

FIT for HP 48 G/GX
Version 1.0
(C) 1994-2001 Gian Piero Luciani

All file of program " FIT " are property of Gian Piero Luciani.
This software are free and no guarantee of some type is supplied, the author does not assume responsibilities derived from the use of the program, damages derived from the use of this software are to cargo of the user.

This is the english documentation, it comes brought back with to the italian documentation because my English is rather poor and so this documentation could be insufficient and/or compresibile and full of errors. Very thanks to anyone will want help me to translate in correct way this document.

Who found the program useful or interesting or simply wants, can send me one postcard with a beautiful stamp!

What makes program FIT.

This program is born from the requirement to calculate, on a series of data collected in scientific experiments (chemical, kinetic, etc.), the best least squared interpolation with functions that of solid are not easy calculable with this system.

The program contains the algoritimi in order to calculate the least squared of the following functions:

1) n. Degree polinomial.: $Y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$

2) Gaussian: $Y = a \cdot e^{-b(x-x_0)^2}$

3) Bi exponential: $Y = a \cdot e^{-bx} + c \cdot e^{-dx}$

4) Van Deemter: $Y = a + \frac{b}{x} + cx$

5) Morse: $Y = a(1 - e^{-b(x-c)})^2$

6) function: $Y = a(1 - e^{-bx})$

7) function: $Y = a - b \cdot e^{-cx}$

8) function: $Y = a \cdot x^n \cdot e^{-bx}$

Installation.

The program is written almost totally in USER RPL introduces, there is a library, installabile and working on whichever port (from 0 to 32), or a directory like the personal tastes.

File: fit.lib
Library number: 1700
Bytes: 7619
CRC: # 13098d
File: fit.dir
Bytes: 7593.5

CRC:

9785d

The program can be loaded in one of the usual ways (sees User's Reference Manual chapters 27 and 28) and works correctly in whichever port.

As it is used:

In order to use the program enough to give command FIT.

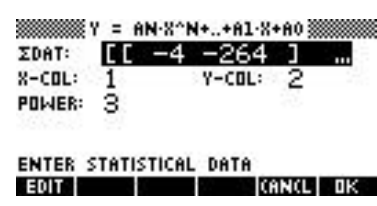
It comes shown the menu of the various functions from which one can be chosen the type of curve with the arrow and the key OK.



At this point it comes introduced a second window in which are introduced the data necessary to complete the calculation. The various options are described bellow:

N. Degreee polinomial:

Program for least square polinomiale interpolation, the program executes the interpolation with any degree (but power must be smaller to the number of the points) using the system of the least square interpolation.



The necessary information are: the matrix of the data of the points from interpolator that are constituted from the usual statistical variable sigmadat, the indication of the column that contains independent variable x, the indication of the column that contains the dependent variable y, the power of the polinomial to calculate.

Introduced the data and pressa OK the program executes the calculation and transfers the result, under shape of algebric expression, in the stack. It is enough easy to see the result of the interpolation with program PLOT of HP, the first what to do is enter in the program with RS 8 and select the type of plot to SCATTER, cancels previous pict and plot points adopting the autoscale. Then you must return to menu PLOT and select the type of diagram FUNCTION, takes the function calculated from the stack and plot over the points as soon as visualized.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
N: degree of the polinomio

Algorithm:

The coefficients of the polinomio are obtained resolving the following equation:

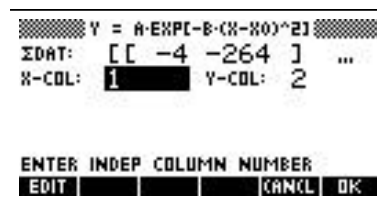
$$\begin{pmatrix} N & \sum_{i=1}^n x_i & \dots & \sum_{i=1}^n x_i^n \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^n x_i^n & \dots & \dots & \sum_{i=1}^n x_i^{2n} \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i x_i \\ \dots \\ \sum_{i=1}^n y_i x_i^n \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix}$$

Example:

Xi	Yi	N=3 (y=x^3-7x+6)		
0	6		exact	found
1	0	a0	6	6
2	0	a1	-7	-7.00000000001
3	12	a2	0	3.75E-12
4	42	a3	1	1
5	96			

Gaussian:

Program for interpolation of the Gaussian function, starting from a series of points the program is able to calculate the best Gaussian.



The necessary information are: the matrix of the data of the points from interpolator, the indication of the column that contains independent variable x, the indication of the column that contains dependent variable y.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable

Algorithm:

The equation that it wants interpolator is the Gaussian one that comes expressed from the equation:

$$y = A \times e^{-B \cdot (x-x_0)^2}$$

This equation can be transformed in a polinomio of 2° degree, that it can be resolved with the same algorithm used in the previous case.

The first operation to do is to assume

$$\ln(y) = \ln(A) - B \cdot (x - x_0)^2$$

$$\ln(y) = \ln(A) - Bx^2 + 2BX_0 - BX_0^2$$

$$\ln(y) = (-B)x^2 + (2BX_0)X + (\ln(A) - BX_0^2)$$

therefore the polinomio is estimated of interpolation of second degree from which the coefficients are obtained:

$$\begin{aligned} a_0 &= \ln(A) - BX_0^2 & A &= e^{(a_0 - \frac{a_1^2}{4a_2})} \\ a_1 &= 2BX_0 & B &= -a_2 \\ a_2 &= -B & X_0 &= -\frac{a_1}{2a_2} \end{aligned}$$

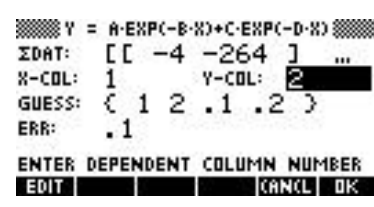
Example:

Xi	Yi	y = 3*exp(-0.2*(X-2.3)^2)		
0	1.04145			
1	2.13958			
2	2.94648	A	exact	found
3	2.71995	B	0.2	3.00000051775
4	1.68306	X0	2.3	0.200000299828
5	0.69810			2.29999886033

Bi exponential

Program of least square bi exponential interpolation, the program executes the interpolation with the biesponenziale function, that is

$$y = A \cdot \exp(-B \cdot x) + C \cdot \exp(-D \cdot x)$$



In this case beyond to the information demanded in the previous programs list of four values is necessary to introduce one (guess) and the error. In order to resolve this problem (that it is most complex of the entire group and is that one for which they have been written these programs) is necessary to resort to a iterative method of Newton Raphson who in order to work well demands an approximate value of the parameters that must try; much better they are these values the much best one and faster it is the calculation.

Initially it is well as an example to place Err to a enough great value (=0.1) then riduce Err in second time. In fact repeating the calculation as guess the values of the result of the previous calculation are taken.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable

Y: index of the column for dependent variable
X0: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm

uses the algorithm of the least squared interpolation, but the function in object cannot be led back to a polinomio, then is necessary a iterative method (Newton-Raphson) that it takes advantage of the following algorithm: It is the function from interpolat:

$$Y = a \cdot e^{-bx} + c \cdot e^{-dx}$$

Then the function chi-square is equal (almost equal, some constants that do not influence the calculation are omitted) to:

$$\chi^2 \propto (y_i - a \cdot e^{-bx_i} + c \cdot e^{-dx_i})^2$$

If the best values of the parameters (a, b, c, d) are wanted to be found it must diminish this function, that is find the values of (a, b, c, d) that they render the derivative of chi-square null. In order to make this the iterative algorithm of Newton-Raphson is used who bringing back of under:

$$\begin{bmatrix} a_n \\ b_n \\ c_n \\ d_n \end{bmatrix} = \begin{bmatrix} a_{n-1} \\ b_{n-1} \\ c_{n-1} \\ d_{n-1} \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 \chi^2}{\partial a \partial a} & \frac{\partial^2 \chi^2}{\partial a \partial b} & \frac{\partial^2 \chi^2}{\partial a \partial c} & \frac{\partial^2 \chi^2}{\partial a \partial d} \\ \frac{\partial^2 \chi^2}{\partial b \partial a} & \frac{\partial^2 \chi^2}{\partial b \partial b} & \frac{\partial^2 \chi^2}{\partial b \partial c} & \frac{\partial^2 \chi^2}{\partial b \partial d} \\ \frac{\partial^2 \chi^2}{\partial c \partial a} & \frac{\partial^2 \chi^2}{\partial c \partial b} & \frac{\partial^2 \chi^2}{\partial c \partial c} & \frac{\partial^2 \chi^2}{\partial c \partial d} \\ \frac{\partial^2 \chi^2}{\partial d \partial a} & \frac{\partial^2 \chi^2}{\partial d \partial b} & \frac{\partial^2 \chi^2}{\partial d \partial c} & \frac{\partial^2 \chi^2}{\partial d \partial d} \end{bmatrix}^{-1} \times \begin{bmatrix} \frac{\partial \chi}{\partial a} \\ \frac{\partial \chi}{\partial b} \\ \frac{\partial \chi}{\partial c} \\ \frac{\partial \chi}{\partial d} \end{bmatrix}$$

In truth this algorithm is rather slow for which the calculation it comes executed in two phases, in before a reduced and approximated shape is executed and then applies the true algorithm and just. If all it goes well, to every iterance the values of (a, b, c, d) are nearer the searched values and on the display it comes shown a more and more small value of X2.

When X2 is sufficiently small the program executes the complete algorithm (comes shown Y2) until to the attainment of the intentional error.

The error is calculated on the base of the difference of the sum of the single parameters in two successive iterances.

To the onlookers council to watch the program, I can only add that it works.

Seen the complexity of the calculations, it is not said that an acceptable result is obtained always, in these cases must be tried to change the values of guess of departure.

Moreover the calculations demand a time for which and not having too much haste well.

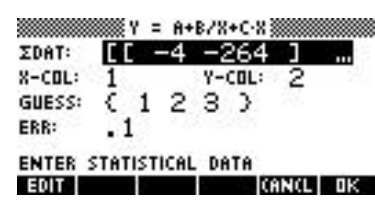
Example:

Xi	Yi	y=10*exp(-0.5*x)+5*exp(-0.05*x)		
0	15			
1	10.82			
2	8.2	a	9	Er = 0.1
3	6.53	b	1	

4	5.45	c	3		
5	4.71	d	0.1		
6	4.2				
7	3.82			exact	found
8	3.53	a	10		9.999329
9	3.3	b	0.5		0.500251
10	3.1	c	5		5.000664
11	2.92	d	0.05		0.050101

Van Deemter

Program of least square interpolation of the function of Van Deemter



All the next programs use the same algorithms seen for the program of bi-exponential interpolation and need of the same information of base. In this case the guess and composed from one list of 3 parameters. For the onlookers it can be said that the curve of Van Deemter is used in chromatography (chemistry) in order to put in relation the flow of the eluent with the efficiency of the column. In practical the constructors of the columns they declare which is the optimal flow in base to the used eluent and since this is much similar for similar columns this program has one usefulness reduced enough.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
Xa: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm

Uses the same algorithm explained before, in the shape original without approximations. In this case to every cycle it comes shown Er that is the sum of the differences of the parameters tried between one iteration and the other. As it will be looked at making of the tests, this function is the much less sensitive one to the choice of the initial values.

Example:

Xi	Yi	y=A+B/X+CX			
1	29				
2	26				
3	25.7				
4	26				
			guess		
		A	10		Er = 0.1
		B	10		

5	26.6	C	10		
6	27.3				
7	28.1			esatto	trovato
8	29	A	20		20.003197
9	29.9	B	8		8.002840
10	30.8	C	1		0.998690

Morse:

Program of least square interpolation with the function of Morse

```

Y = A*(1-EXP(-B*(X-C)))^2
SDAT:  [[ -4 -264 ] ...
X-COL: 1      Y-COL: 2
GUESS: ( 0 0 0 )
ERR:    .1
ENTER MAXIMUM ACCEPTABLE ERROR
EDIT  CANCEL OK

```

The program uses the same algorithms seen for the program of bi-exponential and need of the same information of base, in this case however is a function of forecast of the value of guess that it enters in function when $\text{guess} = \{ 0 \ 0 \ 0 \}$.

The Morse function describes the curve of potential of a chemical bond in function of the distance between the atoms that compose the bond.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
Xa: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm:

Uses the same algorithm explained for the bi-exponential least square interpolation in the shape original without approximations, when $\text{guess} = \{ 0 \ 0 \ 0 \}$ according to algorithm it tries to indovinare the best parameters to use in the iterativa routine. In this case to every cycle it comes shown Er that is the sum of the differences of the parameters tried between one iterance and the other.

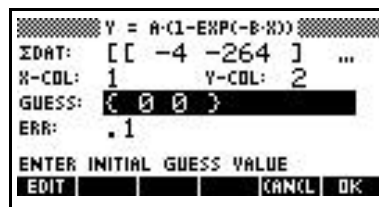
Example:

X_i	Y_i	$y=A(1-\exp(-B(X-C)))^2$			
0	124.77				
1	32.30		guess		
2	2.62	A	0		
3	1.94	B	0		Er = 0.01
4	13.13	C	0		
5	27.84				

6	42.26		exact	found
7	54.87	A	100	119.16
8	65.28	B	0.3	0.255
9	73.57	C	2.5	2.735
10	80.03			

$Y=A(1-\exp(-BX))$

This program executes the interpolation with the function: $y=A(1-\exp(-BX))$.



The program uses the same algorithms seen for the program of bi-exponential interpolation and needs of the same information of base; as in the case of the Morse function a function of forecast of the value of guess is present that it enters in function when $\text{guess} = \{ 0 \ 0 \}$

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
Xb: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm:

Uses the same algorithm used for the bi-exponential interpolation in the shape original without approximations, when $\text{guess} = \{ 0 \ 0 \}$ according to algorithm it tries to find for semiempiricist the best parameters to use in the iterativa routine. In this case to every cycle it comes shown Er that is the sum of the differences of the parameters tried between one iterance and the other.

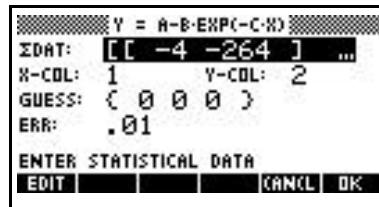
Example:

X_i	Y_i	$y=A(1-\exp(-BX))$		
0	0.00			
1	25.92		guess	
2	45.12	A	0	
3	59.34	B	0	Er = 0.01
4	69.88			
5	77.69			
6	83.47		exact	found
7	87.75	A	100	99.99917
8	90.93	B	0.3	0.300005

9	93.28
10	95.02

Y=A-Bexp(-CX)

This program executes the interpolation with the function: $y=A-B\exp(-CX)$



The program uses the same algorithms seen for the program of bi-exponential interpolation and needs of the same information of base; as in the case of the Morse function a function of forecast of the value of guess is present that it enters in function when $\text{guess} = \{ 0 \ 0 \ 0 \}$.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
Xa: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm

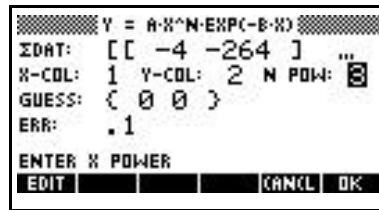
Uses the same algorithm used for the bi-exponential interpolation in the shape original without approximations, when $\text{guess} = \{ 0 \ 0 \ 0 \}$ according to algorithm it tries to find for via semiempiricist the best parameters to use in the iterative routine. In this case to every cycle it comes shown Er that is the sum of the differences of the parameters tried between one iteration and the other.

Example:

X_i	Y_i	$y=A-B\exp(-CX)$		
0	5.00			
1	5.91		guess	
2	6.65	A	0	Er = 0.01
3	7.26	B	0	
4	7.75	C	0	
5	8.16			
6	8.49		exact	fount
7	8.77	A	10	9.993044
8	8.99	B	5	4.991586
9	9.17	C	0.2	0.200418
10	9.32			

$$Y=A \cdot X^n \cdot \exp(-BX)$$

This program executes the interpolation with the function:
 $Y=A \cdot X^n \cdot \exp(-BX)$.



The program uses the same algorithms seen for the program of bi-exponential interpolation and needs of the same information of base; as in the case of the Morse function a function of forecast of the value of guess is present that it enters in function when guess = { 0 0 }.
The program need also the n power.

Variable used:

MSG: list of the messages of the graphical interface
SigmaDAT: statistical variable of HP 48
X: index of the column for independent variable
Y: index of the column for dependent variable
Xa: list of the initial values of the coefficients of the tried equation
Er: error; it is the sum of the differences between the coefficients of the equation tried calculated in two consecutive iteration.

Algorithm

Uses the same algorithm used for the bi-exponential interpolation in the shape original without approximations, when guess = { 0 0 } according to algorithm it tries to find for via semiempiricist the best parameters to use in the iterativa routine. In this case to every cycle it comes shown Er that is the sum of the differences of the parameters tried between one iterance and the other.

Example:

Xi	Yi	$y=A \cdot X^n \cdot \exp(-BX)$		
0	0.00			
1	9.51		guess	
2	72.39	A	15	
3	232.39	B	0.01	Er = 0.01
4	523.99	n	3	
5	973.50			
6	1600.17		esatto	trovato
7	2417.08	A	10	10.000124
8	3432.04	B	0.2	5.000114
9	4648.31			
10	6065.31			