2.05984542296e-68)\*X+9.172676001e-64)\*X-3.45829252723e-59)\*X+1.10109000948e-54)\*X-2.94854105574e-50)\*X+6.60274774225e-46)\*X-1.22708309395e-41)\*X+1.87412804271e-37)\*X-2.32316065069e-33)\*X+2.30046310128e-29)\*X-1.78304928219e-25)\*X+1.0534361431e-21)\*X-4.5791256059e-18)\*X+1.39448963045e-14)\*X-2.76963858661e-11)\*X+3.20180968581e-08)\*X-1.7483403854e-05)\*X+10.006685239

The file csvfun1.csv gives a more realistic example of polynomial fitting – the screen shot below shows a 1st, 2nd and 3rd order fit to exp(-i/10) – it can be seen that the 3rd order polynomial is a very good fit:



Note that a polynomial fit may require a quite high order (and thus a complex equation) to fit shapes that could be better described by one of the other equations listed earlier.

As an example, see the graph below which shows a Logarithmic regression fits it exactly with a simple equation (with two coefficients) y=m\*log(x)+c with m=-2 and c=3, whereas even an 10th order polynomial (with eleven coefficients) is not as accurate (while it captures the overall shape quite well it has an error of 0.56 which is 19% at x=1).

The accuracy of the polynomial fit is given in the 2nd csv graph window together with the equation – in this case this gives:

Polynomial approximating function is:

Y=(((((((((3.95024207162e-19\*X-4.19902777376e-16)\*X+1.92483796192e-13)\*X-4.9806768972e-11)\*X+7.99852880753e-09)\*X-8.25907300266e-07)\*X+5.5007335504e-05)\*X-0.00231419319452)\*X+0.0590981377101)\*X-0.902570371669)\*X+3.28864444205

with orthogonal poly : max abs error is 0.557087795729, rms error is 0.0613672600078

with conventional poly: max abs error is 0.557087795729, rms error is 0.0613672600078

This is why it worth trying all the simpler regressions offered by csvgraph before resorting to a high order (greater than 3) polynomial.



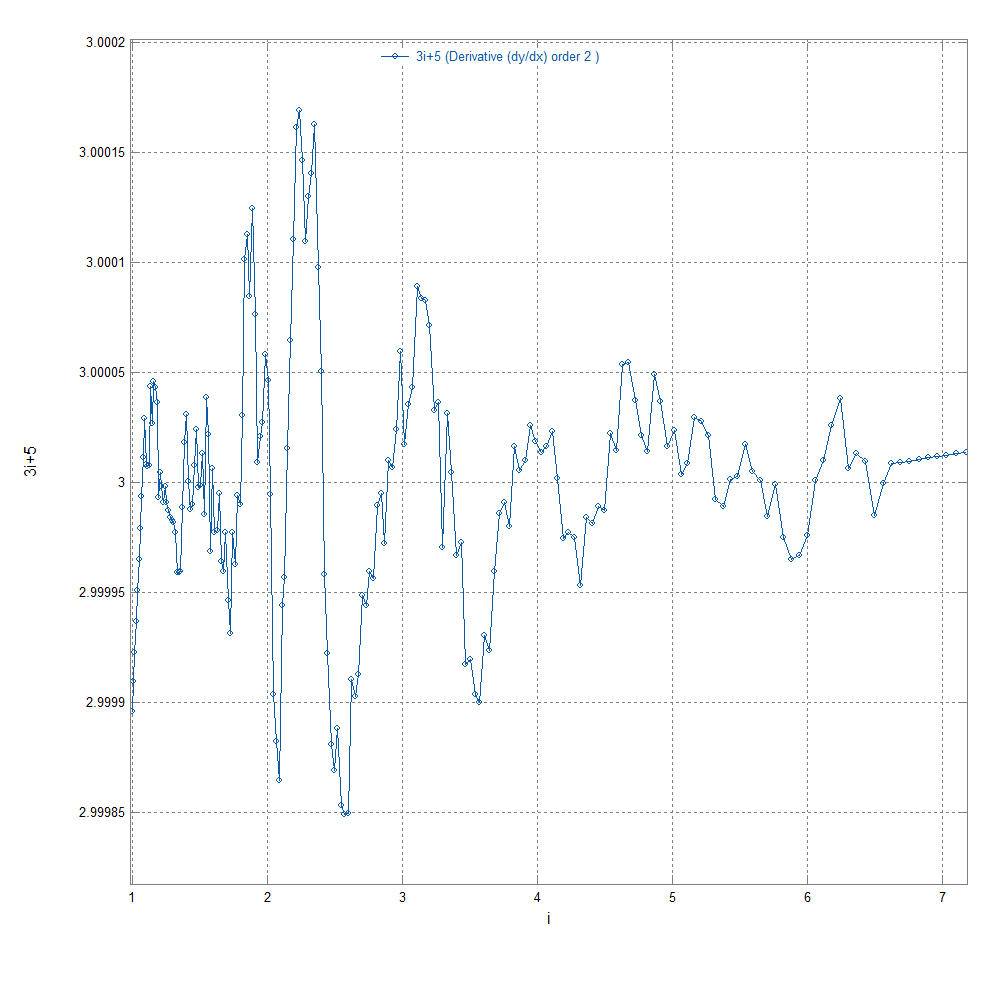
## Derivative



Selecting this option results in the derivative of the input (dy/dx).

As of version 3v9, the derivative is calculated using 17-point Savitzky Golay smoothing and then symbolically differentiating the fitted polynomial see section 7.3 . This means a degree of filtering is applied before the derivative is calculated, and the setting for the polynomial order also has an effect.

Example (from csvfun1.csv):



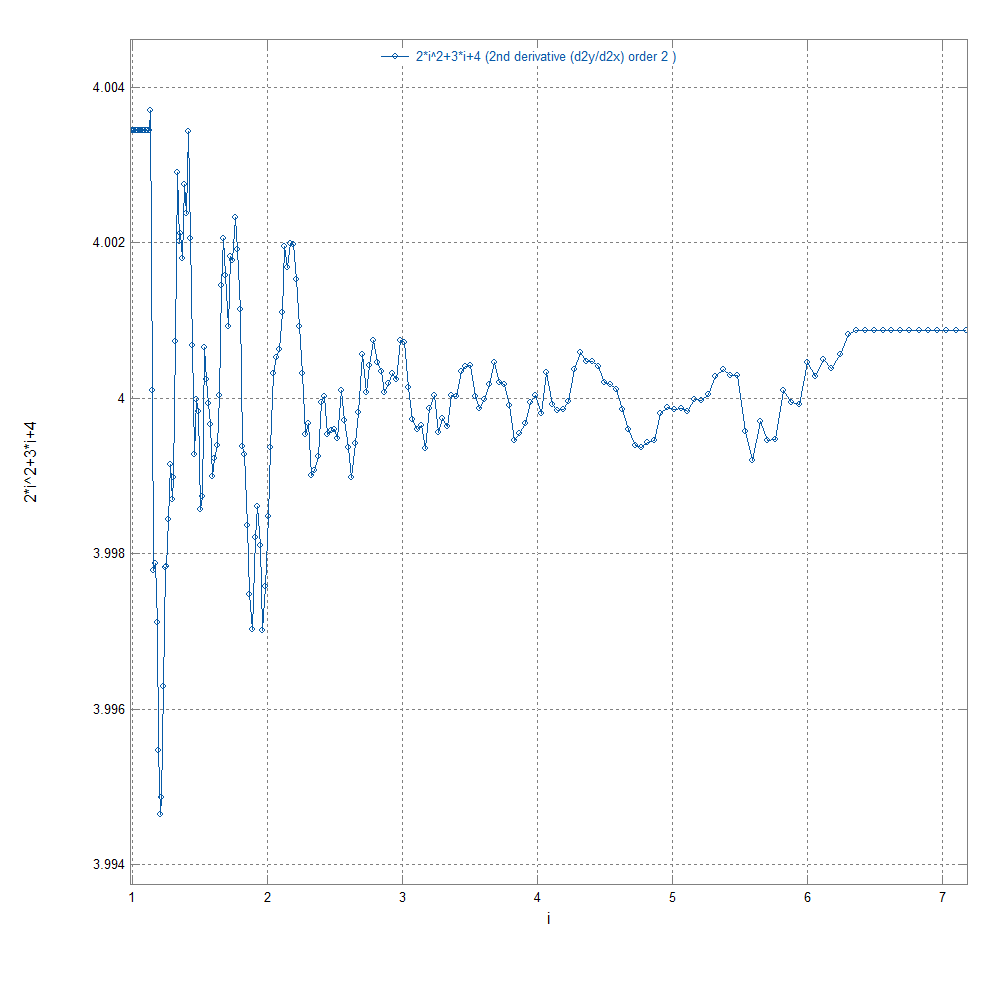
This example shows the “noise” resulting from the derivative (which should be a constant 3).

## 2nd Derivative

Selecting this option results in the 2nd derivative of the input (d2y/d2x).

This is a new function in version 3v9, the 2nd derivative is calculated using 25-point Savitzky Golay smoothing and then symbolically twice differentiating the fitted polynomial see section 7.3 . This means a degree of filtering is applied before the 2nd derivative is calculated, and the setting for the polynomial order also has an effect.

Example (from csvfun1.csv):



This example shows the “noise” resulting from the 2nd derivative (which should be a constant 4).

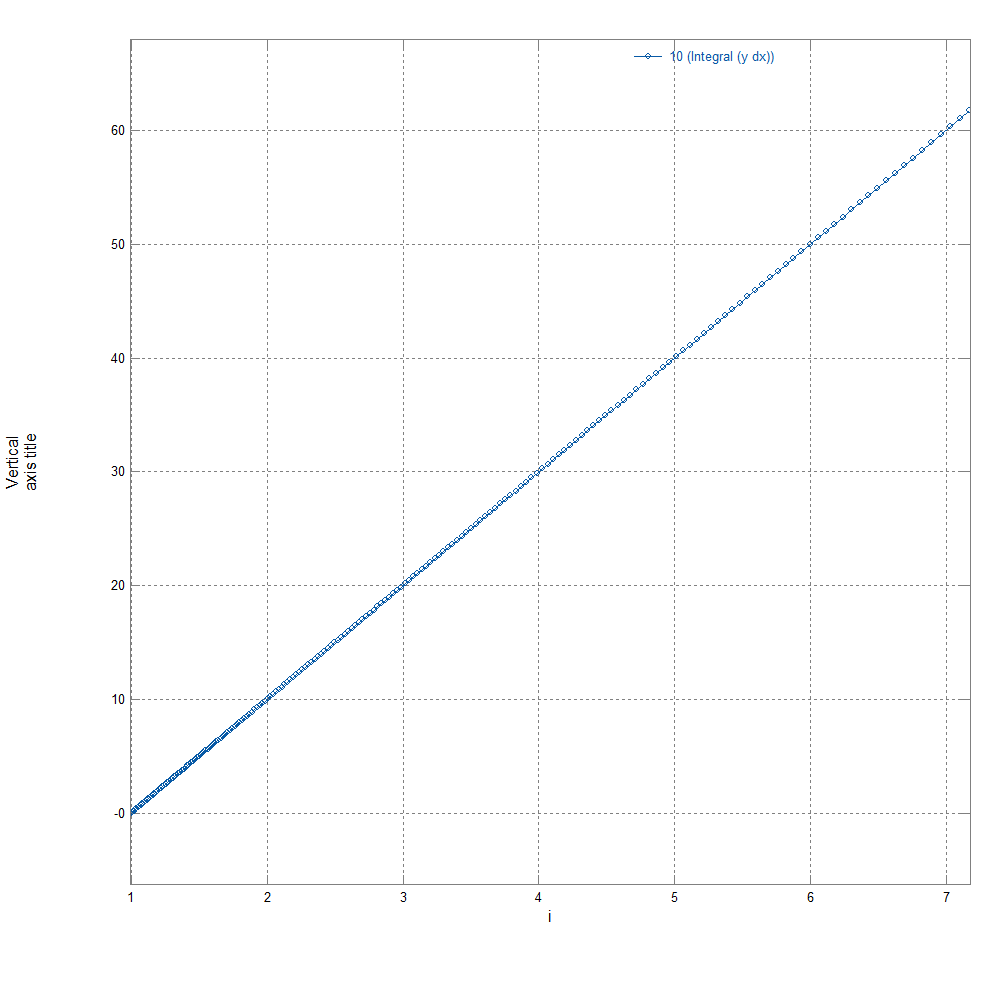
## Integral



Selecting this option results in the integral of the input

The value of the integral at the first x value on the trace is 0

Example (csvfun1.csv):



This example shows the integration of a constant which give a steadily increasing value.

## FFT

The FFT filters apply a Fast Fourier Transform to the data to create the frequency spectrum.

If the original data had an x axis in seconds the new x axis will be scaled in frequency (Hz).

The FFT assumes a constant time step is present in the supplied csv file – csvgraph will warn you if this is not true.

The result can either be viewed as a magnitude or log(magnitude) [ in dB], using the log form compresses the dynamic range so that small values are easier to see.

The supplied file csvfun2.csv has a number of waveforms that illustrate the Fourier transform.

The first graph below is column 7 of this file which shows 271Hz, 50Hz and 1Hz sinewaves – in the time domain (with “None” selected as a filter) this gives:



While in the frequency domain we can clearly see the 3 individual frequencies:



If we view the log(magnitude) plot:



It can be seen that what appeared to be zero when viewed on a linear scale are actually very small numbers – but again the peaks are clearly visible.

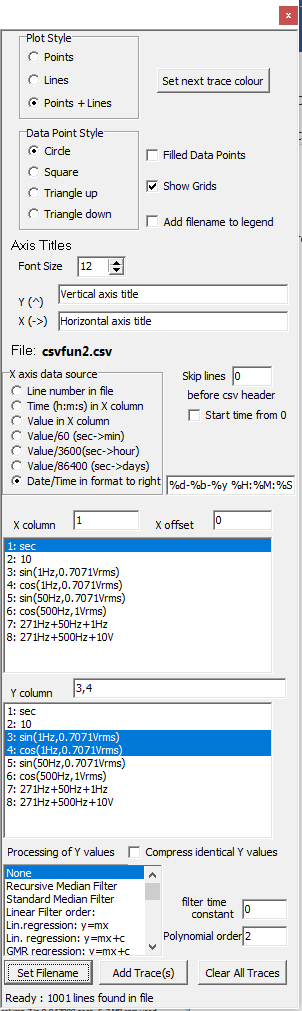
This effect (small non-zero values) can be reduced a little by using a “window” function and csvgraph offers the Hanning (sometimes called the Hann) window function and its effect can be seen below:



For more information on FFT’s and windowing functions please search the web or read one of the many books on this subject.

# Undocking the control panel

If the right-hand control panel is clicked in an area where there is no control then it will “undock” from the graph window to give a 3rd csvgraph window:



The graph will expand to take up the whole of the original window so this can be useful for use with smaller displays. Clicking the “x” at the top right of this control window will make it disappear, it can be recovered by using the main menu, Panel, visible.

# Help

Csvgraph is designed to be simple to use and most active areas of the screen have a tooltip that appears when you hover the mouse over them. From the main menu Help/About gives a brief summary of the available functions, while Help/Manual shows (this) manual assuming its available (see installation).

# Installation

Cvsgraph is a portable program which does not need installation.

Copy the file csvgraph.exe to any location on your computer (or run it from a USB-stick or similar).

From version 3.2 two versions are available, 32-bit and 64-bit – please select the version that matches the operating system version you are running. This can be found in settings – System – About under system type. If you are unsure the 32-bit version will run on both versions of the operating system but will be limited to using 4GB of memory (which will be adequate for all but extremely large files).

If you wish the Help/Manual function to work then copy csvgraph.pdf to the same directory (location) as csvgraph.exe. If you change the name of csvgraph.exe (e.g., to csvgraph64.exe) then the name of the pdf file also has to change (to csvgraph64.pdf in the case).

A shortcut on your desktop makes it simple to execute csvgraph.

The first time you run csvgraph you may see a Windows warning “The Publisher could not be verified. Are you sure you want to run this software” (or similar), you can either run anyway (the executable from github - <https://github.com/p-j-miller/csvgraph> should be safe) or compile your own executable from the source files (a free version of the required Builder C++ compiler is available at [www.embarcadero.com/products/cbuilder/starter](http://www.embarcadero.com/products/cbuilder/starter) ). Builder C++ 10.4 is required to build the 64-bit version. As of version 3.7, csvgraph is built with Builder C++ 11.3.

You may also get a similar message from your pdf reader the first time you use Menu/Help/Manual function, again you can accept this as you know why the pdf reader was invoked.

See the file LICENSE for details but csvgraph is free for both commercial and non-commercial use.

# Appendix A – expressions

Csvgraph allows expressions that are basically a subset of those available in the language AWK (which for expressions is very similar to C), and AWK also provides the syntax to select a column ($n to select column n, e.g., $5 to select column 5). You can also use the syntax $Tn to select the value of an existing trace on the graph e.g., $T1 is the value of the first trace plotted on the graph.

The operators available in decreasing priority are:

+/-constants, +/- (), +/-$n,+/-$Tn, variables

~,!

\*,/,%

+,-

>>,<<

<,>,>=,<=

==,!=

&

^

|

&&

||

Conditional expressions can be written using ? and : for example ($2==5)?3:4 which gives 3 if the value in column 2 is 5 and 4 if it’s not. Multiple ?: pairs can be used in an expression if necessary.

Two constants (pi and nan) are available and the following functions:

abs()

acos()

asin()

atan()

cos()

cosh()

exp()

log()

max(,)

min(,)

pow(,)

sin()

sinh()

sqrt()

tan()

tanh()

Functions take one argument except when shown “(,)” when they take two.

All the trig functions work in radians.

log () is log base e.

Two variables can also be used x (which is the current x value) and line which is the current line number. The values for these variables are automatically set and updated by csvgraph.

If a column ($1, etc.) does not contain a number then it returns nan (Not A Number). The constant nan is also available. Nans propagate through expressions as expected, including nan==nan is true and nan!=nan is false. If the whole expression evaluates to nan the line is skipped.

This means the expression

$3==nan?line:nan

will plot a point for every line that has a valid x value but does not have a valid number in column 3 with a y value equal to the corresponding line number.

Numbers can be integers, normal floating-point numbers (e.g., 0.1 or 1e-20) or hex numbers (0xnnnn) – the operators &,^,| (and, xor, or) work with 32 unsigned integers.

For efficiency expressions are “compiled” (and optimised) before a csv file is loaded.

For more complex processing on csv files the author recommends the use of AWK, see for example <https://github.com/p-j-miller/wmawk2> .

# Appendix B – defining date/time formats

As noted in the section Selecting columns of the csv file , it is possible to read in a date/time in a user defined format. That section also includes some examples. This appendix contains the complete list of all possible conversion specifiers.

Conversion specifiers follow a % sign, those defined are:

%a The day of the week (Monday, Tuesday,...); either the abbreviated (3 character) or full name may be specified. Both upper and lower case is allowed.

%A Equivalent to %a.

%b The month (January, ...) ; either the abbreviated (3 characters) or full name may be specified. Both upper and lower case is allowed.

%B Equivalent to %b

%c date and time [ this is equivalent to %a %b %e %T %Y ]

%C All but the last two digits of the year

Normally used before %y, but can also be used before %g. This means a %C also sets the ISO 8601 century

%d The day of the month [01,31]; leading zeros are permitted but are not required.

%D The date as %m/%d/%y.

%e The day of month (1-31)

%f fraction of a second (the values after the decimal point). The decimal point is implied

(so, needs to be in the format string if it’s actually present)

E.g., "%H:%M:%S.%f" will read 12:59:59.12345

%F Equivalent to %Y-%m-%d (the iso 8601 date format)

%G The ISO 8601 week-based year with century as a decimal number.

The 4-digit year corresponds to the ISO week number (see also %V etc).

This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead.

%G (or %C%g or just %g) should only be used with %V and the day of the week (%a,%A,%u,%w)

See also https://en.wikipedia.org/wiki/ISO\_week\_date and https://webspace.science.uu.nl/~gent0113/calendar/isocalendar.htm

%g Replaced by the same year as in %G, but as a decimal number without century (00-99).

%h Equivalent to %b.

%H The hour (24-hour clock) [00,23]; leading zeros are permitted but are not required.

%I The hour (12-hour clock) [01,12]; leading zeros are permitted but are not required.

%j The day number of the year [001,366]; leading zeros are permitted but are not required.

%m The month number [01,12]; leading zeros are permitted but are not required.

%M The minute [00,59]; leading zeros are permitted but are not required.

%n Any white space. Note that a space has the same effect

%p "am" or "pm"

%r 12-hour clock time using the AM/PM notation; equivalent to %I:%M:%S %p

%R The time as %H:%M.

%s seconds since the epoch as a multidigit signed integer.

As this implicitly includes the date and time it should not be used with any other conversion specifiers except perhaps as “%s.%f”.

Limits in years are + 2,147,485,547 to - 2,147,481,747

Note leap seconds are ignored when calculating a date/time from a given number of seconds (mainly as the dates when future leap seconds will be added cannot be predicted).

%S The seconds [00,60]; leading zeros are permitted but are not required.

%t Any white space.

%T The time as %H:%M:%S

%u Weekday as a number 1->7 where Monday=1 and Sunday=7

%U The week number of the year (Sunday as the first day of the week) as a decimal number

[00,53]; leading zeros are permitted but are not required.

If %Y or %C%y or %y together with %U and the day of the week (%a,%A,%u,%w) are all read by the format then a full unique date is defined.

%V The ISO 8601 week number of the year (Monday as the first day of the week) as a

decimal number (01-53).

If the week containing January 1 has four or more days in the new year, then it is week 1; otherwise, it is the last week of the previous year, and the next week is week 1. See %G above for more information.

%w The weekday as a decimal number [0,6], with 0 representing Sunday.

%W The week number of the year (Monday as the first day of the week) as a decimal

number [00,53]; leading zeros are permitted but are not required.

If %Y or %C%y or %y together with %U and the day of the week (%a,%A,%u,%w) are all read by the format then a full unique date is defined.

%x The date, the same as %D.

%X The time, the same as %T

%y The last two digits of the year.

When the format contains neither a C conversion specifier nor a Y conversion specifier, values in the range [69,99] refer to years 1969 to 1999 inclusive and values in the range [00,68] shall refer to years 2000 to 2068 inclusive;

Leading zeros are permitted but are not required.

%Y The full year. Leading zeros are permitted but are not required.

An optional leading sign (+/-) is also allowed. Limits in years are + 2,147,485,547 to - 2,147,481,747.

%z Time zone offset from UTC; a leading plus sign stands for east of UTC,

a minus sign or west of UTC, hours and minutes follow with two digits each and no delimiter between them (as in ISO8601 & common form for RFC 822 date headers). e.g., “-0500” or "+0000". The sign is always required.

This value is ignored by csvgraph.

%Z time zone name. e.g., “EDT”, "UTC", "GMT","AKST","ET" etc. 2, 3 or 4 letters is required.

This value is ignored by csvgraph.

% Replaced by %.

Any other character will match the same character in the input.

# Changes

1v0 - 3/1/2021 - 1st release on Github

1v1 – 6/1/2021 – Improvements to (this) manual.

- Bug fix to potentially incorrect DC component of FFT

- ability to access (this) manual with csvgraph using menu/Help/Manual

1v2 – 24/1/2021 – bug fix “inf” in csv file would be read as an extremely large number (infinity)

which then caused issues when csvgraph tried to scale numbers and draw the

graph

* Added many more options for “filtering” including exponential, power, hyperbolic and sqrt.

1v3 – 3/2/2021 - more curve fitting options added, y=mx, y=mx+c with GMR, minimum absolute

error and minimum relative error, and y=a\*x+b\*sqrt(x)+c.

2v0 – 17/2/2021 – Major internal changes to reduce RAM usage and improve speed.

No changes to function.

2v1 – 21/3/2021 – more curve fitting options added; y=a+b\*sqrt(x)+c\*x+d\*x^1.5, y=(a+bx)/(1+cx),

y=(a+bx+cx^2)/(1+dx+ex^2), polynomials on sqrt(x) of user defined order and rational functions (polynomial/polynomial) to a user defined order.

2v2 - 5/4/2021 - $T1 to Tn allowed in expressions to use values from existing traces on the graph.

Traces are numbered from 1. Invalid trace numbers (too big) return 0.

User can now set order of the linear filter. This is implemented as a nth order Butterworth filter (10\*order dB/decade). Order=0 gives no filtering. Order =1 gives same filtering as previously.

"filters" for integral and derivative added.

All filters now report progress as a % (previously min. abs error and min. relative error did not report progress and they can be quite slow).

Option to skip N lines before csv header added for cases where csv header is not on the 1st row of the file

Added column numbers to X column and Y column list boxes to make it easier to select columns when names are not very descriptive (or missing).

2v3 - 20/1/2022 – sorting of x co-ordinates (when required) makes use of all available processors

and can be significantly faster.

Added curve fit for y=m\*x\*log2(x)+c .

2v4 – 3/2/2022 – bug fix, using variables $Tn in an expression with a set of x-axis values that were

not in numerically increasing order (and so needed to be sorted) did not work correctly in 2v2 and 2v3 – sorry.

2v5 – 16/2/2022 – Added option “start time from zero”.

Improved error reporting for incorrectly formatted csv files.

Minor bug fixes.

2v6 – 27/2/2022 – Median1 filter improved, for <=10,000 data points it’s now exact and it’s (much)

more accurate for traces with a larger number of data points.

Position of trace legends moved left so more text can be seen.

2v7 – 22/3/2022 - prints -3dB frequency for linear filter.

Display to user 1 example of every type of error in csv file.

If dates are present on some lines of the csv file, then flag lines without a date as a potential error when x column is set to date/time.

New (exact) median (recursive median filter) algorithm, which falls back to sampling if the execution time becomes long. Median and Median 1 renamed as recursive median filter and standard median filter.

2v8 - 23/5/2022 – added ability to read a date and time as the x value

csvsave added % complete and (significantly) sped up writing to file.

csvsave interpolates if required so x values do not need to be identical on all traces.

2v9 – 7/6/2022 – bug fix – if x-offset is not equal to 0 and multiple traces added incorrect x-offset

was applied to the 2nd trace added onwards.

Higher resolution internally on reading dates & times, so if “start time from zero” is

ticked the results may be slightly more accurate.

3v0 – 15/8/22 - Never released – 1st 64-bit version

3v1 – 17/8/22 - Never released. Source code refactored with “common files”

(that may be used in other programs) separated out. Some files moved from C++ to C where they were pure C and C++ compiler generated lots of warnings.

3v2 – 14/9/2022 – 1st 64-bit release. 64-bit version will use all available memory, while 32-bit version

is limited to 4GB of ram, apart from that functionality is identical. The 64-bit version can read files with more than 232 lines, the 32-bit version will run out of RAM before that many lines can be read.

Minor changes:

When the Scales menu was invoked multiple times previously the scaling would change very slightly – fixed.

Allowed range of font sizes for main title and X/Y axis titles expanded.

FFT now makes use of multiple processor cores if they are present to deliver faster results.

3v3 – 26/9/2022 – internal changes to allow even larger files to be read with the 64-bit version.

No change to functionality.

3v4 – 1/10/2022 – issue with dpi awareness with multiple monitors on 64-bit version – fixed.

No change to functionality.

3v5 – 6/11/2022 – on a right mouse click when a line is selected then the slope of this line (dy/dx) is

given as well as the coordinates of both end of the line. A filename can be given on the command line (on earlier versions this did not work for the 64-bit version). An expression containing a function is now allowed for the y axis (e.g. max($2,0) ).

3v6 3/6/2023 –

Long column headers now cause a scroll bar to be automatically added to the X & Y listboxes so they can be fully seen.

Save x range on screen as CSV added to File menu.

Option (tickbox) added to add basename of filename to legends of traces on the graph, which is useful if the same column is read from multiple files.

Y axis title automatically added unless user specifies one (based on column header of 1st trace added).

Added option to load X as Value/60 (sec->min), Val/3600 (sec->hrs), val/86400 (sec->days).

Error handling for X values in a user defined date/time format improved, and trailing whitespace now allowed.

3v7 10/6/2023 –

“Show legends” tick-box added.

Changed to using Builder C++ 11.3 compiler.

Title is now centred above the graph.

Minor changes to sizes, fonts, etc in csvgraph.

3v8 4/7/2023

csvsave where 2nd trace had less points than 1st trace caused an error - fixed.

3v9 26/2/2024

Can now open a file that excel already has open (and error messages are better on failing to open files)

Better trapping of user pressing a "command" button while a previous command is still running.

Derivative now uses 17-point Savitzky Golay algorithm with user specified order (1->10 is actually used, can be set 1->infinity by user).

Savitzky Golay smoothing added as a filtering option (25 points, with user specified order, 1->10 is actually used, can be set 1->infinity by user).

Added 2nd derivative (d2y/d2x) to list of filters which uses a 25-point Savitzky Golay algorithm with user specified order (1->10 is actually used, can be set 1->infinity by user).

If a number is missing in a column referred to in an expression this will be set to nan.

Added constant "nan" for expressions.

If an expression evaluates to nan the line is skipped so this can be a powerful way to select points for csvgraph to display

Added "variables" x and line to expressions. x is current x value, and line is current line number.

Updated expression handler so nan==nan and nan!=nan work as expected in expressions.

This allows for example $3==nan?x:nan which shows all invalid rows in column $3

3v10 6/4/2024

Smoothing spline filter added.