HOW TO GUIDE CHEMICAL ENGINEERING STUDENTS IN THE SOLUTION OF COMPLEX ENGINEERING PROBLEMS

Carlos Carbonell, Salvador C. Cardona, Iván Domínguez, Vicent Fombuena, María Fernanda López-Pérez, Jaime Lora

Universitat Politècnica de València (UPV), Departamento de Ingeniería Química y Nuclear, Escuela Politécnica Superior de Alcoy (SPAIN)

Abstract

If we analyse some of the specific skills of Chemical Engineering degree, and it would not be very different in other disciplines of engineering, the verbs design, analyse or simulate stand out above all. In essence, calculating is an intrinsic activity for the engineer and, therefore, for engineering students.

How is this activity developed? When we face a real problem we translate it into mathematical language (modelling). We solve the resulting mathematical problem to obtain the mathematical solution (simulation). The analysis of the results allows us to extract information from the real problem. If the interpretation of the results does not fit with the real problem studied, we must rethink the mathematical model obtained, and so on.

Teachers often restrict the real problems that students face to simple situations so that the resulting mathematical problem is simple and, if possible, provides an analytical solution. Why should we do this today if we have numerical methods, a big computing power and mathematical software available to our students? Examples of complex mathematical problems arise from analysis and design of heat exchangers, chemical reactors, distillation columns, etc, both in steady and transitory regimes. These mathematical problems can be classified in groups: solving algebraic equations, ordinary differential equations or partial differential equations systems.

According to this classification, some teachers of the Chemical Engineering degree at Campus of Alcoy are providing our students with Matlab guide templates to solve these types of mathematical problems, regardless of the subjects in which these problems appear. This action makes it easier for students to face real problems by reducing the mathematical difficulties associated.

This paper shows how these Matlab guide templates are and how students use them in different subjects. It allows students to focus on the engineering aspect of the processes, leaving the difficulty of the associated mathematical problem in a second plane. It also makes easy to modify process parameters to quickly see the effect on the main variables of the process (what if? analysis). Consequently, students improve their ability to analyse and interpret the results obtained and their critical thinking skills. And, finally, they know useful computer tools that can be used both academically and professionally.

Keywords: Mathematical modelling, Problem solving, Matlab.

1 INTRODUCTION

From the analysis of the specific skills belonging to Chemical Engineering degree, some verbs can be highlighted: to design, to analyse, to model or to simulate stand out above all. In short, calculating is an intrinsic activity for the chemical engineer and, therefore, for chemical engineering students. It would not be very different in other disciplines of engineering, so the methodology described in this paper can be easily extrapolated to other engineering fields.

Figure 1 shows the process followed for modelling and calculating [1]. When we face a real world problem, the first step is to translate it into a mathematical problem using our knowledge of the phenomena that we think are involved in the real problem. This activity is the modelling step and we apply in it all our engineering understanding and expertise about the problem we are dealing with. The next step is to solve the resulting mathematical problem for obtaining a mathematical solution. This task is the simulating step where we apply in it our best knowledge about mathematical methods and use the appropriate support mathematical tools. The solution is analysed in the next step for extracting information from the real world, using once more our engineering knowledge about the real problem. If the interpretation of the results does not fit with the real problem studied, we must rethink again the mathematical model obtained or the mathematical strategy used for solving it, and so on.

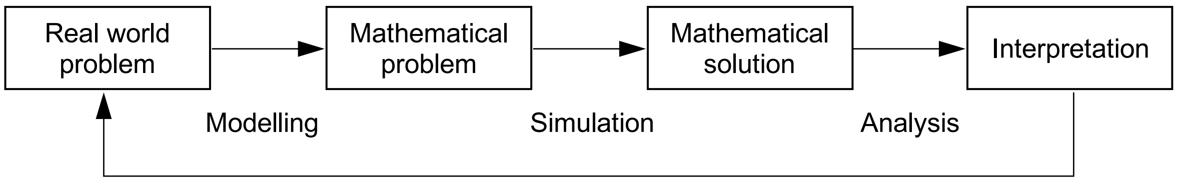


Figure 1. Real world modelling and calculating process [1]

According to the process above described, engineering teachers often restrict the real world problems that students face to simpler situations so that the resulting mathematical problem is easier to solve and, if possible, provides an analytical solution. From our point of view, this action moves our students away from the real problems that they will face in the future in their working life. Why should we do this today if we have numerical methods, a big computing power and mathematical software available to our students? Almost all the engineering students have mathematical and numerical methods subjects in their syllabus, most of them manage own or university powerful personal computers and can access to commercial (Matlab, Mathematica, MathCad, etc) or open source (Scilab, Octave, SageMath, etc) mathematical software.

A big amount of complex mathematical problems arises in Chemical Engineering from real problems related to analysis and design of hydraulic installations, heat exchangers, chemical reactors, distillation columns, absorption columns, etc. These mathematical problems can be classified in groups according to the type of equations resulting from the mathematical models: linear and non-linear algebraic equations, ordinary differential equations (ODEs) or partial differential equations (PDEs).

According to this mathematical classification, we have developed Matlab guide templates that are provided to our students. They can use those templates to solve different types of mathematical problems, regardless of the subjects in which these problems appear and regardless of the type of unit operation involved. This action facilitates students to face real problems by reducing the mathematical difficulties associated. This paper shows how these Matlab guide templates are and how students use them.

2 METHODOLOGY

The first stage that some teachers of the Chemical Engineering degree at Campus of Alcoy have undertaken is the identification of the different types of model equations that arise from the application of engineering tools as mass and energy balances, mass and heat transfer, chemical kinetics, thermodynamical and equilibrium relations and state equations. These tools usually are applied to the analysis and design of hydraulic installations, heat exchangers, chemical reactors, distillation columns, absorption columns, etc. working in steady or not steady state inside control volumes considered both lumped and distributed parameters, Table 1. Algebraic equations, ODEs and PDEs are the typical equations involved in the mathematical problems that students need to solve in subjects as Chemical Engineering Fundamentals (2nd course), Chemical Kinetics (2nd course), Chemical Engineering Lab (2nd and 3rd course), Chemical Reactors (3rd course), Process Analysis and Simulation (3rd course) and Chemical Engineering Industrial Processes (4th course).

For each type of mathematical problem, we have developed some Matlab files that guide students for solving them, Table 1. These template files include void spaces along their structured programming lines that students fill with the equations corresponding to the problem they want to solve and the additional required information as constants, parameters, operation variables, initial conditions, boundary conditions, integration time, etc. So, these Matlab files help students in the simulation step but not in the modelling one, because they have to develop their own mathematical models prior to use the templates. This strategy allows students to focus on the engineering aspect of the processes, leaving the difficulty of the associated mathematical problem in a second plane. It also makes easy to modify process parameters or operation variables to quickly see the effect on the dependent variables of the processes, in the line of a what if? analysis. Students are trained in class in Matlab and in the use of these template files that can be used for solving their homework or teamwork, in their exams or also after graduating, along their working life.

Table 1. Model equations classification

Real Problems	Engineering Tools	Regime	Spatial Distribution	Model Equations	Matlab Templates	Subjects involved
Analysis and design of hydraulic installations, heat exchangers, chemical reactors, distillation columns, absorption columns, etc.	Mass and energy balances, mass and heat transfer, chemical kinetics, thermodynamical and equilibrium relations and state equations	Steady-state	Lumped parameters	Algebraic equations	steady_state_solver.m, mass_balance.m and mass_energy_balances.m	Chemical Engineering Fundamentals, Chemical Engineering Industrial Processes, Chemical Engineering Lab, Chemical Kinetics, Chemical Reactors
			Distributed parameters	Ordinary differential equations (ODEs)	data.m, principal.m and ODEs.m	Chemical Engineering Lab, Chemical Reactors, Process Analysis and Simulation
		Dynamics (non steady-state)	Lumped parameters	Ordinary differential equations (ODEs)		
			Distributed parameters	Partial differential equations (PDEs)	data.m, principal_PDE.m and ODEs_PDE.m	Process Analysis and Simulation

Three Matlab files have been developed for solving algebraic equation systems mainly derived from the application of mass and energy balances in steady state, Table 1: *steady_state_solver.m*, *mass_balance.m* and *mass_energy_balances.m*. While the last two files let students to write all the algebraic equations, the first one is the main file that manages *mass_balance.m* and *mass_energy_balances.m* and includes the calculation algorithm [2].

The study of distributed parameters systems in steady state or lumped parameters systems in non-steady state generates ODEs that are solved using also three Matlab files, Table 1: *data.m*, *principal.m* and *ODEs.m*. While *data.m* is used for incorporating all the constants, parameters, operation variables, initial conditions and integration time, *ODEs.m* let students to write the ODEs system. On the other hand, *principal.m* manages the data file, solves the ODEs system and plots all the figures necessary to analyse and correctly interpret the results obtained. The content and application of these templates is described in detail in the following sections of this paper

For solving PDEs obtained from distributed parameters systems working in non-steady state, also three Matlab files have been prepared, in a similar way to ODEs: *data.m*, *principal_PDE.m* and *ODEs_PDE.m*. The numerical method adopted for the solution of PDEs is the Method of Lines (MOL), that is based on the discretization of the spatial variable so transforming the PDEs system in an ODEs system, with good approximations [1]. Therefore, these files are similar to those used for solving ODEs systems. In that sense, *data.m* is also used for defining all the constants, parameters, operation variables, initial conditions, number of slides used in the discretization and integration time. In *ODEs_PDE.m* the ODEs system is written together the discretized boundary conditions. Finally, *principal_PDE.m* also manages the data file, solves the ODEs system, plots all the figures and does some videos that show the temporary evolution of variables profiles, making easier the correct analysis of the results [3].

When the computation skills of students improve along their academic journey, they are capable of enhancing the template files, adapting them to their needs including more capabilities not considered in the basic templates. Alternatively, they can combine some of the templates when they have to solve problems that include more than one type of equations, as algebraic equations together with ODEs, for example.

3 RESULTS

In this section the content and application of the template files used for solving ODEs system is described in detail using a class example solved in the subject Process Analysis and Simulation (3rd course)

3.1 Problem description

Figure 2 shows an isotherm continuous stirred tank reactor (CSTR), with constant volume V_{reb}, where a reaction mechanism including series and parallel reactions takes place, with kinetic rate constants k_1, k_2 and k_3. Considering that there is only component A at the inlet stream, with concentration C_{Ae} and flow rate Q_e, determine the temporary evolution of components A, P, S and T at the outlet stream (C_A, C_P, C_S and C_T), assuming the data shown in Table 2.

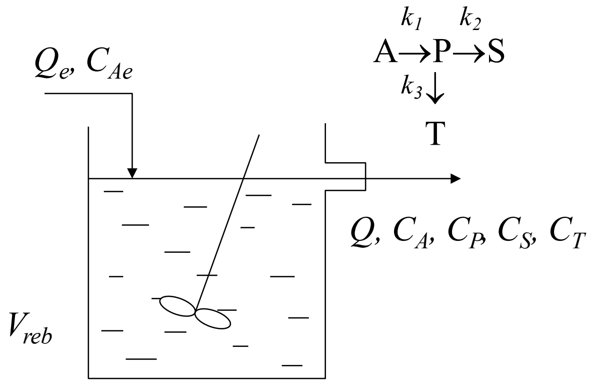


Figure 2. Process diagram

Table 2. Problem data

Variable	Value	Units
V_{reb}	3	L
C_{A0}	50	mol/L
C_{P0}	5	mol/L
C_{S0}	10	mol/L
C_{T0}	0	mol/L
C_{Ae}	5	mol/L
k_1	1	min^{-1}
k_2	1.5	min^{-1}
k_3	0.5	min^{-1}
Q_e	2	L/min
t_{max}	10	min

3.2 Modelling

According to the modelling and calculating process shown in Figure 1, the first step is the modelling of the CSTR applying mass balances to each component considering non-steady state and perfect mixing in the tank (lumped parameter system). The mathematical model obtained in this case is the ODEs system illustrated in Figure 3, with its corresponding initial conditions.

$$\frac{Q_e}{V_{reb}} \cdot (C_{Ae} - C_A) - k_1 \cdot C_A = \frac{dC_A}{dt}$$

$$-\frac{Q_e}{V_{reb}} \cdot C_P + k_1 \cdot C_A - (k_2 + k_3) \cdot C_P = \frac{dC_P}{dt}$$

$$-\frac{Q_e}{V_{reb}} \cdot C_S + k_2 \cdot C_P = \frac{dC_S}{dt}$$

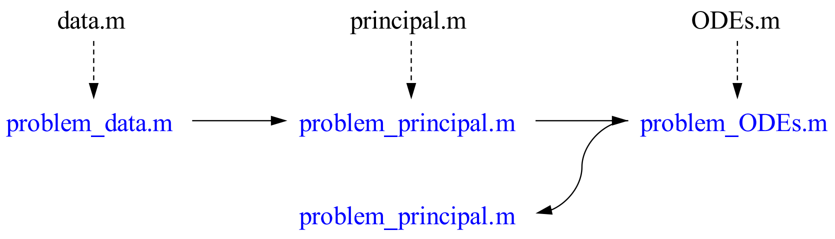
$$-\frac{Q_e}{V_{reb}} \cdot C_T + k_3 \cdot C_P = \frac{dC_T}{dt}$$

$$C_A(0) = C_{A0} \quad C_P(0) = C_{P0} \quad C_S(0) = C_{S0} \quad C_T(0) = C_{T0}$$

Figure 3. Mathematical modelling

3.3 Simulation

It is in the simulation step where the template files *data.m*, *principal.m* and *ODEs.m* are used, following the scheme described in Figure 4. These files are adapted to the specific problem based on ODEs systems that students face in each moment, as the problem studied in this paper.



$$\begin{tikzcd}
\text{data.m} \arrow[d, dashed] & \text{principal.m} \arrow[d, dashed] & \text{ODEs.m} \arrow[d, dashed] \\
\text{problem\_data.m} \arrow[r] & \text{problem\_principal.m} \arrow[r] & \text{problem\_ODEs.m} \arrow[dl, bend right] \\
 & \text{problem\_principal.m} &
\end{tikzcd}$$

Figure 4. Scheme followed for using the template files.

First of all, students prepare *problem_data.m* file from *data.m* file incorporating tank dimensions, kinetic rate constants, initial conditions, inlet flowrate, inlet A concentration and simulation time enumerated in Table 2, as can be seen in Figure 5. For the sake of clarity, the information introduced by students in the files is written in red, while the remainder is the programming lines included in the original template files. Users are free to define the name of the variables using the nomenclature they desire. Preparing this data file can be a very time-consuming process when there is many information to take into account. In these cases, the teacher can provide this file to students when they are doing work in class or in exams, for example.

```
% Definition of constants, operating conditions, initial conditions,
% integration time, etc.

% Tank dimensions
Vreb=3; % L
% Kinetic rate constants
k1=1; % min^(-1)
k2=1.5; % min^(-1)
k3=0.5; % min^(-1)
% Initial conditions
CA0=50; % mol/L
CP0=5; % mol/L
CS0=10; % mol/L
CT0=0; % mol/L
% Inlet flowrate
Qe=2; % L/min
% Inlet A concentration
CAe=5; % mol/L
% Simulation time
tmax=10; % min
```

Figure 5. problem_data.m file. The information entered by students is written in red.

Secondly, students fill in *principal.m* file transforming it into *problem_principal.m* file, Figure 6. This file reads the data included in *problem_data.m* file and store them automatically in a Matlab structure named *param.* The goal is to reduce the number of arguments in the function that will be described later. After that, the initial conditions C_{A0}, C_{P0}, C_{S0} and C_{T0} are assigned to a generic dependent variable named Y_0. In order to use the template files for any problem, regardless of the name of the variables involved, it is necessary to name generic variables, both independent X and dependent Y, and relate these generic variables with the specific ones. In this problem in particular, the specific independent variable is the time, t, and the specific dependent variables are the concentrations, C_A, C_P, C_S and C_T. It is also indicated the integration interval, with X_{span} variable, and the name of the file, *problem_ODEs.m*, that includes the ODEs system described in Figure 3.

```
clc
clear all
close all
%% DATA READ AND ACQUISITION
% Read data file and automatically create param structure.
problem_data
variables_workspace=whos;
num_variables=length(variables_workspace);
for i=1:num_variables
    eval(['param.',variables_workspace(i).name,'=',variables_workspace(i).name,';']);
end
%% DEFINE ADDITIONAL NEEDED FUNCTIONS
% Define additional functions needed for ODE's system solving
%% ODE SYSTEM SOLVING
% Initial conditions for ODE system. Assign the particular initial
% conditions to the generic initial conditions Y0
Y0=[CA0,CP0,CS0,CT0];
% Define the generic integration interval Xspan=[Xmin Xmax].
% Other option is Xspan=[Xmin X1 X2 ... Xmax]
Xspan=[0 tmax];
% Identify the ODE system to solve
odefun=@(X,Y) problem_ODEs(X,Y,param);
```

Figure 6. Initial part of problem_principal.m file. The information entered by students is written in red.

Thirdly, students complete *problem_ODEs.m* file on the basis of *ODEs.m* file, Figure 7. This file contains a Matlab function only with three arguments, *X*, *Y* and *param*, regardless of the complexity of the problem addressed. In this case, after extracting automatically all the information included in structure *param*, the generic variables *X* and *Y* are assigned to the particular ones *t*, C_A, C_P, C_S and C_T. This lets students to program ODEs system likewise they write them in the paper, which makes it easier to debug possible transcription errors. At the end of this file, the left hand side members of ODEs are assigned to the generic variable *dYdX.*

```
function dYdX=problem_ODEs(X,Y,param)
% Automatically extract parameters from param structure
nombres_variables=fieldnames(param);
longitud_param=length(nombres_variables);
for i=1:longitud_param
    eval([nombres_variables{i},'=param.',nombres_variables{i},';']);
end
% Assign generic independent variable X to the particular one
t=X;
% Assign generic dependent variables Y to the particular ones
CA=Y(1);
CP=Y(2);
CS=Y(3);
CT=Y(4);
% Define additional variables for writing the ODE system more easily
% Write the ODE system
dCAdt=(Qe/Vreb)*(CAe-CA)-k1*CA;
dCPdt=-(Qe/Vreb)*CP+k1*CA-(k2+k3)*CP;
dCSdt=-(Qe/Vreb)*CS+k2*CP;
dCTdt=-(Qe/Vreb)*CT+k3*CP;
% Assign particular ODE left hand side to the generic dYdX (vector column)
dYdX=[dCAdt;dCPdt;dCSdt;dCTdt];
end
```

Figure 7. problem_ODEs.m file. The information entered by students is written in red.

Finally, students come back to *problem_principal.m* file, Figure 8. This file contains the numerical method used for solving ODEs system, *ode15s* in this case, the absolute and relative tolerances and the maximum step size applied. Normally this information is not modified, but students have the possibility to change it, if necessary. Again, the generic variables X_{sim} and Y_{sim} are assigned to the particular ones t_{sim}, C_{Asim}, C_{Psim}, C_{Ssim} and C_{Tsim}. Although it is not the case in this problem, students can do additional calculations from the variables obtained after using *ode15s* solver.

The last part of the file is devoted to plot the variables in the best possible way that facilitates the interpretation and analysis of the results. In this particular case, temporary evolution of components A, P, S and T concentrations has been plotted, Figure 9, after running *problem_principal.m* file. It can be clearly seen that component A behaves as a reactant while components P, S, T have the typical behavior of products in series reactions.

Once the three files are completed it would be quick and easy to evaluate what would happen if the values of some parameters changed. For example, the volume of the tank, the kinetic rate constants, the inlet flowrate, etc. Students only need to change the value of the required variable in *problem_data.m* file and run *problem_principal.m* file again, as can be seen in Figure 10 where the effect of inlet A concentration, C_{Ae}, is analysed. Incorporation of a *for* loop in *problem_principal.m* file let students to compare quickly the effect of modifying those parameters.

```
% Solve the ODE system and assign the results to the particular dependent
% variables. By default MaxStep=(Xspan(end)-Xspan(0))/10.
opciones=odeset('RelTol',1e-8,'AbsTol',1e-8,'Refine',10,'MaxStep',[]);
[Xsim,Ysim]=ode15s(odefun,Xspan,Y0,opciones);
tsim=Xsim;
CAsim=Ysim(:,1);
CPsim=Ysim(:,2);
CSsim=Ysim(:,3);
CTsim=Ysim(:,4);
%% ADDITIONAL DATA MANIPULATION
% Additional variables are calculated or save data to an ASCII file.
%% PLOT VARIABLES
figure(1)
subplot(2,2,1)
plot(tsim,CAsim)
xlabel('t (min)')
ylabel('C_A (mol/L)')
subplot(2,2,2)
plot(tsim,CPsim)
xlabel('t (min)')
ylabel('C_P (mol/L)')
subplot(2,2,3)
plot(tsim,CSsim)
xlabel('t (min)')
ylabel('C_S (mol/L)')
subplot(2,2,4)
plot(tsim,CTsim)
xlabel('t (min)')
ylabel('C_T (mol/L)')
savefig('figura.fig')
```

Figure 8. Final part of problem_principal.m file. The information entered by students is written in red.

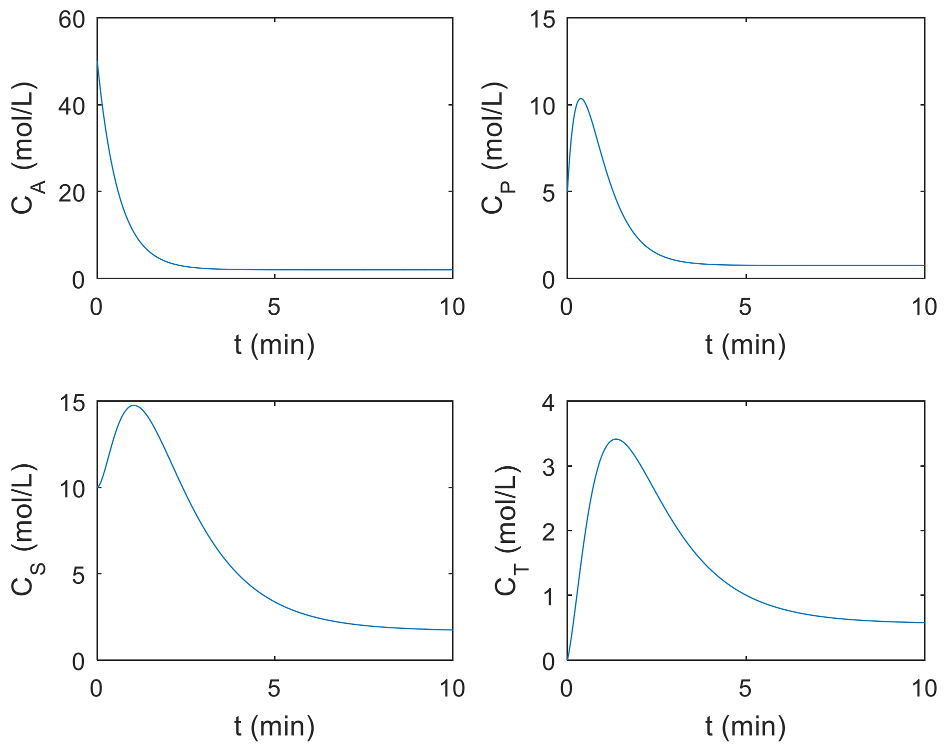


Figure 9. Problem simulation.

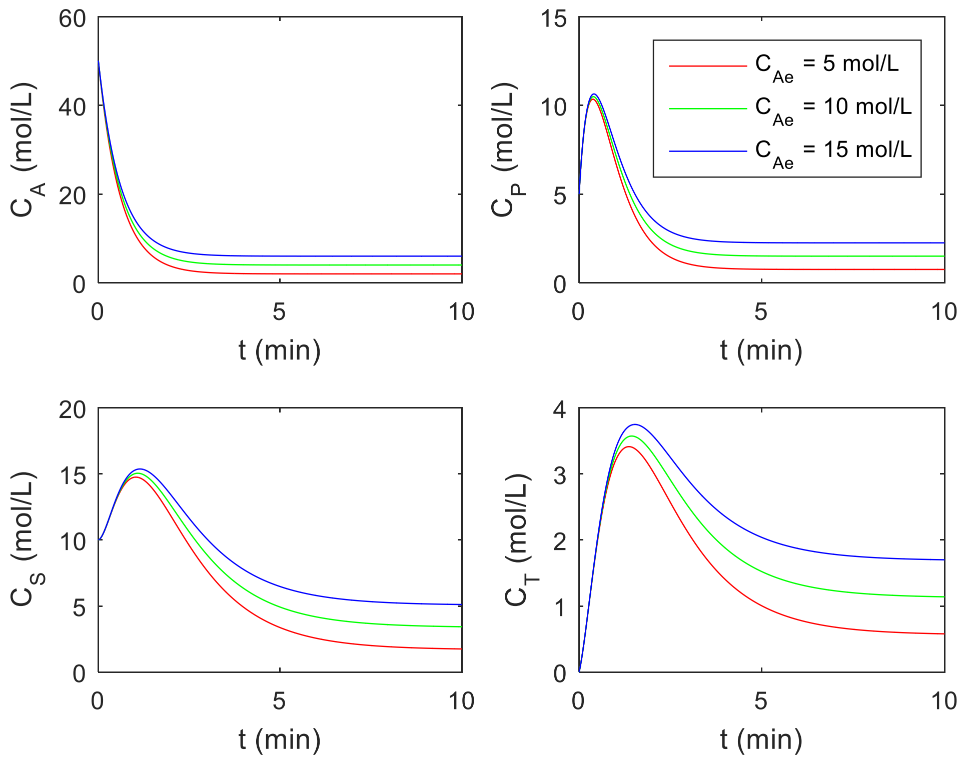


Figure 10. Effect of changing inlet A concentration, C_{Ae}.

4 CONCLUSIONS

Use of Matlab guide templates described in this paper allows students to focus on the engineering aspect of complex processes, leaving the difficulty of the associated mathematical problem in a second plane. It also makes it easy to modify process parameters, operation variables, physic-chemical properties, etc. to quickly see the effect on the dependent variables of the process (what if? analysis). These templates are devoted to solve mathematical problems whose equations belong to algebraic equation, ODEs and PDEs systems, which are those that are frequently found in the mathematical modelling of Chemical Engineering processes, and in other branches of engineering.

Consequently, students improve their ability to analyse and interpret the results obtained and their critical thinking skills. And, finally, they know useful computer tools that can be used both academically and professionally.

REFERENCES

[1] K. Hangos and I. Cameron, *Process modelling and model analysis*. San Diego: Academic Press, 2001.

[2] S. C. Cardona Navarrete and V. Fombuena Borrás, “Cómo obtener con Matlab una solución numérica a un problema de balance de materia en régimen estacionario,” 2020, Accessed: Jan. 13, 2021. [Online]. Available: https://riunet.upv.es:443/handle/10251/145909.

[3] S. C. Cardona Navarrete, L. Agud Albesa, M. L. Pla Ferrando, and M. Boix García, “Cómo guiar a los alumnos en la simulación de modelos matemáticos complejos en Ingeniería Química,” *Model. Sci. Educ. Learn.*, vol. 13, no. 1, p. 37, Jan. 2020, doi: 10.4995/msel.2020.12128.