Advanced Spreadsheet Features for Chemical Engineering Calculations

Carl T. Lira
Michigan State University

Submitted to Chemical Engineering Education

January 17, 2000

Abstract

Spreadsheets are popular computational tools because the table-based calculations simplify program development and debugging. Use of named ranges and labels enhances the readability of formulas. Matrices can be manipulated. Macros can be used for looping and other higher-level programming needs. Examples implementing these features are discussed and available on the Internet.

Keywords: Spreadsheets, Excel, Thermodynamics

I. Introduction

Spreadsheets are popular computational tools among students and have powerful capabilities for engineering calculations[1,2,3]. Spreadsheets offer a very short learning curve compared to code-based computer techniques. Usually students have had exposure to spreadsheets in high school and they understand how to program simple mathematical formulas such as exponents, logarithms and powers. Students usually know the rules for mathematical precedence. The graphical interface and table-based results are an advantage over code-based techniques. The tables enable monitoring of intermediate results and can provide more rapid debugging in most cases. Using labels or defining variable names can overcome the disadvantage of default formulas using cell references, and then the formulas look like traditional program code. Matrices may be manipulated. A large variety of mathematical functions are available. Linear and non-linear regression can be performed rapidly. Simultaneous equations can be solved easily.

A primary advantage of spreadsheets is their widespread availability and portability. Spreadsheets are available on almost all university computers and students usually have access to them on home computers. Specialized mathematical software is normally available on a more restricted basis, however tools such as MATHCAD® are also popular and alternatives to spreadsheets[4]. Spreadsheets also offer portability between PC's and MAC's that is not possible with compiled code. Naturally, there are differences in commercial spreadsheet packages. This paper discusses features of Microsoft® Excel because of my familiarity with the capabilities. In addition, I have limited discussion to Excel97 (version 7) rather than including features that are only available to users with the latest version. This paper discusses intermediate and advanced features that are useful in preparing spreadsheets for assignments and class use. It is assumed that readers have basic knowledge of selecting cells, copying, pasting and entering formulas.

II. Techniques

To discuss programming and use of tools, we will create a collection of spreadsheets for solving problems using Raoult's law. The collection of spreadsheets is called a *workbook* and saved as a single file. The examples that we illustrate could be achieved without the use of advanced features, but the simple problem permits focus on the programming rather then on communication of the problem. For Raoult's law in a binary system of l-propanol (1) and l-butanol (2) law phase equilibria is given by

$$y_i P = x_i P_i^{Sat} \quad (1)$$

$$P = x_1 P_1^{Sat} + x_2 P_2^{Sat} \quad (2)$$

where y_i is vapor-phase mole fraction, P is pressure, x_i is liquid-phase mole fraction and P_i^{sat} is the component vapor pressure.

Entering/editing formulas. Entering formulas with cell references is the default technique. After typing an "=", Excel expects a formula, and cell references may be typed or entered automatically by clicking on the cell. The function wizard (available via the f_x button in the standard tool bar) can be used to assure the proper syntax. Formulas entered in this manner can be edited by selecting the cell and pressing F2. When editing using F2, Excel uses color coding with outlined cells to assist in proofing or editing. A more powerful technique for programming uses named or labeled ranges.

Named ranges. Traditional programmers prefer to use variable names, and this technique provides more rapid proofing and editing of formulas. In Excel, when a cell location is given a variable name, it is called a *named range*. For this example, Table 1 is constructed to contain the component data for the problem, where cells C4:E5 hold the Antoine coefficients, and C1, C2, B4: B5 will hold the variable values. To enter a variable name, the menu may be used, (Insert… Name… Define…), however it is usually quicker and more convenient to use the formula bar that appears below the toolbar. For this problem, it would be optimal to use variable names like x1, x2, yl, y2, but Excel doesn't distinguish between upper and lower case letters in variable names, and since these names refer to default cell locations they are not permitted. A workaround is to use underbars or other characters like periods in names. Move to cell B4. Note in the formula bar that the cell location is listed in leftmost box. Highlight the cell name in the formula bar by clicking the cell name in the leftmost box of the formula bar, and type x1_, then enter. Likewise give B5 the name x2_, give C1 the name T, and give C2 the name P. Note that the named ranges now appear as a drop down list in the leftmost box of the formula bar, and selecting one from the list will select the appropriate cell in the worksheet. Using the named ranges is just like programming in high-level languages. In B5 type the formula =1-xl_.

Using labels. More work would be involved to use name all the ranges, and the formulas for all components will be virtually the same except for the component number. The table headings can be used as variable names without explicitly declaring them. In addition, the use of labels must be enabled using Tools…. Options…Calculation tab, Accept labels in formulas. Make sure this option is enabled before continuing with the example.

Move to cell F4 and enter =10^(A–B/(T+'C')), then copy this formula and paste in F5. In this case the single quotes on variable C are necessary for Excel to recognize the label name C, although Excel generally doesn't require explicit specification except when the label is redundant with a default cell reference, e.g. x1 or p1. The capability to use labels without declaration and the ability to use labels that match default cell references are advantages of labels compared to named ranges.

Arrays. The total pressure may be calculated using Eq. (2). This can be done most quickly using array formulas. In cell C2 program = sum(B4: B5 * F4: F5) where cntrl + shift + enter must be pressed to explicitly tell Excel that this is an array formula (use COMMAND and RETURN on a MAC). The sum function programmed in this manner multiplies the ranges element by element and sums, which is exactly what we seek. Arrays can be named also. For example, select B4:B5 and name it “xs”. Select F4:F5 and name it “Psats”. Move to D2 and enter = sum (xs*Psats), remembering to press cntrl + shift + enter. To complete this exercise, move to G4 and enter = x * Psat/P. Copy the formula into G5.

Occasionally, variable names are defined after a spreadsheet has already been programmed using default cell names. The formulas need to be updated by applying the named cells. For this exercise, compare the formulas of C2 and D2. The formula in C2 needs to be updated. To use this feature, choose Insert…Names…Apply…click xs and cntrl + click Psats, then click OK. Now compare the formulas in C2 and D2. This method applies the names to all locations.

Matrix mathematics. Excel also permits matrix multiplications, transpositions, matrix inversions, determinants, traces, etc. Matrices can be named which simplified programming. Below the Raoult’s law table created from the earlier discussion, enter the values shown in Table 2 in the specified cells.

Give the column A7:A8 the name D. Give C7:D8 the name E. Matrix multiplication E*D results in a column, and D^T*E results in a row. The entire matrix output range must be selected before entering the formula, which requires the programmer to know the number of rows and columns in the answer. Select A10:A11 and type = MMULT(E,D), using cntrl + shift + enter. Select C10:D10 and type = MMULT(TRANSPOSE(D),E) using cntrl + shift + enter.

Solver. The Solver add-in isn't installed by default, but is a powerful tool for solving simultaneous equations and for optimization. When it is installed, it will be listed under the Tools…drop down menu. If Solver is not installed, select Tools… Add-ins… and follow the on-line instructions for installation. Discussions for the use of solver for simultaneous equations are available[5]. Solver can be used for fitting parameters. Consider non-linear regression for fitting Antoine constants. Move to the second spreadsheet of the workbook by clicking on the tab near the bottom of the window. Enter the experimental vapor pressure data for cyclohexane as shown in Table 3[6]. Note that names A, B, and C have already been used within this workbook, so A., B., and C. will be used for the Antoine coefficients. Name the cells C2 through E2 with these names. Then in D5 insert the formula = log10(Pexpt). In E5 enter =A.-B./(C. + Temp). In F5 enter the square of the error, = $(\text{log\_Pexp-log\_Pcalc})^2$. In C5, enter =10^(log_Pcalc). Then copy C5:F5 through the bottom row of the table. In F2, enter = sum(error). Then call up Solver using Tools…Solver… In the dialog box, set Target Cell F2… Equal to… Min… by changing… C2:E2. Note the options button, that permits the choice of Newton-Raphson or conjugate gradient methods and other options. Click Solve. When Solver finishes, the answer can be accepted or rejected in the final dialog box. Change the initial guesses by entering new values in C2:E2, and reconverge. Fitting of Antoine's equation is sensitive to the data selected, which provides opportunity for class discussion. Repeat the exercise deleting F9:F10 or F5:F6. Often students are surprised to learn that there aren't unique Antoine coefficients for a substance.

Protecting Cells. When a spreadsheet is to be distributed for student use some control over accidental modification of cells is usually desirable. Suppose the vapor pressure sheet is to be distributed protecting the headings and formulas, while permitting the Antoine parameters and data to be changed. All cells are unprotected by default, however protection can be enabled, locking all the cells. In the case discussed here, the cells that should be editable need to remain unlocked when the sheet is protected. To specify the unlocked state, select the cells, then Format…Cells…Protection tab…clear the "locked" checkbox and click OK. To protect the remaining cells choose Tools…Protection Protect Sheet…add a password if desired. To make the sheet more user-friendly, make the unlocked cells stand out by changing the color of the font or use borders available via the Format…Cells…options. The spreadsheet must be unprotected to use the Solver add-in.

Pivot Tables. Pivot tables are useful for counting, adding, averaging, or otherwise manipulating tabular data that match a certain criteria. One case useful for instructors is the summary of exam scores. Move to Sheet3 and enter the list of scores shown in Table 4. Select A1:B10, then choose Data…Pivot Table Report…Microsoft Excel list or database…$A$1:$B$10…At this point, the pivot table is to be constructed with a column of scores and an accompanying column of the number of occurrences of each score. Drag the score button to the ROW field. Drag another copy of the Score button into the DATA region. The button will change to the default Sum of Score. For score reporting, the count of scores is desired. Double-click the button in the Data region and use the dialog box to select the COUNT function and click OK to accept it. Click Next, and then specify to place the pivot table on the existing sheet in cell D2.

Plotting a Histogram. Excel doesn't offer a histogram plot option through standard chart types, however with a few extra selections it can be obtained. Highlight the scores and counts in the pivot table. Select Insert…Chart…XY(Scatter)…choose the plot with points only…Follow the dialog boxes usually accepting the default values until the dialog box for Chart Options where the axis labels can be typed in. Enter the desired labels and then place the chart as a new sheet. On the new chart, right-click on one of the data points to select the data series, select Format Data Series…Y Error Bars tab… select minus and set the Percentage to 100%.

Comments. Comments are "yellow sticky notes" for spreadsheets. Select the cell that needs description, then Insert…Comment… After completing the comment, the comment can be hidden using View…Comments. Cells with hidden comments are designated with a red triangle in the upper right corner. The hidden comment can be viewed temporarily by hovering the mouse over the cell.

Auditing. Understanding a formula or tracking an error in a spreadsheet is facilitated with the auditing feature. Move to cell E5 of Sheet2, unprotect the sheet, then perform, Tools…Auditing…Trace Precedents. The resulting arrows show the cells referenced in the formulas. Other features include finding all cells dependent on a given cell (useful before deleting a cell), and error tracing.

II. Macros and Visual Basic.

Transcendental equations or single loops can be solved in principle by using circular references. However, I encountered difficulty in running circular references running Excel 4.0 under Windows 95 (possibly due to a bug?) and I have generally avoided them subsequently. Macros can accomplish the same effect by pasting successive substitutions to the cell at the beginning of the loop[5]. Macros range from playback of recorded keystrokes to high-level programming. Recording of keystrokes simply creates Visual Basic code that can be later edited for minor cleanup. On-line documentation of Visual Basic functions is thorough but lacks indexed descriptions of how to code common engineering tasks. User guides are available[7-8], but are usually focused on business users rather than engineering. Usually the code needed for engineering requires reading of values from a spreadsheet, calculations based on the values, and then pasting of the results back into a spreadsheet. In addition, it may be desirable to use Solver within a macro. Examples are available for free download on the Internet[9]. It is possible to paste most existing legacy code in the Visual Basic Editor and then modify the I/O statements.

Also, dropdown menus appear automatically if typing code. The object browser within the Visual Basic Editor is helpful in building statements with correct syntax.

Macros for inserting or deleting a row are given in Table 5. Row insertion or deletion are trivial functions, but these examples include other useful functions including branching using an IF statement, unprotecting the sheet, and protecting the sheet after the action. The Msg box function permits warnings or decision input in a dialog box. To access visual basic within Excel use Tools…Macro >, then Visual Basic Editor. Add a macro module using Insert…Module. Type one of these macros, return to the spreadsheet using the Excel icon on the toolbar, and run the macro using Tools…, Macro >, > Macros…highlight the macro name and click options. This dialog box permits a keystroke to be assigned to the macro. Return to the Macro Dialog box and click the Run button.

Other useful functions are summarized in Table 6 along with a description. Note that values of specific cells on specific sheets can be used within the macro. Values calculated within the macro can be printed onto the spreadsheet.

Available Examples. Several Excel spreadsheets are available for internet download that illustrate further the use of features discussed in this paper as listed in Table 7. These examples may serve as templates for developing other exercises, and may also serve to clarify by example implementation of the features.

Limitations. Excel has a wide variety of mathematical, statistical[10], database, and financial functions. Advanced feature use available including imaginary mathematics, error functions, Bessel functions, Fourier transforms. Even the Gamma function is available via GAMMALN. Math software has features that are not in Excel such as numerical or symbolic integration, although numerical integration can be handled by a prepared spreadsheet[11]. One limitation that is sometimes encountered is that Excel only permits one definition of a variable per workbook, which can be annoying if several worksheets are variations of each other. However, labels are local to individual sheets, so they can be used in similar formulas on different sheets. The primary limitation of labels is the need to use them within tables. Another limitation is that when a calculation within a macro results in a #NUM! or #DIV/0 error, the undo feature might not be available. The easiest solution is usually to reload the file and reinitialize the calculation.

Summary. This paper has presented an overview of features of Microsoft Excel for engineering education. This paper has focused on features that I have found useful and I have often wished I had found summarized in a handbook or manual. Students appreciate the ease of use of the tabular interface and find debugging easy and the availability of intermediate values helpful. Examples implementing many of the features are available in the software for Introductory Chemical Engineering Thermodynamics[5]. The advanced features discussed here should be useful for your courses.

	A	B	C	D	E	F	G
1		T(C)	105				
2		P(mmHg)					
3	comp	x	A	B	C	Psat	yP
4	1	0.1	8.37895	1788.02	227.438		
5	2		7.81028	1522.56	191.95		

Table 1. Table to hold information for examples using labels and named ranges.

	A	B	C	D	E	F
7	1		3	4		
8	2		5	6		

Table 2.Table for illustrating matrix multiplication.

	A	B	C	D	E	F
1			A	B	C	OBJ
2			7.5	1500.0	273.0	
3	(C)	(mmHg)				
4	Temp	Pexpt	Pcalc	log_Pexp	log_Pcalc	Error
5	6.7	40				
6	14.7	60				
7	25.5	100				
8	42.0	200				
9	60.8	400				
10	80.7	760				
11						

Table 3. Table for fitting Antoine coefficients.

	A	B	C	D	E
1	Student	Score		Histogram Data	
2	1	100			
3	2	98			
4	3	37			
5	4	67			
6	5	84			
7	6	98			
8	7	67			
9	8	100			
10	9	80			

Table 4. Illustration of creating histogram of exam scores.

`` Sub DeleteRow() ' This macro warns the user that a row is about to be deleted using ' a dialog box with a Yes and No button, and if the user clicks Yes, ' the row will be deleted.   Answer = MsgBox("You are about to delete the select rows. This _ cannot be undone. Are you sure?", vbYesNo, "Delete Warning")   If Answer = vbNo Then   Exit Sub   End If   Selection.EntireRow.Delete End Sub ```

```
Sub DeleteRow()
' This macro warns the user that a row is about to be deleted using
' a dialog box with a Yes and No button, and if the user clicks Yes,
' the row will be deleted.
    Answer = MsgBox("You are about to delete the select rows. This _
cannot be undone. Are you sure?", vbYesNo, "Delete Warning")
    If Answer = vbNo Then
    Exit Sub
    End If
    Selection.EntireRow.Delete
End Sub
```

```
Sub InsertRow()
' This macro checks if the current cursor location is greater than
' row 8, displays a message box if it is, otherwise the macro
' unprotects the sheet from which the macro was run, deletes the row,
' then reprotects the sheet.
    If (Selection.Row < 8) Then
    MsgBox "Selected row must be below 7 to insert row! Request _
Aborted.", vbExclamation, "Warning"
    Exit Sub
    End If
    ActiveSheet.Unprotect
    Selection.EntireRow.Insert
    ActiveSheet.Protect DrawingObjects:=True, Contents:=True, _
Scenarios:=True
End Sub
```

```
Sub PrintRange()
' This macro finds the last contiguous filled row in column A below A6
' and sets the print range from row 1 to that row.
LastRow = Range("a6").End(xlDown).Row
ActiveSheet.PageSetup.PrintArea = "$1:$" & LastRow
End Sub
```

Table 5. Three different macros illustrating message boxes, inserting rows, deleting rows, finding contiguous regions, and setting print ranges. Statements ending with “_” are continued on the next line, and the “_” should be omitted if the entire command is typed on a single line.

Macro Statement	What statement does
`StartRow = _` `Worksheets("Sheet1").Range("K14").Value`	Sets the macro variable StartRow to the value stored in Cell K14 of the sheet titled “Sheet1” so that it can be used in code for looping, etc.
`Range(Cells(CurrentRow, 1), _` `Cells(CurrentRow, 3)).Copy`	Copies columns A-C of the row given by variable CurrentRow to the clipboard.
`Range("B9").PasteSpecial _` `Paste:=xlValues, Operation:=xlNone, _` `SkipBlanks:=False, Transpose:=False`	Paste into cell B9 (and relevant contiguous cells) the values from the clipboard, but no formats or other special changes.
`Range("B6").Formula = "='Sheet1'!D" & _` `CurrentRow`	Inserts into Cell B6 the formula “='Sheet1'!D*x*”, where *x* is the value of variable CurrentRow
`Range("B9").Value = CurrentRow`	Puts the value of variable CurrentRow into Cell B9 of the active sheet.

Table 6. Illustration of several useful Visual Basic macro statements. Statements ending with “_” are continued on the next line, and the “_” should be omitted if the entire command is typed on a single line.

Features	Workbook
Named Ranges, Labels, Comments	Preos.xls, Actcoeff.xls, Kcalc.xls, Gammafit.xls, Prfug.xls, Residue.xls, Virialmx.xls
Circular References	Dewcalc.xls
Macros that loop and create tables of results. Residue.xls calls Solver from within macro.	Residue.xls, Integration.xls
Matrix multiplication	Prfug.xls and Sheets UNIFAC(VLE), UNIFAC(LLE) and UNIQUAC5 in Actcoeff.xls.
Solver for non-linear regression	Gammafit.xls
Solver for simultaneous equations	Rxns.xls

Table 7. Spreadsheets available for download that illustrate the principles discussed in this paper. Integration.xls is available online[11] as well as the other files[9].

References

[1] P.E. Savage, *Chem. Eng. Ed.*, **29**, 262 (1995)

[2] Burns, M.A., Sung, J.C., *Chem. Eng. Ed.*, **30**, 68 (1996)

[3] Rosen, E.M., *CACHE News*, (No. 47), 1 (1998), (No. 48), 4 (1999) and (No. 49), 1 (1999)

[4] Sandler, S.I., *Chem. Eng. Ed.*, **31**, 18 (1997)

[5] Elliott, J.R., Lira, C.T., *Introductory Chemical Engineering Thermodynamics*, Prentice-Hall, 1999, Appendix A, also available at http://www.egr.msu.edu/~lira/thermtxt.htm.

[6] Perry, R.H, Chilton, C.H., eds., *Chemical Engineer's Handbook*, 5th ed., McGraw-Hill, **3**-52, (1973).

[7] E. Boonin, *Using Excel Visual Basic for Applications*, Que® Corporation, Indianapolis, IN, 1996. This book is written for Excel6, but many of the features still apply. The main difference in Excel7 is the implementation of the modules in the Visual Basic editor instead of an extra module worksheet.

[8] Blattner, P., Ulrich, L., Cook, K., Dyck, T., *Special Edition Using Microsoft® Excel 2000*, Que® Corp., Indianapolis, IN, 1999.

[9] Software for Ref. 1, http://www.egr.msu.edu/~lira/readcomp.htm.

[10] Mitchell, B.S., *Chem. Eng. Ed.*, **31**, 194 (1997)

[11] See the link *numerical integration* link under heading *Additional Educational Aides* on page http://www.egr.msu.edu/~lira/thermtxt.htm.