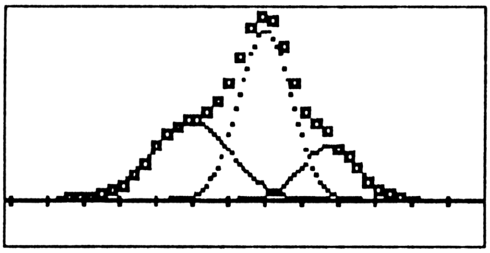
The Regression Analysis Package

for the HP 48



The Regression Analysis Package for the HP 48

First Edition

Acknowledgements

My thanks to Bill Stabnow for providing useful input on the various aspects relating to the look and feel of the software and users manual.

CONTENTS

Getting Started

The centerpiece of the Regression Analysis Package is the CFIT library which provides the HP 48 community with a powerful curve fitting capability never before seen in a handheld calculator. Two routines perform both multiple linear regressions and nonlinear regressions with up to 16 fit parameters. The package also contains a menu driven constants library as an alternative to the one found in the HP Equation Library Application Card. The constants library contains 21 constants from the physical sciences, 60 astronomical constants, and 43 mathematical constants. Finally, a Time Value of Money program is included which is modeled after the routine found in the PPC ROM for the HP 41. The version here includes many enhancements over the TVM found in the HP Equation Library Application Card, including the ability to generate amortization tables and summaries quickly and with great precision. The program allows for full and independent control over the payment and compounding frequencies, including continuous compounding, and handles ordinary annuities and annuities due.

NAME	TYPE	LIB ID	SIZE (kb)	LIBRARY COMMANDS
CFIT	Library	913	11.1	NLR, MLR
CONSTANT	Library	914	8.4	CONSTANT, PHYSC, EARTHC, SOLARC, GALAXYC, COSMC, MATHC
TVOM	Program	--	5.9	--

Installing the Software

The software in this package consists of two libraries and a program. The program can be renamed in the event that a name conflict exists. The libraries have a number of library commands. A library command must not exist as a named object anywhere else in the current path. Notice that CONSTANT is a library command as well as the name of the library. The function of the library command, CONSTANT, is to create and display a menu of the other commands in the library. The advantage in doing this is that CONSTANT can be evaluated, making the constant library available from the CST menu, or even from the command line.

Refer to the HP 48 Owners Manual for information on installing libraries. In brief, the procedure is as follows.

- Download the library to the HP 48.
- Recall the library to the stack.
- Store the library object in a port. For example, 0 STO.
- Purge the variable that the library was stored in.
- Turn the calculator off, then on again. The calculator performs a system halt due to the existence of the new library. Both CFIT and CONSTANT automatically attach themselves to the HOME directory.

To install TVOM, simply download the program. As a program object, it can be stored in any variable and evaluated like any other program.

Removing the Software

The procedure for removing the libraries is as follows.

- Make sure that the library object does not exist on the stack.
- Go to the HOME directory.
- Enter the library number, tagged by its port number, for example :0:913, and execute DETACH.
- Enter the library name as above and execute PURGE.

Remove the program object TVOM as with any other named variable.

CFIT
Curve FIT

Introduction

A commonly encountered problem in science, medicine, and social studies is - given a collection of measured data y_i, associated with some quantity x_i, find a function, or adjust the parameters of a function such that the quantity $\chi^2=[y_i-F(x_i)]^2$ is minimized. This minimization condition is termed the method of maximum likelihood, or the method of least squares. Of course, the method assumes that the function, F, is the correct function for the problem to be solved. Also implicit in the method is the assumption that the observed measurements, y_i, about the actual value, $F(x_i)$, are normally distributed, with zero mean. In most typical problems of practical concern, these assumptions hold true.

The general problem of fitting a function to experimental data can be classified as linear or nonlinear. The nonlinear case is characterized by the function containing terms nonlinear in the fitting parameters. In other words, at least one of the partial derivatives of F with respect to the fit parameters, depends on at least one of the fit parameters. This distinction is a critical one. The linear problem reduces to the problem of simultaneous equations which can be solved analytically. The nonlinear problem introduces considerable complexity. An analytic solution, in general, is not possible. Solutions are obtained by providing a "first guess" and iterating to a solution.

Linear methods are fairly easy to understand and implement. Straight line and polynomial fit programs abound, many in the public domain. Nonlinear methods are a different story. One generally goes, physically or electronically, to a library of scientific or mathematical routines to locate the proper tools. These packages usually run on a fairly substantial machine, usually one that will not fit conveniently on your desk.

The two curve fit programs included in this package are the most powerful curve fit programs available for a handheld calculator. NLR

(NonLinear Regression) is a general purpose program used to fit any type of single variable function to a data set, from a straight line to a function totally nonlinear in the fitting parameters. MLR (Multiple Linear Regression) is restricted to functions that are linear in the fitting parameters. MLR will fit a simple polynomial, an orthogonal polynomial set, or any linear single variable function to a data set. Both NLR and MLR generate fits which correctly account for Poisson, or instrumental (user defined) weighted data. Both NLR and MLR can solve for up to 16 fitting parameters.

Measurement Uncertainties (Weighting of Data)

The role that measurement uncertainties play in determining the fit function is not always appreciated. Data weighting is in essence the reality that the numerical values of measured data have some intrinsic uncertainty associated with them. The precise nature of the uncertainty depends on the specifics of the experiment. The data from one experiment may have the same random uncertainties for each measurement. The data from another may involve numerous electronic scale changes in the instrumentation, each scale characterized by a particular noise or random error. Still a third experiment may involve counting of events, say photons or decays. Counting experiments are characterized by Poisson noise. The relative uncertainties or "error bars" associated with the data set can frequently produce fits that differ from the unweighted case. In order for a fit, or regression analysis, to have meaning it must be done correctly, taking into account measurement uncertainties.

Even when measured data have equal uncertainties, care is needed to avoid the pitfalls encountered when transformations are used. The simple function y=A1*EXP(-x/A2) is often fitted to exponential decay data by transforming it to a linear form by taking the logarithm of both sides of the equation, yielding LOG(y)=-x/A2+LOG(A1). We take the logarithm of all of our y values and proceed with a simple linear straight line fit. The answer obtained is useful but not correct. It is not correct because when we transform a measured data set, we must also transform the uncertainties. The net effect of neglecting the transformed uncertainties in the example cited was to weight the data points in the exponential tail too heavily during subsequent curve fit. In many cases where weighting or data transformation is involved, the answer obtained by a simple (or simple minded) curve fit which ignores measurement uncertainties is no better than the answer obtained by graphical estimation using a pencil and ruler.

NonLinear Regression (using Marquardt's algorithm)

NLR will fit data to a function containing up to 16 fitting parameters, for the general case where the fitting parameters are nonlinear, eg. the Gaussian, $F(X) = A_1 \exp[-\frac{1}{2}((X-A_2)/A_3)^2]$. This type of fit is usually performed using a computing engine of considerable power, and the algorithms used are frequently sophisticated enough to account for somewhat pathological functions or very bad first guesses. The algorithm implemented here is not as robust and "bullet proof" as some of the routines running on large computer systems, but will do an excellent job of fitting data using well behaved functions of the type encountered in the real world. The algorithm chosen is of necessity a compromise, but perhaps most importantly, the method has the advantage of converging to a solution rapidly, which is of paramount importance for a machine of limited computing capabilities. NLR has the capability of providing for three types of data weighting; no (equal) weighting, Poisson weighting, or instrumental weighting.

Background

Most nonlinear regression algorithms are usually variants of one of two methods. The first method is one of linearization of the function, or a Gauss-Newton approach. The method begins, conceptually, with the first order Taylor expansion of the function in the fitting parameters, A_i, resulting in a function which is linear in the A's. The method of linear least squares can then be applied to arrive at a "solution" for the A's. Repeated linearization of the function for each new set of A's results in convergence to a final solution. This method has the advantage that it converges fairly rapidly to the minimum in χ^2 from points nearby. However,it does not do such a great job for points outside the region where χ^2 is nearly parabolic. In fact it can totally fail!

The second method, the method of steepest descent, is one which relies on calculation of the gradient $\nabla\chi^2$ to determine the magnitude and direction by which the A's are incremented. The method of steepest descent works very well for values of the A's far from the minimum in χ^2, since it does not assume or require that χ^2 be nearly parabolic. A disadvantage is that the method becomes computationally inefficient as the minimum is approached. The method will usually converge, though, and is generally more robust than the Gauss-Newton method.

Marquardt's Method

The method of Marquardt, also called the gradient expansion algorithm, relies on a variable conditioning factor, λ, which inflates the diagonal of the curvature matrix multiplicatively when χ^2 fails to decrease. This will become clear shortly.

First some definitions:

i	index over the number of data points.
j	index over the fitting parameters.
A_j	the j^{th} fitting parameter.
X_i	the i^{th} independent data point.
y_i	the i^{th} dependent data point.
$\mathbf{A}$	the vector of A's, $[A_1, \ldots A_n]$.
$F(X_i,\mathbf{A})$	the value of the fit function at X_i, $\mathbf{A}$.
$\mathbf{I}$	the identity matrix.
λ	conditioning factor.
σ_i	Standard deviation or error bar (weight) of measurement data.
$\mathbf{A*B}$	term-by-term product of $\mathbf{A}$ and $\mathbf{B}$. $(\mathbf{A*B})_{jk}=a_{jk}*b_{jk}$.

Using these definitions, we want to find $\mathbf{A}$ such that chi-square is minimized:

$$\chi^2=\sum_i(\sigma_i^{-2}[y_i-F(X_i,\mathbf{A})]^2).$$

We will need to calculate the following quantities:

$\mathbf{K}_{1i}=(y_i-F(X_i,\mathbf{A}))$	;a 1-D matrix which is obtained when χ^2 is calculated.
$\mathbf{D}_{ij}=\partial F_i/\partial A_j$	;a 2-D matrix.

The quantity $\mathbf{DD}^T$ (where $\mathbf{D}^T$ is the transpose of **D**) is called the curvature matrix. The curvature matrix yields the local shape of the χ^2 hypersurface, and contains information on the behavior of parameter space in the neighborhood of the current estimate for **A**. The curvature matrix provides information on the direction of the correction to **A** ,ie the direction of the $\delta\mathbf{A}$ vector. **K** is a measure of the distance from the evaluated function to the real data, using the current **A**. **K** therefore provides a measure of the required magnitude of the correction to **A**.

Without providing any derivation, the resulting algorithm is very simple:

$$\begin{tikzcd}
& \begin{array}{l}\text{Set tolerance: }\%\text{tol}=1\\ \text{Set conditioning factor: }\lambda=0.01\end{array} \arrow[d] & \\
& \text{Compute }\chi^2 \arrow[d] & \\
& \text{Compute }\mathbf{D} \arrow[d] & \\
& \begin{array}{l}\delta\mathbf{A}=(\mathbf{DD}^T+\lambda\mathbf{I}*\mathbf{DD}^T)^{-1}\mathbf{DK}\\ \mathbf{A}=\mathbf{A}+\delta\mathbf{A}\end{array} \arrow[d] & \\
\begin{array}{l}\lambda=10\lambda\\ \mathbf{A}=\mathbf{A}-\delta\mathbf{A}\end{array} \arrow[ur] & |\chi^2_{new}-\chi^2_{old}|/\chi^2<\%\text{tol}/100? \arrow[r, "\text{yes}"'] \arrow[d, "\text{no}"] & \text{END} \\
& \chi^2_{new}>\chi^2_{old}\ ? \arrow[ul, "\text{yes}"] \arrow[d, "\text{no}"] & \\
& \lambda=\lambda/5 \arrow[uuuu, dashed, bend right=60] &
\end{tikzcd}$$

On the first pass, λ inflates the diagonal of the curvature matrix by 1%. An uninflated diagonal is equivalent to a Gauss-Newton approach, which converges very quickly, if the χ^2 minimum is nearby. If the first iteration does result in a reduced χ^2, λ is decreased, and the algorithm continues to converge to a solution with each successive pass. If the new χ^2 is greater than the old, the step (or $\delta\mathbf{A}$) was in the wrong direction, meaning that the current point in χ^2 hyperspace is too far from the minimum for linearization of the function to be a valid approach. The conditioning factor is increased and the curvature matrix is modified to get a new step direction. Notice that in this case, the derivatives associated with **D** need not be recalculated, rather the diagonal is simply inflated. This loop executes rapidly. As the diagonal terms come to dominate, the curvature matrix begins to look like that obtained when gradients are used, since cross terms become negligible with respect to the diagonal terms. In other words, the method resorts to a steepest descent approach. In reality, Marquardt's method is more than an either/or method. It can be shown that the best step direction is in fact somewhere between the directions given by Gauss-Newton and steepest descent. Marquardt always chooses a direction between the two.

Data Weighting

NLR permits 3 types of data weighting;

Equal weighting (no weighting).
The standard deviations of the measured data are set internally to 1, (σ=1).

Poisson weighting.
Poisson weighting is characteristic of counting experiments. The standard deviations of the measured data are equal to the square root of the measurement ($\sigma_i=\sqrt{y_i}$).

Instrumental weighting.
Instrumental weighting is user specified for each data point. This weighting is appropriate when the random errors for the dependent data vary from measurement to measurement in a way determined by the instrumentation used.

Instructions

First create the function F(X,A1,A2,..An), in a manner that assumes that the variables X,A1,A2,...,An exist (they will be created at run time). The variables are all upper case. Store the function (a program or an algebraic) in any variable name. Create, or port in, the data array and store it under any variable name. The data array is a 2 or 3 column matrix (ordered as X,Y or X,Y,σ) with the number of rows equal to the number of data points. If the third column is present, it represents the error bars or standard deviations of the measured data. Create a first guess vector or 1-D matrix and store it under any variable name. The vector is ordered A1 through An. Generally, it will be necessary to inspect a rough plot of the data, or at least do a few trial calculations to gain some information as to the approximate values of the fit parameters.

Run NLR. A sign-on message appears as well as a menu with choices of CONTINUE, HELP, and QUIT. The help menu choice summarizes the paragraph above and is designed to remind the experienced user of the required procedures. The continue choice bypasses the help screens and begins the body of the program. Once the program begins, only 3 bits of information are required. The first prompt asks for the data variable name. The second prompt asks for the fit function name and the third prompt asks for the name of the first guess vector. If a third column exists in the data array, MLR knows that instrumental weighting is required. If a third column does not exist, NLR asks for the user to specify EQUAL or POISSON weighting using a highlight bar. The up and down arrow keys are active until a choice is made using the ENTER key. Before execution, the user is offered a menu choice as to whether or not a solution plot is desired.

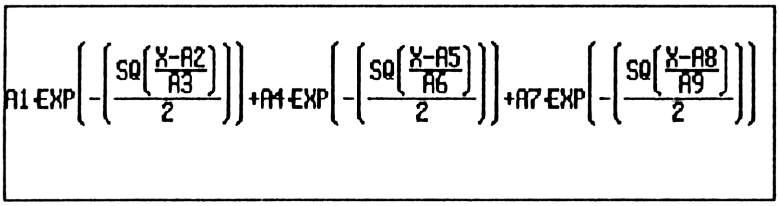
Upon completion, the program beeps and the final values of χ^2 and the resultant fit parameters are placed on the stack, all tagged for easy identification. NLR also creates two global variables.

$COEF -	The solution fit coefficient matrix, the first column containing the A's, and the second column containing the P.E., or probable error in the corresponding coefficient. (P.E. = 0.6745σ). The statistical probability of the "actual" coefficient falling within the range $A_i \pm PE_i$ is ½.
CHISQR -	The solution χ^2 for the fit.

Example 1

We will fit a function of 3 Gaussians to a data set composed of 32 points. Each Gaussian has 3 unknown parameters, so there will be a total of 9 fit parameters. The function is shown below in both RPN and algebraic representation, as well as EquationWriter form.

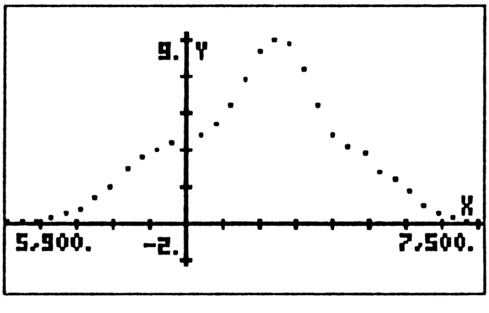
```
FX :    « A1 X A2 - A3 / SQ        'A1*EXP(-(SQ((X-A2)
        2 / NEG EXP * A4 X         /A3)/2))+A4*EXP(-(
        A5 - A6 / SQ 2 /           SQ((X-A5)/A6)/2))+
        NEG EXP * + A7 X A8        A7*EXP(-(SQ((X-A8)/
        - A9 / SQ 2 / NEG          A9)/2))'
        EXP * + »
```



$$A1\cdot EXP\left(-\left(\frac{SQ\left(\frac{X-A2}{A3}\right)}{2}\right)\right)+A4\cdot EXP\left(-\left(\frac{SQ\left(\frac{X-A5}{A6}\right)}{2}\right)\right)+A7\cdot EXP\left(-\left(\frac{SQ\left(\frac{X-A8}{A9}\right)}{2}\right)\right)$$

The data array, which can be entered using the matrix editor, or in real life might be ported in over the serial line from a PC, is shown below.

```
DAT1NLR :
        [   [ 5900 .05  ]
            [ 5950 .1   ]
            [ 6000 .12  ]
            [ 6050 .35  ]
            [ 6100 .5   ]
            [ 6150 .7   ]
            [ 6200 1.2  ]
            [ 6250 1.72]
            [ 6300 2.5  ]
            [ 6350 3.05]
            [ 6400 3.4  ]
            [ 6450 3.72]
            [ 6500 3.8  ]
            [ 6550 4.1  ]
            [ 6600 4.6  ]
            [ 6650 5.5  ]
```



```
[ 6700 6.68 ]
[ 6750 8.15 ]
[ 6800 8.68 ]
[ 6850 8.5  ]
[ 6900 7.2  ]
[ 6950 5.55 ]
[ 7000 4.15 ]
[ 7050 3.6  ]
[ 7100 3.22 ]
[ 7150 2.45 ]
[ 7200 1.95 ]
[ 7250 1.55 ]
[ 7300 .88  ]
[ 7350 .42  ]
[ 7400 .2   ]
[ 7450 .1   ]  ]
```

The data array might represent spectral data, where the first column would be wavelength and the second column, output voltage from a digital meter which could represent light intensity.

The first guess vector can be obtained by inspection of a crude data plot and is chosen to be:

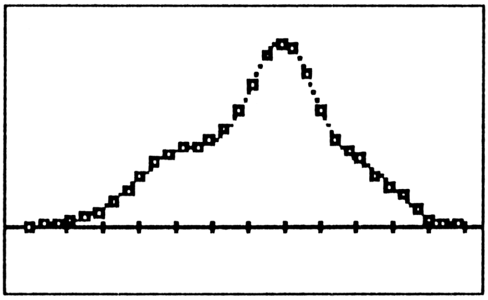
GUESS: [3 6400 120 8 6800 100 2.5 7100 100],

where the members of each triplet represent intensity, position of the peak, and peak width at half maximum, for each component Gaussian.

Now run NLR and enter the names DAT1NLR, FX, and GUESS at the prompts for data name, function name, and first guess vector. Select EQUAL when prompted for the weighting type. Finally, select YES from the "Plot Data" menu to begin execution.The program displays a status screen during computation, showing the current iteration, the current value of χ^2, and the current value of the conditioning factor, λ. This example requires about 7¾ minutes to converge to a solution after 4 iterations. Upon completion the program will beep and proceed to plot the resulting data and fit curve. The results for the problem here are displayed on the stack (CHISQR is also stored as a variable);

```
10.     CHISQR : 0.213
9.      A1 : 3.670
8.      A2 : 6469.774
7.      A3 : 183.129
6.      A4 : 7.994
5.      A5 : 6815.541
4.      A6 : 121.928
3.      A7 : 2.517
2.      A8 : 7112.859
1.      A9 : 127.634
```

The final plot produced by NLR showing both the data and the function is shown in the figure below.



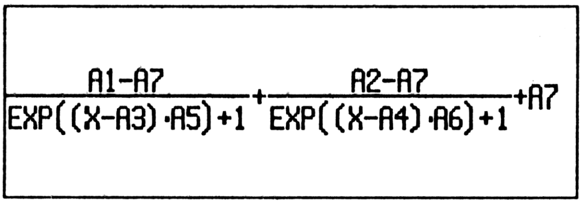
It is usually worth the effort to spend some time inspecting the data to develop a good first guess. The following first guess results in convergence to the same solution, but requires 7 iterations and 13 minutes.

GUESS2: [2 6400 250 8.5 6800 200 2 7150 100]

Example 2

In this example we will fit a function to a data set that represents the latitude dependence of some hypothetical parameter. Let's assume that a function is desired which is of the form shown below:

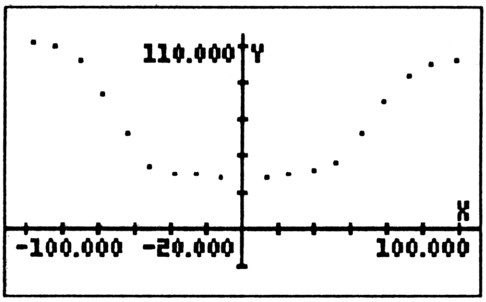
```
FX :    « A1 A7 - X A3 - A5          '(A1-A7)/(EXP((X-A3
        * EXP 1 + / A2 A7 -          )*A5)+1)+(A2-A7)/(
        X A4 - A6 * EXP 1 +          EXP((X-A4)*A6)+1)+
        / + A7 + »                   A7'
```



$$\frac{A1-A7}{EXP((X-A3)\cdot A5)+1}+\frac{A2-A7}{EXP((X-A4)\cdot A6)+1}+A7$$

The data array is:

```
DAT2NLR :
        [   [ -90   105 ]
            [ -80   104 ]
            [ -70   95  ]
            [ -60   77  ]
            [ -50   54  ]
            [ -40   35  ]
            [ -30   31  ]
            [ -20   32  ]
            [ -10   29  ]
            [ 0     30  ]
            [  10   30  ]
            [  20   31  ]
            [  30   33  ]
            [  40   37  ]
            [  50   54  ]
```



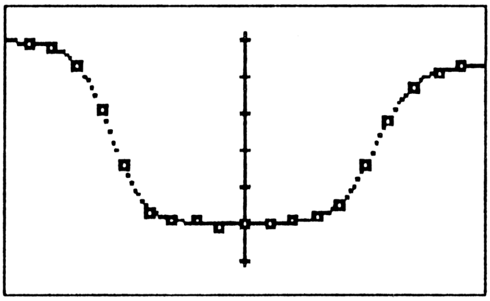
```
[ 60    73   ]
[ 70    87   ]
[ 80    93   ]
[ 90    95   ] ]
```

A first guess is made which is not very close to the solution, but shows what happens when the first guess yields a χ^2 neighborhood whose curvature results in a parameter step which is in the wrong direction. The guess is:

```
GUESS :  [[ 75 ]
          [ 75 ]
          [-75 ]
          [ 75 ]
          [ .5 ]
          [-.5 ]
          [ 50 ]]
```

Now run NLR, respond to the prompts (assume equal weighting), and observe iteration 1, followed by sub-iterations 1.1, 1.2, & 1.3. Finally, the algorithm has found the correct direction to the solution and continues with iterations 2, 3, 4, 5, & 6. At iteration 1.3 the value of the conditioning factor had increased to 10, meaning the diagonal terms were weighted 10 times more than the off diagonal terms. The algorithm at this point is nearly equivalent to a steepest descent. The execution time is 5 minutes. The plot produced by NLR is shown in the figure below, along with the solution placed on the stack.

```
8.      CHISQR : 17.117
7.      A1 : 105.93
6.      A2 : 95.54
5.      A3 : -56.26
4.      A4 : 54.72
3.      A5 : 0.137
2.      A6 : -0.127
1.      A7 : 29.67
```



Example 3

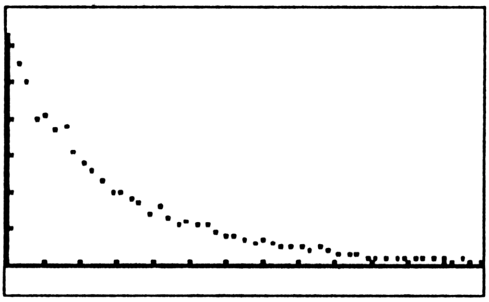
In this example the data set represents a decay process with Poisson statistics governing the measurement uncertainties. Lets assume that a major contaminant is producing a significant fraction of the observed decay events. The contaminant is thought to be about 20% of the target species and have a lifetime on the same order as the target species. We therefore wish to fit the data using a double exponential decay in order separate the decay constant of the target species from that of the contaminant. The function is shown in RPN, algebraic and EquationWriter form:

```
F(X):   « A1 X NEG A2 / EXP      'A1*EXP(-X/A2)+A3*EXP(-X/A4)'
        * A3 X NEG A4 / EXP
        * + »
```

$$\boxed{A1\cdot EXP\left(\frac{-X}{A2}\right)+A3\cdot EXP\left(\frac{-X}{A4}\right)}$$

The data array is:

```
DAT3NLR:  [ [ 0.1   608 ]
            [ 0.2   554 ]
            [ 0.3   441 ]
            [ 0.4   455 ]
            [ 0.5   408 ]
            [ 0.6   417 ]
            [ 0.7   342 ]
            [ 0.8   310 ]
            [ 0.9   293 ]
            [ 1.0   253 ]
            [ 1.1   221 ]
            [ 1.2   223 ]
            [ 1.3   195 ]
            [ 1.4   188 ]
            [ 1.5   154 ]
[ 1.6  176 ]
[ 1.7  150 ]
[ 1.8  124 ]
[ 1.9  137 ]
[ 2.0  123 ]
[ 2.1  123 ]
[ 2.2  99  ]
[ 2.3  89  ]
[ 2.4  87  ]
[ 2.5  74  ]
[ 2.6  67  ]
[ 2.7  74  ]
[ 2.8  66  ]
[ 2.9  52  ]
[ 3.0  52  ]
[ 3.1  53  ]
[ 3.2  45  ]
[ 3.3  53  ]
[ 3.4  42  ]
[ 3.5  34  ]
[ 3.6  28  ]
[ 3.7  31  ]
[ 3.8  27  ]
[ 3.9  21  ]
[ 4.0  24  ]
[ 4.1  25  ]
[ 4.2  17  ]
[ 4.3  20  ]
[ 4.4  27  ]
[ 4.5  20  ]
[ 4.6  22  ]
[ 4.7  14  ]
[ 4.8  18  ]
[ 4.9  12  ]
[ 5.0  12  ]   ]
```



Examination of the plot shows that the data decays to a value of 1/e times the t=0 value at about 1/5 of the x range. Time ranges from 0 to 5, so the decay constant is about 1. We know that the total counts at t=0 is about 650 and that the contaminant is about

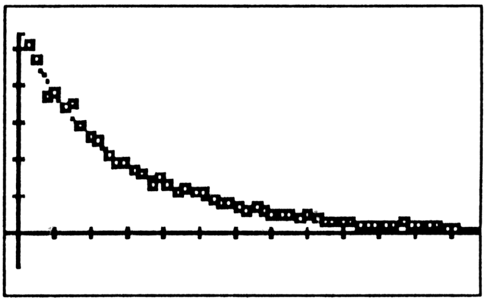
20%, so we might guess A1=500, A3=150, and A2=1. We don't know what A4 is, so we make two calculations, one with A4=2 and one with A4=0.5.

```
GUESS1:  [ 500 1 150 2 ]
GUESS2:  [ 500 1 150 0.5 ]
```

Now run NLR and enter the names DAT3NLR, FX, and GUESS1 at the prompts for data name, function name, and first guess vector. Select POISSON when prompted for the weighting type. Finally, select YES from the "Plot Data" menu to begin execution.The program displays a status screen during computation, showing the current iteration, the current value of χ^2, and the current value of the conditioning factor, λ. This example requires about 2 minutes to converge to a solution after 2 iterations. Upon completion the program will beep and proceed to plot the resulting data and fit curve. The results for the problem here are displayed on the stack.

```
    5.      CHISQR : 38.49
    4.      A1 : 483.9
    3.      A2 : 0.973
    2.      A3 : 158.4
    1.      A4 : 1.81
```

The final plot produced by NLR showing both the data and the function is shown in the figure below. Save the contents of $COEF into another variable.



Now run NLR again with GUESS2. With this first guess, the program converges in 5 iterations after 4¾ minutes. The stack output is shown below:

```
    5.      CHISQR : 37.71
    4.      A1 : 486.0
    3.      A2 : 1.35
    2.      A3 : 174.2
    1.      A4 : 0.563
```

The plot from GUESS2 is not substantially different from that obtained by GUESS1 yet the value of χ^2 for GUESS2 is better than for GUESS1. More importantly, look at the probable errors associated with the coefficients for the two guesses. For GUESS1, the probable errors for A1-A4 are ±260, ±0.22, ±268, and ±0.97. For GUESS2 they are ±84, ±0.078, ±75, and ±0.21. Problems associated with double or triple exponential decays are frequently ill defined and more data would help, but based on the data at hand, it appears that the fit parameters produced by GUESS2 are the correct ones.

Hints

Execution time.

In order to run the regression in reasonable times (a few minutes as opposed to many tens of minutes) it may be necessary to edit the number of data points in large data sets to a few dozens of points, at least for fits with 5 to 10 fitting parameters. Once a first guess is found which leads to rapid convergence to the proper solution, the entire data set can be included for the final calculation.

Validity of results.

As with any non-analytic regression, always check the results for reasonableness. Paying attention to the plot of the fitted data is usually a good indicator. It is sometimes the case that many different "solutions" are possible, differing only slightly in their χ^2. This is often the case for multiple closely spaced Gaussians and multiple exponentials, where the minima in χ^2 hyperspace are shallow and there may be many of them. The final answer in these cases is often dependent on which minimum was closest to the first guess. A sampling of first guesses may be required for certain problems.

Unexpected Termination.

If convergence does not occur, the fit parameters can go to infinity as the algorithm goes unstable. If a program termination occurs due to a system error such as overflow or singular matrix, check the values of the Ai fit parameters that are left behind as variables in the current directory to determine if lack of convergence was the underlying cause. Examination of the stack may also provide a clue. A poor first guess is likely to be the cause of termination. The stack and fit parameters are not normally saved when program termination occurs. To view of these items, create the variable 'nlrdebug' and store the value of 1 into it before running NLR. The automatic cleanup of the stack and variables created by the program will be suppressed.

Parameter control.

For certain applications, the user may wish control over the loop control parameters. These include; tolerance: %tol, initial conditioning factor: $λ, inflation factor: $λinf, and the deflation factor: $λdef. NLR checks to see if any of the variables exist at run time. If any of these variables do exist when NLR is run, the value of that variable will be used rather than the built-in default values. The default values are shown in the flow chart. They are %tol=1, $λ=0.01, $λinf=10, and $λdef=5.

References

"Data Reduction and Error Analysis for the Physical Sciences" Philip R. Bevington, McGraw-Hill.

"Nonlinear Regression Analysis and it's Applications" D. Bates & D. Watts, Wiley.

Multiple Linear Regression

MLR can analytically solve for up to 16 fitting parameters for the case where the fit function is linear in the fit coefficients, i.e. F = A1*F1 + A2*F2 + ... An*Fn, where the A's are the fit coefficients and the F's are the user supplied fit functions. Examples are polynomial regressions, where $Fi=X^{i-1}$, and Fourier fits, where $Fi=\sin[2\pi(i-1)X]$. MLR has the capability of providing for three types of data weighting; no (equal) weighting, Poisson weighting, or instrumental weighting.

Theory

The solution coefficients, A_j, result when the weighted chi-square is minimized. MLR uses the matrix inversion technique to obtain an analytic solution to the equation resulting from minimization:

$$\partial\chi^2/\partial A_k = \partial/\partial A_k \sum_i \{ \sigma_i^{-2} \{ y_i - \sum_j [A_j F_j(X_i)] \}^2 \} = 0 \qquad (1)$$

where: i - index over data points

j,k - index over fit parameters

$\chi^2 = \Sigma\sigma_i^{-2} \{ y_i - F(X_i) \}^2$ (weighted chi-square)

σ_i = standard deviation or error bar (weight) of measurement

After differentiation, Equation (1) results in n simultaneous equations, one for each kth partial derivative.

$$\sum_i [\sigma_i^{-2} y_i F_k(X_i)] = \sum_j \{ A_j \sum_i [\sigma_i^{-2} F_j(X_i) F_k(X_i)] \} \qquad (2)$$

In matrix form; **B** = **A C**,

with $B_k = \Sigma[\sigma_i^{-2} y_i F_k(X_i)]$

and $C_{jk} = \Sigma[\sigma_i^{-2} F_j(X_i) F_k(X_i)]$

The symmetric matrix **C** is called the curvature matrix.

Solving for **A**, we have;

$$\mathbf{A} = \mathbf{B}\,\mathbf{C}^{-1}. \qquad (3)$$

Data Weighting

MLR permits 3 types of data weighting;

Equal weighting (no weighting).
The standard deviations of the measured data are set internally to 1, (σ=1).

Poisson weighting.
Poisson weighting is characteristic of counting experiments. The standard deviations of the measured data are equal to the square root of the measurement ($\sigma=\sqrt{y}$).

Instrumental weighting.
Instrumental weighting is user specified for each data point. This weighting is appropriate when the random errors for the dependent data vary from measurement to measurement in a way determined by the instrumentation used.

Instructions

Before running MLR, prepare an i row by j column X;y data matrix (or X,y,σ matrix, where the third column of the data matrix is required for instrumental, or user specified weighting). Also create n function definitions, F_j, corresponding to each of the fit coefficients, A_j. When writing the function definitions, assume the variable X exists as a global. It will be created at run-time.

Run MLR. The introduction page displays a menu with choices; CONTINUE, HELP, and QUIT. HELP serves as a reminder of what to prepare as input, and how to interpret the output of MLR. After choosing CONTINUE or paging through the HELP option, MLR prompts for the number of independent variables, ie. a 1D or a 2D fit. MLR then prompts for the data name. Enter the name of the data array. If a third column exists in the data array, MLR knows that instrumental weighting is required. If a third column does not exist, MLR asks for the user to specify EQUAL or POISSON weighting using a highlight bar. The up and down arrow keys are active until a choice is made using the ENTER key. MLR finally prompts for the list of fit functions (or the name of a previously defined list). The list consists of global names for the F_j's created by the user. The order of the variable names implicitly corresponds to the order of the A_j's that are returned as the solution. Before execution, the user is offered a menu choice as to whether or not a solution plot is desired.

Upon completion, the program beeps and displays the results. The stack contains the final value of χ^2 and all of the resultant fit parameters, all tagged for easy identification. MLR creates 3 global variables. They are:

$COEF -	The solution fit coefficient matrix, the first column containing the A's, and the second column containing the P.E., or probable error in the corresponding coefficient. (P.E. = 0.6745σ). The statistical probability of the "actual" coefficient falling within the range $A_i \pm PE_i$ is ½.
$FX -	An algebraic representation of the fit function, constructed from the function list and the fit coefficient vector.
CHISQR -	The solution χ^2 for the fit.

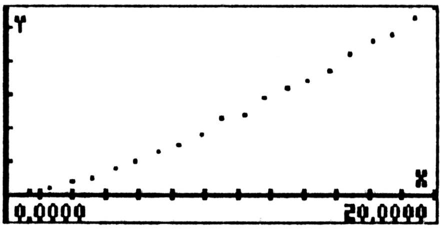
Example 1

We will do a quadratic polynomial fit to the first 20 primes. Three fit functions are required.

```
F1 : «1»        '1'
F2 : «X»        'X'
F3 : «X SQ»     'SQ(X)'
```

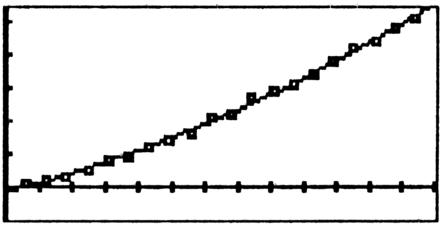
The data array is constructed as shown below.

```
DAT1MLR:
[  [1  2]
   [2  3]
   [3  5]
   [4  7]
   [5  11]
   [6  13]
   [7  17]
   [8  19]
[9  23]
[10 29]
[11 31]
[12 37]
[13 41]
[14 43]
[15 47]
[16 53]
[17 59]
[18 61]
[19 67]
[20 71]   ]
```



Run MLR, select 1 independent variable, and enter the name DAT1MLR in response to the data name query. Hit the ENTER key in order to select the default condition of equal weighting. Finally, enter the list of fit functions to be used for the fit; {F1 F2 F3}. Select YES from the "Plot data" menu. About 25 seconds later, MLR beeps and generates a solution plot, shown below with the stack output.

```
4:  CHISQR: 22.9901
3:  A1: -1.9237
2:  A2: 2.2055
1:  A3: 0.0747
```



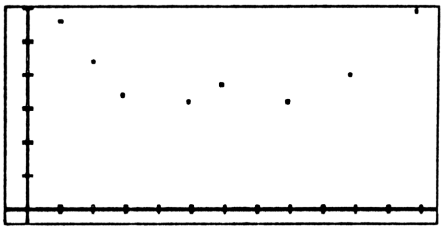
Example 2

We will use a function composed of the first three even Legendre polynomials to fit a data set which represents counts versus angle. The function we want is: F(X) = A1*P0 + A2*P2 + A3*P4, where P0=1, P2=$(3t^2-1)/2$, and P4=$(35t^4-30t^2+3)/8$, with t=cos(X).
The three function definitions are:

```
P0: «1»
P2: «X COS SQ 3 * 1 - 2 /»
P4: «X COS 4 ^ 35 * X COS SQ 30 * - 3 + 8 /»
```

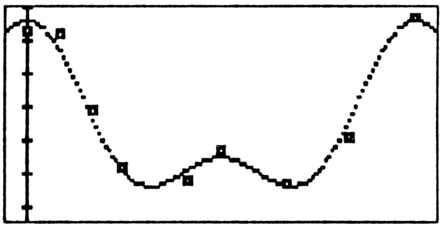
The data set is constructed as shown below. Set the angular mode to degrees.

```
DAT2MLR:
[   [0  301]
    [15  296]
    [30  230]
    [45  181]
    [75  170]
    [90  194]
    [120  167]
    [150  208]
    [180  312] ]
```



Run MLR, select 1 independent variable, and enter the name DAT2MLR in response to the data name query. Hit the down arrow, then the ENTER key in order to select Poisson weighting (count data is governed by Poisson statistics). Finally, enter the list of fit functions to be used for the fit; {P0 P2 P4}. Select YES from the "Plot data" menu. About 15 seconds later, MLR beeps and generates a solution plot, shown below with the stack output.

```
4:  CHISQR : 3.0146
3:  A1 : 189.8206
2:  A2 : 52.6633
1:  A3 : 66.4901
```



Lets look at the the probable error in the coefficients. Recall the variable $COEF and examine the second column for the probable errors. We see that A1, A2, and A3 have probable errors of ±3.73, ±6.83, and ±8.12 respectively.

Example 3

A final example will illustrate the use of instrumental weighting of data. Assume that a hypothetical experiment is performed and data is collected. All of the data is good data and is free of systematic errors. Half of the data is measured with instrumentation which yields ±0.5 error bars (uncertainties) and the other half of the data is measured with instrumentation which yields ±0.15 error bars. Let's say we know that a linear function describes the relationship between the X and y values and we wish to extrapolate a linear fit to obtain an estimate for some physical parameter at X=0. The data is shown below, where the third column contains the instrumental uncertainties.

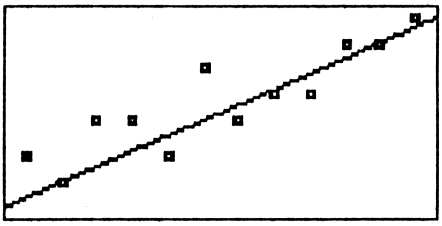
```
DAT3MLR:
[   [2.5  2.125 0.5 ]
    [3.0  1.875 0.5 ]
    [3.5  2.50   0.5 ]
    [4.0  2.50   0.5 ]
    [4.5  2.125 0.5 ]
    [5.0  3.00   0.5 ]
    [5.5  2.50  0.15]
    [6.0  2.75  0.15]
    [6.5  2.75  0.15]
    [7.0  3.25  0.15]
    [7.5  3.25  0.15]
    [8.0  3.50  0.15]   ]
```

We will fit the data to F(X) = A1*F1 + A2*F2. The functions are constructed as follows.

```
F1 : <1>          '1'
F2 : <X>          'X'
```

Run MLR,select 1 independent variable, and enter the name DAT3MLR in response to the data name query. Enter the list of functions to be used for the fit: {F1 F2}. Select YES from the "Plot Data" menu. About 10 seconds later, MLR beeps and generates a solution plot, shown below with the stack output.

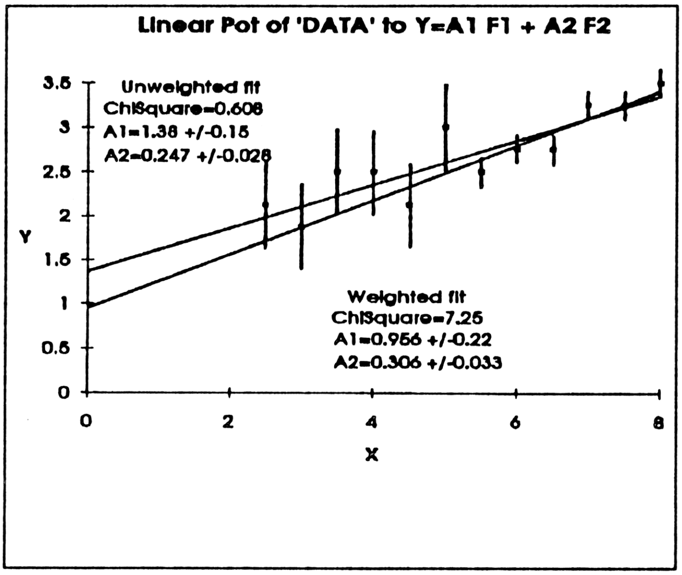
```
3: CHISQR:  7.2505
2: A1:  0.9559
1: A2:  0.3062
```



The best estimate for the X=0 intercept, given the available data, is y=0.96.

A Comment on Weighting of Data

If the fit in example 3 were done without utilizing the different weights for the two halves of the data set, the calculated intercept would be y=1.38, as shown in the figure below. This may be acceptable if one needs only a quick estimate for the intercept, but this answer is not the correct one because important information was thrown away in arriving at the value of the intercept. Inclusion of weighting in general (instrumental or Poisson) is especially important when the number of data points is small. If the data is really free of non-random errors, then the unweighted fit will converge to the weighted fit as the number of regularly spaced data points goes to infinity.



References

"Data Reduction and Error Analysis for the Physical Sciences" Philip R. Bevington, McGraw-Hill.

2 Dimensional Linear Regression

The minimization condition presented in Equation 1, need not be restricted to functions of a single independent variable, X. The method can be broadened to include functions of more than one variable, ie F(X1,X2). In this case our data set would consist of an array with 3 columns, one each for the independent variables X1 and X2, and one for the dependent variable Y. In a fashion analogous to the 1 dimensional case, instrumental weighting of the Y data values can be represented by a fourth column in the data array.

Instructions

The instructions are similar to those described in the general section on multiple linear regression. First create or port in an X1,X2,y data matrix (or X1,X2,y,σ if instrumental weghting). Create n function definitions, F_j, corresponding to each of the fit coefficients, A_j. When writing the function definitions, assume the variables X1 and X1 exist as globals. They will be created at run time.

When MLR is run, select 2 as the number of independent variables, and proceed as described in the general section on Multiple Linear Regression. Plots of the data and fit function are not done when the two dimension option is chosen.

Example 1

We will fit a function to a data set which represents the dependence of some parameter on both latitude and time, y=F(L,t). The function which describes the dependence is

F(L,t) = a + b*(1-COS(L)) + c*SIN(360*(t+d)/52),

where the arguments of the SIN function indicate that we must set the angular mode to degrees, and that the unit of time in the data set is week. The function in its current form is not linear in the fit parameters, but is easily transformed to a linear representation. After expanding the SIN function,

```
F(X1,X2)  = A1 + A2*(1-COS(X1))
            + A3*SIN(360*X2/52) + A4*COS(360*X2/52),
```

where X1=L, X2=t,
$c=(A3^2+A4^2)^{1/2}$, d=52*ATAN(A4/A3)/360.

The fit functions are:

```
F1:   <1>                      '1'
F2:   <1 X1 COS ->             '1-COS(X1)'
F3:   <360 X2 * 52 / SIN>      'SIN(360*X2/52)'
F4:   <360 X2 * 52 / COS>      'COS(360*X2/52)'
```

Since the seasons in the northern and southern hemispheres are 26 weeks out of phase and the time dependence is a seasonal one, the raw data for the minus latitudes has to be adjusted by adding 26 weeks (modulo 52). The data set shown below incorporates this modification:
DAT4MLR:

```
[   [ -77 44  301 ]
    [ -77 48  330 ]
    [ -77 50  292 ]
    [ -77 52  318 ]
    [ -77  3  331 ]
    [ -32 30  269 ]
    [ -32 35  255 ]
    [ -32 43  275 ]
    [ -32 14  309 ]
    [ -32 17  320 ]
    [ -17 31  241 ]
    [ -17 38  240 ]
    [ -17 52  269 ]
    [ -17 16  270 ]
    [ -17 20  260 ]
    [   4  2  296 ]
    [   4 10  302 ]
    [   4 14  284 ]
    [   4 32  245 ]
[  4 42  263 ]
[ 21  1  284 ]
[ 21  4  289 ]
[ 21 11  306 ]
[ 21 30  261 ]
[ 21 46  289 ]
[ 46  4  298 ]
[ 46  6  296 ]
[ 46  9  297 ]
[ 46 25  278 ]
[ 46 38  253 ]
[ 62  6  335 ]
[ 62  7  314 ]
[ 62 31  270 ]
[ 62 49  315 ]
[ 62 52  296 ]
[ 82  8  345 ]
[ 82 11  333 ]
[ 82 18  309 ]
[ 82 29  300 ]
[ 82 31  287 ]     ]
```

Run MLR, select 2 independent variables, and enter the name DAT4MLR in response to the data name query. Enter the list of functions to be used in the fit: {F1 F2 F3 F4}. About 90 seconds later MLR beeps and displays the results on the stack.

```
CHISQR:  5545.266
A1:      270.717
A2:      47.982
A3:      19.478
A4:      12.793
```

The values of c and d are calculated from A3 and A4.

c= 23.3	-the amplitude of the seasonal dependence.
d= 4.8 weeks.	-the phase difference

TVOM
Time Value of Money

Introduction

This financial program is an improved and expanded version of the one developed for the HP-41 PPC ROM. This version of TVOM includes the ability to generate amortization schedules and summaries quickly and with great precision. The program allows for full and independent control over the payment and compounding frequencies, including continuous compounding, and handles ordinary annuities and annuities due.

Assumptions/Defaults

The interest rate defines the base period. A one year base period is usually implied, ie a 10% mortgage rate defines the base period as 1 year. Other base periods are possible. The payment and compounding frequencies are the number of payments or compoundings in a single base period, ie. 12 if the payment and compounding periods are monthly with a yearly base period, 13 if the payment and compounding periods are weekly with a quarterly base period. The default mode assumes 12 payment and compounding periods per base period, which is useful for the most common problem of monthly compounding and monthly payment frequencies, with an implied yearly base period. (ie. yearly interest rate). Ordinary annuity is the default. The financial parameters (n, %int, PV, PMT, & FV are set to zero when TVOM is started.

Definitions/Functions

n	Total number of payments, or number of compounding periods if no payments. • If no payments and continuous compounding, then n is defined as the number of base periods.
%int	Nominal interest rate per base period (usually a year).
PV	Present value.

PMT	Payment amount.
FV	Future value.
PF	Payment frequency, or number of payments per base period. * Automatically set to CF if there are no payments (PMT=0). * Automatically set to 1 if PMT=0 and continuous compounding.
CF	Compounding frequency, or number of compoundings per base period. * Ignored if compounding is continuous.
C/D	Toggle function: continuous or discrete compounding.
B/E	Toggle function: E (ordinary annuity) - payment due at the end of the payment period. B (annuity due) - payment due at the beginning of the payment period.

Equations

Standard financial definitions are used. Money received is positive, money paid out is negative.

The basic financial equation is:

$PV(1+i_e)^n + PMT(1+i_e*OAAD)*[(1+i_e)^n-1]/i_e + FV = 0$

where OAAD=0 for ordinary annuity
OAAD=1 for annuity due

and i_e=effective interest rate per payment period.

The relation between nominal interest rate per base period and effective interest rate per payment period is:

$i_e = e^{(\%int/PF)}-1$ continuous compounding
$i_e = (1+\%int/CF)^{(CF/PF)}-1$ discrete compounding.

The solution for interest is accomplished iteratively by root solving which executes rapidly for a sufficiently good initial guess. The initial guess used is:

$$i_0 = |PMT/(|PV|+|FV|)| + |(|PV|+|FV|)/(PMT*n^3)|.$$

The amortization formulas used are (through N periods):

interest contribution = $N*PMT - (PV+PMT/i_e)*((1+i_e)^N-1)$.
principal contribution = N*PMT - interest contribution.
balance remaining = PV - principal contribution.

Menu Keys

n, %int, PV, PMT, FV
Inverse background keys that function as those in SOLVER, ie, unshifted to enter a value, left-shift to solve, right-shift to recall.

QUIT
The only proper method of exit. 'Quit' purges globals and restores user and system flags.

INIT
Initializes all parameters to their default values.

VIEW
Temporary display showing current option values.

C/D, B/E
Toggle functions described in Definitions/Functions.

CF, PF
Compounding and Payment Frequency functions. Unshifted key stores a new value, right-shifted key recalls the value.

AMRT
Amortization summary for input number of periods.

AT.I
Amortization Table showing Incremental interest and principal contributions at each pay period, along with new balance.

AT.C
Amortization Table showing Cumulative interest and principal at each pay period, along with new balance.

INDX
Used to index amortization tables, ie. a new column is added to the table to represent payment number. This function is

intended for use where a table is to be ported out to a PC. An index is required to make the raw data readable outside the HP-48 environment.

Caveats/Limitations

The program gives accurate results for terms of 100 years with monthly payment and compounding frequencies, at least for reasonable values of the financial parameters. The cumulative errors over 1200 payments might amount to a few cents. The results for each period are rounded and displayed to the nearest cent, except for the amortization summary which is rounded to the nearest 100th cent. The only real limitations the user is likely to find are resource driven limitations encountered in generating amortization tables. Generating tables for 1200 payments takes about 5 minutes, and more importantly, uses large amounts of memory. Be aware that an out of memory condition may occur if these large tables need to be generated.

References:

PPC ROM User's Manual, 1981, and references contained therein.
In particular:
Greynolds, Aronofsky, Frame, "Financial Analysis Using Calculators", McGraw-Hill, 1980.

Examples

The following examples illustrate some of the uses of TVOM. Each example assumes a fresh start of TVOM or execution of the INIT function which initializes all variables to their default values. In these examples, gold-shift or left-shift is represented by the symbol ↖ and blue-shift or right-shift is represented by the symbol ↗. The ENTER key is represented by the symbol ↑. Menu keys will be represented by outlined characters eg. **%int**. What you see will be noted by the superscript s if it is displayed in the status area rather than the stack.

<u>Example 1</u> Find the monthly payments for a 4 year, $12,000 car loan at 11½%. (Assume PF=CF=12 and ordinary annuity).

Do:	See:	Comment:
4 ↑ 12 * **n**	^{s} n: 48	48 monthly payments
11.5 **%int**	^{s} int: 11.50	nominal interest rate
12000 **PV**	^{s} PV: 12000.00	cash received (+)
↖**PMT**	PMT: -313.07	monthly payment, cash out (-)

<u>Example 2</u> Find the monthly end-of-period payment necessary to fully amortize a 15 year $70,000 loan at 8.75%, compounded quarterly.

Do:	See:	Comment:
4 **CF**	^{s} COMPFREQ: 4	quarterly compounding
15 ↑ 12 * **n**	^{s} n: 180	180 payments
8.75 **%int**	^{s} int: 8.75	nominal interest rate
70000 **PV**	^{s} PV: 70000	cash received (+)
↖**PMT**	PMT: -697.01	payment, cash out (-)

How much interest was paid out the first year?

12 **AMRT**	Princpl: -2376.39	
	Interest: -5987.74	shows total interest paid
	Bal: 67623.61	& balance remaining

Example 3 Compute the future value of bi-weekly savings of $100 for 3 years at 7.5%, compounded daily. Interest is paid on deposits beginning from when they are deposited, so set B/E to B (or annuity due). The account is opened with a $300 deposit.

Do:	See:	Comment:
26 **PF**	s PMTFREQ: 26	deposit every 2 weeks
365 **CF**	s COMPFREQ: 365	daily compounding
B/E	s Annuity Due (BEG)	
3 ↑ 26 * **n**	s n: 78	3 years; (26/yr)
7.5 **%int**	s int: 7.50	nominal interest rate
-300 **PV**	s PV: -300	$300 to open (cash out)
-100 **PMT**	s PMT: -100	$100 deposits
↖**FV**	FV: 9135.39	withdrawal at 3 years

Example 4 If you purchase a single payment annuity with a value of $60,000, invested at a nominal interest rate of 14%, compounded continuously, what is the maximum monthly return which does not disturb the principal? (This is an ordinary annuity). Notice n is a dummy entry since the annuity is a perpetuity one. Also note that CF is irrelevant in this case.

Do:	See:	Comment:
C/D	s Continuous Compounding	
1 **n**	s n: 1	dummy entry
14 **%int**	s int: 14.00	
-60000 **PV**	s PV: -60000	initial cost
60000 **FV**	s FV: 60000	final value (unchanged)
↖**PMT**	PMT: 704.10	monthly return.

CONSTANT

Introduction

CONSTANT is a library of physical, astronomical, and mathematical constants accessible via a menu. The library contains 21 constants from the physical sciences, 60 constants from astronomy and cosmology, and 43 mathematical constants.

The units for the entries are, with some exceptions, SI units (MKS). The exceptions are the Hubble and galactic distance units, which are in Mpc or kpc, and the constants for the planets other than earth, which are expressed as earth unit ratios and are therefore dimensionless.

The available library commands are:

• PHYSC	Physics & chemistry
• EARTHC	Earth & moon
• SOLARC	Sun & planets
• GALAXYC	Galaxy
• COSMC	Cosmology
• MATHC	Mathematics
• CONSTANT	

Notice that CONSTANT is not only the name of the library, but is also a library command. The library command CONSTANT, when evaluated, creates a temporary menu of the other library commands. The benefit of this feature is that the menu of library commands can become available by evaluating the name CONSTANT, rather that having to go into the LIBRARY menu, page to the menu entry for CONSTANT (the library name), and push its menu key.

The library's entries are accessed through a menu structure which emulates a directory with subdirectories. The library commands are the first level of entries in the directory-like structure. The entry for EARTHC contains an entry named MOON which emulates another subdirectory, one level down. The entry for SOLAR also contains a subdirectory-like entry named PLAN (planets). PLAN contains entries for all of the planets except earth, which are themselves subdirectory-like. The planetary constants are finally contained as entries under each of the planet names. Each level of menus contains an up arrow key (↑) which allows easy switching among the various levels of menu pages.

Instructions

To bring up the constant menu, evaluate the command CONSTANT, or go to the library menu and find and push the key labeled CONS. A menu of the 6 categories of constants will appear. Press one. In order to bring the value of a constant to the stack, press the menu key corresponding to the constant's symbol. The value returned will be the constant value, usually with units, and always with a descriptive tag. As a convenience, the label can be stripped by immediately pushing the same menu key a second time. If the value has units, a third press of the same menu key will strip the units and leave the bare number in level 1.

Library Contents

The various constants contained in the library are shown below.

PHYSC	universal gas constant
	Boltzmann's constant
	Planck's constant
	speed of light
	gravitational constant
	Avogadro's number
	molar volume
	electron charge
	electron mass
	proton mass
	Bohr radius
	Rydberg constant
	fine structure constant
	Compton wavelength (electron)
	Bohr magneton
	nuclear magneton

```
        vacuum permittivity
        Vacuum permeability
        Faraday constant
        Stefan-Boltzmann constant
        Wein displacement

EARTHC  mass
        equatorial radius
        flattening factor
        mean density
        equatorial surface gravity
        escape velocity
        orbital velocity at 350 km
        Geosynchronous altitude from the equator
        albedo
        solar constant at the earth

        MOON mass
             radius
             earth-moon distance
             equatorial surface gravity
             escape velocity
             orbital velocity at 100 km
             albedo

SOLARC  mass
        equatorial radius
        equatorial surface gravity
        solar luminosity

        PLAN MERC
                mass
                radius
                distance to sun
             VEN
                ""
             MARS
                ""
             JUP
                ""
```

SAT
""
URAN
""
NEPT
""
PLUTO
""

GALAXYC	mass of galaxy
	radius of galaxy
	number of stars
	sun to galaxy center distance
	relative velocity of sun
	galaxy luminosity
COSMC	Hubble constant
	critical density of the universe
	Planck length
	Planck time
	Planck mass
	nuclear density
	radiation temperature
MATHC	Euler's constant
	golden ratio
	Catalan's constant
	Bernoulli numbers (0-16)
	Euler numbers (0-16)
	perfect numbers (1-6)

(the last three entries require an argument)