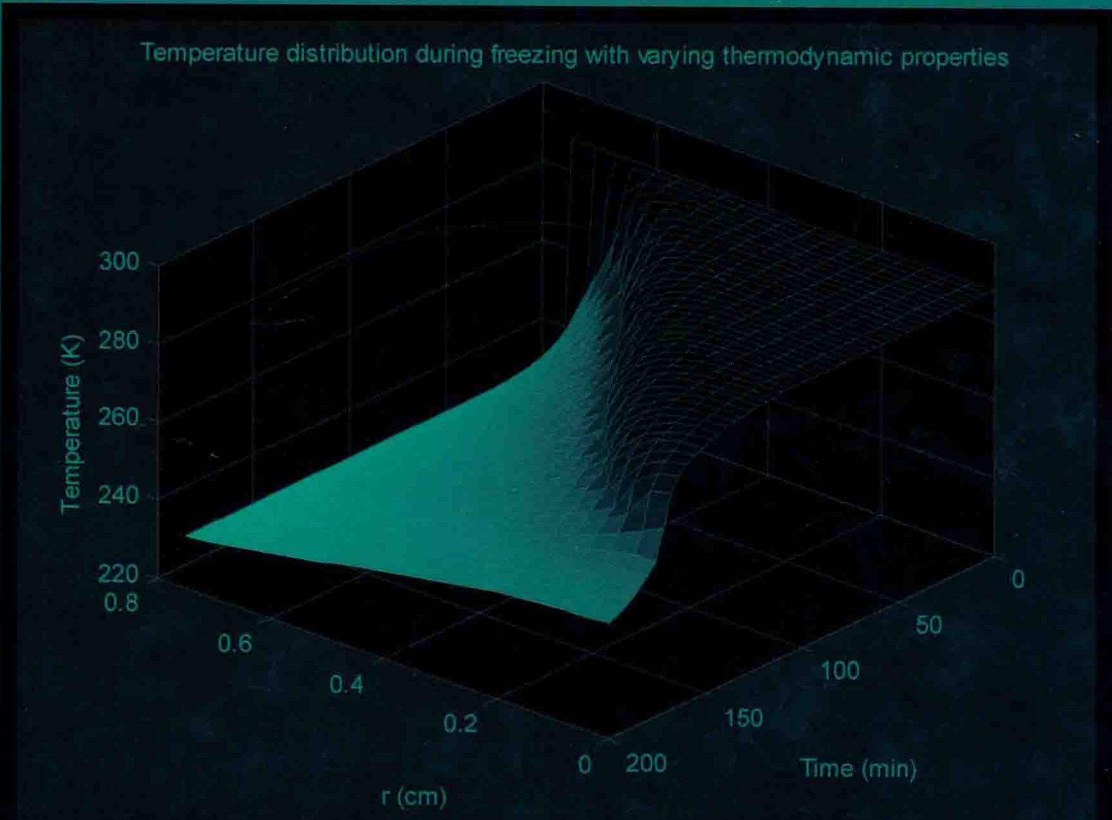
Handbook of Food Process Modeling *and* Statistical Quality Control

with Extensive MATLAB® Applications

SECOND EDITION



Mustafa Özilgen

Includes CD-ROM with MATLAB® Codes

CRC Press
Taylor & Francis Group

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

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4398-1486-4 (Hardback)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Handbook of Food Process Modeling *and* Statistical Quality Control

with Extensive MATLAB® Applications

SECOND EDITION

To Sibel and Arda

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.

Contents

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

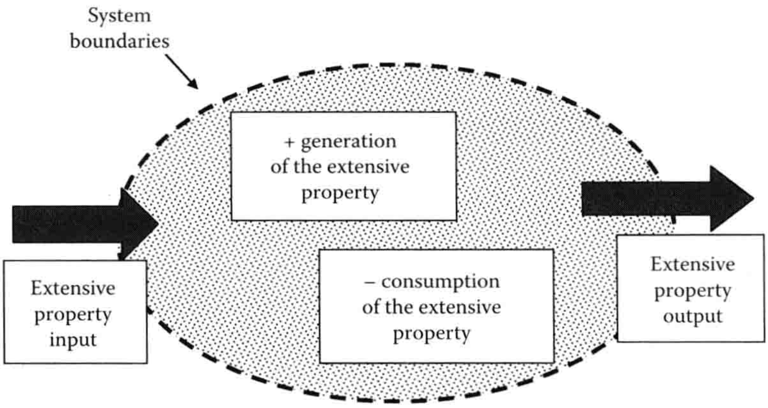


FIGURE 1.1
Description of the system for the application of the conservation laws.

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