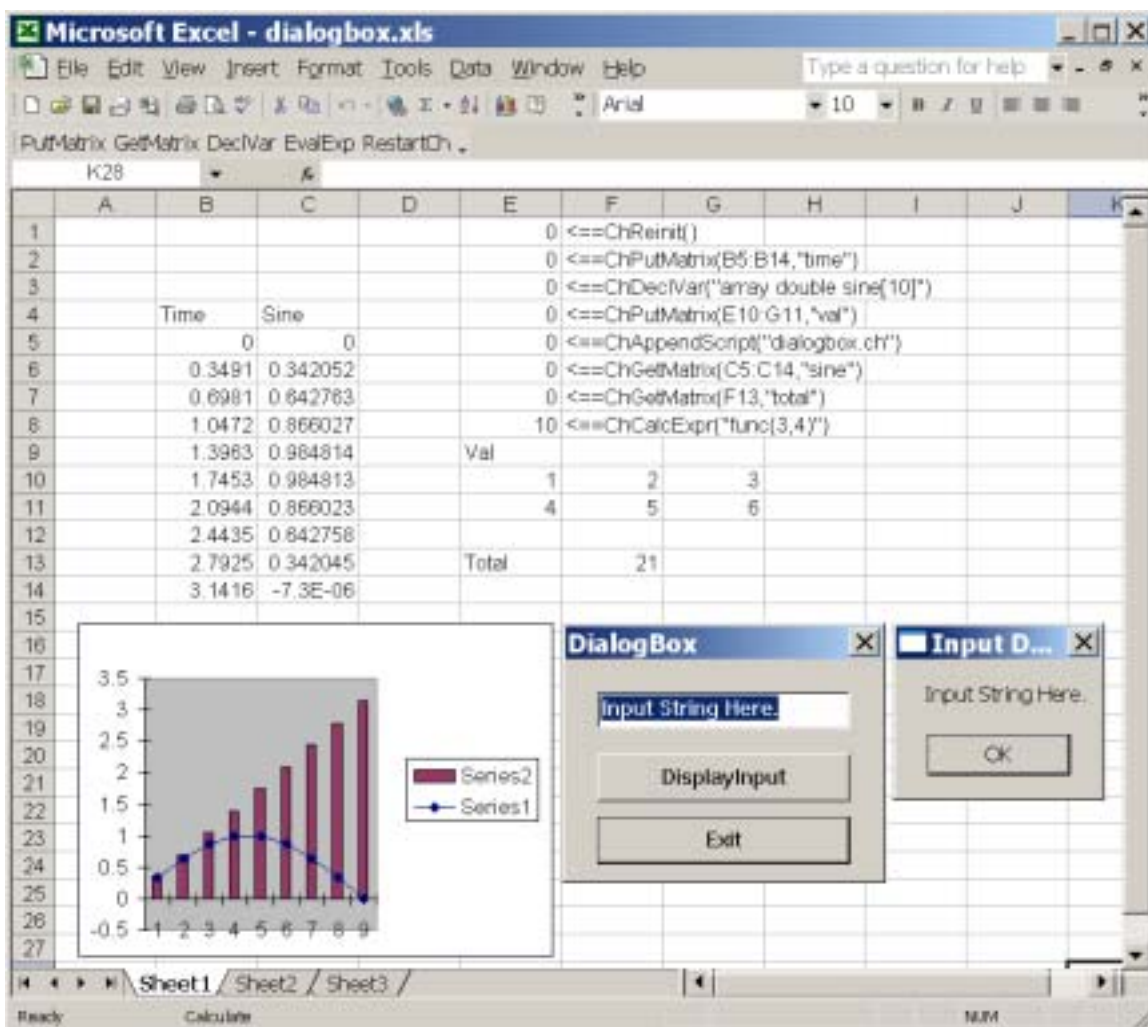


ChExcel User's Guide

Version 1.2



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - dialogbox.xls". The spreadsheet contains data in columns A and B, with headers "Time" and "Sine". The data points are as follows:

Time	Sine
0	0
0.3491	0.342052
0.6981	0.642763
1.0472	0.866027
1.3963	0.984814
1.7453	0.984813
2.0944	0.866023
2.4435	0.642758
2.7925	0.342045
3.1416	-7.3E-06

The spreadsheet also includes a "Total" row with values 1, 2, 3, 4, 5, 6, and 21. A chart is displayed in the bottom-left corner, showing two series: "Series2" (red bars) and "Series1" (blue line). The chart shows a sine wave pattern for Series1 and a bar chart for Series2. Two dialog boxes are overlaid on the spreadsheet: "DialogBox" and "Input D...". The "DialogBox" has an input field "Input String Here.", a "DisplayInput" button, and an "Exit" button. The "Input D..." dialog has an input field "Input String Here." and an "OK" button.

How to Contact SoftIntegration

Mail SoftIntegration, Inc.
 216 F Street, #68
 Davis, CA 95616
Phone + 1 530 297 7398
Fax + 1 530 297 7392
Web <http://www.softintegration.com>
Email info@softintegration.com

Copyright ©2001-2005 by SoftIntegration, Inc. All rights reserved.
Revision 1.2.0, October 2006

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SoftIntegration, Inc.

Ch, SoftIntegration, and One Language for All are either registered trademarks or trademarks of SoftIntegration, Inc. in the United States and/or other countries. Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, Windows XP, and Excel are trademarks of Microsoft Corporation. Other product or brand names are trademarks or registered trademarks of their respective holders.

Typographical Conventions

The following list defines and illustrates typographical conventions used as visual cues for specific elements of the text throughout this document.

- Interface components are window titles, button and icon names, menu names and selections, and other options that appear on the monitor screen or display. They are presented in boldface. A sequence of pointing and clicking with the mouse is presented by a sequence of boldface words.

Example: Click **OK**

Example: The sequence **Start**->**Programs**->**Ch5.0**->**Ch** indicates that you first select **Start**. Then select submenu **Programs** by pointing the mouse on **Programs**, followed by **Ch5.0**. Finally, select **Ch**.

- Keycaps, the labeling that appears on the keys of a keyboard, are enclosed in angle brackets. The label of a keycap is presented in typewriter-like typeface.

Example: Press <Enter>

- Key combination is a series of keys to be pressed simultaneously (unless otherwise indicated) to perform a single function. The label of the keycaps is presented in typewriter-like typeface.

Example: Press <Ctrl><Alt><Enter>

- Commands presented in lowercase boldface are for reference only and are not intended to be typed at that particular point in the discussion.

Example: “Use the **install** command to install...”

In contrast, commands presented in the typewriter-like typeface are intended to be typed as part of an instruction.

Example: “Type `install` to install the software in the current directory.”

- Command Syntax lines consist of a command and all its possible parameters. Commands are displayed in lowercase bold; variable parameters (those for which you substitute a value) are displayed in lowercase italics; constant parameters are displayed in lowercase bold. The brackets indicate items that are optional.

Example: `ls [-aAbcCdfFgilLmnopqrRstux1] [file ...]`

- Command lines consist of a command and may include one or more of the command’s possible parameters. Command lines are presented in the typewriter-like typeface.

Example: `ls /home/username`

- Screen text is a text that appears on the screen of your display or external monitor. It can be a system message, for example, or it can be a text that you are instructed to type as part of a command (referred to as a command line). Screen text is presented in the typewriter-like typeface.

Example: The following message appears on your screen

```
usage:  rm [-fiRr] file ...
```

```
ls [-aAbcCdfFgilLmnopqrRstux1] [file ... ]
```

- Function prototype consists of return type, function name, and arguments with data type and parameters. Keywords of the Ch language, typedefed names, and function names are presented in boldface. Parameters of the function arguments are presented in italics. The brackets indicate items that are optional.

Example: **double derivative**(**double** (**func*)(**double**), **double** *x*, ... [**double** **err*, **double** *h*]);

- Source code of programs is presented in the typewriter-like typeface.

Example: The program **hello.ch** with code

```
int main() {
    printf("Hello, world!\n");
}
```

will produce the output `Hello, world!` on the screen.

- Variables are symbols for which you substitute a value. They are presented in italics.

Example: module *n* (where *n* represents the memory module number)

- System Variables and System Filenames are presented in boldface.

Example: startup file **/home/username/.chrc** or **.chrc** in directory `/home/username` in Unix and **C:\ >_chrc** or **_chrc** in directory `C:\ >` in Windows.

- Identifiers declared in a program are presented in typewriter-like typeface when they are used inside a text.

Example: variable `var` is declared in the program.

- Directories are presented in typewriter-like typeface when they are used inside a text.

Example: Ch is installed in the directory `/usr/local/ch` in Unix and `C:/Ch` in Windows.

- Environment Variables are the system level variables. They are presented in boldface.

Example: Environment variable **PATH** contains the directory `/usr/ch`.

Table of Contents

Section	Page
1 Introduction	1
2 Installing ChExcel	1
3 Getting Started	3
3.1 Configure Excel to Work with ChExcel	3
3.2 The ChExcel Toolbar	3
3.2.1 PutMatrix	3
3.2.2 GetMatrix	4
3.2.3 DeclVar	4
3.2.4 EvalExpr	4
3.2.5 RestartCh	4
3.3 ChExcel Toolbar Examples	4
3.4 ChExcel Functions	7
3.4.1 Calling ChExcel Functions	8
3.4.2 Argument Passing Conventions	10
3.5 Data Management Functions	12
3.6 Toolbar Buttons vs. Functions	12
4 Variable Declaration Functions	12
4.1 ChDeclVar()	12
4.2 ChPutMatrix()	13
4.3 ChRemVar()	13
4.4 Variable Declaration Function Examples	13
5 Variable and Data Operation Functions	14
5.1 Functions Returning Scalar Values	14
5.1.1 ChCalcExpr()	14
5.1.2 ChFunc()	14
5.1.3 Examples of Functions Returning Scalar Values	14
5.2 Functions Returning Matrices	15
5.2.1 ChCalcExprMatrix()	15
5.2.2 ChFuncMatrix()	15
5.2.3 Examples of Functions Returning Matrices	15
5.3 Function ChGetMatrix() for Obtaining Both Scalar and Matrix Values from Variables	16
5.4 Function to Evaluate Both Scalar and Matrix Expressions	16
5.4.1 ChExprEval()	16
5.4.2 Expression Evaluation Function Examples	17
5.5 Function to Modify Matrices	17
5.5.1 ChAppendMatrix()	17
5.5.2 ChAppendMatrix Examples	17
5.6 Functions Running Ch Programs	18
5.6.1 ChRunScript()	18
5.6.2 ChAppendScript()	18
5.6.3 ChReinit()	18

5.6.4	Examples of Functions Running Ch Programs	19
5.7	Functions Capable of Plotting	24
5.7.1	Plotting Examples	24
5.8	User Defined Functions	26
5.8.1	User Defined Function Example	27
6	Using Win32 for Graphical User Interface	28
7	Programming in Visual Basic for Applications (VBA) with ChExcel	30
7.1	ChGetVar()	30
7.2	ChPutVar()	31
7.3	VBA Programming Examples	31
8	Examples	38
8.1	Example 1: Data Interpolation	38
9	References	45
10	Function Reference	45
10.1	ChExcel and VBA Functions	46
	ChAppendMatrix	47
	ChAppendScript	48
	ChCalcExpr	49
	ChCalcExprMatrix[VB]	50
	ChDeclVar	51
	ChExprEval	52
	ChFunc	53
	ChFuncMatrix[VB]	54
	ChGetMatrix[VB]	55
	ChPutMatrix	56
	ChReinit	57
	ChRemVar	58
	ChRunScript	59
10.2	VBA Functions	60
	ChPutVar	60
	ChGetVar	61
Index		62

1 Introduction

Note: The source code of Excel and Ch programs for all examples described in this document are available in CHEXCEL_HOME/demos where CHEXCEL_HOME is the home directory where ChExcel is installed. By default, the home directory for ChExcel is C:/Program Files/SoftIntegration/ChExcel. It is recommended that you try these examples while reading this document.

Although this documentation assumes that the user has experience in using both Microsoft Excel and Ch, C/C++ programmers without prior experience of using Excel and Ch shall be able to use ChExcel after reading this document. For more information on Ch and Excel, please refer to *Ch User's Guide*[1] and Microsoft's Excel documentation, respectively.

This documentation describes how ChExcel is designed and used. It also provides examples with detailed explanation to help you learn to develop your own ChExcel application programs.

What is ChExcel?

ChExcel is a Microsoft Excel add-in that embeds Ch (a C/C++ Interpreter) into Excel using Embedded Ch as shown in Figure 1. ChExcel allows users to access the scripting and computation power of Ch from Excel worksheets and Visual Basic for Applications (VBA). ChExcel lets you communicate and exchange data between Ch and Excel. With ChExcel, Excel spreadsheets can be manipulated through C/C++ scripts.

The ChExcel Enviroment

The ChExcel add-in allows users to communicate between Ch and Excel workspaces as shown in Figure 1. Excel is a user interface to Ch, allowing the user to utilize Ch statemetns, functions, programs, toolkits, and packages from Excel spreadsheets. The communication between Excel and Ch is controlled by a few simple ChExcel functions, keeping the interface simple and easy to use.

A Ch session is initialized for each Excel process. This means that variables can be shared between worksheets of the same workbook or even between workbooks in the same Excel workspace. If Ch is reinitialized any time during the Excel session all variables will be erased. Ch can be initialized by either calling the ChReinit() function or clicking the "RestartCh" button.

2 Installing ChExcel

This section describes installation of ChExcel as an Excel add-in.

System Requirements

ChExcel requires approximately 15 Mbyte disk space. It also requires the following software to be installed.

- Microsoft Windows 98/ME/NT/2000/XP
- Microsoft Excel 97 or above

Installing ChExcel

Make sure to install Excel before trying to install ChExcel by executing the downloaded ChExcel toolkit such as chexcel-1.0.0.exe.

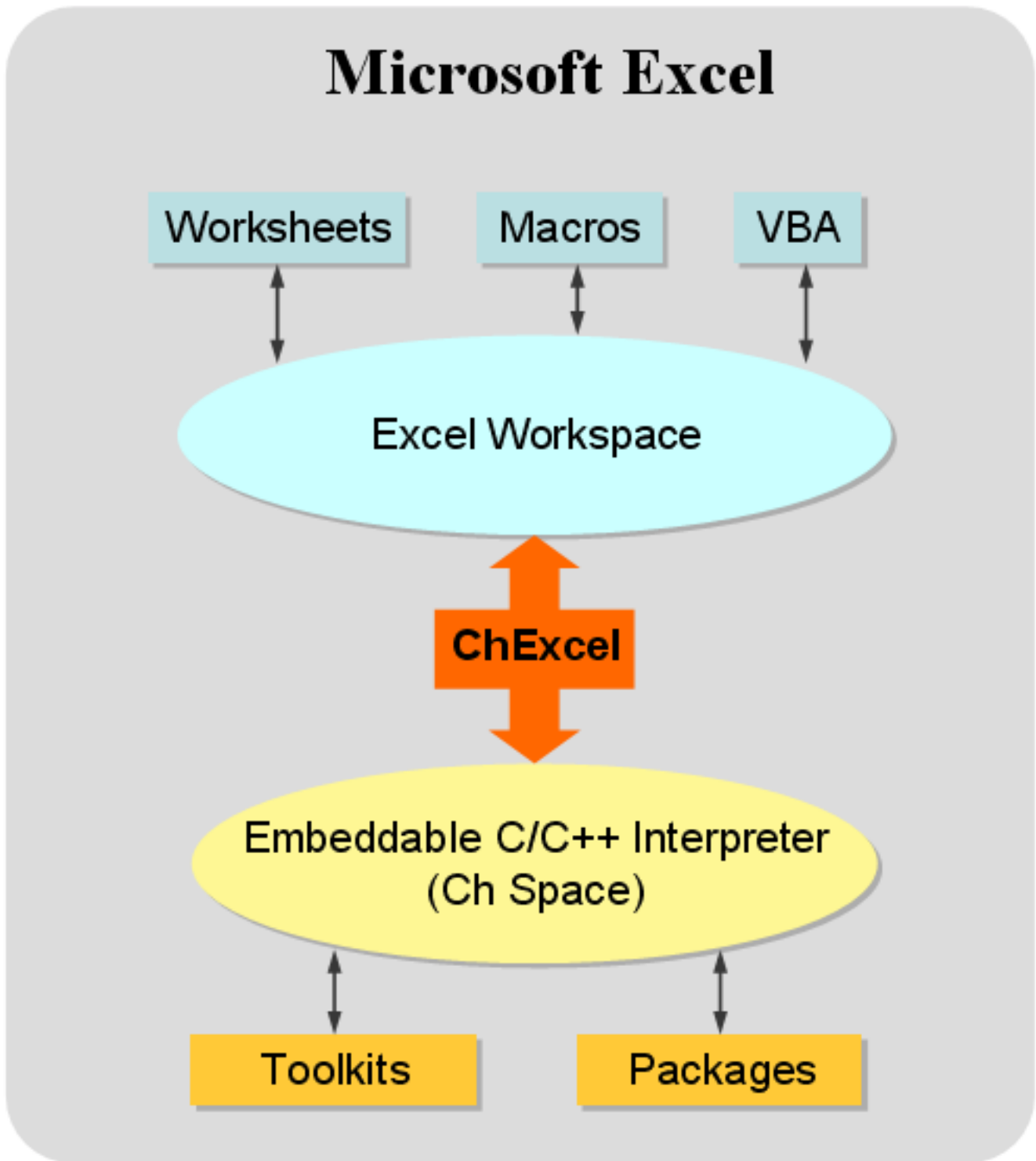


Figure 1. Embedded Ch inside Microsoft Excel through ChExcel.

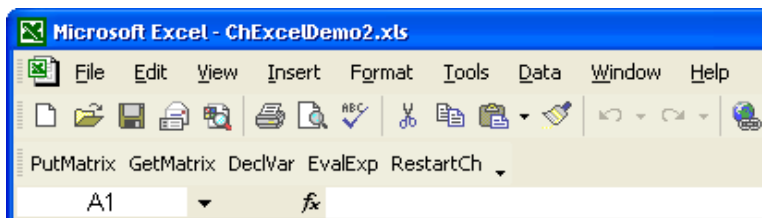


Figure 2. The ChExcel toolbar.

3 Getting Started

3.1 Configure Excel to Work with ChExcel

To configure Excel to work with ChExcel follow the steps below:

1. Start Microsoft Excel.
2. Select Tools from the menu toolbar, click Add-Ins.
3. In the Add-In window check the ChExcel box and click OK.
4. The ChExcel toolbar should pop up on you Excel Taskbar as illustrated in Figure2.

There are two ways to try ChExcel.

1. To use ChExcel from ChExcel Toolbar in Excel, follow instructions in sections 3.2 and 3.3.
2. To try ChExcel functions inside Excel spreadsheets described in the remaining sections, run spreadsheets such as `plot.xls` located in `CHEXCEL_HOME/demos` directory where `CHEXCEL_HOME` is the home directory for ChExcel such as `C:/Program Files/SoftIntegration/ChExcel`.

3.2 The ChExcel Toolbar

The ChExcel toolbar provides the user with fast access to commonly used ChExcel macro operations. There are five buttons located on the toolbar: `PutMatrix`, `GetMatrix`, `DeclVar`, `EvalExp`, and `RestartCh` as shown in Figure 2.

3.2.1 PutMatrix

The first toolbar button is `PutMatrix`, which allows the user to put a matrix from the Excel environment into a Ch variable. A matrix can be either a one or two dimensional array in Ch. To use the button, first select the cells that contain the data, then click the `PutMatrix` button. When the window opens, enter the name of the variable you would like the data to be stored in, then click OK. This will store the data from the spreadsheet into a Ch variable of type array double if a number of cells were selected, or into a Ch variable of type double if a single cell was selected. If worksheet cells in a single row or column are selected, a one dimensional array in Ch will be declared and initialized with the values in the worksheet cells. If a variable of the specified name already exists in Ch, it will be first removed, then declared and initialized.

3.2.2 GetMatrix

The button GetMatrix, lets the user retrieve a matrix or a single value from Ch and place it on the Excel spreadsheet. Before using the button, first select a range where you would like the data to be placed. If the variable is a one dimensional array it can be placed into the worksheet as either a row or a column. To specify row format select a single row. To specify column format select a single column. If the range selected is smaller than the dimensions of the data, the the data will overflow into the adjacet cells. If the range selected is larger than the dimensions of the data, the data will only fill the required region. After selecting the range, press the GetMatrix button and enter the Ch variable you would like to get, and click OK.

3.2.3 DeclVar

The third button, DeclVar, allows the user to declare a variable in Ch syntax. The user can declare a variable of any type and dimension. To declare an array of integers, simply type, `"array int MyArray[4][5]"` in the area provided in the window. To declare a character string type, `"char * MyString"`.

3.2.4 EvalExpr

The fourth ChExcel toolbar button, EvalExpr, allows the user to evaluate a Ch expression. When the button is pressed, the user can enter any valid Ch expression to pass to Ch. When entering an expression, make sure that the variables being used have been declared beforehand.

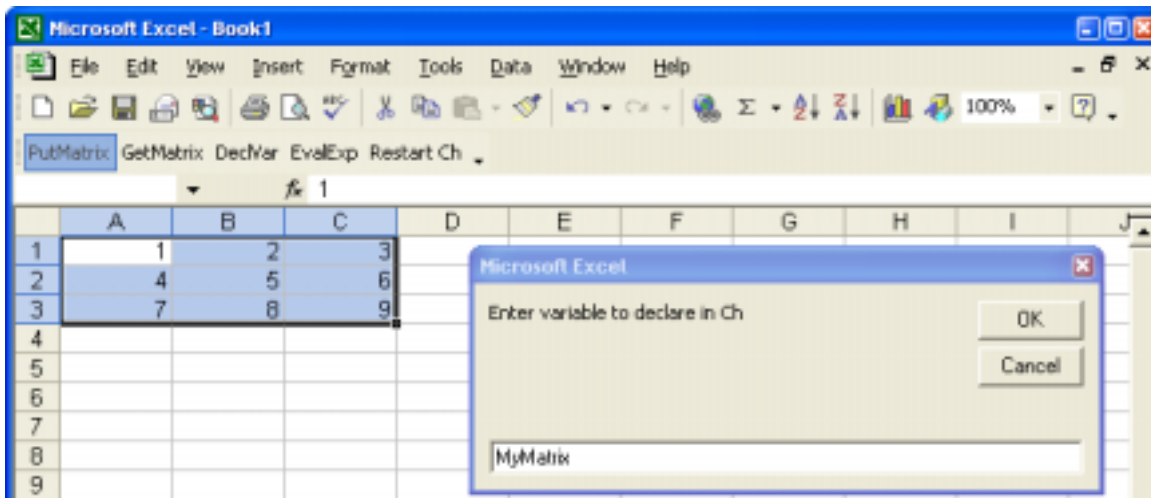
3.2.5 RestartCh

The RestartCh button, allows the user to refresh the Ch session. This erases all variables and begins a new Ch session. Be careful not to push this by accident because you cannot recover work from the previous Ch session.

3.3 ChExcel Toolbar Examples

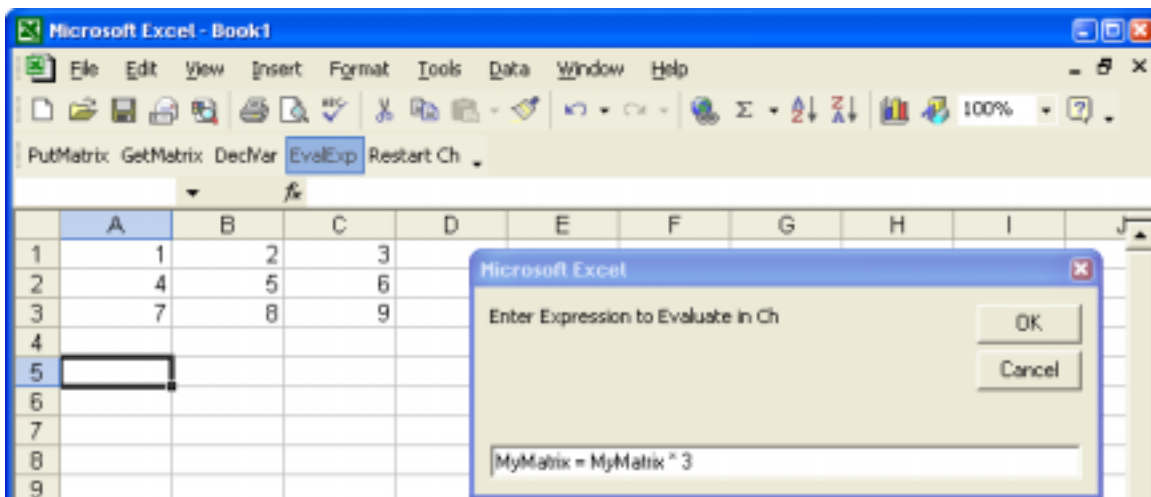
Put data into Ch Variable

1. Highlight the cells you want to initialize the variable with.
2. Click the **PutMatrix** button.
3. Type in the variable name and click OK.



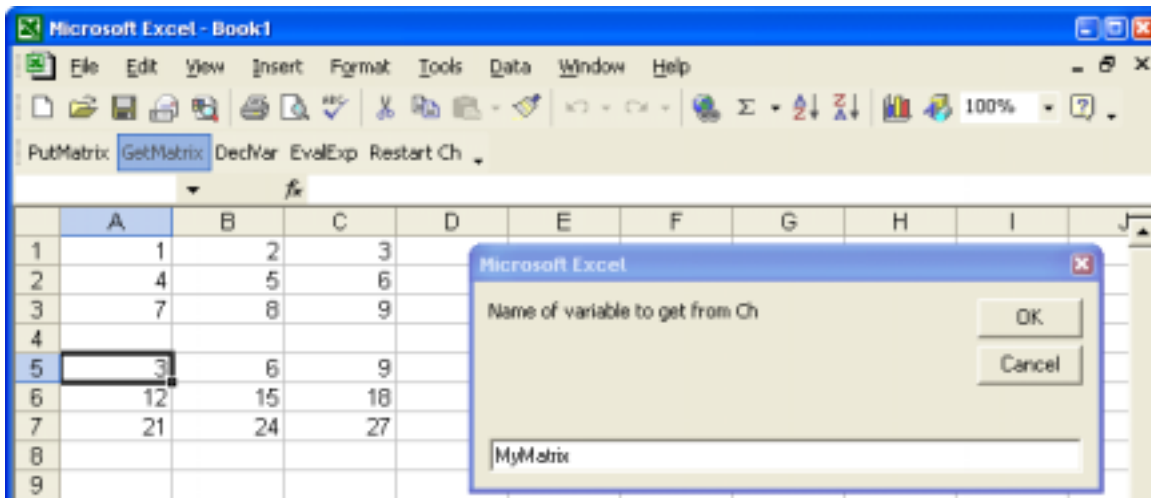
Manipulate Ch variable

1. Click the **EvalExp** button.
2. Type in Ch Expression and click OK.



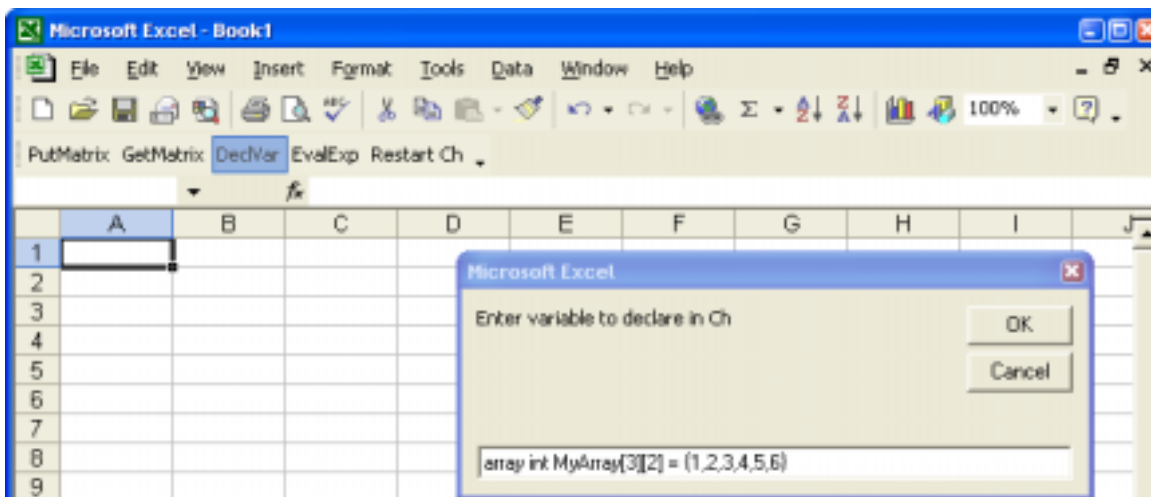
Retrieve data from Ch

1. Click the cell or select the range you would like the data to be put.
2. Click the **GetMatrix** Button.
3. Type in variable name and click OK.



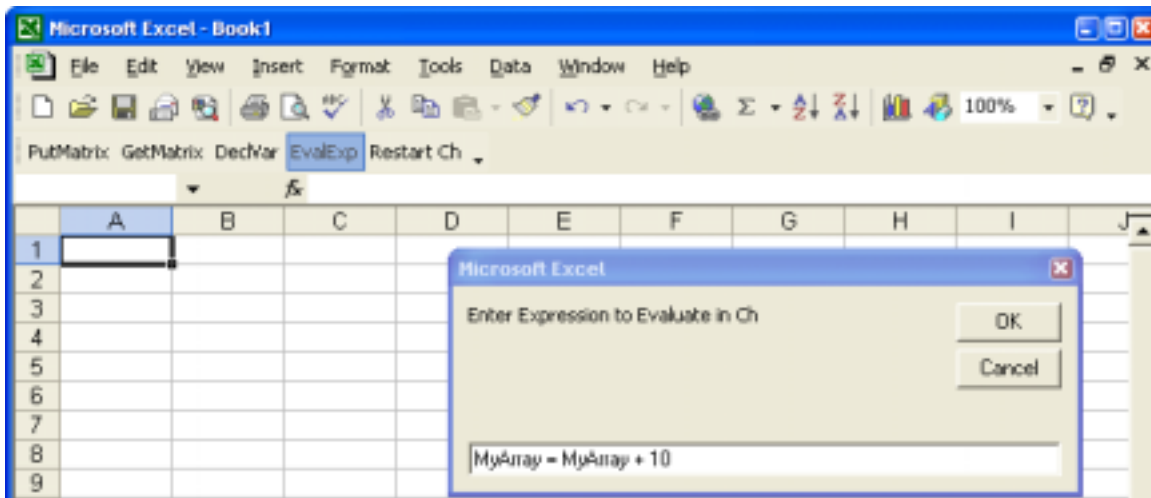
Declare and initialize Ch variable

1. Click the **DeclVar** button.
2. Type in variable declaration and click OK.



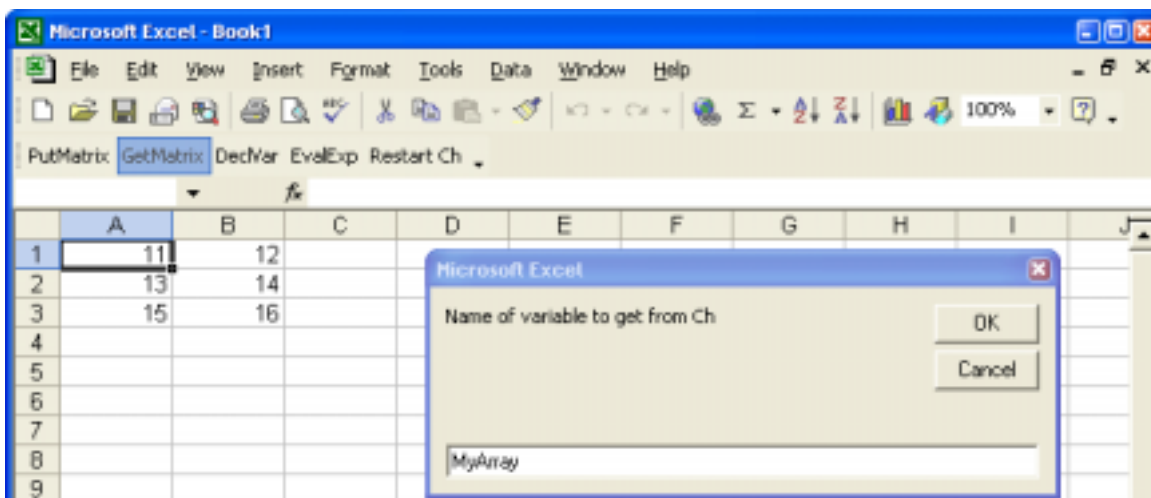
Manipulate Ch variable

1. Click the **EvalExpr** button.
2. Type in a Ch expression and click OK.



Retrieve data from Ch

1. Click the cell or select the range you would like the data to be put.
2. Click the **GetMatrix** Button.
3. Type in variable name and click OK.



3.4 ChExcel Functions

With ChExcel installed according to the preceding instructions, Ch will automatically be initialized when Excel is started. There are two types of ChExcel functions. The first is the Link Management function. They help control, initialize, and end Excel connections to Ch. The "RestartCh" button located in the ChExcel toolbar is the only link management function provided with ChExcel because there is the overhead for running Ch in Excel is very small.

The second type of function is the Data Management function. This collection of functions allows the user to place data from Excel into Ch, manipulate the data within Ch, and retrieve the data from Ch back onto the Excel spreadsheet. These functions will be discussed in detail in the next section.

3.4.1 Calling ChExcel Functions

ChExcel functions can be placed into a worksheet cell in two ways. The first is to use the function wizard. To enter a function using this method, click on the cell you want to function to be, then go to the menu and click on Insert and Function. In the function wizard select the User Defined category. Click on the function you want and the function will pop-up. Fill in the arguments and click OK.

For example, Figure 3 shows the Excel worksheet function menu. This menu lists all the worksheet functions available to the user. ChExcel functions are located in the User Defined section. Figure4 shows the function wizard for ChFunc () The function wizard lets the user input arguments one at a time. Figure5 shows the result of running the function wizard. In this figure, cell A1 has been assigned the function ChFunc ("pow" , 2 , 3).

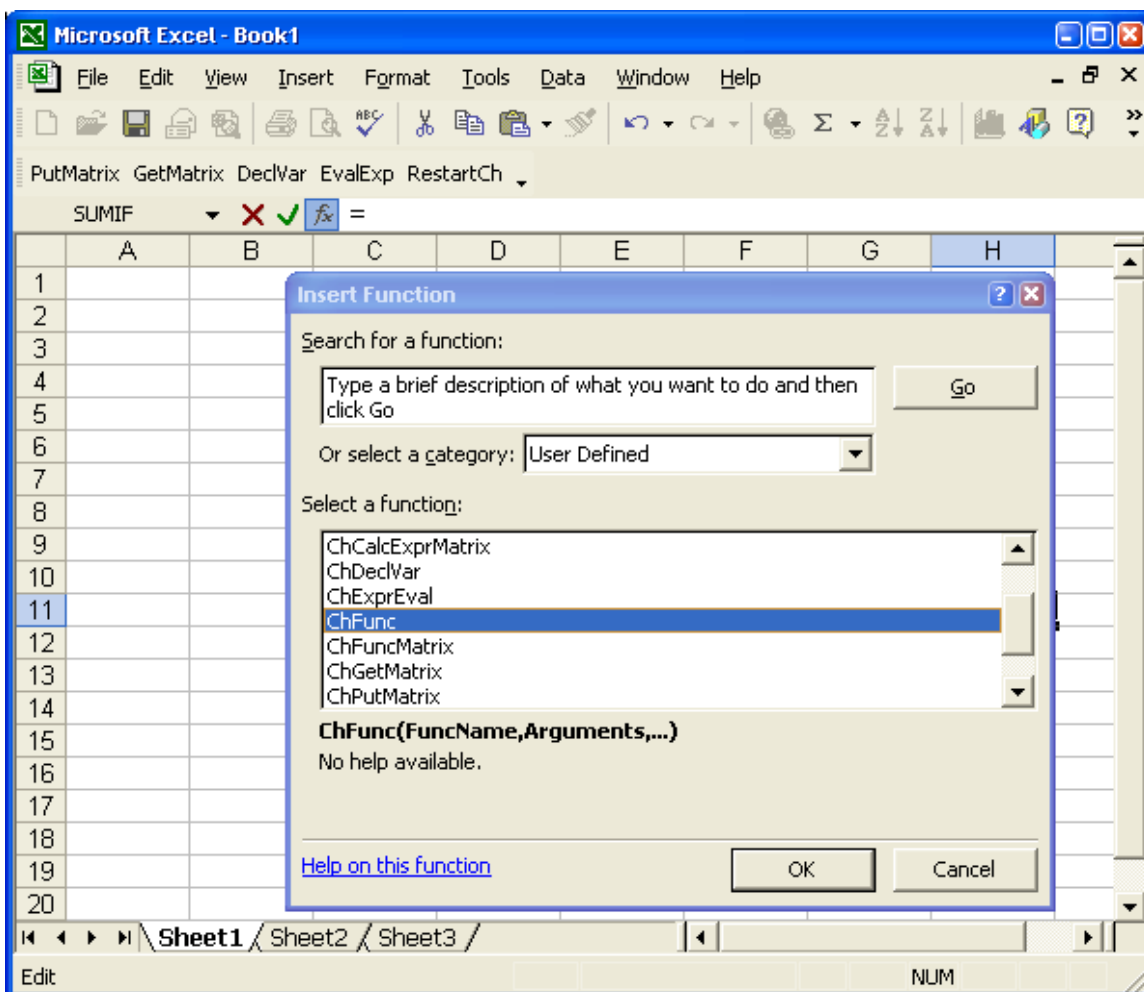


Figure 3. The Excel function menu.

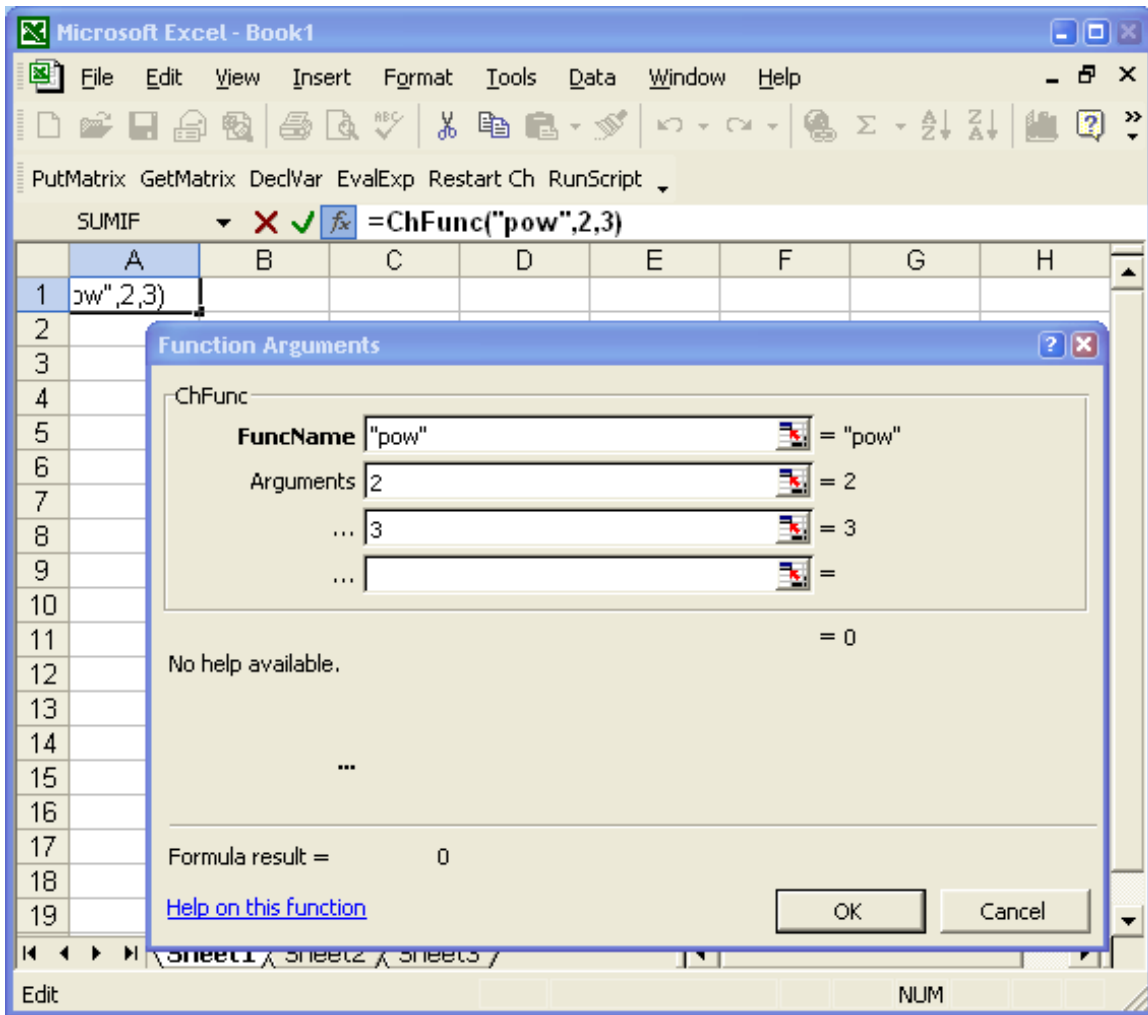


Figure 4. The ChFunc function wizard.

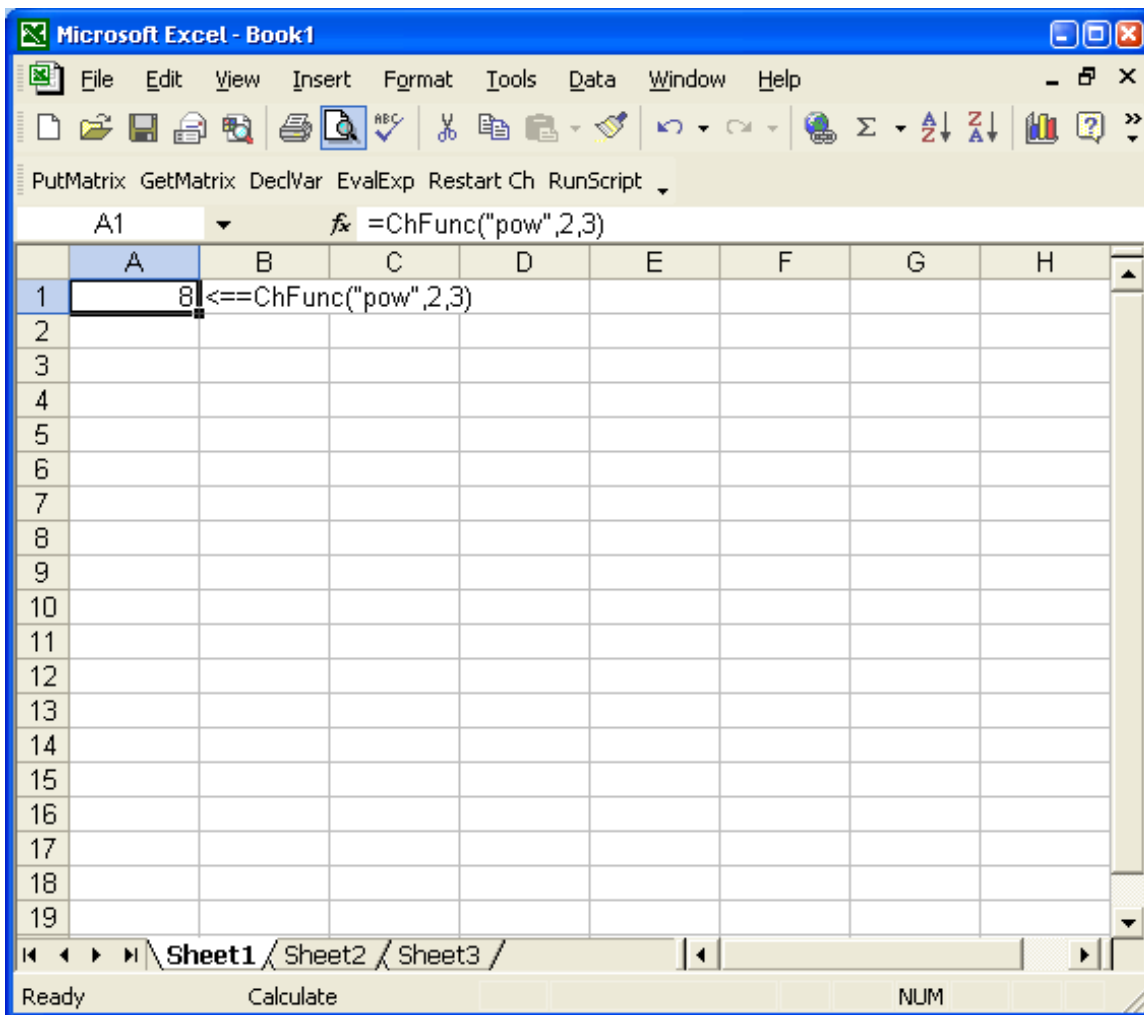


Figure 5. Result of the function wizard.

The second method of entering a ChExcel function into the worksheet is to type it in by hand as shown in Figure 5. To use this method, go to the cell you want the function to be located. To indicate that you are entering a function type "=" and enter the function and arguments. In the examples provided, all function cells have a comment cell located to its right. The comment cell starts with "<==" and contains the exact text of the function. To execute the function in a cell, move the cursor to the cell and then press function key F2.

3.4.2 Argument Passing Conventions

ChExcel functions recognize four argument passing conventions illustrated in Figure 6. Details of the ChExcel functions used in the spreadsheet in Figure 6 will be described in two sections 4.2. and 5.3.

The first convention is for passing ranges, range names and numeric values. You do not have to inclose ranges, range names, or numeric values in double quotes. To pass the range A1:B2 to `ChPutMatrix()` type A1:B2 directly into the field as follows, `ChPutMatrix(A1:B2, "a")`. To pass an Excel range named "MyRange" to `ChPutMatrix()` enter `ChPutMatrix(MyRange, "x")`. A value from a drop-list in a cell such as cell B4 in Figure 6 can also be obtained. To pass numerical values, also type them directly into the function field, like in `ChFunc("pow", 2, 3)`, which is the equivalent of `pow(2, 3)` in

C.

The second conversion is for passing variable names and function names. To pass a variable name or function name to ChExcel, simply enclose it in a pair of double quotes. For example, to get the values for variable `a` from Ch and place them in cells A11:B12, call the ChExcel function `ChGetMatrix(A11:B12, "a")` with `a` in double quotes.

The third convention is for passing strings. If you pass all arguments as a single string, you must enclose all embedded strings in two sets of double quotes. For example, `ChExprEval()` takes a single string arguments. If you want to set the string variable `s`, with type `string_t` or pointer to `char`, to the environment variable `CHHOME`, then you must enclose `CHHOME` in double quotes. The resulting call would be `ChExprEval("s = getenv("CHHOME")")`, which is the equivalent of `s = getenv("HOME")` in Ch.

The last calling convention is for passing arguments as separate strings. For the function `ChFunc` and `ChFuncMatrix`, each argument is passed into ChExcel as separate strings. If you want to pass the plot name `ChPlot` to the function `plotxy()`, then you would need to enclose `ChPlot` in three sets of double quotes. The resulting function will be `ChFunc("plotxy", "x", "sin(x)+2*xy", ""ChPlot"", ""xlabel"", ""ylabel"")`, which is equivalent to `plotxy(x, sin(x)+2*x, "ChPlot", "xlabel", "ylabel")` in Ch.

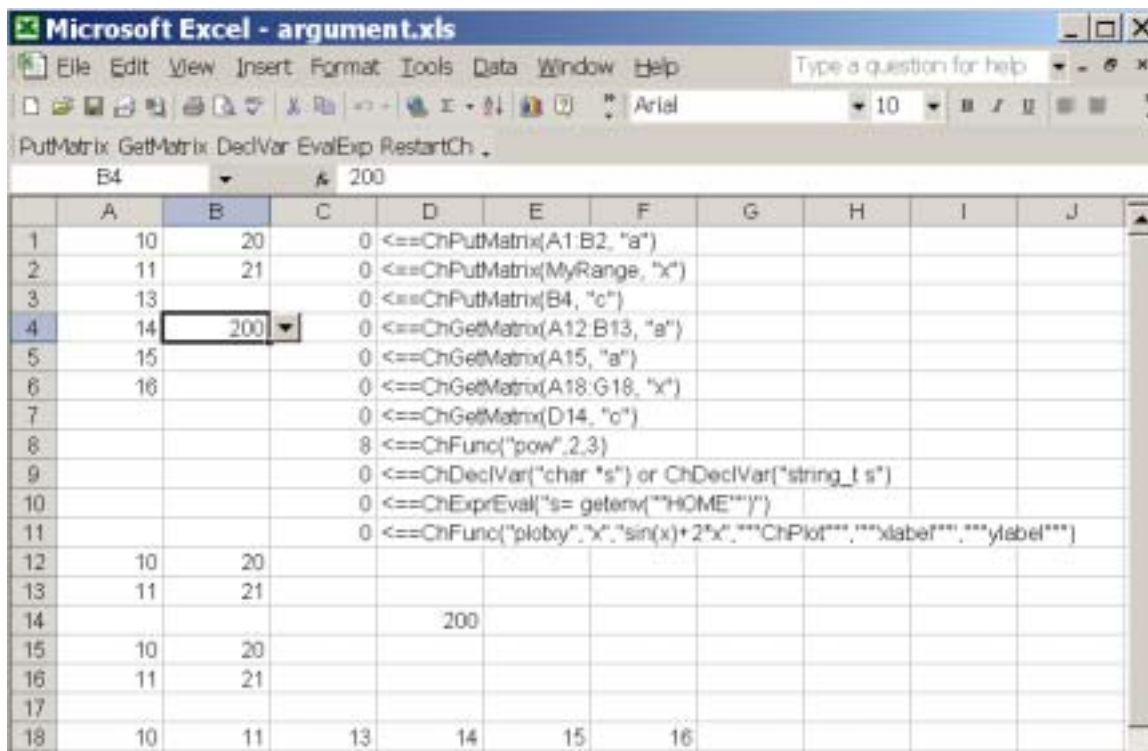


Figure 6. Example of argument passing conventions.

3.5 Data Management Functions

Functions below can be used for handle data. These functions will be described in detail in sections 4 and 5. Functions **ChGetVar()** and **ChPutVar()** are only available in VBA.

Function	Description
ChAppendMatrix()	Append data to a Ch matrix variable.
ChAppendScript()	Append script to Ch session.
ChCalcExpr()	Returns value of scalar Ch Expression.
ChCalcExprMatrix[VB]()	Returns value of matrix Ch Expression.
ChDeclVar()	Declare a Ch variable.
ChExprEval()	Evaluate a Ch expression in Ch.
ChFunc()	Evaluate a Ch function with return of scalar.
ChFuncMatrix[VB]()	Evaluate a Ch function with return of array.
ChGetMatrix[VB]()	Get scalar or matrix value of Ch variable and place in Excel.
ChGetVar()	Get value of variable from Ch and place in VBA.
ChPutMatrix()	Put Excel matrix or scalar value into Ch variable.
ChPutVar()	Put value of a VBA variable into Ch variable.
ChRemVar()	Remove variable from Ch.
ChReinit()	Reinitialize Ch session.
ChRunScript()	Run script in Ch session.

3.6 Toolbar Buttons vs. Functions

When using ChExcel, one might wonder what is the difference between using the toolbar buttons and the ChExcel worksheet functions? To answer this question, we must take a closer look at the Excel environment. In Excel there are two types of procedures: macros and functions. A macro can accept passed arguments, but cannot return a value or be associated with a worksheet cell. A function can accept arguments, be associated with a worksheet cell and return a value. The difference between the ChExcel toolbar buttons and ChExcel functions is that the ChExcel toolbar buttons actually call macros and therefore cannot return values, but only modify worksheet cells.

In general, if you want to save your work, you should use worksheet functions because they can be invoked from worksheet cells. To retrieve the result the next time you open your Excel workbook, simply recalculate the formulas in the correct order by pressing F2 and Enter on each cell. A macro can be used for fast calculations and function calls, but once invoked only their results can be saved in the workbook.

4 Variable Declaration Functions

4.1 ChDeclVar()

The function ChDeclVar() is a worksheet function that declares a variable in Ch. This function takes a string as an argument and will return 0 for success or #ERROR# on failure. The string argument must be a single

Ch variable declaration. The declaration can be of any data type. The variable can also be initialize at this time. Make sure that you only declare one variable per ChDeclVar() function. If you redeclare a variable name it will overwrite the previous variable declaration, so be sure not to redefine variables by accident.

4.2 ChPutMatrix()

The function ChPutMatrix() places data from the Excel environment into a Ch variable. The function takes two arguments, a range and a variable name. It returns 0 on success and #ERROR# on error. If a range is passed to ChPutMatrix(), the data will be placed into a Ch variable of array double type with dimensions corresponding to the Excel matrix. If a single cell is selected, the data will be placed into a Ch variable of double type. If the variable name passed to ChPutMatrix() has been previously declared, the original variable will be overwritten.

In Figure 6, the values in cells A1:B2 is placed in variable a by function call ChPutMatrix(A1:B2, "a"), which is equivalent to the following declaration.

```
array double a[2][2]={10, 20,
                    11, 21};
```

The Excel variable MyRange contains A1:A6. Therefore, the values for cells A1:A6 are placed in variable x of array type. The value 200 in drop-list in cell B4 is placed in variable c of double type by function call ChPutMatrix(B4, "c"), which is equivalent to the following declaration.

```
double c = 200;
```

4.3 ChRemVar()

The function ChRemVar() removes a variable from the Ch session. ChRemVar() takes a variable name as an argument. The function returns 0 if the variable is removed successfully, or #ERROR# on failure.

4.4 Variable Declaration Function Examples

Cell A1 in Figure 7 contains the ChDeclVar() function. It declares the variable i of int type. Variables can also be declared with initialization. Cell A2 declares and initializes an integer computational array variable a. The cell A7 contains the ChPutMatrix function. It takes the data from cells A1 through C5 and places them in the Ch double array variable b. The cell A9 contains the ChRemVar function. This cell will remove the variable a from the Ch session.

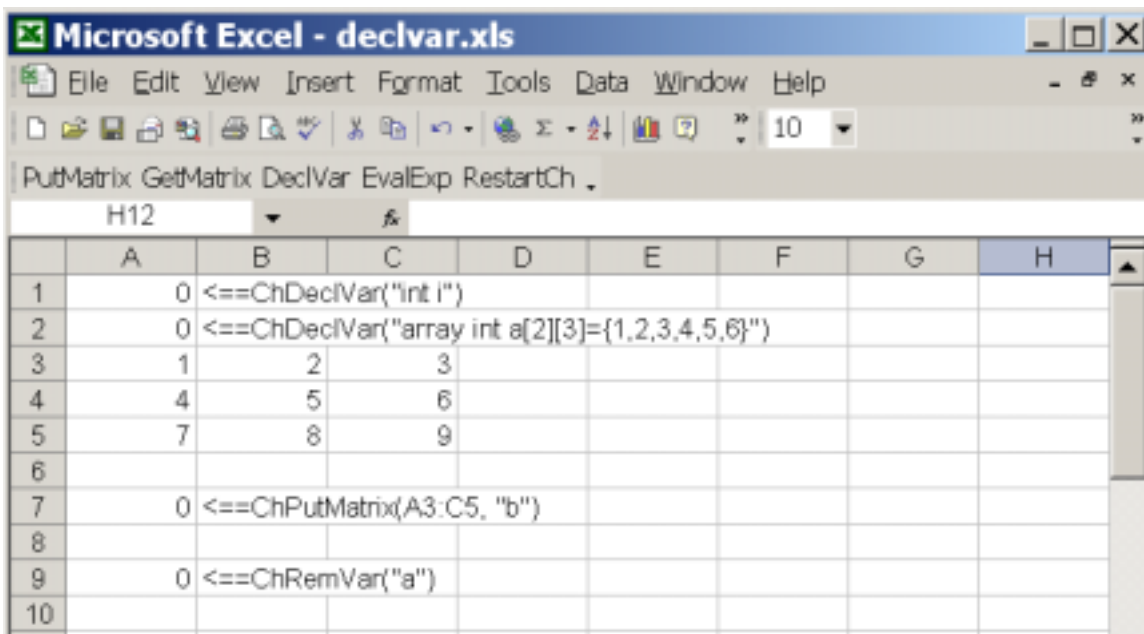


Figure 7. Variable declaration.

5 Variable and Data Operation Functions

5.1 Functions Returning Scalar Values

5.1.1 ChCalcExpr()

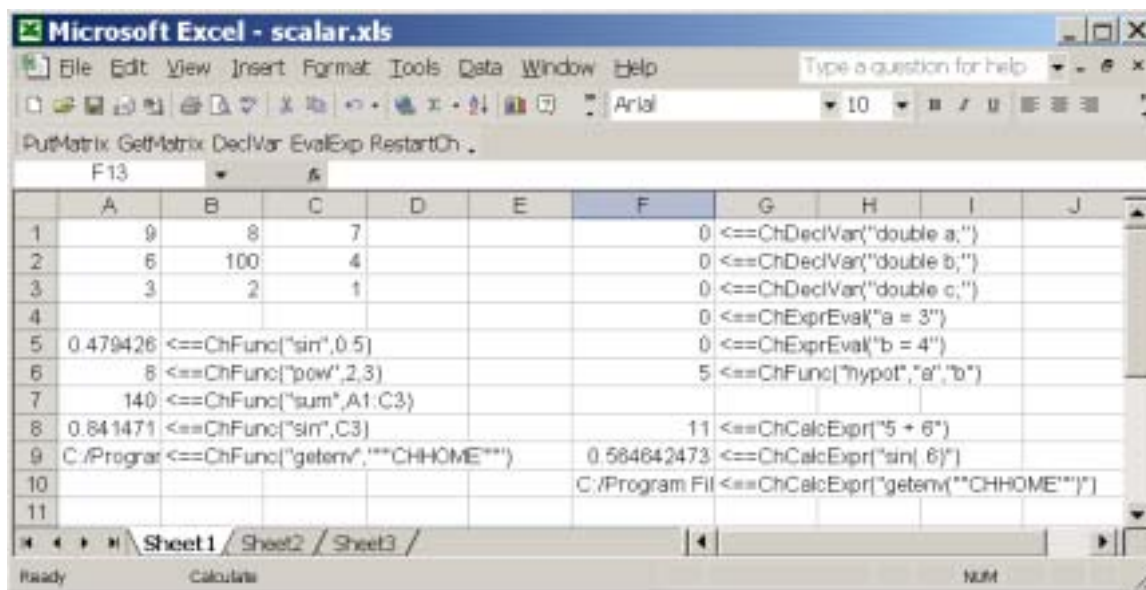
ChCalcExpr() calculates a Ch expression of scalar type and returns the value of the expression. It takes a string as an argument and can return a single value of data type double, float, short, char, int, or string. ChCalcExpr() will return the value of the expression on success and #ERROR# on failure.

5.1.2 ChFunc()

The function ChFunc() makes a call to a Ch function and returns the value of the called function. ChFunc() takes a function name and variable number of parameters as arguments. The function passed to ChFunc() must have a return value of double, int, float, char, short, or string. This function can return a single value of data type double, int, float, char, short, or string. ChFunc() will return the value of the function on success and #ERROR# on failure.

5.1.3 Examples of Functions Returning Scalar Values

The cells A5, A6 and A9 in Figure 8 evaluate Ch functions given numeric and string arguments. Cells A7 and A8 evaluate Ch functions given Excel ranges. Cell F6 evaluates the Ch function hypot(), which returns the hypotenues of a triangle given two lengths. Cell 7 adds values in the range of A1 to C3 using Ch numerical function sum(). Cells F8 through F10 give examples of ChCalcExpr(). Cell F8 returns an integer, cell F9 returns a double, and F10 returns a string.



5.2 Functions Returning Matrices

5.2.1 ChCalcExprMatrix()

ChCalcExprMatrix() evaluates an expression returning a matrix, and places the matrix on the spreadsheet. The function takes two arguments, a Ch expression and a range. The matrix can be either a one or two dimensional array in Ch. The expression is evaluated and the result is placed in the specified range. The specified range must have the exact same dimensions as the result or the data will be incorrectly formatted. Make sure not to include the function cell into the range that you pass to ChCalcExprMatrix(), or the function will be overwritten. This function returns 0 on success and #ERROR# on failure.

5.2.2 ChFuncMatrix()

ChFuncMatrix() calls a Ch function and returns the resulting matrix. The function takes a Ch function name, range, and a variable number of parameters as arguments. The Ch function passed to ChFuncMatrix() must return a matrix of double or integer type. ChFuncMatrix() will evaluate the function and place the returned matrix into the specified range. The specified range must have the exact same dimensions as the result or the data will be incorrectly formatted. Make sure not to include the function cell into the range that you pass to ChFuncMatrix(), or the function will be overwritten. This function returns 0 on success and #ERROR# on failure.

5.2.3 Examples of Functions Returning Matrices

Cell A7 in Figure 9 calculates the addition of two matrices, r and p , and returns the result to cells A9 through C11. Cell A14 calls the Ch function `inverse()` and returns the inverse of the range A1:B2 to A16:B17. Cell F10 returns the matrix, resulting from the function `transpose()` for cells A1 through C2 to F12 through G14.

5 VARIABLE AND DATA OPERATION FUNCTIONS

5.3 Function ChGetMatrix() for Obtaining Both Scalar and Matrix Values from Variables

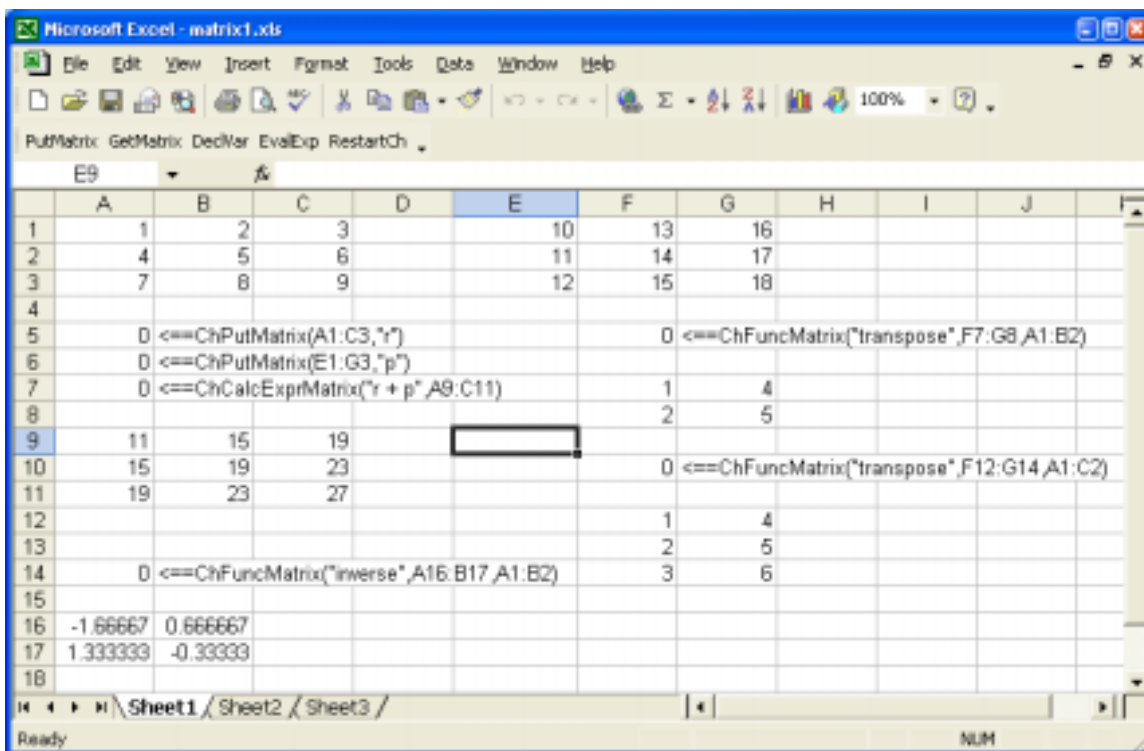


Figure 9. Function returning matrices.

5.3 Function ChGetMatrix() for Obtaining Both Scalar and Matrix Values from Variables

ChGetMatrix() retrieves the value of a Ch variable and places it into the Excel worksheet as shown in Figure 6. The function takes two arguments, a range and a variable name. The Ch variable will be placed in the specified range. If the variable is a scalar, it will be placed in the specified cell in the first argument. If the variable is a one dimensional array it can be placed into the worksheet as either a row or a column. To specify row format select a single row range. To specify column format select a single column range. If the Ch matrix is smaller than the specified range, it will only fill the required range. If the Ch matrix is larger than the specified range, it will overflow into the adjacent cells. Make sure not to include the function cell into the range that you pass to ChGetMatrix(), or the function will be overwritten. This function returns 0 on success and #ERROR# on failure.

In Figure 6, the values for two-dimensional array a are placed in cells A12:B13 and A15:B16. The values for one-dimensional array x is placed in cells A18:F16. The value for scalar c is placed in cell D14.

5.4 Function to Evaluate Both Scalar and Matrix Expressions

5.4.1 ChExprEval()

ChExprEval() evaluates a Ch expression. It takes a single expression as an argument, and returns 0 on success and #ERROR# on failure. This function can be used to initialize variables, modify variables, call functions, and modify the Ch session.

5.4.2 Expression Evaluation Function Examples

The example in Figure 10 illustrates modifying Ch variables using the ChExcel function ChExprEval(). Cell A5 declares the variable a, and initializes it to the data in cells A1 through C3. The function ChGetMatrix() in cell A6 adds 5 to each element in the matrix a. The value of a is placed onto the spreadsheet using ChGetMatrix(). Cell E6 evaluates the expression "a = transpose(a)" and sets the value of a equal to its transpose. The result is then returned to the spreadsheet by ChGetMatrix().

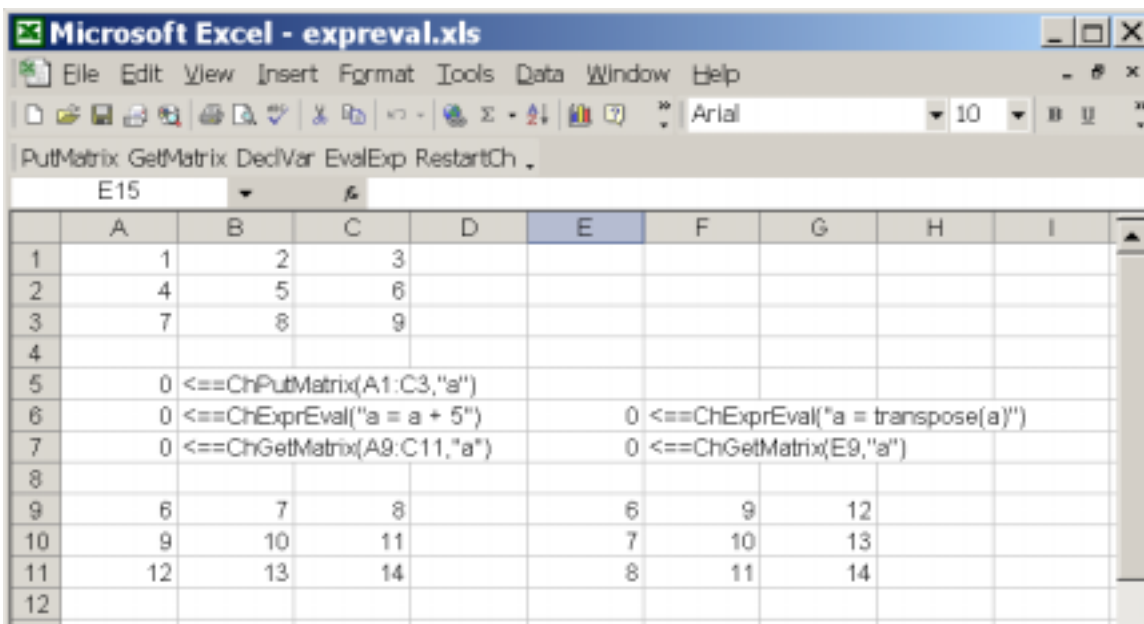


Figure 10. Expression evaluation.

5.5 Function to Modify Matrices

5.5.1 ChAppendMatrix()

The function ChAppendMatrix() lets the user append data to Ch matrices. This function takes a Ch variable name, a range, and 0 or 1 as arguments. The last arguments specify where on the Ch variable the range will be appended: 0 for horizontal or 1 for vertical. ChAppendMatrix() will takes the range and append it to the right or to the bottom of the Ch variable. If you want to append a range to the right of a Ch variable make sure the number of rows match. If you want to append a range to the bottom of a Ch matrix, make sure the number of columns match. This function returns 0 on success and #ERROR# on failure.

5.5.2 ChAppendMatrix Examples

The cell A6 in Figure 11 appends the range E1 to F3 to the right of the Ch variable r. The result is placed on the spreadsheet using the function ChGetMatrix(). The cell G6 appends the range E1 to G2 to the bottom of the Ch variable p.

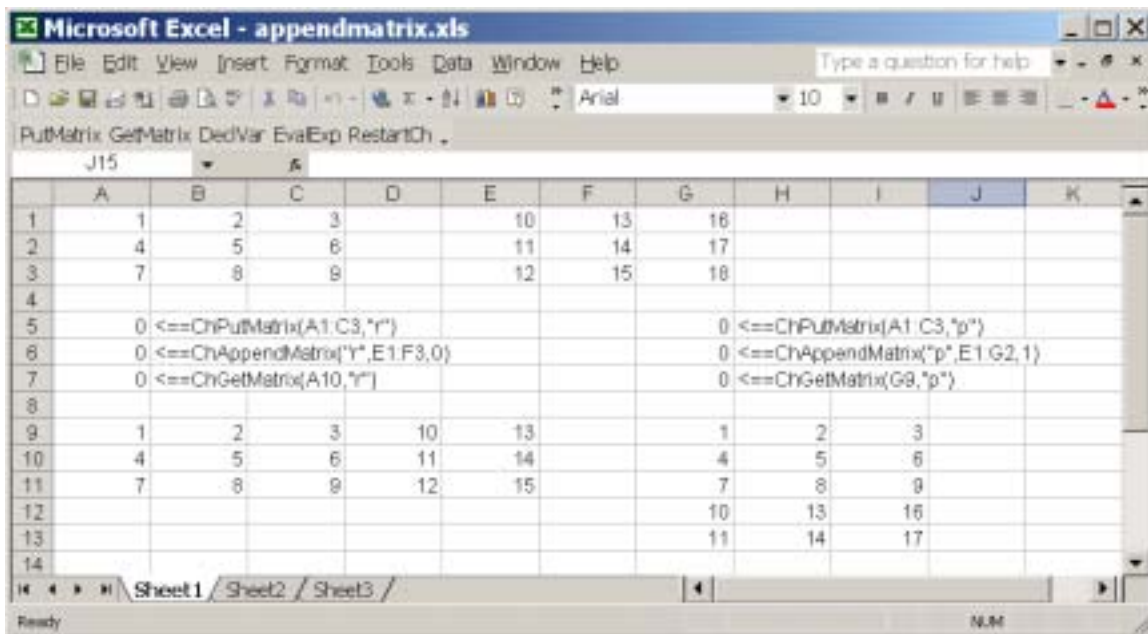


Figure 11. Matrix modification.

5.6 Functions Running Ch Programs

5.6.1 ChRunScript()

The function ChRunScript(), allows the user to run a Ch program. This function reinitializes the Ch session so all variables in the current session are erased. The user will input the name of the program and ChExcel will execute it from the function main() if it exists. Any global variable declared in the program will then be accessible to ChExcel. Also, any functions that have been declared can now be used by the ChExcel. The program must reside in Ch's program path specified by the Ch variable `_path` or the user must enter a complete path to the program. This function is best used to begin a ChExcel session to import variables from Ch to Excel. An example of a program run with ChRunScript() in ChExcel is provided in section 5.6.4.

5.6.2 ChAppendScript()

The function AppendScript, allows the user to run a Ch script in the current session. The user will input the name of the script and ChExcel will execute it. Any global variable declared in the script will then be accessible to ChExcel. Also, any functions that have been declared can now be used by the ChExcel. Ch variables cannot be redeclared in the Ch script so make sure not to use variable names that currently exist in ChExcel. The script must reside in Ch's program path specified by the Ch variable `_path` or the user must enter a complete path to the script. This function is best used to transfer variables to and from Ch and Excel. An example of a program run with ChAppendScript() in ChExcel is provided in section 5.6.4.

5.6.3 ChReinit()

The function ChReinit() reinitializes the Ch environment. When Ch is reinitialized all the variables from the previous session are erased. This function takes no arguments and returns 0 for success and #ERROR# for failure.

5.6.4 Examples of Functions Running Ch Programs

The Ch script in Program 1 consists of four global variables and two functions. The function ChRunScript() will begin running in the function main(). The function main() modifies the global variables and produces a plot.

```
#include <numeric.h>
#include <array.h>

double total;
array double val[2][3];
array double time[10], sine[10];

int main()
{
    linspace(val, 1, 6);
    total = sum(val);
    linspace(time, 0, M_PI);
    sine = sin(time);
    plotxy(time,sine,"Sin Plot", "time", "sin");
    return 0;
}

double func(double x, double y)
{
    return 2*hypot(x,y);
}
```

Program 1. Script for ChRunScript (runscript.ch).

Figure 12 shows the Excel worksheet runscript.xls. Cell E1 contains the ChRunScript() function that runs the program runscript.ch. Cells E2 through E5 use the ChExcel function ChGetMatrix() to retrieve data from Ch and place them on the Excel spreadsheet. Cell E6 calls the function func() defined in the program runscript.ch. Figure 13 shows the worksheet runscript.xls after all the functions have been evaluated. Figure 14 shows the plot generated by runscript.ch.

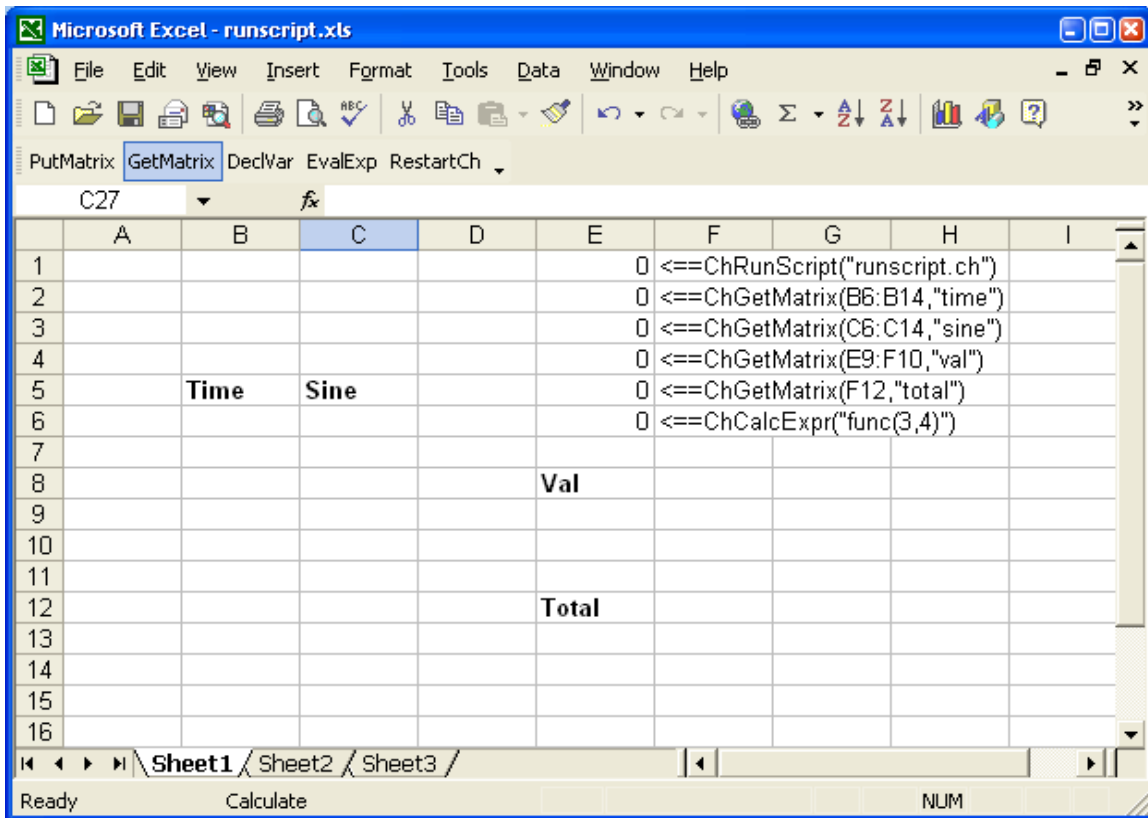


Figure 12. Before evaluation of ChRunScript.

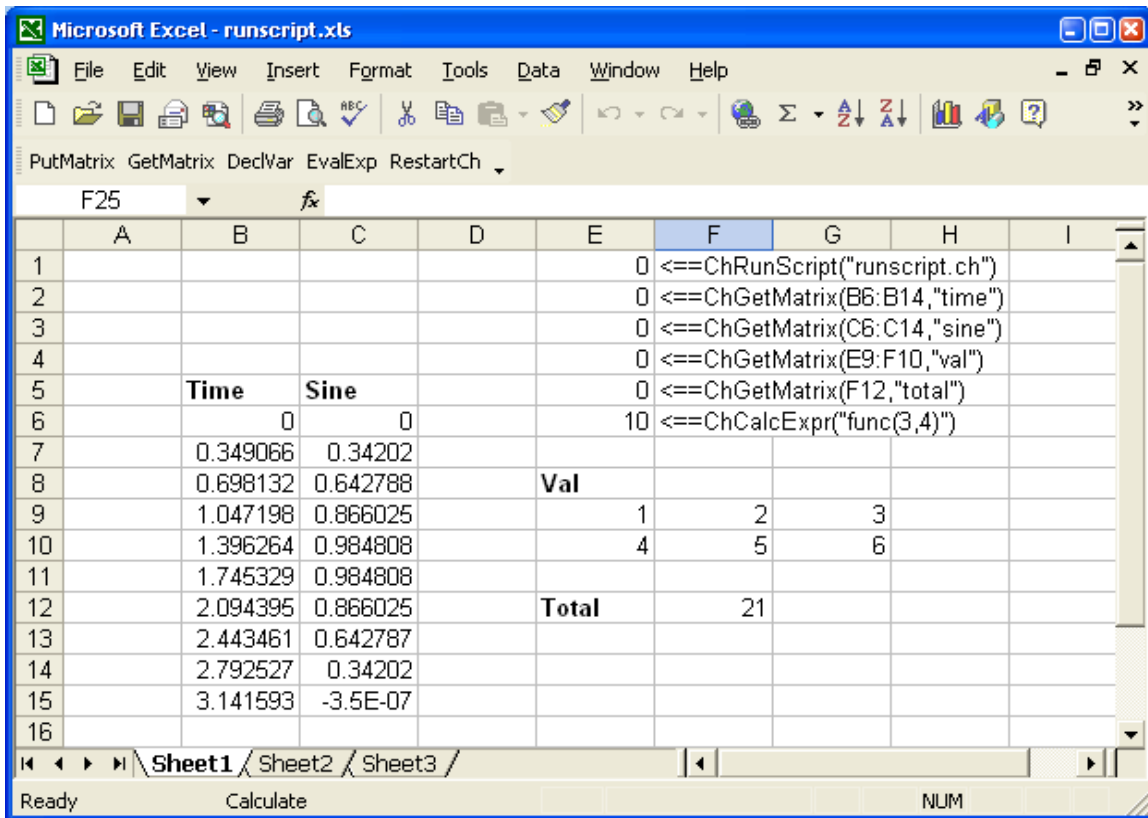


Figure 13. After evaluation of ChRunScript.

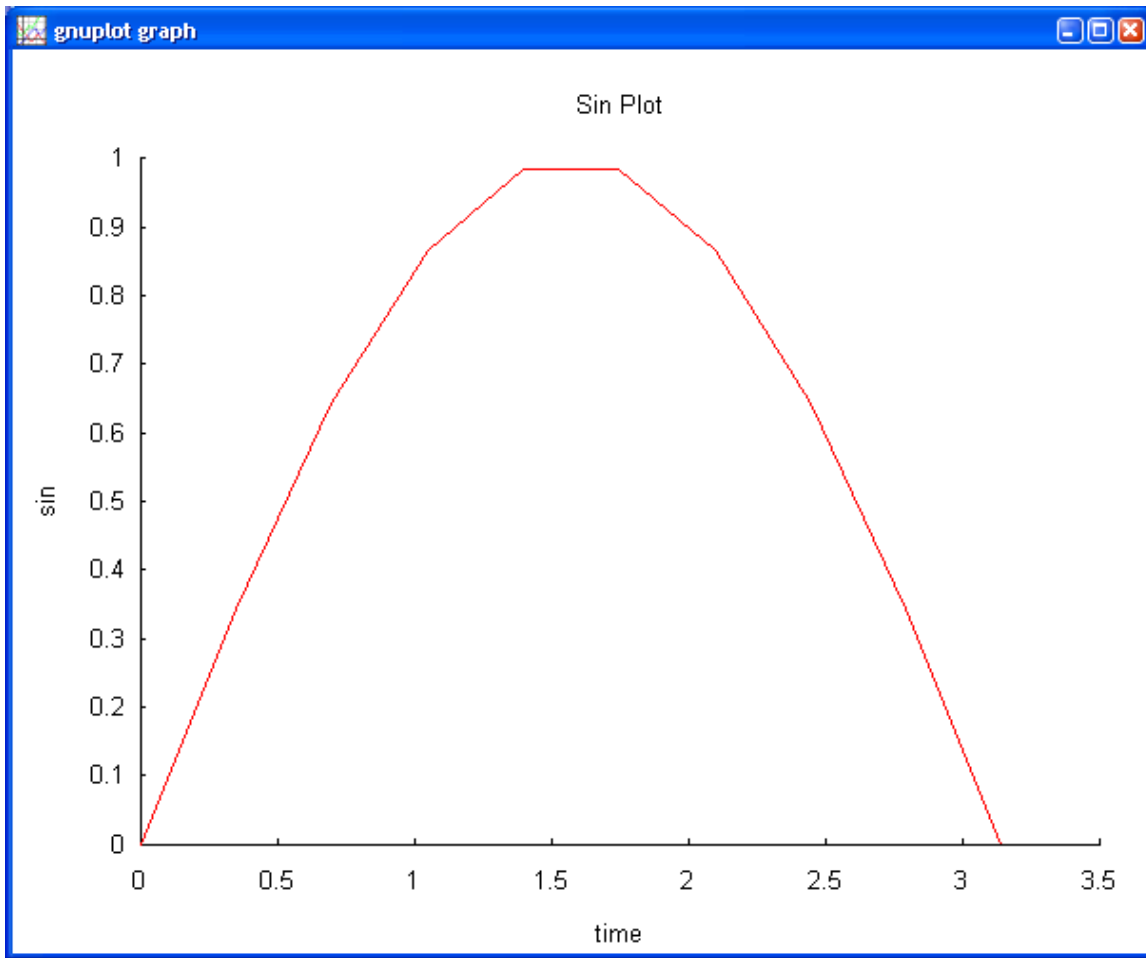


Figure 14. Plot generated by (runscript.ch) and (appendscript.ch).

The Ch script `appendscript.ch` in Program 2 modifies variables declared in the current Ch session. The worksheet `appendscript.xls`, shown in Figure 15, declares three Ch variables using the functions `ChPutMatrix()` and `ChDeclVar()`. Cell E5 contains the ChExcel function `ChAppendScript()` which runs the script `appendscript.ch`. Figure 16 shows the resulting worksheet after the `ChGetMatrix()` functions have been evaluated. Cell E8 displays the return value of the function `func()` defined in `appendscript.ch`. The plot generated by `appendscript.ch` is shown in Figure 14.

```
#include <numeric.h>
#include <array.h>

/* these variables are declared in ChExcel functions */
//array double val[2][3];
//array double time[10], sine[10];

double total;
total = sum(val);
sine = sin(time);
plotxy(time,sine,"Sin Plot", "time", "sin");

double func(double x, double y) {
    return 2*hypot(x,y);
}
```

Program 2. Script for ChAppendScript (appendscript.ch).

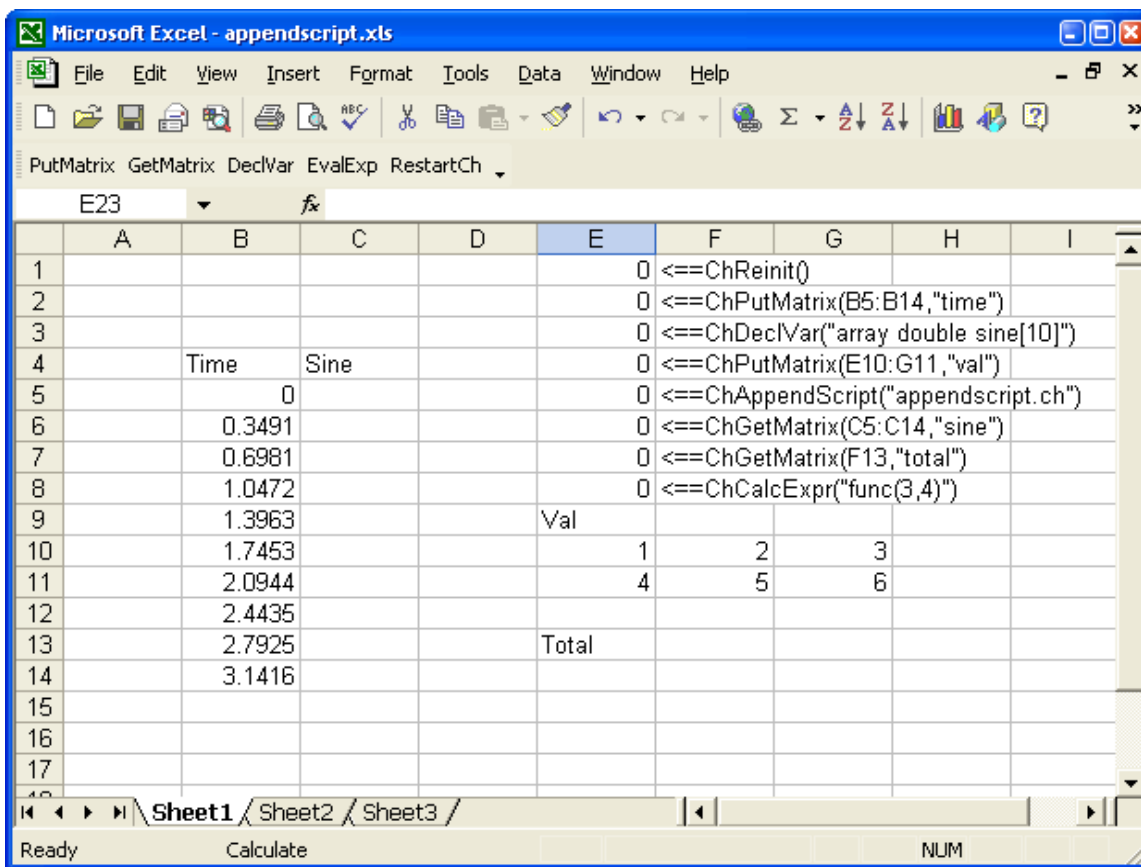


Figure 15. Before evaluation of ChAppendScript.

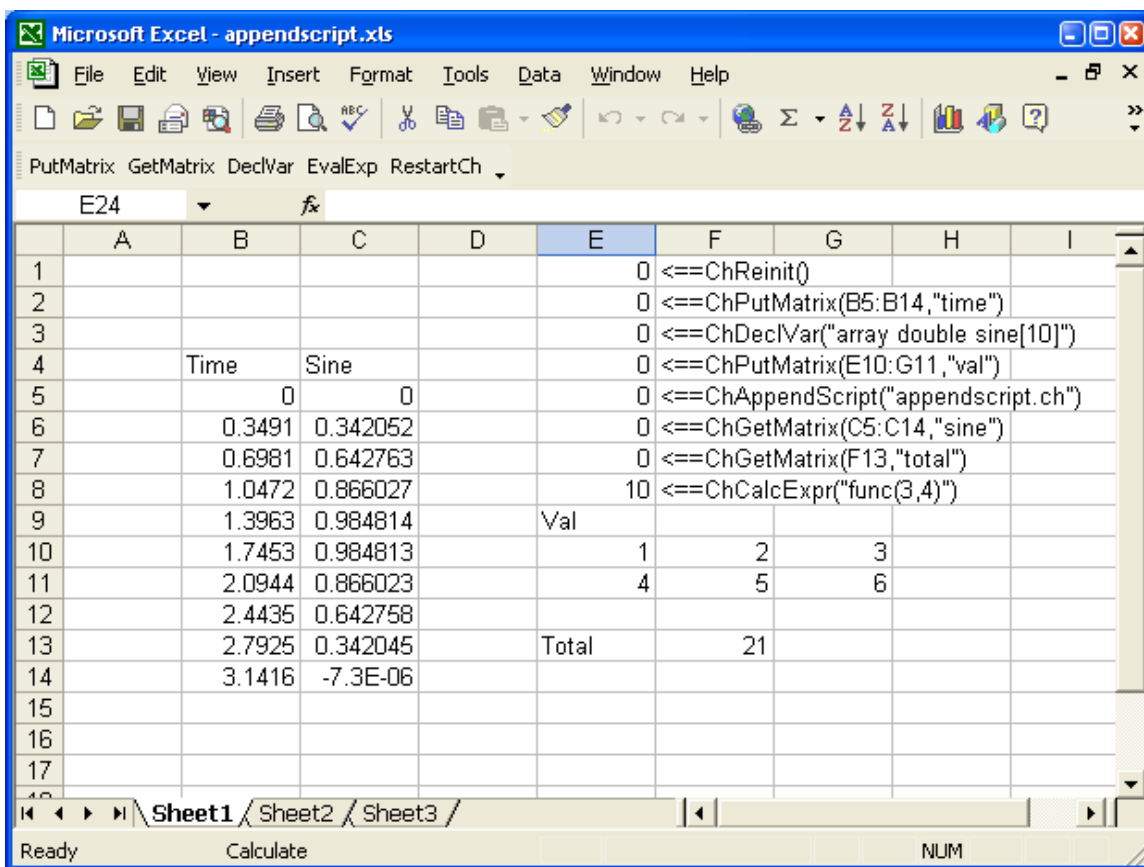


Figure 16. After evaluation of ChAppendScript.

5.7 Functions Capable of Plotting

There are a few ChExcel functions that are capable of producing plots. The first is ChExprEval(), which can evaluate the Ch function plotxy() and plotxyz(). The second function that can produce plots is ChFunc() which can call plotting functions and pass them arguments.

5.7.1 Plotting Examples

In the following example, three plots are generated using ChExprEval() and ChFunc(). The cells A1 and A2 declare the variables x and y. Cell A3 evaluates the linspace() function to fill the array x. Cell A4 calculates the values for the array y. Cell A5 evaluates the function plotxy() with the arguments x and y. Cell A6 evaluates the function plotxy() with the three additional arguments to specify the title, and the axis labels. Cell A7 uses ChExcel function ChFunc() to call Ch function plotxy() to generate a plot.

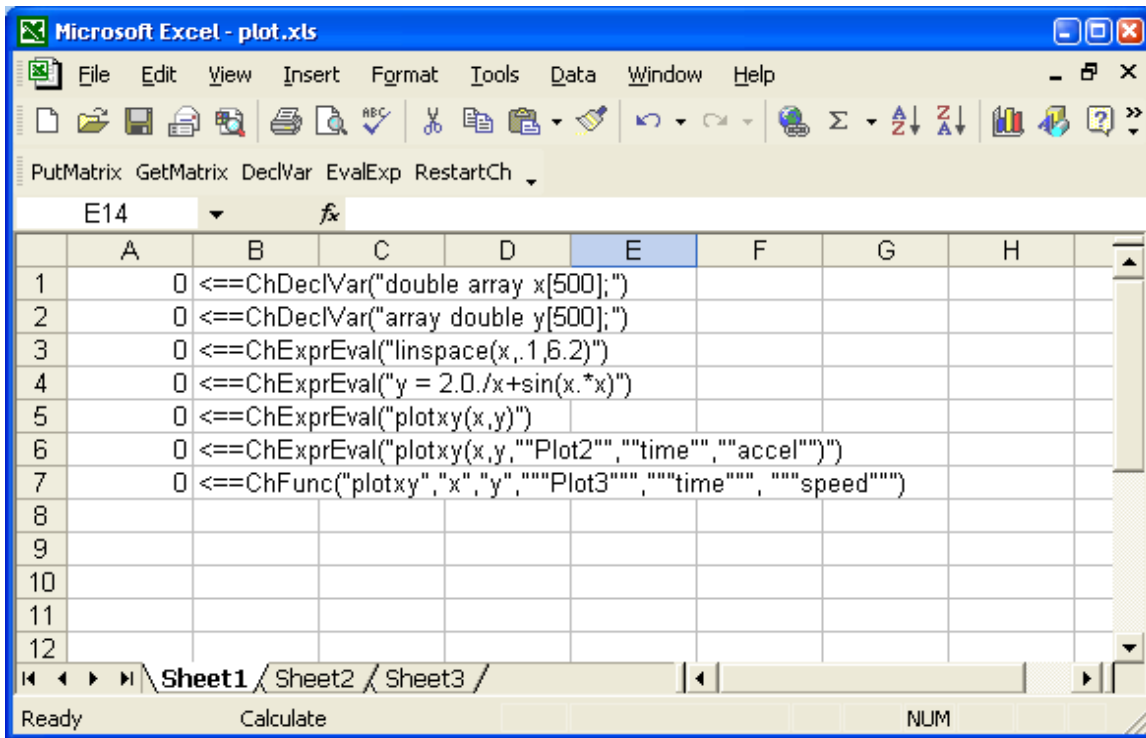


Figure 17. Plotting functions.

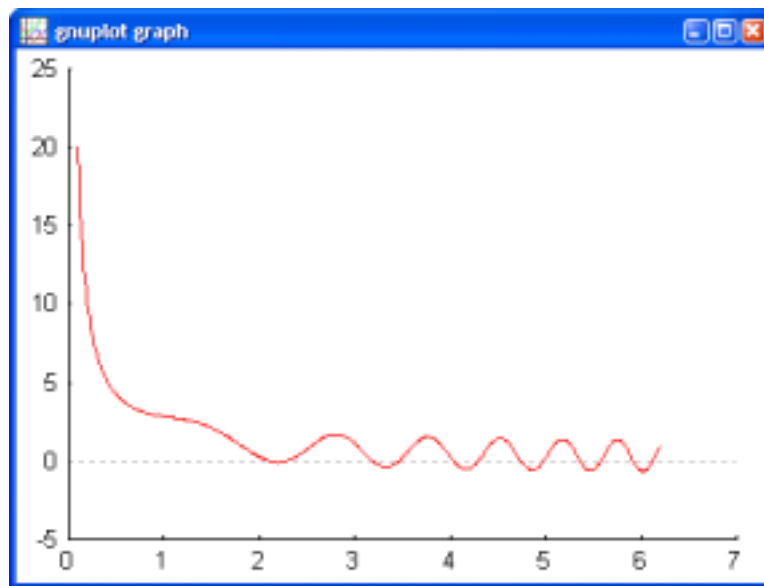


Figure 18. Plot generated by cell A5.

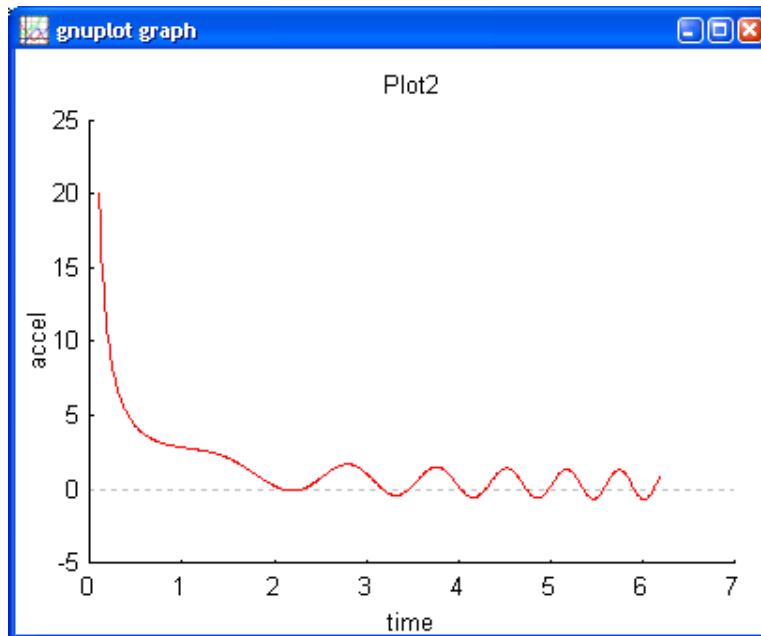


Figure 19. Plot generated by cell A6.

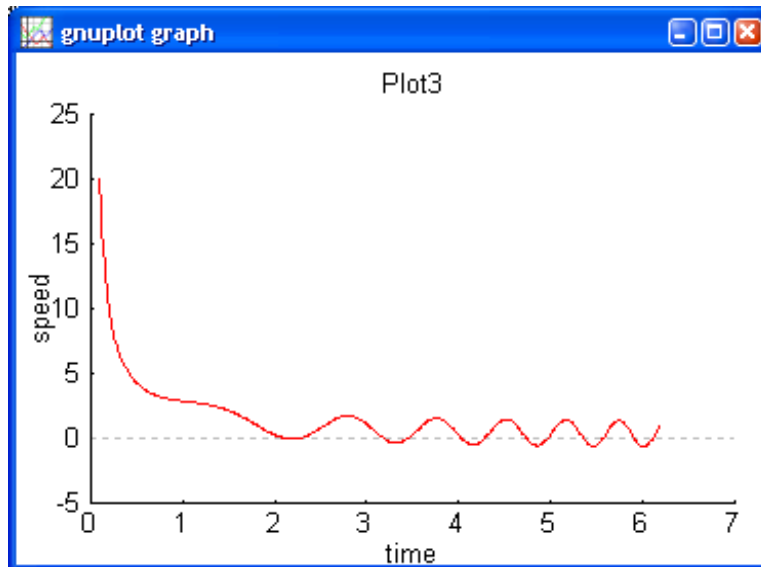


Figure 20. Plot generated by cell A7.

5.8 User Defined Functions

A user defined function is a function file that contains a single function definition. The file and function must have the same name and the file must end in `.chf`. The file also must be in the Ch's function path specified in the system variable `_fpath` in Ch. For ChExcel to be able to call a user defined function, the file must conform to a strict structure. The file must be in the following order:

1. Preprocessor directives.

2. Function declaration.
3. Global variable declaration.
4. Function definition.

There are a few ChExcel functions that are capable of running user defined function. ChFunc() can evaluate a user defined function and return a scalar value. ChFuncMatrix(), and ChCalcExpr() can evaluate a function and return a matrix to the spreadsheet. ChExprEval() can also evaluate a user defined function.

5.8.1 User Defined Function Example

Program 3 defines a function xfunc() in function file xfunc.chf. The program begins by including the header file numeric.h with function prototype for numerical function sum(), followed by function prototype for xfunc() and declaration of the global variable total. The function xfunc() takes an integer as an argument, assigns the sum for each element of the array val declared in ChExcel to total. and returns the product of the integral argument and value of variable total.

```
#include <numeric.h>

/* function prototype right after preprocessing directives */
double xfunc(int n);
/* variable val is declared in ChExcel */
//array double val[2][3];
/* declare the global variables here to be retrieved by ChExcel Functions */
double total;

double xfunc(int n)
{
    total = sum(val);
    return n*total;
}
```

Program 3. User defined function(xfunc.chf)).

Figure 21 shows the spreadsheet xfunc.xls. Cell E1 uses ChExcel function ChPutMatrix() to place the matrix in A2 through C3 into the variable val. Cell E2 uses ChExcel function ChFunc() to call the Ch function xfunc() with the argument 10. The returned value from calling function xfunc() in function file xfunc.chf is placed in Cell E2. Cell E3 uses ChExcel function ChGetMatrix() to get the value of variable total in Ch and places it in Cell B5. The resulting spreadsheet is shown in Figure 22.

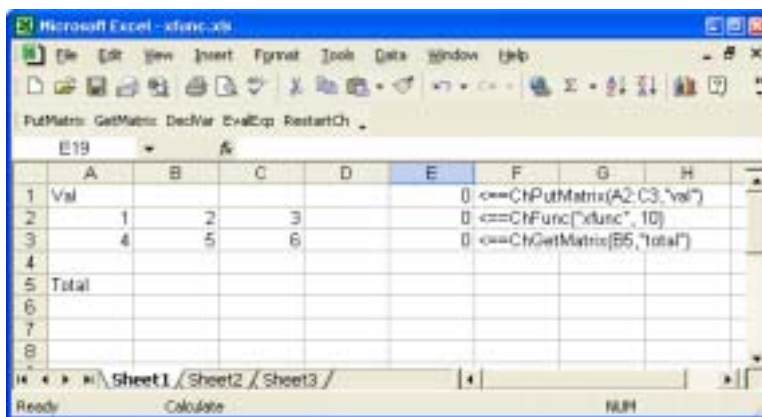


Figure 21. Spreadsheet before running function file xfunc . chf.

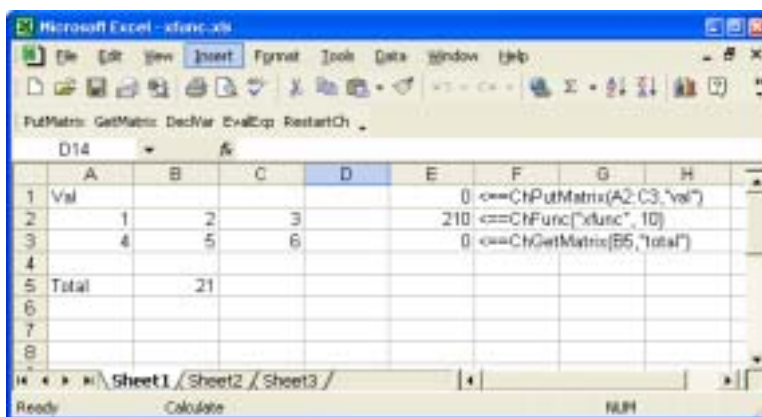


Figure 22. Spreadsheet after running function file xfunc . chf.

6 Using Win32 for Graphical User Interface

Ch supports Win32 API. Ch program with Win32 API can be invoked as a script from Excel using ChExcel. The user defined function messagefunc() in Program4 uses Win32 API MessageBox() to display a message. When function messagefunc() is invoked by ChExcel function ChFunc() from a spreadsheet a message box will be displayed as shown in Figure 23. The other Win32 APIs can also be used inside a Ch script. Figure 24 displays dialog boxes created when Ch script dialogbox . ch is invoked from an Excel spreadsheet dialogbox . xls. created from.

6 USING WIN32 FOR GRAPHICAL USER INTERFACE

```
#include <numeric.h>
#include <windows.h>
#define APP_NAME "MessageBox Title"

/* function prototype right after preprocessing directives */
double messagefunc(int n);
/* these variables are declared in ChExcel functions */
//array double val[2][3];
/** declare the global variables here to be retrieved by ChExcel Functions */
double total;

double messagefunc(int n) {
    CHAR string[100];

    total = sum(val);
    sprintf(string, "This is a message generated from MessageBox().\n"
                "total = %f\n", total);
    MessageBox( NULL, string, APP_NAME, MB_OK | MB_SYSTEMMODAL | MB_NOFOCUS);
    return n*total;
}
```

Program 4. User defined function messagefunc.chf using Win32 API MessageBox().

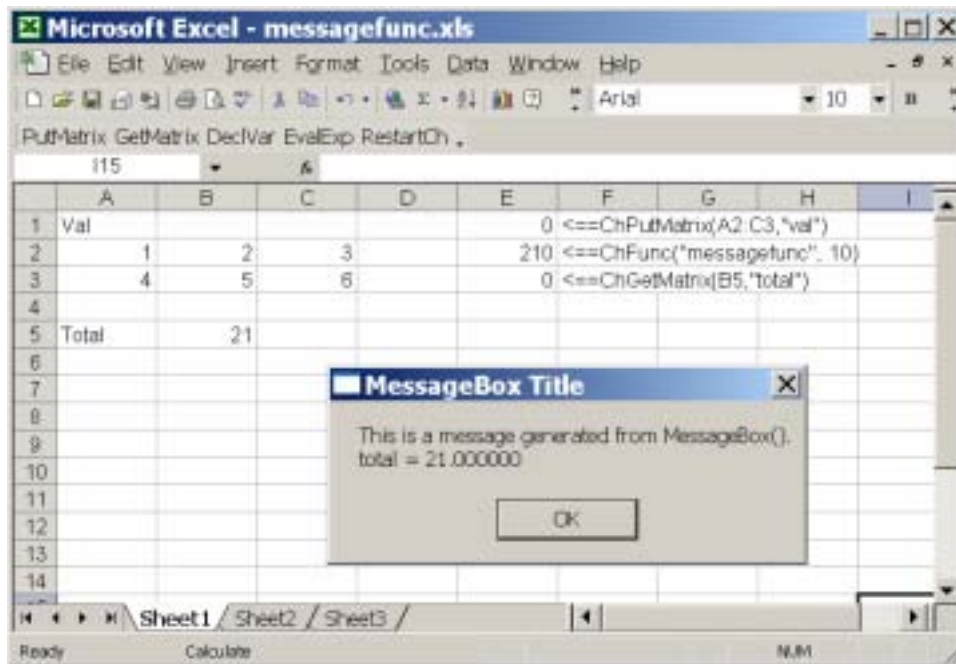


Figure 23. Spreadsheet and messagebox using MessageBox() in Win32 API.

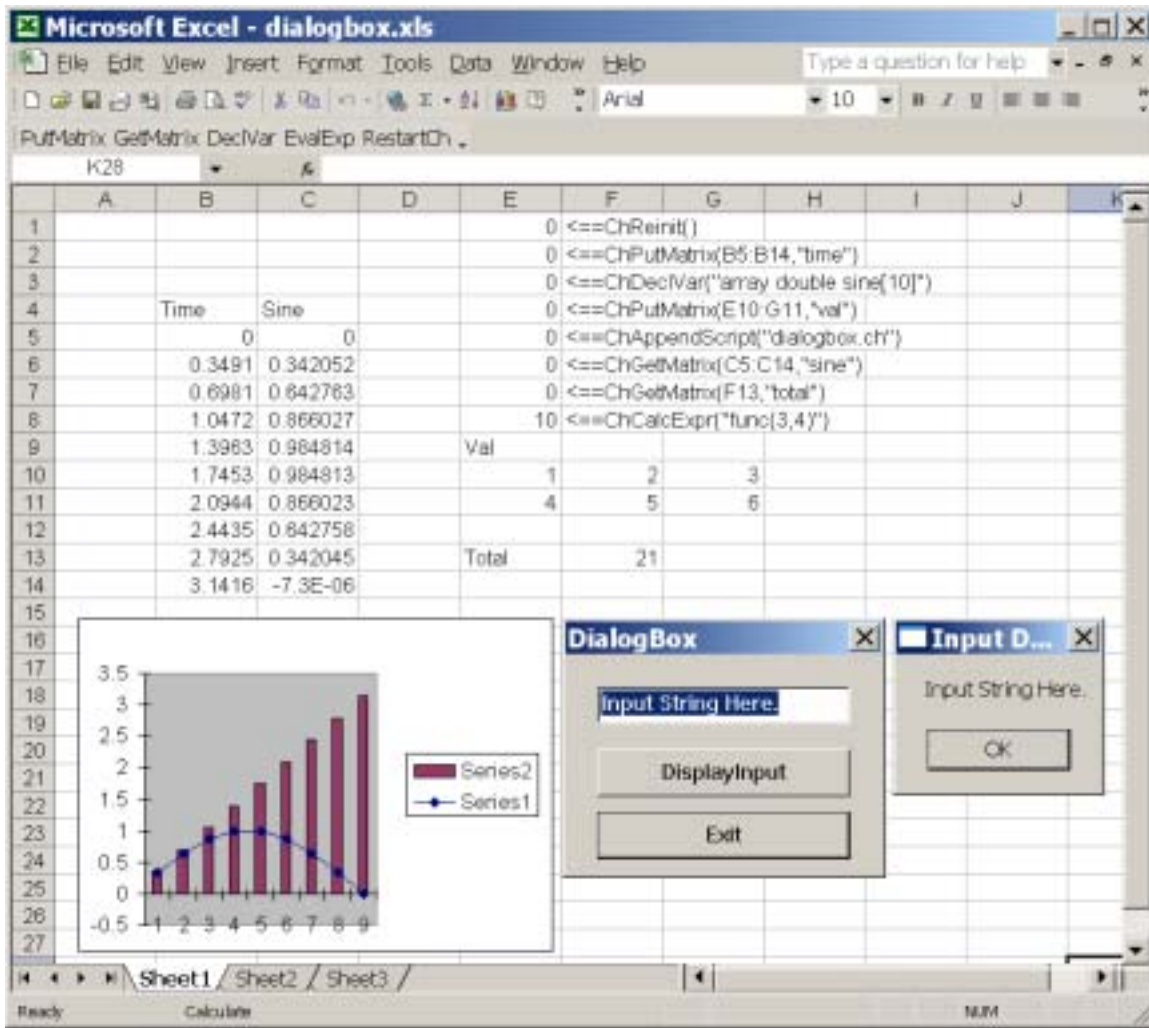


Figure 24. Spreadsheet and dialog boxes from Win32 API.

7 Programming in Visual Basic for Applications (VBA) with ChExcel

VBA is a convenient Excel programming tool. With ChExcel you can use the power of Ch in VBA. To open the VBA editor in Excel press ALT + F11. To use ChExcel functions within an Excel VBA program, you need to go to the tools menu of the VBA editor and click on References. In the References window, check the box next to ChExcel. Excel allows macros to be defined within spreadsheets and modules, while user defined Excel functions can only be defined within modules. ChExcel provides two additional functions to interact between the VBA environment and Ch.

7.1 ChGetVar()

The first of the functions is ChGetVar. ChGetVar takes a Ch variable and a VBA variable as arguments. It then places the data or matrix in the Ch variable into the VBA variable. The VBA variable must be declared with the same dimensions as the VBA variable or it will produce errors.

7.2 ChPutVar()

The second of the function is ChPutVar. This function takes a Ch variable and a VBA variable as arguments and places the data or matrix from the VBA variable into the Ch variable. ChPutVar declares the Ch variable passed to it, which overwrites any previous declarations of the variable name.

7.3 VBA Programming Examples

The macros PlotCh, AppendScript, and RunScript are located in the ThisWorkbook sheet for Excel program VBA.xls as shown in Figure 25. Figure 26 shows these macros in the Excel VBA Editor. Macros located in this sheet are accessible from all worksheets in the workbook. To run a macro from the VBA editor, click on the macro definition so that the macro name shows up in the upper right window and click the run button as shown in Figure 25. To run a macro from a worksheet, go to the Tools menu click on Macro and then click on Macros. From the macro menus select the macro you wish to run and click Run. The macro menu is shown in Figure 25.

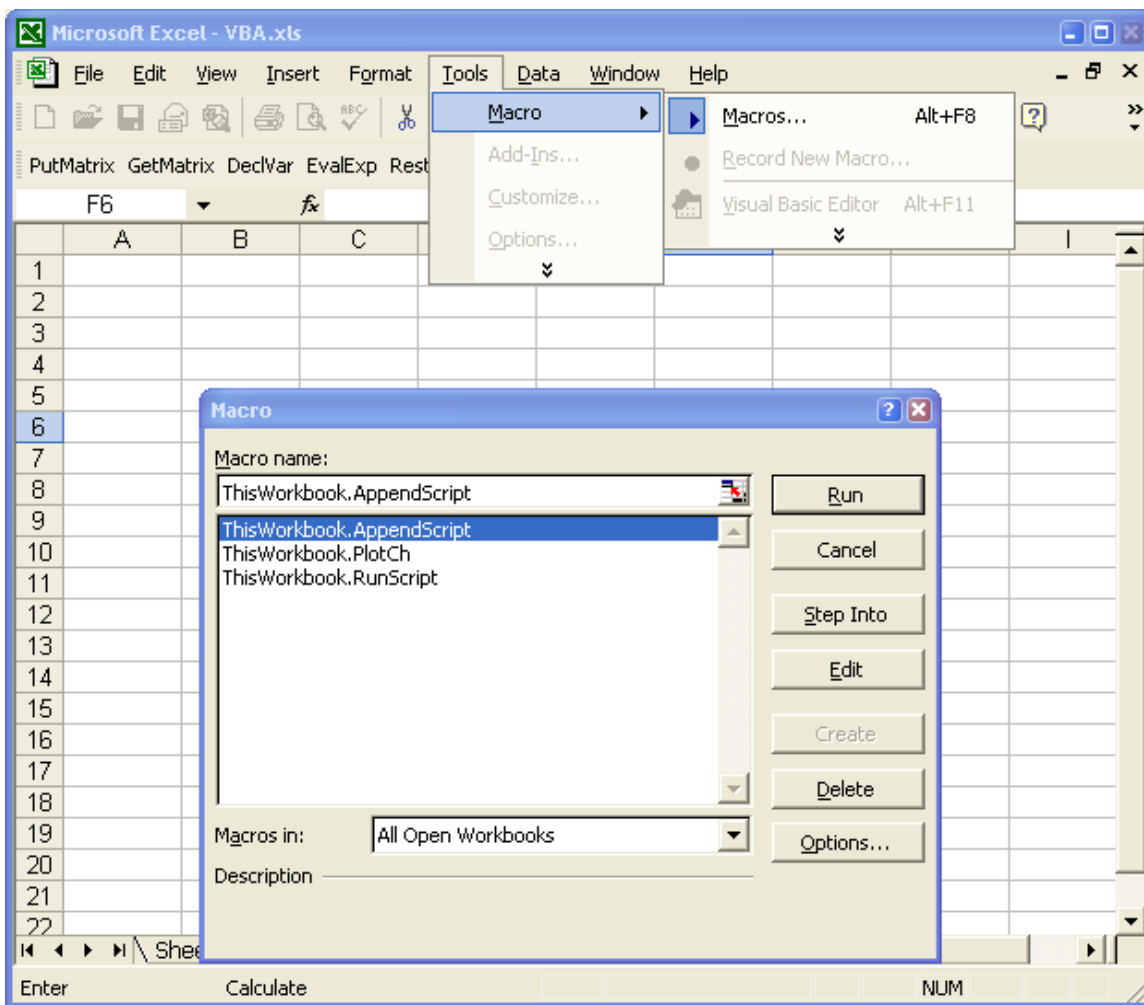


Figure 25. Excel macro menu.

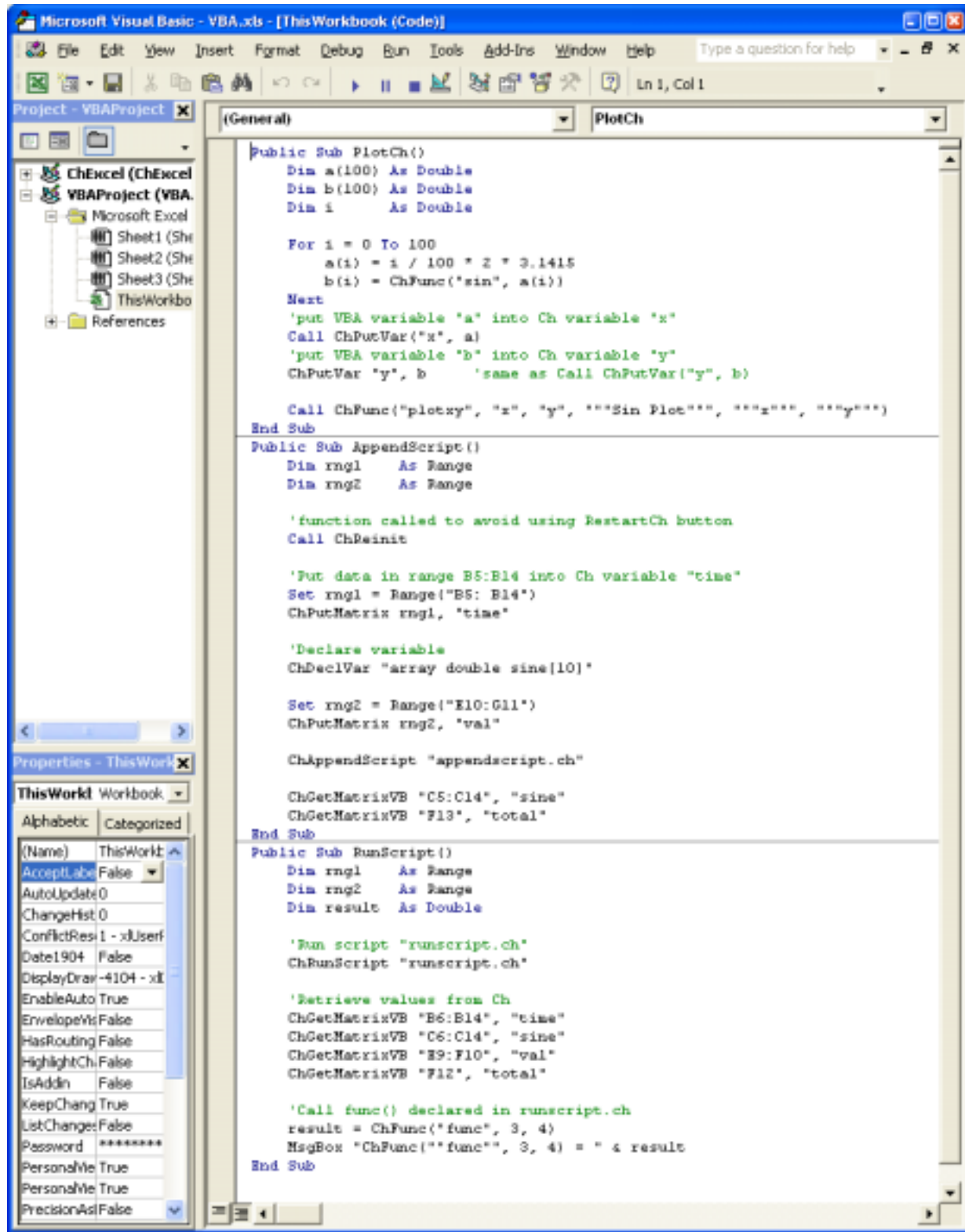


Figure 26. Excel VBA editor.

```
Public Sub PlotCh()  
    Dim a(100) As Double  
    Dim b(100) As Double  
    Dim i      As Double  
  
    For i = 0 To 100  
        a(i) = i / 100 * 2 * 3.1415  
        b(i) = ChFunc("sin", a(i))  
    Next  
    'put VBA variable "a" into Ch variable "x"  
    Call ChPutVar("x", a)  
    'put VBA variable "b" into Ch variable "y"  
    ChPutVar "y", b      'same as Call ChPutVar("y", b)  
  
    Call ChFunc("plotxy", "x", "y", ""Sin Plot"", ""x"", ""y"")  
End Sub
```

Program 5. VBA macro (PlotCh).

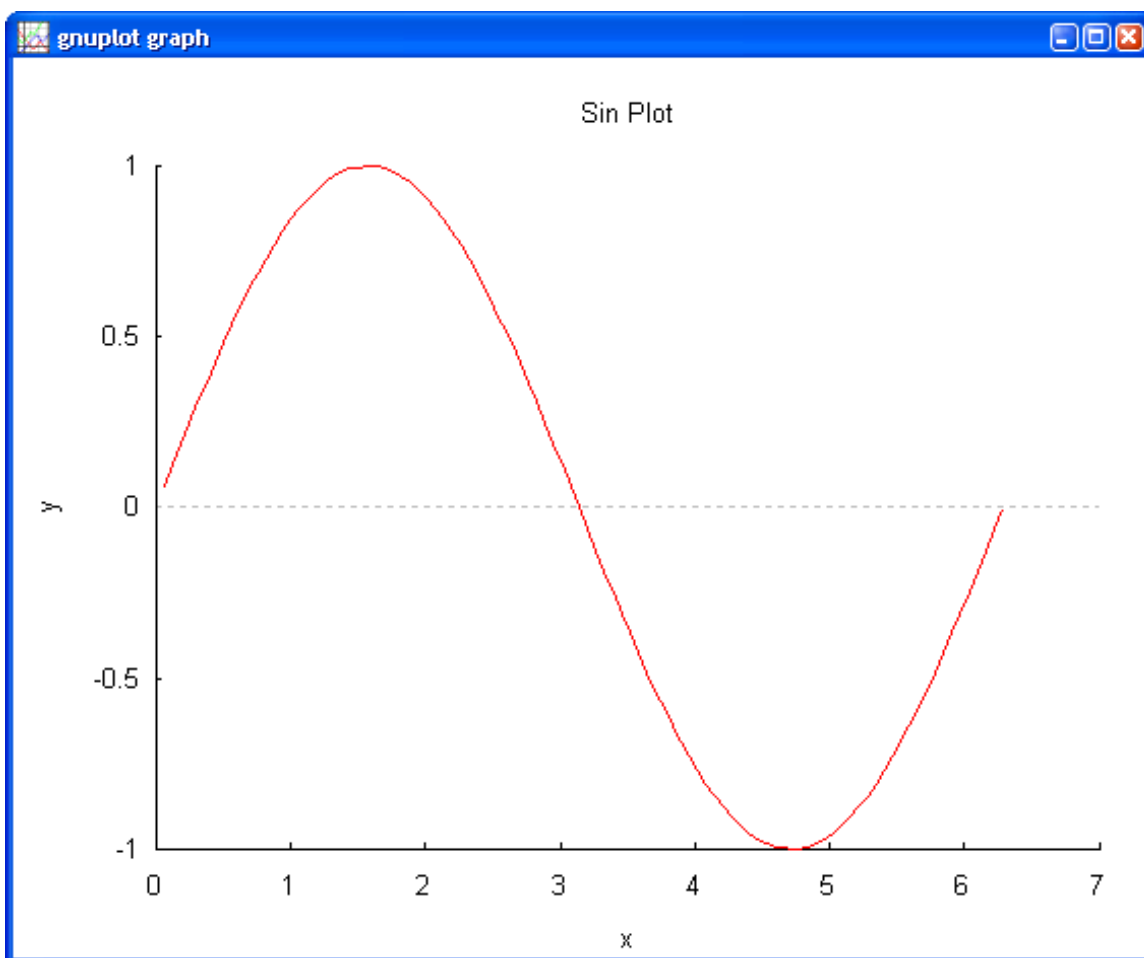


Figure 27. Plot generated by macro PlotCh.

The macro `PlotCh`, shown in Program 5, begins by declaring two VBA arrays. The arrays are then filled with values using the ChExcel function `ChFunc()`. The VBA variables are placed into the Ch environment by the function `ChPutVar()`. A plot is then generated from the Ch variables `x` and `y` by the function `ChFunc()`.

The macro `AppendScript` is shown in Program 6. It begins by reinitializing the Ch session so there won't be any variable name conflicts. It then places the data in the range B5:B14 into the Ch variable `time`. The reason that we have to cast the string "B5:B14" into the range variable `rng1` is because the ChExcel function `ChPutMatrix` takes a range as the first argument. The macro then declares the array `sine` and the variable `val`. `ChAppendScript` is then called to run the script `appendscript.ch`, shown in Program 2. The matrices `sine` and `total` are then placed onto the spreadsheet. Figure 28 shows the spreadsheet in Sheet1 before the macro `AppendScript` is run and Figure 29 shows the spreadsheet after `AppendScript` is run.

```
Public Sub AppendScript()
    Dim rng1    As Range
    Dim rng2    As Range

    'Function called to avoid using RestartCh button
    Call ChReinit

    'Put data in range B5:B14 into Ch variable "time"
    Set rng1 = Range("B5: B14")
    ChPutMatrix rng1, "time"

    'Declare variable
    ChDeclVar "array double sine[10]"

    Set rng2 = Range("E10:G11")
    ChPutMatrix rng2, "val"

    ChAppendScript "appendscript.ch"

    ChGetMatrixVB "C5:C14", "sine"
    ChGetMatrixVB "F13", "total"
End Sub
```

Program 6. VBA macro (appendscript).

7 PROGRAMMING IN VISUAL BASIC FOR APPLICATIONS (VBA) WITH CHEXCEL
7.3 VBA Programming Examples

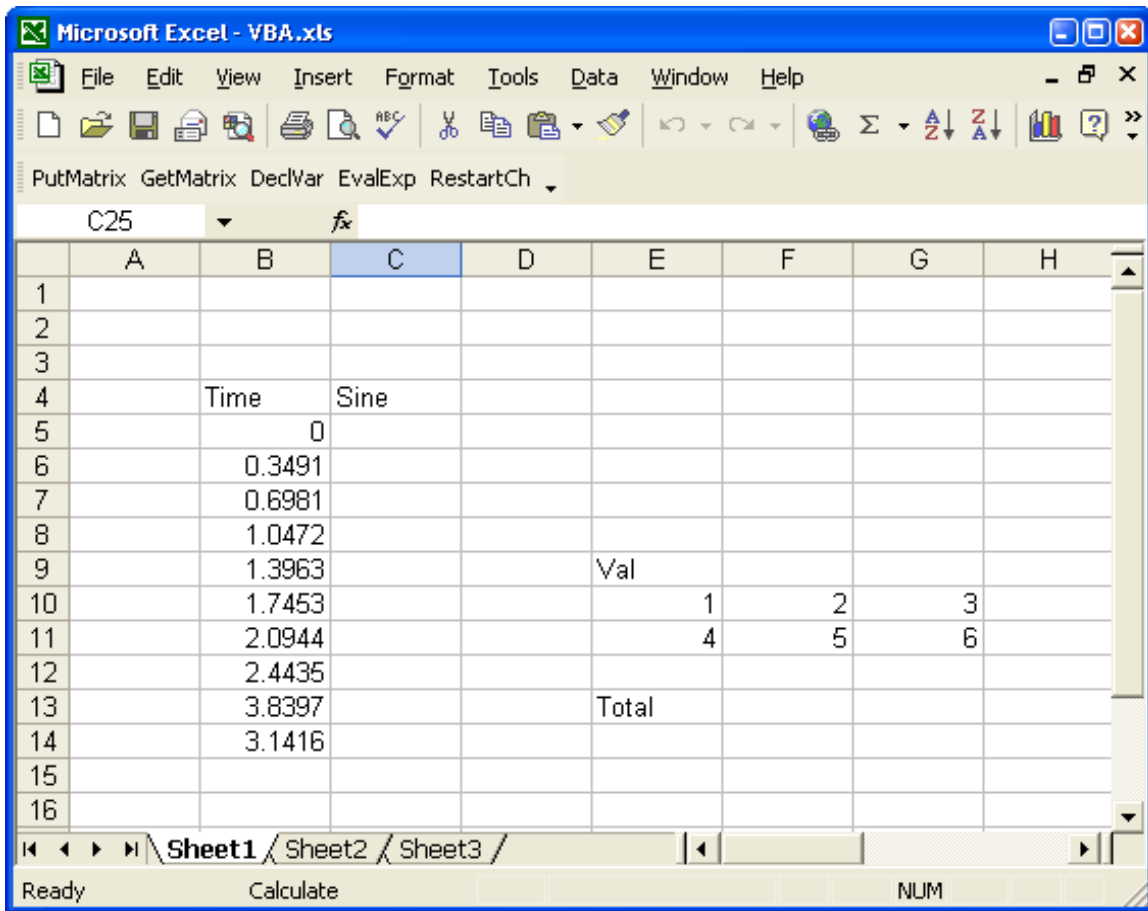


Figure 28. Before running appendscript.

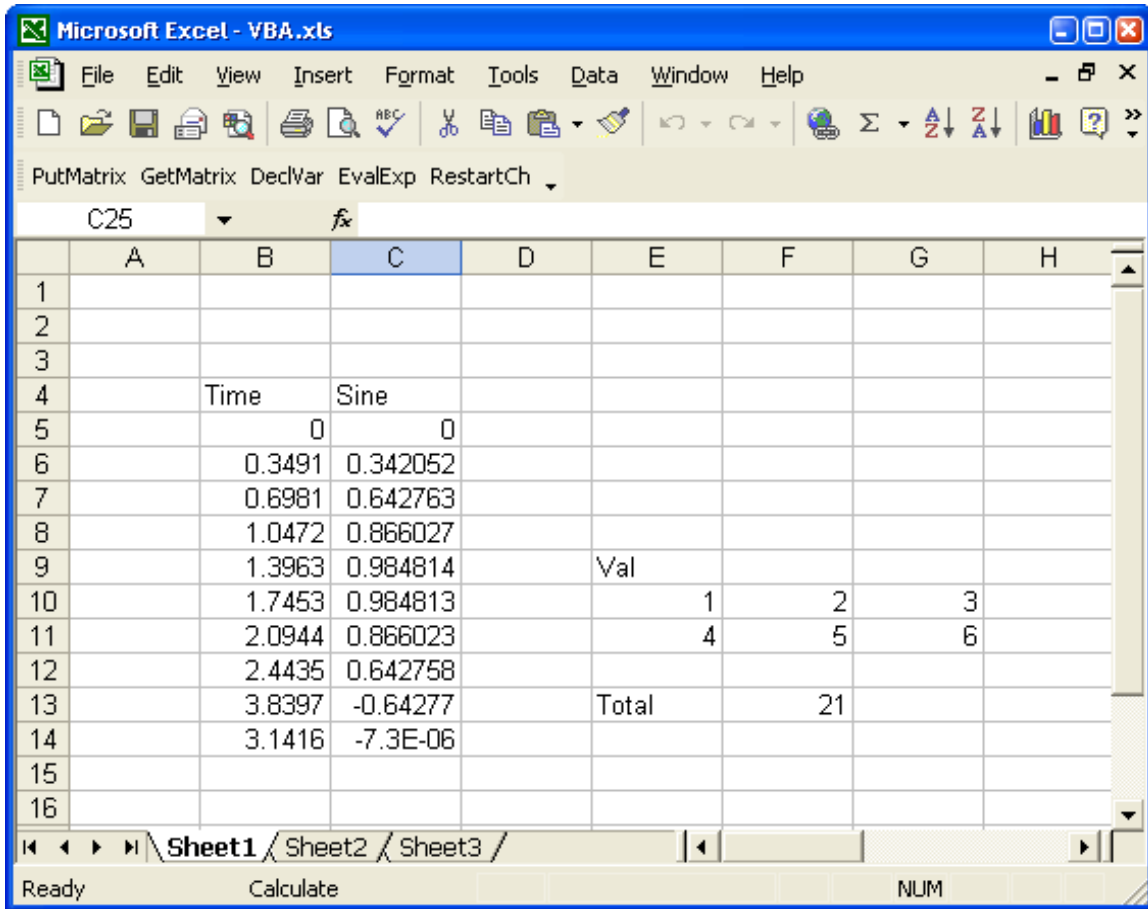


Figure 29. After running appendscript.

The macro RunScript is shown in Program 7. It begins by calling ChRunScript to run the script runscript.ch, shown in Program 1. A few variables are then retrieved from Ch and placed onto the spreadsheet. The VBA variable is then assigned the value returned from the function func () and a message box is displayed showing the value of result. Figure 30 shows the spreadsheet in Sheet2 before the macro RunScript is run and Figure 31 shows the spreadsheet after RunScript is executed.

7 PROGRAMMING IN VISUAL BASIC FOR APPLICATIONS (VBA) WITH CHEXCEL

7.3 VBA Programming Examples

```
Public Sub RunScript()  
    Dim result As Double  
  
    'Run script "runscript.ch"  
    ChRunScript "runscript.ch"  
  
    'Retrieve values from Ch  
    ChGetMatrixVB "B6:B14", "time"  
    ChGetMatrixVB "C6:C14", "sine"  
    ChGetMatrixVB "E9:F10", "val"  
    ChGetMatrixVB "F12", "total"  
  
    'Call func() declared in runscript.ch  
    result = ChFunc("func", 3, 4)  
    MsgBox "ChFunc(""func"", 3, 4) = " & result  
End Sub
```

Program 7. VBA macro (runscript).

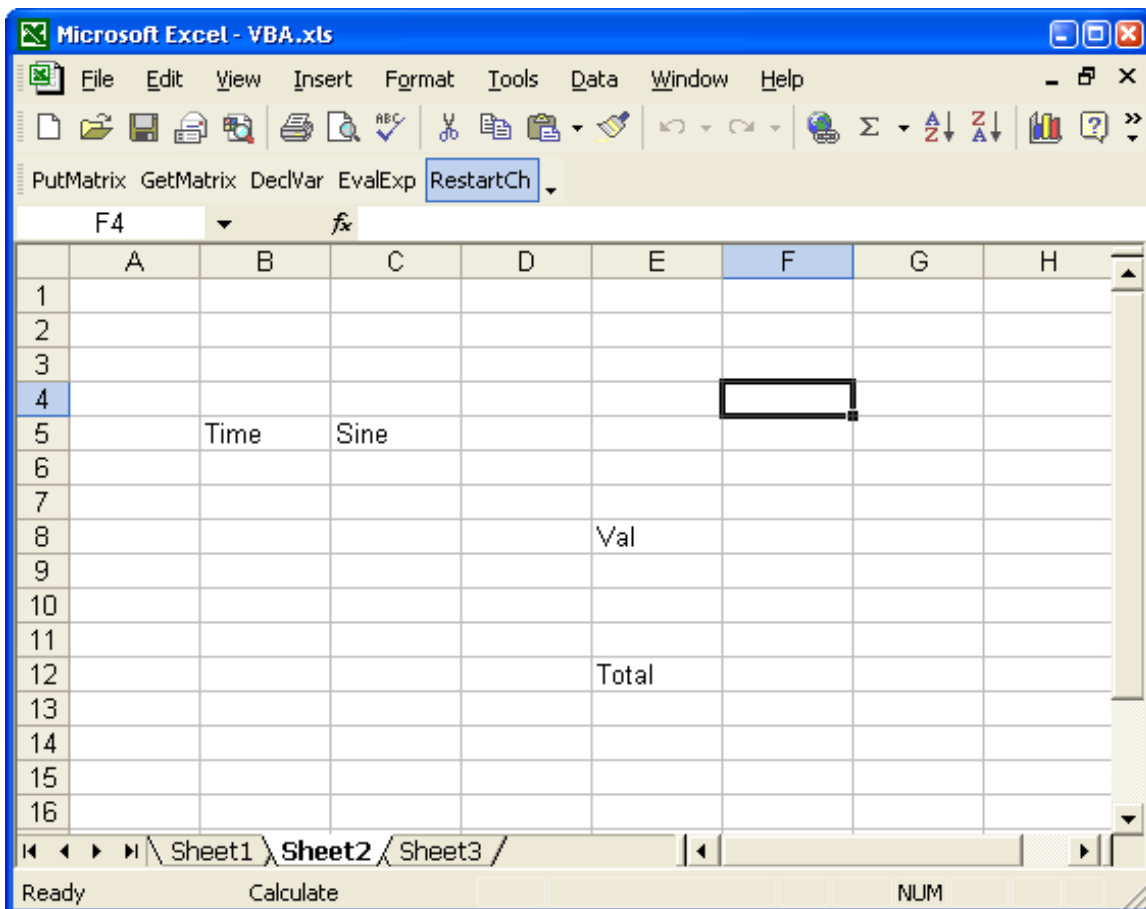


Figure 30. Before running runscript.

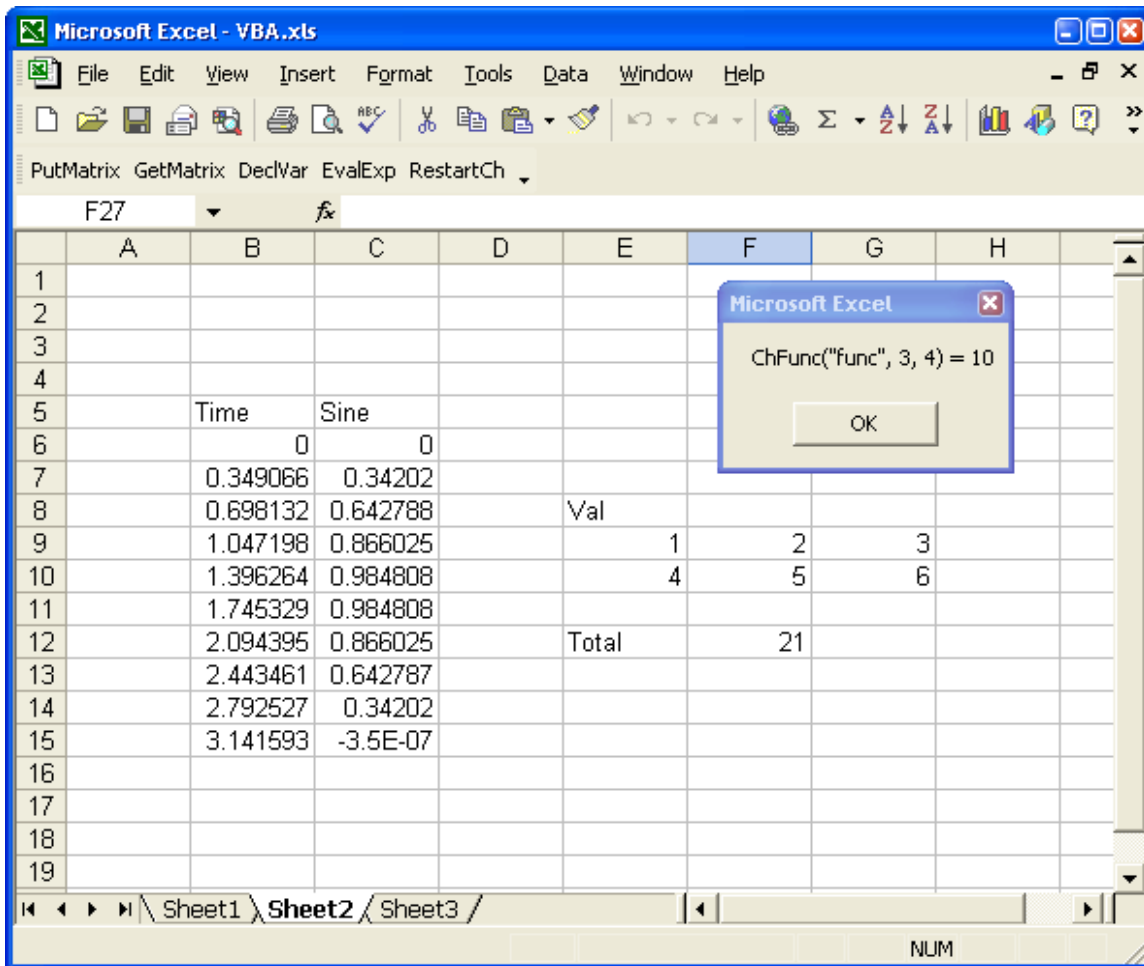


Figure 31. After running runscript.

8 Examples

8.1 Example 1: Data Interpolation

The Ch script in Program 8 performs interpolation for two dimensional data. Interpolation is a method of estimating intermediate values between data points. Ch provides the function `interp2()` to interpolate two dimensional data. The function `interp2()` takes seven arguments. The first three arguments are matrices to be filled with interpolated data. The second set of three matrices contain known data points. The last argument specifies the type of interpolation to perform.

The script `chinterp.ch` is a stand alone Ch program. It begins by declaring four matrices and filling them with data. It then calls the function `interp2()` and fills the two dimensional matrix `za` with interpolated data.

```

#include <chplot.h>
#include <numeric.h>

int m=12, n = 8;
array double x[m],y[n], z[m][n],zd[m*n];
int i,j;

/* Construct data set of the peaks function */
linspace(x, -3, 3);
linspace(y, -4, 4);
for(i=0; i<m; i++) {
    for(j=0; j<n; j++) {
        z[i][j] = 3*(1-x[i])*(1-x[i])*
            exp(-(x[i]*x[i])-(y[j]+1)*(y[j]+1))
            - 10*(x[i]/5 - x[i]*x[i]*x[i]-
            pow(y[j],5))*exp(-x[i]*x[i]-y[j]*y[j])
            - 1/3*exp(-(x[i]+1)*(x[i]+1)-y[j]*y[j]));
        printf("%5.2f ", z[i][j]);
    }
    printf("\n");
}

zd = (array double[m*n])z;

/* For interpolated data */
int mx = 20, ny = 16;
array double xa[mx],ya[ny], za[mx][ny], zad[mx*ny];
class CPlot plot;

linspace(xa, -3, 3);
linspace(ya, -4, 4);

/* 2-dimensional cubic spline interpolation*/
interp2(za,xa,ya,x,y,z,"spline");

/* add offset for display */
zad = (array double[mx*ny])za + 100;

plot.data3D(x, y, zd);
plot.data3D(xa, ya, zad);
plot.label(PLOT_AXIS_X, "x");
plot.label(PLOT_AXIS_Y, "y");
plot.label(PLOT_AXIS_Z, "z");
plot.ticsLevel(0);
plot.text("Spline", PLOT_TEXT_RIGHT,3.5,3.5,120);
plot.text("Original", PLOT_TEXT_RIGHT,3.5,3.5,20);
plot.plotType(PLOT_PLOTTYPE_LINES,0,1,1);
plot.plotType(PLOT_PLOTTYPE_LINES,1,1,1);
plot.plotting();

```

Program 8. Script for ChRunScript (chinterp.ch).

The worksheet `chinterp.xls` is shown in Figure 32. It starts by running the script `chinterp.ch` with `ChRunScript("chinterp.ch")`. It then retrieves the two matrices `z` and `za` and places them on the spreadsheet. The result of the evaluation is shown in Figure 33. The interpolation plot is shown in Figure 34.

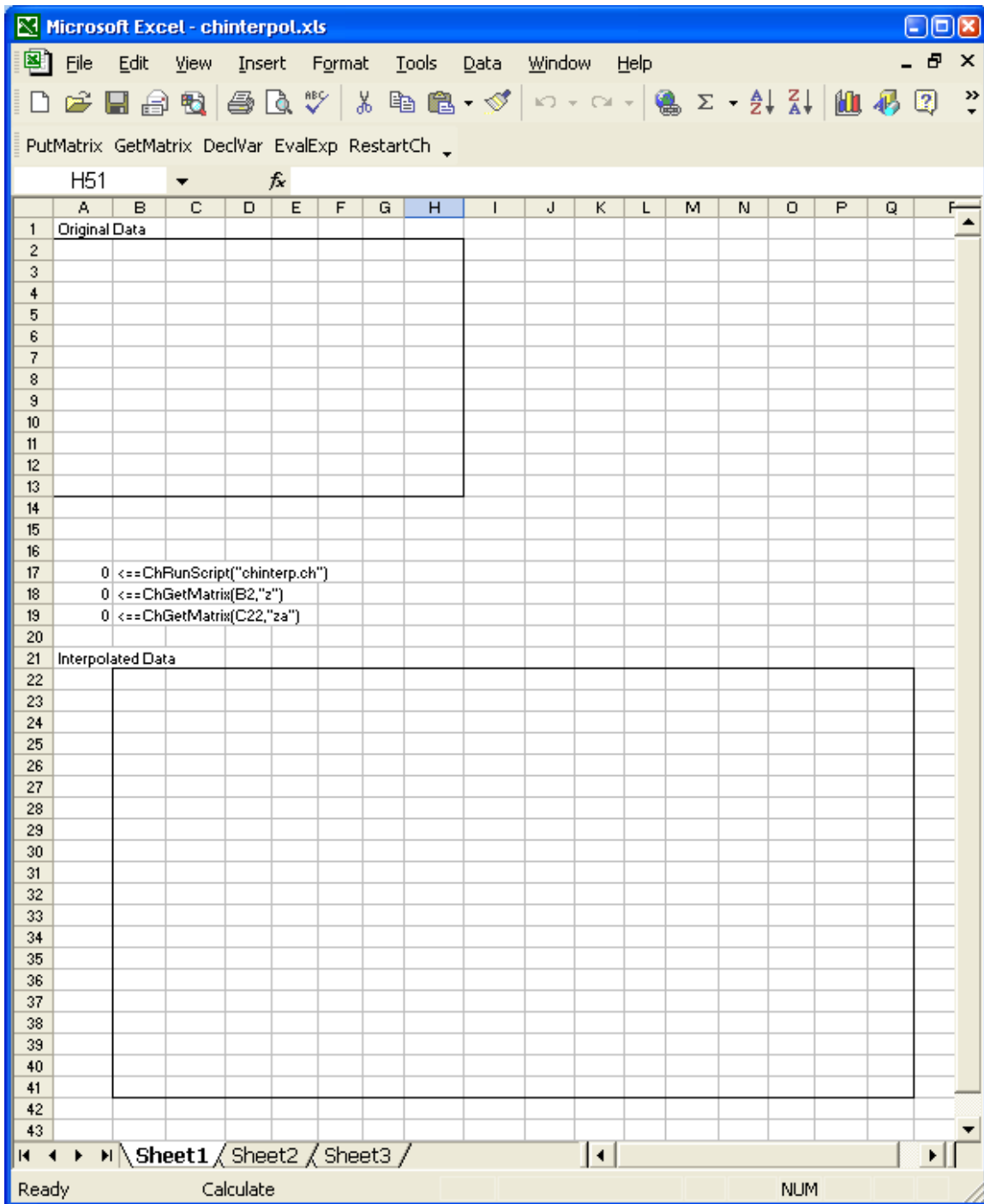


Figure 32. Before evaluation of ChRunScript.

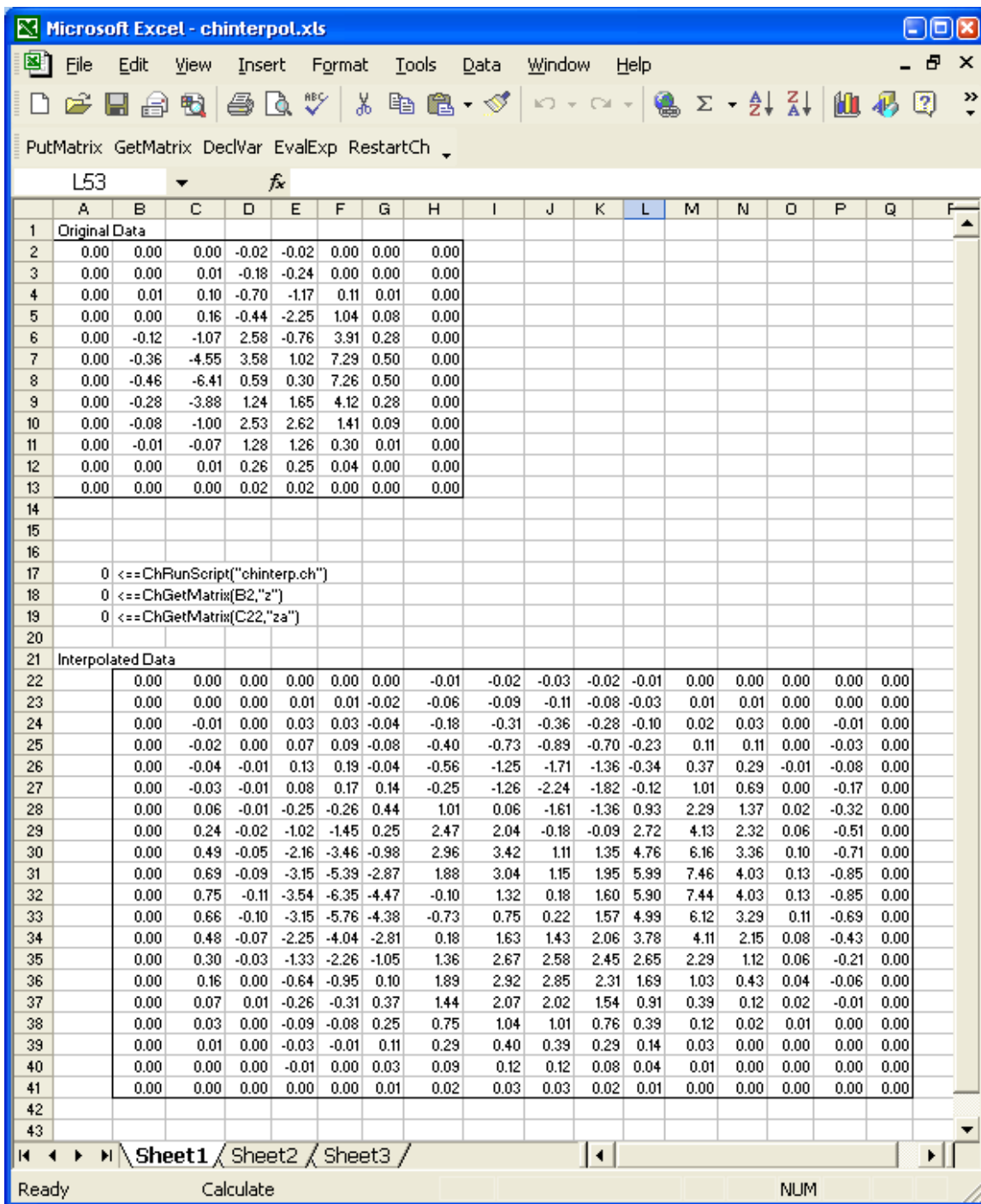


Figure 33. After evaluation of ChRunScript.

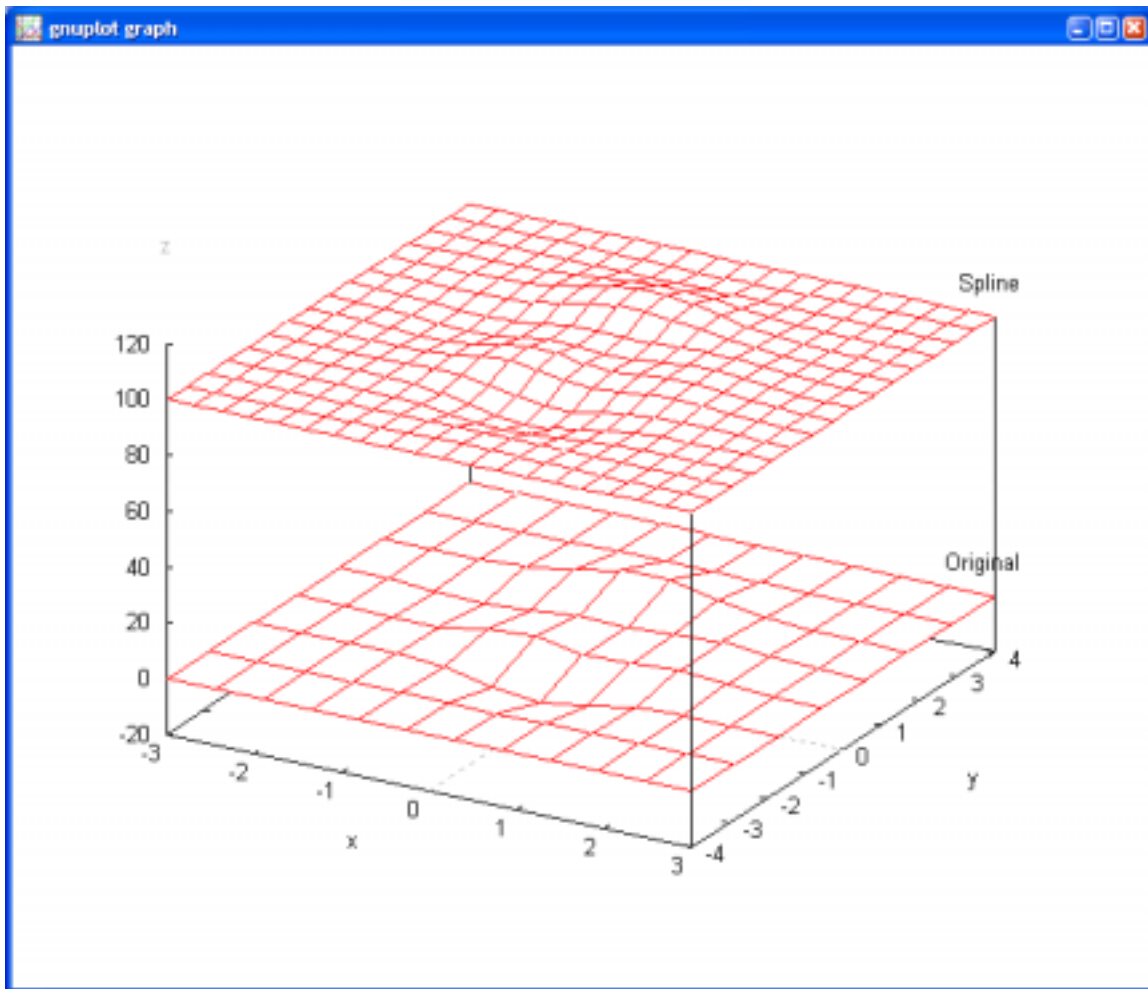


Figure 34. Plot generated by `chinterp.ch`.

The worksheet `excelinterp.xls` displayed in Figure 35 runs the script `excelinterp.ch`. The worksheet begins by declaring a variable `z`. The function in cell A15 puts the data located in A2 through H13 into the Ch variable `z`.

The script `excelinterp.ch`, shown in Program 9, uses the Ch variable `z` to generate the matrix `za`. The dimensions `m` and `n` for row and column of array `z` are obtained by function `shape()`.

The plot produced by `excelinterp.ch` is shown in Figure 34. The resulting spreadsheet is shown in Figure 36.


```

#include <chplot.h>
#include <numeric.h>

/* variable z is declared inside ChExcel function */
//array double z[m][n];

//int m=12, n = 8;
array int dim[2] = shape(z);
int m = dim[0], n = dim[1];

/* For plotting of the original data */
array double x[m],y[n], zd[m*n];
zd = (array double[m*n])z;
linspace(x, -3, 3);
linspace(y, -4, 4);

/* For interpolated data */
int mx = 20, ny = 16;
array double xa[mx],ya[ny], za[mx][ny], zad[mx*ny];
class CPlot plot;

linspace(xa, -3, 3);
linspace(ya, -4, 4);

/* 2-dimensional cubic spline interpolation*/
interp2(za,xa,ya,x,y,z,"spline");

/* add offset for display */
zad = (array double[mx*ny])za + 100;

plot.data3D(x, y, zd);
plot.data3D(xa, ya, zad);
plot.label(PLOT_AXIS_X, "x");
plot.label(PLOT_AXIS_Y, "y");
plot.label(PLOT_AXIS_Z, "z");
plot.ticsLevel(0);
plot.text("Spline", PLOT_TEXT_RIGHT,3.5,3.5,120);
plot.text("Original", PLOT_TEXT_RIGHT,3.5,3.5,20);
plot.plotType(PLOT_PLOTTYPE_LINES,0,1,1);
plot.plotType(PLOT_PLOTTYPE_LINES,1,1,1);
plot.plotting();

```

Program 9. Script for ChAppendScript (excelinterp.ch).

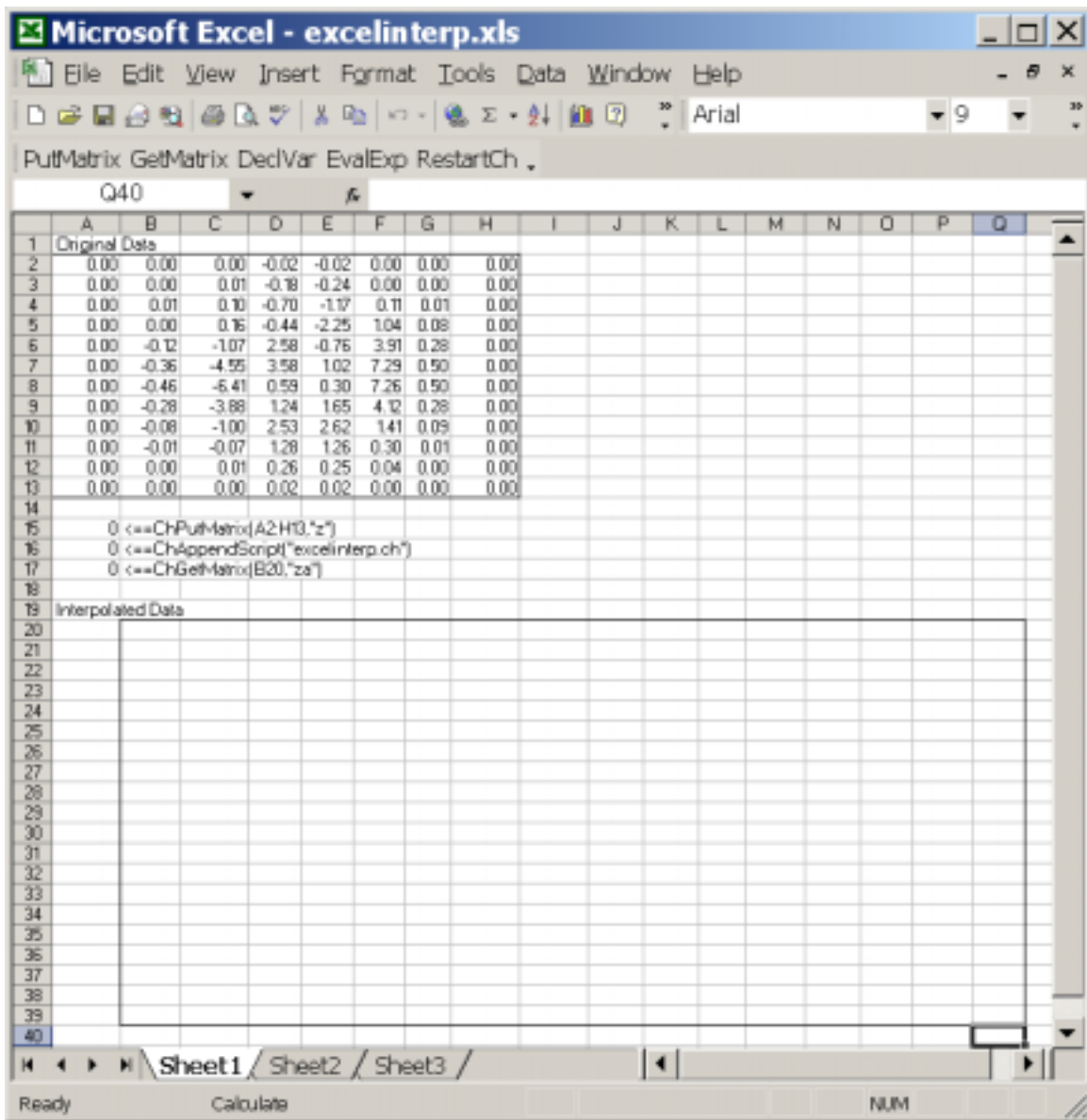


Figure 35. Before evaluation of ChAppendScript.

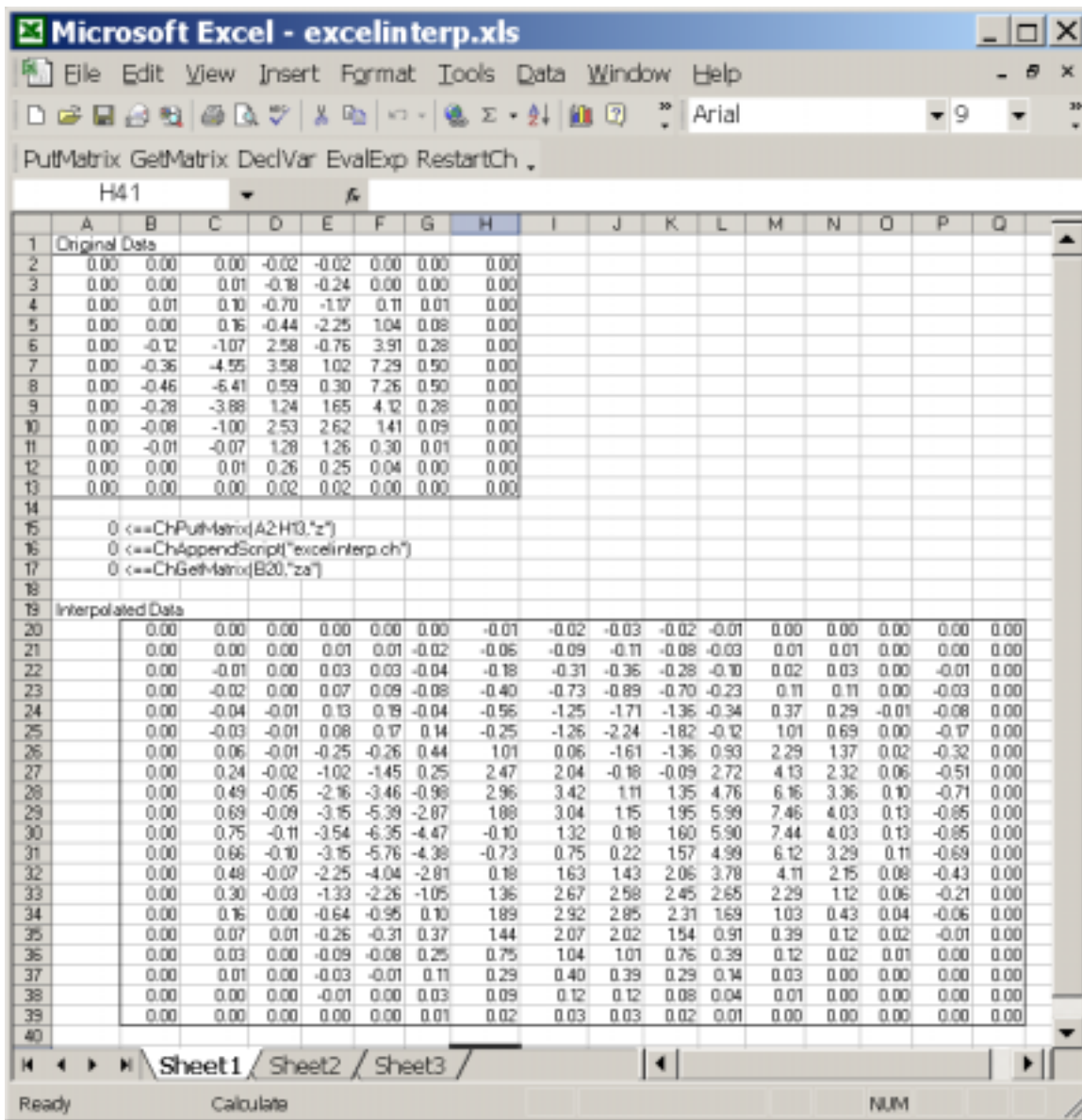


Figure 36. After evaluation of ChAppendScript.

9 References

1. SOFTINTEGRATION, INC., 2003. *The Ch Language Environment User's Guide*. <http://www.softintegration.com>.

10 Function Reference

ChExcel and VBA Functions

Function	Description
----------	-------------

ChAppendMatrix()	Append data to a Ch matrix variable.
ChAppendScript()	Append script to Ch session.
ChCalcExpr()	Returns value of scalar Ch Expression.
ChCalcExprMatrix[VB]()	Returns value of matrix Ch Expression.
ChDeclVar()	Declare a Ch variable.
ChExprEval()	Evaluate a Ch expression in Ch.
ChFunc()	Evaluate a Ch function with return of scalar.
ChFuncMatrix[VB]()	Evaluate a Ch function with return of array.
ChGetMatrix[VB]()	Get scalar or matrix value of Ch variable and place in Excel.
ChPutMatrix()	Put Excel matrix or scalar value into Ch variable.
ChRemVar()	Remove variable from Ch.
ChReinit()	Reinitialize Ch session.
ChRunScript()	Run script in Ch session.

VBA Functions

Function	Description
ChGetVar()	Get value of variable from Ch and place in VBA.
ChPutVar()	Put value of a VBA variable into Ch variable.

10.1 ChExcel and VBA Functions

ChAppendMatrix

Synopsis in Worksheet

```
int ChAppendMatrix(ChVar, Range, Direc);
```

Synopsis in VBA

```
int ChAppendMatrix(ChVar, Range, Direc);
```

Purpose

Append data to Ch variable.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

ChVar Ch variable name
Range Excel Range
Direc Direction: 0 for right, 1 for bottom

Description

The function **ChAppendMatrix()** takes a Ch variable name, a range of cells, and a direction as arguments. *ChVar* must be previously declared. The data in *Range* is appended to the right of the variable *ChVar* if *Direc* is 0, or to the bottom if *Direc* is 1.

Examples

```
ChAppendMatrix("x", A3:B5, 0)
```

Appends the data in range A3:B5 to the right of the Ch variable x.

ChAppendScript

Synopsis in Worksheet

```
int ChAppendScript(ScriptName);
```

Synopsis in VBA

```
int ChAppendScript(ScriptName);
```

Purpose

Append script to Ch Session.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

ScriptName Ch script name

Description

The function **ChAppendScript()** takes a script name as an argument. The script will be run in the current Ch session. After the script is run, any global variable declared in the script will then be accessible to ChExcel. Also, any functions that have been declared can now be used by the ChExcel. Ch variables cannot be redeclared in the Ch script. The script must reside in Ch's program path specified by the Ch variable `_path` or the user must enter a complete path to the script. This function is best used to transfer variables between Ch and Excel.

Examples

```
ChAppendScript("Myscript.ch")
```

Appends the script `Myscript.ch` to the current Ch session.

```
ChAppendScript("C:/Programs/Scripts/Myscript.ch")
```

Appends the script `Myscript.ch` to the current Ch session.

ChCalcExpr

Synopsis in Worksheet

```
VARIANT ChCalcExpr(Expr);
```

Synopsis in VBA

```
VARIANT ChCalcExpr(Expr);
```

Purpose

Evaluate a Ch expression evaluating to an integer, double or string value.

Return Value

This function returns the value of the expression on success, #ERROR# on failure.

Parameters

Expr Ch Expression

Description

The function **ChCalcExpr()** takes a Ch expression as an argument. If the expression contains a function, it should be located in the current session or in a function file specified by the system variable_ fpath in Ch. The expression *Expr* must evaluate to either an integer, double, or string data type.

Examples

```
ChCalcExpr("5 * 4 + (pow(2,3) / 2)")
```

Returns the value 24.

ChCalcExprMatrix[VB]

Synopsis in Worksheet

```
int ChCalcExprMatrix(Expr, Range);
```

Synopsis in VBA

```
int ChCalcExprMatrixVB(Expr, Range);
```

Purpose

Evaluate a Ch expression returning a integer or double matrix.

Return Value

This function returns the value 0 on success, #ERROR# on failure.

Parameters

Expr Ch Expression

Range Excel Range

Description

The function **ChCalcExprMatrix()** takes a Ch expression and a range as an arguments. If the expression contains a function, it should be located in the current session or in a function file specified by the system variable `_fpath` in Ch. The result of the expression is placed in *Range*. The expression *Expr* must evaluate to either an integer, or double matrix.

Examples

```
ChDeclVar(A1:C3, "a")  
ChDeclVar(D1:F3, "b")  
ChCalcExprMatrix("a + b", A4:C7)
```

Places the matrix resulting from "a + b" to A4:C7.

ChDeclVar

Synopsis in Worksheet

```
int ChDeclVar("ChDecl");
```

Synopsis in VBA

```
int ChDeclVar("ChDecl");
```

Purpose

Declare and initialize Ch variable.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

ChDecl Ch declaration

Description

The function **ChDeclVar()** takes a Ch declaration as an argument. If *ChDecl* was previously defined, it is overwritten. The declaration can be of any data type.

Examples

```
ChDeclVar("double array x[500];")
```

Declare Ch variable x of array double type.

```
ChDeclVar("char * y = \"hi\";")
```

Declare and initialize Ch variable y of character pointer type.

ChExprEval

Synopsis in Worksheet

```
int ChExprEval("Expr");
```

Synopsis in VBA

```
int ChExprEval("Expr");
```

Purpose

Evaluate Ch Expression.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

Expr Ch Expression

Description

The function **ChExprEval()** takes a Ch expression as an argument. If the expression contains a function, it should be located in the current session or in a function file specified by the system variable `_fpath` in Ch. It evaluates the expression and saves the changes in the Ch session.

Examples

```
ChExprEval(" linspace(x, .1, 5.0) ")
```

Linearly space the array x from .1 to 5.0.

```
ChExprEval("a = sin(y) ")
```

Set the variable a equal to the sin of y.

ChFunc

Synopsis in Worksheet

```
VARIANT ChFunc(FuncName, arg ...);
```

Synopsis in VBA

```
VARIANT ChFunc(FuncName, arg ...);
```

Purpose

Evaluate Ch function with return type double, integer, float, char, or string.

Return Value

This function returns the result of the function on success, and #ERROR# on failure.

Parameters

FuncName Ch function name

arg argument

...

Description

The function ChFunc makes a call to a function and returns the value of the function. ChFunc takes a function name and variable number of parameters as arguments. The function should be located in the current session or in a function file specified by the system variable `_fpath` in Ch. The function passed to ChFunc must have a return value of double, int, float, char, short, or string. This function can return a single value of data type double, int, float, char, short, or string. ChFunc will return the value of the function on success and #ERROR# on failure.

Examples

```
ChFunc("pow", 2, 3)
```

Returns the value 8.

```
ChFunc("plotxy", "x", "y", ""ChPlot"", ""xlabel"", ""ylabel"")
```

Generates a plot using `x` and `y` with title "ChPlot" and `x` and `y` labels.

ChFuncMatrix[VB]

Synopsis in Worksheet

VARIANT ChFuncMatrix(*FuncName*, *Range*, *arg* ...);

Synopsis in VBA

VARIANT ChFuncMatrixVB(*FuncName*, *Range*, *arg* ...);

Purpose

Evaluate Ch function with return type array and place result in worksheet.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

FuncName Ch function name

Range Excel range

arg argument

...

Description

ChFuncMatrix calls a Ch function and returns the resulting matrix. The function takes a Ch function name, range, and a variable number of parameters as arguments. The function should be located in the current session or in a function file specified by the system variable `_fpath` in Ch. The Ch function passed to ChFuncMatrix must return a matrix of double or integer type. ChFuncMatrix will evaluate the function and place the returned matrix into the specified range. The specified range must have the exact same dimensions as the result or the data will be formatted incorrectly. Make sure not to include the function cell into the range that you pass to ChFuncMatrix, or the function will be overwritten. This function returns 0 on success and #ERROR# on failure.

Examples

```
ChFuncMatrix("transpose", A3:C6, "r")
```

Places the inverse of the matrix `r` into the cells A3 through C6.

ChGetMatrix[VB]

Synopsis in Worksheet

```
int ChGetMatrix(Range, "ChVar");
```

Synopsis in VBA

```
int ChGetMatrixVB(Range, "ChVar");
```

Purpose

Get matrix or scalar variable and place into worksheet.

Return Value

This function returns 0 on success, and #ERROR# on failure.

Parameters

Range Excel range
ChVar Ch variable name

...

Description

ChGetMatrix retrieves the value of a Ch variable and places it into the Excel worksheet. The function takes two arguments, a range and a variable name. The value for a Ch variable will be placed in the specified range. If the variable is a scalar, it will be placed in the specified cell in the first argument. If the Ch matrix is smaller than the specified range, it will only fill the required range. If the Ch matrix is larger than the specified range, it will overflow into the adjacent cells. If the variable is a one dimensional array it can be placed into the worksheet as either a row or a column. To specify row format select a single row range. To specify column format select a single column range. Make sure not to include the function cell into the range that you pass to ChGetMatrix, or the function will be overwritten.

Examples

```
ChDeclVar("array double y[5][5]")  
ChGetMatrix(A4:E9, "y")
```

Place the matrix y into the cells A4 through E9.

```
ChDeclVar("int x = 10")  
ChGetMatrix(A4, "x")
```

Places the value of x into the cell A4.

ChPutMatrix

Synopsis in Worksheet

```
int ChPutMatrix(Range, "ChVar");
```

Synopsis in VBA

```
int ChPutMatrix(Range, "ChVar");
```

Purpose

Place the value of a cell or the value of a range of cells into a Ch variable.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

<i>Range</i>	Excel Range
<i>ChVar</i>	Ch variable name

Description

The function **ChPutMatrix()** takes a range of cells, and a Ch variable name as arguments. If a range is pasted to ChPutMatrix, the data will be placed into a Ch variable of array double type with dimensions corresponding to the Excel matrix. If a single cell is selected, the data will be placed into a Ch variable of double type. If the variable name passed to ChPutMatrix has been previously declared, the original variable will be overwritten.

Examples

```
ChPutMatrix(A3:B5, "y")
```

Places the matrix in cells A3 through B5 into the Ch variable *y*. Declared as array double *y*[3][2].

ChReinit

Synopsis in Worksheet

```
int ChReinit();
```

Synopsis in VBA

```
int ChReinit();
```

Purpose

Reinitialize Ch session.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

This function does not take any arguments.

Description

The function **ChReinit()** reinitializes the current Ch session. All variables and changes are permanently erased.

Examples

```
ChReinit()
```

Reinitialize the current Ch session.

ChRemVar

Synopsis in Worksheet

```
int ChRemVar(ChVar);
```

Synopsis in VBA

```
int ChRemVar(ChVar);
```

Purpose

Remove Ch variable from current session.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

ChVar Ch variable name

Description

The function **ChRemVar()** takes a Ch variable name as an argument. The variable is removed from the session along with its data. The variable name can then be redeclared.

Examples

```
ChRemVar ( "x" )
```

Remove the Ch variable x from the current Ch session.

ChRunScript

Synopsis in Worksheet

```
int ChRunScript("ScriptName");
```

Synopsis in VBA

```
int ChRunScript("ScriptName");
```

Purpose

Run script in Ch session.

Return Value

This function returns 0 on success, #ERROR# on failure.

Parameters

ScriptName Ch script name

Description

The function **ChRunScript()** takes a Ch script name as an arguments. This function reinitializes the Ch session so all variables in the current session are erased. The user will input the name of the program and ChExcel will execute it from the function `main()` if it exists. Any global variable declared in the program will then be accessible to ChExcel. Also, any functions that have been declared can now be used by the ChExcel. The program must reside in Chs program path specified by the Ch variable `_path` or the user must enter a complete path to the program. This function is best used to begin a ChExcel session to import variables from Ch to Excel.

Examples

```
ChRunScript("Myscript.ch")
```

Run the script `Myscript.ch` in the current Ch session.

```
ChRunScript("C:/Programs/Scripts/Myscript.ch")
```

Run the script `Myscript.ch` in the current Ch session.

10.2 VBA Functions

ChPutVar

Synopsis in VBA

```
int ChPutVar(ChVar, VBAvar);
```

Purpose

Put value of Ch variable and into VBA variable.

Return Value

This function returns 0 on success, and -1 on failure.

Parameters

ChVar Ch variable name
VBAvar VBA variable name

Description

ChPutVar places the value of a VBA variable into a Ch variable. The function takes two arguments, a Ch variable name and a VBA variable name. The Ch variable will be declared with the same data type and dimensions as the VBA variable. If the Ch variable has been previously declared, it will be overwritten.

Examples

```
Dim b(10,15) As Double  
ChPutVar("y", b)
```

Place the value the VBA variable b into the Ch variable y declared as "array double y[10][15]".

ChGetVar

Synopsis in VBA

```
int ChGetVar(ChVar, VBAvar);
```

Purpose

Get value of Ch variable and place into VBA variable.

Return Value

This function returns 0 on success, and -1 on failure.

Parameters

ChVar Ch variable name
VBAvar VBA variable name

Description

ChGetVar retrieves the value of a Ch variable and places it into a VBA variable. The function takes two arguments, a Ch variable name and a VBA variable name. The value of the Ch variable will be placed into the VBA variable. If the Ch variable is a one or two dimensional array, the VBA variable must be declared with matching dimensions. The VBA variable also must be declared with the same data type as the Ch variable.

Examples

```
Dim b(10,15) As Double  
ChDeclVar("array double y[10][15])  
ChGetVar("y", b)
```

Place the value the Ch variable *y* into the VBA variable *b*.

Index

Argument passing conventions, 10

ChAppendMatrix(), 12, 17, 45, **47**

ChAppendScript(), 12, 18, 45, **48**

ChCalcExpr(), 12, 14, 45, **49**

ChCalcExprMatrix(), 12, 15, 46, **50**

ChCalcExprMatrixVB, 12, 46

ChDeclVar(), 12, 46, **51**

ChExcel, **1**

ChExprEval(), 12, 16, 24, 26, 46, **52**

ChFunc(), 12, 14, 24, 26, 46, **53**

ChFuncMatrix(), 12, 15, 46, **54**

ChFuncMatrixVB, 12, 46

ChGetMatrix(), 12, 16, 46, **55**

ChGetMatrixVB, 12, 46

ChGetVar(), 12, 30, 46, **61**

ChPutMatrix(), 12, 13, 46, **56**

ChPutVar(), 12, 31, 46, **60**

ChReinit(), 1, 12, 18, 46, **57**

ChRemVar(), 12, 13, 46, **58**

ChRunScript(), 12, 18, 46, **59**

copyright, ii

Data management functions, 12

DeclVar, **4**

EvalExpr, **4**

Excel, 1

GetMatrix, **4**

installation, 1

interpolation, 38

Microsoft Excel, 1

PutMatrix, **3**

RestartCh, 1, **4**

typographical conventions, iii

VBA, 1, 30

Visual Basic for Applications, 1, 30