pp. 41 – 52 in A. Kraslawski and I. Turunen (Eds.),
Proceedings Supplement Volume of the European Symposium on Computer Aided
Process Engineering-13, Lappenranta University of Technology, Lappenranta, Finland

Modular and Sequential Construction of Complex Process Models – Applications to Process Hazard Assessment

M. Shacham[1], N. Brauner[2] and M. B. Cutlip[3]
[1]Chem. Eng. Dept., Ben-Gurion University, Beer-Sheva 84105, Israel
[2]School of Engineering, Tel-Aviv University, Tel-Aviv 699 78, Israel
[3]Chem. Eng. Dept., University of Connecticut, Storrs, CT 06269, USA

Abstract

Construction of dynamic simulation programs for complex processes by a modular and sequential approach is considered. The complex simulation model is broken down into smaller modules. These modules are modeled and tested using a mathematical software package that requires modest technical coding efforts for computer implementation of the model. The verified model is then used as the basis for the program that represents the entire process and for documentation. The proposed approach is demonstrated for the construction of a simulation program for two-stage operation of an exothermic batch reactor under emergency conditions. It is shown that the effort required for the construction of the simulation program and the associated documentation is significantly reduced. Also, it is ensured that the models presented in the documentation are identical those used in the program.

1. Introduction

Dynamic modeling and simulation of chemical processes is being extensively used for quantitative hazard assessment (Shacham, Brauner and Cutlip, 2000, Srinivasan and Venkatasubramanian, 1999) and in operator training for emergency situations. Process hazard assessment (PHA) involves several steps, which include hazard identification, assessment of the likelihood of failure and estimation of the potential for damage and/or injury associated with specific incidents. Dynamic simulation is a very important component of PHA, since it can quantitatively predict the consequences of critical component failures. It is essential for operator training, as in reality process faults rarely happen and intentional creation of emergency situations for training purposes is usually considered unacceptable.

The process models used for PHA must represent the process operation in both normal and abnormal conditions. Commercial simulation packages can represent the process operation only in normal operating conditions, thus models for dynamic simulation programs to be used for PHA must be prepared individually, for every process studied.

Preparation and debugging of complex process models is a tedious and time-consuming task. After the program has been prepared and tested, some of the effort must be duplicated for preparing the documentation. In particular, the model equations and the values of the numerical constants must be retyped in a format appropriate for inclusion in documents read by human beings, not the computer. This duplication of efforts is also an error prone process. Brauner et al. (1996) and Shacham et al. (2002) present several examples where the published mathematical models behave completely differently than

the actual process models because of typographical errors and rounding of numbers in the published model.

There are several mathematical software packages available (Maple[1], Mathcad[2] and POLYMATH[3], for example) which accept the model equations in formats which are very close to their mathematical form. These enable simulation of processes of small and medium size and complexity (a single unit operation or one particular stage of operation, for example) with little technical coding efforts. However, for simulating an entire process, consisting of several unit operations and several stages of operation, where different solvers (i.e. stiff or non-stiff) may be required for the different modules, the power and flexibility of a programming language or a mathematical software package that supports programming (such as MATLAB[4]) are needed.

In this paper we present a new approach, where a complex simulation model is broken down into smaller modules. These modules are modeled and tested using a mathematical software package. The verified model is then used as the basis for the program that represent the entire process and also as the basis for the documentation. The use of this approach is demonstrated by modeling two stages of operation of a cooled exothermic reactor under emergency conditions. The POLYMATH software package is used to prepare and debug the models of the various stages. The POLYMATH models are used as the basis for the documentation and they are compiled for execution by MATLAB. The complete simulation model is assembled and executed using MATLAB.

2. Sequential and Modular Model Building

In the first step of the model building, the complex simulation model is broken down into smaller modules where each module represents one unit operation or one particular stage of operation. The modules are modeled and tested using a software package (say, POLYMATH) that requires very little technical coding effort. The "user friendly" features of POLYMATH in entering the equations are worth mentioning here. The notation used in the equation entry is almost the same as in the problem definition. POLYMATH issues warnings for undefined variables, so that errors due to 'misspelling' of the variable names (e.g. using 0 instead of the letter *o)* can be easily detected. The needed equations can be entered in the same order as they appear in the problem definition, even if the calculation order must be different, since POLYMATH reorders the equations according to the calculation sequence.

After testing each of the modules separately, they are combined into a simulation program using a programming language, or a mathematical software package that supports programming (say, MATLAB). To minimize the probability of introducing errors into the model equations, the POLYMATH input for the various modules can be converted directly to MATLAB functions. The changes required are essentially: 1

---

[1] Maple is trademark of Waterloo Maple, Inc. (http://www.maplesoft.com)
[2] Mathcad is trademark of MathSoft, Inc. (http://www.mathsoft.com)
[3] POLYMATH is copyrighted by M. Shacham, M. B. Cutlip and M. Elly (http://www.polymath-software.com)
[4] MATLAB is a trademark of The Math Works, Inc. (http://www.mathworks.com)

Putting the expressions of the derivatives and the associated variables in column vectors and 2. Reordering the equations so that the variable and constant definitions are first, the explicit algebraic equations are second and the differential equations are the last. There are also some syntax changes in function names, "if" statements and the identification of comments.

The main effort, in preparing the MATLAB program is involved in the preparation of the main (executive) program. The main program includes means enabling the user to introduce changes in the simulation parameters (e.g. initial values of the variables, model parameter values, starting time and duration of various events). It calls the appropriate integration routines to integrate the modules of the various stages until the appropriate stopping criterion is fulfilled and transfers the pertinent data to the subsequent module. After the solution is completed, the data sets of the results of the various modules for the entire time period are combined for presentation of results in tabular and graphical forms.

The POLYMATH model equations and the associated "comments" serve as a major part of the documentation. Because these equations were actually run on the computer, this approach provides error free documentation. Only the documentation of the main program should be added to provide a complete description of the simulation program.

3. **Simulation of an Exothermic Batch Reactor**

This example is based on a problem presented by Luyben (1990). An exothermic liquid-phase reaction $A \rightarrow B \rightarrow C$ is carried out in a batch reactor. The batch reactor is sketched in Figure A1[5].

After the reactant is charged into the vessel, steam is fed into the jacket to heat the reaction mass up to the desired temperature. Thereafter, cooling water is fed into the jacket to remove the exothermic heat of reaction and to make the reactor follow a prescribed temperature-time curve, where the objective is to maximize the production of the desired product, B. The objective of the simulation of this reactor is to study the effects of operation in abnormal conditions, such as reactant overcharging, failure to control duration of the steam heating, cooling water pipe blockage and cooling water failure of various durations in various phases of the reaction.

The complete set of equations, parameters and initial values for the stage where steam of pressure *Psteam* is used to heat up the reaction mass to the temperature of *Theatmax* = 200 °F, in normal operating conditions, is shown in Table A1. Following the principles of the proposed technique, the equations and constants shown in the "Definition" column of Table A1 have been copied directly from the POLYMATH input, while the descriptions of the variables were copied from the "comments" field of the POLYMATH input. Thus, the verified model of this module that represents the heating

[5]The complete paper including Figure A1 and Table A1 can be downloaded from the ftp site: ftp://ftp.bgu.ac.il/shacham/batchsim/

stage of the batch reactor is used for documentation. Most of the model details are self explanatory, as they appear in Table A1.

The only exception is the calculation of the temperature inside the jacket, which involves solution of a differential algebraic system of equations (DAE). The "controlled integration" method of Shacham *et al.*(1996) is used for solving this DAE. A detailed explanation on the use of the controlled integration method for this particular problem can be found in this reference.

After the temperature inside the reactor has reached *Theatmax,* the steam heating is switched off and the flow of the cooling water is turned on. For the cooling stage the equation set, as it is included in the MATLAB function, is presented in Table 1. The initial values of the variables at this stage are the final values of the same variables at the heating stage.

The two separate modules, representing the two stages of operations can be run in sequence to simulate the operation of the reactor in normal operating conditions in order to verify the correctness of the mathematical models used. The next step is the development of a complete simulation program that combines the two stages of the operation and enables easy and fast parameter changes and reruns to study the effects of. abnormal operating conditions. MATLAB is used for development of the simulation program

POLYMATH input files are converted to MATLAB functions using the principles outlined in the previous section. The model equations in the MATLAB functions are the same as in the POLYMATH model (see Table 1, for example), thus they do not require

Table 1 MATLAB Function for Modeling the Cooling Stage of the Batch Reactor.

```
function dycdt=dycdt(t,y)
global HR1 HR2 rho V rhom Cpm Vm rhoj Tinj Theatmax Vjmax RAMP hi A0m Ajmax hos how Hvap
Psteam Cvs Cvw Wp
T = y(1);                                                    %  Temp. in the reactor vessel (deg. F)
Ca = y(2);                                                   % Concentration of A (mol/cu. ft.)
Cb = y(3);                                                   % Concentration of B (mol/cu. ft.)
Tm = y(4);                                                   % Temp. of the metal wall (deg. F)
Tj = y(5);                                                   % Heating/cooling jacket temp (deg. F)
Vj = y(6);                                                   % Cooling water volume in jacket (cu. ft.)
Pset = y(7);                                                 % Set point signal (psi)
k1 = 729.5488*exp(-15000/(1.99*(T+460))) ;                   % Reaction rate coeff. for A -> B (1/min)
k2 =  6567.587*exp(-20000/((T+460)*1.99)) ;                  % Reaction rate coeff. for B -> C (1/min)
Qm =  hi*A0m*(T-Tm)/60   ;                                   % Metal wall's heat flux (Btu/min)
if (Vj<Vjmax) A0 =  Vj*Ajmax/Vjmax ; else A0=Ajmax; end      % Heat transfer area for cooling (cu. ft.)
Qj = (how*A0*(Tm-Tj)/60);                                    % Heat transferred to the jacket (Btu/min)
Ptt = 3+(T-50)*12/200;                                       % Output signal from temp. transmit. (psi)
P1 =  7+2*(Pset-Ptt)  ;                                      % Controller output pressure (psi)
if (P1<3) Pc =  3; elseif (P1>15) Pc=15; else Pc=P1; end     % Controller adjusted output press. (psi)
xw1 =  (9-Pc)/6;                                             % Cooling water valve - fraction open
if (xw1<0) xw =  0; elseif (xw1>1) xw= 1; else xw=xw1;  end  % Water valve - fraction open ,adjusted
Fw0 =  (Cvw*sqrt(Wp)*8.33*xw/rhoj)  ;                        % Cooling water mass flow rate (lb/min)
if (Vj<Vjmax) dVjdt=Fw0; else dVjdt=0; end
dycdt =[(-HR1*k1*Ca-HR2*k2*Cb)/rho-Qm/(rho*V);
 -k1*Ca; k1*Ca-k2*Cb;
(Qm-Qj)/(rhom*Cpm*Vm) ;
((Fw0*(Tinj-Tj)+Qj/rhoj)/Vj);
dVjdt; RAMP];
```

further debugging or validation. The parameters were put into global variables in order to allow their modification in the main program in the course of the simulation studies, but they, also, were directly copied from the POLYMATH input. The MATLAB function representing the heating stage of the reactor is prepared similarly, by simple modification of the POLYMATH input file.

The main program for this example was prepared according to the principles mentioned earlier. This program provides means to change all the parameter values and initial values. In particular, the user can change C_{A0} (to simulate overcharging), *Theatmax* (to simulate failure to control duration of steam heating) and *Wp* (to simulate cooling pipe blockage). The user can, also, conveniently specify starting time and duration for cooling water failure. A stiff integration routine (ode15s) is used for integrating the model of the heating stage up to the time when the temperature in the reactor reaches *Theatmax*. The final values of the variables in this stage are transferred as initial values to the cooling stage, the model of which is integrated by a non-stiff integration routine (ode45). The data sets obtained from the two stages are combined. Plots of the temperature in the reactor and in the heating/cooling jacket, the concentrations of the reactant (A) and desired product (B) are displayed.

In Figure 1, the variation of the temperatures in the reactor and the jacket under normal operating conditions are displayed. The temperature in the reactor increases steadily during the steam heating. It reaches its maximum a short time after the heating is turned off and cooling is turned on and decreases gradually until the end of the batch. In Figure 2, the effect of overcharging (C_{A0} = 1.0 instead of 0.8) on the temperature profile is demonstrated. In this case the temperature keeps increasing even after the cooling is turned on. The increase is gradual at first, but after about 35 minutes runaway conditions develop. The maximal temperature reached is over 1700 °F, which is well over the specifications for safe operation of the reactor.

The POLYMATH input files for the two stages of the operation of the batch reactor and the complete MATLAB simulation program can be downloaded from the ftp site: ftp://ftp.bgu.ac.il/shacham/batchsim/

4. Conclusions

It has been shown that the effort involved in construction of dynamic simulation programs for complex processes can be reduced considerably using the proposed "modular and sequential" approach. This is achieved by using the most effective software tools for the various tasks. Using the "user friendly" mathematical software packages for the preparation of the individual models reduces to minimum the technical effort involved in coding and limits the potential errors that need debugging to the mathematical model itself. Conversion of the verified separated modules into a simulation program requires essentially the addition of a main, executive program, which takes care of the interaction with the user, sequencing and integration of the results of the various modules and reporting the results.

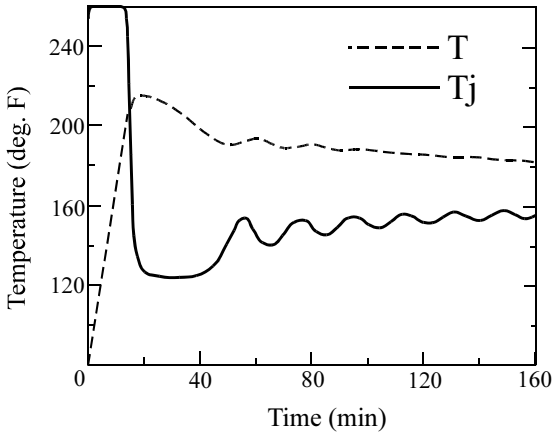

Figure 2 Temperatures in the reactor (T) and in the jacket (Tj) in normal operating conditions.

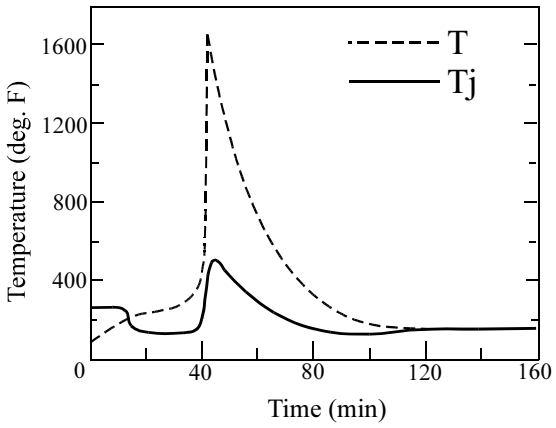

Figure 3 Temperatures in the reactor (T) and in the jacket (Tj) following reactant overcharging.

In addition to reducing the time and effort required for building the process models, the proposed approach yields modular programs, which are much better documented and easier to modify. It also ensures that the models presented in the documentation are identical to the models used in the program.

5. References

Brauner, N., M. Shacham and M. B. Cutlip, 1996, Chem. Eng. Educ., **30**, 20.

Luyben, W.L., Process Modeling Simulation and Control for Chemical Engineers, 2nd Ed., McGraw-Hill, New York.

Shacham, M., N. Brauner, and M. Pozin, 1996, Computers chem. Eng., **20**, S1329.

Shacham, M., N. Brauner and M. B. Cutlip, 2000, Computers chem. Eng, **24**, 413.

Shacham, M., N. Brauner and M. B. Cutlip, 2002, Computers chem. Eng. **26**, 547.

Srinivasan, R. and V. Venkatasubramanian, 1998, Computers chem. Eng., **22**, S961.