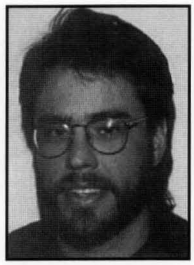
MATHEMATICA IN THE ChE CURRICULUM

JOHN R. DORGAN, J. THOMAS MCKINNON
Colorado School of Mines • Golden, CO 80401-1887

We have been active in incorporating Mathematica, a multifunctional general computer programming environment, into the chemical engineering senior year curriculum at the Colorado School of Mines (CSM). This integrated platform for symbolic, numeric, and graphical analyses has been used in the process control and reaction engineering courses. Students in these classes have had mixed but generally positive reactions toward its use. In this article we will describe our experiences and provide information that allows access to example problems posted on the Internet.

The use of Mathematica provides several advantages in teaching chemical engineering concepts—graphical accuracy in problem solutions presented in class, the ability to do more involved problems, and most importantly, discovery learning on the part of the student. Furthermore, the multimedia capabilities of Mathematica can be used to stimulate student interest in subjects that are inherently heavily mathematical in nature. Finally, we believe that the incorporation of this program into the curriculum provides a general engineering tool that can be useful to the students throughout their careers.



John R. Dorgan *is Assistant Professor of chemical engineering and petroleum refining at Colorado School of Mines. He received his BS from the University of Massachusetts at Amherst and his PhD from the University of California at Berkeley. His research interests include phase transformations in liquid crystalline polymers, self-assembly of block copolymers at surfaces, polymer encapsulation of mixed wastes, and pedagogical methods in chemical engineering.*

J. Thomas McKinnon *is Assistant Professor of chemical engineering and petroleum refining at the Colorado School of Mines. He received his BS in chemical engineering from Cornell University in 1979 and his PhD from MIT in 1989. He teaches kinetics and thermodynamics at the undergraduate and graduate levels. His research interests are in combustion chemistry and applied industrial pyrolysis.*

COMPUTER INTEGRATION AT CSM

Our department's A. Bernard Coady Computing Laboratory is an excellent facility for such activities. It consists of 25 IBM RISC-6000 Model 220 computer workstations, a Model 350 file server, a Model 560 computational server, and a three-gun overhead projection system for the instructor's workstation. The ability to integrate powerful computer packages into classroom use rests on the availability of appropriate computer equipment, and we are truly fortunate to have such facilities readily available.

Our interest in incorporating Mathematica was sparked by Professor Stanley Sandler's essay, "Technological and Societal Change and Chemical Engineering Education."[1] It pointed out that while computers have revolutionized the workplace, the biggest innovation in most college classrooms has been the introduction of the overhead projector. In the essay, Professor Stanley predicted the coming of a revolution in the delivery of education and educational methods due to changing technologies. We are in wholehearted agreement with this assessment.

It should be noted at this point that the hardware requirements for Mathematica are not severe. The program can be run on either an IBM compatible PC or on a Macintosh, provided that there is 8MB of RAM. We have used the program running on such machines coupled to a liquid crystal display device that sits atop an overhead projector. This allows quick in-class demonstrations, and the hardware necessary for this type of implementation is modest in cost. Such an arrangement works well and represents a viable alternative to dedicated computer classrooms.

MATHEMATICA AT CSM

CSM, like many other colleges, has a site license for Mathematica that allows an unlimited number of campus machines to run the program. All students at CSM are introduced to the capabilities of Mathematica in their freshman calculus sequence.[2] Additionally, the program is used in our elementary stoichiometry class for finding roots of alge-

braic equations. This early integration into the curriculum provides an excellent background for use at the senior level.

Mathematica was introduced to the process control class in the spring of 1994 and into the reaction kinetics class the following semester. Thus, the program has been in classroom use for five consecutive semesters. During this time, several example programs have been developed that are of general interest to both the academician and the practicing engineer wishing to explore the use of the Mathematica platform.

WHAT IS MATHEMATICA?

Mathematica, in our opinion, is best described as a computational environment that allows symbolic manipulation, numerical analysis, and powerful graphical representation. Examples of the symbolic manipulation capabilities include differentiation and indefinite integration of functions, solving systems of linear algebraic or ordinary differential equations, and manipulation of Laplace transforms. Numerical capabilities include fitting data with polynomials or splines, root finding of algebraic equations, and solution of systems of nonlinear ordinary differential equations through numerical integration. Graphical capabilities include 2-D and 3-D representation and contour and parametric plots, as well as animation of a series of sequential graphs.

The syntax of Mathematica is logical, but can be cumbersome; we have found this to be the biggest stumbling block for students in our classes. For example, the following command analytically solves a linear differential equation:

```
DSolve[{T'[t] - a T[t] + b == 0, T[0] == 0}, T[t], t]
```

This command, translated, says to solve the differential equation

$$\left[\frac{dT}{dt}\right] - aT + b = 0$$

subject to the following initial condition:

$$T(t = 0) = 0$$

Mathematica returns with the analytical solution in the form

```
{{T[t] → b/a - (b * E^(a * t))/a}}
```

The mixed use of parenthesis, square brackets, and curled brackets can be confusing for engineers used to programming in FORTRAN.

Fortunately, the built-in Help Menu found in Mathematica can be used to great advantage in overcoming syntactical errors. This is due to the fact that within the Help Menu a "Function Browser" can be invoked. Searching through this Browser by subject allows the user to find the command of interest along with a short description of its function. In addition, the syntax of the command is given and may be pasted directly into the program being written. This obviates the need for directly typing in command syntax.

. . . Mathematica provides several advantages in teaching chemical engineering concepts—graphical accuracy in problem solutions presented in class, the ability to do more involved problems, and most importantly, discovery learning on the part of the student.

The resources associated with developing Mathematica applications are considerable and can be used to great advantage when incorporating the program into the chemical engineering curriculum. Most significant is the existence of MathSource, an electronic bulletin board for public domain programs and example problems. The bulletin board can be accessed through the Mathematica home page using any web browser. The URL for MathSource is

http://www.wri.com.mathsource

All of the usual web searching capabilities are supported at this location.

In addition, the bulletin board may be accessed by e-mail at mathsource@wri.com. Searches for topics of interest can be performed by simply sending the e-mail message "Find *subject*" where *subject* is the field of interest (chemical engineering, perhaps). For general help in using MathSource, send the above one-line message with *intro* as the subject. In addition to the applications bulletin board, the program is supported by a very competent technical staff that can be contacted regarding questions and possible bugs at the e-mail address support@wri.com.

MATHEMATICA IN PROCESS CONTROL

Our use of Mathematica in the process control class centers on the subjects introduced in the early part of the semester. Roughly the first one-third of the course deals only with process dynamics; formulation of state models and calculation of uncontrolled response to input changes are addressed during this time period. Experience shows that the program can be used to great advantage in demonstrating the difference between nonlinear process models and the corresponding linear models that form the basis of modern control theory.

A relevant example of this type of use can be taken from a homework assignment during the spring 1995 semester in which the students were asked to compare the behavior of draining liquid tanks of varying geometry (cylindrical, conical, and spherical) for the case in which the outflow is proportional to the square root of the liquid level. The process model for the cylindrical tank appears in most textbooks, the conical tank model is developed in lecture, and the spherical tank model is assigned as an earlier, independent homework assignment. A simple representation of the tank geometries and the resulting process models is given in

Figure 1 (next page). In these models, F_i is the volumetric inlet flow, F_o is the outlet flow, β represents the discharge coefficient, h is the liquid level height in the tank, and h_s is the initial steady-state level.

The objective of the Mathematica assignment is to compare the full nonlinear process behavior to the linearized approximation and to draw conclusions about the severity of the linearization approximations.

For this particular problem, an example solution for the cylindrical tank is provided as part of the problem statement. The heart of the assignment consists of using only three Mathematica commands: DSolve[] for solving the linearized equations, NDSolve[] for numerically solving the nonlinear equations, and Plot[] for comparing the solutions. Figure 2 is part of an actual program listing for a solution submitted by one of the students.

From this straightforward application, students learn that linear approximations may either under- or overestimate the actual system response (in this case, the time required to empty the given tank). In addition, they experience the phenomena that some systems deviate more quickly from their linear approximations than do others and that this depends on the strength of the nonlinearities present in the process. The type of discovery learning which the program allows is thought to be advantageous toward student learning and content retention.[3]

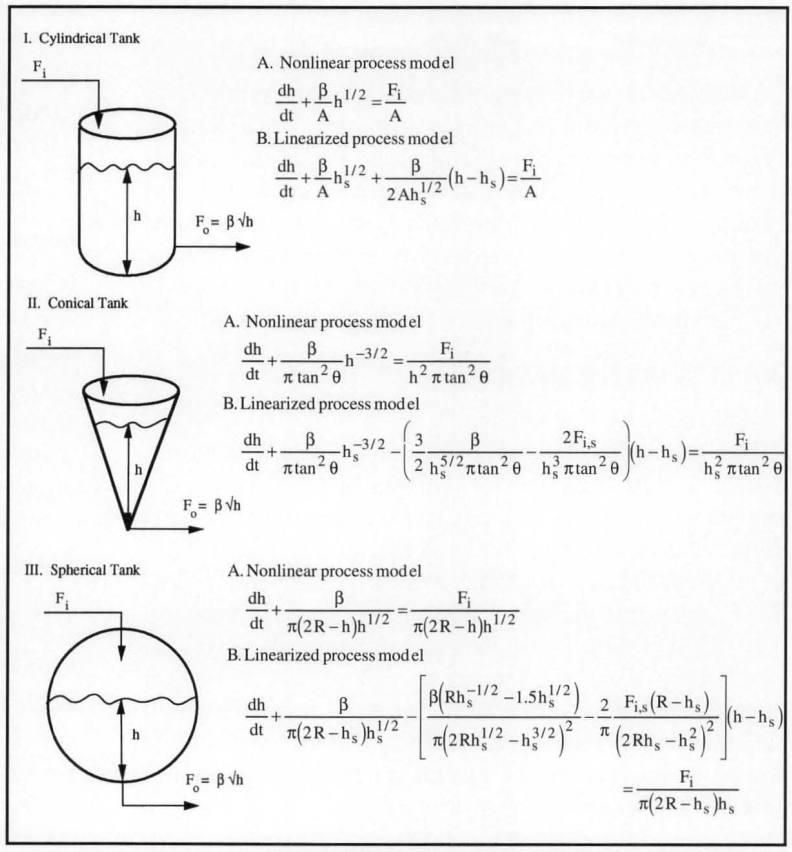


Figure 1

Another example from the process control class involved the use of the symbolic capabilities for manipulation of Laplace transforms. The assignment in this case was to write a general purpose simulator for the response of any linear second-order system and to test the response of the system toward different types of forcing functions. This assignment was given in the fall 1994 and spring 1995 semesters.

As a preface to this assignment, the students are given an example program that constitutes a general purpose simulator for a first-order system. Recall that the standard form for a linear first-order system is

$$\tau_p\left(\frac{dy}{dt}\right)+y=K_pf(t)$$

where τ_p is the process time constant, K_p is the static gain, and f(t) is the forcing function. The resulting transfer function is

$$G(s)=\frac{K_p}{\tau_ps+1}$$

where s represents the independent variable in Laplace space.

The demonstration program allows the user to input his or her own function for f(t). This user-defined forcing function is then transformed, multiplied by the transfer function, and inverted to give the time domain response of the system to the given forcing function (which is defined as a plot versus time). The students are asked to run this program for a variety of different forcing functions: step inputs, ramps, and sine waves. In addition, the students run the program for different values of the system parameters K_p and τ_p, thus experiencing the meaning of these quantities directly. This type of exploration learning can be highly effective.

Based on the example given, the students are asked to rewrite the program to be applicable for second-order systems. The standard form of the transfer function is this case is

$$G_2(s)=\frac{K_p}{\tau^2s^2+2\tau\zeta s+1}$$

where ζ is the damping coefficient, while τ_p and K_p and f(t) have the same meanings as above. For second-order systems, when the damping coefficient is less than unity, oscillatory behavior is observed.

While the students' solutions generally mimic the example program, we wish to emphasize that we are not trying to teach Mathematica programming, per se, but rather are using the capabilities of the program in order to demonstrate concepts in chemical engineering. Again, the students are asked to test their simulator with a variety of forcing func-

```
(* PART 2 - THE CONICAL TANK*)
Clear [Fi, A, hs, beta, theta, Fis]

(* Define a function which is the linear O.D.E.*)
conf[t_] := conh'[t] - Fis hs^-2 + beta/(N[Pi] N[(Tan [theta])^2]) hs^(-3/2)
            -hs^-2 (Fi-Fis)-(3 beta/(2 N[Pi] N[(Tan [theta])^2]) hs^(-5/2) -
            2 Fis hs^-3) (conh[t]-hs)

(* Solve the linear O.D.E with the initial tank height. *)
sol3 = DSolve[{conf[t]==0, conh[0]==hs},conh[t],t];

(* Define a function which can be plotted. *)
conhsolve[t_] := conh[t] /. sol3[[1]]

(* Define a function which is the non-linear O.D.E. *)
connlf[t_] := connlh'[t] + beta/(N[Pi] N[(Tan [theta])^2]) connlh[t]^(-3/2) -
              Fi connlh[t]^(-2)

(* Numerical solution requires parameter values. *)
Fi = 0;
Fis = 0;
beta =1;
hs = 5;
theta = N[Pi]/6;
(* Solve the non-linear O.D.E. with the initial tank height set to hs m. *)
sol4 = NDSolve [{connlf[t]==0,connlh[0]==hs},connlh[t],{t,0,23}];

(* Convert the numerical solution to a function which may be plotted. *)
connlhsolve[t_] := connlh[t] /. sol4

(* Plot the two functions for comparitive purposes. *)
Plot[{conhsolve[t],connlhsolve[t]},{t,0,23}];
```

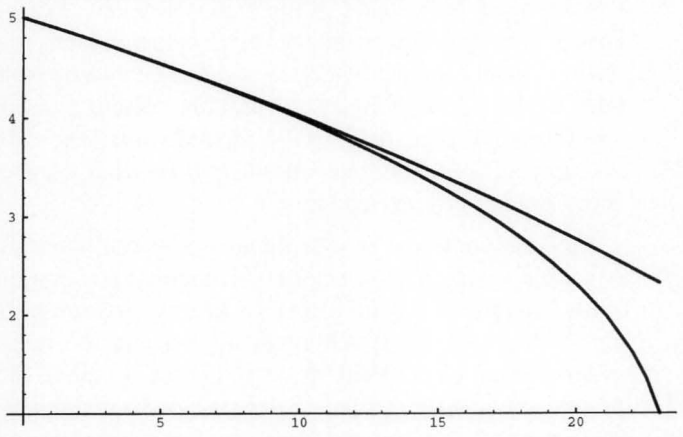


Figure 2

tions for different parameter values. The responses of a second-order system to a unit step input for both overdamped ($\zeta > 1$) and underdamped ($\zeta < 1$) cases are explored.

The ability of students to explore system dynamics using these approaches is important in making the connection between process models and real-world behavior. Later in the semester, the excellent process simulation capabilities of PICLES are used as part of the class.[4] Thorough preparation in the fundamentals means the students are better prepared for logically connecting the mathematics of process modeling with observable system dynamics.

MATHEMATICA IN REACTION KINETICS

We introduced Mathematica in our senior-level kinetics and reactor design class in the fall 1994 semester. Solving differential equations that result from the reactor design equations is a large part of this class, along with root finding and plotting of results. In previous semesters, we used Polymath for these tasks.[5] This program has the advantage of being very easy to use; we found that the students needed little more than a ten- or fifteen-minute introduction to the program in lecture to use it effectively.

But we found that the ODE solving routines in Polymath are not as robust as those in Mathematica, particularly with regard to the stiff differential equations that routinely arise in chemical kinetics problems. Furthermore, the programming environment allows more complex problems to be constructed; the results of one calculation (*e.g.,* a numerical integration) can be fed directly into the subsequent one (*e.g.,* finding a root). The price to pay for this increased usefulness is, of course, a much steeper learning curve.

We describe below some of the homework problems for which Mathematica was used. It should also be noted that our textbook, *Elements of Chemical Reaction Engineering*[6] also lists a few Mathematica examples.

◑ **1. Chemical equilibrium** • The first week of our class is a review of chemical reaction equilibrium. The FindRoot[] function in Mathematica is used to evaluate the roots of high-order polynomials to solve for equilibrium conversion. The programming environment in Mathematica allows us to easily explore the effect of temperature changes on equilibrium composition. One difficulty for new users is in discovering which, apparently equivalent, Mathematica function is appropriate for a given problem. In the case of reaction equilibrium, both NSolve[] and FindRoot[] should work, as described by Wolfram.[7] The former function finds all the roots to a polynomial, while the later finds a root to any function over a specified interval. It is only by trial and error that the user discovers that NSolve[] rarely works for these problems and that FindRoot[] is the more useful approach. This highlights the importance of the instructor directing the students in lecture before they start out on a problem.

◑ **2. Comparison of ODE solution methods** • Since solving ordinary differential equations (ODEs) is such a large part of the class, we spend one entire homework assignment exploring different solution methods. We start with a relatively simple first-order batch reactor problem with the reaction steps

$$A \rightarrow B \rightarrow C$$

The solution for [B] as function of time can be found analytically by hand using the integrating factor method. Next, we move on to a more complicated reaction network with parallel reactions and reversibility:

This system of ODEs has an analytical solution, but it would be rather difficult to solve by hand (and solving difficult ODEs is not the objective of our course). For this job, we turn to the NSolve[] function in Mathematica. The raw output of NSolve[] for this

system is exceedingly cumbersome, but with the use of the Simplify[] function, it becomes at least somewhat manageable.

We then use this same reaction network to explore numerical solutions to ODEs. The existence of the analytical solution allows the students to examine the errors that can arise in numerics. They first do a solution by implementing Euler integration using a spreadsheet. In the lecture discussion of Euler integration, we also show how Runge-Kutta methods improve the Euler integration results. The integration stepsize can be varied and the resulting time profiles are compared to the analytical solution. Finally, we solve the same system numerically, using the NDSolve[] function. Here there is no discernible difference between the analytical and the numerical result.

◑ **3. Multiple reaction with phase change** • Our most ambitious Mathematica homework problem was to explore the effect of temperature on a system that includes both multiple reactions (both series and parallel) and phase change. We postulated that a new methanol synthesis catalyst had been discovered that has reasonable activity around 400K and 50atm. The reaction considered are

$$CO + 2H_2 \Leftrightarrow CH_3OH \qquad \text{Rxn. 1}$$

$$CO + H_2O \Leftrightarrow CO_2 + H_2 \qquad \text{Rxn. 2}$$

$$CH_3OH \Rightarrow CH_2O + H_2 \qquad \text{Rxn. 3}$$

Arrhenius parameters are provided for all the reactions. The students find the reverse rates of Rxns. 1 and 2 from the reaction thermodynamics. The feed is specified to be a mixture of CO, CO_2, H_2, steam, and inert (N_2). The volume of the reactor and the total molar feed rate are specified. The vapor pressure of methanol is computed from the Harlacher equation. Although the deviations from ideality are high at these conditions, we still use the ideal gas law for the purpose of simplification. The objective is to find the temperature at which the methanol production rate is maximized. Increasing the temperature obviously favors faster kinetics, but this advantage is rapidly diminished by the approach to equilibrium of Rxn. 1 and the more rapid product decomposition by Rxn. 3. Furthermore, the condensation of methanol at low temperatures pulls the equilibrium conversion of Rxn. 1 to the right.

A multistep solution process is required for this problem and can be performed in the Mathematica environment. First, the students must numerically solve the six coupled ODEs (one for each species) for the gas-phase problem over the whole reactor volume. Next, the volume V_{crit}, at which methanol first begins to condense, must be identified from the methanol mole fraction and the solution to the Harlacher equation. After condensation ($V > V_{crit}$), the relationships between moles and mole fractions are different, and a second numerical integration must be carried out from V_{crit} to V_{final}. The process is repeated at different temperatures until the optimum temperature is identified.

◑ **4. Unsteady flow problems** • Unsteady flow problems are ideal as examples where all the terms of the mole balances must be considered. One of our favorite problems of this nature is the unsteady-state startup of three CSTRs in series (found in Fogler,[6] Problem 4-32). The students write three coupled time-dependent ODEs for the conversions in each stirred tank reactor and solve them using NDSolve[]. Figure 3 gives a listing of the Mathematica commands needed for this problem.

IMPLEMENTATION PROBLEMS

While we are in general very pleased with Mathematica and plan to continue using it, we would be remiss in an article of this nature if we failed to point out some of the difficulties we encountered. Probably the greatest source of frustration on the part of the students is the inscrutable error messages that are spawned by Mathematica. The incorrect definition of a variable within NDSolve[] generates many lines of error messages giving no real hint of the problem, while at the same time (and more troublesome) holding out several red herrings leading to misplaced debugging efforts. The second greatest source of frustration is that Mathematica holds values of variables until they are explicitly cleared. One solution to this problem is to include the function Clear[] at the beginning of every Mathematica problem where every variable used is included in the square brackets. When a change is made to the notebook, the problem is rerun from the top (menu commands: Action, Evaluate, Evaluate Notebook). This action removes all previous values from the variables and resets them to the desired value. The value of carrying out this step, although somewhat cumbersome, cannot be stressed enough.

The solution to the first problem is more difficult. We generally assign it to a lack of practice using debugging skills brought on by the increased use of software packages by our students. Although all students do take a programming class (FORTRAN is taught at CSM), they practice very little in subsequent classes. In many ways, Mathematica can be considered to be a programming language, thus the same debugging strategies can be used. We find that it is necessary to include a discussion of debugging methods in lectures when discussing problem solutions using Mathematica.

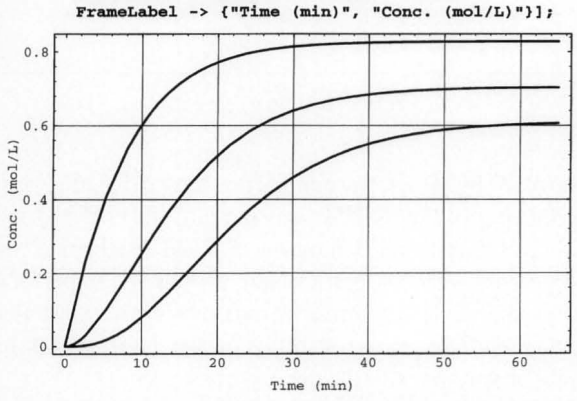
CONCLUSIONS

Mathematica is a very powerful computational tool that can be used successfully in the undergraduate chemical engineering curriculum. We have used the program for several semesters and in different classes at the senior level. Based on our experiences and recent trends in the development of new media for the delivery of education, we expect the use of programs that can combine symbolic, numeric, and graphical capabilities to increase.

In the process control class, Mathematica has been implemented in order to elucidate the difference between nonlinear system response and the linearized approximation of this behavior. Additionally, the symbolic manipulation capabilities associated with the Laplace transform have been exploited in order to allow discovery learning on the part of the students.

The use of Mathematica in the reaction engineering class has greatly expanded our horizons regarding the types of problems we can address. The robust ODE solver in Mathematica frees us from the constraint of artificially selecting reaction parameters. The programming environment within Mathematica allows results of one calculation to be automatically cascaded into the next. Furthermore, these assignments are somewhat open ended, and there are few software limits placed on the more ambitious students who want to develop elegant solutions. Finally, the advanced plotting capabilities of the program allow visualization of the results for different cases.

```
(*  CR418       HW 7.1  Fogler Problem 4-32              *)
(*  Unsteady Flow Problem - Startup of 3 Series CSTRs  *)
(*  Rxn: A + B --> C, elementary.  Equimolar feed rate  *)
(*  Find the time required for the exit of the third   *)
(*  reactor to reach 99% of its steady-state value     *)


(*  Important!  Start every problem by clearing        *)
(*  previous values                                    *)
Clear[dna1dt, dna2dt, dna3dt, fao, vdot, ca1, ca2, ca3,
      na1, na2, na3, na1r, na2r, na3r, v, fao, vdot,
      tfinal, k]

(* Mole balances for species A in each reactor.        *)
(* Writing  these equations in the form:               *)
(* "accum. = in - out + generation" aids debugging     *)
dna1dt = fao - vdot ca1 - k ca1^2 v;
dna2dt = vdot ca1 - vdot ca2 - k ca2^2 v;
dna3dt = vdot ca2 - vdot ca3 - k ca3^2 v;

(* Auxiliary equations.  Relate all the variables in   *)
(* problem to the dependent variables of the ODEs, and *)
(* assign values to the constants.  "ca3ss" is the     *)
(* steady-state value solved in a separate notbook.    *)
ca1 = na1[t]/v; ca2 = na2[t]/v; ca3 = na3[t]/v;
fao = 20;  v = 200; vdot = 20; tfinal = 65; k = 0.025;
ca3ss = 0.609;

(* NDSolve numerically evaluates the coupled ODEs      *)
(* given the initial conditions.                       *)
sol = NDSolve[{ na1'[t] == dna1dt,
                na2'[t] == dna2dt,
                na3'[t] == dna3dt,
                na1[0] == 0,
                na2[0] == 0,
                na3[0] == 0},
                {na1, na2, na3},
                {t, 0, tfinal}];

(* The direct output of NDSolve is a List.  The        *)
(* Replace operator (/.) extracts the desired function *)
(* from the List.                                      *)
na1r = na1 /. sol[[1]];  na2r = na2 /. sol[[1]];
na3r = na3 /. sol[[1]];

Plot[{na1r[t]/v, na2r[t]/v, na3r[t]/v}, {t, 0, tfinal},
      GridLines -> Automatic, Frame -> True,
      FrameLabel -> {"Time (min)", "Conc. (mol/L)"}];
```




```
(* Find the point at which the exit conc. of Reactor 3 is *
(* 99% of its steady-state value.                         *
FindRoot[na3r[t]/v == 0.99 ca3ss, {t, 0}]

{t -> 59.6826}
```

Figure 3.

Beyond the benefits of enhanced education in chemical engineering, we believe that the type of computer environment that Mathematica represents is here to stay. As a result, the exposure that students receive when using the program introduces them to a computational tool that should prove valuable to them in the future regardless of the specifics of their career paths.

REFERENCES

1. Sandler, S.I., "Technological and Societal Change and Chemical Engineering Education," Phillips Petroleum Company Lecture (1993)
2. Hagin, F.G., and J.K. Cohen, *Calculus Explorations with Mathematica,* Prentice-Hall, Englewood Cliffs, NJ (1995)
3. Wankat, P.C., and F.S. Oreovicz, *Teaching Engineering,* McGraw-Hill, New York, NY (1993)
4. Cooper, D.J., "PICLES," *Chem. Eng. Ed.,* **27**(4), 176 (1993) and Cooper, D.G., "PICLES: The Process Identification and Control Laboratory Experiment Simulator," *CACHE News,* 6-12 (1993)
5. Polymath, developed by M. Cutlip and M. Sacham. Available from CACHE Corporation, PO Box 7939, Austin, TX 78713
6. Fogler, H.S., *Elements of Chemical Reaction Engineering,* 2nd ed., Prentice Hall (1992)
7. Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer,* 2nd ed, Addison Wesley (1991)

APPENDIX

Obtaining Examples Through e-mail

As stated in the body of this article, the MathSource bulletin board may be accessed through the Internet at the e-mail address

mathsource@wri.com

To access the process control examples described in this article as well as other control examples, send the e-mail message

Find *Control*

The MathSource server will send a listing of all examples containing these keywords. In the returned file, each of the listings will have an identifying number (as of this writing, the specific numbers for our own examples are not known).

One process control application that is already available is called *control.m.* This public domain package allows students to easily generate Bode and Nyquist diagrams, root locus plots, and most other types of plots used in a control course.

To obtain copies of the examples of interest, simply send MathSource another e-mail message asking it to send the appropriate identifier. As an example, if the message

Send 0202-554-0011

is sent to the MathSource address, an ASCI text file of the MathSource Technical Report will be returned to the requester through e-mail. Further details of the downloading procedure may be found in this document. ❐