The object of this column is to enhance our readers' collections of interesting and novel problems in chemical engineering. Problems of the type that can be used to motivate the student by presenting a particular principle in class, or in a new light, or that can be assigned as a novel home problem, are requested, as well as those that are more traditional in nature and that elucidate difficult concepts. Manuscripts should not exceed 14 double-spaced pages and should be accompanied by the originals of any figures or photographs. Please submit them to Professor James O. Wilkes (e-mail: wilkes@umich.edu), Chemical Engineering Department, University of Michigan, Ann Arbor, MI 48109-2136.

CAN I TRUST THIS SOFTWARE PACKAGE?

an Exercise in Validation of Computational Results

MORDECHAI SHACHAM
Ben-Gurion University of the Negev • Beer-Sheva 84105, Israel
NEIMA BRAUNER
Tel-Aviv University • Tel-Aviv 69978, Israel
W. ROBERT ASHURST
Auburn University • Auburn, Alabama 36849
MICHAEL B. CUTLIP
University of Connecticut • Storrs, CT 06269

Mathematical software packages such as Polymath, MATLAB and Mathcad are widely used for engineering problem solving. The application of several packages to typical chemical engineering problems has been demonstrated by Cutlip, et al.[1] The main characteristic of these packages is that they provide a "problem solving environment (PSE)"[2] rather than just scientific subroutine libraries callable from programming languages that were popular a few years ago. For routine use of the software packages in problem solving, there is no need to be an expert in either numerical methods or programming. In most cases, it is sufficient to specify the mathematical model of the problem using the syntax required by the package. The technical details of the solution are carried out by the package. There are cases, however, where the solution is obviously incorrect and/or the program stops with an error message.

The failure of a software package to reach a correct solution is commonly due to the presence of errors in the mathematical model, or the selection of an inappropriate solution algorithm and/or inappropriate algorithm parameters (such as initial estimates, error tolerances, etc). Validation of the mathematical models has been discussed in detail by Brauner, et al.[3] In this paper, we present an exercise that can be used

Mordechai Shacham *is the Benjamin H. Swig professor of chemical processes and former chair at Ben-Gurion University of the Negev in Israel. He received his B.Sc. and D.Sc. degrees from the Technion, Israel Institute of Technology. His research interests include analysis, modeling and regression of data, applied numerical methods and prediction, and consistency analysis of physical properties. He is past president and an honorary fellow of the Israel Institute of Chemical Engineers, and the recipient of the 2000 CACHE Award for excellence in computing in chemical engineering education.*

Neima Brauner *is a professor at Tel-Aviv University in Tel Aviv, Israel. She received her B.Sc. and M.Sc. in chemical engineering from the Technion Institute of Technology in Haifa, Israel, and her Ph.D. in mechanical engineering from the Tel-Aviv University. Her research interests include hydrodynamics and transport phenomena in two-phase flow systems and development of interactive statistical and numerical methods for data analysis in process analysis and design. She is an editor for Reviews in Chemical Engineering, and associate editor of Heat Transfer Engineering.*

W. Robert Ashurst *is currently an assistant professor of chemical engineering at Auburn University. His research focuses on design of molecular precursors for advanced monolayer films, tribology at the micro- and nano-scale, and the influence of surface chemical treatments on micro- and nano-scale devices. He completed his Ph.D. in chemical engineering at the University of California at Berkeley in 2003 under the support of a National Science Foundation Graduate Fellowship. Dr. Ashurst is an active member of AIChE.*

Michael B. Cutlip *has spent his entire academic career at the University of Connecticut, where he served as department head of chemical engineering and director of the University's honors program. He is a former Chair and National Program Chair for the Chemical Engineering Division, and he co-chaired the ASEE Summer School for chemical engineering faculty in 2002. He is also co-author of the POLYMATH software package.*

in educating students to identify difficulties associated with an algorithm for the solution of a problem and help them to select appropriate parameters for the algorithm.

This exercise involves numerical solution for a system of ordinary differential equations (ODE's). It can be used at two levels.

1. *The first is in courses where students routinely use numerical software for solving ODE's (*i.e., *Reaction Engineering, Process Control and Process Simulation). In such courses, this part of the exercise can be used to demonstrate that computer solutions can be inaccurate, or even completely incorrect, and therefore it is always necessary to validate a numerical solution.*
2. *At the more advanced level, the second part of the exercise can be used in courses related to mathematical modeling and numerical analysis. In such courses, this part of the exercise provides training and demonstration of advanced topics, such as error estimation and step size control, error propagation, and stiffness of ODE's.*

In this work, the mathematical software packages Polymath 6.1[1] and MATLAB[2] are used for the demonstration, but other packages may be used in a similar manner.

ANALYZING THE BEHAVIOR OF THE RUNGE-KUTTA ALGORITHM FOR AN "INTERACTING TANKS" SIMULATION PROBLEM

The problem considered in this exercise involves the dynamic simulation of a system of three interacting tanks, as depicted in Figure 1. The model equations (as were provided Ashurst and Edwards[4]) that can be derived from mass balances on each tank are

$$q_1 - q_4 = \frac{\pi}{4}\left[D_2 + h_1\left(\frac{D_1 - D_2}{H_1}\right)\right]^2 \frac{dh_1}{dt} \qquad (1)$$

$$q_2 + q_4 - q_5 = \pi e^{2h_2} \frac{dh_2}{dt} \qquad (2)$$

$$q_5 + q_3 - q_6 = h_3^2 \frac{dh_3}{dt} \qquad (3)$$

where q_j represents the in/out volumetric flow-rate (m^3/s), H_i the tank height (m), h_i the liquid height (m), $D_{1,2}$ the upper/lower diameter ratio of tank 1 (m), and *t* is the time (s). The volumetric flow-rates out of the tanks, through a drainage area, Ao (m^2), are given by

$$q_4 = Ao\sqrt{2g(h_1 - h_2)} \text{ if } h_1 > h_2$$
$$q_4 = -Ao\sqrt{2g(h_2 - h_1)} \text{ otherwise} \qquad (4)$$
$$q_5 = Ao\sqrt{2g(h_2 - h_3)} \text{ if } h_2 > h_3$$
$$q_5 = -Ao\sqrt{2g(h_3 - h_2)} \text{ otherwise} \qquad (5)$$
$$q_6 = Ao\sqrt{2gh_3} \qquad (6)$$

The response of the system to a step change in the inlet flow-rates is to be investigated. The model equations and the numerical data pertinent for a particular case, as they have been entered into the Polymath 6.1 software package, are shown in Table 1. Note that the equations and comments in the Polymath input provide a complete documentation of the problem (in Polymath the comments are indicated by the sign: #).

TABLE 1
Polymath Model for the "Train of Interactive Tanks" Problem

No.	Equations and # Comments
1	# Mass balances on the tanks
2	d(h1)/d(t) = If h1 >= H1 Then (0) Else (q1 - q4) / ((pi / 4) * ((D2 + ((D1 - D2) / H1) * h1) ^ 2)) # Liquid level in Tank 1 (m)
3	d(h2)/d(t) = If h2 >= H2 Then (0) Else (q2 + q4 - q5) / (pi * exp(2 * h2)) # Liquid level in Tank 2 (m)
4	d(h3)/d(t) = If h3 >= H3 Then (0) Else (q5 + q3 - q6) / (h3 ^ 2) # Liquid level in Tank 3
5	# Outlet flow rates are functions of potential flow theory.
6	q4 = If h1 < h2 Then - Ao * sqrt(2 * g * (h2 - h1)) Else Ao * sqrt(2 * g * (h1 - h2)) # Volumetric flow rate out of Tank 1 (m^3/s)
7	q5 = If h2 < h3 Then - Ao * sqrt(2 * g * (h3 - h2)) Else Ao * sqrt(2 * g * (h2 - h3)) # Volumetric flow rate out of Tank 2 (m^3/s)
8	q6 = Ao * sqrt((2 * g * h3)) # Volumetric flow rate out of Tank 3 (m^3/s)
9	# Input variables
10	q1 = .05 # Volumetric flow rate into Tank 1(m^3/s)
11	q2 = .05 # Volumetric flow rate into Tank 2(m^3/s)
12	q3 = If t <= 50 Then .05 Else .05 + .05 * .1 # Volumetric flow rate into Tank 3(m^3/s)
13	# Constants
14	H1 = 4 # Height of Tank 1(m)
15	H2 = 2.5 # Height of Tank 2 (m)
16	H3 = 2.5 # Height of Tank 3 (m)
17	D1 = 4 # Upper diameter of Tank 1(m)
18	D2 = 1 # Lower diameter of Tank 2 (m)
19	Ao = pi / 100 # Area of drainage orifice at bottom of tanks (m^2)
20	pi = 3.141592654
21	g = 9.81
22	# Initial Conditions
23	h1(0) = 1.809 # Liquid level in Tank 1 at steady state(m)
24	h2(0) = 1.68 # Liquid level in Tank 2 at steady state(m)
25	h3(0) = 1.163 # Liquid level in Tank 3 at steady state(m)
26	t(0) = 0
27	t(f) = 2400.

[1] *POLYMATH is a product of Polymath Software (<http://www.polymath-software.com>)*
[2] *MATLAB is a product of The MathWorks, Inc. (<http://www.mathworks.com>)*

The system is started at steady-state values of h_1, h_2 and h_3 (see lines 23 through 25), where the inlet flow rates to the different tanks are $q_1 = q_2 = q_3 = 0.05$ m^3/s. At t = 50 s, a step change of +10% is introduced into q_3. The system is simulated up to the final time of $t_f = 3600$ s.

The integration of the model equations using the RKF45 algorithm of Polymath with the default parameters yields the error message "*Error: sqrt of negative number is not allowed, sqrt(2*g*h3),*" which indicates that the height of the liquid level in the 3rd tank (h_3) has somehow reached a negative value during the numerical integration. A quick check can be made on the dynamics by reducing the integration final time to 2400 s. As there is no abnormal behavior or unexpected trends during this numerical integration, as shown in Table 2 and Figure 2, the assignment is to check and verify the error in the integration up to the time of 3600s. The options that need to be investigated for this problem are:

1. *There are possible errors in the implementation of the Runge-Kutta algorithm.*
2. *The default truncation error tolerance of the program is too large for this problem, and the negative value of h_3 is a result of large truncation error and propagated error in some of the variables.*
3. *The system of equations is stiff. The explicit RKF45 may exhibit oscillatory behavior if the integration step size is greater than that dictated by the stability limit.*

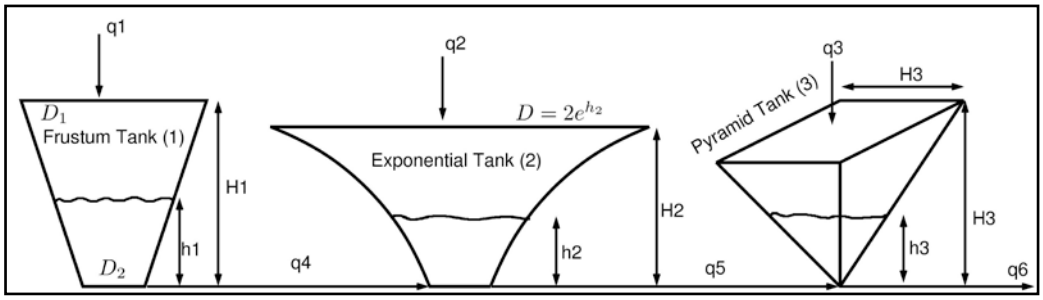

Figure 1. *A Train of three interactive tanks.*[4]

TABLE 2
Change of the Variable Values after Introduction of a 10% Step Change in Inlet Flow Rate to Tank 3

Variable	Initial value	Minimal value	Maximal value	Final value
t	0	0	2400	2400
h_1	1.807	1.807	1.854328	1.854328
h_2	1.68	1.679873	1.725494	1.725494
h_3	1.162	1.162	1.221394	1.221394
q_4	0.0495908	0.0495908	0.0499622	0.0499475
q_5	0.1001531	0.0971087	0.1001531	0.0988002
q_6	0.1500039	0.1500039	0.1537897	0.1537897
dh_1/dt	9.39E-05	8.66E-06	9.39E-05	1.17E-05
dh_2/dt	-6.22E-06	-6.22E-06	3.06E-05	1.16E-05
dh_3/dt	1.11E-04	-5.39E-07	2.71E-04	7.08E-06

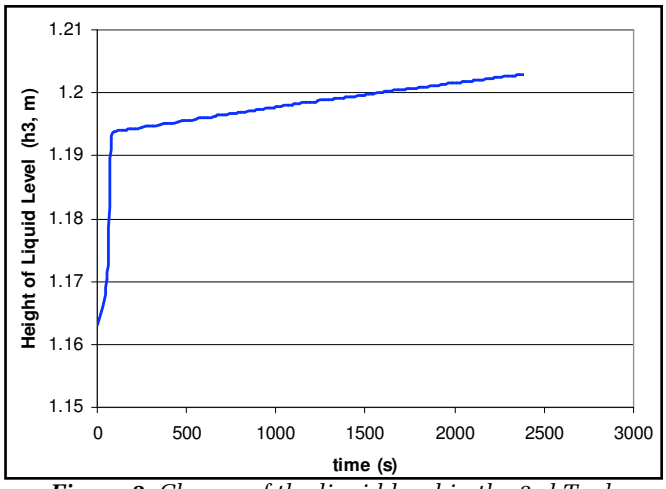

Figure 2. *Change of the liquid level in the 3rd Tank after introduction of a 10% step change in its inlet flow rate (Polymath Solution).*

RECOMMENDED STEPS AND ADDITIONAL INFORMATION FOR SOLUTION

1. Most software packages include some variation of the Runge-Kutta algorithm – with truncation error estimation and step size control – as one of the ODE solver tools. Using the same or a similar algorithm in a different software package can provide additional information regarding the source of the error.

2. The appropriateness of the truncation error tolerance can be tested by rerunning the problem using various error tolerances and comparing the variable values obtained at a specified time (say t_f). It is preferable to check the values at the final time, as at this point the error propagated throughout the integration interval is considered.

3. The stiffness of the problem can be determined by defining the f vector as

$$f = \begin{bmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \\ \frac{dh_3}{dt} \end{bmatrix} \quad (7)$$

where dh_1/dt, dh_2/dt, and dh_3/dt are given by Eq's. (1-3), respectively. The stiffness of the system can be assessed by calculating the stiffness ratio of the ∂f/∂h (Jacobian) matrix. The stiffness ratio (SR) is defined by (see, for example, p. 245 in Rice and Do[5])

$$SR = \frac{\max|\lambda_i|}{\min|\lambda_i|} \tag{8}$$

where max $|\lambda_i|$ is a negative eigenvalue of the ∂f/∂h matrix with maximal absolute value, and min $|\lambda_i|$ is a negative eigenvalue of the ∂f/∂h matrix with minimal absolute value. If SR>1000, the system of ODE's is considered a stiff system. If the system is stiff, using the appropriate algorithms (*i.e.*, stiff or stiffbs in Polymath) should prevent obtaining the negative h_3 value during integration.

SOLVING THE PROBLEM WITH A DIFFERENT SOFTWARE PACKAGE

Polymath 6.1 can be used to convert the model of the problem into a MATLAB function. The Polymath generated function is shown in lines 21 through 42 of Table 3 (next page). Note that Polymath reorders the equations and changes the syntax of the model according to the requirements of MATLAB. Some manual editing of the Polymath generated function was done in order to enable more concise presentation (the comments were put in the same line with the commands and the if-else statements were put in a single line).

The "main program" that runs the Polymath generated function (shown in lines 1 through 19 of Table 3) is available as a template in the "Help" section of Polymath. Only the function name has to be added (see line 1) and the initial values of the variables had to be copied from the Polymath generated model (see lines 3 and 4). The *ode45* library function of MATLAB can be used to solve the problem. This function is based on essentially the same algorithm as the RKF45 routine of Polymath.

MATLAB solution of the program in Table 3 yields a similar solution to that obtained with Polymath, except that that the liquid level in the 3rd tank (h_3) exhibits oscillatory behavior (see Figure 3). These oscillations indicate that there may be difficulties with the MATLAB solution as well, and suggest that the error tolerances should be examined.

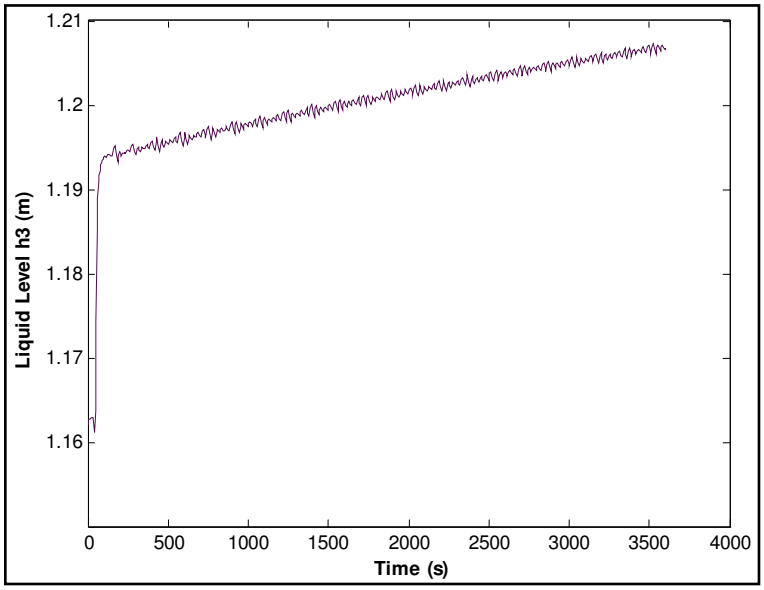

Figure 3. *Change of the liquid level in the 3rd tank after introducing a 10% step change in its inlet flow rate (*ode45 *Algorithm,* RelTol = 10^{-3}*).*

CHECKING THE ROLE OF THE TRUNCATION ERROR TOLERANCE

In the Polymath RKF45 algorithm, the integration step size is controlled by two parameters: the relative error tolerance (*RelTol*) and the minimum number of reporting points (RP). The default values of these parameters can be found under the Polymath icon of "Setup preferences and numerical parameters," which is under Ordinary Differential Equations and RKF45: *RelTol* = 10^{-6} and *RP* = 100. The RKF45 program includes an adaptive step size control algorithm that monitors the estimate of the integration error and adjusts the step size of the integration in order to keep the error below the specified threshold value. The step size is increased only as long as at least RP full steps are carried out inside the integration interval. Therefore, a solution of a lower accuracy can be obtained if the error tolerance is increased, and, at the same time, the number of reporting points is reduced. For instance, if the error tolerance is increased to *RelErr* = 10^{-4} and the number of reporting points is reduced to *RP* = 30, difficulties are encountered earlier. At some time t < 2400, the error message ("negative square root") is issued by the program. With adjustment of *RelErr* = 10^{-10} while holding *RP* = 30, however, no error message is issued when the integration is carried out up to t_f = 3600. This implies that a correct solution has been obtained. In the MATLAB *ode45* function, the integration step size is adjusted so as to keep the estimated error below the value of ε_i, which is a function of the specified relative error tolerance, *RelTol*, and the specified absolute error tolerance, *AbsTol*.

$$\varepsilon_i = \max(|h_i| \mathrm{RelTol}, \mathrm{AbsTol}) \tag{9}$$

The default values of the *ode45* function for the absolute error tolerance and for the relative error tolerance are *AbsTol* = 10^{-6} and *RelTol* = 10^{-3}, respectively. Since the values of h_1, h_2, and h_3 are of the order of magnitude of 1 (one), the *AbsTol* is inactive and the ε_i values are of the order of 10^{-3}. With the default error tolerances in MATLAB, the oscillatory solution (shown in Figure 3) is obtained. Reducing the rela-

TABLE 3
MATLAB Program for the "Train of Interactive Tanks" Exercise

No.	Commands and % Comments
1	function TankTrain
2	clear, clc, format short g, format compact
3	tspan = [0 3600]; % Range for the independent variable
4	y0 = [1.807; 1.68; 1.162]; % Initial values for the dependent variables
5	disp(' Variable values at the initial point ');
6	disp([' t = ' num2str(tspan(1))]);
7	disp(' y dy/dt ');
8	disp([y0 ODEfun(tspan(1),y0)]);
9	[t,y]=ode45(@ODEfun,tspan,y0);
10	for i=1:size(y,2)
11	disp([' Solution for dependent variable y' int2str(i)]);
12	disp([' t y' int2str(i)]);
13	disp([t y(:,i)]);
14	plot(t,y(:,i));
15	title([' Plot of dependent variable y' int2str(i)]);
16	xlabel(' Independent variable (t)');
17	ylabel([' Dependent variable y' int2str(i)]);
18	pause
19	end
20	%- -
21	function dYfuncvecdt = ODEfun(t,Yfuncvec);
22	h1 = Yfuncvec(1);
23	h2 = Yfuncvec(2);
24	h3 = Yfuncvec(3);
25	Ao = pi / 100; %Area of drainage orifice at bottom of tanks (m^2)
26	g = 9.81;
27	q1 = .05; %Volumetric flowrate into Tank 1(m^3/s)
28	q2 = .05; %Volumetric flowrate into Tank 2(m^3/s)
29	if (t <= 50), q3 = .05; else q3 = .05 + .05 * .1; end %Volumetric flowrate into Tank 3(m^3/s)
30	H1 = 4; %Height of Tank 1(m)
31	H2 = 2.5; %Height of Tank 2 (m)
32	H3 = 2.5; %Height of Tank 3 (m)
33	D1 = 4; %Upper diameter of Tank 1(m)
34	D2 = 1; %Lower diameter of Tank 2 (m)
35	if (h1 < h2), q4 = 0 - (Ao * sqrt(2 * g * (h2 - h1))); else q4 = Ao * sqrt(2 * g * (h1 - h2));end %Volumetric flowrate out of Tank 1 (m^3/s)
36	if (h2 < h3), q5 = 0 - (Ao * sqrt(2 * g * (h3 - h2))); else q5 = Ao * sqrt(2 * g * (h2 - h3)); end %Volumetric flowrate out of Tank 2 (m^3/s)
37	q6 = Ao * sqrt(2 * g * h3); %Volumetric flowrate out of Tank 3 (m^3/s)
38	if (h1 >= H1), dh1dt = 0; else dh1dt = (q1 - q4) / (pi / 4 * (D2 + (D1 - D2) / H1 * h1) ^ 2); end %Liquid level in Tank 1 (m)
39	if (h2 >= H2), dh2dt = 0; else dh2dt = (q2 + q4 - q5) / (pi * exp(2 * h2)); end %Liquid level in Tank 2 (m)
40	%Liquid level in Tank 3
41	if (h3 >= H3), dh3dt = 0; else dh3dt = (q5 + q3 - q6) / (h3 ^ 2); end
42	dYfuncvecdt = [dh1dt; dh2dt; dh3dt];

tive error tolerance to *RelTol* = 10^{-6} yields a more accurate, non-oscillatory solution. Since the latter solution is more accurate by three orders of magnitude, the difference between the solutions can serve as a measure of the "error" in the less accurate solution. This "error" in h_3 is shown in Figure 4 (note that a few points of larger errors, in the region where the liquid level changes more rapidly, are not shown). Note that the maximal errors are of the order of 10^{-3} (while $h_3 \sim 1.2$, see Table 2). Thus, the integration algorithm indeed keeps the error below the desired relative error tolerance.

The error in the slope dh_3/dt is shown in Figure 5 (a few points of larger errors are not shown in this plot). The maximal absolute errors are of the order 10^{-4}, but since the value of $dh_3/dt \sim 4\times10^{-6}$, the relative error is of the order of 25. Thus, a reasonably low relative error of 10^{-3} in the variable (h_3) value yields absurd slope values, with up to 2500% error.

It is interesting to investigate the reasons for these huge differences in the relative error values. In Table 4, the values of some of the variables are presented at t = 3600 as calculated by the *ode45* function with the two error tolerances. Observe that there are four correct decimal digits in the values of h_3, q_5+q_3, and q_6 when the higher error tolerance of 10^{-3} is used. Introducing these numbers into Eq. (3) to calculate dh_3/dt requires subtraction of two numbers that are very close. This operation causes the relative error to increase by seven orders of magnitude!

We conclude that the erroneous oscillations in the ode45 solution and the "negative square-root" error message of the RKF45 algorithms are caused by the lack of consideration of the slope value errors by the step-size control logic of the two algorithms.

CHECKING THE ROLE OF THE PROBLEM STIFFNESS

The cause of the erroneous oscillations and error messages was identified in the previous section. To complete the exercise, however, the stiffness of the problem must also be checked.

There are several possible ways to find the elements of the $\partial f/\partial h$ matrix and to calculate the eigenvalues of this matrix at various times. One option is to use the symbolic manipulation options of MATLAB. The program for derivation of the elements of the Jacobian matrix and calculation of the eigenvalues at t = 3600 s is shown in Table 5. First, the vector of functions (Eq. 7) is defined using symbolic variables (lines 2 through 4). Note that the expressions for q_4, q_5, and q_6 (as functions of h_1, h_2, and h_3) were introduced into the functions assuming that $h_1<h_2<h_3$ for the entire time period. The symbolic differentiation is carried out in lines 5 through 9, and partial derivative expressions are stored in the matrix df. Numerical values are introduced into the constants (lines 10 and 11) and the final values of h_1, h_2, and h_3, from Table 2 are introduced into these variables in line 12. Finally, the numerical values of the terms of the Jacobian matrix (line 13) and the eigenvalues of this matrix (line 15) are calculated. The eigenvalues are: λ_1 = 0.37297e-4; λ_2 = -0.04478; and λ_3 = -0.106922. Introducing these values into the stiffness-ratio equation yields:

$$SR = \frac{\lambda_3}{\lambda_1} = 286.67$$

As the stiffness ratio is less than 1000, the system is not considered to be stiff according to the stiffness-ratio criterion.

TABLE 4
Comparison of Some Variables Values as calculated by the ode45 Function with Two Error Tolerances.

Relative Tolerance	t (s)	h3 (m)	q5+q3 (m3/s)	q6 (m3/s)	dh3/dt
10^{-6}	3600	1.2003444895	0.1524588034	0.15245874	4.41426e-08
10^{-3}	3600	1.2000989262	0.1524832835	0.15244314	2.78699e-05
% Error		0.020	-0.016	0.010	63036.0

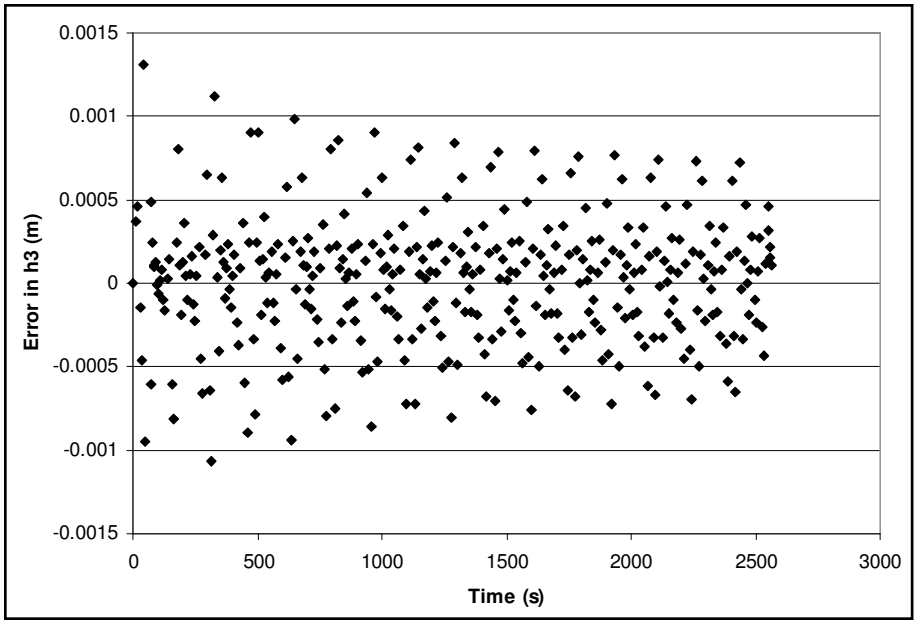

Figure 4. *Error in the Liquid Level h_3 (*ode45 *Algorithm,* RelTol = *10^{-3}).*

CONCLUSIONS

The exercise presented here demonstrates a systematic approach for identification of causes for a failure in an ODE integration algorithm used to reach a correct solution of a particular problem. The procedure starts with checking whether a similar algorithm in a different package yields the same solution, and whether the default algorithm parameters are appropriate. Then, the stiffness of the problem is assessed, and a comparison of solutions obtained by stiff and non-stiff algorithms is carried out.

In courses where students routinely use numerical software for solving ODE's, carrying out the two parts of the exercise can be very beneficial. They demonstrate that results of ODE solvers must be verified. The reduction of error tolerances and problem solution by different packages are very efficient in detecting inaccurate solutions. The Polymath package, in particular, is well-suited for such tests as it enables exporting the model equation to other packages.

For students with appropriate numerical analysis background, this two-part exercise provides a good demonstration of the concepts of error estimation, step size control, error propagation and stiffness in systems of ODE's using a real-life example. This exercise demonstrates that a complete analysis and validation of the results are important in problem solving and for maintaining confidence in a particular software package. While students may use software packages for routine problem solving with a limited knowledge of the numerical methods utilized by the package, there is also a need to include numerical methods within the curriculum. This knowledge is needed when difficulties are encountered during numerical problem solving.

ACKNOWLEDGMENTS

The authors wish to thank Auburn University chemical engineering student Robert Edwards for first bringing this problem to our attention.

REFERENCES

1. Cutlip, M. B., J.J. Hwalek, H.E. Nuttall, M. Shacham, J. Brule, J. Widman, T. Han, B. Finlayson, E. M. Rosen, and R. Taylor, "A Collection of Ten Numerical Problems in Chemical Engineering Solved by Various Mathematical Software Packages," *Comput. Appl. Eng. Educ.*, **6**(3), 169(1998)
2. Enright, W.H., "The Design and Implementation of Usable ODE Software," *Numerical Algorithms*, **31**, 125 (2002)
3. Brauner, N., M. Shacham and M.B. Cutlip, "Computational Results: How Reliable Are They? A Systematic Approach to Modal Validation," *Chem. Eng. Ed.*, **30**(1), 20 (1996)
4. Ashurst, W.R., and R.J. Edwards, *Personal Communications* (2006)
5. Rice, R.G. and D.D. Do, *Applied Mathematics and Modeling for Chemical Engineers*, John Wiley (1995) ❐

TABLE 5
MATLAB Program for Calculating the Eigenvalues of the ∂f/∂h Matrix

No.	Commands
1	syms q1 q2 q3 g h1 h2 h3 H1 Ao D1 D2 g
2	f(1)= (q1 - (Ao * sqrt(2 * g * (h1 - h2)))) / ((pi / 4) * ((D2 + ((D1 - D2) / H1) * h1) ^ 2));
3	f(2)= (q2 + (Ao * sqrt(2 * g * (h1 - h2)))- (Ao * sqrt(2 * g * (h2 - h3)))) / (pi * exp(2 * h2));
4	f(3)= ((Ao * sqrt(2 * g * (h2 - h3))) + q3 - ((Ao * sqrt((2 * g * h3))))) / (h3 ^ 2);
5	for i=1:3
6	df(i,1)=diff(f(i),h1);
7	df(i,2)=diff(f(i),h2);
8	df(i,3)=diff(f(i),h3);
9	end
10	Ao = pi / 100; g = 9.81; q1 = 0.05; q2 = 0.05; q3=0.05 + 0.05 * 0.1;
11	H1 = 4; H2 = 2.5; H3 = 2.5; D1 = 4; D2 = 1;
12	h1=1.8656; h2= 1.7367; h3=1.2282;
13	df_numeric=subs(df);
14	format long g
15	Lambda=eig(df_numeric)

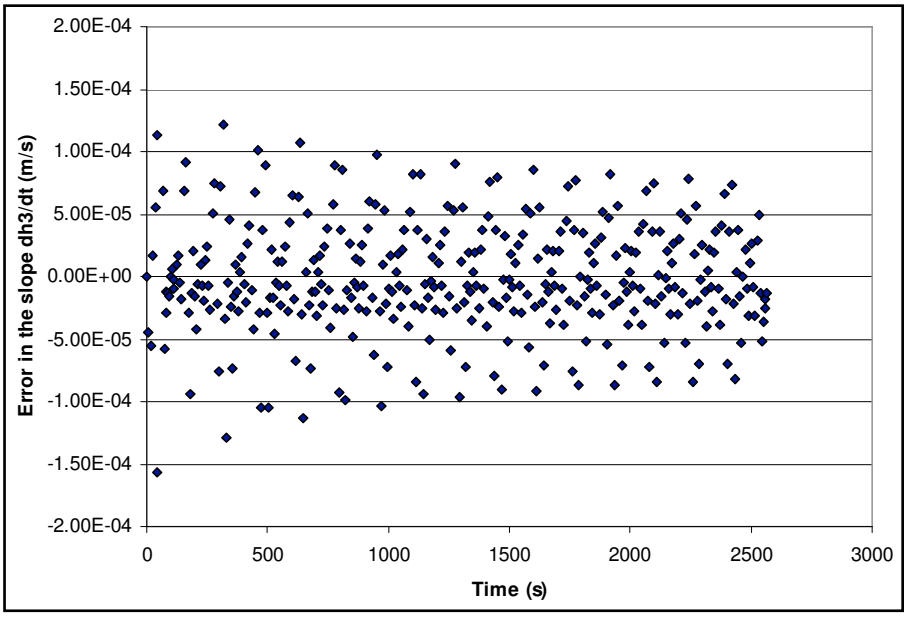

Figure 5. *Error in the liquid level slope:* dh_3/dt *(*ode45 *Algorithm,* RelTol = 10^{-3}*).*