THE CHEMICAL ENGINEER'S TOOLBOX
A Glass Box Approach to Numerical Problem Solving

Daniel G. Coronell and M. Hossein Hariri
Rose-Hulman Institute of Technology • Terre Haute, IN 47803

An introductory computer programming course has been a permanent staple of the undergraduate engineering curriculum ever since the appearance of mainframe computers. While the principal programming language that is taught has changed over the years, much remains the same. The course is typically taken during the freshman year, well before any of the core engineering courses. Students wonder why they have to take the class when they're in it, and this viewpoint is unfortunately reinforced in retrospect as they are rarely if ever called upon to apply their programming skills. When they are asked to solve numerical problems, a commercial software package is typically used in a "black box" fashion, thus perpetuating the notion that their computer programming course was a waste of time.

Hardware and software advances and curricular adjustments have improved this situation. Laptop computers are used in the classroom to a greater extent. Access to computing facilities, in general, is less of an issue than ever before. In addition, the availability of programming languages such as Matlab[1] that come equipped with built-in solvers and graphics makes it easier to apply computer programs to solve engineering problems. The use of such software, however, is only effective if one can broker a departmental or institutional arrangement where a common numerical software package is used in related engineering courses. Another consideration is that many commercial numerical software packages are designed to be used in a "black box" mode. This can foster their use in a way that does not require the user to understand the underlying numerical algorithms. In an academic setting, this may not meet all of the intended educational objectives.

Another approach taken to promote numerical problem solving in the undergraduate engineering curriculum has been to have the individual engineering departments teach computer programming rather than farming out this task to the computer science department. This arrangement offers the advantage of being able to offer discipline-specific instruction and example applications. Nonetheless, the impact of a more enlightened experience in the introductory computer programming course quickly fades if the students do not regularly apply their computer application skills in problem solving. As the saying goes, "use it or lose it."

***Daniel Coronell** received his B.S. in chemical engineering from the University of Illinois, Urbana, and his Ph.D. in chemical engineering from the Massachusetts Institute of Technology. After graduation, he worked in the chemical, semiconductor, and engineering software industries over a period of nine years before joining the Department of Chemical Engineering at Rose-Hulman Institute of Technology, where he is an associate professor.*

***M. Hossein Hariri** received his B.S. in gas engineering from Abadan Institute of Technology in Iran, his M.S. in gas engineering from IIT in Chicago and his Ph.D. in chemical engineering from the University of Manchester in England. After graduation, he worked in the petroleum industry and R&D in energy for 11 years and was an assistant professor at IIT before joining the Department of Chemical Engineering at Rose-Hulman Institute of Technology, where he is a professor.*

COMPUTER PROGRAMMING IN RHIT'S CHEMICAL ENGINEERING DEPARTMENT

The Department of Chemical Engineering at Rose-Hulman Institute of Technology has made similar modifications to its curriculum to address the issues outlined above. Since 1996, all undergraduate students at Rose-Hulman have been equipped with a laptop at the start of their freshman year. In 2002, the department inherited from its computer science colleagues the task of teaching introductory programming to students enrolled in chemical engineering. This two-credit-hour course, Programming and Computation for Chemical Engineers, is taken in the spring quarter of the students' freshman year.[2] It has included instruction on Microsoft Excel[3] (~1⁄3) as well as computer programming (~2⁄3). Excel's integrated Macro language, Visual Basic for Applications (VBA), has been used as a medium for teaching programming concepts to the students.

The decision to use VBA was motivated by several factors. First, the convenience and efficiency of using a single software package for the entire course reduced the instructional overhead. Since VBA is integrated with Excel, this also lessened the students' anxiety due to their familiarity with this application. Assuming that the use of VBA would continue into their chemical engineering courses, it circumvented the need for faculty in the department to embrace an unfamiliar software package. For example, Matlab was used in an earlier version of this course, and ~1⁄2 of the faculty had rarely or never used it before. Other important factors included the results of a comprehensive survey conducted by CACHE[4] that suggested Excel was the most widely used software by practicing chemical engineers. Lastly, it was recognized that the programming elements in VBA (*e.g.*, loops, decision constructs) were sufficiently similar to those in other common programming languages. As such, it would not be too difficult for students to subsequently learn how to program in a different language if required.

After several years of teaching the students how to program in VBA, it was recognized that their programming skills were still largely abandoned after the freshman year.

After several years of teaching the students how to program in VBA, it was recognized that their programming skills were still largely abandoned after the freshman year. It is believed that much of this is due to the relatively large investment in resurrecting their VBA course material and programs for application in subsequent coursework. This inevitably involves relearning some aspects of VBA, especially if more than a quarter or two has passed since they took their programming course. Most students are reluctant to do so, thus creating an obstacle to the objective of teaching them how to apply their programming skills to chemical engineering problem solving.

The principal pedagogical challenge that remains is two-fold. First, we simply need to increase the opportunities for students to apply their computer application skills to numerical problem solving while minimizing the instructional overhead. The use of a common numerical software package throughout the curriculum is a key part of a solution to this issue. Additionally, to be most effective, the software must be used by students in a manner that increases their understanding of the numerical algorithms while minimizing the actual programming effort.

THE CHEMICAL ENGINEER'S TOOLBOX

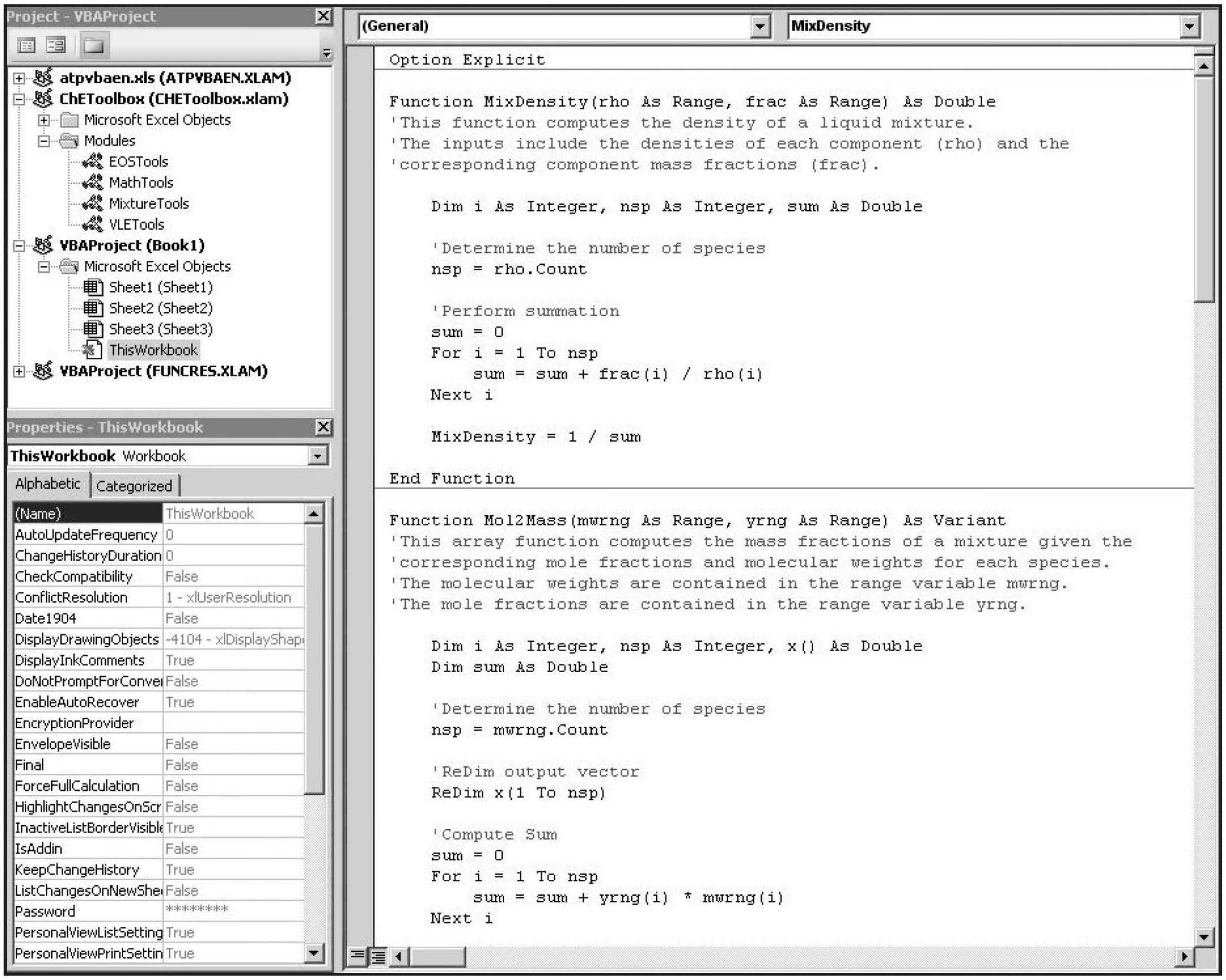
A novel approach to address this challenge is being evaluated with the students who have recently completed the Programming and Computation for Chemical Engineers course. The central idea is to compile the general-purpose programs that the students developed in this course into an Excel Add-In. This addresses the need to increase the numerical problem-solving opportunities by making the programs more accessible. The programs appear as functions built into their Excel application. This approach also addresses the need for the students to understand the underlying numerical algorithms. They will use their own programs in a transparent software environment. One may refer to this as a "glass box" approach to numerical problem solving, in contrast with the more familiar "black box" approach. The Excel Add-In can also be continuously populated with more complicated programs and numerical algorithms as the students move up the curriculum. This Add-In, hereafter referred to as the Chemical Engineer's Toolbox, is distributed to sophomore-level students at the start of the fall quarter for use in their introductory Material and Energy Balances course.

We envision that the students will use the Toolbox in a couple of different ways that serve to improve their learning of chemical engineering principles. For relatively routine computing tasks such as interpolation or numerical integration, the availability of Excel functions that streamline these calculations will enable them to spend more time learning the fundamentals of the problem and less time setting up a spreadsheet. In addition, as the students move up the curriculum, the relatively simple programs that they developed as freshmen can serve as templates or starting points for more sophisticated numerical problem solving. For example, a program that computes the specific volume using a simple pure component equation of state may be modified to account for nonideality or multiple components as an assignment in a thermodynamics course.

PROGRAMMING IN VBA

Before describing the Chemical Engineer's Toolbox, we offer a summary description of the VBA programming environment and language. The VBA programs are developed using the Visual Basic Editor, a graphical computer programming package that is seamlessly integrated with Excel. It consists of a multiple-pane window as shown in Figure 1. The two panes on the left are the Project Explorer and the Properties Window. The Project Explorer includes a listing of all open workbooks and installed Add-Ins, both of which are referred to as projects in VBA. Each project includes any associated objects (*e.g.*, worksheets), forms, and modules. A form is a customized graphical user interface, or GUI, that is created in the Visual Basic Editor and can be used to conveniently facilitate input and output for a VBA program. A module is simply a sheet onto which related VBA procedures are typed and stored. The code contained in a module appears in the pane to the right of the Project Explorer. The Properties Window can be seen below the Project Explorer pane. This window is used to specify the properties associated with objects contained in a VBA project.

The VBA programming language contains all of the traditional structured programming constructs (loops, decisions, etc.) and some aspects of object-oriented programming as well. A detailed description of the VBA programming language may be found in several recently published books.[5-7] VBA programs are referred to as procedures. There are several different types of VBA procedures, and methods to execute them. In Excel, the simplest approach is to use VBA function procedures since the protocol for calling one is identical to that of any of Excel's built-in functions (*e.g.*, SUM, SQRT, MINVERSE). For this reason, the collection of VBA programs in the Toolbox is comprised exclusively of function procedures.



```
atpvbaen.xls (ATPVBAEN.XLAM)
ChEToolbox (CHEToolbox.xlam)
    Microsoft Excel Objects
    Modules
        EOSTools
        MathTools
        MixtureTools
        VLETools
VBAProject (Book1)
    Microsoft Excel Objects
        Sheet1 (Sheet1)
        Sheet2 (Sheet2)
        Sheet3 (Sheet3)
        ThisWorkbook
VBAProject (FUNCRES.XLAM)
```

ThisWorkbook Workbook

(Name)	ThisWorkbook
AutoUpdateFrequency	0
ChangeHistoryDuration	0
CheckCompatibility	False
ConflictResolution	1 - xlUserResolution
Date1904	False
DisplayDrawingObjects	-4104 - xlDisplayShap
DisplayInkComments	True
DoNotPromptForConvei	False
EnableAutoRecover	True
EncryptionProvider	
EnvelopeVisible	False
Final	False
ForceFullCalculation	False
HighlightChangesOnScr	False
InactiveListBorderVisibl	True
IsAddin	False
KeepChangeHistory	True
ListChangesOnNewShe	False
Password	********
PersonalViewListSetting	True
PersonalViewPrintSettin	True

```
Option Explicit

Function MixDensity(rho As Range, frac As Range) As Double
'This function computes the density of a liquid mixture.
'The inputs include the densities of each component (rho) and the
'corresponding component mass fractions (frac).

    Dim i As Integer, nsp As Integer, sum As Double

    'Determine the number of species
    nsp = rho.Count

    'Perform summation
    sum = 0
    For i = 1 To nsp
        sum = sum + frac(i) / rho(i)
    Next i

    MixDensity = 1 / sum

End Function

Function Mol2Mass(mwrng As Range, yrng As Range) As Variant
'This array function computes the mass fractions of a mixture given the
'corresponding mole fractions and molecular weights for each species.
'The molecular weights are contained in the range variable mwrng.
'The mole fractions are contained in the range variable yrng.

    Dim i As Integer, nsp As Integer, x() As Double
    Dim sum As Double

    'Determine the number of species
    nsp = mwrng.Count

    'ReDim output vector
    ReDim x(1 To nsp)

    'Compute Sum
    sum = 0
    For i = 1 To nsp
        sum = sum + yrng(i) * mwrng(i)
    Next i
```

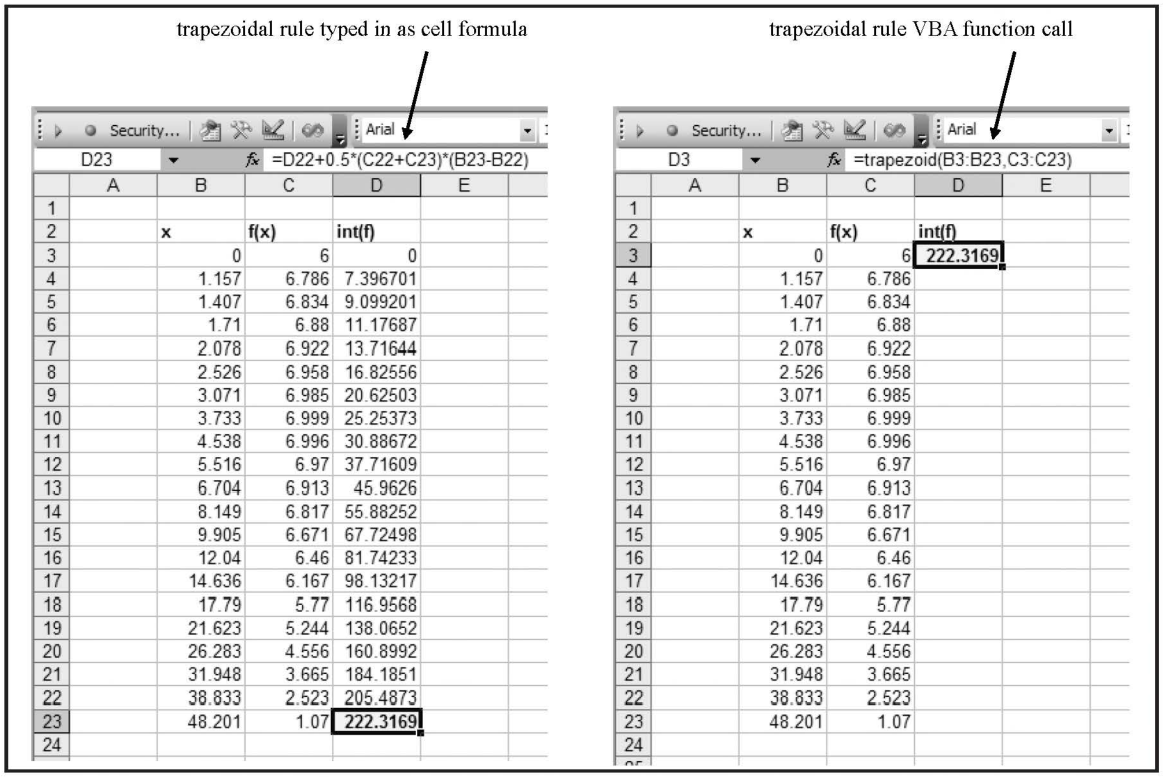
Figure 1. *Visual Basic Editor used to create VBA programs in Excel.*

A CLOSER LOOK AT THE CHEMICAL ENGINEER'S TOOLBOX

The initial version of the Toolbox is tailored to chemical engineering students beginning their sophomore year of instruction. It contains a collection of Excel functions that includes general math tools as well as tools designed to solve problems specific to their introductory Material and Energy Balances course. A list of the Toolbox functions can be found in Table 1.

It is seen that some of the functions do not require specification of a system of units (*e.g.*, Trapezoid, LinearSys), while others do (*e.g.*, Pvap, BubbleT). For the units-dependent functions, the Toolbox uses the SI system. This simplifies the coding of the VBA programs, but also requires the user to ensure that the function inputs are in SI units, and to convert the function output units if necessary. Note that the interested student could easily modify their VBA programs to accomodate additional systems of units. This type of creative enhancement of their Toolbox functions is encouraged.

We next consider a specific Toolbox function to demonstrate how it is applied to solve a problem and to illustrate some important characteristics of the underlying VBA code. A VBA function procedure that performs numerical integration using the trapezoidal rule [see Eq. (1) below] is developed by the


trapezoidal rule typed in as cell formula
trapezoidal rule VBA function call
Security...
Arial

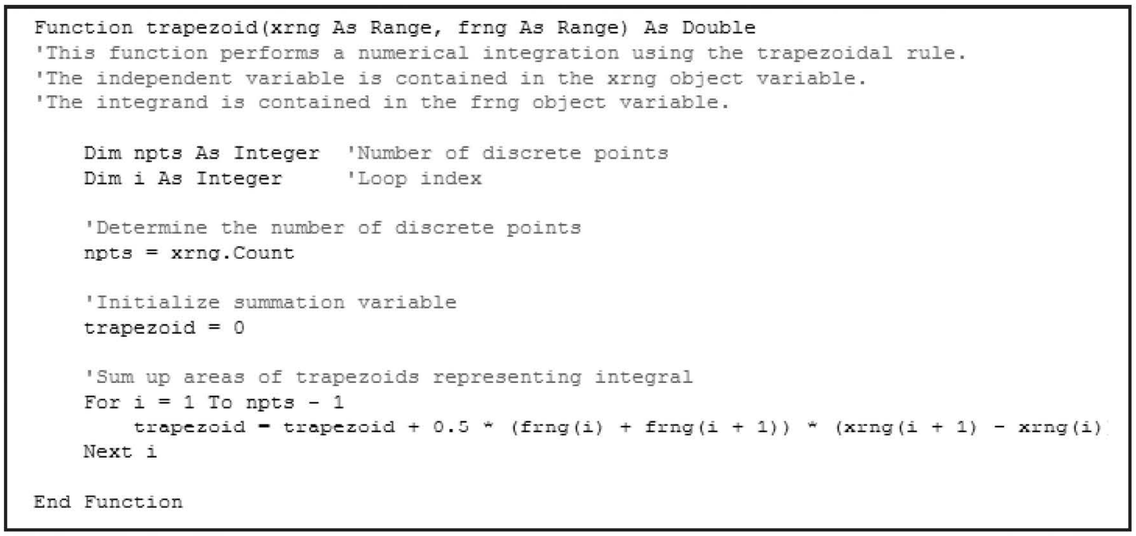
D23 =D22+0.5*(C22+C23)*(B23-B22)

	A	B	C	D	E
1					
2		x	f(x)	int(f)	
3		0	6	0	
4		1.157	6.786	7.396701	
5		1.407	6.834	9.099201	
6		1.71	6.88	11.17687	
7		2.078	6.922	13.71644	
8		2.526	6.958	16.82556	
9		3.071	6.985	20.62503	
10		3.733	6.999	25.25373	
11		4.538	6.996	30.88672	
12		5.516	6.97	37.71609	
13		6.704	6.913	45.9626	
14		8.149	6.817	55.88252	
15		9.905	6.671	67.72498	
16		12.04	6.46	81.74233	
17		14.636	6.167	98.13217	
18		17.79	5.77	116.9568	
19		21.623	5.244	138.0652	
20		26.283	4.556	160.8992	
21		31.948	3.665	184.1851	
22		38.833	2.523	205.4873	
23		48.201	1.07	222.3169	
24					

D3 =trapezoid(B3:B23,C3:C23)

	A	B	C	D	E
1					
2		x	f(x)	int(f)	
3		0	6	222.3169	
4		1.157	6.786		
5		1.407	6.834		
6		1.71	6.88		
7		2.078	6.922		
8		2.526	6.958		
9		3.071	6.985		
10		3.733	6.999		
11		4.538	6.996		
12		5.516	6.97		
13		6.704	6.913		
14		8.149	6.817		
15		9.905	6.671		
16		12.04	6.46		
17		14.636	6.167		
18		17.79	5.77		
19		21.623	5.244		
20		26.283	4.556		
21		31.948	3.665		
22		38.833	2.523		
23		48.201	1.07		
24					

Figure 2. *Comparison of a trapezoidal rule numerical integration using a spreadsheet with that using a Chemical Engineer's Toolbox VBA function.*



```
Function trapezoid(xrng As Range, frng As Range) As Double
'This function performs a numerical integration using the trapezoidal rule.
'The independent variable is contained in the xrng object variable.
'The integrand is contained in the frng object variable.

    Dim npts As Integer  'Number of discrete points
    Dim i As Integer     'Loop index

    'Determine the number of discrete points
    npts = xrng.Count

    'Initialize summation variable
    trapezoid = 0

    'Sum up areas of trapezoids representing integral
    For i = 1 To npts - 1
        trapezoid = trapezoid + 0.5 * (frng(i) + frng(i + 1)) * (xrng(i + 1) - xrng(i)
    Next i

End Function
```

Figure 3. *VBA code for the trapezoidal rule function.*

students in their Programming and Computation for Chemical Engineers course.

$$\int_{x_o}^{x_f} f(x)dx \approx \sum_{i=1}^{n-1} \frac{\left(f(x_i)+f(x_{i+1})\right)}{2} \times (x_{i+1}-x_i) \qquad (1)$$

This simple computational task can be accomplished using only a spreadsheet. In fact, the students are required to do so during the first part of the course. This serves to reinforce the value of having a customized function available for this purpose—it streamlines the calculation and eliminates the possibility of incorrectly implementing the trapezoidal rule formula once it has been programmed correctly. This is shown in Figure 2, where an example numerical integration is performed using a spreadsheet-only implementation and again using the Toolbox function.

The corresponding VBA code for the trapezoidal rule function is seen in Figure 3. The students are taught to create computer programs that are reusable, *i.e.*, not specific to a given problem. For example, the trapezoidal rule function can be used to integrate using any number of discrete independent variable points that are equally or unequally spaced. In addition, the students are encouraged to write programs that can be easily modified if necessary. An example would be to expand the trapezoidal rule function so that it can perform numerical integration using Simpson's 1/3 rule, as well. If the code includes sensibly named variables and is sufficiently commented, this task is more easily accomplished.

In the following section, we provide a couple of example applications of the Chemical Engineer's Toolbox. These are taken from Felder and Rousseau's textbook *Elementary Principles of Chemical Processes*[8] to demonstrate how the Toolbox may be used for problem solving in an introductory Material and Energy Balances course.

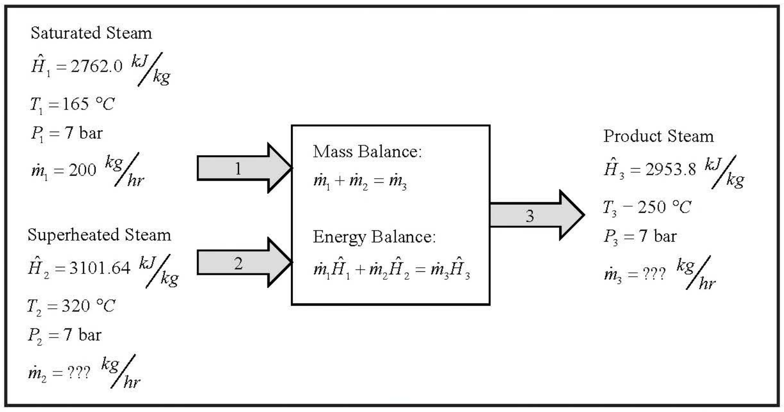


***Figure 4.** Schematic illustration of the steam-mixing problem based on problem 7.35b in Felder and Rousseau.*[8]

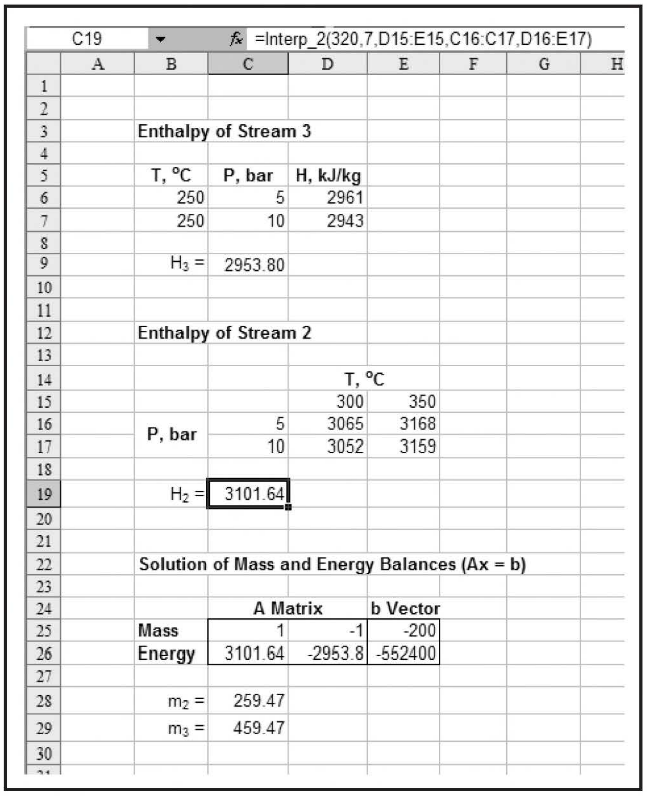
EXAMPLE APPLICATIONS OF THE CHEMICAL ENGINEER'S TOOBOX

The first example problem involves the adiabatic mixing of two streams containing saturated and superheated steam as shown in Figure 4. This corresponds to problem 7.35b in Felder and Rousseau,[8]

TABLE 1
Functions in the Sophomore Version of the Chemical Engineer's Toolbox

Function Name	Description
Trapezoid	Performs numerical integration using the trapezoidal rule.
LinearSys	Solves a system of linear equations.
Interp_1	Performs linear or cubic interpolation in one dimension.
Interp_2	Performs linear or cubic interpolation in two dimensions.
NewtonRaphson	Solves a single nonlinear algebraic equation using the Newton-Raphson method.
ModEuler	Solves an initial value ODE using the modified Euler method.
Mol2Mass	Converts mole fractions to mass fractions.
Mass2Mol	Converts mass fractions to mole fractions.
MW_mix	Computes the average molecular weight of a mixture.
Density_mix	Computes the density of a liquid mixture.
V_VDW	Computes the specific volume of a gas using the van der Waals EOS.
V_Virial	Computes the specific volume of a gas using the virial EOS.
V_RK	Computes the specific volume of a gas using the Redlich-Kwong EOS.
Pvap	Computes the vapor pressure of a pure component using the Antoine equation.
BubbleP	Computes the bubble point pressure and corresponding vapor composition using Raoult's law and the Antoine equation.
BubbleT	Computes the bubble point temperature and corresponding vapor composition using Raoult's law and the Antoine equation.
DewP	Computes the dew point pressure and corresponding liquid composition using Raoult's law and the Antoine equation.
DewT	Computes the dew point temperature and corresponding vapor composition using Raoult's law and the Antoine equation.

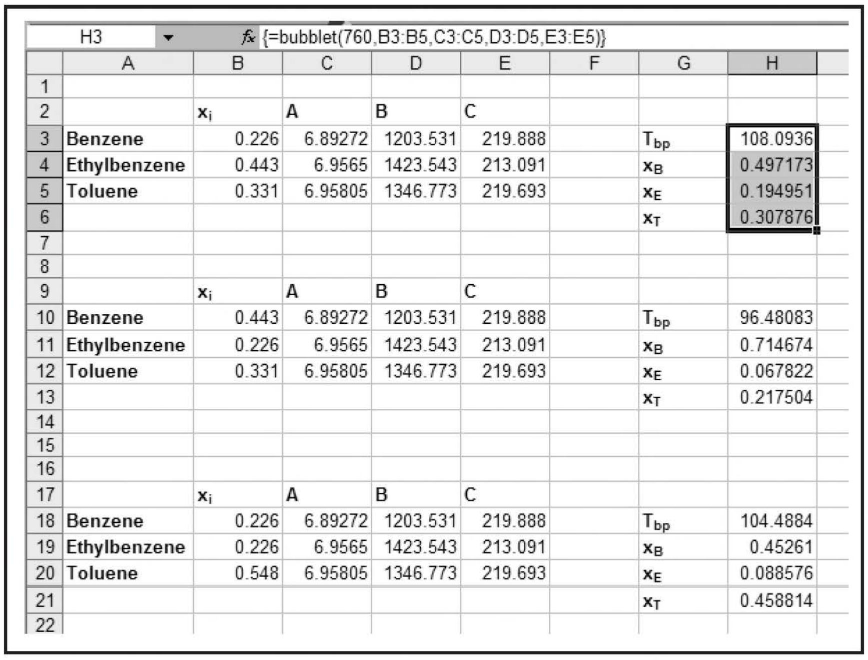
with some of the pressures and temperatures modified to underscore the use of the Toolbox functions. The problem requires the students to simultaneously solve material and energy balances around the system by making use of saturated and superheated steam tables.



C19 =Interp_2(320,7,D15:E15,C16:C17,D16:E17)

	A	B	C	D	E	F	G	H
1								
2								
3		Enthalpy of Stream 3						
4								
5		T, °C	P, bar	H, kJ/kg				
6		250	5	2961				
7		250	10	2943				
8								
9		H_3 =	2953.80					
10								
11								
12		Enthalpy of Stream 2						
13								
14				T, °C				
15				300	350			
16		P, bar	5	3065	3168			
17			10	3052	3159			
18								
19		H_2 =	3101.64					
20								
21								
22		Solution of Mass and Energy Balances (Ax = b)						
23								
24			A Matrix		b Vector			
25		Mass	1	-1	-200			
26		Energy	3101.64	-2953.8	-552400			
27								
28		m_2 =	259.47					
29		m_3 =	459.47					
30								

Figure 5. *Toolbox-assisted solution to problem 7.35b in Felder and Rousseau.*[8]



H3 {=bubblet(760,B3:B5,C3:C5,D3:D5,E3:E5)}

	A	B	C	D	E	F	G	H
1								
2		x_i	A	B	C			
3	Benzene	0.226	6.89272	1203.531	219.888		T_{bp}	108.0936
4	Ethylbenzene	0.443	6.9565	1423.543	213.091		x_B	0.497173
5	Toluene	0.331	6.95805	1346.773	219.693		x_E	0.194951
6							x_T	0.307876
7								
8								
9		x_i	A	B	C			
10	Benzene	0.443	6.89272	1203.531	219.888		T_{bp}	96.48083
11	Ethylbenzene	0.226	6.9565	1423.543	213.091		x_B	0.714674
12	Toluene	0.331	6.95805	1346.773	219.693		x_E	0.067822
13							x_T	0.217504
14								
15								
16								
17		x_i	A	B	C			
18	Benzene	0.226	6.89272	1203.531	219.888		T_{bp}	104.4884
19	Ethylbenzene	0.226	6.9565	1423.543	213.091		x_B	0.45261
20	Toluene	0.548	6.95805	1346.773	219.693		x_E	0.088576
21							x_T	0.458814
22								

Figure 6. *Toolbox-assisted solution to problem 6.58b in Felder and Rousseau.*

This problem is particularly well-suited for application of the Chemical Engineer's Toolbox since it requires repetition of many routine calculations. For example, the steam enthalpies for streams 2 and 3 ($\hat{H}_2$ and $\hat{H}_3$ in Figure 4) must be determined by two- and one-dimensional interpolation, respectively, using the values available in the steam tables. The interpolation tasks are conveniently performed using the Interp_2 and Interp_1 functions. In addition, the solution of the material and energy balance equations are obtained using the LinearSys function. The Toolbox-assisted solution to this problem is illustrated in Figure 5. Clearly, the solution could have been obtained without the Toolbox, but having the functions available minimizes the time spent performing rote calculations and provides more time for the students to learn the fundamental steps that lead to the solution. Additionally, the underlying code that comprises the functions is fully accessible to the students.

The second example application of the Toolbox makes use of the bubble point temperature function, BubbleT. Problem 6.58b in Felder and Rousseau[8] involves the calculation of the bubble point temperature of an ideal mixture of benzene, ethylbenzene, and toluene. The students are asked to compute the bubble point temperature at atmospheric pressure for three different liquid mixture compositions by using Raoult's Law and the Antoine equation.

$$\sum_i y_i P = \sum_i x_i P_i^{sat} \tag{2}$$

$$\log_{10} P_i^{sat} = A_i - \frac{B_i}{T + C_i} \tag{3}$$

This numerical problem consists of solving a single nonlinear algebraic equation where the unknown variable is the temperature, T. Excel's Goal Seek utility is recommended in the problem statement.

This problem represents an example where the Toolbox may be used in a manner that expands the students' learning opportunities beyond the original problem statement. For example, the students can be asked to create a Txy diagram by using the BubbleT and DewT functions since they return the temperature and phase composition. In this example, the BubbleT function was applied to solve for the bubble point temperatures of three ternary mixtures specified in the problem statement. BubbleT is an array function (*i.e.*, returns an array of numbers) that computes not only the bubble point temperature but also the corresponding vapor composition. The required inputs include the total pressure, liquid phase mole fractions, and the Antoine equation coefficients for each species. The Toolbox-assisted solution to problem 6.58b is shown in Figure 6. The additional function output consisting of the vapor phase composition enriches the students' learning associated with the problem. Unlike the "black box" Goal Seek approach, the students can also see the underlying numerical algorithm (Regula-Falsi method as described in Appendix A of Felder and Rousseau) implemented in the BubbleT function, providing them with a more fundamental understanding of the solution technique. In addition, the BubbleT function can serve as a basis for incorporating non-ideal effects when the students learn more about vapor-liquid equilibrium in a subsequent thermodynamics course.

SUMMARY AND CONCLUDING REMARKS

The Chemical Engineer's Toolbox was developed as a proactive approach to improve students' computer application skills in the undergraduate chemical engineering curriculum. The primary benefits of the Toolbox that enhance student learning are its convenient access as an Excel Add-In and the transparent "glass box" environment in which the programs are created and stored. The students are provided with the Toolbox at the start of their sophomore year for use in their introductory material and energy balances course. It is anticipated that additional VBA program modules will be defined and created by faculty and students alike in courses to follow in fluid mechanics, heat transfer, thermodynamics, mass transfer, and reactor engineering.

REFERENCES

1. The Mathworks, Inc., Natick, MA. <http://www.mathworks.com>
2. Coronell, D.G., "Computer Science or Spreadsheet Engineering?" *Chem. Eng. Educ.*, **39**(2), 142 (2005)
3. Microsoft, Inc. Redmond, WA. <http://www.microsoft.com>
4. Edgar, T., "Computing Through the Curriculum: An Integrated Approach for Chemical Engineering," *CACHE Fall 2003 Newsletter*, <http://www.che.utexas.edu/cache/newsletters/fall2003_cover.html>
5. Billo, E.J., *Excel for Scientists and Engineers: Numerical Methods*, Wiley, New York (2007)
6. Walkenbach, J., *Excel 2003 Power Programming with VBA*, Wiley, New York (2004)
7. Kimmel, P.T., S. Bullen, J. Green, R. Bovey, and R. Rosenberg, *Excel 2003 VBA Porgrammer's Reference*, Wiley, New York (2004)
8. Felder, R.M., and R.W. Rousseau, *Elementary Principles of Chemical Processes*, 3rd Ed., Wiley, New York (2005) ❒