PolyFit

COLLABORATORS

	TITLE : PolyFit		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PolyFit

1.1 PolyFit Guide

```
---- Contents ----
```

1.2 Disclaimer

```
          You may use and copy PolyFit if you stick with the following rules ↩
             :

1)   No profit is made by selling PolyFit to others. A small fee
     for copying, disk costs etc. may be asked. This should not
     be more than $5. Also, this price may not be increased by supplying
     translations of the manual etcetera.

2)   PolyFit is distributed in its original form. No changes are made
     and this documentation file and the supplied examples are distributed
     with it.
```

3) PolyFit is not distributed as part of a commercial product, unless you have the written permission of the author. Also, PolyFit is not distributed on a disk or other data carrier which contains commercial data as well.

4) The author can not be held responsible for damage or data loss, which are caused by failures in the program, incorrect use of the program, errors in the manual or any other reason. All responsibility remains with the user of the software.

5) PolyFit is Freeware. This means you need not to pay for using it. However, if you appreciate this program and like to see new versions or other improvements, it is not forbidden to send cash of any kind to the author. If you have any bug reports or suggestions for improvement then they are even more welcome and you can send me a letter to my
address
.

1.3 Introduction

PolyFit stands for "polynome fit". This program uses the

Method of least Squares
, invented by
Gauss
, to calculate the best curve
to fit a given set of points in a graph.
For example, suppose you had a piston filled with a gas and you would measure the volume of this gas at different temperatures. You could then draw a graph of these points, with the temperature along the x-axis and the volume along the y-axis (see the example file T-V.data). If you then want to draw a line through these points, you could take your ruler and draw a line at best guess. Also, you could take your Amiga and PolyFit and enter the points. PolyFit generates the (mathematically) best fitting line, gives you the function of this line and shows a graph with your points and the line in it. From the results of the calculation you could get the constant in Gay-Lussac’s law or the absolute temperature zero.

Apart from drawing straight lines through a set of points, PolyFit can also draw straight lines that are forced to cross the origin, exponentional curves and polynome functions of up to the 16th degree. (Theoretically, this program needs no maximum degree, but degrees greater than 16 are not making sense and very high degrees would give erratic results because of the limited precision of the floating point variables.)
The graphs can be scaled lineary or logarithmically in both the x- and y-direction.

The rendered graphs can be saved as IFF-ILBM pictures, or printed using the preferences printer-settings.

1.4 Starting PolyFit

PolyFit can be started form workbench or CLI/Shell.

To start PolyFit from CLI/Shell just type PolyFit, or give the name of a previously created datafile as a load file:

```
   PolyFit [datafile]
```

To start PolyFit from workbench, double-click on the PolyFit icon, or double-click on the icon of a PolyFit-datafile. You can also load datafiles using the multiselect method:

Select a datafile icon by clicking it once. Now press a shift key and while you keep shift pressed down, double click on the PolyFit icon.

If you select a datafile, whether by clicking on its icon or by multiselect, the chosen datafile will be loaded on startup. If you have more datafiles selected (with multiselect) only the first selected file will be loaded.

1.5 Entering Data

If you started PolyFit without any datafile selected, you are presented with a window, which is dominated by a listview-gadget that at that time will be empty. In this gadget the entered data will be shown. On the left-hand side the x-value and on the right-hand side the matching y-value. On top of this gadget are three string gadgets. Two labeled X and Y respectively and one with a title. The labels X and Y can be changed to reflect the nature of the data you are entering. For example, if you had measured the volume of a certain amount of gas as a function of the temperature, you could change X to "temperature" and Y to "volume". The title can be used to enter some general info about your data. This title will later be shown on top of the graph you can generate.

The data itself can be entered in the two boxes underneath the listview gadget. In the left-hand one, you can enter an x-value, in the right-hand one, you enter the matching y-value. If you press <return> after having entered your x-value, the cursor will automatically be placed in the y-box.

If you have entered an x plus an y-value, you can add these to the data list by clicking on the 'Add'-button. If you select the 'Auto-add'-button, the x,y-set will automatically be added whenever you press <return> when the cursor is in the y-box.

Using this method you can keep entering points untill all your data is shown in the listview-gadget. If you enter more than ten points, only ten of these will be shown in the listview-gadget, but you can select which part you want to see by dragging the bar on the right of the listview-gadget.

If you accidently entered a faulty value, you can select it in the listview-box. The x and y value of that set will be placed back in the x and y-boxes. You can now change the values and replace them with the faulty

ones by clicking on the 'Replace' button. (Note that if you have auto-add selected, the data will be automatically replaced whenever you press <return> in the y-box). You could also add the values to the data, by clicking the 'Add'-button. This way, both the old and the new values will be in the listview-box.
Finally, you can remove the values from the listview-box by clicking on the 'Delete'-button. Note that the x and y values will remain in their boxes, so that if you accidently select 'Delete' you always can add them again to the data by clicking on 'Add'.

For the sake of input/output of data, some menuitems are available in the 'Project'-menu.
'New' (shortcut: Amiga N) will clear all data entered. If your data was not saved yet, it will first ask if you really want to do this.

'Open' (shortcut: Amiga O) will open a datafile and load its data into PolyFit. If you have data that is not saved yet, you will be asked if you are sure. PolyFit saves its data in an IFF-like format. If you selected a file that was not of the IFF-type an error-message will appear saying 'file of the wrong type'. If the file indeed was an IFF-file, but did not contain PolyFit data, you will be presented with an appropiate message.

'Save' (shortcut: Amiga S) will save the data to the file that was last opened, whether for loading or saving. If no file was opened before you will be asked to select a filename first.
CAUTION: since save automatically uses the last-opened file to save the data to, old data can easily be overwritten. For example, if you open a file for loading data, then select 'New' and then select 'Save', an empty data set will be written to a file that previously contained data.

'Save As' (shortcut: Amiga A) will also save data to a file, but will always ask you to select a file to write the data to.

'Print' (no shortcut) will print the data as a table to your printer.

'Print As' (no shortcut) will print data to a printer or a file. You will be asked to select an output-file/device.

'About' (shortcut: Amiga ?) will show some info about the version of PolyFit and the author.

'Quit' (shortcut Amiga Q) finally, will quit the program. If there is some data that is not saved yet, you will be asked if you are sure you want to quit.

1.6 Processing Data

If you have entered all your data, these can be processed further. First of all you have to select what kind of function you want to fit to your points. This is being done with the radio-buttons on the right-hand side from the listview-gadget. 'Linear' will calculate the best straight line to fit your points. 'Linear through origin' will also calculate a straight line, but this one is forced to cross the origin. This will only give reliable results if the points represent a line that only nearly misses the origin. Use of these fit could be seen as 'cheating', so avoid using it

when it is not necessary. ’Polynome’ will calculate a polynome-function of the nth degree to match your function, where the degree n can be set with the slider gadget below the set of radiobuttons.
Note that n must be smaller than the number of points you entered. Theoratically, the curve will fit the points better when n is higher, but also the curve will incline to ’wave’ through your points (enter about ten fake points and draw a polynome of the nineth degree, then you will know what I mean with waving...). Also calculation time will increase dramatically when n increases, so keep n as low as is desired.
Note also the a polynome of the 0th degree has a constant as result. This constant reflects the average y-value of your points. A polynome of the 1st degree is a straight line and will produce the same result as a ’Linear’ fitting. A polynome of the 2nd degree is a parabola.

If you have chosen the curve type, you can press one of the buttons ’Calculate line’ and ’Graph’. ’Calculate line’ calculates the function of the desired curve and shows its coefficients in a window. The contents of this window can be printed to the printer or a file by pressing this window’s ’Print’ button.
’Graph’ draws a graph in which the points and the calculated curve are shown. If you have not yet calculated a curve, or made changes to the data since the last calculation, the curve is automatically recalculated; the results however will not be shown.
The graph will be drawn on a seperate screen. The screen’s resolution can be set with the menuitem ’Set screenmode’ in the Settings menu.

In the Settings menu, you can select how the x- and y-axis should be scaled. Lin stands for linear and this means a normal scale: all divisions have the same width. Log stands for logarithmic and this means that the width of a division is ten times wider than that of the previous division.
A logarithmically scaled axis is only possible if there are no points that have a coordinate on that axis that is smaller than or equal to zero. In this case you will be warned and told that a linear axis will be used.

It is useful to use a logarithmically scaled y-axis when you suspect there is an exponentional relation between x and y. If this relation exists the line will be straight in this mode. It is useful to use a logarithmically scaled x-axis and y-axis (double logarithmic) when you suspect that y equals a certain power of x, thus if y = a*x^b. If this is correct the line is a straight with an angle equal to b.

Other graph settings include ’Grid’, ’Curve’ and ’Black & White’. ’Grid’ displays a grid in the backgroundof the graph. If ’Curve’ is set, the function’s curve is drawn in the graph. If ’Curve’ is not set, only the points will be shown. You could use this to see which kind of fit you could use best. ’Black & White’ draws the graph with only black, white and grey colours. This will probably give better results on most printers.

When you have seen enough of the graph press the left or right mousebutton, or just press a key and you will be taken back to the main window.
From here you can save the graph as a IFF-ILBM picture by selecting ’Save as IFF’ from the ’Graph’-menu, or print the graph with ’Print graph’.
’Print graph’ uses the preferences setting for setting the graph on paper. So if you think the print-out could be done better, try to change your printer(gfx)-preferences settings, especially the density setting.

Added later was the possibility to view, save and print a graph of the residu. This graph shows the same series of points as the normal graph, but now the fitted curve is subtracted from the point's y-values. This shows the curve as the line with y = 0. This enables you to see exactly how the points deviate from the curve.

1.7 examples

I provided a set of example-datafiles with this program. In this chapter I will explain their contents and how to use them in PolyFit.

- 68030speed:
This is the most simple example. It contains only four points. It is data from some measurements by the redaction of Amiga Magazin. The points show how much the processing speed (particulary the integer operations) of an Amiga is increased when installed with a 68030 CPU at various clock speeds. It is obvious that a CPU with twice a clock speed, should have twice the performance. Therefore the points should be fit using linear fit. You will see that the line does not cut every point. This is because the frequencies are not exactly given. 25 MHz could be 25.34123... MHz in real life. Also the speed increase can't be measured too exactly. But with the results of the calculation you could very well predict what the speed of a 100 MHz 68030 would be (can you imagine!)

- Delivery:
This file contains data I found as example for a linear fit in a math book. It shows how many days it takes a transport company to deliver some goods, as a function of the distance. Because the unity of time is chosen quite large (days), the y-data is fairly unexact. But because there is a lot of data we can make a prediction about the average time it takes to deliver. This is also a linear function. Note that if the distance is zero, it still takes about one tenth of a day to deliver. This is overhead time for loading/unloading. You could also try to match this function with a polynome of the nineth degree. If you look at the graph of that function, you'll see that that function never will predict the delivery time.

- T-V:
This is also data for use with a linear fit. I got this data from a book on Thermodynamics. It is just to show what PolyFit was made for: to derive functions from scientific data.

- exponentional:
I believe this data were to represent the weight of a bacteria colony as a function of days gone by. You can draw an exponentional curve through these points and calculate what the weight was on day 0. Note that 3rd and higher order polynomes fit quite well either. This is because an exponentional function can be approximated by a high order polynome. This function converges quickly for exponentional functions. But when you use a logarithmically scaled y-axis, you'll see that the 3rd order approximation is not as beautiful as it seemed with lineary scaled axises.

- Thermo:
This data was taken from a book of tables for Dutch schools. It gives the voltage over a thermo-couple as a function of the temperature. With PolyFit you can generate a function that matches this data. Such a function gives you the possibility to calculate a temperature for every voltage you read.

This makes it possible the use the device as a thermometer. But it takes at least a 3rd order polynome to get a reliable fit of the points in the graph.

- TestDrive2:
This one is a joke actually. I played this game lately and wondered how the computer generates a score out of the averege speed you had on the first stage. After playing this stage a few times and entering the scores in PolyFit I discovered that it represents a parabolic curve. After calculation you can see that you always get at least 1 point, even if your average speed is zero.

1.8 Comments and bug reports

If you have any comments on this program, suggestions for improvements or if you have found a bug, please send me a letter at:

Camiel Rouweler
Weldam 2
5655 JG Eindhoven
the Netherlands

I am sorry, but I do not have an E-mail address or something like that, so you will have to use the regular mail.

P.S. I would be pleased to hear that people are using this program and like it, so if you do, please send me a letter or postcard.

1.9 Mathematics

WARNING: this chapter is purely mathematical. If you don’t like maths or just are not interested in it, please don’t bother to read this.

Note: in this chapter the ^ character will be used, to write an involution, and * for a multiplication, as is use in most programming languages. SUM replaces the capital sigma, that is used in mathematical literature to define a sum.

The Method of Least Squares is a way to optimize a function mathematically, so that it deviates the least possible from the given points. Therefore, the function’s formula must already be given, with only some constants to be filled in.

Presume that you have a series of points with the co-ordinates:

    (x1,y1), (x2,y2), (x3,y3), ... , (xn,yn),

with n the total number of points. Now you want the best function f(x) to fit to these points. A series of points is best described by a function if f(x1) is approximately y1 and f(x2) is approximately y2 and so on. We now define the error of f(x) as the sum of the squares of the differences in f(x1) and y1, f(x2) and y2.... So

```
   Error(f) = (f(x1) - y1)^2 + (f(x2) - y2)^2 + ... + (f(xn) - yn)^2
            = SUM (f(xi) - yi)^2, i = 1...n                          (1)
```

The best function f is of course the one with the least error. To find this function, we think of f not only as a function of x, but also as a function of all its parameters. By choosing the parameters and their meaning in the function, we define the nature of the formula. For example, for a parabolic function we would define f as:

```
   f(a0,a1,a2,x) = a0 + a1*x + a2*x^2.                               (2)
```

With this function we can write every possible parabola.
To find the function f that has the least error (with a given set of points) we can differentiate (1) to all its parameters. We then get in this case three formulas, since we have three parameters (a0, a1 and a2).
They look like this:

```
    d
   --- ( SUM (f(xi) - yi)^2 ) = 0                                    (3a)
   da0

    d
   --- ( SUM (f(xi) - yi)^2 ) = 0                                    (3b)
   da1

    d
   --- ( SUM (f(xi) - yi)^2 ) = 0                                    (3c)
   da2
```

In the case of our parabolic function this results in:

```
   SUM ( 2*(a0 + a1*xi + a2*xi^2 - yi) ) = 0,                        (4a)

   SUM ( 2*(a0 + a1*xi + a2*xi^2 - yi)*x ) = 0,                      (4b)

   SUM ( 2*(a0 + a1*xi + a2*xi^2 - yi)*xi^2 ) = 0.                   (4c)
```

And if we put all the y’s on the right side of the = sign we get:

```
   SUM ( a0 + a1*xi + a2*xi^2 ) = SUM ( yi ),                        (5a)

   SUM ( a0*xi + a1*xi^2 + a2*xi^3 ) = SUM ( yi*xi ),                (5b)

   SUM ( a0*xi^2 + a1*xi^3 + a2*xi^4 ) = SUM ( yi*xi^2 ).            (5c)
```

We now have three equitations, with three variables, so this can be solved. But to write this out is a lot of work (you can do it for two variables, but I once tried three, but after half an hour I gave up, but if you are used to working hours on a boring job, you should really try it!). But then, we have a computer to solve this for us. In order to make this possible, we write the three equitations in the form of a matrix, with a width of 4 and a height of 3. It then looks like this:

```
[  1           SUM( xi )   SUM( xi^2)  |  SUM ( yi )     ]
[  SUM( xi )   SUM( xi^2)  SUM( xi^3)  |  SUM ( xi*yi)   ]
[  SUM( xi^2)  SUM( xi^3)  SUM( xi^4)  |  SUM ( xi^2*yi) ]
```

We solve this matrix-equitation (I hope you know how to do that; if you don’t, trust me that it is not difficult, but too complex to explain in a few lines). We then get a matrix of the form:

```
[  1   0   0  |  s0  ]
[  0   1   0  |  s1  ]
[  0   0   1  |  s2  ]
```

This says that a0 = s0, a1 = s1 and a2 = s2. And with a0, a1 and a2 known to us, we also know f, so the best function f can be found by solving a matrix equitation. The matrix only contains sums of integer powers of x and sums of integer powers of x multiplied by the matching y, when f is a polynome.

In this same way we could find a best function for every possible function. The only restriction is that you must define, what kind of function you are trying to find. If the points you have match a sinus, but you try to fit it with a straight line, the results won’t be great (although you will find a function that is the best, the best of the straights that is).

For more and deeper information about the Method of Least squares and matrix equitation, I recommend reading a good math book about the subject.

1.10 Gauss

CARL FRIEDRICH GAUSS (1777-1855), great German mathematician. He already made the first of his great discoveries as a student at Helmstedt and Göttingen. In 1807 he became a professor and director of of the Obversatory at Göttingen. His work was of basic importance in algebra, number theory, differential equitations, differential geometry, non-Euclidean geometry, complex analysis, numerical analysis, astronomy, geodesy, electromagnetism and theoratical mechanics. He also paved the way for a general ans systematic use of complex numbers.

(This text from E. Kreysig, Advanced Engineering Mathematics)

1.11 PolyFit History

```
   Linear regression program history

12-jul-93:  Started programming.

            Created GUI with complete numerical input routines.

13-jul-93:  Added string-gadgets to rename X- and Y-axis names.

            Numerical string-gadgets would not get refreshed after they were
            cleared. - Fixed.
            Added Menu structures.

16-jul-93:  Implemented Project-menu features:
            -New
            -Open
            -Save
            -Save As
            -About
            -Quit

19-jul-93:  Programmed output-window with IntuiText-structures.
            Added print facility.

            Implemented Linear Fit and Linear Fit through origin.

20-jul-93:  Implemented Exponentional Fit and Polynome Fit.

21-jul-93:  Included a busy-pointer for use when computing.

            Bug Found: The maximum order of the polynome was 16, but the
            array that contained the coefficients was only 11 numbers wide.
            - Fixed.

            Started programming the graph-draw-routine. The routine now
            includes: draw circles around datapoints, draw lines around
            graph area, draw calculated line in graph, display x-axis and
            y-axis names (display y-axis name tilted over).

22-jul-93:  Continued graph-programming. Added scale divisions + numbers.

            Bug found: I discovered that the library-function pow() does
            not correctly calculate 0^0 (= 1), but instead returns 0. This
            affected the value of a polynome when x = 0 and would cause a
            strange 'jump' of the graph-line. - Fixed.

23-jul-93:  Started & completed programming the Save as IFF routine.

25-jul-93:  Implemented 'Save as IFF' as a menu option.
            Bug found. In the drawgraph-routine memory was reserved for
            storing the y-values. This memory was never returned to the
            system. - Fixed.

            Added a gadget for giving the graph a title. Changed load &
            save routines to store and use this title.
            Changed the input-routine. If auto-add is on and a x,y set in
            the listview-gadget is selected, pressing return in the y-gadget
            will result in 'auto-replace'. This prevents moving your hands
            between mouse and keyboard a few times.

26-jul-93:  Added choice of a linear or logarithmic scale for both the
            x-axis and the y-axis. Adapted the scale-divisions-draw-
            routine to process a logarithmic scale properly.

            Added keyboard shortcuts for gadgets.

27-jul-93:  If a save function is performed, an icon can be saved as well.
            A pre-load file can be selected with Default Tools or with
            CLI input.

28-jul-93:  Revisioned the way the colours of the menu and the menutext are
            determined. Now uses the DrawInfo_Pens. I could not test if this
            works on a 2.0 system, but it does on 3.0.

            Adapted text to use proper English words. (f.e. I used 'grade'
            instead of 'degree', because in Dutch you would say 'graad').
            This is the official v1.0 release of PolyFit.

03-aug-93:  Changed all variables of type float to double. This is because
            when calculating a 16th degree curve of a data set, the 16th
            power of every x-value is calculated. If this amounts to say
            1000, the 16th power is equal to 10^48, which does not fit in
            the float's limits. This would result in nonsense results and
            this led in some cases to a crash.

            Added 'Print' and 'Print as' to the Project menu.
            Bug fixed: the message reqester would not adjust its size to the
            size of the screen font. - Fixed.
            New version: v1.1

04-aug-93:  I borrowed a printer and added 'Print graph' to the Graph menu.
            It lets the printer.device do the job so it makes use of the
            preferences settings. I also changed the graph colours from
            blue/light blue to light blue/black to get better print outs.

05-aug-93:  Changed the file format to an IFF-like type. It cost me a lot of
            work to change it and the only advantage of this format is that
            I now can check if the file to be loaded is indeed a PolyFit-
            datafile. But I believe Commodore likes to see it this way...

            Minor bug found: Input from the graph-window would be blocked
            when it is in background with a zero-requester. The memory for this
            requester was not freed again before closing the window. - Fixed.

            The graph screen will be displayed in NTSC mode (of course only if
            the user's computer supports NTSC). The graph then fills the
            complete screen and also reduces the flicker of the interlace mode.

06-aug-93:  Disbanded keeping the graph-screen on the background. When the
            user presses a mouse button, the screen will be closed. This gives
            the screen memory free for other tasks and also prevents the screen
            from receiving input while it should not.

            Changed my own list-handling routines to use the exec-lists
            functions. This makes the code hardly smaller, but my source a lot
            more comprehensive.

14-aug-93:  I forgot to turn CTRL-C handling off. Now it's off.
            I changed the default mode for the graph to be drawn in from NTSC
            to the Default Monitor. This is done to prevent flicker when the
            program switches between public and graph screen. The size of the
            graph is dependent of the mode, so that the graph screen is always
            filled to its maximum.
            Provided a slider-gadget for the user to set the polynome degree
            instead of the integer-gadget I used before.
```

23-nov-93: I replaced my fairly inefficient polynome evaluation algorithm by "Horner's alogorithm". This results in much quicker calculations and less errors.
I also made a minor change in the way the polynome fit is calculated. This gives better results but as well a very small calculation time increase.
Optimized some calculation functions by avoiding the use of the pow() function and replacing it by a simple loop.
I rewrote the "save as iff" function, so that it is more generally usable (that is only easy for ME) and it now compresses the data, which has as effect that the picture files will be about five to six (!) times smaller.

27-dec-93: Totally recoded the graph part. User can now select if he/she wants to have a grid, wants to have a curve, where he/she wants the title and in what screenmode the graph will be drawn.

02-jan-94: The user now can also choose to have the graph drawn in black and white. This probably will give better results on most printers.
I added the option to draw a residu plot of the data. In such a plot you can see how much the points deviate from the fit curve.

04-feb-94: Inplemented localisation. This works with all systems that can use the locale.library (V38 and up). By default English will be the language that PolyFit uses. I also supplied a catalog for Dutch support.