

# On the Necessary and Sufficient to Efficient Use of Software in the Teaching of Chemical Engineering

Ruan Varjão Dias<sup>1</sup>, Luís Gonzaga Sales Vasconcelos<sup>1</sup> & Romildo Pereira Brito<sup>1</sup>

<sup>1</sup> Department of Chemical Engineering, Federal University of Campina Grande, Bodocongó, Brazil

Correspondence: Romildo Pereira Brito, Department of Chemical Engineering, Federal University of Campina Grande, Av Aprígio Veloso 882, Bodocongó, Campina Grande - PB 58109-970, Brazil. E-mail: romildo.brito@deq.ufcg.edu.br

Received: September 4, 2012

Accepted: February 14, 2013

Online Published: February 23, 2013

doi:10.5539/mas.v7n3p43

URL: <http://dx.doi.org/10.5539/mas.v7n3p43>

## Abstract

The development of computers brought an increase in the number of software programs with a great potential to assist in the task of teaching engineering, among them are: Mathcad<sup>®</sup>, Mathematica<sup>®</sup>, Matlab<sup>®</sup>, Maple<sup>™</sup> and Polymath. Several articles have been published with the aim of demonstrating the advantages of using these softwares, as well comparing their performance. The most recent books have clearly indicated the importance of these tools in engineering. Even so, however, the computer has not become a strong helper of the student in solving more complex engineering problems. The point of view presented in this article is a result of over twenty years of teaching in Chemical Engineering. In our opinion, the fact that students know to use a specific software program (syntax) is a necessary condition for solving engineering problems, but not sufficient if the student do not think in a systematic way. The methodology recommended in this article is applied for the resolution of the problem in modules and can be used in both simple and complex problems.

**Keywords:** teaching, algorithm, programming, software, chemical engineering

## 1. Introduction

More important than simplifying a problem to obtain a solution is to make assumptions, generate results and choose the most appropriate solution for the problem under consideration. Thus, the computer has transformed the practice of engineering at all levels, allowing the resolution of problems without need of simplifications and even being able to reveal different approaches of the problems, answering the question: what if...? Consequently, it is natural that the teaching of algorithms and programming language occupies a prominent place in the training of future engineers.

The inclusion of subjects on programming and numerical methods has been a reality for students of engineering since the early 1970s. Traditionally, FORTRAN was the language chosen, however the time spent to obtain success in an implementation of a numerical method and then use the developed code in the resolution of the problem in question, was the main difficulty observed.

This way, many students stopped using the computer to solve problems of engineering and, in fact, for them the computer has become a tool for editing text, web browsing and preparation of presentations. Those who persisted in using the computer with the purpose of solving engineering problems became more qualified professionals, because they had the ability to solve more complex problems using the computer. Then, it is essential to register the importance of FORTRAN on good training of actual engineers and scientists.

The use of commercial softwares such as Mathcad<sup>®</sup>, Mathematica<sup>®</sup>, Matlab<sup>®</sup>, Maple<sup>™</sup> and Polymath is an alternative to eliminate the stage of numerical code's implementation and to increase the number of professionals who use the computer to solve problems of engineering. Many universities have added courses to their curricula, specifically to teach the systematic use of computer in solving problems.

A software that requires a minimum effort in the transcript of the mathematical model, for example, Polymath, is highly recommended by some authors, in addition to the use of these together with more robust software programs (Shacham et al., 2001; Shacham et al., 2003; Shacham & Cutlip, 2004; Shacham et al., 2008).

Nasri and Binous (2008) and Binous (2008) presented articles demonstrating the use of Mathematica<sup>®</sup> and Matlab<sup>®</sup> to solve problems involving Thermodynamics and Separation Processes, and the authors concluded that the Mathematica<sup>®</sup> is more didactic than Matlab<sup>®</sup>.

Vasconcelos et al. (2008) used Mathcad<sup>®</sup> to demonstrate the application of the McCabe-Thiele Method for problems with non-conventional specification.

Shacham et al. (2009) recommended the use of different software for each phase of the Chemical Engineering curriculum.

Using Simulink<sup>®</sup> and Mathematica<sup>®</sup>, Binous et al. (2011) presented an article showing how one can apply both mass and energy balance equations in order to understand the separation, the dynamic behavior and the control of a distillation column.

With respect to books that consider the use of computers in solving Chemical Engineering problems, some are devoted to specific topics (Skogestad, 2008), others have a variety of subjects and their focus is the description of how to use the software (Finlayson, 2006; Cutliffe, 2008).

In fact, the solutions proposed to solve the problematic of the little or non-use of computers in solving engineering problems can be summarized as: i) teaching programming language and algorithm; ii) use of a software; iii) publication of books; iv) publication of articles.

The point of view presented in this article is a result of over twenty years of teaching in Chemical Engineering, using a wide range of programming languages and software. In our opinion, the fact that students know to use a software program (syntax) is a necessary condition for solving engineering problems, but not sufficient if the students do not think in a systematic way; regardless of which software program is being used.

## 2. Problem Statement

With few exceptions, what the student learns in the disciplines of programming language and numerical methods is not used in the subsequent disciplines of the course of Chemical Engineering. In most cases, the students use the computer in the final courses of the curriculum through commercial software that operates as a black box, such as Aspen and Hysys.

It is also interesting to note that normally the student will find the teacher using the computer. However, according to Felder and Brent (2005), in the article "Death by PowerPoint", the teacher spends most of his time using the computer to prepare presentations for using it in the classroom.

The non-use of the computer as a tool in solving engineering problems has been evidenced in the past fifteen years (Kanto & Edgar, 1996; Jones, 1998) and, considering the number of articles concerning the subject, even with the commercial software mentioned previously, the computer has not become a strong helper of the student in solving more complex problems of engineering.

Fundamental disciplines (such as Thermodynamics, Transport Phenomena, Unit Operations and Chemical Reactor Design) offer many opportunities to use a computer, especially for realistic problems. Excellent examples of how the computer can be useful in solving problems, of these and other subjects, are found in the articles cited above; however, no comment is made about the necessity of students thinking in a systematic manner before using a software program.

Figures 1, 2 and 3, present the prompt command of Matlab<sup>®</sup>, Mathcad<sup>®</sup> and Polymath, respectively, which were used to find one of the roots of a polynomial function. At this point, it is important to emphasize that Polymath uses control programming based on a menu, while both Matlab<sup>®</sup> and Mathcad<sup>®</sup> use control programming based on command.

It is a typical situation where the student can demonstrate the use of the software program, but it does not necessarily mean that the student is able to solve problems of engineering, because, in general, these problems are not established explicitly as a polynomial equation.

In most cases the student must define a sequence of steps (programming) to reach a final equation. And this is the point of the question, because, for the first step in using the software, the students have to consider the equations of the problem and they must do it in a manner that the computer (software) can be used to obtain the solution (or solutions). In addition, the students should be aware of how the process of finding the solution works. That is, the students must be both sufficiently knowledgeable and sophisticated in terms of algorithm and programming.

Observing Figures 1, 2 and 3, a more pragmatic teacher may argue that the important thing is the end result: one of the roots was found. Then, consider a Mathcad<sup>®</sup> code used to demonstrate the application of the method of McCabe-Thiele to problems with unconventional specifications and that involves one stage of optimization (Vasconcelos et al., 2008), as is shown in Figure 4. Mathcad<sup>®</sup> is like an electronic book, where the mathematical functions are used almost the same as appear in the books. In our opinion, despite that it seems a very friendly software, only a student with sufficient knowledge in algorithm and programming, and obviously knowing the syntax of the software and about the problem of Chemical Engineering will have the ability to create (or understand) an algorithm like the one presented in Figure 4.

### 3. Algorithm, Programming and Software

Logical and systematic thinking is a unique pre-requisite to the use of computer software to solve engineering problems, and the only way to get this pre-requisite is through the use of algorithms and programming. To explore this question, the first step is to define what programming means.

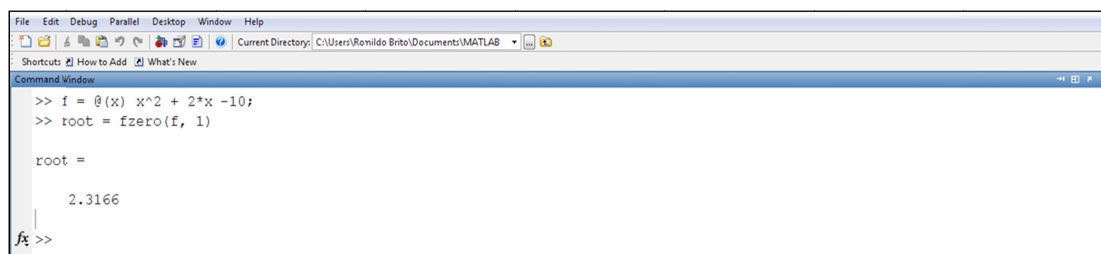


Figure 1. Matlab<sup>®</sup> prompt command used to find one of the roots of a polynomial function

A program is an algorithm implemented; a sequence of activities (instructions) to be executed to reach an objective. It is possible to use a programming language, software or spreadsheet (Bjedov & Andersen, 1996).

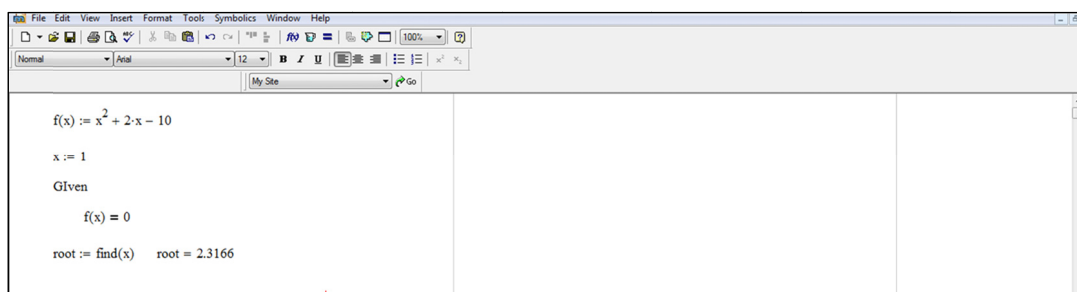


Figure 2. Mathcad<sup>®</sup> prompt command used to find one of the roots of a polynomial function

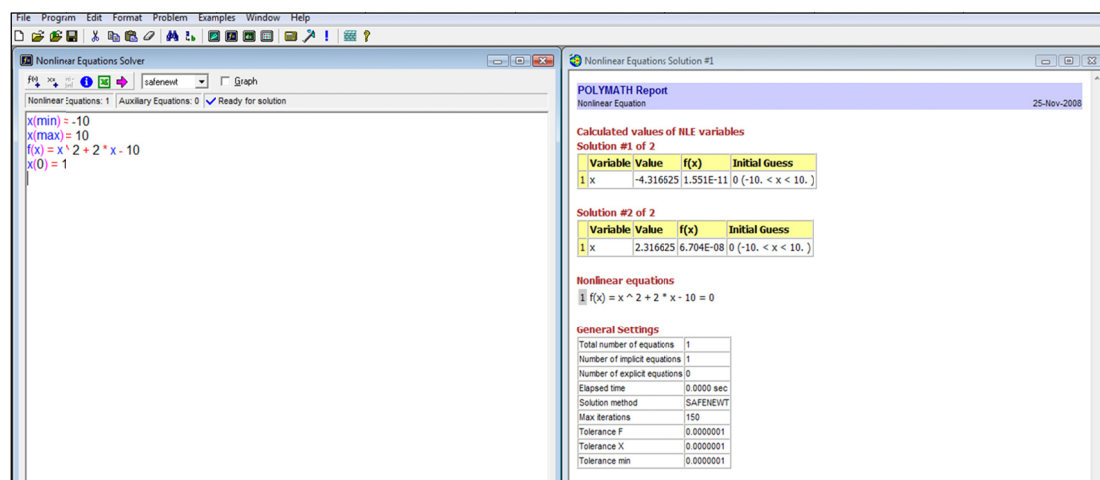


Figure 3. Polymath prompt command used to find one of the roots of a polynomial function

When teaching algorithms and programming, the first recommendation is to avoid the implementation of more complex codes, so as not to discourage the students, because the goal is not to challenge them to draw up codes to compete with commercial software, but encourage them to use and discover that programming can be a pleasurable activity. It is important to emphasize that with simple commands, for example, with three basic commands: for, if and while (in the case of Matlab®) most problems can be solved (Chapra & Canale, 2008).

Figure 5 shows the code used in Matlab® to calculate the sum of  $n$  terms of the series:  $S = \frac{1}{2} + \frac{3}{5} + \frac{5}{8} \dots$ . The purpose of this example is to make the student think logically: i) the first term is  $1/2$ ; ii) two unities are being added to the numerator for each term, while 3 unities are being added to the denominator for each term; iii) one structure of repetition must be used; iv) the number of terms must be defined. It also represents an example of where the software probably will not provide the result using a single internal function (built-in).

Throughout the course, the implementation of simple numerical methods can be used, for example, Newton for nonlinear algebraic equations and Euler for ordinary differential equations. At this moment, the use of internal functions (solvers) of the software chosen should also be introduced and the results of the two forms compared.

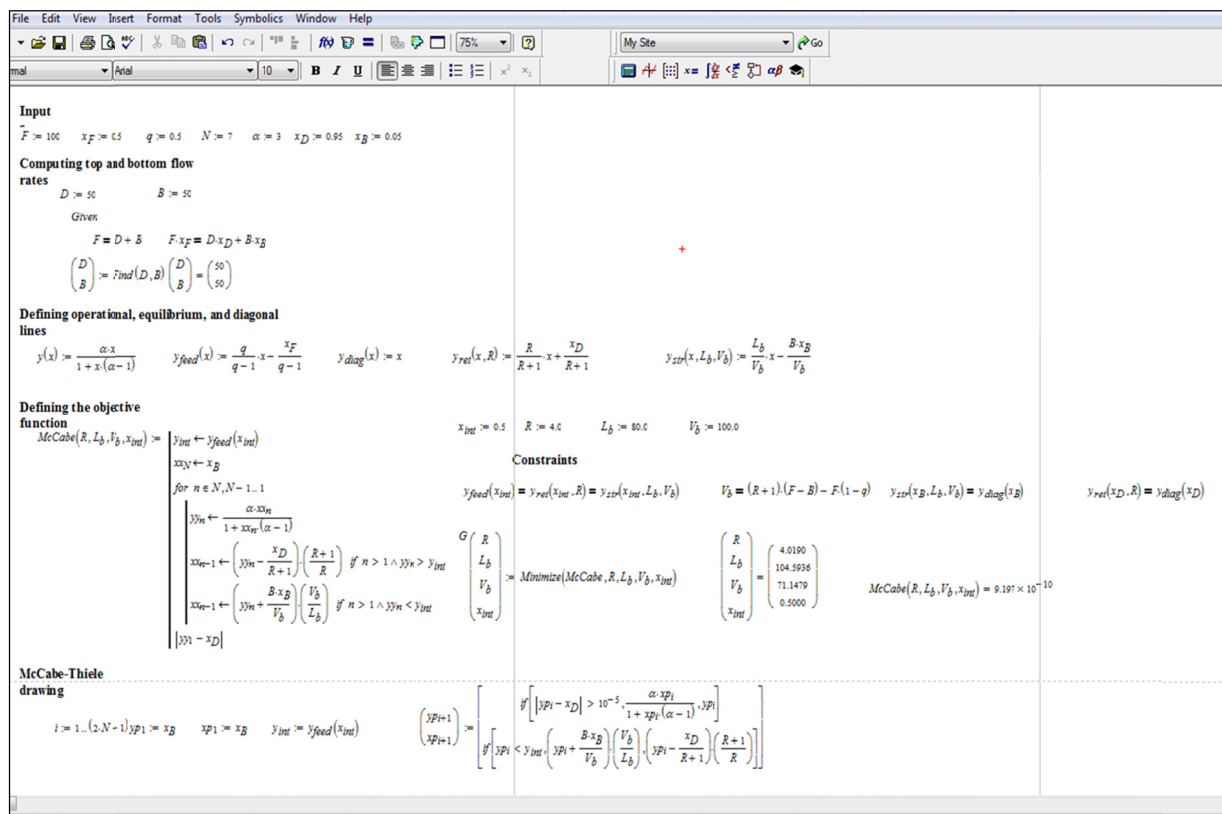
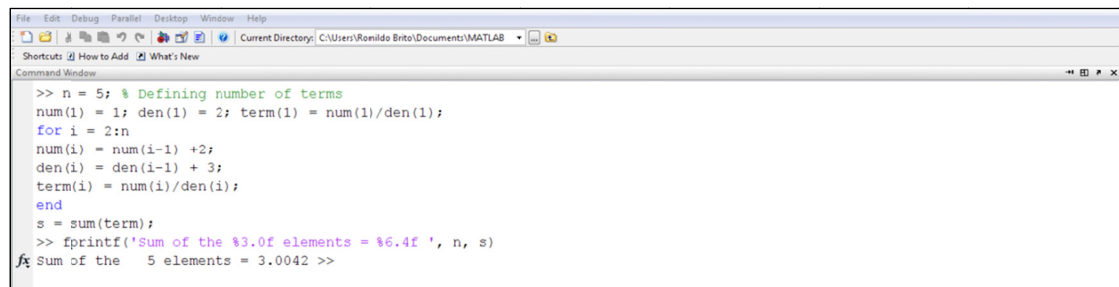


Figure 4. Mathcad® prompt command used to demonstrate the application of the method of McCabe-Thiele to problems with unconventional specifications

As can be observed, we do not separate programming language, algorithms and numerical methods into different subjects, since, in terms of engineering, it is difficult to discern where one ends and where the other begins. A discipline called Computer for Chemical Engineering, for example, splitted into two parts and taught by a chemical engineer, can be used to cover the entire content.



```

>> n = 5; % Defining number of terms
num(1) = 1; den(1) = 2; term(1) = num(1)/den(1);
for i = 2:n
    num(i) = num(i-1) + 2;
    den(i) = den(i-1) + 3;
    term(i) = num(i)/den(i);
end
s = sum(term);
>> fprintf('Sum of the %3.0f elements = %6.4f ', n, s)
Sum of the 5 elements = 3.0042 >>

```

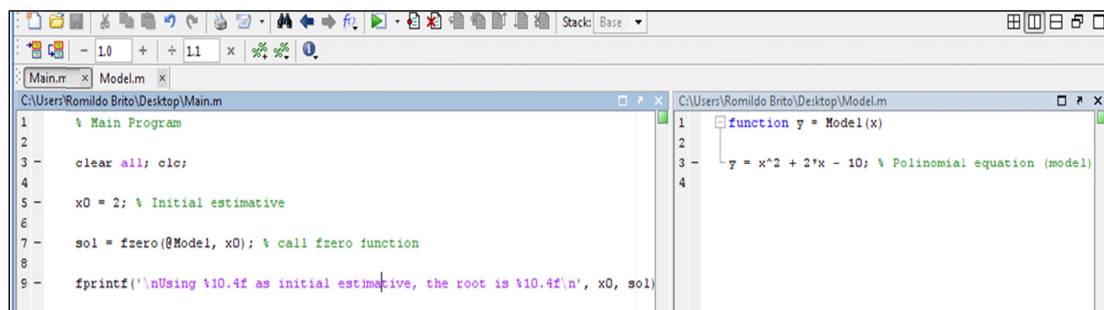
Figure 5. Matlab<sup>®</sup> code used to calculate the sum of n terms of series

The methodology used in the following examples is applied for structure the resolution of the problem in modules (main program, the problem model, parameters and the solver). This methodology can be used in both simple and complex problems and these are the same steps used in the early days with the first computers for engineering, with a difference: one of these modules (solver) is ready and available in the software.

Matlab<sup>®</sup> was chosen to be a hybrid of software and programming language, which makes it possible to have the model of the problem completely separate from the solver; a key feature required for the methodology presented in this article. Considering the cost of acquisition of Matlab<sup>®</sup>, one option is to use the Scilab or Octave that are free distributed.

Figure 6 shows the Matlab<sup>®</sup> files used to determine the roots of a polynomial equation  $y = x^2 + 2x - 10$ , using m-files. It may seem an unnecessary effort by the student, but the goal is to systematize the way to solve problems, whether simple or complex.

It is essential to clarify for the student that the file on the left (Main.m), usually called the main program is the link between the student and the computer. From an initial estimative this file communicates with the built-in function (file fzero.m) of Matlab<sup>®</sup> to solve the equation (model) in the file on the right side (Model.m).



```

Main.m
1 % Main Program
2
3 clear all; clc;
4
5 x0 = 2; % Initial estimative
6
7 sol = fzero(@Model, x0); % call fzero function
8
9 fprintf('\nUsing %10.4f as initial estimative, the root is %10.4f\n', x0, sol)

Model.m
1 function y = Model(x)
2
3 y = x^2 + 2*x - 10; % Polynomial equation (model)
4

```

Figure 6. M-files of Matlab<sup>®</sup> used to find the roots of a polynomial equation of degree 2

Figure 7 shows a block diagram of communication between the files. The student must understand that:

- 1) The file Main.m call the file fzero.m, which calls the file Model.m;
- 2) Each time the file Model.m is called, a value for y is calculated and returned to the file fzero.m;
- 3) This mechanism (1-2-3-2-3-2-3 ... 2-3-4) occurs until a tolerance for the value of y is achieved;
- 4) Since we are looking for the roots of the polynomial equation, the value of y should be very close to zero.

It is important to emphasize that the built-in functions of the Matlab<sup>®</sup> appears only in the block diagrams.

Probably other software programs mentioned in this article also solve the problem using a statement similar to the showed in Figure 7 and also have their own programming language (for, while, if, break, etc., in case of Mathcad<sup>®</sup>); however, they do not allow the student to solve the problem in modules, so that, for a more complex problem, the final file is extensive and more difficult to understand (see, for example, Figure 4).

In our understanding, algorithms, programming and software are closely linked. If a program is an algorithm implemented, it is impossible to program without having learned algorithm. Similarly, you cannot use software

efficiently, without knowing programming, regardless of how friendly the software is. Probably, it is more efficient to start with a robust software and then, if necessary, migrate to a more friendly one.

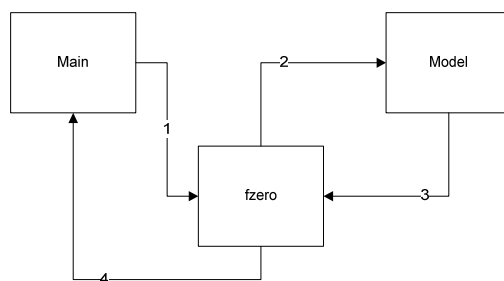


Figure 7. Block diagram of communication between the files of Figure 6

#### 4. Solving Problems in Chemical Engineering

The examples presented below are typical of Chemical Engineering problems, and despite the need to use the computer, they are relatively simple. It is also important to emphasize that the implementations were made to the teaching aspect.

##### 4.1 Problems Involving Algebraic Equations

The equation of state (EOS) for ideal gases represents satisfactorily the relation between pressure, volume and temperature (PVT) only for cases where the pressure is low (near atmospheric pressure). For higher pressure more complex EOS should be used and, in such cases, the calculation of molar volume and compressibility factor demand the use of numerical techniques.

The equation of state of Redlich-Kwong relates PVT data by Equation (1):

$$P = \frac{RT}{(V-b)} - \frac{a}{V(V+b)\sqrt{T}} \quad (1)$$

The constants **a** and **b** are expressed by Equations (2) and (3).

$$a = 0.42747 \left( \frac{R^2 T_c^{2.5}}{P_c} \right) \quad (2)$$

$$b = 0.08664 \left( \frac{RT_c}{P_c} \right) \quad (3)$$

For a given value of temperature and pressure, an EOS can be used to determine the molar volume of the substance and the compressibility factor. Data of critical temperature ( $T_c$ ) and critical pressure ( $P_c$ ), in addition to the Universal Gas Constant ( $R$ ), are necessary to the calculations.

Since values are given for  $P$  and  $T$  and the constants **a** and **b** are specific to each substance, the problem is to find a value of  $V$  which satisfies the Equation (1).

When we compare this problem with the example of polynomial equation, the students note that the equations (models) are different. In this case, the model is not explicitly defined, so that, it is essential, to show that Equation (1) must be modified, before proceeding with the resolution of the problem.

From Equation (1) it is clear that the variable  $V$  cannot be isolated. Nevertheless, moving  $P$  to the right of equality, we obtain the equation (4), which is a function only of  $V$ .

$$\frac{RT}{(V-b)} - \frac{a}{V(V+b)\sqrt{T}} - P = 0 \quad (4a)$$

$$f(V) = \frac{RT}{(V-b)} - \frac{a}{V(V+b)\sqrt{T}} - P \quad (4b)$$

Thus, we have a model in a standard form, where the number of variables is equal to the number of equations and the software program can be used to determine the value of  $V$  for which the value of  $f$  is nearly zero; as in

the case of Figure 6 we find a value of  $x$  for which  $y$  was approximately zero. However, we still have a problem: the constants **a** and **b**, which must be calculated before completing the model.

Models of Chemical Engineering usually have parameters associated with the chemical species present in the process. It is recommended that these parameters should be available in a specific location, an m-file just for this purpose, so when necessary only the file containing the parameters will be modified when there are changes in the chemical species.

Figure 8 shows the m-files used to solve this problem, in which an m-file is only used to store specific data. When the word “Parameters” appears, the data in this file will be available from that point. For example, in the file Main.m with the presence of the word “Parameters”, it is possible to calculate the initial estimate, which needs the value of  $R$ , while in the file Model.m with the word “Parameters”, it is possible to calculate  $a$  and  $b$ , which requires  $R$ ,  $T_c$  and  $P_c$ , before calculating the value of the function  $f$ .

Figure 9 shows the block diagram of communication between the files that is summarized: 1-2-3-4-5-6-7-4-5-6-7 ... 4-5-6-7-8. After reaching the tolerance specified, the file fzero.m returns the solution to the file Main.m.

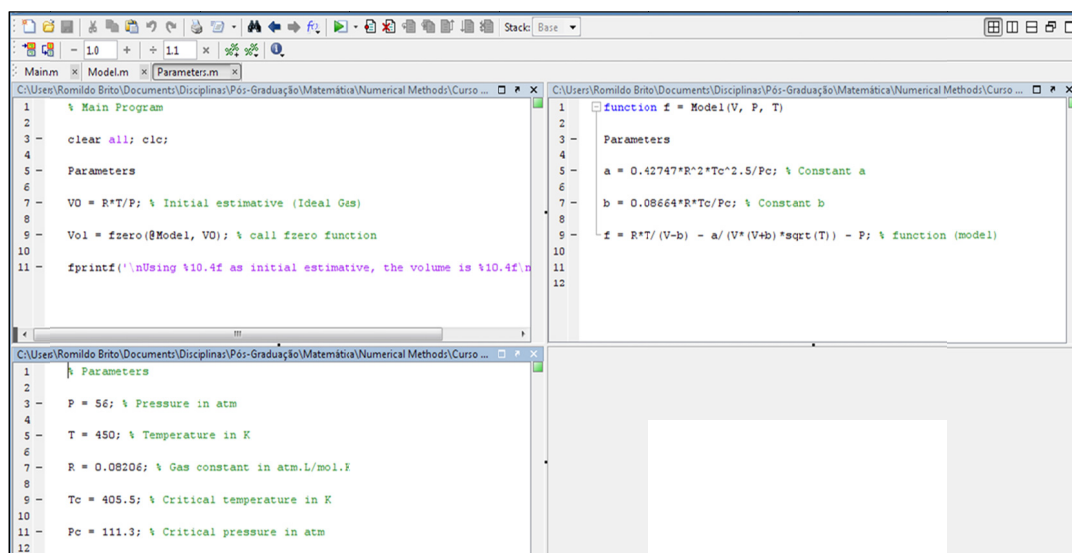


Figure 8. M-files used to determine the molar volume from the equation of Redlich-Kwong

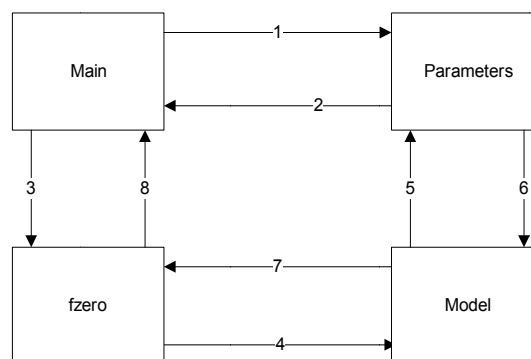


Figure 9. Block diagram of communication between the files of Figure 8

A more elaborate example is shown in Figure 10, which is in the simplified block diagram of the production process of styrene, in which two reactions are considered. The following data are provided:

- ✓ Selectivity for toluene as a function of temperature;
- ✓ Operating temperature of the reactor;
- ✓ Ratio of the input flow of ethyl benzene and water in the reactor;

- ✓ Pressure of operation of the reactor;
- ✓ Flow of production of styrene;
- ✓ Equilibrium constant for the reaction of formation of styrene.

The objective of this problem is to determine the flow rate of each component in each stream, and the first step is to give numbers to the chemical species: ethyl benzene (1), Water (2), styrene (3), Toluene (4), Methane (5) and Hydrogen (6). The next step is to perform an analysis of degrees of freedom (Table 1). The model of this problem has eighteen variables (species flow rates and reactions rates) and eighteen equations (Table 2).

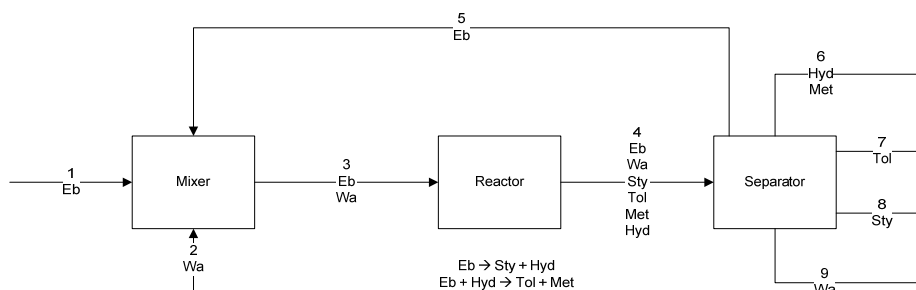


Figure 10. Block diagram of the production process of styrene

Table 1. Degrees of freedom of the problem presented in Figure 10

	Mixer	Reactor	Separator	Global	Process
Variable	5	8+2	12	7+2	16+2
Mass balance	2	6	6	6	14
Flow			1	1	1
Selectivity		1			1
Equilibrium		1			1
Ratio	1	1			1
DF	2	1	5	2	0

Table 2. Mathematical model to the problem of Figure 10

Equation	Origin	Number
$n_{1,3} = n_{1,5} + n_{1,1}$	Mass balance in the mixer for the species 1	(5)
$n_{2,3} = n_{2,2}$	Mass balance in the mixer for the species 2	(6)
$n_{1,4} = n_{1,3} - \varepsilon_1 - \varepsilon_2$	Mass balance in the reactor for the species 1	(7)
$n_{2,4} = n_{2,3}$	Mass balance in the reactor for the species 2	(8)
$n_{3,4} = \varepsilon_1$	Mass balance in the reactor for the species 3	(9)
$n_{4,4} = \varepsilon_2$	Mass balance in the reactor for the species 4	(10)
$n_{5,4} = \varepsilon_2$	Mass balance in the reactor for the species 5	(11)
$n_{6,4} = \varepsilon_1 - \varepsilon_2$	Mass balance in the reactor for the species 6	(12)
$n_{1,4} = n_{1,5}$	Mass balance in the separator for the species 1	(13)
$n_{2,4} = n_{2,9}$	Mass balance in the separator for the species 2	(14)
$n_{3,4} = n_{3,8}$	Mass balance in the separator for the species 3	(15)
$n_{4,4} = n_{4,7}$	Mass balance in the separator for the species 4	(16)
$n_{5,4} = n_{5,6}$	Mass balance in the separator for the species 5	(17)
$n_{6,4} = n_{6,6}$	Mass balance in the separator for the species 6	(18)



$$Sel = \frac{n_{4,4}}{n_{1,3} - n_{1,4}} \quad \text{Selectivity for toluene} \quad (19)$$

$$K_{eq} = \frac{\frac{n_{3,4}}{n_{1,4} + n_{3,4} + n_{6,4}} \frac{n_{6,4}}{n_{1,4} + n_{3,4} + n_{6,4}}}{\frac{n_{1,4}}{n_{1,4} + n_{3,4} + n_{6,4}}} p \quad \text{Equilibrium constant of reaction 1} \quad (20)$$

$$R = \frac{n_{2,3}}{n_{1,3}} \quad \text{Ratio of reactants to the reactor} \quad (21)$$

$$n_{3,8} = \frac{100000}{104} \quad \text{Production of styrene} \quad (22)$$

For the students, this problem is completely different from that presented earlier in this article (polynomial equation of degree 2), when in fact it is not; it only needs to be written in standard form before solving it.

Turning the terms on the right side of the equations that constitute the model for the left, we have 18 functions in standard form, as shown in Table 3. In this case, the problem is to solve a system of nonlinear algebraic equations. Figure 11 shows the parts of the m-file used to solve this problem, and in this case we used the function fsolve (file fsolve.m) of Matlab®.

In the case of a problem involving more than one equation, the initial estimate provided in the main program must be in a vector form (with eighteen elements). In the file containing the model (Model\_II.m) the variables are contained in vector x, so, before calculating the value of functions (f1, f2, f3, ..., f18), it is recommended to didactically associate each variable written in modeling with an element of the vector x.

Communication between the files occurs as shown in Figure 7, only changing the embedded function (from fzero to fsolve).

Table 3. System of nonlinear algebraic equations for the problem of Figure 9

Equation	Origin	Number
$f_1 = n_{1,3} - n_{1,5} - n_{1,1}$	Mass balance in the mixer for the species 1	(23)
$f_2 = n_{2,3} - n_{2,2}$	Mass balance in the mixer for the species 2	(24)
$f_3 = n_{1,4} - n_{1,3} + \varepsilon_1 + \varepsilon_2$	Mass balance in the reactor for the species 1	(25)
.	.	.
.	.	.
.	.	.
$f_{18} = n_{3,8} - \frac{100000}{104}$	Production of styrene	(40)

The model of the problem still presents the equation that relates the selectivity for toluene with the temperature; a second-degree polynomial equation is obtained using Matlab®. The equilibrium constant ( $K_{eq}$ ) is calculated using Equation (41):

$$\ln K_{eq} = 15.5408 - \frac{14852.6}{T} \quad (41)$$

#### 4.2 Problems Involving Parameters Estimation (Optimization)

Table 4 presents the data of pressure versus composition (liquid and vapor) for the system: methanol (1) - water (2). The objective is to determine the parameters of the Margules equation that best represents the data, considering that the mixture follows Raoult's law modified.

The Raoult's law modified can be represented by the following:

$$y_i = \frac{P_{sat}^i x_i \gamma_i}{P} \quad (42)$$

where  $\gamma$  is the activity coefficient of the liquid phase,  $P$  the pressure of the system,  $x$  the composition of the liquid phase and  $y$  the composition of the vapor phase. The subscript  $i$  represents the components.

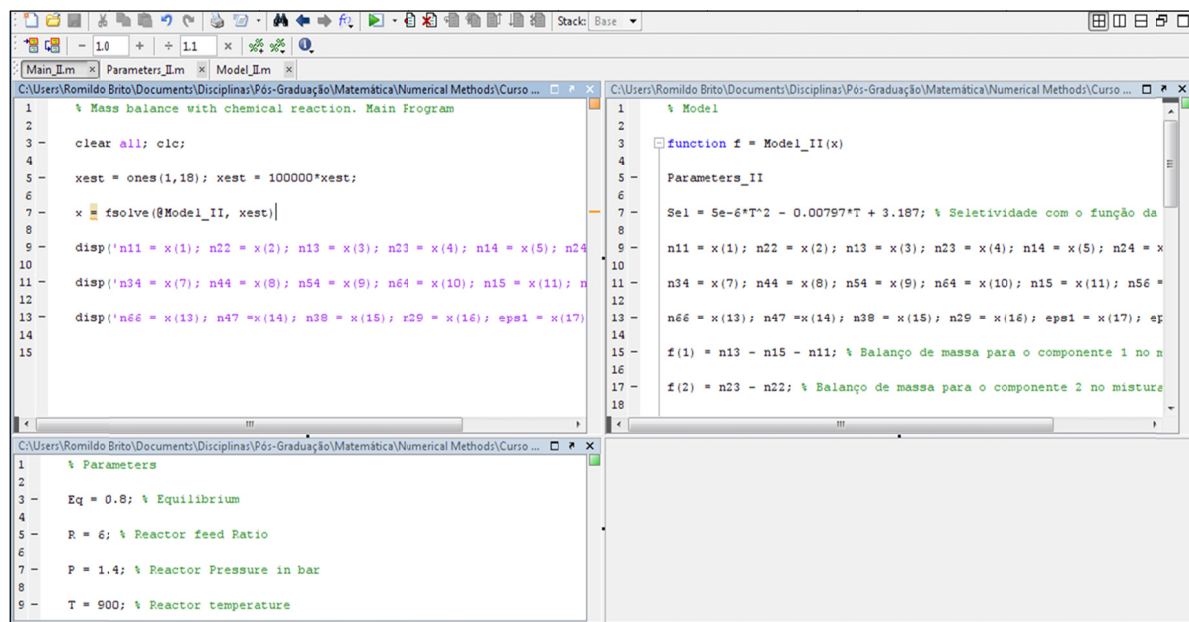


Figure 11. M-files used to determine the flow of flowchart of Figure 10

An expression commonly used to calculate  $\gamma$  is the equation of Margules:

$$\ln \gamma_1 = x_2^2 \left[ A_{12} + 2(A_{21} - A_{12})x_1 \right] \quad (43a)$$

$$\ln \gamma_2 = x_1^2 \left[ A_{21} + 2(A_{12} - A_{21})x_2 \right] \quad (43b)$$

where  $A_{12}$  and  $A_{21}$  are the constants of Margules.

The vapor pressure  $P_{sat}^i$  is calculated using the Antoine equation:

$$\ln P_{sat}^i = A_i + \frac{B_i}{C_i + T} \quad (44)$$

Knowing that the sum of molar fractions in each phase is equal to 1, we obtain the equation:

$$P = P_{sat}^1 x_1 \gamma_1 + P_{sat}^2 x_2 \gamma_2 \quad (45)$$

The Equation (45) is used to calculate the value of  $P$  (calculated) from the data of  $T$  and  $x$ . However, this is only possible if we have the constants of Antoine and Margules.

Considering that the constants of Margules are unavailable, the only variables in Equation (45) are the constants of Margules. Since Table 4 provides data of  $P$  (experimental), the following equation can be defined:

$$f_{obj} = \sum_j (P_{calc} - P_{exp})^2 \quad (46)$$

The function  $f_{obj}$  represents the sum of the difference for all trials  $j$ , between the experimental value ( $P_{exp}$ ) and the theoretical value ( $P_{calc}$ ). To minimize this difference, the parameters to be varied are the constants of Margules ( $A_{12}$  and  $A_{21}$ ). And this is what makes the routines for optimization: minimize or maximize a function.

Since we have no restrictions on the values of  $A_{12}$  and  $A_{21}$ , the problem was solved using the function `fminunc` of Matlab® (also the function `fminsearch` could have been used). Figure 12 shows the m-files used to solve the problem, while Figure 13 shows the communication between them (1-2-3-4-5-2-3-4-5-2-3-4-5 ... 2-3-4-5-6-7-8).

It must be clear to the students that Equation (46) can only be obtained after calculation of  $P_{\text{sat}}^i$  and  $\gamma_b$  which depend on the data of  $P$ ,  $T$  and  $x$ . For this reason, the file containing the parameters and experimental data must be called at the beginning of the file that contains the model. It must be emphasized to the students that all data needed for training the objective function must be provided in the previous rows.

Another important observation is about the calculation of  $P_{\text{sat}}^i$ : In the main file the calculation of this variable is performed using the main feature of Matlab®, that is: operations with matrices and vectors. Moreover, the file containing the model of the problem calculates  $P_{\text{sat}}^i$  using a command “for”, slowing down the execution of the program. The objective was to demonstrate the use of command “for” when the Matlab® is used as programming language.

```

1 % Program to determine Margules parameters using fminunc
2 clear; clc;
3
4 xest = [1 1];
5
6 xopt = fminunc(@Margules_Model, xest);
7
8 A12 = xopt(1); A21 = xopt(2);
9
10 fprintf('\n Margules A12 = %10.4f \n', A12);
11
12 fprintf('\n Margules A21 = %10.4f \n', A21);
13
14 Parameters_Margules
15
16 Peat = exp(A - B ./ (C + Texp));
17
18 G1 = exp( A12*(A21*(1-xexp) ./ (A12*xexp+A21*(1-xexp)))^2 );
19
20 G2 = exp( A21*(A12*xexp ./ (A12*xexp+A21*(1-xexp)))^2 );
21
22 Pcalc = Psat(1)*xexp.*G1 + Psat(2)*(1-xexp).*G2;
23
24 plot(xexp, Pexp, 'o', xexp, Pcalc); xlabel('Liquid');
25 ylabel('P (mmHg)'); legend('Experiments', 'Fitting');
26

```

```

1 % Model
2
3 function f = Margules_Model(A)
4
5 A12 = A(1); A21 = A(2);
6
7 Parameters_Margules
8
9 for i = 1:2
10
11     Psat(i) = exp(A(i) - B(i)/(C(i) + Texp));
12
13 end
14
15 G1 = exp( ((1-xexp).^2).*(A12 + 2*(A21-A12).*xexp) );
16
17 G2 = exp( (xexp.^2).*(A21 + 2*(A12-A21).*(1-xexp)) );
18
19 Pcalc = Psat(1)*xexp.*G1 + Psat(2)*(1-xexp).*G2;
20
21 dif = (Pexp - Pcalc).^2;
22
23 f = sum(dif);
24
25
26

```

```

1 % File with necessary data to obtain the objective function
2
3 A = [16.5938 16.2620]; B = [3644.30 3799.89]; C = [239.76 226.35];
4
5 xexp = [0.0 0.1686 0.2167 0.3039 0.3681 0.4461 0.5282 0.6044 0.6804 0.7255 0.
6
7 yexp = [0.0 0.5714 0.6268 0.6943 0.7345 0.7742 0.8085 0.8383 0.8733 0.8921 0.
8
9 Pexp = [19.953 39.223 42.984 48.852 52.784 56.652 60.614 63.998 67.924 70.226
10
11 Texp = 60;

```

Figure 12. M-files used to determine the set of Margules

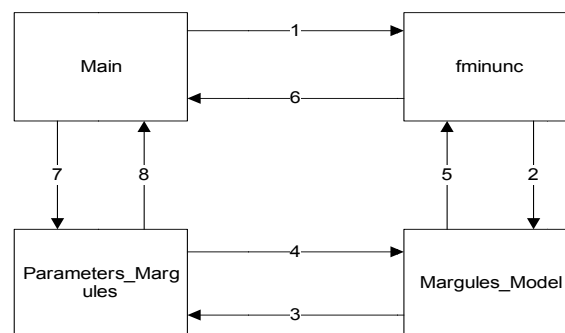


Figure 13. Block diagram of communication between the files of Figure 12

#### 4.3 Problems Involving Ordinary Differential Equations (ODE)

Figure 14 shows two CSTR reactors. Considering the isothermal operation, 1st order reaction, constant reactor volumes ( $V_1$  and  $V_2$ ) and density of the mixture, the mathematical model that describes the variation of the concentration of component A ( $CA_1$  and  $CA_2$ ) in the output of the reactor is given by two nonlinear ordinary differential equations (ODE):

$$V_1 \frac{dCA_1}{dt} = F_0 CA_0 - F_1 CA_1 - KCA_1 V_1 \quad (47a)$$

$$V_2 \frac{dCA_2}{dt} = F_1 CA_1 - F_2 CA_2 - KCA_2 V_2 \quad (47b)$$

where  $F$  is the input flow (or output) of the reaction and  $K$  is the rate of specific reaction.

The ODE system was solved using the ode45 function (file ode45.m) of Matlab®. Figure 15 shows the m-files used to solve the problem. Communication between the files occurs as shown in Figure 7, only changing the embedded function (from fzero to ode45), and the end of the process is defined by the final time of integration.

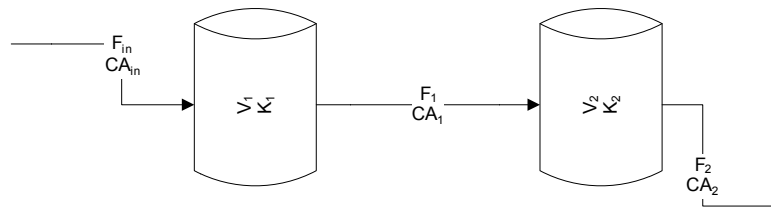


Figure 14. Isothermal CSTR

In the file containing the problem itself, the ODEs are placed as elements of a vector (dydt), similarly to a system of algebraic equations, and it will only be formed at the last line of the command. It is also interesting to pay attention to the fact that the elements of the vector dydt are defined by the right hand side of each EDO. Again, a command “if” was used with the aim to demonstrate the use of Matlab® as a programming language.

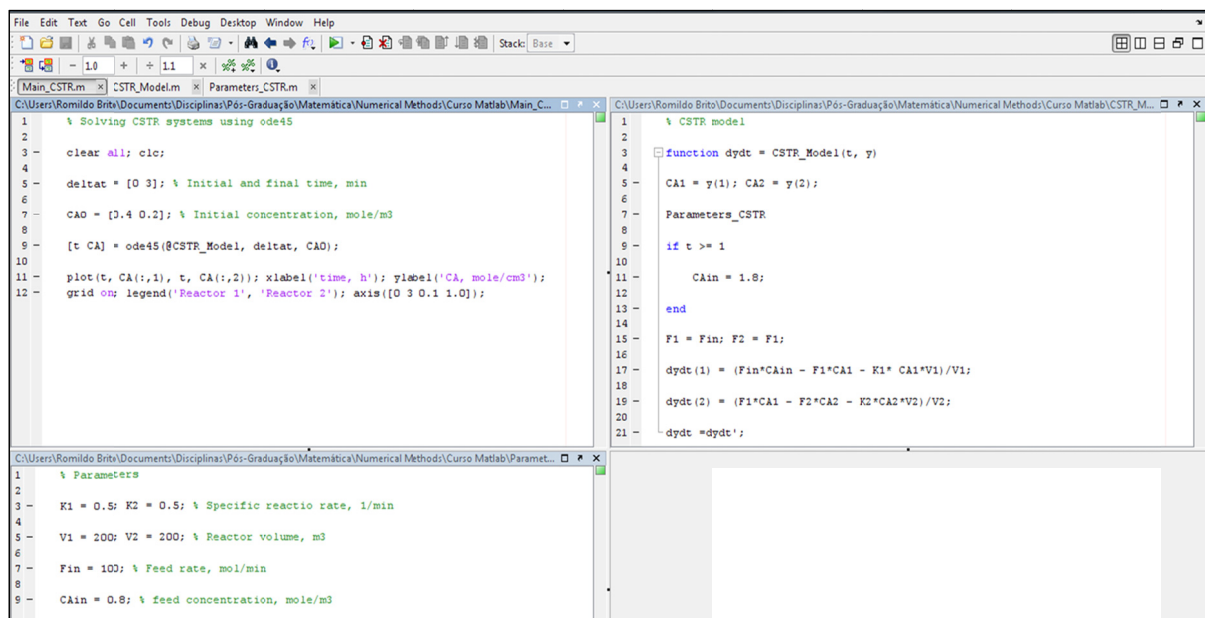


Figure 15. M-files used to solve the problem of Figure 14

## 5. Concluding Remarks

Among the solutions proposed to solve the considered problem, teaching algorithms and programming language is the most important, because if the students think in a systematic way, the choice of the software program will depend on the convenience and the type of problem being addressed. In other words, regardless of the chosen software program, it is essential that engineering students have a solid basis in algorithm and programming.

The use of the software cited in this paper has great potential to assist in the task of teaching engineering. However, it is necessary to be cautious in the use of the software to avoid the students minimizing the importance of programming, which is usually related to the logical and systematic thinking.

After nearly two decades working with the problem of how to make students think logically and systematically, we can say: minimizing the task of teaching engineering by only introducing tools that make this task easier for the students might end up being a pitfall for the future of the engineering.

Professors with an aversion to the use of computers in solving problems will not change their minds. If we want our students using the computer for solving systematic problems of engineering, the first step is to get professors who lecture the first courses of the professional content of the Chemical Engineering curriculum to use software programs repeatedly. We also find that professors should teach students how to solve problems both as they already do and by using a software program. The more students graduating with a solid programming background, the more professors inclined to the use of software programs there will be. Furthermore, it is essential that the disciplines of algorithms and programming language are taught by Chemical Engineering professors, for these professionals have the capability of using programming language to solve specific problems that call for a previous knowledge of Chemical Engineering, which is vital for any major in Chemical Engineering.

## References

- Binous, H. (2008). Equilibrium-Staged Separation Using Matlab<sup>®</sup> and Mathematica<sup>®</sup>. *Chemical Engineering Education*, 42(2).
- Bjedov, G., & Andersen, P. K. (1996). Should Freshman Engineering Students Be Taught a Programming Language? *Proceedings of the 26th Annual Frontiers in Education*, 1, 92-96.
- Chapra, S. C., & Canale, R. P. (2008). *Numerical Methods for Engineers*. McGraw-Hill.
- Felder, R. M., & Brent, R. (2005). Death by PowerPoint. *Chemical Engineering Education*, 39(1).
- Finlayson, B. A. (2006). *Introduction to Chemical Engineering Computing*. Wiley. <http://dx.doi.org/10.1002/0471776688>
- Jones, J. B. (1988). The Non-use of Computers in Undergraduate Engineering Science Courses. *J. Eng. Educ.*, 88.
- Kantor, J. C., & Edgar, T. (1996). Computing Skills in the Chemical Engineering Curriculum. Computers in Chemical Engineering Education, CACHE, Austin, TX.
- Nasri, Z., & Binous, H. (2008). Application of Soave-Redlich-Kwong Equation of State Using Mathematica<sup>®</sup>. *Journal of Chemical Engineering of Japan*, 40.
- Ninous, H., Al-Mutairi, E., & Faqir, N. (2011). Study of the Separation of Simple Binary and Ternary Mixtures of Aromatic Compounds. *Computer Applications in Engineering Education*. <http://dx.doi.org/10.1002/cae.20533>
- Shacham, M., Brauner, N., Ashurst, W. R., & Cutlip, M. B. (2008). Can I Trust This Software Package? An Exercise in Validation of Computational Results. *Chemical Engineering Education*, 42(1).
- Shacham, M., Brauner, N., & Cutlip, M. B. (2003). Efficiently Solve Complex Calculations. *Chem. Eng. Prog.*, 99(10), 56-61.
- Shacham, M., Brauner, N., & Cutlip, M. B. (2011). Prediction and Prevention of Chemical Reaction Hazards. *Chemical Engineering Education*, 35(4).
- Shacham, M., & Cutlip, M. B. (2004). *Combining Engineering Problem Solving with Numerical Methods to Enhance Learning Effectiveness*. International Conference on Engineering Education and Research (iCEER).
- Shachan, M., & Cutlip, M. B. (2008). *Problem Solving in Chemical Engineering with Numerical Methods*. Prentice Hall.
- Shacham, M., Cutlip, M. B., & Brauner, N. (2009). From Numerical Problem Solving to Model-Based Experimentation. *Chemical Engineering Education*, 43(4), 315-321.
- Skogestad, S. (2008). *Chemical and Energy Process Engineering*, CRC.
- Vasconcelos, L. G. S., Alves, J. J. N., Araújo, A. C. B., & Brito, R. P. (2008). McCabe-Thiele Method Revisited - Solving Binary Distillation Problems with Nonconventional Specifications. *Journal of Chemical Engineering of Japan*, 41(9), 933-938. <http://dx.doi.org/10.1252/jcej.08we024>