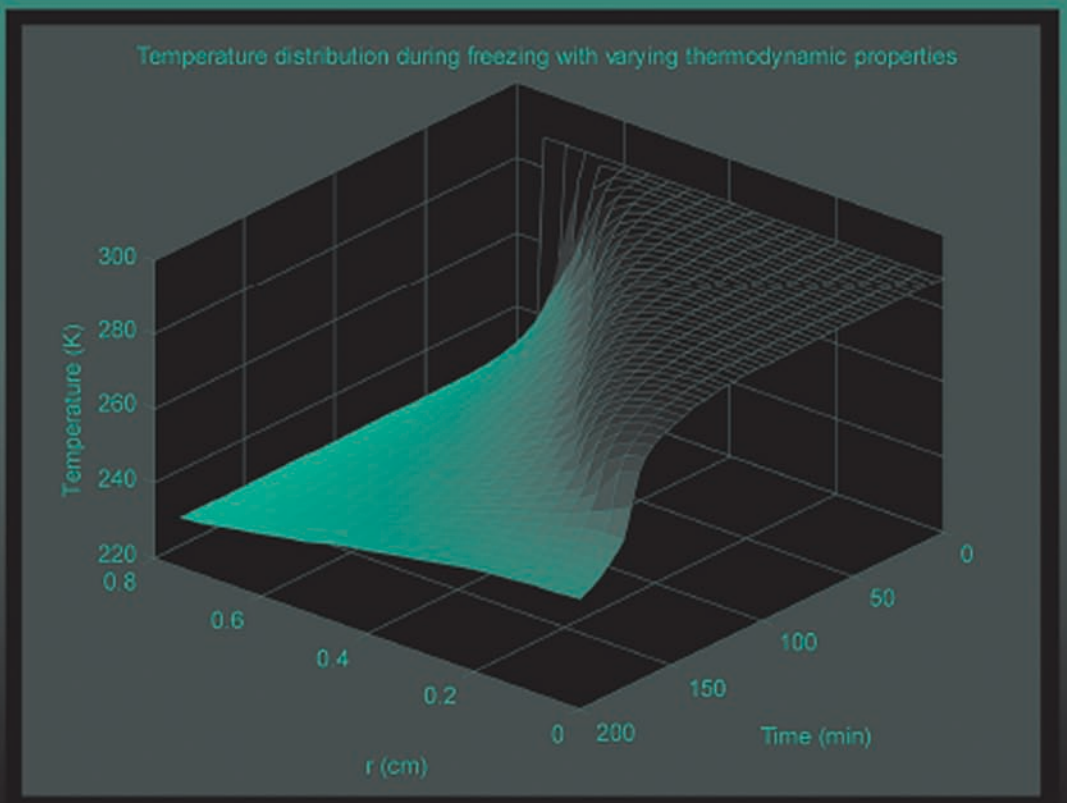
Handbook of Food Process Modeling *and* Statistical Quality Control

with Extensive MATLAB® Applications

SECOND EDITION



Mustafa Özilgen

CRC Press
Taylor & Francis Group

Handbook of Food Process Modeling *and* Statistical Quality Control

with Extensive MATLAB® Applications

SECOND EDITION

Supplementary Resources Disclaimer

Additional resources were previously made available for this title on CD-ROM. However, as CD-ROM has become a less accessible format, all resources have been moved to a more convenient online download option.

You can find these resources available here: www.routledge.com/9781439814864

Please note: Where this title mentions the associated disc, please use the downloadable resources instead.

Handbook of Food Process Modeling *and* Statistical Quality Control

with Extensive MATLAB® Applications

SECOND EDITION

Mustafa Özilgen

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20141208

International Standard Book Number-13: 978-1-4398-7767-8 (eBook - PDF)

To Sibel and Arda

Contents

Preface

It has been more than a decade since the first edition of this book appeared on shelves. Paperback, hardbound, and e-book versions of the first edition were available in the market. More than 130 Internet booksellers included the first book on their lists; I was more than happy with the welcome of the scientific community. Students who used the first edition in their classes are now directors of major food establishments and, I am proud of them all.

The second edition developed by way of an opportunity that presented itself. I taught classes at the Massey University in New Zealand; we established our own company in Ankara, Turkey. I chaired the Chemical Engineering Department, Yeditepe University in Istanbul, where the most notable contributors were starting a PhD program and a food engineering department. Teaching bioengineering classes to the genetic engineering students was one of my most exciting experiences.

Turkey has the 18th largest economy in the world and the food industry makes up a big part of it. There are about 45 food engineering departments in Turkish Universities. I was honored to be among the founders of the first and 39th departments. The 39th department was the first food engineering department in a foundation (private) Turkish university.

I appreciate the contributions of Seda Genc, Fatih Uzun, and all of my undergraduate and graduate students, who helped to write the MATLAB® codes through their projects or homework. I appreciate the help provided by Dr. Esra Sorguven of the Mechanical Engineering Department of Yeditepe University, in the solutions of the examples involving the partial differential equation toolbox. I also appreciate the author's license from MATLAB® (MathWorks Book Program, A#: 1-577025751).

The second edition of the book is substantially different from the first edition in the sense that

i. The title of the book is modified following the recommendations of experts from academia and the industry. It is intended to present the book as a compendium of applications within its scope.
ii. The new edition covers extensive MATLAB applications. The model equations are solved with MATLAB and the resulting figures are generated by the code. The models are compared mostly with real data from the literature. Some errors occurred while reading the data; therefore, the model parameters sometimes had different values than those of the references.
iii. Tabular values and plots of mathematical functions are produced through MATLAB codes.
iv. All of the MATLAB codes are given on the CD accompanying the book. A summary of the important features and functions of the MATLAB codes used in the book are given in Table 1.1. The readers may refer to this table to locate the functions or syntax they need. They may copy lines from the examples and write their own code with them. I wrote my own codes by following this procedure. I would recommend achieving each task in the code in a stepwise manner and then going on to the next task. Each task is usually defined in the examples with phrases such as

% enter the data, % plot the data, % modeling, % plot the model, etc. I tried to maintain this order in the codes when possible.

v. A few food processing methods (i.e., *pulsed electric field* and *high pressure processing*) gained importance after the first edition. Some examples are included to cover the development.

vi. Feedback to the first edition indicated that simple models were welcomed, while the sophisticated ones were avoided. The *80%–20% rule; that is*, *obtaining 80% of the total possible benefits within the 20% of the highest difficulty level of the models* continued to be the motto. Examples to comprehensive and easy mathematical models with sound theoretical background and a larger scope of application are given priority. Using MATLAB helped to achieve this goal.

vii. It should be noted that this book was authored for educational purposes only. The models were compared with real experimental data to make them as realistic as possible. Some commercial applications, design, or research may need more accuracy, which is beyond the scope of this book.

viii. The statistical toolbox of MATLAB was used extensively in Chapter 5. In some examples, relatively longer solutions were preferred to the shortcut alternatives because of their educational value. Sampling methods with new acceptance were published during the last decade. They are included in the book, while the older practices were removed.

ix. I have gone through the files of classes that I have been teaching over the years and added selected exam questions to the end of each chapter. The comprehension questions were designed to test the students' understanding of the topics. Correct answers to these questions are prerequisite for understanding the rest of the material.

I will be more than happy to hear recommendations. I will evaluate them carefully to make future editions of the book more useful.

Summary

The *Handbook of Food Process Modeling and Statistical Quality Control* is written along the guidelines of the "80%–20% rule," which means obtaining 80% of the total possible benefits within the 20% of the highest attainable modeling difficulty level. Fundamental techniques of mathematical modeling of processes essential to the food industry are explained in this text. Instead of concentrating on detailed theoretical analysis and mathematical derivations, important mathematical prerequisites are presented in summary tables. Readers' attention is focused on understanding modeling techniques, rather than the finer mathematical points. Examples of comprehensive and easy mathematical models with sound theoretical background and a larger scope of application are given priority. MATLAB has been used extensively to achieve this goal.

Topics covered include modeling of transport phenomena, kinetics, and unit operations involved in food processing and preservation. Statistical process analysis and quality control as applied to the food industry are also discussed. The book's main feature is the large

number of fully worked examples presented throughout. Included are examples from almost every conceivable food process, most of which are based on real data provided from numerous references. Each example is followed by a clear, step-by-step worked solution, and the associated MATLAB code.

Tabular values and plots of mathematical functions are also produced with MATLAB. All of the codes are given in the CD accompanying the book. A summary of the MATLAB functions and syntax used in the book are given in a table, so the readers will be able to locate them easily. Most of the codes are written in the same sequential order and the readers are informed about them in the code with remarks like *% enter the data*, *% plot the data*, *% modeling*, *% plot the model*, and so on. There are also in-depth explanations in the codes to help readers understand them easily. Comprehension questions were added to each chapter to test the students' understanding of the topics.

This book contains 163 fully solved examples, 217 MATLAB codes (provided in full detail), 273 figures (most of which are printouts of the codes), and 52 tables.

Mustafa Özilgen, PhD
Professor of Food Engineering
Yeditepe University
Istanbul, Turkey

Author

Mustafa Özilgen is a chemical engineer. He has a BS and MS from the Middle East Technical University in Turkey and a PhD from the University of California–Davis. He is author or coauthor of numerous refereed publications and the author of two books. The first edition of this book was published with the title *Food Process Modeling and Control, Chemical Engineering Applications* (Gordon & Breach; Amsterdam, 1998). A recent book authored by Professor Özilgen is *Information Build-up During the Progression of Industrialization* (in Turkish, Arkadas Publishing Co., Turkey, 2009).

Mustafa Özilgen has taught numerous classes at the University of California–Davis, Middle East Technical University, Ankara, Turkey, and the Massey University in New Zealand. He was a member of the organizing committee and co-editor of the proceedings of CHEMECA 1998, the annual Australian and New Zealander Chemical Engineering conference. He also worked for the Marmara Research Center of Turkish Scientific and Technical Research Center, Gebze. He was a recipient of one of the major research awards offered by the Turkish Scientific and Technical Research Center in 1993. He is currently working as a professor and chairperson of the Food Engineering Department at Yeditepe University, Istanbul, Turkey.

1

Introduction to Process Modeling

1.1 The Property Balance

Most of the mathematical models, which appear in the engineering literature, are based on the balance of one or more conserved properties. The property balance starts after choosing an abstract or conceptual system. The universe, which remains outside of the system, is referred to as the surroundings. The property balance around the system described in Figure 1.1 may be stated as

$$\left\{\begin{array}{l}\text{input}\\ \text{rate of the}\\ \text{property}\\ \text{to the}\\ \text{system}\end{array}\right\} - \left\{\begin{array}{l}\text{output}\\ \text{rate of the}\\ \text{property}\\ \text{from the}\\ \text{system}\end{array}\right\} + \left\{\begin{array}{l}\text{generation}\\ \text{rate of the}\\ \text{property}\\ \text{within the}\\ \text{system}\end{array}\right\} - \left\{\begin{array}{l}\text{consumption}\\ \text{rate of the}\\ \text{property}\\ \text{within the}\\ \text{system}\end{array}\right\} = \left\{\begin{array}{l}\text{accumulation}\\ \text{rate of the}\\ \text{property}\\ \text{within the}\\ \text{system}\end{array}\right\}$$

or

$$\psi^{\bullet}_{in} - \psi^{\bullet}_{out} + \psi^{\bullet}_{gen} - \psi^{\bullet}_{con} = \psi^{\bullet}_{acc}, \tag{1.1}$$

where:

$\psi^{\bullet}_{in}$ = rate at which an extensive property enters the system
$\psi^{\bullet}_{out}$ = rate at which an extensive property leaves the system
$\psi^{\bullet}_{gen}$ = rate at which an extensive property generated in the system
$\psi^{\bullet}_{con}$ = rate at which an extensive property consumed in the system
$\psi^{\bullet}_{acc}$ = rate at which an extensive property accumulates in the system.

In this formulation the *input* includes all extensive property that is added to the system across the system boundary and the output describes the amount of extensive property that leaves the system across the system boundary. Properties, like heat, may be transferred to or from the system without any material crossing the system boundary. The generation and the consumption terms describe the quantity of an extensive property that is created or destroyed in the system, respectively.

In Equation 1.1 the "*conserved property*" Ψ may be total mass, an atom, a molecule, linear or angular momentum, total energy, mechanical energy, or charge. Property balances

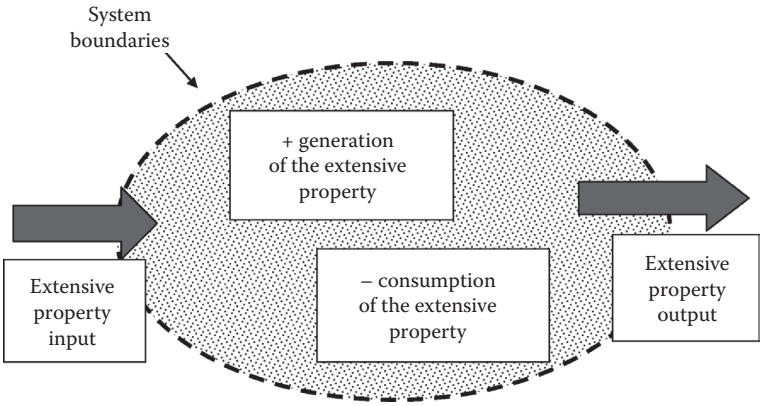


FIGURE 1.1
Description of the system for the application of the conservation laws.

were performed for many centuries without recognizing their common nature. Newton's second law of motion, developed in 1687 to relate the net force on an object to its mass and acceleration, is indeed an application of the conservation law to linear momentum. It was one of the earliest examples to the conservation laws. The first law of thermodynamics is actually an energy balance, engineering Bernoulli equation that is used to calculate energy dissipation by a liquid while flowing through a pipe and Kirchhoff's voltage law, which was formulated in 1845, are indeed different expressions for the first law of thermodynamics. Kirchhoff's current law, which states that the total charge flowing into a node must equal the total charge flowing out of the node, is developed on the concept of conservation of charge, and based on the same concept with the mass balance.

We will use MATLAB® extensively to solve the equations resulting from the property balances. The word MATLAB stands for *matrix laboratory*. The "Command Window" consists of a sequence of executable statements. It calls the built-in functions or the M-files and executes them (*MATLAB, Getting Started Guide* 2008). Table 1.1 gives a list of the selected examples where you may see the application of some skills, syntax, and functions in a working code. You may refer to MATLAB help for further information.

MATLAB skills, syntax, and functions described in Table 1.1 are also used in numerous other examples, which are not listed here. Tables 5.1, 5.2 and 5.3 provide more information about statistical MATLAB tools. Tables 2.10 through 2.17 provide more information about MATLAB functions related with Bessel functions and error functions. Table 2.20 describes how to obtain the Laplace transforms by using the symbolic toolbox. In order to get additional information about any topic covered in Table 1.1 (e.g., *plotyy*) type *lookfor plotyy* or *help plotyy* or doc plotyy and enter while you are in the command window. You may also search for *MATLAB plotyy* on the Internet; in addition to the large number of documents provided by MATLAB and its users, you may join the forums where users share information.

You may experience problems with the apostrophe '. Although most of the computers support the ASCII code apostrophes, there are some computers that do not. If you have one of these computers, the apostrophe may appear on the screen, but the software may not recognize it. If your code should not work because of this reason, get a working apostrophe from a running example and replace the nonworking ones by the copy and paste method.

TABLE 1.1

MATLAB® Skills, Syntax, and Functions Guide

Skills, Syntax, and Functions	Explanation and Examples to Refer
1. Plotting skills, syntax, and functions guide	
plot	plot(B(1:4,3),D(1:4),'--*') % plots first to fourth row, third column elements of matrix B versus, from first to fourth elements of D vector. Notation '--*' makes a dashed line with legend * at the data or computation points. Example 1.1 plot(tData1,a,'x',tData2,b,'o'); hold on, % plots a versus tData1 with legend 'x' and b versus tData2with legend ,'o'. Remark hold on makes the plot wait for the execution of the next lines, so they are plotted together. Example 1.4 f = ['ro', 'bd', 'gv', 'ks', 'm*']; % defines the color and the legends (ro is red o, bd is blue diamond, gv is green triangle, ks is black square, m* is magenta *) hold on; plot(time, C, f((2*(6-i)-1):(2*(6-i)))); % makes the plot by using the colors and the legends described by f. Example 3.11
plotyy	[AX,H1,H2] = plotyy(time, T(2,:), time, F(2,:)); hold on % prepare a plot with y axis on both left- and right-hand sides. Example 4.10
semilogy semilogx	semilogy(tau,Lethalithy,'--') % makes a semi log plot with y axis in log scale, x axis in linear scale, Example 4.6. Other option semilogx. Figure 3.5.
loglog	loglog(tData2,fData2, 'x'); hold on % makes a plot with both x and y axis are in log scale. Example 5.51
legend	legend('T fluid','T particle surface','T particle center','Location','West') % inserts a legend to the graph 'Location','West' describes the location where you want to place the legend. Location options: East, West, South, North, SouthEast, SouthWest, NorthEast, NorthWest, Best. Example 4.10 legend(H2,'F particle surface','F particle center','Location','East') % insert a legend to the figure for the parameters in association with the right-hand side y axis. Example 4.10
figure	% if you should type figure in your code it will start a new figure. figure Example 2.1
xlim, ylim	ylim([0 5]); % limit of the range of the y axis xlim([0 1000]); % limit of the range of the x axis Example 4.11 ylim(AX(1), [80 140]) % limit of the range of the left-hand side y axis in plotyy ylim(AX(2), [0 40]) % limit of the range of the right-hand side y axis in plotyy Example 4.10 % REMARKS: Among other uses, xlim and ylim are needed when you plot a few figures on top of each other (do not forget to use hold on, after each plot). If you should not use xlim or ylim, each figure may set different limits and you may not be able to compare the model with the data.
xlabel ylabel	xlabel('Time (s)') % label for x axis ylabel('Temperature \circ C') % label for y axis set(get(AX(2),'ylabel'), 'string', 'F (min)') % label for right-hand side y axis in plotyy Example 4.10 zlabel('cratio') % z axis in a 3-D plot Example 3.30

(*Continued*)

TABLE 1.1 (CONTINUED)

MATLAB® Skills, Syntax, and Functions Guide

Skills, Syntax, and Functions	Explanation and Examples to Refer
line style	plot(t,log(c(:,1)),'-',t,log(c(:,2)),':'); hold on % line style is described with '-' (solid line) and ':' (dashed line made of point). Other options are '--' (dashed solid line) and '.-' (dashed line with point and dash characters). Example 3.1. set(H1,'LineStyle','- -') % line style of a property, which is related with the left-hand side y axis in plotyy. set(H2,'LineStyle','-') % line style of a property, which is related with the left-hand side y axis in plotyy. Example 4.10
legend style	plot(tData,log(cData1),'s',tData,log(cData2),'o'); hold on % 'v'legends: 's' square, 'o' o, other options are 'v' , '^', ' < ' , ' > ' triangles, ,'d' diamond, 'p' pentagene, 'h' hexagone, 'x', ' + ' and '*'. Example 3.1
title	title('bar chart of the log cell areas') % inserts a title to a plot. Example 5.2
surface	surf(c11,c21,r21total); hold on % plots a surface in a 3-D plot. Example 3.15
colormap	colormap gray % sets the color of the surface, other options for gray are jet, HSV, hot, cool, spring, summer, autumn, winter, bone, copper pink, lines. Example 3.15
grid	% inserts grids to a figure. Example 4.41
meshgrid	[e,t] = meshgrid(e,t); % [X,Y] = meshgrid(x,y) transforms the x and y vectors into X and Y arrays, which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. Example 3.30
set(line…)	set(line([0 1],[0 1]),'Color',[0 0 0]); % draws a line between points defined by ([0 1],[0 1]). Color of the line is defined by 'Color',[0 0 0] (black). Other alternatives are [1 1 0] yellow, [1 0 1] magenta, [0 1 1] cyan, [1 0 0] red, [0 1 0] green, [0 0 1] blue, [1 1 1] white. Example 4.41
2. Printing skills, syntax, and functions guide	
fprintf	% fprintf writes formated data. fprintf('Waxy rice starch contains 100 %% Amylopectin and 0 %% Amylose') % prints the characters between ' '. fprintf('\nTotal bond energy of the waxy rice starch is %.2g kJ/mol',Energy) % prints the characters between ' ' and value of variable Energy to the location marked as %.2g (number before . describes the field length, number after . describes the precision alternatives to g are c character, f fixed point, e exponential notation). Example 1.1 % tabulate the data: fprintf('\nAge and fraction of the liquid pockets\n\n') fprintf(' t(s) Fraction\n') fprintf('---------- ----------\n') for i = 1:11 fprintf('%-15g %7g\n',a(i,1), a(i,4)) end % \n means skipping a line the number of \'s defines the number of the lines to be skipped. Example 3.11
display	% display(X) prints the value of a variable or expression, X. display(z1); Example 5.5

TABLE 1.1 (CONTINUED)

MATLAB® Skills, Syntax, and Functions Guide

Skills, Syntax, and Functions	Explanation and Examples to Refer
3. Line and curve fitting skills, syntax, and functions guide	
polyfit	c = polyfit(tData,xDataFreeMoisture,N) % fits a Nth order polynomial to the data. Example 4.12
nlinfit	beta = nlinfit(x,y,fun,beta0) % returns a coefficients vector beta for nonlinear regression, y is the dependent variables vector, x is dependent variables vector, beta0 is the vector of the initial estimates of beta. Notice: beta may depend on beta0. Examples 2.7, 4.7
lsline	% after plotting a linear data if you should write lsline least squares line will be plotted. Example 1.3
polyval	xModel = polyval(c,tModel); % returns the value of a polynomial of degree n evaluated at tModel. The input argument c is a vector of length $n + 1$ whose elements are the coefficients in descending powers of the polynomial to be evaluated. $xModel = c(1)tModel^n + c(2)tModel^{n-1} + \ldots + c(n + 1)$ Example 4.12
4. Mathematical functions and operations skills, syntax, and functions guide	
sum	sum_xi = sum(xi); % sum all xi values Example 1.2
mean	DeffAvg50 = mean(Deff50); % mean of Deff50 values Example 4.34
factorial	Pa2 = (factorial(n)/(factorial(x)*factorial(n-x)))*(p^x)*(1-p)^(n-x); % factorial(n) is the product of all the integers from 1 to n, i.e. prod(1:n). The answer is accurate for n < = 21. Example 5.35
square root	Se = sqrt(mean(d2)); % square root of mean of d2 values Example 2.15
range	% y = range(x) returns the range of the values in x. Example 5.16
error function	c(j,k) = c1 + (c0-c1)*erf(z) ; % erf(z) error function of z. Example 2.13
complementary error function	error_func(i) = erfc(L/(2*sqrt(a*times(i)))); % complementary error function of (L/(2*sqrt(a*times(i))). Example 4.5
Bessel function	s(n) = exp(-(alpha* time(j)/(Radius^2))*(Bn(n)^2))*besselj(0,z)/ ((Bn(n)^2)*besselj(1,Bn(n))); % besselj(0,z) zero order Bessel function evaluated at z, besselj(1,Bn(n)) first order Bessel function evaluated at Bn(n). Example 2.10
complementary Bessel function	K14_besselfunction(i) = besselk(0.25,(R^2)/(8*a*times(i))); % K1/4 is the modified Bessel function of the second kind of order ¼. Example 4.5
integral	NTU = quad(@calculateNTU,yB,yA) % quad computes the integral described by function 'calculate' between the limits yB and yA with fourth order Runge-Kutta method. Example 4.39
fminsearch	Lmin = fminsearch(y,5); % starts at 5 and attempts to find a minimizer Lmin of y. Example 5.54
ceil round floor fix	B = ceil(A); % rounds the number A toward plus infinity (the other alternatives are B = round(A), which rounds A to the nearest integer or B = floor(A), which rounds A to the nearest integers less than or equal to A, B = fix(A) rounds the elements of A toward zero). Example 5.54

(*Continued*)

TABLE 1.1 (CONTINUED)

MATLAB® Skills, Syntax, and Functions Guide

Skills, Syntax, and Functions	Explanation and Examples to Refer
5. Statistical functions skills, syntax, and functions guide	
correlation coefficient	rmatrix = corrcoef(lnXdata1,tData1); % find the correlation coefficient matrix of the data given in vectors lnXdata and tData1 r = rmatrix(1,2); % convert the correlation coefficient matrix into a single number. Example 2.15

standard error	% determine the standard error
	for j = 1:1:size(tData1);
	lnXmodel = lnX0-K*tData1;
	d1 = (lnXdata1-lnXmodel);
	d2 = d1*d1';
	end;
	Se = sqrt(mean(d2));
	Example 2.15

hypothesis testing concerning one mean	H = ztest(xBarExp,mu,sigma) % where xBar is the sample mean and sigma is the population standard deviation. The outcome H = 0 indicates that the hypothesis cannot be rejected, the outcome H = 1 indicates that the null hypothesis can be rejected. Example 5.14
hypothesis testing concerning two means when the population standard deviations are known	% refer to Example 5.15
binocdf	% y = binocdf(x,n,p) returns the binomial cumulative distribution function with parameters n and p at the values in x. ATI2 = n + (N-n)*(1-binocdf(c,n,p)); Example 5.54
normcdf	% p = normcdf(x,mu,sigma) returns the cumulative distribution function of the normal distribution with mean mu and standard deviation sigma, evaluated at the values in x. fModel2 = normcdf(log(t),lnMuTemp2,sigma); Example 5.51
anova1	% (anova one means one way of analysis of variance) p = 100*anova1(x,[ ], 'off'); % p = probability of having the rows of matrix x be the same. Example 5.25
anova2	% [p,table] = anova2(...) returns two items where p is a vector of p-values for testing column, row, and if possible interaction effects, table is a cell array containing the contents of the anova table. [p Table] = anova2(x,1, 'off') Example 5.26
vartest2	H = vartest2(X,Y) performs an F test of the hypothesis that two independent samples, in the vectors X and Y, come from normal distributions with the same variance H = vartest2(Vjuice,Vbeverage,alpha) Example 5.24
normspec	normspec(specs,mu,sigma,*region*) % shades the region either 'inside' or 'outside' the specification limits. Figure 5.2.
6. Solution of algebraic, ordinary, and partial differential equations	
division of matrices, solution of set of linear algebraic equations	H = D./MM; Example 1.1
fzero	xe(i) = fzero('EthanolEquilibrium',0.5); % finds the value of the independent variable xe, which makes the value of the function 'EthanolEquilibrium',zero. around xe = 0.5. Example 4.41

TABLE 1.1 (CONTINUED)

MATLAB® Skills, Syntax, and Functions Guide

Skills, Syntax, and Functions	Explanation and Examples to Refer
ode45	% ode45 solves the ordinary differential equations. [t,c] = ode45(@ascorbicacid1,tspan,c0); solves the ordinary differential equation described in the m-function 'ascorbicacid1' tspan is the time span (time from beginning to end, you may also enter a time matrix) of the solution, c0 is the initial condition. The solution returns t and c values, which are given on the left-hand side of the syntax. Example 3.1. % we may also use ode45 to solve a set of simultaneous ordinary differential equations. Example 3.4
pdepe	sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,r,t); % solves initial-boundary value problem defined in m-function 'pdex1pde' with the initial condition defined in function 'pdex1ic' with the boundary conditions defined in function 'pdex1bc'. m = 2 for the given problem, r = computation points along the radius, t = time matrix. Example 4.20
pdetoolbox	% The Partial Differential Equation Toolbox contains tools for solution of partial differential equations in 2-D space and time. A set of command-line functions and a graphical user interface let you preprocess, solve, and postprocess generic 2-D PDEs. Example 4.16

Example 1.1: Application of the First Law of Thermodynamics to Nutrition

Equation E.1.1.1 may be used to describe the conservation of energy around the system boundaries (Figure E.1.1.1) after substituting

$$\psi^{\bullet}_{m} = m^{\bullet}_{in}(u + e_p + k + P\nu)_{in} + Q, \tag{E.1.1.1}$$

$$\psi^{\bullet}_{out} = m^{\bullet}_{out}(u + e_p + k + P\nu)_{out}, \tag{E.1.1.2}$$

and

$$\psi^{\bullet}_{acc} = \frac{d}{dt}\left[m_{syst}(u + e_p + k)_{syst}\right], \tag{E.1.1.3}$$

where $m^{\bullet}$ is the mass flow rate, u is the *internal energy* per unit mass, e_p is the *potential energy* per unit mass, and k is the kinetic energy per unit mass. The term Q represents the heat entering into the system without being associated with the incoming mass.

Internal energy is associated with the energy stored in the atomic and the molecular structure. Potential energy is associated with the position of an object with respect to a reference position. The term $P\nu$ is the pressure–volume work describing the work done by the molecules behind a specific molecule that pushes it into or out of the system, where P is the pressure and ν is the volume per unit mass of the fluid. Heat transfer with conduction, convection, or radiation may be achieved independent of the mass influx or out flux from the system. Therefore the term Q appears as a separate entity in Equation E.1.1.1. If the system consumes energy by performing work, the term $\psi^{\bullet}_{con}$ of the general property balance equation E.1.1.1 may be described as

$$\psi^{\bullet}_{con} = W. \tag{E.1.1.4}$$

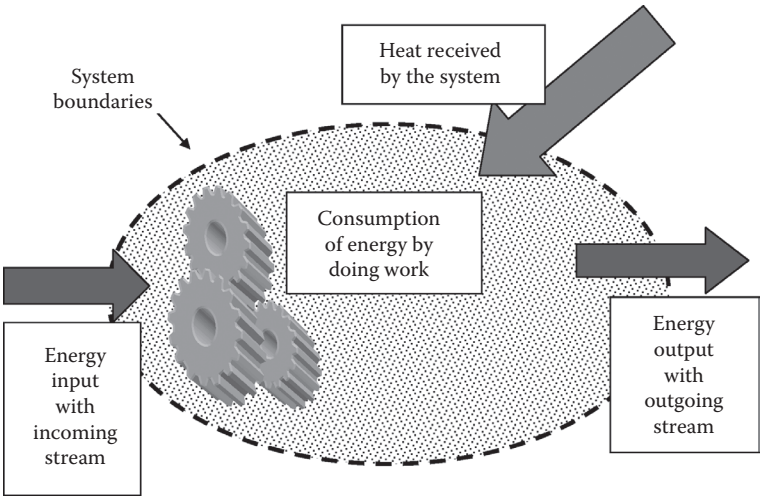


FIGURE E.1.1.1
Description of the system for conservation of energy.

After substituting Equations E.1.1.1 through E.1.1.4 in Equation 1.1 we will obtain the *first law of thermodynamics* as

$$\left[m^{\bullet}(u+e_p+k+Pv)\right]_{\text{in}}-\left[m^{\bullet}(u+e_p+k+Pv)\right]_{\text{out}}+Q-W=\frac{d}{dt}\left[m_{\text{acc}}(u+e_p+k)_{\text{acc}}\right] \quad \text{(E.1.1.5)}$$

Organisms store internal energy, u, within their structures. High energy bonds of the fat and starch molecules are the most common energy reserves of the animal and the plant cells, respectively. These bonds are broken to release their energy to achieve the biological processes. Energy and the number of the interatomic bonds of some common fatty acids are given in Table E.1.1.

a. Starch is produced as granules in the plants cells. All granules consist of amylose and amylopectin in percentages that change with the source. About 1000–4000 glucose units are linked with α-(1→4) bonds to make amylose. There are about 2000–20,000 α-(1→4) linked glucose units in amylopectin. There is also one residue in about every 20 glucose units linked with α-(1→6) bonds and form the branch points in amylopectin. Amylose is made up of between 1000 and 4400 and amylopectin is formed of 2000–200,000 glucose units. MATLAB® code E.1.1.a computes the total bond energy of starch molecules originating from waxy rice (code 1), rice (code 2), cassava (code 3), corn (code 4), wheat (code 5), and sweet potato (code 6). In order to run this m-function, after saving it in your computer, go to the command window, type starch, enter it. You will be asked to enter the code of the starch of interest. Enter it. The total bond energy of the specified starch with 10,000 glucose monomers will appear on the screen.
b. Calculate the total bond energy of each fatty acid.
   We may describe the total bond energy of the fatty acids in matrix form as

$$|D|=|B||C|^{T}. \quad \text{(E.1.1.6)}$$

   Where corresponding elements of $|B|$ and $|C|$ give the number and the energy of the bonds in one mole of a fatty acid. Matrix $|D|$ gives the total bond energy of each fatty acid stated in $|A|$. MATLAB® code E.1.1.b is employed to print out the total bond energy of the fatty acids. The total bond energy of the fatty acids is also plotted as a function of their C–H bonds. The bond energy of 17 C containing fatty acids are also plotted as a number of the C = C bonds in their structure.
c. When a 70 kg person runs at a speed of 10 km/hour, he burns 50 kJ in a minute. Determine how many grams of each fatty acid he consumes in 1 h.

TABLE E.1.1

Molar Mass, Number of the Interatomic Bonds/Molecule, and Energy/Bond of the Glucose Units of Starch and Fatty Acids

BOND and its Energy (kJ/mol) →	**C = C (cdc) 606**	**C–C (cc) 334**	**C–H (ch) 410**	**C = O (cdo) 723**	**O–H (oh) 456**	**C–O (co) 330**
Molar mass ↓	**Number/ Molecule ↓**	**Number/ Molecule ↓**	**Number/ Molecule ↓**	**Number/ Molecule ↓**	**Number/ Molecule ↓**	**Number/ Molecule ↓**
Glucose in amylose 180 (g/mole)	0	6	5	0	2	4
Glucose in amylopectin 180 (g/mole)	0	7	7	0	4	5
Lauric acid 200 (g/mole)	0	11	23	1	1	1
Myristic acid 228 (g/mole)	0	13	27	1	1	1
Palmitic acid 257 (g/mole)	0	15	31	1	1	1
Stearic acid 285 (g/mole)	0	17	35	1	1	1
Palmitoleic acid 299 (g/mole)	1	14	29	1	1	1
Oleic acid 283 (g/mole)	1	16	33	1	1	1
Linoleic acid 281 (g/mole)	2	15	31	1	1	1
Linolenic acid 278 (g/mole)	3	14	29	1	1	1

MATLAB® CODE E.1.1.a

M-File:

```
function starch
disp({'1. Waxy Rice';'2. Rice';'3. Corn';'4. Cassava';'5. Wheat';'6.
Sweet Potato';});
EN = [410 334 330 456];
BRAMILO = [5 6 4 2];
AMILO = [7 7 5 4];
choice = input('Please enter the code of the source of the starch:
','s');
switch choice

case '1'
n = 10000/25;
Energy = n*sum(EN*BRAMILO')+ (10000-n)*sum(EN*AMILO');
fprintf('Waxy rice starch contains 100 %% Amylopectin and 0 %%
Amylose')
fprintf('\nTotal bond energy of the waxy rice starch is %.2g kJ/
mol',Energy)

case '2'
n = 8000/25;
Energy = n*sum(EN*BRAMILO') + (8000-n)*sum(EN*AMILO') +
2000*sum(EN*AMILO');
```

```
fprintf('Rice starch contains 80 %% Amylopectin and 20 %% Amylose')
fprintf('\nTotal bond energy of the rice starch is %.2g kJ/
mol',Energy)

case '3'
n = 7200/25;
Energy = n*sum(EN*BRAMILO') + (7200-n)*sum(EN*AMILO') +
2800*sum(EN*AMILO');
fprintf('Corn starch contains 72 %% Amylopectin and 28 %% Amylose')
fprintf('\nTotal bond energy of the corn starch is %.2g kJ/
mol',Energy)

case '4'
n = 8300/25;
Energy = n*sum(EN*BRAMILO') + (8300-n)*sum(EN*AMILO') +
1700*sum(EN*AMILO');
fprintf('Cassava starch contains 83 %% Amylopectin and 17 %% Amylose')
fprintf('\nTotal bond energy of the cassava starch is %.2g kJ/
mol',Energy)

case '5'
n = 7400/25;
Energy = n*sum(EN*BRAMILO') + (7400-n)*sum(EN*AMILO') +
2600*sum(EN*AMILO');
fprintf('Wheat starch contains 74 %% Amylopectin and 26 %% Amylose')
fprintf('\nTotal bond energy of the wheat starch is %.2g kJ/
mol',Energy)

case '6'
n = 8200/25;
Energy = n*sum(EN*BRAMILO') + (8200-n)*sum(EN*AMILO') +
1800*sum(EN*AMILO');
fprintf('Sweat potato starch contains 82 %% Amylopectin and 18 %%
Amylose')
fprintf('\nTotal bond energy of the sweat potato starch is %.2g kJ/
mol',Energy)

otherwise
disp({'Unknown Starch!!! Please retry!!';})
end
```

A sample run of the code is given here:

```
'1. Waxy Rice'
    '2. Rice'
    '3. Corn'
    '4. Cassava'
    '5. Wheat'
    '6. Sweet Potato'

Please enter the code of the source of the starch: 5
Wheat starch contains 74 % Amylopectin and 26 % Amylose
Total bond energy of the wheat starch is 8.6e+007 kJ/mol
```

MATLAB® CODE E.1.1.b

Command Window:

```
clear all
close all

% enter the fatty acid names as characters
A = char('Lauric Acid', 'Myristic Acid', 'Palmitic Acid', 'Stearic
Acid', 'Palmioletic Acid', 'Oleic Acid', 'Linoleic Acid', 'Linolenic
Acid');

% enter the number of the C=C bonds, C-C bonds, C-H bonds, C=O
bonds, O-H bonds, and C-O bonds as the rows of a matrix

B = [ 0 11 23 1 1 1; 0 13 27 1 1 1; 0 15 31 1 1 1; 0 17 35 1 1 1; 1
14 29 1 1 1; 1 16 33 1 1 1; 2 15 31 1 1 1; 3 14 29 1 1 1];

C = [606 334 410 723 456 330]; % bond energies of the C=C, C-C, C-H,
C=O, O-H and C-O bonds

% compute the molar bond energy of each fatty acid
D = B*C';

% tabulate the results:
fprintf('\nBond energy of the fatty acids\n\n')
fprintf('Fatty acid Bond energy (J/mole)\n')
for i = 1:8
fprintf('%-15s %8d\n',A(i,:), D(i))
end

plot(B(1:4,3),D(1:4),'--*') % plot the molar bond energy of a fatty
acid as a function of the number of the C-H bonds in the saturated
fatty acids ('Lauric Acid', 'Myristic Acid', 'Palmitic Acid',
'Stearic Acid')
xlabel('number of the C-H bonds in a saturated fatty acid')
ylabel('Molar bond energy (J/mol)')

j = 1;
for i = 1:8
k = B(i,1) + B(i,2);
if k == 17
E(j,1) = j-1;
E(j,2) = D(i);
j = j + 1;
end
end

figure
plot(E(1:4,1),E(1:4,2),'--*')
xlabel('number of the C=C bonds in a 17 C fatty acid')
ylabel('Molar bond energy (J/mol)')
```

The following Table and the Figures E.1.1.2 and E.1.1.3 will appear on the screen when we run the code:

```
Bond energy of the fatty acids

Fatty acid          Bond energy   (J/mole)
Lauric Acid            14613
Myristic Acid          16921
Palmitic Acid          19229
Stearic Acid           21537
Palmioletic Acid       18681
Oleic Acid             20989
Linoleic Acid          20441
Linolenic Acid         19893
```

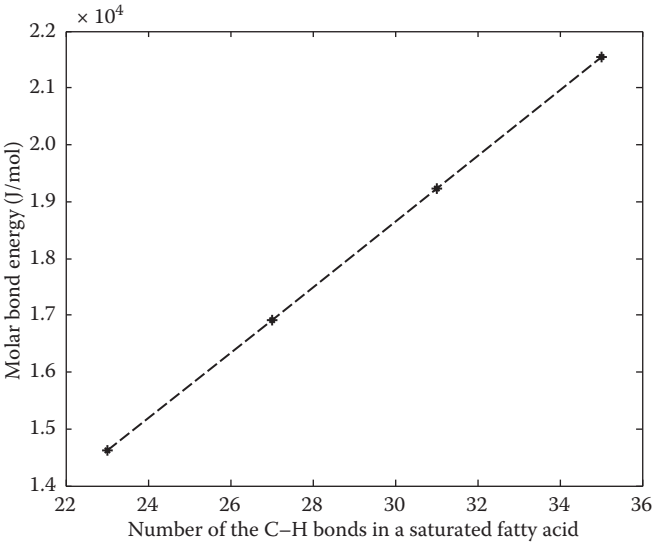


FIGURE E.1.1.2
Variation of the molar bond energy of the saturated fatty acids with a number of the C–H bonds.

There is no energy input or output or heat input, kinetic energy, or potential energy change is involved. Therefore Equation E.1.1.5 is referred to as the *first law of thermodynamics;* Equation E.1.1.5 is simplified as:

$$-W = \frac{d}{dt}\left[m_{acc}(u)_{acc}\right]. \tag{E.1.1.7}$$

Equation E.1.1.7 may be rearranged as:

$$W\Delta t = -\Delta U. \tag{E.1.1.8}$$

It is given in the problem statement that when Δt = 1h, $W\Delta t = -\Delta U$ = 50 kJ, implying that the work is done by extracting 50 kJ from the bonds of the fatty acids. MATLAB code E.1.1.c computes the fatty acid consumption requirement for producing 50 kJ of work.

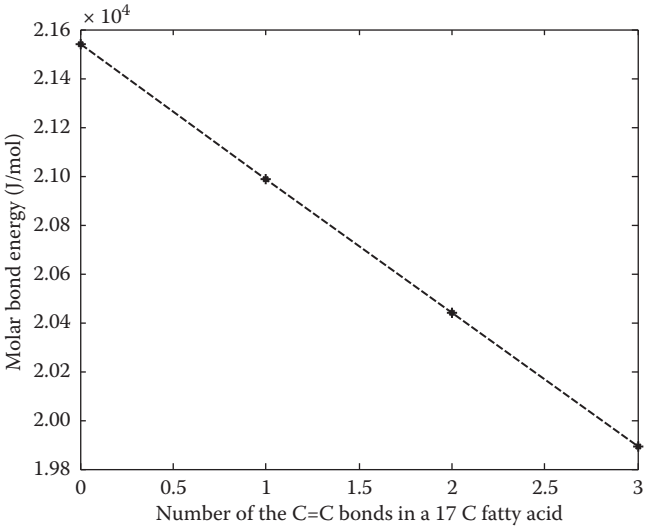


FIGURE E.1.1.3
Variation of the molar bond energy of the 17 carbon fatty acids with a number of the C = C bonds.

MATLAB® CODE E.1.1.c
Command Window:

```
clear all
close all
format short g

% enter the fatty acid names as characters
A = char('Lauric Acid', 'Myristic Acid', 'Palmitic Acid', 'Stearic
Acid', 'Palmioletic Acid', 'Oleic Acid', 'Linoleic Acid', 'Linolenic
Acid');

% enter the molar mass of the fatty acids
MM = [200 228 257 285 299 283 281 278];

% enter the molar bond energies of the fatty acids
D = [14613 16921 19229 21537 18681 20989 20441 19893];

% calculate the molar bond energy/molar mass for each fatty acid
H = D./MM;

% enter the energy requirement for one hour of running
DeltaU = 50000;

% calculate the grams of fatty acid requirement to meet DeltaU
mFattyAcd = DeltaU./MM;

% tabulate the results:
fprintf('\Fatty acid consumption in one hour\n\n')
fprintf('Fatty acid consumption (g)\n')
```

```
for i=1:8
fprintf('%-15s %7g\n',A(i,:), mFattyAcd(i))
end
```

The following Table will appear on the screen when we run the code:

```
Fatty acid          consumption   (g)
Lauric Acid           250
Myristic Acid         219.298
Palmitic Acid         194.553
Stearic Acid          175.439
Palmioletic Acid      167.224
Oleic Acid            176.678
Linoleic Acid         177.936
Linolenic Acid        179.856
```

1.2 What Is Process Modeling?

A process transforms sets of inputs into sets of desired outputs (Oakland and Followell 1995). Within the context of this book the inputs will mostly be the ingredients and energy, the outputs will be foods, and the process units will be the equipment, which are designed to achieve the purpose. A *mathematical model* is an approximate representation of a process in mathematical terms. A mathematical model can never be an actual representation of a process, since it would be very difficult, confusing, or impossible to describe the whole system with mathematical formulations. The way that people describe real life in mathematical terms is highly subjective and depends on their previous experience and education.

In typical process inputs x_1, x_2 ... x_N may generate the outputs y_1, y_2 ... y_M (Figure 1.2). With a model, the cause-and-result relation between the major process inputs (x_1, x_2 ... x_n) and outputs (y_1, y_2 ... y_m) may be formulated in mathematical terms after simplification. Negligible inputs x_{n+1} ... x_N and outputs y_{m+1} ... y_M are not included with the model. The decision about designation of the negligible and nonnegligible inputs or outputs involves personal preferences. That is the point where modeling becomes a subjective operation, not an objective work. One of the common mistakes made by engineers inexperienced in modeling is to get lost in the complexity of the process. One of the best working guides in process modeling is the *80%–20% rule*, which means *"you get 80% of the benefit with the first 20% of the model complexity"* (Glasscock and Hale 1994). The 80%–20% rule is followed in this book. Fundamental principles of mathematical modeling have been reviewed in numerous studies (Rand 1983; Meyer 1985; Riggs 1988; Teixeira and Shoemaker 1989; Luyben 1990).

Mathematical modeling is usually done after obtaining the data in tabular and graphical forms. The model is a shorthand description of the data and estimates the values of the outputs (y_1, y_2 ... y_m) when the values of the inputs (x_1, x_2 ... x_m) are entered. The model may help to understand the details of the relation between the inputs and the outputs, which may not be understood by plotting the data only, and may explain the mechanism of the events. In the following pages, we will frequently have the sentence *comparison of the experimental data and the model is shown in Figure ....* This sentence actually means that two of the boxes given in the figure are compared. Usually experimental data will be presented with symbols and will represent the process; the mathematical model will be obtained after the

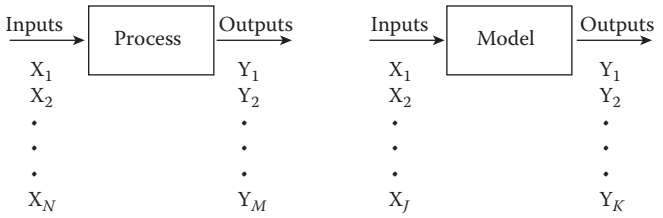


FIGURE 1.2
Comparison of input/output for a process and its model.

pertinent steps, will consist of the other box, and will be presented with solid, dashed, or dotted lines.

A good mathematical model should be general (apply to a wide variety of situations), realistic (based on correct assumptions), precise (its estimates should be finite numbers, or definite mathematical entities), accurate (its estimates should be correct or very near to correct), and there should be no trend in the deviations of the model from the experimental data. A good model should be robust (relatively immune to errors in the input data) and fruitful (its conclusions are useful or points the way to other good models).

Mathematical models may be categorized as *empirical*, *analog*, or *phenomenological* models depending on the basis that the functional relation is suggested. An empirical model assumes the form of the functional relation between the input and the output variables. There is usually no theoretical background sought while suggesting this relation. Empirical models are best when used within the range of the experimental data they are based on. An analogy model may be suggested for a relatively less known process by considering its similarity to a well-known process; that is, electrical circuit analogs may be used for modeling heat transfer or stress/strain relations. The phenomenological models use theoretical approach based on conservation of mass, energy, momentum, and so on to suggest the form of the mathematical model. They may include many different types including microscopic (distributed parameter) or macroscopic (lumped parameter) models.

The first step to building a mathematical model is a definition of the system. The answer to the question "What is going to be predicted by the model by what input data?" should be given while defining the system. Controlling factors of the system should be identified and the data should show the effects of the individual controlling factors. The system may be simplified after neglecting the effects of the marginal inputs (i.e., x_{n+1} ... x_N and outputs y_{m+1} .... y_M). The form of the mathematical model may be suggested by an empirical, analog, or phenomenological approach. The availability of information in the literature about the system, skills, and education of the modeler and purpose of modeling usually determines the form of the model suggested. In Chapters 2 through 4, fundamental principles of phenomenological model building with an application of kinetics, transport phenomena, and unit operations to food engineering processes will be discussed in detail with a theoretical background and examples. We usually end up with a single or a set of mathematical equations after using these fundamental principles. Techniques for a solution to these equations will be discussed in Chapter 2. Empirical model building will be discussed in Chapter 3. An application of mathematical modeling to process control will be discussed in Chapter 5. A comparison of the mathematical model (i.e., solution of the equations) with the experimental data is the final stage of modeling. The model is validated if it agrees with the data. If such an agreement should not be obtained, all the steps of modeling, starting with the definition of the system, is repeated until obtaining satisfactory representation (Figure 1.3).

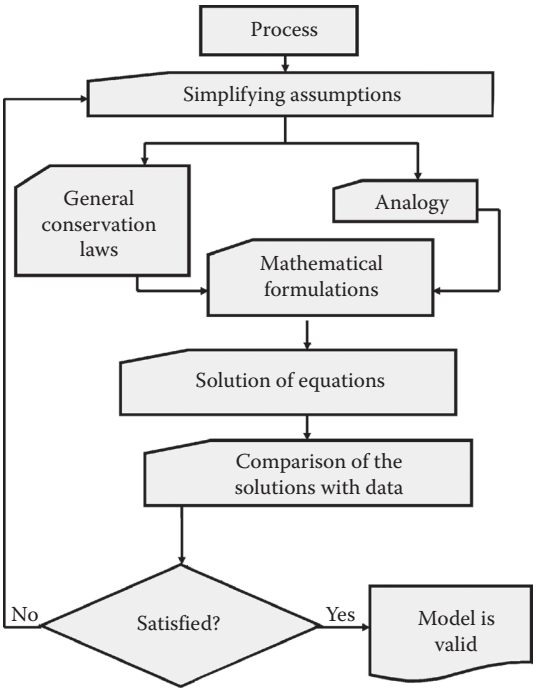


FIGURE 1.3
Schematic description of modeling.

1.3 Empirical Models and Linear Regression

Representation of large amounts of experimental data by means of empirical equations is a practical necessity in science and engineering. The empirical models are easy to use in mathematical operations over a continuous range. The form of the empirical models may be suggested by theoretical or dimensional analysis or by intuition. The simplest empirical model is a line:

$$y = ax + b, \tag{1.2}$$

where y is the dependent variable and x is the independent variable. In a plot of y versus x, parameter a is slope and b is the intercept with $x = 0$ axis. If it is possible to linearize an equation, parameters a and b may be evaluated with linear regression. Procedures of linearization to evaluate the slope and the intercept of some common simple models may be summarized as

Model	Linearization Procedure
$y = a/x + b$	Plot y vs. $1/x$, slope = a, intercept = b
$y = \dfrac{x}{a + bx}$	Rewrite the equation as $1/y = a/x + b$, plot $1/y$ vs. $1/x$, slope = a, intercept = b
$y = be^{ax}$	Rewrite the equation as $\ln y = \ln b + ax$, plot $\ln y$ vs. x, slope = a, intercept = $\ln b$

In a plot of y vs. x if almost all the data points fall on the line, values of parameters a and b may be easily determined from the slope and the intercept of the graph. When the data points are scattered, it may be possible to draw many lines passing through them. The *best line* is the one with the smallest sum of squares of difference defined as $\sum d_i^2 = \sum_{i=1}^{n}\left[y_i - \left(ax_i + b\right)\right]^2$, where $ax_i + b$ is the value of parameter y predicted by Equation 1.2 and y_i is the experimentally determined value of y corresponding to x_i. The term $d_i = y_i - (ax_i + b)$ is the difference between the experimentally determined and predicted values of y at the point x_i. This difference might be either negative or positive. Regardless of the sign, the magnitude of the difference describes the deviation of the line from the data. When the differences are added up over the entire data set, the negative and positive differences may cancel each other and cause an erroneous conclusion. Working with the squares of the differences eliminates the cause of the erroneous conclusion. The minimum value of the sum of the squares difference is obtained with the following best line parameters:

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} (x_i)^2\right)}, \tag{1.3}$$

$$b = \frac{\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i\right) - \left(\sum_{i=1}^{n} x_i y_i\right)\left(\sum_{i=1}^{n} x_i y_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} (x_i)^2\right)}. \tag{1.4}$$

Although Equations 1.3 and 1.4 give the values of a and b of the best fitting line, it does not mean that the best fitting line will always describe the correct relation between variables y and x. It is always possible that the relation between y and x may not be actually described with Equation 1.2. The goodness of the best fitting line to describe n sets of data points may be evaluated with a standard error of estimate:

$$s_e = \sqrt{\frac{\sum d^2}{n}}. \tag{1.5}$$

The standard error of estimate is a measure of the degree of association between the data points and the regression line. The larger the standard error of estimate, the larger the scatter of the data points around the fitted line. If all the data points are scattered by approximately normal distribution around the regression line, 99.73% of all the data points are scattered within the range of $\pm 3s_e$.

The correlation coefficient of the data and the best line is

$$r = \sqrt{1 - \frac{s_e^2}{s_y^2}}, \tag{1.6}$$

where s_y is the standard deviation of values of y around its mean value:

$$s_y = \sqrt{\frac{\sum y^2}{n} - \left(\frac{\sum y}{n}\right)^2}. \tag{1.7}$$

The term S_e^2/S_y^2 is actually a ratio of the variance of the data points around the fitted line to their variance around the mean value of y. Having s_e is much smaller than s_y implies that the scatter of the data points around the fitted line is almost negligible when compared to the scatter around the average value of y. At the limiting condition of this case the ratio $S_e^2/S_y^2 = 0$, thus $r = 1$ implying a perfect fit. Having s_e the same as s_y implies that the scatter of the data points around the fitted line is almost the same as the scatter around the average value of y, implying that the fitted line does not represent the data. At the limiting condition of this case the ratio $S_e^2/S_y^2 = 1$, thus $r = 0$ implying no fit. Values of the correlation coefficient r vary between the limits 0 and 1. The correlation coefficient may also be defined as

$$r = \frac{s_{xy}}{s_x s_y}, \tag{1.8}$$

where s_x is the standard deviation of values of x around their mean value:

$$s_x = \sqrt{\frac{\sum x^2}{n} - \left(\frac{\sum x}{n}\right)^2}, \tag{1.9}$$

and s_{xy} is defined as

$$s_{xy} = \frac{\sum xy}{n} - \left(\frac{\sum x}{n}\right)\left(\frac{\sum y}{n}\right). \tag{1.10}$$

Equation 1.8 also gives the sign of the correlation, whereas Equation 1.6 gives its absolute value only. Having $r = -1$ implies that there is perfect correlation between the two parameters, but while one of them is increasing the other one is decreasing. The square of the correlation coefficient, r^2, may be preferred to describe the correlation between the variables, since it magnifies the deviation from the perfect correlation when r is close to 1.

Parameters a, b, and r as described by Equations 1.3, 1.4, and 1.8, respectively, may be easily calculated with simple scientific calculators containing the related built-in functions by entering the data only. Common spreadsheet computer software also serves the same purpose. Nonlinear regression is more sophisticated, but may be easily done with the common commercially available statistics and mathematics software.

Example 1.2: Survival Kinetics of the Freeze-Dried Lactic Acid Bacteria

The number of the viable microorganisms per gram of a freeze-dried preparation is the major quality factor of the freeze-dried cultures. Variation of the number of the freeze-dried viable microorganisms with time in storage may be expressed as

$$\frac{dx}{dt} = -k_d x. \tag{E.1.2.1}$$

Equation E.1.2.1 may be integrated as

$$\log x = \log x_0 - \frac{k_d}{2.303} t, \tag{E.1.2.2}$$

where x_0 is the number of the viable microorganisms at $t = 0$. The following counts (number of viable microorganisms/ml) of the freeze-dried lactic acid starter culture microorganisms were reported by Alaeddinoglu, Guven, and Özilgen (1988):

***t*(days)**	**0**	**15**	**30**	**45**	**60**	**90**
log*x*	7.8	7.0	6.2	5.4	4.8	4.4

a. Calculate values of the constants $\log x_0$ and k_d and the correlation coefficient.

Solution: We may change the notation as $y = \log x$, $b = \log x_0$, $a = -k_d/2.303$, then the above equation may be expressed as $y = ax + b$. It may also be shown that $\Sigma x_i = 240$, $\Sigma y_i = 35.6$, $\Sigma x_i\, y_i = 1218$, $\Sigma x_i^2 = 14850$, $(\Sigma x_i^2) = 57{,}600$, and also $n = 6$. After filling them into Equations 1.3 and 1.4, $a = -0.039$, $b = 7.5$. Since $a = -k_d/2.303$ and $b = \log x_0$, we may calculate $k_d = 0.09$ day^{-1} and $\log x_0 = 7.5$.

The fitted equation is $\log x_{reg} = 7.5 - 0.039t$. The squares of the difference between log*x* and $\log x_{reg}$ may be calculated as

***t*(days)**	**log*x***	**$\log x_{reg}$**	**d^2**
0	7.8	7.5	0.09
15	7.0	6.9	0.01
30	6.2	6.3	0.01
45	5.4	5.7	0.09
60	4.8	5.1	0.09
90	4.4	4.0	0.16
			$\Sigma d^2 = 0.4$

$$s_e = \sqrt{\frac{\sum d^2}{n}} = 0.27,\ s_y = \sqrt{\frac{\sum y^2}{n} - \left(\frac{\sum y}{n}\right)^2} = 1.2, \quad \text{and} \quad r = \sqrt{1 - \frac{s_e^2}{s_y^2}} = 0.95,$$

Equation 1.6 is used here and implied an almost perfect fit. Equation 1.8 may also be used to calculate the correlation coefficient as

$$r = \frac{s_{xy}}{s_x s_y} = -0.97,$$

where

$$s_x = \sqrt{\frac{\sum x^2}{n} - \left(\frac{\sum x}{n}\right)^2} = 29.6 \quad \text{and} \quad s_{xy} = \frac{\sum xy}{n} - \left(\frac{\sum x}{n}\right)\left(\frac{\sum y}{n}\right) = -34.3.$$

Equation 1.8 implies a perfect agreement with the previous result. It gives additional information that there is negative correlation between log*x* and *t*; that is, the log number of the viable

freeze-dried microorganisms decreases as time passes by. Comparison of the best line with the experimental data (symbols) is shown in Figure E.1.2.

b. How many microorganisms/ml will remain viable on the 75th day of storage? State with 99.73% probability.

Solution: When $t = 75$ we may calculate $\log x_{reg} = 7.5 - 0.39t$. Since with $p = 0.9973$ experimental data are scattered within $\pm 3s_e$ range of the best line, the confidence limits are $\log x_{reg} - 3s_e \le \log x \le \log x_{reg} + 3s_e$ after substituting the numbers $3.75 \le \log x \le 5.37$.

MATLAB® code E.1.2 carries out the same computations.

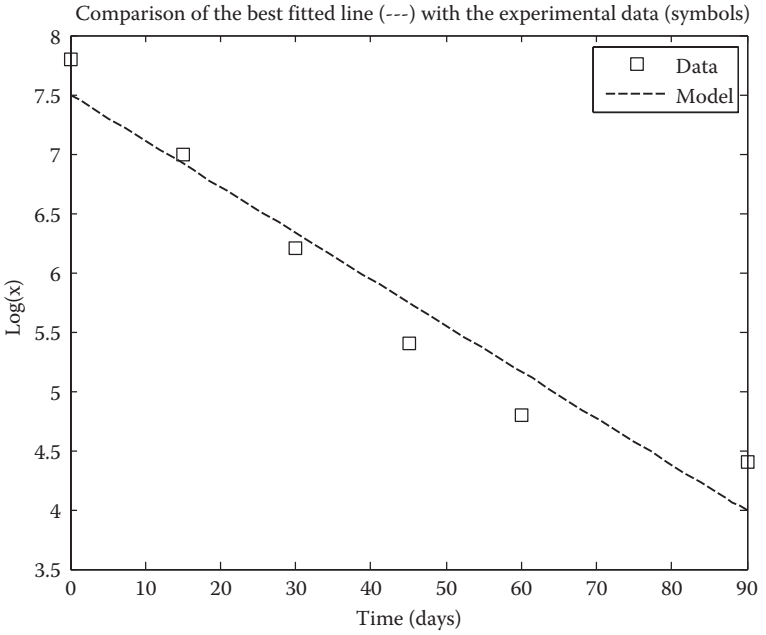


FIGURE E.1.2
Comparison of the model with the experimental data. There is a perfect agreement between the model and the data, with $r = 0.9687$ and $s_e = 0.2973$.

MATLAB® CODE E.1.2

Command Window:

```
clear all
close all
format compact
global kDeath

kDeath = 0.039; % death rate constant
xi = [0 15 30 45 60 90];% times the data collected (day)
yi = [7.8 7.0 6.2 5.4 4.8 4.4]; % log(x)
plot(xi,yi,'s');
sum_xi = sum(xi);
sum_yi = sum(yi);
n = length(xi);
xi_times_yi = 0;
```

```
xi_square = 0;
yi_square = 0;
for i = 1:n
    xi_times_yi = xi_times_yi + (xi(i)*yi(i));
    xi_square = xi_square + (xi(i)^2);
    yi_square = yi_square + (yi(i)^2);
end
sum_of_Xi_square = (sum(xi))^2;
a = ((n * (xi_times_yi)) - (sum_xi * sum_yi)) / ((n * xi_square)
- (sum_of_Xi_square))
b = ((xi_square * sum_yi) - (xi_times_yi * sum_xi)) / ((n * xi_
square) - (sum_of_Xi_square))
kd = -a * 2.303
logx0 = b
logxreg = (b + a.*xi)
d_square = ((yi - logxreg).^2)
sum_d_square = sum(d_square)
s_e = sqrt(sum_d_square/n)                % standard error of estimate
s_y = sqrt((yi_square/n)-((sum_yi/n)^2)) %standard deviation of
values of y
r = sqrt(1-((s_e^2)/(s_y^2)))             % correlation coefficient
hold all;
box on;
[xi,x] = ode45('microbialdeath',90,exp(7.5));
plot(xi,log(x),'--');
xlabel('Time(days)')
ylabel('log(x)')
title('comparison of the best fitted line (---) with the
experimental data (symbols)')
legend('data','model')
```

M-File:

```
function y = microbialdeath(t,x)
global kDeath
y = -kDeath*x;
```

When we run the code the following lines and Figure E.1.2 will appear on the screen:

```
a =
   -0.0392
b =
    7.5029
kd =
    0.0904
logx0 =
    7.5029
logxreg =
    7.5029   6.9143   6.3257   5.7371   5.1486   3.9714
d_square =
    0.0883   0.0073   0.0158   0.1137   0.1215   0.1837
sum_d_square =
    0.5303
s_e =
    0.2973
```

```
s_y =
    1.1981
r =
    0.9687
```

Example 1.3: Comparison Between Two Models: Death Kinetics of Microorganisms in Dough

The colony counts of *Saccharomyces cerevisiae* after 60 minutes of leavening for sour dough with an inoculum initially containing 80% *S. cerevisiae* and 20% *Lactobacillus plantarum* were (Yöndem, Özilgen, and Bozoglu 1992)

***t*(min)**	**60**	**90**	**120**	**150**	**180**	**210**
x(cfu/g)	4.5×10^6	3.85×10^6	3.45×10^6	3.3×10^6	3.1×10^6	3.2×10^6

Suggest a mathematical model to describe the death of the yeast.

Solution: The process is the death of the microorganisms during constant temperature pasteurization. The model is expected to predict (output) the colony counts of the surviving microorganisms with time input. It might be assumed that constant fractions of the viable microorganisms die at a constant time interval with the mathematical formulation:

$$\frac{dx}{dt} = -kx. \tag{E.1.3.1}$$

The model assumes that the microorganisms are equally labile when subject to heat treatment and do not affect each other. This is the most common microbial death rate expression in the literature (Chapter 3). The form of the model has been based on assumptions; therefore, it is an empirical model. The solution of the equation is

$$x = x_0 e^{-k} \tag{E.1.3.2}$$

and may be linearized as

$$\ln(x) = \ln(x_0) - kt. \tag{E.1.3.3}$$

MATLAB® code E.1.3.a compares Equation E.1.3.3 with the data.

Disagreement of the model with the data requires us to repeat all the steps of modeling starting from the definition of the system. The process seems like it is defined properly, but the simplifying assumptions may not be correct; that is, microbial death may slow down as microbial concentration approaches a highly resistant small fraction. Such an observation may be caused by nonuniform resistance of the microbial population to death, or dead microorganisms (or their constituents) may protect or increase the resistance of the surviving microorganisms. The model may be revised as

$$\frac{dx}{dt} = -k(x - x_m), \tag{E.1.3.4}$$

where x_m represents the finally attainable highly resistant microbial concentration. The new expression may be solved:

$$x = (x_0 - x_m)e^{-kt} + x_m \tag{E.1.3.5}$$

MATLAB® CODE E.1.3.a

Command Window:

```
clear all
close all
format compact

global k % make k available to all the m-functions where the word
global is written

k = 0.003; % death rate constant

[t,x] = ode45('deathkinetics1',[60 210], 4501854.5);
plot(t,log(x))
t = [60 90 120 150 180 210];
x = [4.5e006; 3.85e006; 3.45e006; 3.3e006; 3.1e006; 3.2e006];
% calculate x at t = 60, t = 90,...., t = 210 min with x0 = 4501854.5
[t,x2] = ode45('deathkinetics1', [60 90 120 150 180 210], 4501854.5);
d = log(x2)-log(x);
dsqr = d.^2;
sumdsqr = sum(dsqr);
% compute the standard error
se = sqrt(sumdsqr/6)
sy = std(log(x));
sx = std(t);
% compute the correlation coefficient with [1.6]
r1 = sqrt(1-se^2/sy^2)
sxy = sum(log(x).*t)/6-sum(log(x))/6*sum(t)/6;
% compute the correlation coefficient with [1.8]
r2 = sxy/(sx*sy)
hold on,plot(t,log(x),'x')
xlabel('t(min)'),ylabel('ln x')
```

M-File:

```
function y = deathkinetics1(t,x)
global k
% This function models microbial death in analogy with first order
irreversible unimolecular chemical reaction
y = -k*x;
```

When we run the code, the following lines and Figure E.1.3.1 will appear on the screen:

```
se =
    0.0651
r1 =
    0.8852
r2 =
   -0.7632
```

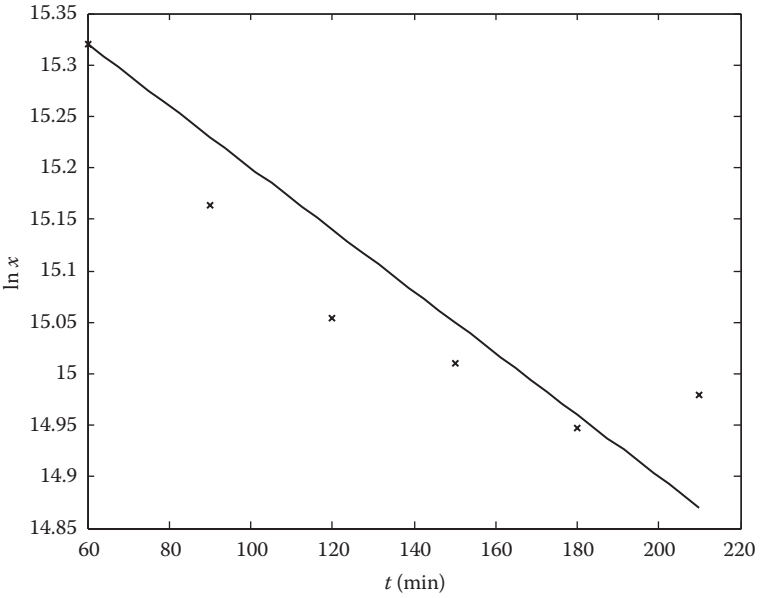


FIGURE E.1.3.1
Comparison of the model (-------) with the experimental data (x). Equation E.1.3.1 does not agree with the data ($r_1 = 0.89$, $r_2 = -0.76$, and $s_e = 0.0651$).

and linearized:

$$\ln(x - x_m) = \ln(x_0 - x_m) - kt, \tag{E.1.3.6}$$

when $x_m = 3.15 \times 10^6$ cfu/g constants of the linear equation may be evaluated with the linear regression as $\ln(x - x_m) = 15.64 - 0.025t$, with correlation coefficient $r = -0.999$ implying that $\ln(x_0 - x_m) = 15.64$ and $k = 0.025$ min^{-1}. MATLAB® code E.1.3.b compares Equation E.1.3.6 with the data.

Example 1.4: Kinetics of Galactose Oxidase Production
The logistic model (Chapter 3) is frequently used to simulate microbial growth:

$$\frac{dx}{dt} = \mu x\left(1 - \frac{x}{x_{\max}}\right), \tag{E.1.4.1}$$

where μ is the initial specific growth rate and $x_{\max}$ is the maximum attainable value of x. The logistic equation is an empirical model, because it simulates the data without any theoretical basis. It is based only on experimental observations: When $x << x_{max}$, the term in parenthesis is almost one and neglected, then the equation simulates the exponential growth ($dx/dt = \mu x$), and when x is comparable with $x_{\max}$, the term in parenthesis becomes important and simulates the inhibitory effect of overcrowding on microbial growth. When $x = x_{\max}$, the term in parenthesis becomes zero, then the equation will predict no growth ($dx/dt = 0$). The logistic equation may be integrated as

$$x = \frac{x_0 e^{\mu t}}{1 - \frac{x_0}{x_{\max}}(1 - e^{\mu t})}. \tag{E.1.4.2}$$

MATLAB® CODE E.1.3.b

Command Window:

```
clear all
close all
format compact

% enter the data
tData = [60 90 120 150 180 210]; % times when the data were recorded
xData = [4.5e006 3.85e006 3.45e006 3.3e006 3.1e006 3.2e006];
xm = 3.15e006; % minimum attainable value of x
kDeath = 0.025; % death rate constant

global xm kDeath; % make xm and kDeath be available to all the
m-functions where the word global is written

plot(tData,log(xData),'*'); hold on % plot the data
xlabel('t(min)'),ylabel('ln x')
lsline % least squares line
[t,x] = ode45('deathkinetics2',[60 210], 4501854.5); % compute values
of x as a function of t
plot(t,log(x), ':') % plot the model

% evaluate the standard error and the correlation coefficients
xminusxm = xData-xm;
[t,x2] = ode45('deathkinetics2', [60 90 120 150 180 210], 4501854.5);
xminusxm2 = x2-xm;

for k = 1:1:size(x2)
    dsqr(k) = (xminusxm2(k)-xminusxm(k))^2;
    prod(k) = (xminusxm2(k))*tData(k);
end

sumdsqr = sum(dsqr);
se = sqrt(sumdsqr/6)
sy = std(xminusxm2) ;
sx = std(tData) ;
r1 = sqrt(1-((se^2)/(sy^2)))
sxy = std(prod);
r2 = sxy/(sx*sy)
legend('data','least squares line','model', 'location', 'NorthEast' )
```

M-File:

```
function y = deathkinetics2(t,x)
% This function models logistic microbial death, i.e, a minimum
microbial population described as xm survives
global xm kDeath
y = (-kDeath)*(x-xm); % death rate model
```

When we will run the code the following lines and Figure 1.3.2 will appear on the screen:

```
se =
    5.4663e+004
```

```
r1 =
    0.9942
r2 =
    1.0125
```

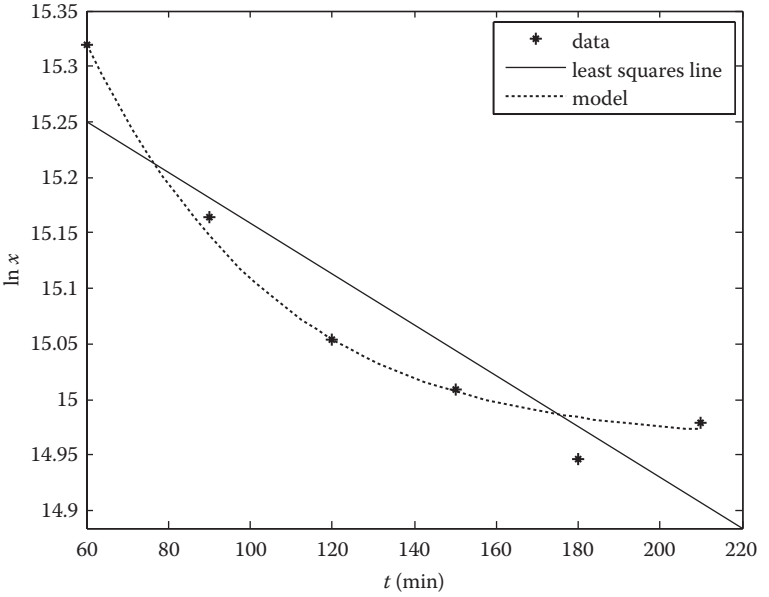


FIGURE E.1.3.2
Comparison of Equation E.1.3.4 with the experimental data. Parameters $s_e = 5.4663e+004$ (equals to 1.2% of the initial microbial population), $r_1 = 0.9942$ and $r_2 = 1.0125$ indicate almost perfect agreement between the model and the data. It is seen clearly that the least squares line goes far away from the data points.

MATLAB® code E.1.4.a compares the model Equation E.1.4.2 with the data.

Galactose oxidase production was simulated with the Luedeking–Piret model (details are explained in Chapter 3; Ogel and Özilgen 1995):

$$\frac{dc_P}{dt} = \alpha x + \beta \frac{dx}{dt}, \tag{E.1.4.3}$$

where α and β are constants. The term αx represents the galactose oxidase production by the microorganisms regardless of their growth; $\beta dx/dt$ represents the additional enzyme production in proportion with the growth rate. This is an empirical equation, because it simply relates the experimental observations without any theoretical basis. Integrated and differential forms of the logistic equation were used to simulate biomass concentration x and dx/dt, respectively, and the Luedeking–Piret model was integrated as

$$c_p = c_{p0} + \alpha A + \beta B, \tag{E.1.4.4}$$

where c_{p0} is the amount of the product in the fermentor when $t = 0$ and

$$A = \frac{x_{max}}{m} \ln\left[1 - \frac{x_0}{x_{max}}(1 - e^{\mu t})\right] \quad \text{and} \quad B = x_0\left(\left(e^{\mu t} \Big/ \left(1 - \frac{x_0}{x_{max}}\left(1 - e^{\mu t}\right)\right)\right) - 1\right).$$

MATLAB code E.1.4.b compares the model Equation E.1.4.3 with the data.

MATLAB® CODE E.1.4.a

Command Window:

```
clear all
close all
format compact
global mu xmax

% enter the constants
mu = 0.037;
xmax = 0.6;

% enter the data
tData1 = [25 48 72 97 132];
tData2 = [25 48 72 97 120 132];
a = [0.15 0.25 0.38 0.45 0.55];
b = [0.09 0.26 0.4 0.48 0.57 0.58];

plot(tData1,a,'x',tData2,b,'o'); hold on, % plot the data
xlabel('t (min)'),ylabel('gbiomass/L')
x1 = [0.15; 0.25; 0.38; 0.45; 0.55];
x2 = [0.09; 0.26; 0.4; 0.48; 0.57; 0.58];

[t,x] = ode45('logisticgrowth',150,0.068); % compute the model x
values
plot(t,x) % plot the model
legend('data','data','model', 'location', 'SouthEast' )
[t,x3] = ode45('logisticgrowth', [25 48 72 97 132], 0.068); % compute
the x values corresponding to tData1 (notice dimensions of t1 and t2
are not the same)
[t,x4] = ode45('logisticgrowth', [25 48 72 97 120 132], 0.068); %
compute the x values corresponding to tData2
dsqr1 = (x1-x3).^2;
dsqr2 = (x2-x4).^2;
sumdsqr = sum(dsqr1) + sum(dsqr2);
se = sqrt(sumdsqr/13)
```

M-File:

```
function y = logisticgrowth (t,x)
global mu xmax
y = mu*x*(1-(x/xmax));
```

When we run the code, the following lines and Figure E.1.4.1 will appear on the screen:

```
se =
    0.0858
```

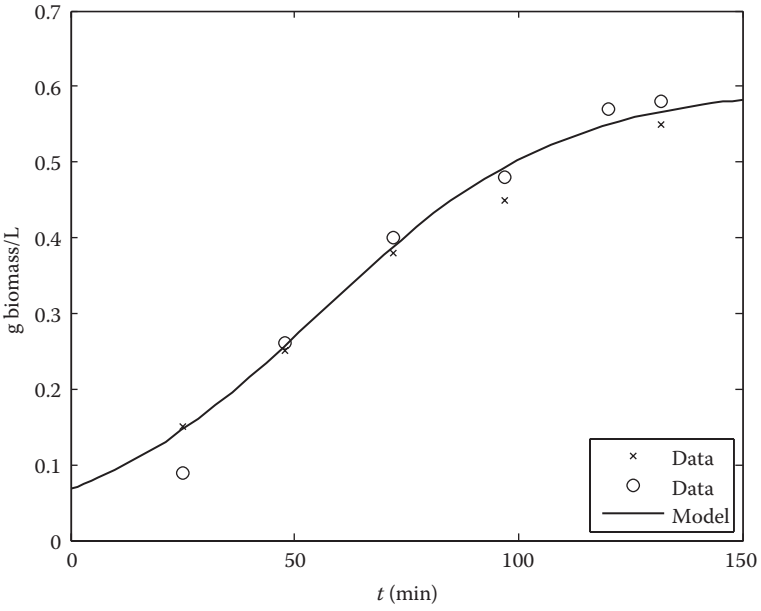


FIGURE E.1.4.1
Comparison of the biomass growth model with the data. (From Shatzman, A. R. and Kosman, D. J., *Journal of Bacteriology,* 130, 455–63, 1977.)

MATLAB® CODE E.1.4.b

Command Window:

```
clear all
close all

[t,x] = ode45('LuedekingPiret', 150, [0.068 0]);
plot(t, x( :,2))
xlabel('t(h)');
ylabel('Cp(Eu/L)');
c = [0 24 48 72 96 120];
d = [0 2400 5600 12000 14800 16400];
hold on, plot(c, d, 'r*')
```

M-File:

```
function dx = LuedekingPiret(t,x);
% This function models product formation with Luedeking-Piret model
mu = 0.037;
xmax = 0.6;
alpha = 43;
beta = 32000;
dx1 = mu*x(1)*(1-(x(1)/xmax));
dx2 = alpha*x(1) + beta*dx1;
dx = [dx1; dx2];
```

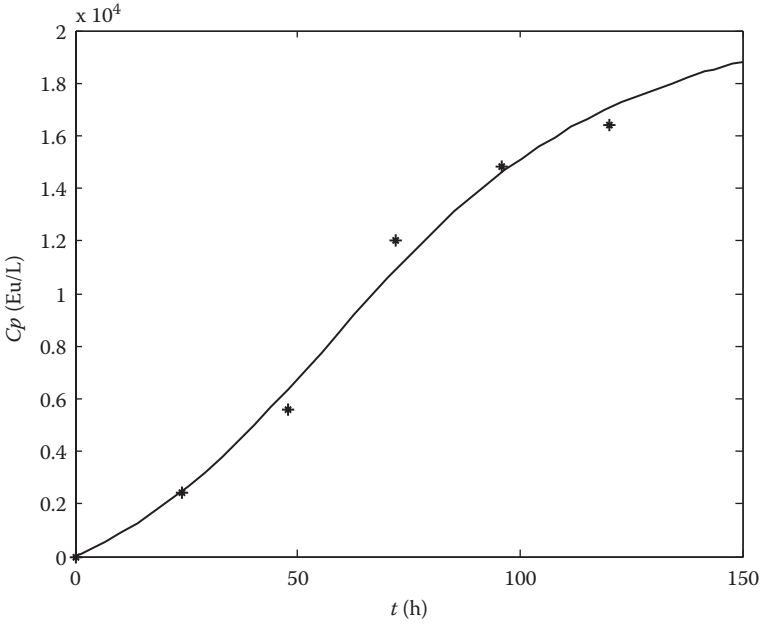


FIGURE E.1.4.2
Comparison of the product formation model (---) with the data (x). (Model parameters: $\alpha = 43$ Eu/g biomass h^{-1} and $\beta = 32 \times 10^3$ Eu/g biomass were taken from Ogel, B. Z. and Özilgen, M. *Enzyme and Microbial Technology*, 17, 870–76, 1995.)

Example 1.5: Kinetics of Lipid Oxidation in Foods

Lipid oxidation is one of the major reasons for spoilage of the lipid foods and occurs stagewise:

Initiation: $RH \rightarrow R\cdot + H\cdot$
Propagation: $R\cdot + O_2 \rightarrow RO_2\cdot$
$RO_2\cdot + RH \rightarrow RO_2H + R\cdot$
Termination: $R\cdot + R\cdot \rightarrow RR$
$R\cdot + RO_2\cdot \rightarrow RO_2R$
$RO_2\cdot + RO_2\cdot \rightarrow RO_2R + O_2$,

where RH represents the lipids, R·, H·, and RO_2· the free radicals, whereas RO_2H and RO_2R are the oxidation products. Parameter R actually represents different chemical species and too many reactions, involving many intermediary products, occurring simultaneously during lipid oxidation. Such a complex reaction mechanism makes it impossible to monitor the concentration of all the individual chemicals and it is difficult to use the conventional chemical kinetic rate expressions for modeling.

There is an analogy between lipid oxidation and microbial growth: both of them have initiation, propagation, and termination phases. Equation E.1.3.1 simulates microbial growth successfully. Microbial growth, like lipid oxidation, occurs through a large number of complex reactions. Therefore a logistic equation may be used to simulate lipid oxidation:

$$\frac{dc}{dt} = kc\left[1 - \frac{c}{c_{max}}\right], \tag{E.1.5.1}$$

where c is the total amount of the oxidation products, dc/dt is the lipid oxidation rate, c_{max} is the total amount of the lipids available for oxidation, t is time, and k is the reaction rate constant. As the reaction rate constant k increases c_{max} is attained faster. At the beginning of the lipid oxidation process (i.e., when $c \ll c_{max}$) the term $(1 - c/c_{max})$ may be neglected and Equation E.1.5.1 becomes

$$\frac{dc}{dt} = kc.$$

At this stage of the process, the termination reactions do not occur and the free radicals accumulate in the food and the reaction rate seems to be increasing with the accumulation rate of the free radicals. In the termination phase, free radicals react with each other and the term $(1 - c/c_{max})$ becomes important. When $c = c_{max}$, the term $(1 - c/c_{max})$ becomes zero and Equation E.1.5.1 estimates the end of the lipid oxidation process. The solution to Equation E.1.5.1 is

$$c = \frac{c_0 e^{kt}}{1 - \frac{c_0}{c_{max}}(1 - e^{kt})}, \tag{E.1.5.2}$$

where c_0 is the concentration of the oxidized products at the beginning of oxidation. MATLAB® code E.1.5 carries out a sample set of simulations and compares the model with the data as shown in Figure E.1.5.

MATLAB® CODE E.1.5

Command Window:

```
clear all
close all

[t,c] = ode45('lipidoxidation1',20,2.61);
plot(t,c)
xlabel('t (days)');
ylabel('c (TBA)');
[t,c] = ode45('lipidoxidation2',20,0.73);
hold on, plot(t,c)
s = [0 2 6.4 10.8 14];
a = [2.2 6 10.4 14.2 14.6];
b = [0.8 1.92 3.96 4 5.4];
hold on, plot(s,a,'o')
hold on, plot(s,b,'x')
```

M-File:

```
function f = lipidoxidation1(t,c);
k = 0.38;
cmax = 14.9;
f = k*c*(1-(c/cmax));
```

M-File:

```
function f = lipidoxidation2(t,c);
k = 0.54;
cmax = 5;
f = k*c*(1-(c/cmax));
```

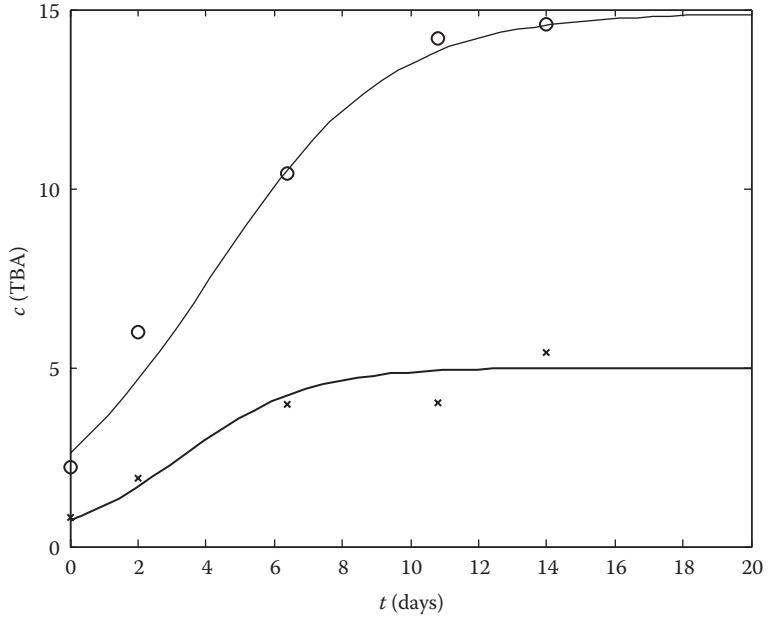


FIGURE E.1.5
Comparison of the model (lines) with the data (symbols, o:pH 4.70, $k = 0.38$, $c_{max} = 14.9$ TBA, $c_0 = 2.61$ TBA; x: pH 6.25, $k = 0.54$, $c_{max} = 5$ TBA, $c_0 = 0.73$ TBA) during lipid oxidation in ground raw poultry meat. (From Özilgen, S. and Özilgen, M., *Journal of Food Science*, 55, 498–502, 536, 1990.)

Example 1.6: Analogy Between Cake Filtration and Ultrafiltration Processes

In a cake filtration process (Chapter 4) the total pressure drop through the filter (ΔP) is

$$\Delta P = \Delta P_c + \Delta P_m, \tag{E.1.6.1}$$

where ΔP_c and ΔP_m are the pressure drops through the cake and the medium, respectively, and expressed theoretically as

$$\Delta P_c = \frac{\alpha \mu c V}{A^2} \frac{dV}{dt}, \tag{E.1.6.2}$$

and

$$\Delta P_m = \frac{R_m \mu}{A} \frac{dV}{dt}, \tag{E.1.6.3}$$

where α = specific resistance of the cake, μ = viscosity of the filtrate, c = mass of the solids per unit volume of the feed, A = filter area, V = volume of the filtrate, t = time, and R_m = filter medium resistance. Equations E.1.6.2 and E.1.6.3 may be substituted in Equation E.1.6.1 to obtain the Sperry's Equation:

$$\frac{dV}{dt} = \frac{A\Delta P}{\mu}\frac{1}{R_m + \alpha c V/A}. \tag{E.1.6.4}$$

Equation E.1.6.4 has been based on theoretical considerations; therefore, it may be referred to as a phenomenological model. Foods are usually highly complex systems. Theoretical expressions are frequently found inefficient to simulate the processes involving foods and biological materials. De LaGarza and Boulton (1984) modified Equation E.1.6.4 to model wine filtrations:

$$\frac{dV}{dt} = \frac{A\Delta P}{\mu}\frac{1}{R_m + \alpha c (V/A)^n} \tag{E.1.6.5}$$

and

$$\frac{dV}{dt} = \frac{A\Delta P}{\mu}\frac{1}{R_m \exp(kV/A)}. \tag{E.1.6.6}$$

Equation E.1.6.6 may be integrated as

$$V = \frac{A}{\kappa}\left[\ln\left(\frac{\kappa\Delta P}{\mu R_m}t + 1\right)\right]. \tag{E.1.6.7}$$

MATLAB® code E.1.6.a compares Equation E.1.6.7 with the experimental data as shown in Figure E.1.6.1.

MATLAB® CODE E.1.6.a

Command Window:

```
clear all
close all

% introduce the data
tData = [0 30 55 120 190 250 290 300 340 415 500 700 900 1000];
vData = [0 80 120 162 190 210 220 222 230 245 255 277 295 300];
% introduce the constants
A = 30.2; % filtration area
deltaP = 6.5e4; % pressure difference
kappa = 0.447; % cake resistance coefficient
Rm = 2.05e8; % filter medium resistance
mu = 1.65e-3; % viscosity

% plot the data and insert labels
plot(tData, vData, '*'); hold on
xlabel('t (s)')
ylabel('V (cm3)')
```

```
% compute the filtrate accumulation rate
for t = 0:1000;
v(t + 1) = A*(log(((kappa*deltaP)/(mu*Rm))*t + 1))/kappa;
time(t + 1) = t;
end;

% plot the model and insert the legends
plot(time,v)
legend('data','model', 'Location','SouthEast')
```

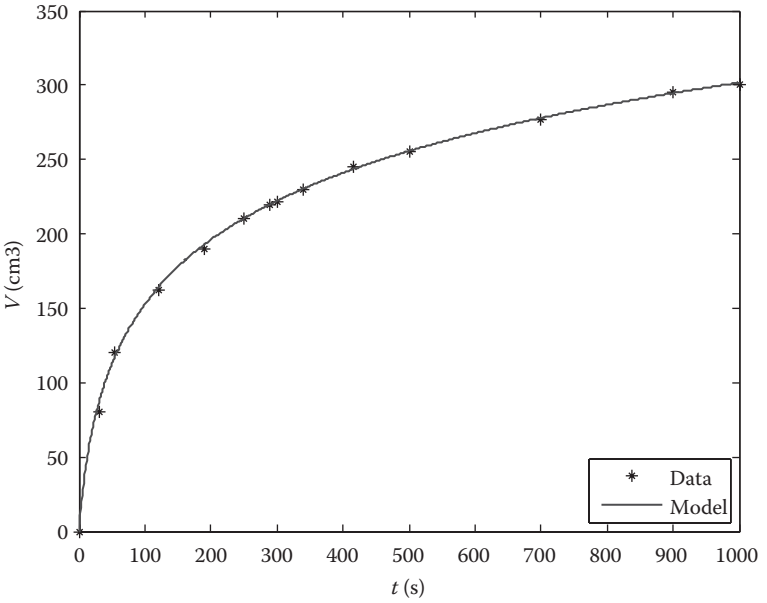


FIGURE E.1.6.1
Comparison of Equation E.1.6.7 (---) with the experimental data. (From Bayindirli, L., Ungan, S., and Özilgen, M., *Journal of Food Science,* 54, 1003–6, 1989.)

In an ultrafiltration process, a fraction of the solute is permitted to pass through the membrane and the remaining solutes are rejected. The use of batch ultrafiltration is limited to laboratory scale separations only due to an accumulation of the rejected particles on the membrane. Although Equations E.1.6.4, E.1.6.5, and E.1.6.6] are suggested for cake filtrations only, they may be used to simulate batch ultrafiltration based on the analogy between the processes: Resistance to flow by the membrane and the rejected solids in an ultrafiltration process is analogous to the medium and the cake resistance in cake filtration. MATLAB code E.1.6.b exemplifies the use of Equation E.1.6.5 with $n = 2$ to the model sequential batch ultrafiltration of red beet extract.

MATLAB® CODE E.1.6.b

Command Window:

```
clear all
close all

% introduce the data
tData = [0 45 90 110 130 180 205 230 260 280 310 340 370 410 430 470
510];
vData = [0 80e-6 150e-6 210e-6 280e-6 340e-6 390e-6 440e-6 500e-6
550e-6 600e-6 640e-6 700e-6 750e-6 800e-6 830e-6 900e-6];

% plot the data
plot(tData, vData, '*'); hold on;
% put the labels
xlabel('t (min)'); hold on
ylabel('V (m3)'); hold on

% compute the filtrate accumulation rate
[t,V] = ode45('DeLaGarzaBoulton', 0:1:600, 0);

% plot the model and insert the legend
plot(t,V)
legend('data','model', 'Location','SouthEast')
```

M-File :

```
function dV = DeLaGarzaBoulton(t,V)
A = 30.2e-4;
deltaP = 4e5;
kappaA2 = 11.94e18; % kappa/Asquare
Rm = 6.29e13;
mu = 1e-5;
% DeLaGarza Boulton Model
dV = (deltaP*A/mu)*(1/(Rm + kappaA2*(V^2)));
```

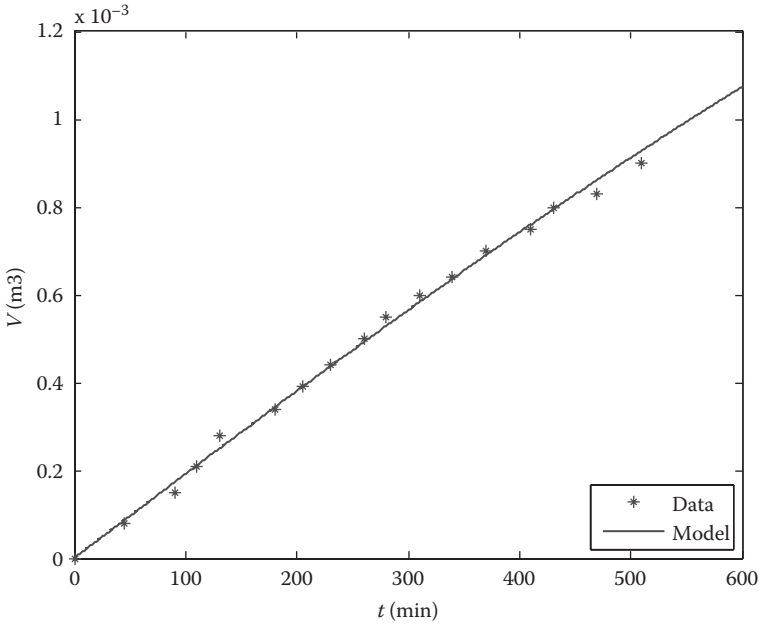


FIGURE E.1.6.2
Comparison of Equation E.1.6.5 ($n = 2$) (---) with the experimental data (cellulose nitrate isotropic membrane, 50,000 daltons molecular weight cutoff. (From Bayindirli, A., Yildiz, F., and Özilgen, M., *Journal of Food Science*, 53, 1418–22, 1988.)

Questions for Discussion and Problems

A. What is Process Modeling?
   1. Explain the philosophical reasoning of mathematical modeling.
   2. What are the properties of a good mathematical model?
   3. What are theoretical, analogy, and empirical modeling?
   4. What is the significance of the 80%–20% rule in process modeling?

B. Empirical Models and Linear Regression
   1. The active component of an artificial flavor decreases during the storage period as indicated by the following data

Time (months)	**1**	**2**	**4**	**6**	**8**	**12**	**18**	**24**	**28**	**36**	**40**	**43**
Active component (IU)	500	497	435	488	486	474	466	450	440	420	425	413

It is believed that the loss of an active ingredient in storage is a linear function of time

$$(\text{remaining active component, } IU) = \alpha - \beta t.$$

   i. Determine the constants α and β.
   ii. Determine the numerical value correlation coefficient and explain what it means to you.
   iii. Determine the numerical value of the standard error and explain what it means to you.

2. The active number of the viable microorganisms per gram of a freeze-dried preparation is the major quality factor of the freeze-dried cultures. Variation in the number of the freeze-dried viable microorganisms with the time in storage may be expressed as

$$\frac{dx}{dt} = -k_d x. \tag{Q.1.B.2.1}$$

Equation 3.67 may be integrated as

$$\log x = \log x_0 - \frac{k_d}{2.303} t, \tag{Q.1.B.2.2}$$

where x_0 is the number of the viable microorganisms at $t = 0$. The following counts (number of viable microorganisms/ml) of the freeze-dried lactic acid starter culture microorganisms were reported by Alaeddinoglu, Guven, and Özilgen (1988):

***t*(days)**	**0**	**15**	**30**	**45**	**60**	**90**
log*x*	7.8	7.0	6.2	5.4	4.8	4.4

A. Calculate values of the constants $\log x_0$ and k_d and the correlation coefficient.
B. How many microorganisms/ml will remain viable on the 75th day of storage? State with 99.73% probability.

References

Alaeddinoglu, G., A. Guven, and M. Özilgen. "Activity loss kinetics of freeze-dried lactic acid starter cultures." *Enzyme and Microbial Technology* 11 (1988): 765–69.

Bayindirli, A., F. Yildiz, and M. Özilgen. "Modeling of sequential batch ultrafiltration of red beet extract." *Journal of Food Science* 53 (1988): 1418–22.

Bayindirli, L., S. Ungan, and M. Özilgen. "Modeling of apple juice filtrations." *Journal of Food Science* 54 (1989): 1003–6.

De LaGarza, F., and R. Boulton. "The modeling of wine filtrations." *American Journal of Enology and Viticulture* 35 (1984): 189–95.

Glasscock, D. A., and J. C. Hale. "Process simulation: The art and science of modeling." *Chemical Engineering* 101, no. 11 (1994): 82–89.

Luyben, W. L. *Process Modeling, Simulation and Control for Chemical Engineers.* Singapore: McGraw-Hill, 1990.

MATLAB® 7 Getting Started Guide. Natick, MA: The MathWorks, 2008.

Meyer, W. J. *Concepts of Mathematical Modeling*. Singapore: McGraw-Hill, 1985.

Oakland, J. S., and R. F. Followell. *Statistical Process Control.* 2nd ed. Oxford: Butterworth-Heinemann, 1995.

Ogel, B. Z., and M. Özilgen. "Regulation and kinetic modeling of galactose oxidase secretion." *Enzyme and Microbial Technology* 17 (1995): 870–76.

Özilgen, S., and M. Özilgen. "Kinetic model of lipid oxidation in foods." *Journal of Food Science* 55 (1990): 498–502, 536.

Rand, W. M. "Development and analysis of empirical mathematical kinetic models pertinent to food processing and storage." In *Computer-Aided Techniques in Food Technology.* Edited by I. Saguy, 49–70. New York: Marcel Dekker, 1983.

Riggs, J. B. "A systematic approach to modeling." *Chemical Engineering Education* 22, no. 1 (1988): 26–29.

Shatzman, A. R., and D. J. Kosman. "Regulation of galactose oxidase synthesis and secretion in Dactylium dendroides: Effects of pH and culture density." *Journal of Bacteriology* 130 (1977): 455–63.

Teixeira, A. A., and C. F. Shoemaker. *Computerized Food Processing Operations.* New York: Avi, 1989.

Yöndem, F., M. Özilgen, and T. F. Bozoglu. "Kinetic aspects of leavening with mixed cultures of Lactobacillus plantarum and Saccharomyces cerevisiae." *Lebensmittel-Wissenschaft und Technologie* 25 (1992): 162–67.

2

Transport Phenomena Models

2.1 The Differential General Property Balance Equation

The phenomenological models are mostly based on conservation principles. Conservation of mass, energy, and momentum are mathematically analogous; discussion of either one equally applies to the others. A microscopic property (ψ) balance around a differential volume element gives

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\psi \, \mathbf{v}) = \nabla \cdot \psi_G + (\nabla \cdot \delta \nabla \psi), \tag{2.1}$$

with constant δ (Equation 2.1) becomes

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\psi \, \mathbf{v}) = \nabla \cdot \psi_G + \delta \nabla^2 \psi, \tag{2.2}$$

where δ is diffusivity (Table 2.1). The $\nabla$and ∇^2 are del and Laplacian operators.

$\partial\psi/\partial t$ = accumulation rate of the property at a fixed point in the volume element

$\nabla \cdot (\psi \mathbf{v})$ = net input rate of the property to the point with convection

ψ_G = generation rate of the property at the point

$\delta\nabla^2\psi$ or $\nabla \cdot \delta \nabla\psi$ = net input rate of the property to the point with molecular diffusion.

A detailed discussion of derivation of the transport equations may be found in the classical *Transport Phenomena* books (Whitaker 1977; Brodkey and Hershey 1988; Bird, Stewart, and Lightfoot 2007).

Parameters α and ν are called the thermal diffusivity and kinematic viscosity (or viscous diffusivity), respectively, and defined as:

$$\nu = \frac{\mu}{\rho}, \tag{2.3}$$

and

$$\alpha = \frac{\rho c_p}{k}, \tag{2.4}$$

where k and μ are thermal conductivity and viscosity, respectively. Parameter D is the diffusion coefficient for mass transfer.

TABLE 2.1

List of ψ and δ for Use in General Property Balance

Transport Phenomena	Flux	Flux Units	Diffusivity (m^2/s)	Concentration of Property y	Units of ψ
Heat	q	J/m^2s	α	$\rho c_p T$	$J\ m^{-3}$
Mass	J	kg/m^2s	D	c	$kg\ m^{-3}$
Momentum	σ	N/m^2	ν	ρν	$kg\ m^{-2}s^{-1}$

Source: Brodkey, R. S. and Hershey, H. C., *Transport Phenomena,* McGraw-Hill, New York, 1988.

TABLE 2.2

Empirical Laws for One Directional Fluxes

Mass transfer	Fick's Law	$J_z = -D\frac{dc}{dz}$	(2.5)
Heat transfer	Fourier's Law	$q_z = -k\frac{dT}{dz}$	(2.6)
Momentum transfer	Newton's Law	$\sigma_{zy} = -\mu\frac{dv_y}{dz}$	(2.7)

Equation 2.2 is the starting point of almost all the transport phenomena problems. We choose the appropriate ψ from Table 2.1, and expand ∇or ∇^2 in the appropriate coordinate systems, then solve the differential equation. Equations for the flux expressions are given in Table 2.2.

Equations 2.5 through 2.7 imply that the fluxes J_z, q_z, and σ_{zv} are proportional with the gradients dc/dz, dT/dz, and dv_y/dz, respectively. Parameters D, k, and μ are actually the proportionality constants. It should be noticed that mass and heat transfers occurs in the same direction with the gradient. The negative signs imply that the mass, heat, and momentum transfers occur from higher to lower concentration, temperature, or momentum regions, respectively. Liquid molecules are in contact with each other. When a layer of these molecules are set in motion in the y direction, momentum is transferred to the other liquid layers in the z direction due to the molecular interactions (σ_{zv} = viscous flux of y momentum in the z direction).

Choosing an appropriate coordinate system (Figure 2.1) with an appropriate origin may facilitate the modeling process substantially. A cylindrical coordinate system with z axis located on the center line of the cylinder will be preferred when the equation of continuity is used to describe drying behavior of a cylindrical rise grain (Example 2.12); the spherical coordinate system with the origin located to the center of the tuber will be preferred when an equation of energy will be used to evaluate the temperature profiles along a spherical potato (Example 2.9).

2.2 Equation of Continuity

Equations 2.1 and 2.2 become the equation of continuity when $\psi = c_A$, $\psi_G^{\bullet} = R_A$ and $\delta = D$:

$$\frac{\partial c_A}{\partial t} + \nabla \cdot (c_A \mathbf{v}) = R_A + (\nabla \cdot D \nabla c_A), \tag{2.8}$$

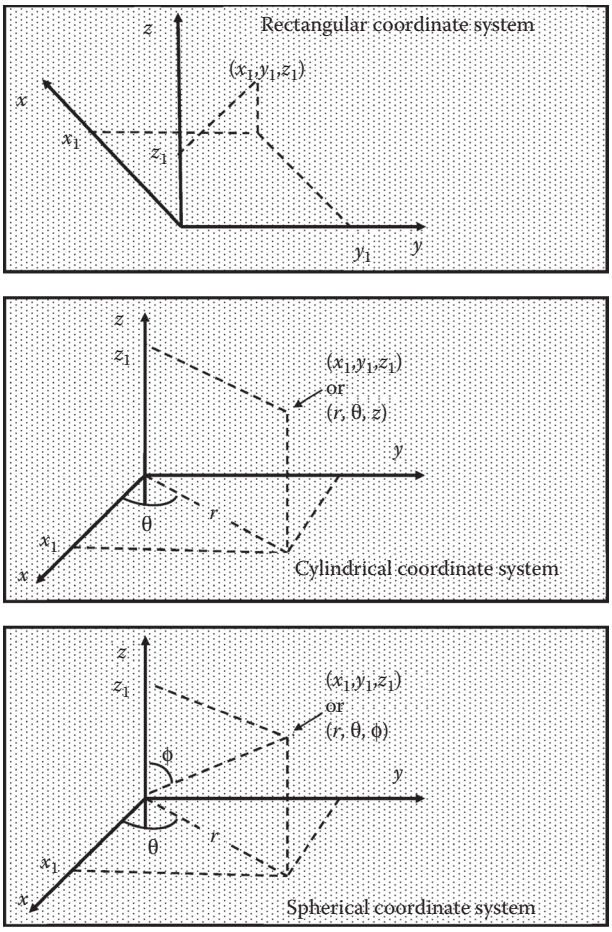


FIGURE 2.1
Coordinate systems employed with transport phenomena models.

$$\frac{\partial c_A}{\partial t} + \nabla \cdot (c_A \mathbf{v}) = R_A + D\nabla^2 c_A. \tag{2.9}$$

Total (convective + diffusive) flux of species A may also be expressed as:

$$N_A = x_A\left(N_A + N_B\right) - D_{AB}\frac{dc_A}{dz}, \tag{2.10}$$

where $N_A + N_B$ is the total flux in the system and $x_A(N_A + N_B)$ is the convective flux of A. The term $D_{AB}dc_A/dz$ represents the diffusive flux superimposed on convection. Equations 2.9 and 2.10 may be combined to express the equation of continuity as:

$$\frac{\partial c_A}{\partial t} + \nabla \cdot \left(N_A\right) = R_A. \tag{2.11}$$

Different forms of equation of continuity as written in rectangular, cylindrical and spherical coordinate systems are depicted in Table 2.3. The term $x_A(N_A + N_B)$ is associated with bulk

TABLE 2.3

Equation of Continuity for Different Coordinate Systems

i) Rectangular coordinates:

$$\frac{\partial c_A}{\partial t}+\left(\frac{\partial N_{Ax}}{\partial x}+\frac{\partial N_{Ay}}{\partial y}+\frac{\partial N_{Az}}{\partial z}\right)=R_A. \tag{2.12}$$

ii) Cylindrical coordinates:

$$\frac{\partial c_A}{\partial t}+\left(\frac{1}{r}\frac{\partial\left(rN_{Ar}\right)}{\partial r}+\frac{1}{r}\frac{\partial N_{A\theta}}{\partial \theta}+\frac{\partial N_{Az}}{\partial z}\right)=R_A. \tag{2.13}$$

iii) Spherical coordinates:

$$\frac{\partial c_A}{\partial t}+\left(\frac{1}{r^2}\frac{\partial\left(r^2N_{Ar}\right)}{\partial r}+\frac{1}{r\sin\theta}\frac{\partial N_{A\theta}\sin\theta}{\partial \theta}+\frac{1}{r\sin\theta}\frac{\partial N_{A\phi}}{\partial \phi}\right)=R_A. \tag{2.14}$$

motion. When our kitchen smells after cooking, we may open the windows to supply air flow and refreshen the air. Here we encourage the bulk motion of the air described by $x_A(N_A + N_B)$. When x_A represents the chemicals causing the smell and after multiplying $(N_A + N_B)$ with x_A, we actually consider the fraction of the smelly chemicals in the bulk flow.

The gas molecules are in random motion in all directions. When we have smelly compounds at one point in the space in our kitchen, we may expect them to move to the neighbor points with molecular motion when there is no bulk flow. If the concentration of the smelly compounds are high at one point and low in the neighborhood, the molecular motion will tend to equalize them because the higher number of molecules leave the concentrated point with random motion. The term $D_{AB}dc_A/dz$ implies that the higher the concentration gradient dc_A/dz, the higher the diffusion rate. Constant D_{AB} is the diffusivity of A in B, where B is the medium that A is diffusing through. The term $D_{AB}dc_A/dz$ also implies that the rate of the diffusion will increase as D_{AB} gets larger.

Solute molecules in the solution, suspended microscopic solids in liquid, or liquid molecules in solids also make random motions and Equation 2.5 may also be used with these systems. The intensity of the random motions tend to decrease with liquid and solid systems as the molecular matrix of the diffusion medium gets more rigid.

Example 2.1: Diffusion of Water Through a Wine Barrel During Aging

During the aging of red wine in a cylindrical wooden barrel (Figure E.2.1.1, radius = R, length = L, thickness of the wood = λ) small amounts of water diffuses through the wood and evaporates. Assuming that all the barrel's surface is available for mass transfer, obtain an expression for the evaporation rate. The fraction of water at the inner surface of the wood is x_{in} and at the outer surface is x_{out}. Write a MATLAB® code and discuss the influence of R, L, λ, D, x_{in}, and x_{out} on the mass transfer rates with appropriate figures.

Solution: It was assumed that the barrel is a perfect cylinder and water diffuses out from all the surfaces. Since the thickness of the wood is much smaller than that of the barrel we may use the rectangular coordinates to simulate water transport through the wood with an equation of continuity:

$$c\frac{\partial x}{\partial t}+\frac{\partial N_w}{\partial z}=R_w.$$

Since $c\partial x/\partial t = 0$ at steady state and $R_w = 0$ (water does not react in the wood) the equation of continuity becomes $dN_w/dz = 0$, implying that N_w is a constant along the z direction. The flux of

water in wood is $N_w = x(N_w + N_{wood}) - D_w(dc_w/dz)$, which becomes $N_w = -cD_w(dx/dz)$ since only a small amount of water is available in the wood ($x \ll 1$). After combining the final forms of the flux expression and the equation of continuity we will obtain $d^2x/dz^2 = 0$. Upon integration, a moisture profile along the wood will be $x = \mathsf{K}_1 z + \mathsf{K}_2$. Constants K_1 and K_2 may be evaluated from the boundary conditions as: $\mathsf{K}_1 = (x_{out} - x_{in})/\lambda$ and $\mathsf{K}_2 = x_{in}$. We may substitute $dx/dz = (x_{out} - x_{in})/\lambda$ in $N_w = -cD_w(dx/dz)$ and obtain $N_w = cD_w(x_{in} - x_{out})/\lambda$. The total evaporation rate through the barrel is $A_{barrel}N_w = 2\pi R(R + L)cD_w(x_{in} - x_{out})/\lambda$ where A_{barrel} is the total mass transfer area through the barrel. MATLAB® code E.2.1 simulates the effects of the barrel radius, barrel length, barrel wood thickness, numerical value of the diffusivity of water vapor through the barrel wood, and the fraction of the water vapor in the cellar and on the inside surface of the wood on the amounts of water loss from the barrel surface by evaporation (Figures E.2.1.2 through E.2.1.7).

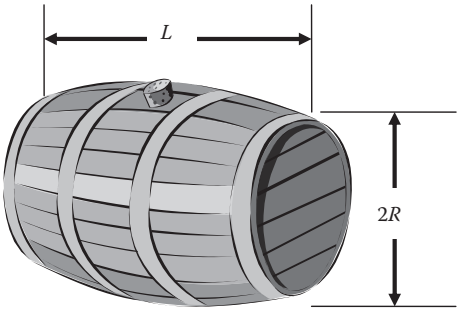


FIGURE E.2.1.1
The wine barrel. Variation of the radius along the longitudinal direction is neglected. Thickness of the wood is λ.

MATLAB® CODE E.2.1

Command Window:

```
clear all
close all

% effect of R on the evaporation rate
L=2; %m
lambda=0.025; % m
xin=0.15;
xout=0.10;
D=5e-6; % D=Dw*c, kg/ms
R1=[0.5: 0.05:1.5]; % m
for i=1:1:length(R1);
A1(i)=2*pi*R1(i)*(L+R1(i))*D*(xin-xout)/lambda; % A=Abarrel*Nw,
kg/s
end
plot(R1,A1, '*-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('R (m)')

% effect of L on the evaporation rate
R=1; % m
L2=[1:0.05:3]; % m
for k=1:1:length(L2);
```

```
A2(k)=2*pi*R*(L2(k)+R)*D*(xin-xout)/lambda; % A=Abarrel*Nw, kg/s
end
figure
plot(L2,A2, 'o-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('L (m)')

% effect of barrel thickness on the evaporation rate
lambda3=[0.01:0.0005:0.03]; % m
for i=1:1:length(lambda3);
A3(i)=2*pi*R*(L+R)*D*(xin-xout)/lambda3(i); % A=Abarrel*Nw, kg/s
end
figure
plot(lambda3,A3, '*-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('lambda (m)')

% effect of diffusivity on the evaporation rate
D4=[4e-6: 1e-7: 6e-6]; % m2/s
for i=1:1:length(D4);
A4(i)=2*pi*R*(L+R)*D4(i)*(xin-xout)/lambda; % A=Abarrel*Nw, kg/s
end
figure
plot(D4,A4, '*-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('Diffusivity (kg/ms)')

% effect of xOut on the evaporation rate
xOut5=[0.0:0.01: 0.20]; % fraction of water at the inner side of
the wood
for i=1:1:length(xOut5);
A5(i)=2*pi*R*(L+R)*D*(xin-xOut5(i))/lambda; % A=Abarrel*Nw, kg/s
end
figure
plot(xOut5,A5, '*-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('xOut (fraction of water at the outside surface of the
wood)')

% effect of xIn on the evaporation rate
xIn6=[0.20: 0.01: 0.30]; % fraction of water at the inner side of
the wood
for i=1:1:length(xIn6);
A6(i)=2*pi*R*(L+R)*D*(xIn6(i)-xout)/lambda; % A=Abarrel*Nw, kg/s
end
figure
plot(xIn6,A6, '^-')
ylabel('EVAPORATION RATE (kg/s)');
xlabel('xIn (fraction of water at the inside surface of the wood)')
```

Figures E.2.1.2 through E.2.1.7, will appear on the screen after running the code.

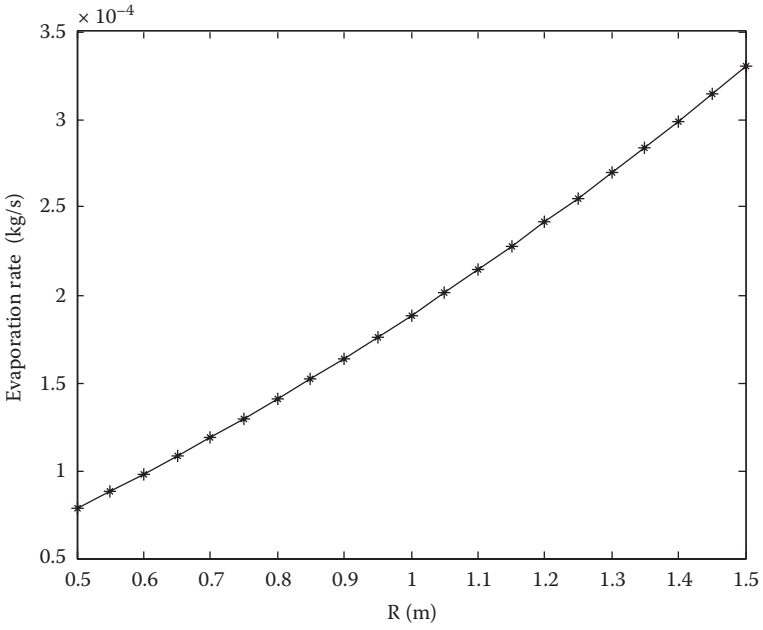


FIGURE E.2.1.2
Effect of the barrel radius on the amount of water lost by evaporation from the barrel surface.

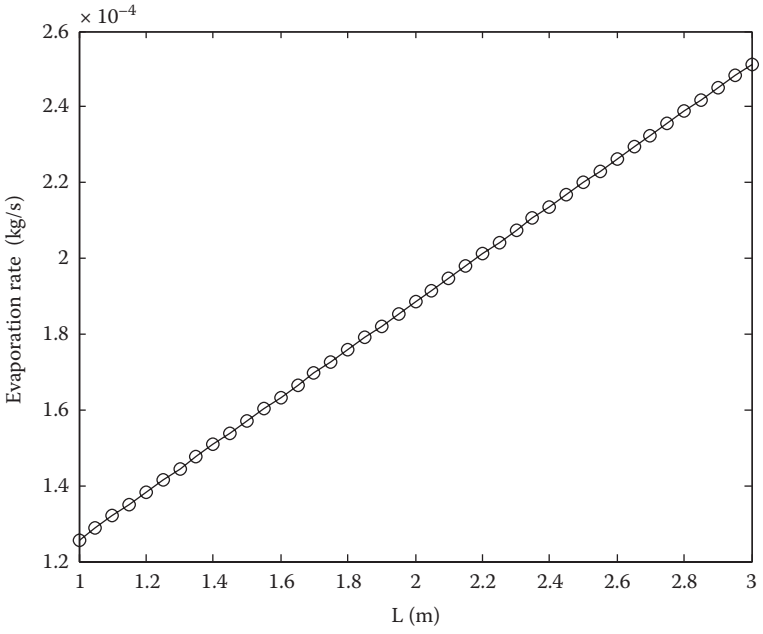


FIGURE E.2.1.3
Effect of the barrel length on the amount of water lost by evaporation from the barrel surface.

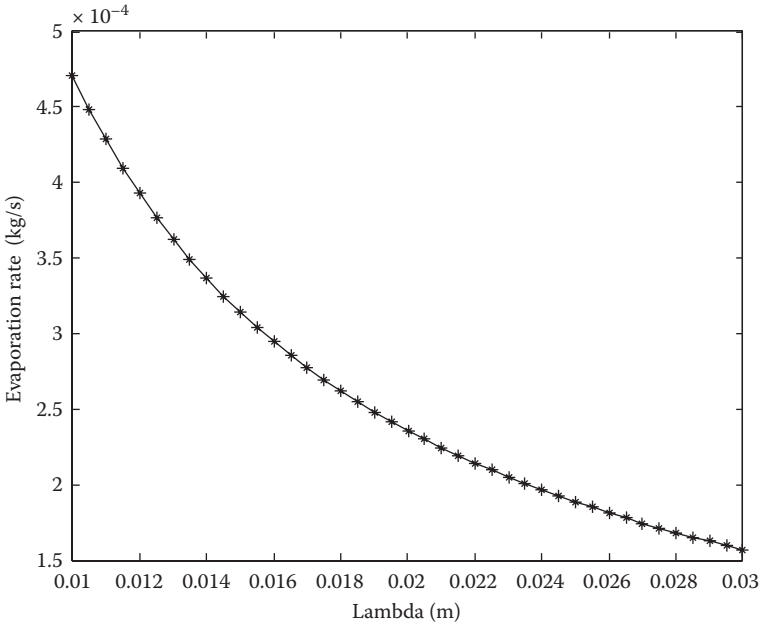


FIGURE E.2.1.4
Effect of the barrel wood thickness on the amount of water lost by evaporation from the barrel surface.

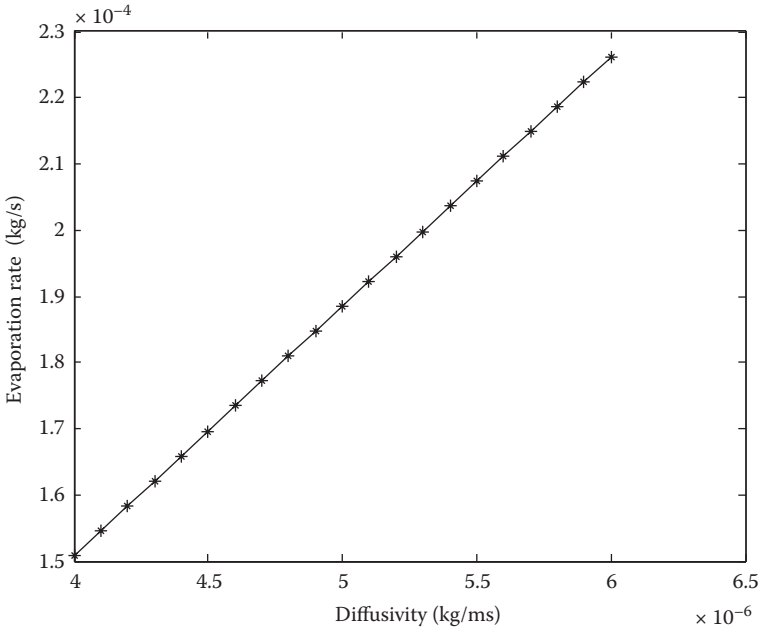


FIGURE E.2.1.5
Effect of the diffusivity on the amount of water lost by evaporation from the barrel surface.

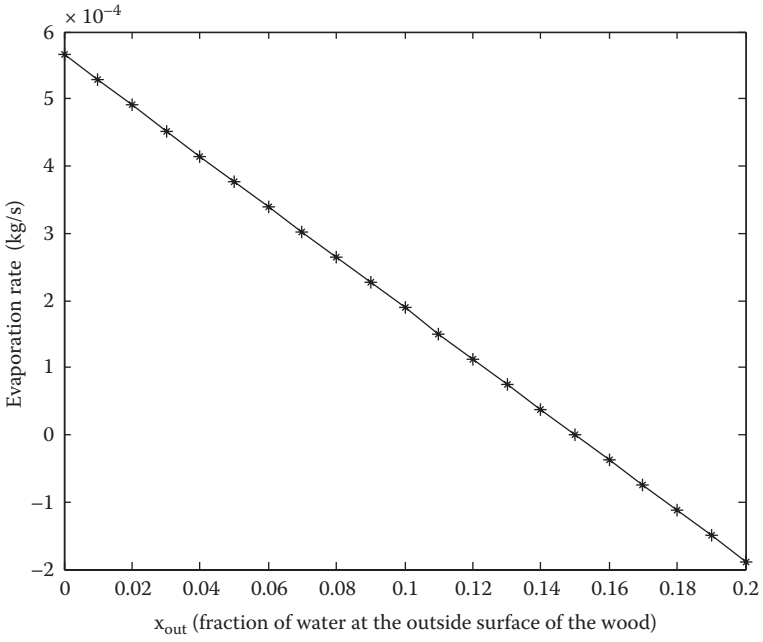


FIGURE E.2.1.6
Effect of x_{out} on the amount of water lost by evaporation from the barrel surface.

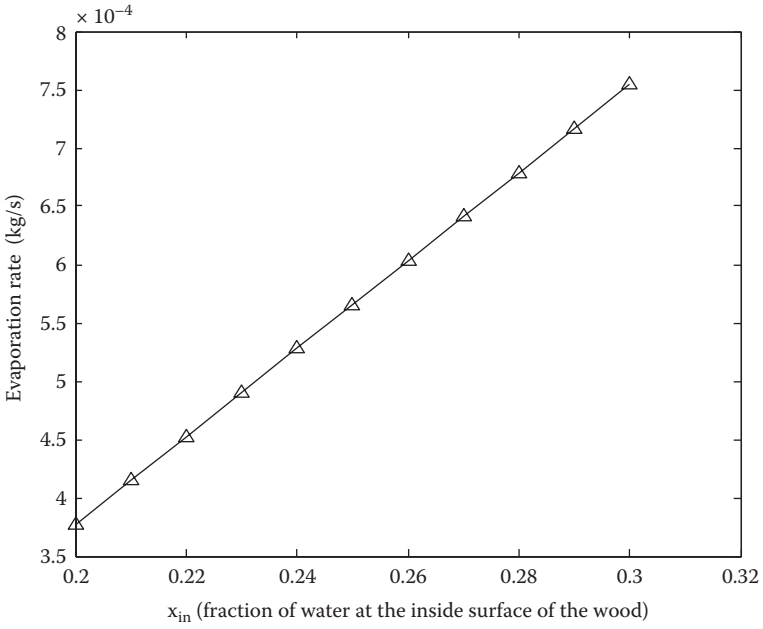


FIGURE E.2.1.7
Effect of x_{in} on the amount of water lost by evaporation from the barrel surface.

TABLE 2.4

Equation of Energy for Different Coordinate Systems

i) Rectangular coordinates:

$$\frac{\partial T}{\partial t}+v_x\frac{\partial T}{\partial x}+v_y\frac{\partial T}{\partial y}+v_z\frac{\partial T}{\partial z}=\frac{\psi_G^{\bullet}}{\rho c}+\frac{\partial}{\partial x}\left(\alpha\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\alpha\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\alpha\frac{\partial T}{\partial z}\right). \tag{2.16}$$

ii) Cylindrical coordinates:

$$\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+\frac{v_\theta}{r}\frac{\partial T}{\partial \theta}+v_z\frac{\partial T}{\partial z}=\frac{\psi_G^{\bullet}}{\rho c}+\frac{1}{r}\frac{\partial}{\partial r}\left(r\alpha\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \theta}\left(\alpha\frac{\partial T}{\partial \theta}\right)+\frac{\partial}{\partial z}\left(\alpha\frac{\partial T}{\partial z}\right). \tag{2.17}$$

iii) Spherical coordinates:

$$\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+\frac{v_\theta}{r}\frac{\partial T}{\partial \theta}+\frac{v_\phi}{r\sin\theta}\frac{\partial T}{\partial \phi}=\frac{\psi_G^{\bullet}}{\rho c}+\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\alpha\frac{\partial T}{\partial r}\right)+\frac{1}{r^2\sin\theta}\frac{\partial}{\partial \theta}\left(\alpha\sin\theta\frac{\partial T}{\partial \theta}\right)+\frac{1}{r^2\sin^2\theta}\frac{\partial}{\partial \phi}\left(\alpha\frac{\partial T}{\partial \phi}\right). \tag{2.18}$$

2.3 Equation of Energy

After substituting $\psi = \rho c_p T$, Equation 2.1 becomes

$$\frac{\partial(\rho\, c_p T)}{\partial t}+\nabla\cdot(\rho\, c_p T\, \mathbf{v})=\psi_G^{\bullet}+(\nabla\cdot\delta\,\nabla\rho\, c_p T). \tag{2.15}$$

The generation term $\psi_G^{\bullet}$ accounts for heat generation by viscous dissipation in the microwave field, radiation, and so on. Equation 2.15, for incompressible fluids, may be expanded for various coordinate systems as given in Table 2.4.

2.4 Equation of Motion

The equation of motion is more complicated than the equations of energy and continuity, because in momentum balance ψ is a vector composed of components ψ_x and ψ_y. When ψ_z is constant and after substituting $\psi_x = \rho v_x$ and $\delta = \nu$ in Equations 2.1 and 2.2, the x component of the momentum balance becomes

$$\frac{\partial v_x}{\partial t}+(\nabla\cdot\mathbf{v})v_x=\frac{\psi_{Gx}^{\bullet}}{\rho}+(\nabla\cdot\psi\nabla v_x), \tag{2.19}$$

and

$$\frac{\partial v_x}{\partial t}+(\nabla\cdot\mathbf{v})v_x=\frac{\psi_{Gx}^{\bullet}}{\rho}+\mathbf{v}\nabla^2 v_x. \tag{2.20}$$

It should be noticed that $\nabla\cdot\mathbf{v}=0$ when ρ is constant. Equations for the y and z components of the momentum balance may be written similarly as Equations 2.19 and 2.20.

TABLE 2.5

Navier–Stokes Equation for Different Coordinate Systems

i) The x component in rectangular coordinates:

$$\frac{\partial v_x}{\partial t}+v_x\frac{\partial v_x}{\partial x}+v_y\frac{\partial v_x}{\partial y}+v_z\frac{\partial v_x}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+g_x+\nu\left(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial^2 v_x}{\partial y^2}+\frac{\partial^2 v_x}{\partial z^2}\right) \tag{2.21}$$

ii) The r component in cylindrical coordinates:

$$\frac{\partial v_r}{\partial t}+v_r\frac{\partial v_r}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_r}{\partial \theta}+v_z\frac{\partial v_r}{\partial z}-\frac{v_\theta^2}{r}=-\frac{1}{\rho}\frac{\partial p}{\partial r}+g_r+\nu\frac{\partial^2 v_r}{\partial r^2}+\frac{\nu}{r}\frac{\partial v_r}{\partial r}-\nu\frac{v_r}{r^2}+\frac{\nu}{r^2}\frac{\partial^2 v_r}{\partial \theta^2}-\frac{2\nu}{r}\frac{\partial v_\theta}{\partial \theta}+\nu\frac{\partial^2 v_r}{\partial z^2}. \tag{2.22}$$

iii) The r component in spherical coordinates:

$$\begin{aligned}&\frac{\partial v_r}{\partial t}+v_r\frac{\partial v_r}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_r}{\partial \theta}+\frac{v_\phi}{r\sin\theta}\frac{\partial v_r}{\partial \phi}-\frac{v_\theta^2}{r}-\frac{v_\phi^2}{r}\\&\quad=-\frac{1}{\rho}\frac{\partial p}{\partial r}+g_r+\frac{\nu}{r^2}\left[\frac{\partial}{\partial r}\left(r^2\frac{\partial v_r}{\partial r}\right)\right]+\left(\frac{\nu}{r^2\sin\theta}\right)\left[\frac{\partial}{\partial \theta}\left(\sin\theta\frac{\partial v_r}{\partial \theta}\right)\right]+\left(\frac{\nu}{r^2\sin^2\theta}\right)\left(\frac{\partial^2 v_r}{\partial \phi^2}\right).\end{aligned} \tag{2.23}$$

When the rate of momentum generation is proportional to the pressure gradient and x component of the gravitational acceleration; that is, $\psi_{Gx}^{\bullet}=-(\partial p/\partial x)+\rho g_x$, Equation 2.19 becomes the Navier–Stokes equation and its expansion for different coordinate systems is given in Table 2.5.

2.5 Theories for Liquid Transport Coefficients

Although viscosity, thermal conductivity, and mass diffusion coefficients are defined with empirical relations in Equations 2.5 through 2.7, there are theoretical explanations to these equations for simple systems. Foods are either liquid or solid or both liquid and solid. Theoretical models for viscosity, thermal conductivity, and mass diffusion coefficients are available for pure liquids or suspension of fine particles. Theoretical models are rarely used to evaluate values of μ, k, or D, empirical models or the actual experimental data are preferred in process calculations.

2.5.1 Eyring's Theory of Liquid Viscosity

In a pure liquid at rest, the individual molecules are considered to be confined in a large *cage* formed by its nearest neighbors (Figure 2.2). This cage represents an energy barrier of $\Delta G^{\#}/N_{Av}$ ($\Delta G^{\#}$ = energy barrier, N_{Av} = Avagadro number). The Glasstone, Laidler, and Eyring theory (1941) suggests that the liquid at rest undergoes rearrangements in which one molecule at a time is transferred to an adjoining *hole*. The rate (frequency) of jumps in each direction is

$$k=\left(\frac{\kappa T}{h}\right)\exp\left(-\frac{\Delta G_0^*}{RT}\right), \tag{2.24}$$

where h is the Planck constant, k is the Boltzman constant, T is temperature, $\Delta G_o^{\#}$ is the energy barrier (molar free energy of activation), and R is the gas constant.

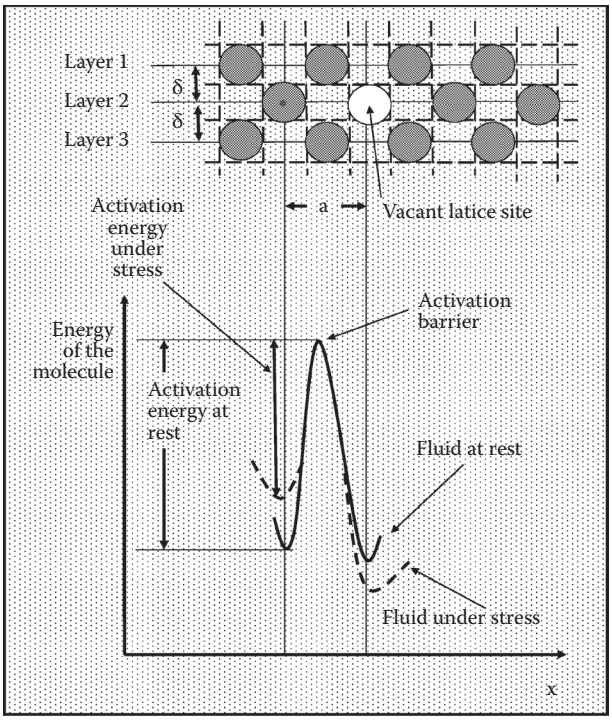


FIGURE 2.2
Illustration of the escape process in flow of a liquid. Molecule marked with * must pass over the activation barrier to reach the vacant site.

When liquid moves in x direction with velocity gradient dv_x/dy, the energy barrier is distorted and the frequency of molecular rearrangements increases. The distorted energy barrier is

$$-\Delta G^{\#} = -\Delta G_o^{\#} \pm \frac{a}{\delta}\frac{\sigma_{yx}V_m}{2}, \tag{2.25}$$

where $\Delta G^{\#}$ is the distorted mean free energy of activation, a is the length of molecular jump, δ is the distance between the molecular layers, V_m is the volume of one mole of liquid, σ_{xy} is the applied shear stress and $(a/\delta)(\sigma_{xy}V_m/2)$ is the approximation of the work done on the molecules as they move to the top of the energy barriers with the applied shear stress. The minus sign is employed to describe the movement of the molecules against the shear stress. When $(\sigma_{xy}aV_m/\delta RT) \ll 1$ and $\delta/a = 1$ product of the net frequency of jumps and a/δ is the shear rate:

$$-\frac{dv_x}{dt} = \frac{\kappa T}{h}\left(-\Delta G_o^{\#}/RT\right)\frac{\sigma_{xy}V_m}{2RT}, \tag{2.26}$$

where h = Planck constant. Equation 2.26 is Newton's Law of viscosity with (Bird, Stewart, and Lightfoot 2007)

$$\mu = \frac{N_{Av}h}{V_m}e^{\Delta G_o^{\#}/RT}. \tag{2.27}$$

Temperature effects on liquid viscosity may be expressed with the Arrhenius expression:

$$\mu = \mu_0 e^{(E_a/RT)}, \tag{2.28}$$

where μ_0 is the preexponential constant, E_a is the activation energy, and R is the gas constant.

Example 2.2: Kinetic Compensation Relations for the Viscosity of Fruit Juices

Gibbs free energy of activation at rest may be expressed as (Özilgen and Bayindirli 1992)

$$\Delta G_o^{\#} = \Delta H^{\#} - T\Delta S^{\#},$$

where $\Delta H^{\#}$ = activation enthalpy and $\Delta S^{\#}$ = activation entropy. After substituting this expression, Equation 2.27 becomes

$$\mu = \frac{N_{Av}h}{V_m}\exp\left(\Delta H^{\#}/RT\right)\exp\left(-\Delta S^{\#}/RT\right).$$

Comparing this equation with Equation 2.28 requires

$$E_a = \Delta H^{\#} + RT$$

and

$$\mu_0 = \frac{N_{Av}h}{2.72V_m}\exp\left(-\Delta S^{\#}/RT\right).$$

In the experiments performed under slightly different experimental conditions, generally linear relationships are observed between the activation energy E_a and the frequency factor μ_0, and the activation enthalpy $\Delta H^{\#}$ and the activation entropy $\Delta S^{\#}$:

$$\ln\mu_0 = \alpha E_a + \beta$$

and

$$\Delta S^{\#} = \delta\Delta H^{\#} + \phi.$$

These equations are called the compensation relations. Since μ_0 and $\Delta S^{\#}$ and E_a and $\Delta H^{\#}$ are interrelated, the validity of either equation implies that the other one is also valid. Although E_a, μ_0, $\Delta H^{\#}$, and $\Delta S^{\#}$ varies with the experimental conditions, α, β, δ, and ϕ are constants through the range of the experiments. A sample set of compensation relations are given in Figure E.2.2.

The compensation relations may be used for interpolation in process design when data are available in the close range, but not under the exactly required conditions (Özilgen and Özilgen 1996). MATLAB® code E.2.2 plots the kinetic compensation model for the viscosity of fruit juices and compares it with the data (Figure E.2.2).

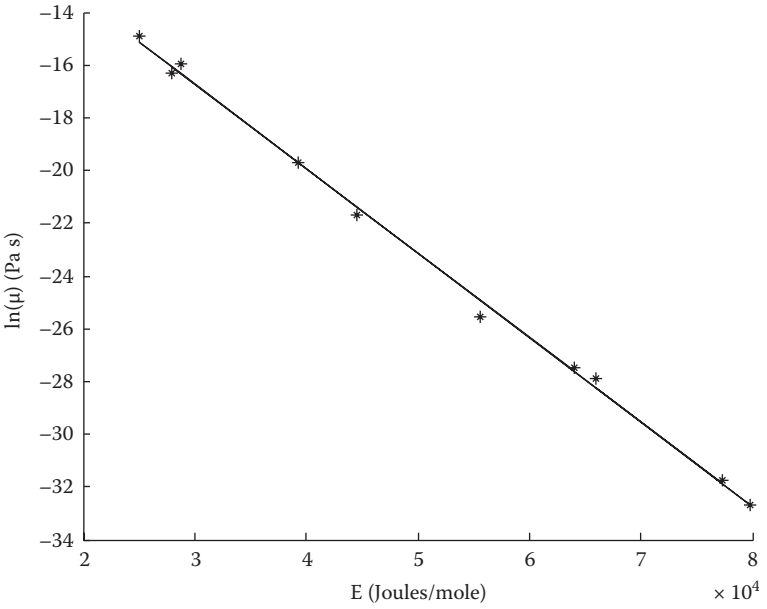


FIGURE E.2.2
Variation of parameter $\ln(\mu_0)$ of apple juice with activation energy. Equation of the line: $\ln(\mu_0) = -7.09 - 3.19 \times 10\text{-}4E_a$. Standard error ($s_e = 0.3054$) and correlation coefficient ($r = 0.9988$) are calculated by using tools of MATLAB®. Standard error was about 2% of the magnitude of the measurements. (From Özilgen, M. and Bayindirli, L., *Journal of Food Engineering*, 17, 143–51, 1992.)

MATLAB® CODE E.2.2

Command Window:

```
clear all
close all

% introduce the data
mu=[8.09e-8 1.61e-14 7.95e-13 6.55e-15 3.38e-7 1.18e-7 2.7e-9
3.91e-10 7.93e-12 1.15e-12];
E = [2.8e4 7.73e4 6.6e4 7.98e4 2.51e4 2.88e4 3.93e4 4.46e4 5.56e4
6.4e4];

% plot the data
ylabel('ln(mu) (Pa s)');
xlabel('E (Joules/mole)');
hold on, plot(E,log(mu),'k*');

% find the best fitting line equation
f = polyfit(E,log(mu),1);
y = polyval(f,E);

% plot the model
plot(E, y);
```

```
% find the correlation coefficient and the standard error
rmatrix=corrcoef(E,log(mu));
   r=rmatrix(1,2)
for i=1:length(E);
d(i) = (log(mu(i))-(f(2)+f(1)*E(i)))^2;
end;
Se = sqrt(sum(d)/length(E))
```

The following lines and Figure E.2.2. will appear in the screen after running the code:

```
r =
   -0.9988

Se =
    0.3054
```

2.5.2 Thermal Conductivity of Liquids

Thermal conductivity of monatomic low density gases is $k = (1/3)c_v u\lambda$ (c_v = specific heat, u = mean molecular velocity, λ = collision length). This equation may be modified for liquids after assuming that pure liquid molecules are arranged in cubic lattice, energy is transferred from one lattice plane to the next, the heat capacity of monatomic liquids at constant volume is about the same as for a solid at high temperature and the mean molecular speed is the same as the sonic velocity v_s and the distance that energy travels per single collision is the same as the lattice spacing $\left(\bar{V}/N_{Av}\right)^{1/2}$ as (Bird, Stewart, and Lightfoot 2007)

$$k = 3\left(\frac{N_{Av}}{\bar{V}}\right)^{1/3} \kappa v_s, \tag{2.29}$$

where $\bar{V}$ is specific volume.

2.5.3 Hydrodynamic Theory of Diffusion in Liquids

Diffusion of a single particle or solute molecule through a stationary medium is calculated after making force balance around the particle as

$$D_{AB} = \kappa T \frac{v}{F}, \tag{2.30}$$

where v is the velocity of the particle, F is the drag force on the particle, and v/F is the mobility of the particle. When $N_{re} \ll 1$ and there is no tendency for fluid to slip at the surface of the diffusing particle:

$$F_A = 6\pi\mu v d_p, \tag{2.31}$$

where d_p is the effective particle diameter. Substituting Equation 2.31 into Equation 2.30 gives

$$D_{AB} = \frac{\kappa T}{6\pi\mu v d_p}. \tag{2.32}$$

Equation 2.32 is called the Stokes–Einstein equation.

When there is no tendency for the fluid to stick to the surface of the diffusing particle then Equations 2.31 and 2.32 becomes

$$F_A = 4\pi\mu v d_p, \tag{2.33}$$

and

$$D_{AB} = \frac{\kappa T}{4\pi\mu v d_p}. \tag{2.34}$$

2.5.4 Eyring's Theory of Liquid Diffusion

As explained for viscosity, Eyring's theory assumes that the liquid molecules are entrapped into a *cage* made by neighbor molecules. The molecule is expected to exceed an activation barrier to jump into the neighboring *holes* with a first-order reaction. A constant average distance is involved into each jump. Derivation of the expression for diffusivity is very similar to that of the viscosity. The resulting equation is

$$D_{AB} = \frac{\kappa T}{\mu}\left(\frac{N_{Av}}{V}\right)^{1/3}. \tag{2.35}$$

Temperature effects on diffusivity may be described with the Arrhenius expression:

$$D = D_0 \exp\left\{-\frac{E_a}{RT}\right\}, \tag{2.36}$$

where D_0 = preexponential constant.

Example 2.3: Temperature Effects on Diffusivity of Water in Starch

Parameters E_a and D_0 were given for diffusion in the hydrated amylopectin (fraction of starch) and potatoes as (Özilgen 1993)

Diffusion System	D_0 (m²/s)	E_a (J/mol)
Water in hydrated amylopectin	3.37×10^{-6}	2.55×10^{4}
Glucose in potato tissue	1.78×10^{-6}	2.03×10^{4}
Potassium in potato tissue	4.08×10^{-6}	2.09×10^{4}

a. Compare the diffusivities of water in amylopectin, glucose in potato tissue, and potassium in potato tissue at 65°C.

Solution: $D = D_0 \exp\{-E_a/RT\}$ and $R = 8.314$ J/mol K after substituting the numbers we will have

Diffusion System	D (m²/s)
Water in hydrated amylopectin	2.04×10^{-9}
Glucose in potato tissue	1.34×10^{-9}
Potassium in potato tissue	2.40×10^{-9}

Water molecules are smaller than the potassium molecules and both of them are smaller than the glucose molecules; potato tissue is cellular, and amylopectin has a simpler structure. The results indicate that the diffusivity of smaller molecules is greater in structurally simpler systems.

b. Compare the diffusivity of potassium in potato tissue at 65°C and 90°C.

Solution: After substituting the numbers in Equation 2.36 we will have

Diffusion System	D (m^2/s)
Potassium in potato tissue at 65°C	2.40×10^{-9}
Potassium in potato tissue at 90°C	4.00×10^{-9}

The results show a 1.67-fold increase in D with 25°C increase in *T*.

Example 2.4: Compensation Relations for Diffusivity of Water in Starch

The compensation relations, similarly as those of viscosity, applies to diffusivity:

$$\ln D_0 = \alpha E_a + \beta.$$

These equations may also be used for interpolation in process design (Özilgen and Özilgen 1996). Figure E.2.4 describes a variation of ln D_0 with the activation energy in a different media. MATLAB® code E.2.4 plots the kinetic compensation model for diffusivity of water in starch and compares it with the data (Figure E.2.4).

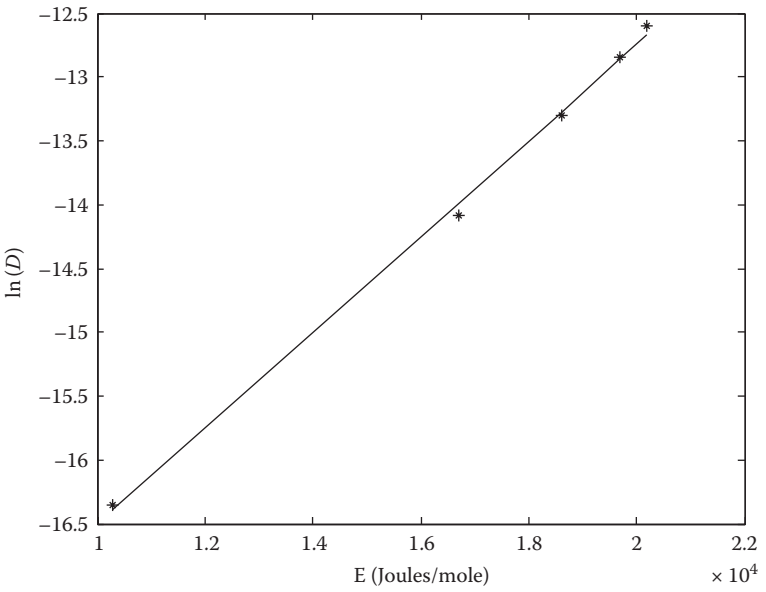


FIGURE E.2.4
Variation of parameter ln(D_0) with activation energy during diffusion of water in hydrated. Standard error (s_e = 0.0572) and correlation coefficient (r = 0.9991) are calculated by using MATLAB® tools. Standard error was less than 0.5% of the mean value of ln(D). (From Özilgen, M., *Starch*, 45, 48–51, 1993.)