Change options

The following options can be changed;

Iterations

Iterations is the maximum iterations done before terminating the custom fitting. The default is 100.

Convergence

Changing convergence will determine how similar two successive parameter evaluations must be before declaring a successful fit. The default is 1.E-6.

Damping factor

Enter here the maximum damping factor. The damping factor is used in determining the magnitude of the change in the parameters during each successive iteration. A smaller value may speed convergence but in more difficult cases lead to no convergence. The default is 1.E+6

Fitting hints

A satisfactory fit

A satisfactory fit is characterized as follows:

1. you require about 2n iterations before convergence where n is the number of parameters
2. the parameters are physically reasonable with tolerable standard errors
3. the smallest eigenvalue is about unity
4. you have no visible outliers

A good final test of convergence is to use Excel's own solver function to minimize the sum of squares of deviations. If Solver gives the same parameters as Custom Fit, then you can be reasonable sure that you have a satisfactory fit.

An unsatisfactory fit

The discussion below on reasons for unsatisfactory fits is very brief and serves only as a reminder of the types of problems that you may encounter. A recent text which gives a detailed description of fitting problems and solutions is " C curve fitting and modeling for scientists and engineers" by Jens-Georg Reich, McGraw-Hill (1992).

There are five principal reasons why your data may not lead to a satisfactory fit;

1. excessive fluctuations in the data

Noisy data will lead to a large sum of squares. You can identify this as the cause if you remove obvious outliers without seeing any significant changes. Normally, the standard error in the parameters in these cases will be large (80%) but not excessive. Plotting your results will visually identify noisy data.

2. your objective function is not compatible with your data

If you objective function is not compatible with your data, you will likely see a large sum of squares and systematic trends in the residuals meaning that you will see clusters of residuals with the same sign (either positive or negative).

3. you have one or more outliers in your data

If you can visually identify an outlier and the removal of the outlier significantly improves the fit, then you have a problem. Your problem is to identify a reason to remove the outlier. Your best bet is to identify an experimental error that gives you cause to remove the outlier. If this cannot be done, then you had best consult a text with more detailed information.

4. your objective function has redundant parameters

Redundant parameters will cause the fitting to slowly converge if at all. The eigenvalues listed at the end of the fitting will have at least one which is very small (<1.e-5). A good test is to use Excel's solver function and if it moves to a very different parameter set, then you likely have a redundant parameter. The remedy to this problem is to reduce the number of parameters, that is, replace one of the parameters with a well chosen constant. If one parameter has a large standard error and a near zero scale factor, then replace this parameter with a constant.

5. more than one function can fit your data because you have not extended your data measurements far enough to distinguish one model from the other.

The best test of this is to graph the results of your models. If the graph shows that the models diverge in a region with no data points, then the problem is visually apparent and easily corrected by extending your data.

If CustomFit exits after the first iteration!

At each iteration, CustomFit insists that the objective function be sensitive to changes in every parameter. If the sum of squares does not change when a parameter is changed, CustomFit will immediately exit. If this happens, then check your definition of the functions in the worksheet. They must contain a reference

to each parameter. If they don't, then you have an obvious error. Correct it and try again.

Graph it!

Finally, there is no better means of identifying the nature of your fit than by calculating and graphing every aspect of your fit. Since CustomFit is an integrated part of Excel, this should be an easy task and it sure beats those old command line driven single function programs.

Overwriting parameters & changing options

Parameters

You may choose to have either the current fitted parameters overwrite the starting parameters or have the starting parameters restored. In either case, the fitted parameters will be written to the output area. If you are having difficulty in converging to a solution see Fitting hints

Options

Selecting Yes and then OK will subsequently bring up a dialog to change the Custom fitting options.

About Xlmath

Xlmath is an Excel dynamic link library add-in. Xlmath adds custom functions for data fitting and smoothing and a real symmetric matrix diagonalization function. The add-in functions can be used via the 'Formula Paste Function' command or via dialogs invoked by selecting the Xlmath menu. For usage see Using Xlmath

Contents

Using Xlmath

Before you use an Xlmath tool, you must organize your data into columns or rows on a worksheet. This is your input range. If you label your variables with text labels, do not include these labels in your input range. All of your data must be in continuous ranges.

There are two alternative ways of using Xlmath tools. The first way is to select the tools in the Xlmath menu. The second way is to enter Xlmath functions directly into array formulae.

Xlmath Menu Tools

When you use an Xlmath menu tool, Excel creates an output table of the results. The contents of the table depend upon the tool that you are using. As a general rule, if your input range is a column vector, a N x 1 range, Excel will write the output table in a column vector. If your input range is a row vector, a 1 x N range, Excel will write the output table in a row vector. An exception to this rule can be found in the CubicSplines tool where the output is always a N x 3 range.

1. From the main menu choose Xlmath.
2. In the Xlmath box, select the tool that you want to use.
3. Type the input range, the output range and any requested numerical input.
   You can type cell ranges in boxes by typing a cell or range reference or a name of a cell or range reference, or by selecting the cell range on the worksheet.
4. Choose the OK button.

Restriction

When you are prompted in a dialog box for an input range, you must enter a range of at least two cells. Entering a reference to a single cell will halt the tool and produce an error message.

Xlmath Custom Functions

The Xlmath custom functions can be chosen from the Excel Paste Function menu. The custom functions are listed under the category Xlmath Add-In and must be pasted into an array formula which has the exact dimensions of the output table returned by the Xlmath menu tools.

See Also

User's Guide (Book1)

Diagonalize	Diagonalize a real symmetric matrix.
MODensity	Calculate charges & bond orders
Polynomial	Polynomial fitting of data points
Cubicspline	Cubic spline function fitting
CalcSpline	Interpolate between data points
CustomFit	Fitting to a user defined nonlinear function
SmoothSG	Savitsky-Golay smoothing of data.
SmoothWt	Data smoothing via weighted convolution
Exit	Remove Xlmath add-in tools
Custom Functions	View help on custom functions
Revision History	View Revision History
License	How you can use Xlmath
Workbook	Accompanying workbook XLMATH.XLW

Chapter 5, "Creating a Worksheet: Using Array Functions"

Workbook

The Xlmath package comes with a workbook called XLMATH.XLW. It is strongly recommended that you carefully examine the workbook files before using the Xlmath tools for the first time. The workbook contains four files.

XLMATH.XLS - this worksheet demonstrates the use of Xlmath tools in the form of custom functions. It is suggested that you not use custom functions unless your input data changes frequently and/or you are very familiar with the use of Excel array formulae.

XLMDLG.XLS - this worksheet demonstrates the use of Xlmath tools via menu choices and dialog boxes. This is the simplest way to use the Xlmath tools.

XLMCFIT.XLS - this worksheet demonstrates the use of the Xlmath custom fitting tool. The macros used in the demonstration fitting can be found in the macro sheet called XLMCFIT.XLM.

License

Xlmath is freeware*. This means that you can freely copy it, use it, modify it, and give copies to all your friends (as long you give them all of the *unmodified* files that you received).If you do encounter problems with Xlmath, or if you think of a way to improve it feel free to contact me.

Although I don't want cash for Xlmath, I am interested in hearing from people who use it. To this end, please send a note via post, EMAIL or a fax to:

Roy Kari
Department of Chemistry & Biochemistry
Laurentian University
Sudbury, Ont.
Canada
P3E 2C6

fax: (705) 675-4844
Internet: "ROY@NICKEL.LAURENTIAN.CA"

*Xlmath is freeware but if you find it useful in your work, a donation to the Chemistry Affect Fund would be very much appreciated. The Fund is used primarily to give scholarships and bursaries to our graduate students. Donations of $25.00 and more will be given a receipt for Income Tax purposes. We are a charitable institution. Donations should be mailed to
The Chemistry Affect Fund
c/o Dr. Werner Rank,
Department of Chemistry & Biochemistry
Laurentian University
Sudbury, Ont
Canada
P3E 2C6

XLMATH Command (Main Menu)

Displays the Xlmath menu with a list of curve fitting and Huckel molecular orbital tools.

Xlmath

Lists the available Xlmath tools.

See Also

Using an Xlmath tool	General instructions
Revision history	View revision history
License	Usage license
Diagonalize	Diagonalize a real symmetric matrix.
MODensity	Calculate charges & bond orders
Polynomial	Polynomial fitting of data points
Cubicspline	Cubic spline function fitting
CustomFit	Fitting to a user defined nonlinear function
CalcSpline	Interpolate between data points
SmoothSG	Savitsky-Golay smoothing of data.
SmoothWt	Data smoothing via weighted convolution
Exit	Remove Xlmath add-in tools
Custom Functions	View help on custom functions

Diagonalize

Employs the Jacobi method to compute the eigenvectors and eigenvalues of a real symmetric matrix. This tool can be used to compute the molecular orbital coefficients via simple Huckel MO theory. In this method, the real symmetric matrix is created by entering unity for element aij if atom i is bonded to atom j or zero if the atoms are not bonded.

Input range

Type the reference for the range of the real symmetric matrix. This range must be square (NxN). Since only the top half is used, the user need not define elements below the diagonal.

Output range

Type the reference for the upper-left cell of the output range. The output will be written in an (N+1 x N) range. The N eigenvectors are returned in N columns (of N rows) and the last row contains the eigenvalues.

CustomFit

In data fitting, one typically has m data values y1, y2, ... ym which have been sampled for values x1, x2, ... xm of some independent variable x. It is then desired to fit a function f(x,p) which has n adjustable parameters, to be chosen so that the function best fits the data. The residuals are given by
ri(p) = f(xi,p)-yi i = 1,2,...m
and a least squares solution is sought by minimizing S(p) which is the sum of the squares of the residuals . The Levenberg-Marquardt method attempts to solve for the solution to S(p) by stepping towards the solution via a sequence of corrections based on the solution to the equation
(**G** + a**I**)**s** = -**g**
where **G** is a matrix of second derivatives, **g** is a matrix of first derivatives, **I** is a unit matrix and a is a damping factor. The Levenburg-Marquardt method is also characterized by the scaling of the matrix **G** by replacing it with a scaled matrix **C** where
Cij = Gij/((Gii)1/2.(Gjj)1/2) or **C** = **D**-1**GD**-1.
The solution for the step **s** is then given by
s = -**D**-1(**G** + a**I**)-1**D**-1**g**
where **D-1** is a diagonal matrix and (**G** + a**I**)-1 is determined from the eigenvalue decomposition as **C** = **A**.**L**.**A**T where L is a diagonal matrix of eigenvalues and A is a matrix of eigenvectors. If none of the eigenvalues are zero, then the inverse of **C** exists and is defined as **C**-1 = A.**L**-1**A**T.

Pragmatically, iterations must also be halted when no significant change is obtained in two successive choices of the parameters. This condition will be encountered when the model nonlinear equation has a parameter redundancy. For reference see Marquardt, D.M. "An algorithm for Least-Squares Estimation of Nonlinear Parameters", J. Soc. Indust. Appl. Math., 11, 431-441, 1963

As a general rule, "best-fitting" is obtained only when
1. termination occurs before the maximum number of iterations (50)
2. S is minimal
3. there is no parameter redundancy indicated by the eigenvalues near zero
4. scale vectors are finite

Data range

Type the reference for the range of the measured and calculated dependent variables. This range must be a single continuous range of m rows by 2 columns with the measured variable (Y) in the first column. The second column must contain a calculated value based on the X values (1 or more independent variables) and the parameters as described below. The contents of the calculated value cells must be functions. CustomFit replaces the parameters as required and obtains the corresponding function values direct from the worksheet. You will not see this until a fit has been obtained.

Parameter range

Type the reference for the range of the parameters and their upper and lower bounds. This range must be a single continuous range of 3 rows and n columns. The first row contains the initial starting parameters, the second row the upper bound for these parameters and the third row is the lower bound for these parameters.

Output range

Type the reference for the upper-left cell of the output range. The output will be written in an 4 x n range. The rows contain information as follows:

row 1: final parameters (p1, p2, ... pn)
row 2: standard deviation (or standard error) of the final parameter estimate (use to construct confidence intervals)
row 3: eigenvalues (non zero if no redundancy)
row 4: scale vectors (measure of response to change in parameter value). These are non-zero for satisfactory fits

Hint:

If you experience difficulties in getting any results, see Fitting hints

PolyCurveFit

Polynomial curve fitting results in a single polynomial equation of order m which is the least squares approximation of the observed data.
y = c0 + c1 * *X* + c2 * *X*2 + c3 * *X*3 ... + cm * *X*m
This tool will compute the coefficients *c*i for the polynomial which minimizes the sum of the squares of the deviations from the calculated and observed values for y. In addition to the coefficients, this command will return values which enable the user to assess the quality of the fit.

The most important measure of the quality of the polynomial fit is the correlation coefficient. The closer this value is to 1, the better the fit. Another measure of the fit is the coefficient of determination, usually referred to as R2. This value is equal to the square of the correlation coefficient The standard error of the estimate, abbreviated as SEE, is a measure of the scatter of the actual data along the fitted line. The smaller the SEE value, the closer the actual data is to the computed polynomial

Input

Xvar range

Type a reference to a column or row range for the *N* independent variables (*X*).

Yvar range

Type a reference to a column or row range for the *N* dependent variables (*Y*).
Xvar and *Yvar* must both be either row or column vectors. Using one as a row vector and the other as a column vector is not supported. It is recommended that both variables be entered as column vectors.

Order

Type a number for the order m of the fitting, i.e. 1 for a linear fit, 2 for a quadratic fit and etc. The order must be one less than the number of variables.

Output

Type the reference for the upper-left cell of the output range. The output range is an N x 3 array if the input is in the form of a column vector or a 3 x N array if the input is in the form of a row vector. Assuming that both Xvar and Yvar are column vectors and the array formulae have been entered into a N x 3 array, then the first column of the output contains the estimated Y values, the second column contains the residuals (differences between calculated and estimated y-values). The third column contains in the first (order + 1) rows, the polynomial coefficients. If fitted to 2nd order, the first three rows contain c0, c1, & c2.

Column 1	**Column 2**	**Column 3**
estimated y values	residuals	first N+1 rows are coefficients for remaining rows see below

The following values are returned directly below the coefficients,

coefsig	a vector of dimension (order+1). Coefsig are the standard errors of coefficient estimates. The values are stored in the same order as the polynomial coefficients.
see	the standard error of the estimate
rsqrval	the r squared value - the sample correlation coefficient
cferror	returns 1 if the curve fit is singular, otherwise 0.

CubicSplines

Fits a discrete set of cubic polynomial equations to a discrete set of data points. Whereas polynomial curve fitting produces a single equation to fit the data points, cubic splines curve fitting produces a family of cubic equations, one cubic equation for each interval in the original data. Cubic spline curve fitting guarantees that the fitted curve will pass exactly through the original data points.
The original X values and returned cubic spline coefficients may be subsequently used to interpolate for points between the original data points. For details see CalcSpline.

Input

Xvar range
Type the reference for a range of N independent variables (*X*).

Yvar range
Type the reference for a range of N dependent variables (*Y*).
Both variables can be entered as row or column vectors.

Output
Type the reference for the upper-left cell of the output range. CubicSplines will ALWAYS return the coefficients in an Nx4 range.

SmoothSG

Performs a Savitsky - Golay simplified least squares smoothing and differentiation of data (see Savitsky, A. and Golay, J., Analytical Chemistry 36 (1964), p. 1627). This technique uses convolution where each data point is recalculated as a weighted average of its original value and the surrounding data points. The degree of smoothing is a function of the number of surrounding data points used in the convolution, and the larger the convolution kernel, the larger the degree of smoothing.

Input

Data range

Type a reference to a column or row vector of the data to be smoothed. If the data is a column vector, then the output is a column vector and *vice versa* if the data is a row vector.

SmoothNum -

Type the integer degree of smoothing
1 = 5 point smooth
2 = 7 point smooth
3 = 9 point smooth
4 = 11 point smooth
5 = 13 point smooth

DerivNum

Type the integer derivative degree
0 = smooth data only
1 = first derivative
2 = second derivative

Output

Type the reference for the upper-left cell of the output range. If the input is entered in a column range, the output will return in a column range. If the input is entered as a row range, the output will return as a row range.

SmoothWT

Is used to reduce the noise in a sample. The technique uses convolution where each data point is recalculated as a weighted average of its original value and surrounding data points. The degree of smoothing is a function of the number of surrounding data points used in the convolution and the user supplied weights used in the recalculation.

Input

Data range

Type a reference for the range of data be smoothed. The data may be entered into a column or row vector.

Weights range

Type a reference for the range of weights used in the convolution process

Output

Type the reference for the upper-left cell of the output range. If the input is entered in a column range, the output will return in a column range. If the input is entered as a row range, the output will return as a row range.

Exit

Select this command to remove the XLMath add-in tools.

MODensity

Will allow the user to calculate the pi atom charge and bond order matrix for a simple Huckel calculation. The charge and bond order matrix in simple Huckel calculations is defined as a matrix multiplication of *C*(T) x *Occ* x *C* where *C* is the matrix of coefficients and *Occ* is a 1 dimensional matrix of occupancies. You must calculate the molecular orbital coefficients prior to calculating the charge and bond orders,

Input

Coef range

Type a reference for the range of the Coefficients(*Coef*).

Occ range

Type a reference for the range of Occupancies(*Occ)*.

Output

Type the reference for the upper-left cell of the output range. The output will be returned in an NxN range. In the output matrix, the diagonal elements represent the charges on the atoms and the off-diagonal elements between two bonded atoms represent the pi bond orders. Off-diagonal elements between two non bonding atoms have no meaning.

CalcSpline

Will calculate the cubic spline interpolated Y value of a given X value. Y values can be interpolated between the first and last original X values used to calculate the cubic spline coefficients. CalcSpline cannot be used to interpolate beyond the end points of the original X values. By definition, X values identical to those used to calculate the coefficients, will have an interpolated Y value identical to the original Y value.

Input

Xorig range

Type a reference for the original set of data points used to calculate the cubic spline coefficients.

Coef range

Type a reference for the range of coefficients previously calculated in CubicSplines.

Xcalc range

Type a reference for the range of independent variables (X) for which you wish to calculate interpolated Y values. NOTE, this is not the range used to calculate the coefficients but a new range of X values for which interpolated Y values are required.

Output

Type the reference for the upper-left-cell of the output range. The output will be returned in a column vector if Xorig is a column vector and in a row vector if Xorig is a row vector.

Array formulae

To enter array formulae, select a range of cells equal to the output area, build the formula and press CONTROL+SHIFT+ENTER. If you do not enter the formula properly, you will not see the complete result. See Chapter 5 of User Guide 1.

Xlmath Custom Functions

All of the commands in the Xlmath menu have corresponding custom functions. These custom functions can be accessed through the Excel Paste Function command and are listed under the heading Xlmath Add-In. All of the custom functions return a single array.which is identical in form to the output returned in the commands. This means that the custom function with the appropriate arguments must be entered into each cell of the output array. If you are not familiar with Excel's array formula usage, please read the section on array formulae in the Excel User's Guide. Custom functions are useful if your input data changes frequently. In this case, entering the custom functions in an array formula will allow for the automatic updating of the output.

The following is a description of each custom function in Xlmath Add-In

Function: **PolyCurveFit(Xvar, Yvar, Order)**
Arguments: Xvar - the range for the dependent variables
Yvar - the range for the dependent variables
Order - the order of the fit

Function: **CubicSplines(Xvar, Yvar)**
Arguments: Xvar - the range for the dependent variables
Yvar - the range for the dependent variables

Function **CalcSpline(Xorig, Coef, Xcalc)**
Arguments Xorig - the range for the original data points
Coef - the range for the coefficients derived from CubicSplines()
Xcalc- the range for the X values for which Y values will be calculated

Function: **SmoothSG(Data, SmoothNum, DerivNum)**
Arguments: Data - the range for the data points
SmoothNum - a reference to the degree of smoothing
DerivNum - a reference for the derivative degree

Function: **SmoothWT(Data, Weights)**
Arguments: Data - the range for the data points
Weights - the range for the convolution weights

Function: **Diagonalize(SymMat)**
Arguments: SymMat - the range for the real symmetric matrix

Function: **MODensity(Coef, Occ)**
Arguments: Coef - the range for the coefiicients derived from diagonalize
Occ - the range for the orbital occupancies

See Also

Diagonalize	Diagonalize a real symmetric matrix.
MODensity	Calculate charges & bond orders
Polynomial	Polynomial fitting of data points
Cubicspline	Cubic spline function fitting
CalcSpline	Interpolate between data points
CustomFit	Fitting to a user defined nonlinear function
SmoothSG	Savitsky-Golay smoothing of data.
SmoothWts	Data smoothing via convolution

Source Code

Source Code

The source code required to write your own XLL is **not** included in XLMATH v3.0. If you have an Internet connection, the author will send you a part of the code required to re-compile Xlmath 3.0 For personal reasons, the author will not release the source code for the actual operational functions. You don't need it since you have a fully executable version of these functions. In addition, the source code for the curve fitting and data smoothing routines are a modified form of the Science & Engineering Tools routines sold by Quinn-Curtis. This source is copyrighted by and belongs to Quinn-Curtis but may be purchased from them at 35 Highland Circle, Needham, MA 02194 USA.

Memory Management

For those of you who have read my article in J.Chem.Ed., a thousand apologies. Memory management has been a most difficult aspect of Windows(TM) and all of the difficulty arises from the "real" mode requirements that pre-date Windows 3.0. Finally, I have got it straight and the straight answer is that memory management is very simple. Just do it as you did before. The following is a quote from an article in the Microsoft Devoper's Network CD by Dale Rogerson (The C/C++ Compiler Learns New Tricks)

1. Use the LARGE model
2. Use malloc().

Now what could be more simple? The source code in Xlmath contains a lot of casts to ensure that the pointers are FAR pointers. You don't need these casts because the large model compiler automatically casts all data to FAR. It was just too much work to remove them so as the saying goes "do as I say and not as I did". Be particularly careful not to use the Windows definitions of near pointers such as NPSTR. These definitions use the keyword "__near" in their definition and the compiler cannot convert this to a __far pointer.

Revision History

Differences between v2.2 and v3.0

1. v3.0 incorporates a major revision of CustomFit which no longer requires that the user write a macro. It should have been originally written in the current manner. SmartHeap(tm) is now once more being used but statically linked into the object code. SmartHeap is faster and more efficient than standard alloc() and makes it easier to debug the code. The source code is no longer distributed with v3.0. Minor corrections have been made to the sample workbook and to the Help file.

Differences between v2.1 and v2.2

1. If you selected an Xlmath menu item while an embedded chart was selected on the worksheet, the whole system crashed. An additional check is now made to ensure that a cell or range of cells is selected on the worksheet. If this is not so, then an error message is displayed and the Xlmath dialog is not executed. A few additions were made to this help file. As an example, this revision history message is not available in prior versions. Finally, this version was created with the aid of Microsoft Visual C/C++ v1.0

Differences between v2.1 and v2.0

1. The Xlmath menu did not appear if you were using a a language variant of Excel that did not have the command "Help". This has been corrected and the Xlmath menu will appear regardless of language. Hopefully, it will also run on all language variants of Excel.
2. There was a memory leak in the routines performing SG and WT smoothing. If you used these routines repeatedly, in previous versions, each use would increase the amount of memory used by Xlmath. You likely have not noticed this unless you inspected the memory usage with Heapwalker.
3. The behaviour of the memory allocation schemes (malloc() etc) in Microsoft C/C++ version 7.0 has become more compatible with Windows 3.1. Hence the memory allocation program SMRTHEAP.DLL has been eliminated and _fmalloc() and _ffree() substituted where required (see MS Developers Network: Allocating Memory the Old-Fashioned Way: _fmalloc and Applications for Windows[TM], 1992, Dale Rogerson, Microsoft Corporation). Xlmath is now a large model DLL. It is still easier to debug your program with SMARTHEAP (and creates a faster executable if you believe the advertising) and hence I have left the SMARTHEAP statements in the code but they are now invoked only when specified in the makefile.

Differences between v2.0 and v1.0

There are two fundamental difference between v1.0 and v2.0. XLMATH v2.0 includes both custom functions and commands. The commands are invoked by selecting the menu *Xlmath* and completing the dialog box prompts. XLMATH v2.0 also uses the Excel API to both register the custom functions and commands and to run the dialog box routines. The Excel API eliminates the need for a macro sheet. Since Excel has its own version of Frequency(), the XLMATH version has been deleted.

Xlmath v1.0

Xlmath v1.0 is described by the author in an article published in the Journal of Chemical Education (2nd quarter of 1993). The intent of the author in this article and in Xlmath was to make persons aware of the ease with which DLL's could be written for Excel and to convince educators and others to attempt to write their own DLL's and to abandon stand-alone programs. Since the writing of the paper and the development of v1.0, Microsoft published the Excel API (Microsoft Press, ISBN# 1-55615-521-2). The publishing of the API made it even easier to write standalone DLL's and made it possible to interface the custom functions and commands without the need for any macro language. Since the publication of the API made v1.0 obsolete, the author decided to revise v1.0 and re-write v2.0 to conform to the API.