MICROSOFT EXCEL: A SIMPLE WAY OF TEACHING COMPLEX TASKS

Luiz Fernando de Moura*
Departamento de Engenharia Química - Universidade Federal do Sao Carlos

Abstract. The main goal of this article is to show how Microsoft Excel can be an easy-to-operate tool to develop undergraduate chemical engineering students' programming skills. By permitting a 'natural language interface' allied to its complete function library, Microsoft Excel allows transparency and a significant shortening of codes. Two case studies -- cooling of a cylinder rod and a BTX multicomponent distillation -- are presented to show how the teaching of Heat Transfer and Unit Operations can be improved.

Keywords: Excel, Unit Operations, Chemical Engineering Education.

1. Introduction

1.1. Identifying the problem.

Chemical Engineering courses are still facing the dilemma of either teaching programming skills to undergraduate chemical engineering students or not doing so. Part of the second option is based on the fact that after graduation, most chemical engineers will never type a line of code for the rest of their professional lives. Edgar (2001) cites a poll among 84 departments where 16% of respondents do not have computer programming classes in their curricula anymore. A question arises: are programming classes really necessary? In this article, we shall explore the pros and cons of both options.

Commercial packages can eliminate the necessity of writing code and are very efficient. On the other hand, their prices can be prohibitive to many schools. Even if those limitations were to be overcome, the risk of exchanging teaching for training would remain. Those courses would be better taught by manufacturers 'in company' to engineers who work with a specific package.

Conventional languages. For many years FORTRAN has been the main option in most of the Chemical Engineering courses, but this language has not been successful in working with Graphical User Interfaces (GUIs), which are typical of modern software. Options like C, C^{++} or Visual Basic initially experienced resistance among teachers, who were not familiar with Object-Oriented Programming. To worsen the scenario, in many countries these languages were taught to freshmen by Computer Science departments. Disagreements between the "preferences" of both departments were very common, which resulted in unmotivated teachers and thus, unmotivated students. Even Chemical

* Address: Departamento de Engenharia Química, UFSCar – 13565-905 Sao Carlos – Brazil
E-mail: mouralf@power.ufscar.br

Engineering departments had a significant number of 'programming-phobic' teachers, creating lacunas in the usage of programming throughout the course. Such teachers generated more 'programming-phobic' students, some of whom became teachers, and so forth.

But the real problem with conventional languages is that they are difficult to learn because students don't have time, and developing codes inside the classroom has been a boring and annoying job both to teachers and to students.

No programming classes. In our point of view, this option should never be taken into consideration. Engineers are not graduated to reproduce existing technology, and technology development is not limited to companies that can buy expensive software. Programming skills are necessary today, and will always be. The only problem is to foster a programming 'culture'.

Higher-level languages. Languages such as Maple or Matlab can be good options and must be taken into account. They don't need extensive programming codes and have good internal function libraries. The only problem is that they are not popular in industry.

1.2. Aim

The aim of this work is to show how Microsoft Excel can be adopted as a teaching tool. The main reasons for such adoption are:

- Its cost-benefit ratio is relatively low when compared with software packages.
- It is included in the Microsoft Office Suite, which lowers its cost.
- The powerful programming language -- Visual Basic for Applications (VBA) -- is fully integrated.
- VBA allows the ability to call routines that are written in other languages (e.g., FORTRAN, C++, MATLAB, MATHCAD, etc).
- VBA is procedural and allows Object Oriented Programming (OOP).
- Excel can be found virtually everywhere: in schools, offices and factories. Graduates do not have difficulties in finding the software in their jobs.
- It has a major function library, which includes matrix functions, algebraic functions, statistical functions, and so forth. Engineering functions allow user to operate with Bessel functions, error functions, etc, without difficulty.
- Contrary to many people's belief, Excel allows natural, algebraic language such as "=a*ln(x)+b*cos(y)".
- Excel’s graphical interface is sensitive to calculations at the spreadsheet level. This fact (1) allows dynamic interpretation of results when working with "what if" problems and (2) facilitates understanding of profiles and spatial gradient phenomena.

- When compared with classical languages, the usage of Excel reduces code programming, thus allowing more time for understanding and interpreting results.
- Most engineering teachers have some experience with spreadsheets.
- Many students use it throughout the curriculum. This fact creates a number of "tutors" outside the classroom.

Despite these advantages, Excel is being underused in engineering courses. Only a lack of knowledge about its possibilities can justify this situation. Excel is a powerful tool and is available in most schools. Rosen (2003) recommends a two-credit-hour course at the freshmen level, where the use of spreadsheets, VBA language, and numerical methods would be introduced. This could be accomplished with its use throughout the curriculum. To illustrate this point of view, two case studies are presented: 1) transient heat transfer in a semi-infinite rod and 2) multicomponent distillation-tower design. The first case study shows how Excel can deal with sophisticated mathematics in a simple way, and the second shows how a 'what if' analysis tool can be developed with few lines of code.

2. Case Studies.

2.1. Cooling in Transient Regime.

This study was intended to show how easy Excel works with complex, computing tasks, such as Bessel functions. Many chemical engineering problems deal with cylindrical shapes, with solutions that result in Bessel equations. Teaching this subject is usually a major challenge to be overcome by teachers. Besides, engineers avoid them by using numerical approaches such as finite differences. In this particular case, Gurney-Lurie charts were used in the past, but with a significant lack of precision.

Statement. A semi-infinite 5-cm diameter steel rod at 800ºC must be cooled in air down to 20ºC. How long does it take for the center temperature to reach 100ºC? Assume the following values:

$k = 20 W.m^{-1}.K^{-1}$ (thermal conductivity);

$\alpha = 6.11 \times 10^{-6} m^2.s^{-1}$ (thermal diffusivity); and

$h = 150 W.m^{-2}.K^{-1}$ (individual convective coefficient).

Solution. Assuming only radial heat transfer, the energy balance reduces to:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right) \qquad (1)$$

where r is a dimensionless radial position ($r = 0$ at center and $r = 1$ at the surface), t is the time and T is the temperature at points (t, r). The initial condition is:

$$T(r,0) = T_i \qquad (2)$$

and the two boundary conditions are:

$$\left.\frac{\partial T}{\partial r}\right|_{r=0} = 0 \tag{3}$$

$$-k\left.\frac{\partial T}{\partial r}\right|_{x=L} = h\left[T(R,t) - T_\infty\right] \tag{4}$$

where T_i is the initial rod temperature and T_∞ is the surrounding air temperature. The analytical solution is the Fourier series:

$$\theta = \sum_{n=1}^{\infty} C_n . exp\left(-\zeta_n^2 . Fo\right) . J_0\left(\frac{\zeta_n . r}{R}\right) \tag{5}$$

where,

$$Fo = \alpha t / R^2 \qquad \text{(Fourier number)} \tag{6}$$

$$\theta = (T - T_\infty)/(T_i - T_\infty) \tag{7}$$

$$C_n = \frac{2}{\zeta_n}\frac{J_1(\zeta_n)}{\left(J_0^2(\zeta_n) + J_1^2(\zeta_n)\right)} \tag{8}$$

J_0 and J_1 are Bessel functions of the zero and the first-order first-kind, respectively. ζ_n are the roots of the transcendental equation:

$$\zeta_n \frac{J_1(\zeta_n)}{J_0(\zeta_n)} = Bi \tag{9}$$

where,

$$Bi = \frac{h.R}{2k} \qquad (Bi \text{ = Biot number}) \tag{10}$$

VBA may be brought up by hitting the keys, Alt+F11. The following code can be introduced in a Module. The only subroutine allowable to the user is *Main*. The others are all hidden. Subroutine *Layout* is created to insert titles and format fonts. Subroutine *Naming* is created to name cells in spreadsheet.

It is best to work with natural language rather than A1 notation. Formulas will be more 'transparent' for students. *Coloring* is the subroutine that plades colors inside spreadsheet cells. It iss becoming an international practice to color cells allowed to the users with yellow color in the background. *DesignVars* uses an interesting way introduce values: the object *InputBox*. It is not necessary for a

beginner to know anything about forms. Subroutine *Formulas* uses two similar ways to introduce formulas in a range, *Formula* and *FormulaR1C1*. These properties are exclusive for Excel in the same manner as is the object *Range*. Finally, *FindingRoots* is the only 'numerical method' used.

```
Sub Main()
   Layout
   Naming
   Coloring
   DesignVars
   Formulas
   FindingRoots
End Sub

Private Sub Layout()
'This routine is not necessary at all. Layout
'can be performed directly in the spreadsheet.
   [A1].Value = "h": [B1].Value = "D": [C1].Value = "k":[D1].Value = "Bi"
   [E1].Value = "alpha": [F1].Value = "time": [G1].Value = "Fo"
   [H1].Value = "radial":[I1].Value = "Tinf": [J1].Value = "Tini"
   [K1].Value = "Tfinal": [A4].Value = "n": [B4].Value = "z."
   [B4].Font.Name = "Symbol": [C4].Value = "TF": [D4].Value = "Cn"
   [E4].Value = "q": [E4].Font.Name = "Symbol"
End Sub

Private Sub Naming()
   [A1:K2].CreateNames True
   [A5:A8].Name = "n": [B5:B8].Name = "zeta"
   [C4:D8].CreateNames True
   [E5:E8].Name = "Series": [E9].Name = "theta"
End Sub

Private Sub Coloring()
'It is becoming usual that design variables values
'be introduced in yellow cells
   [h].Interior.ColorIndex = 36: [D].Interior.ColorIndex = 36
   [k].Interior.ColorIndex = 36: [alpha].Interior.ColorIndex = 36
   [Time].Interior.ColorIndex = 34: [radial].Interior.ColorIndex = 36
   [Tinf].Interior.ColorIndex = 36: [Tini].Interior.ColorIndex = 36
   [Tfinal].Interior.ColorIndex = 35
End Sub

Private Sub DesignVars()
'This routine is not necessary at all. Values
'can be introduced directly in the spreadsheet.
    [h] = InputBox("Type a value in W/(m².K)", _
    "Individual heat coefficient", 150)
    [D] = InputBox("Type a value in m", "Diameter", 0.05)
    [k] = InputBox("Type a value in W/(m.K)", _
    "Thermal donductivity", 20)
    [alpha] = InputBox("Type a value in m²/s", "Thermal diffusivity", _
    "6.11E-6")
    [Time] = 100
    [radial] = InputBox("Type a value between 0(center) and 1(surface)" _
    , "Radial position.", 0)
    [Tinf] = InputBox("Type a value in K", "Surrounding temperature", 20)
    [Tini] = InputBox("Type a value in K", "Initial temperature", 800)
    For I = 1 To 4: [n].Cells(I, 1).Value = I: Next I
'Students must be encouraged to discover how these guesses were chosen:
    [zeta].Cells(1, 1) = 0.1: [zeta].Cells(2, 1) = 3
    [zeta].Cells(3, 1) = 6: [zeta].Cells(4, 1) = 4
End Sub

Private Sub Formulas()
    [Bi].Formula = "=h*D/4/k"
    [Fo].Formula = "=4*alpha*time/D^2"
    [TF].FormulaR1C1 = "=zeta*BESSELJ(RC[-1],1)/BESSELJ(RC[-1],0)-Bi"
    [Cn].FormulaR1C1 = _
    "=2/zeta*BESSELJ(RC[-2],1)/((BESSELJ(RC[-2],0))^2+" & _
     "(BESSELJ(RC[-2],1))^2)"
```

```
    [Series].Formula = "=Cn*EXP(-(zeta^2*Fo))*COS(radial*zeta)"
    [theta].Formula = "=SUM(Series)"
    [Tfinal].Formula = "=Tinf+theta*(Tini-Tinf)"
End Sub

Private Sub FindingRoots()
   For I = 1 To 4
         Range("TF").Cells(I, 1).GoalSeek 0, Range("zeta").Cells(I, 1)
   Next I
'Solving...
   Range("Tfinal").GoalSeek 100, Range("time")
End Sub
```

The necessary time to cool the rod is 1285 s. Results are not as important as building the code and interpret results. Students must be encouraged to build charts for a better understanding of the effects of changing the material (α), or the external resistance to heat transfer (h). They must be encouraged to change the code, too. For instance, they can create a chart of temperatures along the rod radius fixing *t* at 1285s.

Another aspect that can be emphasized is the numerical approach. Students tend to mystify solutions by computer. A change in the initial value of the time can lead to an erroneous solution with no warnings in the way code was built.

An important aspect is that code is completely portable. One can export it, say, as *Module1.bas* and import it into a blank worksheet. Execution through the Macros menu should result in the same solution. This allows code sharing among many people at the same time.

2.1. Design of a multicomponent distillation column.

This study intends to show how Excel works with iterative calculus and context-sensitive graphics. This can be a good option for teaching Unit Operations or Process Design.

Statement. Define the number of equilibrium stages and the optimum feed stage to distill a saturated liquid mixture containing 60% of Benzene, 30% of Toluene and 10% of meta-xylene. The upper stream must contain 99.5% of Benzene and 0.5% of Toluene and the lower stream, 0.5% of Benzene (maximum). Make a graphic showing the temperature and concentration profiles along the column. Suppose ideal behavior both in liquid and in vapor phases. All percentages are given on a molar basis and a reflux ratio 2 is used.

Solution – If we have ideal behavior in both phases we can use Raoult's relationship,

$$K_n = \frac{y}{x} = \frac{Pv}{P} \qquad (11)$$

where K_n is the partition coefficient at stage *n* and *y* and *x* are molar fractions of each substance in the vapor and liquid phases, respectively. *Pv* is the saturation pressure at equilibrium stage temperature

and P is the total pressure, which is set to 1 atm, here. For each substance the Lewis-Matheson method uses one of two equations for stage n, called operational lines, one of them for stripping and the other for the rectification branch of the column. Those equations relate the above stage liquid fractions x_{n+1} to y_n, the actual stage vapor fraction.

$$x_{n+1} = \frac{V_S}{L_S} y_n + \frac{B\,x_B}{L_S} \qquad \text{if n is under the feed plate} \qquad (12)$$

$$x_{n+1} = \frac{V_R}{L_R} y_n - \frac{D\,x_D}{L_R} \qquad \text{if n is above the feed plate} \qquad (13)$$

V_S and L_S are the respective vapor and liquid molar flows in the lower part of the column (stripping). Both are supposed to be constant along that region. V_R and L_R are the corresponding flows under the feed plate. D and B are the distillate and bottom molar flows, respectively. Mass balances applied to entire column results in the following equations:

$$F = D + B \qquad (14)$$

$$F\,z = D\,x_D + B\,x_B \quad , \qquad (15)$$

where z, x_D and x_B are the molar fractions for each substance in the feed, in the top and in the bottom streams.

Students should build a layout identical to the one in Figure 1 and name the cells with self-explanatory names such as "F", "D", etc. Subroutine *Naming* is not necessary, but it shows the choices made here.

Arial 10 N I S

TK = =TC+273,15

	A	B	C	D	E	F	G	H	I	J	K
1	F	100,0	L_R	120,2							
2	D	60	V_R	180,3			Calculate				
3	B	39,9	L_S	220,2							
4	r	2	V_S	180,3							
5	P	1									
6											
7	Comp.	Distil.	Feed	Bott.	Stage	T	Pv_i	K_i	x_i	y_i	x_{i+1}
8		x_D	z	x_B		(°C)	(atm)				
9	Benz.	0,995	0,600	0,005			1,003	1,003	0,995	0,998	0,999
10	Tol.	0,005	0,300	0,744	18	80,1	0,392	0,392	0,005	0,002	0,000
11	Xyl.	0,000	0,100	0,251			0,153	0,153	0,000	0,000	0,000
12					T (K)	353,3			Σ	1,000	1,000
13											

Figure 1. *Layout for BTX Distillation.*

A command button must be drawn inside the spreadsheet and the following code must be associated with it:

```
Private Sub CommandButton1_Click()
Main
End Sub
```

The strategy here is to guess a value for the temperature, following the calculation of the three partition coefficients. That temperature must be changed until the bubble point is reached:

$$\sum_{1}^{3} y_i = 1 \qquad (16)$$

After the bubble point is reached, the x_{i+1} values should be calculated from the operational lines and the resulting values substituted for the former ones in the vector x_i. VBA may be brought up by hitting Alt+F11. The only subroutine accessible to the user is *Main*. Another, easier way of executing it is by clicking the command button. This work is intended to put students in contact with graphical tools that may be interesting in other languages such Visual Basic or C. Subroutine *Main* is used to call other subroutines. Subroutine *Naming* does the job of naming cells for later use with natural language and subroutine *Begin* initiates formulas and cells' values. *BubblePoint* is the subroutine that makes bubble point calculations. Finally, *ChangeX* exchanges x_{n+1} and x_n values.

```
Dim Stripping As Boolean

Public Sub Main()
    Naming
    Flag:
    BubblePoint
    If [y.Bz] >= [xD.Bz] Then GoTo GettingOut
    ChangeX
    If Stripping Then Verify
    GoTo Flag
    GettingOut:
    BubblePoint
End Sub

Private Sub Begin()
    Stripping = True
    [TC] = 80
    [SN].Value = 1: [TK].Formula = "=TC+273.15"
    [D].Formula = "=F*(z.Bz-xB.Bz)/(xD.Bz-xB.Bz)"
    [B].Formula = "=F-D"
    [LR].Formula = "=rr*D": [VR].Formula = "=LR+D"
    [LS].Formula = "=LR+F": [VS].Formula = "=VR"
    [xB].Cells(2, 1).Formula = "=(F*z-D*xD)/B"
    [xB].Cells(3, 1).Formula = "=(F*z-D*xD)/B"
    Range("Pv").Cells(1, 1).Formula = "=40400*EXP(-3746/TK)"
    Range("Pv").Cells(2, 1).Formula = "=51850*EXP(-4166/TK)"
    Range("Pv").Cells(3, 1).Formula = "=87500*EXP(-4683/TK)"
    [K].Formula = "=Pv/P"
    [x].Value = [xB].Value: [y].Formula = "=K*x"
    [Bubble].Formula = "=SUM(y)"
    [Bubble].Offset(0, 1).Formula = "=SUM(xp)"
```

```
    [xp].Formula = "=VS/LS*y+B/LS*xB"
    Range("Z2:AH65536").ClearContents              'clean graphic table
End Sub

Private Sub BubblePoint()
    [Bubble].Activate: [Bubble].GoalSeek 1, Range("TC")
    PlotChart
End Sub

Private Sub ChangeX()
    [x].Value = [xp].Value
    [SN] = [SN] + 1
End Sub

Private Sub Verify()
Dim Response
    If [x.Bz].Value >= [z.Bz].Value Then
        [xp].Formula = "=VR/LR*y-D/LR*xD"
        Stripping = False
        Response = "Feed stage:  " & [SN]
        Response = MsgBox(Response, , "Optimal Stage")
    End If
End Sub

Private Sub PlotChart()
Dim N As Integer, rng As Range
    N = [SN]
    Set rng = Range("Z1")
    rng.Offset(N, 0).Value = [SN].Value
    rng.Offset(N, 1).Value = [TC].Value
    rng.Offset(N, 2).Value = [x].Cells(1, 1).Value
    rng.Offset(N, 3).Value = [x].Cells(2, 1).Value
    rng.Offset(N, 4).Value = [x].Cells(3, 1).Value
    rng.Offset(N, 5).Value = [y].Cells(1, 1).Value
    rng.Offset(N, 6).Value = [y].Cells(2, 1).Value
    rng.Offset(N, 7).Value = [y].Cells(3, 1).Value
End Sub

Private Sub Naming()
    [B1].Name = "F": [B2].Name = "D": [B3].Name = "B"
    [B4].Name = "rr": [B5].Name = "P"
    [D1].Name = "LR": [D2].Name = "VR"
    [D3].Name = "LS": [D4].Name = "VS"
    [B9:B11].Name = "xD": [B9].Name = "xD.Bz"
    [C9:C11].Name = "z": [C9].Name = "z.Bz"
    [D9:D11].Name = "xB": [D9].Name = "xB.Bz"
    [E9].Name = "SN": [F9].Name = "TC": [F12].Name = "TK"
    [G9:G11].Name = "Pv": [H9:H11].Name = "K"
    [I9:I11].Name = "x": [I9].Name = "x.Bz"
    [J9:J11].Name = "y": [J9].Name = "y.Bz"
    [K9:K11].Name = "xp": [K9].Name = "xp.Bz"
    [J12].Name = "Bubble"
    Begin
End Sub
```

Boolean variable *Stripping* changes its value to False when the liquid benzene concentration exceeds its concentration in the feed. At that point, both the operational line equation and the variable value are changed. Subroutine *Verify* does the job. A *Message Box* warns the user when the optimal stage is reached.

PlotChart is responsible for plotting graphics that represent the profiles along the 18 stages found. A sample result is shown in Figure 2, which is created inside of Excel.

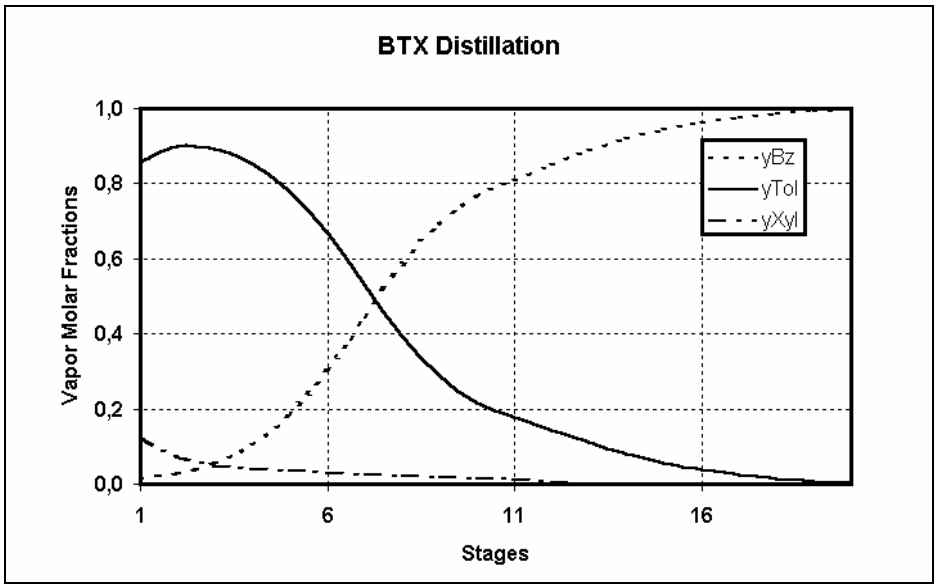

Figure. 2. *Vapor profile along the column.*

Students should be stimulated to change design variables, mainly the reflux ratio. A choice of $r = 1$ will provoke a runaway. The concept of minimum reflux ratio can be reinforced.

These studies intend to show how Microsoft Excel can improve the use of computers in teaching undergraduate students. In fact, the examples presented here could be used without any line of code. The intention is to show that VBA is a friendly language, though most of the calculations could be performed at the spreadsheet level.

References

Edgar, T. F. (2001). Computer Programming in the Chemical engineering Curriculum. *CACHE News*,53, Fall 2001. Austin, TX.

Rosen, E. M.(2003). On the Choice of VBA. *CACHE News*,56, Spring 2003. Austin, TX.

Incropera, F. P., DeWitt, D.P.(2001). Fundamentals of Heat and Mass Transfer, Fifth Edition, John Wiley and Sons, New York.

Wankat, P.C. (1988) Equilibrium staged separations. New York, Elsevier Science.