

Title: MATLAB Applications in Chemical Engineering

Author: Chyi-Tsong Chen (陳奇中), Ph.D.

Professor of Chemical Engineering Department

Feng Chia University, Taichung, Taiwan

台灣逢甲大學化工系; Email: ctchen@fcu.edu.tw

Publisher: Tung Hua Book Co., Ltd. (台灣東華書局)

ISBN: 978-957-483-769-4

Features of the book

This book addresses the applications of MATLAB and Simulink in the solution of chemical engineering problems. By classifying the problems into seven different categories, the author organizes this book as follows:

Chapter One - Solution of a System of Linear Equations

Chapter Two - Solution of Nonlinear Equations

Chapter Three - Interpolation , Differentiation and Integration

Chapter Four - Numerical Solution of Ordinary Differential Equations

Chapter Five - Numerical Solution of Partial Differential Equations

Chapter Six - Process Optimization

Chapter Seven - Parameter Estimation

Each chapter is arranged in four major parts. In the first part, the basic problem patterns that can be solved with MATLAB are presented. The second part describes how to apply MATLAB commands to solve the formulated problems in the field of chemical engineering. In the third and the fourth parts, exercises and summary of MATLAB instructions are provided, respectively. The description of the chemical engineering example follows the sequence of problem formulation, model analysis, MATLAB program design, execution results, and discussion. In this way, learners are first aware of the basic problem patterns and the underlying chemical engineering principles, followed by further familiarizing themselves with the relevant MATLAB instructions and programming skills. Readers are encouraged to do exercises to practice their problem-solving skills and deepen the fundamental knowledge of chemical engineering and relevant application problems.

Contents

Chapter 1: Solution of a System of Linear Equations	1
1.1 Properties of linear equation systems and the relevant MATLAB commands	1
1.1.1 A simple example: the liquid blending problem	1
1.1.2 Relevant properties of linear equation systems	3
1.1.3 Relevant MATLAB commands	5
1.2 Chemical engineering examples	10
Example 1-2-1 Composition analysis of a distillation column system	10
Example 1-2-2 Temperature analysis of an insulated stainless steel pipeline	14
Example 1-2-3 Composition analysis of a set of CSTRs	17
Example 1-2-4 Independent reactions in a reaction system	21
Example 1-2-5 Composition distributions in a distillation column	23
Example 1-2-6 Steady-state analysis of a batch reaction system	33
1.3 Exercises	40
1.4 Summary of the MATLAB commands related to this chapter	46
Chapter 2: Solution of Nonlinear Equations	49
2.1 Relevant MATLAB commands and the Simulink solution interface	49
2.1.1 Nonlinear equation of a single variable	49
2.1.2 Solution of a system of nonlinear equations	59
2.2 Chemical engineering examples	68
Example 2-2-1 Boiling point of an ideal solution	68
Example 2-2-2 Equilibrium concentrations in a reaction system	71
Example 2-2-3 Analysis of a pipeline network	76
Example 2-2-4 A material drying process through heat conduction and forced convection	79
Example 2-2-5 Simulation of a multistage binary distillation	83
Example 2-2-6 Analysis of a set of three CSTRs in series	94
2.3 Exercises	97
2.4 Summary of MATLAB commands related to this chapter	115
Chapter 3: Interpolation, Differentiation, and Integration	119
3.1 Interpolation commands in MATLAB	119
3.1.1 One-dimensional interpolation	119
3.1.2 Two-dimensional interpolation	122
3.2 Numerical differentiation	125
3.2.1 The application of the MATLAB diff command in numerical differentiation	126
3.2.2 Polynomial fitting and its application to the calculation of derivatives of a data set	127
3.2.3 Higher-order derivatives of a data set	128
3.2.4 The derivatives calculation of a known function	138
3.2.5 A method for calculating numerical gradients	143

3.3	Numerical integration	144
3.3.1	Integral of numerical data	144
3.3.2	Integral of a known function	145
3.3.3	Double integral	147
3.4	Chemical engineering examples	149
Example 3-4-1	Preparation of a water solution with a required viscosity	149
Example 3-4-2	Interpolation of diffusivity coefficients	151
Example 3-4-3	Reaction rate equation of a batch reactor	153
Example 3-4-4	Volume fraction of solid particles in a gas-solid fluidized bed	158
Example 3-4-5	Average residence time calculation based on a tracer response	161
Example 3-4-6	Design of an absorption tower	163
Example 3-4-7	Reaction time of an adiabatic batch reactor	166
Example 3-4-8	Breakthrough time determination for an absorption tower	169
3.5	Exercises	173
3.6	Summary of the MATLAB commands related to this chapter	185

Chapter 4: Numerical Solution of Ordinary Differential Equations

.....	187	
4.1	Initial value problems for ordinary differential equations	187
4.1.1	The standard problems	187
4.1.2	The MATLAB ODE solvers	188
4.1.3	Solving ODEs with MATLAB Simulink	194
4.1.4	The DEE solution interface	208
4.2	Higher-order ordinary differential equations	212
4.3	Stiff differential equations	217
4.4	Differential-algebraic equation system	222
4.5	Boundary-valued ordinary differential equations	226
4.5.1	Problem patterns	226
4.5.2	MATLAB commands for solving two-point boundary value problems	227
4.5.3	Multipoint BVP	237
4.6	Chemical engineering examples	242
Example 4-6-1	Temperature and conversion distributions in a catalyzed tubular reactor	242
Example 4-6-2	Temperature and conversion distribution in a plug-flow reactor	248
Example 4-6-3	Biochemical process dynamics in a batch reactor	252
Example 4-6-4	Temperature distribution on a flat panel with heat conduction and radiation ..	255
Example 4-6-5	Flow dynamics of a non-Newtonian fluid	259
Example 4-6-6	Optimal operation temperature for penicillin fermentation	264
4.7	Exercises	270
4.8	Summary of the MATLAB commands related to this chapter	293

Chapter 5: Numerical Solution of Partial Differential Equations

..... 295

5.1	Classifications of PDEs	295
5.1.1	The order of a PDE	295
5.1.2	Nonlinearity of a PDE	296
5.1.3	Categories of initial conditions and boundary conditions	297
5.2	The MATLAB PDE toolbox	300
5.2.1	The MATLAB PDE solver	300
5.2.2	The PDE graphical interface toolbox	314
5.2.2.1	The solvable PDE problem patterns	316
5.2.2.2	Solution of PDE problems with the pdetool interface	318
5.3	Chemical engineering examples	325
	Example 5-3-1 Temperature and reaction rate distributions in a catalyzed reactor	325
	Example 5-3-2 Concentration distribution in a diffusive reaction system	333
	Example 5-3-3 Rapid cooling of a hot solid object	339
	Example 5-3-4 Two-dimensional heat transfer	344
	Example 5-3-5 The permeation of gaseous solute into a liquid film	355
	Example 5-3-6 Concentration distribution of ethanol in a tube	360
	Example 5-3-7 Heat conduction of a long rod	362
	Example 5-3-8 Unsteady-state heat conduction in a flat panel	366
5.4	Exercises	370
5.5	Summary of the MATLAB commands related to this chapter	378
 Chapter 6: Process Optimization		381
6.1	The optimization problem and the relevant MATLAB commands	381
6.1.1	Optimization problems of a single decision variable	381
6.1.2	Multivariate optimization problems without constraints	383
6.1.3	Linear programming problems	386
6.1.4	Quadratic programming problem	388
6.1.5	The constrained nonlinear optimization problems	391
6.1.6	The multi-objective goal attainment problem	395
6.1.7	Semi-infinitely constrained optimization problem	399
6.1.8	The minimax problem	408
6.1.9	Binary integer programming problem	411
6.1.10	A real-coded genetic algorithm for optimization	413
6.1.10.1	Fundamental principles of the real-coded genetic algorithm	413
6.1.10.2	Application of the real-coded genetic algorithm to solve optimization problems ..	417
6.2	Chemical engineering examples	427
	Example 6-2-1 Maximizing the profit of a production system	427
	Example 6-2-2 The optimal photoresist film thickness in a wafer production process ...	431
	Example 6-2-3 Minimal energy of a chemical equilibrium system	432
	Example 6-2-4 Maximal profit of an alkylation process	435
	Example 6-2-5 Maximum separation efficiency in a single-effect distillatory	441
	Example 6-2-6 Dynamic optimization of an ammonia synthesis process	445
	Example 6-2-7 Optimal operating temperature for a tubular reactor	451
6.3	Exercises	456
6.4	Summary of the MATLAB commands related to this chapter	477

Chapter 7: Parameter Estimation	479
7.1 Parameter estimation using the least-squares method	479
7.1.1 Linear least-squares method	480
7.1.2 Nonlinear least-squares methods	486
7.1.3 The confidence interval of parameter estimation	490
7.2 Chemical engineering examples	493
Example 7-2-1 The solubility model of sulfur dioxide	493
Example 7-2-2 The rate equation of a catalyzed reaction system	495
Example 7-2-3 Isothermal adsorption model for activated carbon	499
Example 7-2-4 Transfer function of a heating process	502
Example 7-2-5 Estimation of the reaction rate constants for a fermentation process	506
Example 7-2-6 Parameter estimation for a packed bed reactor described by a partial differential equation model	510
Example 7-2-7 Parameter estimation using D-optimal experimental design method	518
7.3 Exercises	522
7.4 Summary of the MATLAB commands related to this chapter	533
References	535
Index	541



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 1

Solution of a System of Linear Equations

版權所有・請勿翻製

- ① solving a system of linear equations with MATLAB.
- ② the basic concepts and properties of the linear equation systems are first reviewed,
- ③ introduction of the relevant MATLAB commands.
- ④ typical examples are provided to demonstrate how to use MATLAB in the solution of chemical engineering problems that are formulated with a system of linear equations.

1.1 Properties of linear equation systems and the relevant MATLAB commands

- 1.1.1 A simple example: the liquid blending problem

Table 1.1 Volume fraction of components in each tank.

Tank number	Volume fraction of each component (%)				
	A	B	C	D	E
1	55.8	7.8	16.4	11.7	8.3
2	20.4	52.1	11.5	9.2	6.8
3	17.1	12.3	46.1	14.1	10.4
4	18.5	13.9	11.5	47.0	9.1
5	19.2	18.5	21.3	10.4	30.6

- Suppose a customer requests a blended product of 40 liters, in which the volume fraction (%) of each component (A, B, C, D, and E) should be 25, 18, 23, 18, and 16, respectively. Assuming that the density is unchanged, determine the volume usage of each tank to meet the specification.

Mass balance of component A:

$$55.8V_1 + 20.4V_2 + 17.1V_3 + 18.5V_4 + 19.2V_5 = 25 \times 40 \quad (1.1-1a)$$

Mass balance of component B:

$$7.8V_1 + 52.1V_2 + 12.3V_3 + 13.9V_4 + 18.5V_5 = 18 \times 40 \quad (1.1-1b)$$

Mass balance of component C:

$$16.4V_1 + 11.5V_2 + 46.1V_3 + 11.5V_4 + 21.3V_5 = 23 \times 40 \quad (1.1-1c)$$

Mass balance of component D:

$$11.7V_1 + 9.2V_2 + 14.1V_3 + 47.0V_4 + 10.4V_5 = 18 \times 40 \quad (1.1-1d)$$

Mass balance of component E:

$$\left[\begin{array}{ccccc} 55.8 & 20.4 & 17.1 & 18.5 & 19.2 \\ 7.8 & 52.1 & 12.3 & 13.9 & 18.5 \\ 16.4 & 11.5 & 46.1 & 11.5 & 21.3 \\ 11.7 & 9.2 & 14.1 & 47.0 & 10.4 \\ 8.3 & 6.8 & 10.4 & 9.1 & 30.6 \end{array} \right] \left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \right] = \left[\begin{array}{c} 1000 \\ 720 \\ 920 \\ 720 \\ 640 \end{array} \right] \quad (1.1-2a)$$

$$\mathbf{A} \quad \mathbf{x} = \mathbf{b} \quad (1.1-2b)$$

```
>> A=[ 55.8 20.4 17.1 18.5 19.2  
        7.8 52.1 12.3 13.9 18.5  
       16.4 11.5 46.1 11.5 21.3  
      11.7  9.2 14.1 47.0 10.4  
      8.3   6.8 10.4  9.1 30.6]; % matrix A  
>> b=[1000 720 920 720 640]'; % vector b (column vector)  
>> x =A\b % find the solution  
  
x=  
    6.8376  
    4.1795  
    8.6398  
   7.3268  
 13.0163
```

1.1.2 Relevant properties of linear equation systems

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

$$\mathbf{b} = [b_1 \quad b_2 \quad \dots \quad b_m]^T$$

and

$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_n]^T$$

- Existence and uniqueness of solutions:**

The solution of $\mathbf{Ax} = \mathbf{b}$ exists if and only if the following condition holds (*Wylie and Barrett*, 1995)

$$\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A} \ \mathbf{b}])$$

Table 1.2 Solution types of a system of linear equations.

Types of solution	Conditions	Non-homogeneous equations $\mathbf{Ax} = \mathbf{b}$	Homogeneous equations $\mathbf{Ax} = \mathbf{0}$
Case 1	$\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A} \ \mathbf{b}])$ $\text{rank}(\mathbf{A}) = n$	Unique solution	Unique solution $\mathbf{x} = \mathbf{0}$ (trivial solution)
Case 2	$\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A} \ \mathbf{b}])$ $\text{rank}(\mathbf{A}) < n$	Infinite number of solutions	Infinite number of solutions, including nonzero solutions
Case 3	$\text{rank}(\mathbf{A}) < \text{rank}([\mathbf{A} \ \mathbf{b}])$	No solution	

1.1.3 Relevant MATLAB commands

```
>> A=[1 2; 3 4] % input the matrix A to be evaluated  
A=  
    1 2  
    3 4  
>> rank(A) % calculating the rank  
ans =  
    2
```

```
>> A = [1 2; 3 4] % input the matrix A to be transformed  
A =  
    1 2  
    3 4  
>> rref(A)  
ans =  
    1 0  
    0 1
```

```
>> b = [1 2]'
```

```
b =
```

```
1
```

```
2
```

```
>> rank([A b])
```

```
ans =
```

```
2
```

- **Method 1:** finding the solution by an inverse matrix, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

```
>> x = inv (A)*b % as A is a full rank matrix, A-1 exists
```

```
x=
```

```
0
```

```
0.5000
```

- **Method 2:** finding the solution with the MATLAB command
 $x = A \setminus b$

```
>> x = A \ b  
x =  
    0  
    0.5000
```

- **Method 2:** finding the solution with the MATLAB command
 $x = A \setminus b$

```
>> rref([A b])  
ans =  
    1.0000      0      0  
    0    1.0000  0.5000
```

$$\mathbf{x} = [0 \ 0.5]^T.$$

Example 1-1-1

$$\begin{aligned}x + 2y + 3z &= 2 \\x + y + z &= 4\end{aligned}\tag{1.1-6}$$

Ans:

```
>> A=[1 2 3; 1 1 1]; % coefficient matrix A
>> b=[2 4]'; % vector b
>> r1=rank(A)
r1=
    2
>> r2=rank([A b]) % r1=r2 and r1<3, therefore an infinite number of
                      solutions exist.
r2=
    2
```

Example 1-1-1

Ans:

```
>> x=A\b % NOTE: when y is set to 0, one of the solutions is found.
```

```
x=
```

```
5.0000  
0  
-1.0000
```

```
>> x=pinv(A)*b % the solution that is nearest to the origin is found.
```

```
x=
```

```
4.3333  
1.3333  
-1.6667
```

Example 1-1- 2

$$x + y = 1$$

$$x + 2y = 3 \quad (1.1-7)$$

$$x + 5y = a$$

when the parameter a is equal to 9 and 10.

Example 1-1- 2

Ans:

(1) Case 1: $a = 9$

```
>> A=[1 1; 1 2; 1 5];
>> b= [1 3 9]';
>> r1=rank(A)
r1=
    2
>> r2=rank([ A  b])      %      r1=r2 and r1=n=2, the solution is unique
r2=
    2
>> X=A\b
X=
 -1.0000
 2.0000
```

Example 1-1- 2

Ans:

(2) Case 2: $a = 10$

```
>> A=[1 1; 1 2; 1 5];
>> b= [1 3 9]';
>> r1=rank(A)
r1=
    2
>> r2=rank([ A b])      %      r1<r2; no solution theoretically exists
r2=
    3
>> X=A\b
X=
    -1.3846
     2.2692
```

Command	Description
inv(A)	Calculate the inverse matrix of A when $\det(\mathbf{A}) \neq 0$
pinv(A)	Calculate the pseudo inverse matrix of A when $\det(\mathbf{A})=0$
rank(A)	Determine the rank of the matrix A
rref(A)	Transform matrix A into its reduced row Echelon form
x = inv(A)*b	Find the unique solution x
x = pinv(A)*b	Find the solution of x that is closest to the origin
x = A\b	Find the solution x, and the meaning of this solution depends on the characteristics of the system of linear equations

NOTE: In addition, MATLAB has a right divisor by which the simultaneous linear equations $\mathbf{x}\mathbf{A} = \mathbf{b}$ can be solved, where $\mathbf{x} \in R^{1 \times n}$, $\mathbf{A} \in R^{n \times m}$, and $\mathbf{b} \in R^{1 \times m}$. Its usage is $\mathbf{x} = \mathbf{b}/\mathbf{A}$.

1.2 Chemical engineering examples

Example 1-2-1

Composition analysis of a distillation column system

Figure 1.1 schematically illustrates a distillation column system that is used to separate the following four components: *para-xylene*, *styrene*, *toluene*, and *benzene* (Cutlip and Shacham, 1999). Based on the required composition of each exit stream shown in this figure, calculate (a) the molar *flow rates* of D_2 , D_3 , B_2 , and B_3 , and (b) the molar *flow rates* and compositions of streams of D_1 and B_1 .

Problem formulation and analysis:

(a) *overall mass balance*

$$\text{Para-xylene: } 0.07D_2 + 0.18B_2 + 0.15D_3 + 0.24B_3 = 0.15 \times 80$$

$$\text{Styrene: } 0.03D_2 + 0.25B_2 + 0.10D_3 + 0.65B_3 = 0.25 \times 80$$

$$\text{Toluene: } 0.55D_2 + 0.41B_2 + 0.55D_3 + 0.09B_3 = 0.40 \times 80$$

$$\text{Benzene: } 0.35D_2 + 0.16B_2 + 0.20D_3 + 0.02B_3 = 0.20 \times 80$$

$$\mathbf{Ax} = \mathbf{b} \quad (1.2-1)$$

Problem formulation and analysis:

(a) *overall mass balance*

$$\mathbf{A} = \begin{bmatrix} 0.07 & 0.18 & 0.15 & 0.24 \\ 0.03 & 0.25 & 0.10 & 0.65 \\ 0.55 & 0.41 & 0.55 & 0.09 \\ 0.35 & 0.16 & 0.20 & 0.02 \end{bmatrix}$$

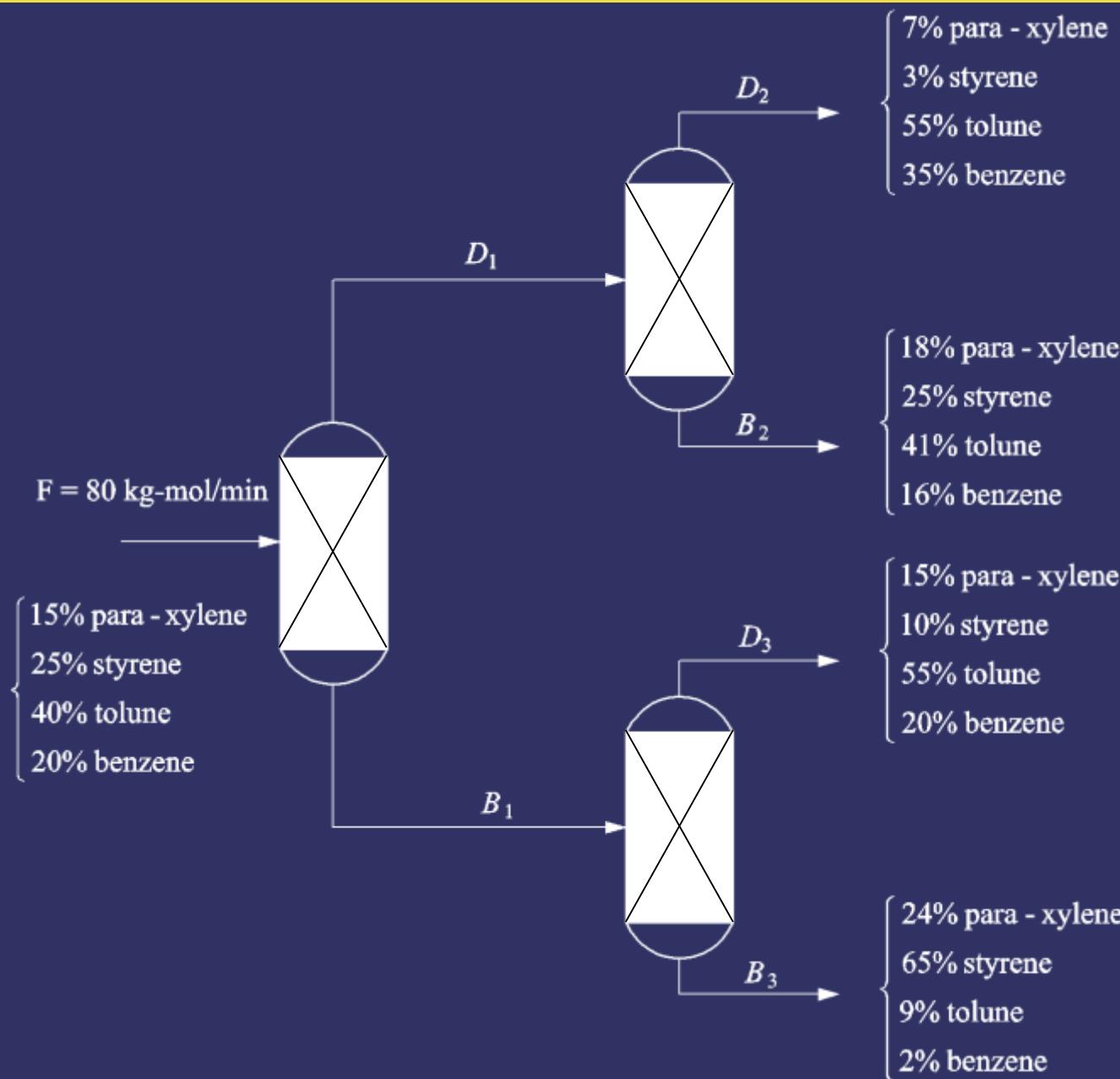
and

$$\mathbf{b} = 80 \times \begin{bmatrix} 0.15 \\ 0.25 \\ 0.40 \\ 0.20 \end{bmatrix}$$

$$\mathbf{x} = [D_2 \ B_2 \ D_3 \ B_3]^T$$

Solution of a System of Linear Equations

01



Problem formulation and analysis:

(b)

$$\text{Para-xylene : } D_1 x_1 = 0.07D_2 + 0.18B_2$$

$$\text{Styrene : } D_1 x_2 = 0.03D_2 + 0.25B_2$$

$$\text{Toluene : } D_1 x_3 = 0.55D_2 + 0.41B_2$$

$$\text{Benzene : } D_1 x_4 = 0.35D_2 + 0.16B_2$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \frac{1}{D_1} \begin{bmatrix} 0.07 & 0.18 \\ 0.03 & 0.25 \\ 0.55 & 0.41 \\ 0.35 & 0.16 \end{bmatrix} \begin{bmatrix} D_2 \\ B_2 \end{bmatrix}$$

$$D_1 = D_2 + B_2$$

Problem formulation and analysis:

(b)

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{bmatrix} = \frac{1}{B_1} \begin{bmatrix} 0.15 & 0.24 \\ 0.10 & 0.65 \\ 0.55 & 0.09 \\ 0.20 & 0.02 \end{bmatrix} \begin{bmatrix} D_3 \\ B_3 \end{bmatrix}$$

$$B_1 = D_3 + B_3$$

MATLAB program design:

— ex1_2_1.m —

```
%  
% Example 1-2-1 Composition analysis of a distillation column system  
%  
clear  
clc  
%  
% given data  
%  
A=[0.07 0.18 0.15 0.24  
    0.03 0.25 0.10 0.65  
    0.55 0.41 0.55 0.09  
    0.35 0.16 0.20 0.02]; % coefficient matrix  
b=80*[0.15 0.25 0.40 0.20]'; % coefficient vector  
%  
r1=rank(A); % the rank of matrix A  
r2=rank([A b]); % the rank of the augmented matrix
```

MATLAB program design:

— ex1_2_1.m —

```
[m,n]=size(A); % determine the size of the matrix A
if (r1==r2)&(r1==n) % check the uniqueness of the solution
%
% solving for D2,B2,D3,B3
%
x=A\b;
D2=x(1); B2=x(2); D3=x(3); B3=x(4);
disp('flow rate at each outlet (kg-mol/min)')
disp('    D2      B2      D3      B3')
disp([D2 B2 D3 B3])
%
% calculating the composition of the stream D1
%
D1=D2+B2;
Dx=A(:,[1 2])*[D2 B2]'/D1;
disp('The composition of the stream D1:')
disp(Dx')
```

MATLAB program design:

— ex1_2_1.m —

```
%  
% calculating the composition of the stream B1  
%B1=D3+B3;  
Bx=A(:,[3 4])*[D3 B3]'/B1;  
disp('The composition of the stream B1:')  
disp(Bx')  
else  
    error('The solution is not unique.')  
    fprintf('\n The rank of matrix A is %i.',rank(A))  
end
```

Execution results:

```
>> ex1_2_1
```

flow rate at each outlet (kg-mol/min)

$D2$	$B2$	$D3$	$B3$
------	------	------	------

29.8343	14.9171	13.7017	21.5470
---------	---------	---------	---------

The composition of the stream $D1$:

0.1067	0.1033	0.5033	0.2867
--------	--------	--------	--------

The composition of the stream $B1$:

0.2050	0.4362	0.2688	0.0900
--------	--------	--------	--------

Example 1-2-2

Temperature analysis of an insulated stainless steel pipeline

Figure 1.2 depicts a stainless steel pipe whose inner and outer diameters are 20 mm and 25 mm, respectively; i.e., $D_1 = 20$ mm and $D_2 = 25$ mm. This pipe is covered with an insulation material whose thickness is 35 mm. The temperature of the saturated vapor (steam) flowing through the pipeline is $T_s = 150^\circ\text{C}$, and the external and internal convective heat transfer coefficients are, respectively, measured to be $h_i = 1,500 \text{ W/m}^2\cdot\text{K}$ and $h_o = 5 \text{ W/m}^2\cdot\text{K}$. Besides, the thermal conductivity of the stainless steel and the insulation material are $k_s = 45 \text{ W/m}\cdot\text{K}$ and $k_i = 0.065$, respectively. If the ambient temperature is $T_a = 25^\circ\text{C}$, determine the internal and external wall temperatures of the stainless steel pipe, T_1 and T_2 , and the exterior wall temperature of the insulation material, T_3 .

Problem formulation and analysis:

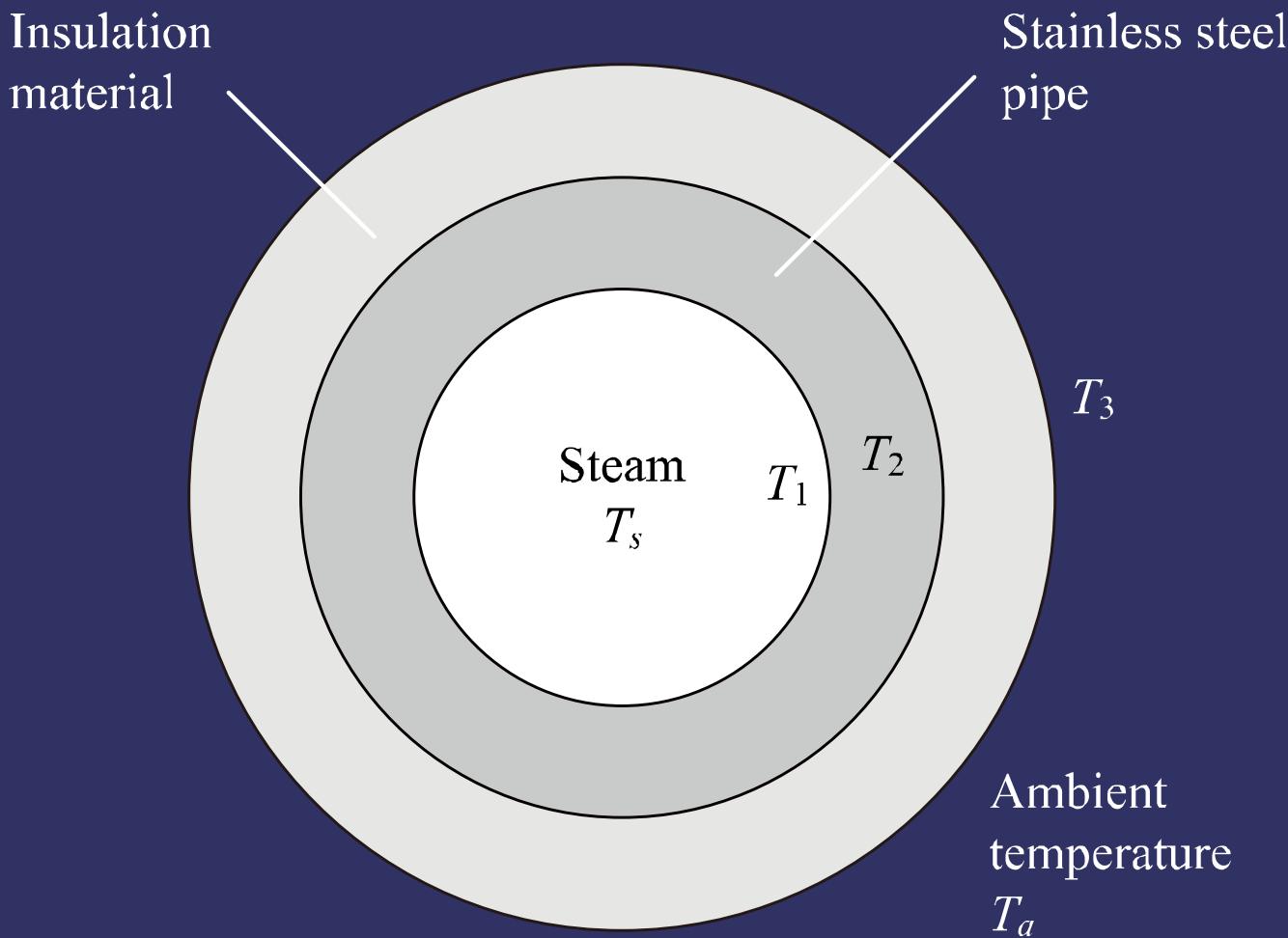


Figure 1.2 A stainless steel pipe covered with an external insulation material.

Problem formulation and analysis:

Heat transfer from steam to pipe

$$h_i \pi D_1 (T_s - T_1) = \frac{T_1 - T_2}{\ln(D_2 / D_1) / (2\pi k_s)}$$

Heat transfer from pipe to the insulation material

$$\frac{T_1 - T_2}{\ln(D_2 / D_1) / (2\pi k_s)} = \frac{T_2 - T_3}{\ln(D_3 / D_2) / (2\pi k_i)}$$

Heat transfer from insulation material to the surroundings

$$\frac{T_2 - T_3}{\ln(D_3 / D_2) / (2\pi k_i)} = h_0 \pi D_3 (T_3 - T_a)$$

Problem formulation and analysis:

$$\left[\frac{2k_s}{\ln(D_2/D_1)} + h_i D_1 \right] T_1 - \left[\frac{2k_s}{\ln(D_2/D_1)} \right] T_2 = h_i D_1 T_s$$

$$\left[\frac{k_s}{\ln(D_2/D_1)} \right] T_1 - \left[\frac{k_s}{\ln(D_2/D_1)} + \frac{k_i}{\ln(D_3/D_2)} \right] T_2 + \left[\frac{k_i}{\ln(D_3/D_2)} \right] T_3 = 0$$

$$\left[\frac{2k_i}{\ln(D_3/D_2)} \right] T_2 - \left[\frac{2k_i}{\ln(D_3/D_2)} + h_0 D_3 \right] T_3 = -h_0 D_3 T_a$$

MATLAB program design:

— ex1_2_2.m —

%

% Example 1-2-2 Temperature analysis of an insulated pipeline

%

clear

clc

%

% given data

%

D1=20/1000; % inner diameter (converting mm to m)

D2=25/1000; % outer diameter (converting mm to m)

DI=35/1000; % thickness of insulating materials (m)

MATLAB program design:

— ex1_2_2.m —

Ts=150+273; % vapor temperature (K)

Ta=25+273; % ambient air temperature (K)

hi=1500; interior convective heat transfer coefficient (W/m^2.K)

ho=5; exterior convective heat transfer coefficient (W/m^2.K)

ks=45; heat conductivity coefficient of the steel pipe (W/m.K)

ki=0.065; heat conductivity coefficient of the insulation material (W/m.K)

%

D3=D2+2*DI; % the total diameter including the insulation material

%

% coefficient matrix A

%

MATLAB program design:

— ex1_2_2.m —

```
A=[2*ks/log(D2/D1)+hi*D1 -2*ks/log(D2/D1) 0  
    ks/log(D2/D1) -ks/log(D2/D1)-ki/log(D3/D2) ki/log(D3/D2)  
    0 2*ki/log(D3/D2) -2*ki/log(D3/D2)-ho*D3];
```

%

```
% coefficient vector b
```

%

```
b=[hi*D1*Ts 0 -ho*D3*Ta]';
```

%

```
% check the singularity
```

%

```
if det(A)==0
```

MATLAB program design:

— ex1_2_2.m —

```
fprintf('\n rank of matrix A=%i', rank(A))  
error('Matrix A is a singular matrix.') % exit in case of error  
end  
%  
% det(A) ~= 0, a unique solution exists  
%  
T=A\b; % find the solution  
%  
% results printing  
%  
fprintf('\n T1=% .3f °C \n T2=% .3f °C \n T3=% .3f °C \n',T-273)
```

Execution results:

```
>> ex1_2_2
```

$T1=149.664 \text{ } ^\circ\text{C}$

$T2=149.639 \text{ } ^\circ\text{C}$

$T3=46.205 \text{ } ^\circ\text{C}$

Example 1-2-3

Composition analysis of a set of CSTRs

Figure 1.3 schematically illustrates a set of continuous stirred-tank reactors (CSTRs) in which the reaction $A \xrightarrow{k_j} B$ is occurring in the tank i (*Constantinides and Mostoufi, 1999*).

Example 1-2-3

Composition analysis of a set of CSTRs

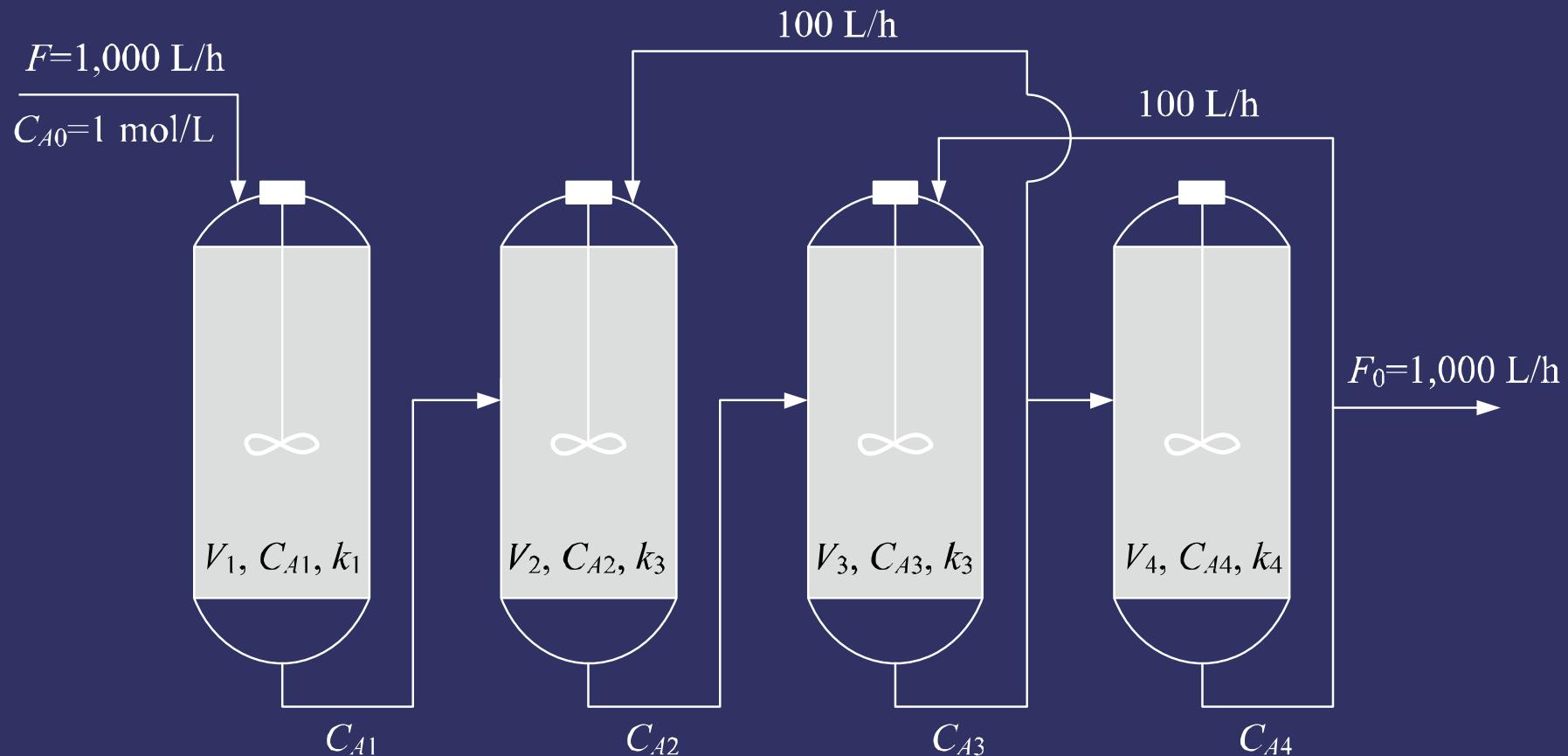


Figure 1.3 A set of continuous stirred-tank reactors.

Tank number	Volume V_i (L)	Reaction constant k_i (h^{-1})
1	1,000	0.2
2	1,200	0.1
3	300	0.3
4	600	0.5

- Furthermore, this reaction system is assumed to possess the following properties:
 - 1) The reaction occurs in the liquid phase, and its *steady state* has been reached.
 - 2) The changes of liquid volumes and density variations in the reaction tanks are negligible.
 - 3) The reaction in the tank i obeys the first-order reaction kinetics and the reaction rate equation is expressed as follows:

$$r_i = V_i k_i C_{Ai} \quad (1.2-8)$$

- Determine the exit concentration of each reaction tank, i.e., $C_{Ai} = ?$ for $i = 1, 2, 3$, and 4 .

Problem formulation and analysis:

input = output + quantity lost due to reaction

$$\text{Tank 1: } 1,000C_{A0} = 1,000C_{A1} + V_1k_1C_{A1} \quad (1.2-9a)$$

$$\text{Tank 2: } 1,000C_{A1} + 100C_{A3} = 1,100C_{A2} + V_2k_2C_{A2} \quad (1.2-9b)$$

$$\text{Tank 3 } 1,100C_{A2} + 100C_{A4} = 1,200C_{A3} + V_3k_3C_{A3} \quad (1.2-9c)$$

$$\text{Tank 4: } 1,100C_{A3} = 1,100C_{A4} + V_4k_4C_{A4} \quad (1.2-9d)$$

$$\begin{bmatrix} 1,000 + V_1k_1 & 0 & 0 & 0 \\ 1,000 & -(1,100 + V_2k_2) & 100 & 0 \\ 0 & 1,100 & -(1,200 + V_3k_3) & 100 \\ 0 & 0 & 1,100 & -(1,100 + V_4k_4) \end{bmatrix} \begin{bmatrix} C_{A1} \\ C_{A2} \\ C_{A3} \\ C_{A4} \end{bmatrix} = \begin{bmatrix} 1,000 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

MATLAB program design:

— ex1_2_3.m —

```
%  
% Example 1-2-3 Composition analysis of a set of CSTRs  
%  
clear  
clc  
%  
% given data  
%  
V=[1000 1200 300 600]; % volume of each tank (L)  
k=[0.2 0.1 0.3 0.5]; % reaction constant in each tank (1/h)  
%  
V1=V(1); V2=V(2); V3=V(3); V4=V(4);  
k1=k(1); k2=k(2); k3=k(3); k4=k(4);  
%  
% the coefficient matrix  
%  
A=[1000+V1*k1 0 0 0
```

MATLAB program design:

— ex1_2_3.m —

```
1000 -(1100+V2*k2) 100 0
  0 1100 -(1200+V3*k3) 100
  0 0 1100 -(1100+V4*k4)];
%
% the coefficient vector b
%
b=[1000 0 0 0]';
%
% check whether A is a singular matrix
%
if det(A) == 0
    fprintf('\n rank=%i \n', rank(A))
    error('Matrix A is a singular matrix.')
end
%
% finding a solution%
```

MATLAB program design:

— ex1_2_3.m —

```
Ab=rref([A b]);
CA=Ab(:, length(b)+1);
%
% results printing
%
disp('The exit concentration of each reactor is (mol/L):')
for i=1:length(b)
    fprintf('CA(%d)=%5.3f \n',i,CA(i))
end
```

Execution results:

```
>> ex1_2_3
```

The exit concentration of each reactor is (mol/L):

CA(1)=0.833

CA(2)=0.738

CA(3)=0.670

CA(4)=0.527

Example 1-2-4

Independent reactions in a reaction system



$$\text{C}_2\text{H}_4 + \text{H}_2 - \text{C}_2\text{H}_6 = 0$$

$$\text{C}_2\text{H}_4 + 2\text{CH}_4 - 2\text{C}_2\text{H}_6 = 0$$

Let A_1 , A_2 , A_3 , and A_4 denote C_2H_4 , H_2 , CH_4 , and C_2H_6 respectively.

$$A_1 + A_2 - A_4 = 0$$

$$A_1 + 2A_3 - 2A_4 = 0$$

$$\begin{bmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 2 & -2 \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
>> A=[1 1 0 -1; 1 0 2 -2];
```

```
>> rank(A)
```

ans=

2

$$\sum_{j=1}^s a_{ij} A_j = 0, \quad i = 1, 2, \dots, R$$



Problem formulation and the MATLAB solution:

Let NH_3 , O_2 , N_2 , H_2O , NO_2 , and NO be respectively denoted as

$$A_1 - A_6$$

$$\mathbf{A} = \begin{bmatrix} -4 & -5 & 0 & 6 & 0 & 4 \\ -4 & -3 & 2 & 6 & 0 & 0 \\ -4 & 0 & 5 & 6 & 0 & -6 \\ 0 & -1 & 0 & 0 & 2 & -2 \\ 0 & 1 & 1 & 0 & 0 & -2 \\ 0 & -2 & -1 & 0 & 2 & 0 \end{bmatrix}$$

Problem formulation and the MATLAB solution:

```
>> A=[-4      -5      0      6      0      4  
       -4      -3      2      6      0      0  
       -4      0      5      6      0      -6  
       0      -1      0      0      2      -2  
       0      1      1      0      0      -2  
       0      -2      -1      0      2      0      ];
```

```
>> r=rank(A)
```

r=

3

Problem formulation and the MATLAB solution:

```
>> [a, jb]=rref(A') % determine the independent reactions based  
on  $A^T$ 
```

a=

$$\begin{array}{ccccccc} 1.0000 & & 0 & -1.5000 & & 0 & -0.5000 \\ & 0 & 1.0000 & 2.5000 & & 0 & 0.5000 \\ & 0 & 0 & 0 & 1.0000 & & 0 & 1.0000 \\ & 0 & 0 & 0 & 0 & & 0 & 0 \\ & 0 & 0 & 0 & 0 & & 0 & 0 \\ & 0 & 0 & 0 & 0 & & 0 & 0 \end{array}$$

jb=

$$\begin{matrix} 1 & 2 & 4 \end{matrix}$$

Example 1-2-4

Composition distributions in a distillation column

Figure 1.4 schematically depicts a distillation column which has a *total condenser* at the top, a *reboiler* at the bottom, and the *side streams* for liquid removal. In a two-component system, the equilibrium relationship can be expressed by the following equation (*Wang and Henke, 1966*):

$$y_1^M = \frac{\alpha_1 x_1}{\alpha_1 x_1 + \alpha_2 x_2} \quad (1.2-13)$$

where x_1 and x_2 represent, respectively, the concentration of component 1 (the one with a lower boiling point) and that of component 2 (the one having a higher boiling point) in the liquid phase, and y_1^M denotes the concentration of component 1 in the gas phase. Suppose the parameters in (1.2-13) are estimated to be $\alpha_1 = 2.0$ and $\alpha_2 = 1.0$ and the operating conditions are as follows:

- 1) The distillation column contains 10 plates, and the raw material, which is fed into the fifth plate, is with a flow rate of $F = 1.0$ and has the q value of 0.5. Besides, the feed composition is $z_1 = 0.5$ and $z_2 = 0.5$.
- 2) The reflux ratio is $R = 3$.
- 3) The distillate is $D = 0.5$ and the bottom product is $B = 0.5$.

Based on the above-mentioned operating conditions, determine the composition distribution on each plate of the distillation column.

Problem formulation and analysis:

(1) Mass balance for component i on plate j

$$L_{j-1}x_{i,j-1} + V_{j+1}y_{i,j+1} + F_jz_{ij} - (V_j + W_j)y_{ij} - (L_j + U_j)x_{ij} = 0$$

Variable	Description
V_j	Amount of vapor moving upward from plate j (excluding W_j)
W_j	Amount of vapor removed from plate j
L_j	Amount of liquid moving downward from plate j (excluding U_j)
U_j	Amount of liquid removed from plate j
x_{ij}	Concentration of component i in the liquid phase on plate j
y_{ij}	Concentration of component i in the vapor phase on plate j
F_j	Amount of materials fed into plate j
z_{ij}	Concentration of component i in the raw materials fed into plate j



Problem formulation and analysis:

(2) The overall mass balance around the total condenser and plate j

$$V_{j+1} + \sum_{k=2}^j F_k = D + L_j + \sum_{k=2}^j (W_k + U_k), \quad 2 \leq j \leq N-1,$$

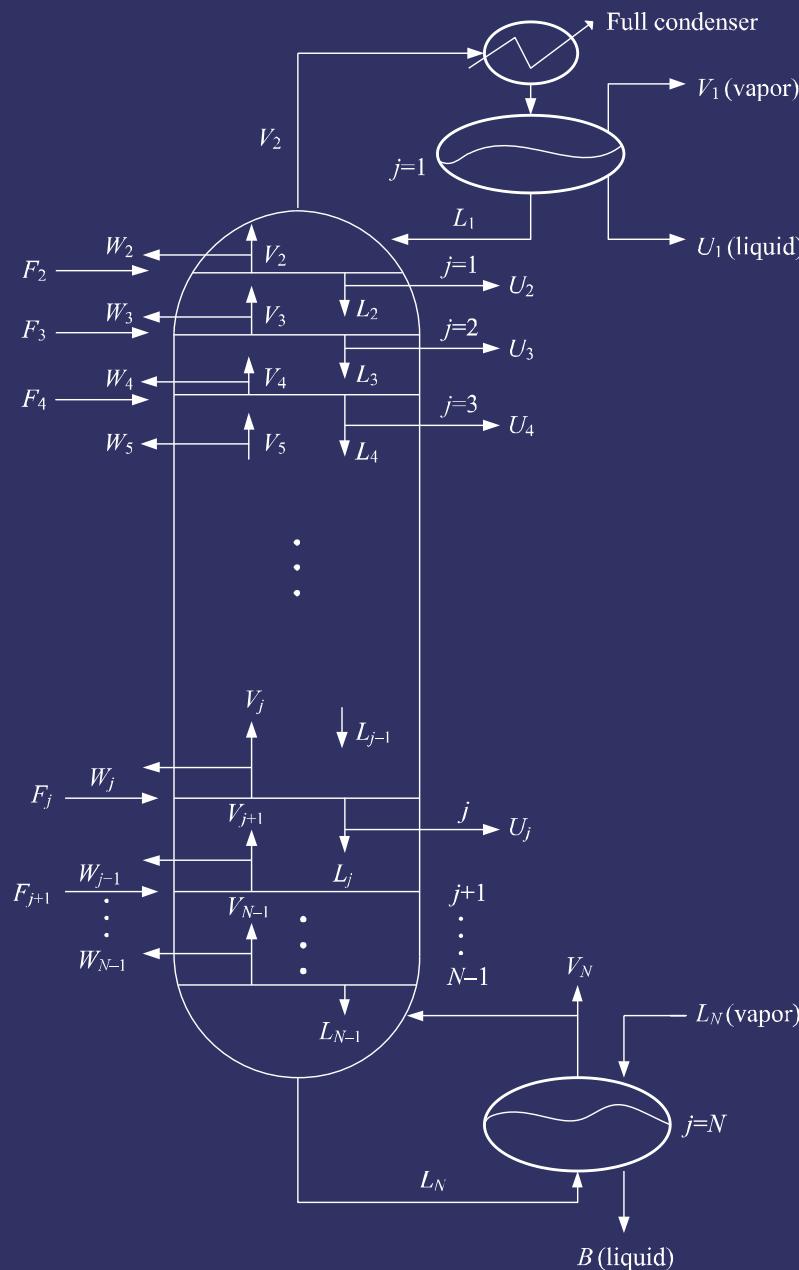
where

$$D = V_1 + U_1$$

Solution of a System of Linear Equations

01

Figure 1.4 A distillation column with side streams.



Problem formulation and analysis:

(3) The gas and liquid flow rates affected by the q -value

$$V_{j+1}(1+q)F_j = V_j + W_j$$

$$L_{j-1} + qF_i = L_j + U_j$$

$$K_{ij} = \frac{y_{ij}}{x_{ij}}$$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 \\ \ddots & \ddots & \ddots & & \\ 0 & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \\ x_{i,N-1} \\ x_{iN} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix}$$

Problem formulation and analysis:

(3) The gas and liquid flow rates affected by the q -value

$$a_j = L_{j-1}$$

$$b_j = -(V_j + W_j)K_{ij} - L_j - U_j$$

$$c_j = V_{j+1}K_{i,j+1}$$

$$d_j = -F_i z_{i,j}$$

for $j = 2, 3, \dots, N - 1$ and those for $j = N$ are

$$a_N = L_{N-1}$$

$$b_N = -V_N K_{iN} - B$$

$$d_N = -F_N z_{iN}$$

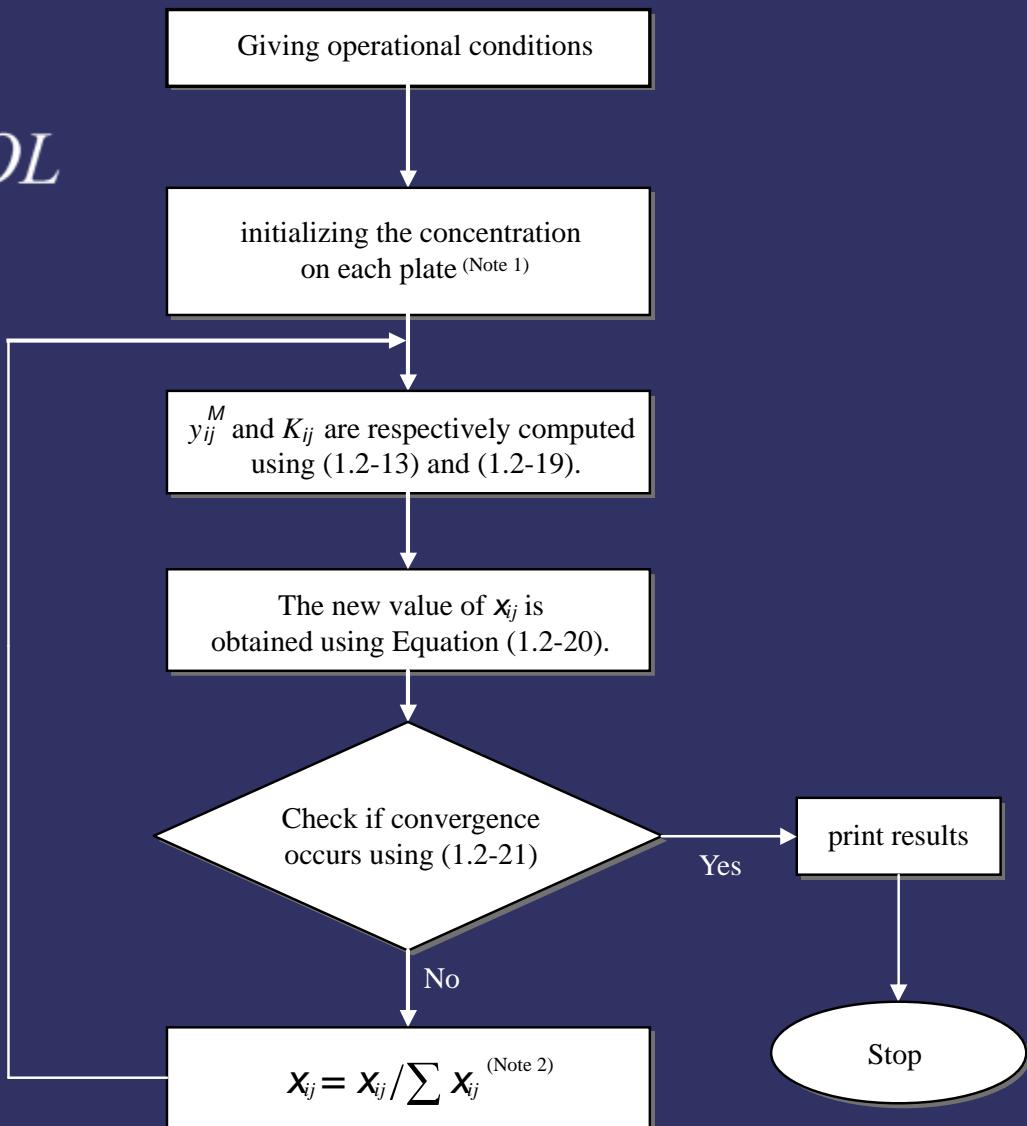
$$b_1 = -L_1 - V_1 K_{i1} - U_1$$

$$c_1 = V_2 K_{i2}$$

$$d_1 = 0$$

MATLAB program design:

$$\sum_{i=1}^m \sum_{j=1}^N |y_{ij}^M - K_{ij}x_{ij}| \leq TOL$$



MATLAB program design:

— ex1_2_5.m —

```
%  
% Example 1-2-5: composition distributions in a distillation column  
%  
clear; close all;  
clc  
%  
% given data  
%  
N=10; % total number of plates (including the total condenser and reboiler)  
m=2; % no. of components  
q=0.5; % the q value  
F=1; % flow rate of the feed  
z=[0.5 0.5]; % composition of the feed  
R=3; % reflux ratio  
D=0.5; % molar flow rate of the distillate  
B=0.5; % molar flow rate of the bottom product  
JF=5; % position of the feed plate  
alpha1=2; % equilibrium constant in equation (1.2-13)
```

MATLAB program design:

— ex1_2_5.m —

```
alpha2=1; % equilibrium constant in equation (1.2-13)
```

```
TOL=1.e-5; % tolerance value
```

```
%
```

```
% operating data of the distillation column
```

```
%
```

```
% side streams
```

```
%
```

```
FI=zeros(1,N);
```

```
FI(JF)=F; % the feed flow rate at the feed plate
```

```
%
```

```
% composition of the side stream inlets
```

```
%
```

```
zc=zeros(m,N);
```

```
zc(:,JF)=z'; % composition of the feed at the feed plate
```

```
%
```

```
% flow rate at the side stream outlets, W and U
```

```
%
```

MATLAB program design:

— ex1_2_5.m —

```
W=zeros(1, N);
U=zeros(1, N);
U(1)=0.5; % output at the top
%
% initial concentration change of each plate
%
x=zeros(m, N);
y=x;
%
for i=1:m
    x(i,:)=z(m)*ones(1,N); % initial guess values
end
%
% vapor and liquid flow rates at each plate
%
Q=zeros(1,N);
Q(JF)=q; % status of the feed
```

MATLAB program design:

— ex1_2_5.m —

```
L(1)=R*D;
V(1)=D-U(1);
V(2)=L(1)+V(1)+U(1);
FL=FI.*Q;
FV=FI.*(1-Q);
for j=2:N-1
    L(j)=L(j-1)+FL(j)-U(j);
end
L(N)=B;
for j=3:N
    V(j)=V(j-1)-FV(j-1)+W(j-1);
end
%
%
ICHECK=1; % stopping flag: 1= not convergent yet
it_no=1;
%
% iteration begins
```

MATLAB program design:

— ex1_2_5.m —

```
%  
while ICHECK == 1  
%  
% normalization of concentration  
%  
for i=1:m  
    x(i,:)=x(i,:)./sum(x);  
end  
%  
% calculating the vapor phase equilibrium composition using (1.2-13)  
%  
y(1,:)=alpha1*x(1,:)/(alpha1*x(1,:)+alpha2*x(2,:));  
y(2,:)=1-y(1,:);  
%  
% calculating the equilibrium constant  
%  
y(1,:)=alpha1*x(1,:)/(alpha1*x(1,:)+alpha2*x(2,:));  
y(2,:)=1-y(1,:);
```

MATLAB program design:

— ex1_2_5.m —

```
%  
% calculating the equilibrium constant  
%  
K=y./x;  
%  
% finding the liquid phase concentration  
%  
for i=1:m  
    for j=2:N-1  
        a(j)=L(j-1);  
        b(j)=-(V(j)+W(j))*K(i,j)-L(j)-U(j);  
        c(j)=V(j+1)*K(i,j+1);  
        d(j)=-FI(j)*zc(i,j);  
    end  
    b(1)=-L(1)-V(1)*K(i,1)-U(1);  
    c(1)=V(2)*K(i,2);  
    d(1)=0;
```

MATLAB program design:

— ex1_2_5.m —

```
a(N)=L(N-1);  
b(N)=-V(N)*K(i,N)-B;  
d(N)=-FI(N)*zc(i,N);  
%  
% forming the coefficient matrix  
%  
A=zeros(N,N);  
A(1,:)=[b(1) c(1) zeros(1,N-2)];  
A(2,:)=[a(2) b(2) c(2) zeros(1,N-3)];  
for j=3:N-2  
    A(j,:)=[zeros(1,j-2) a(j) b(j) c(j) zeros(1,N-j-1)];  
end  
A(N-1,:)=[zeros(1,N-3) a(N-1) b(N-1) c(N-1)];  
A(N,:)=[zeros(1,N-2) a(N) b(N)];  
if det(A) == 0    % check the singularity of matrix A  
    error('det(A) = 0, singular matrix')  
end
```

MATLAB program design:

— ex1_2_5.m —

%

```
% finding a solution
```

```
%
```

```
x(i,:)=(A\d')';
```

```
end
```

```
%
```

```
% checking convergence
```

```
%
```

```
y_eq=K.*x;
```

```
%
```

```
y(1,:)=alpha1*x(1,:)/(alpha1*x(1,:)+alpha2*x(2,:));
```

```
y(2,:)=1-y(1,:);
```

```
%
```

```
difference=abs(y-y_eq);
```

```
err=sum(sum(difference)); % error
```

```
%
```

```
if err<= TOL
```

```
ICHECK=0; % convergence has been reached
```

MATLAB program design:

— ex1_2_5.m —

```
else
    it_no=it_no+1; % not yet convergent and do next iteration
end
end
%
% results printing
%
fprintf('\n iteration no.=%i\n',it_no)
disp('composition at each plate:')
disp('plate no.      x1      x2      y1      y2')
for j=1:N
    fprintf('%3i  %15.5f  %10.5f  %10.5f %10.5f\n',j,x(1,j),x(2,j),y(1,j),y(2,j))
end
%
% results plotting
%
figure(1)
```

MATLAB program design:

— ex1_2_5.m —

```
plot(1:N,x(1,:),'r',1:N,x(2,:),'-.b')
xlabel('plate number')
ylabel('x1 and x2')
legend('x1','x2')
title('liquid phase composition at each plate')
%
figure(2)
plot(1:N,y(1,:),'r',1:N,y(2,:),'-.b')
xlabel('plate number')
ylabel('y1 and y2')
legend('y1','y2')
title('vapor phase composition at each plate')
```

Execution results:

>> ex1_2_5

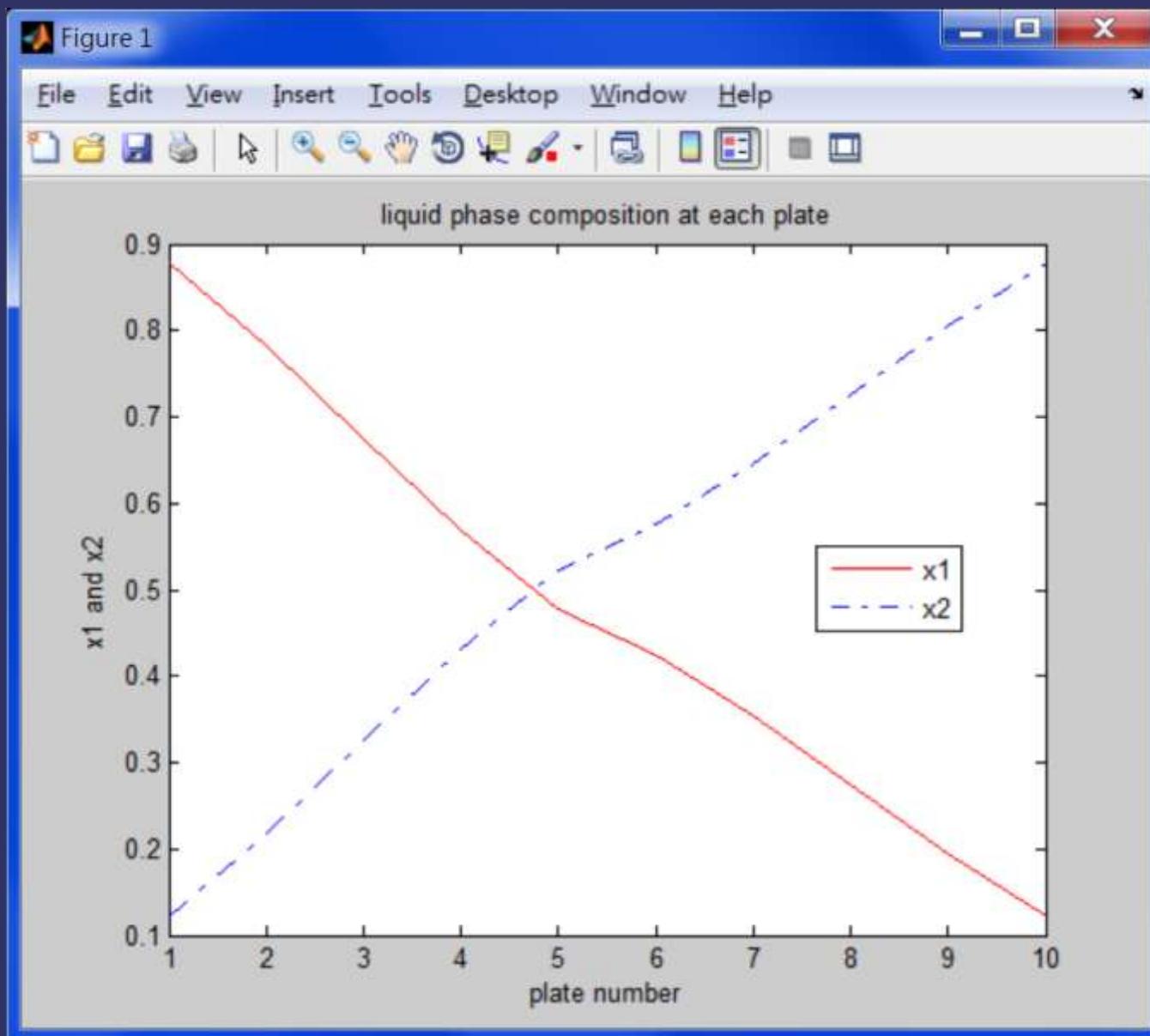
iteration no.=15

composition at each plate:

plate no.	x1	x2	y1	y2
1	0.87723	0.12277	0.93460	0.06540
2	0.78131	0.21869	0.87723	0.12277
3	0.67405	0.32596	0.80529	0.19471
4	0.56843	0.43157	0.72484	0.27516
5	0.47670	0.52330	0.64563	0.35437
6	0.42316	0.57684	0.59468	0.40532
7	0.35437	0.64563	0.52330	0.47670
8	0.27516	0.72484	0.43157	0.56843
9	0.19471	0.80529	0.32595	0.67405
10	0.12277	0.87723	0.21869	0.78131

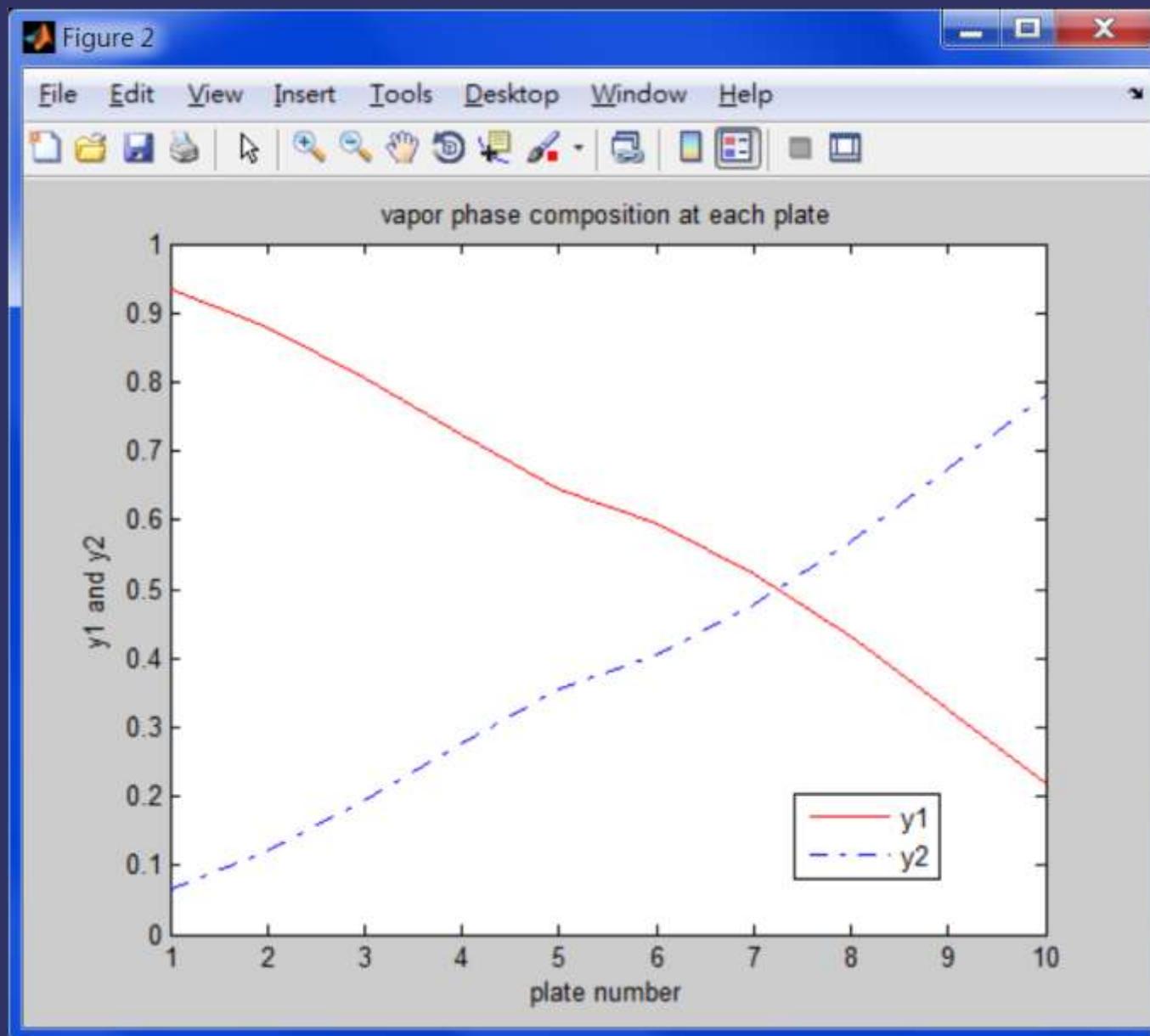
Solution of a System of Linear Equations

01



Solution of a System of Linear Equations

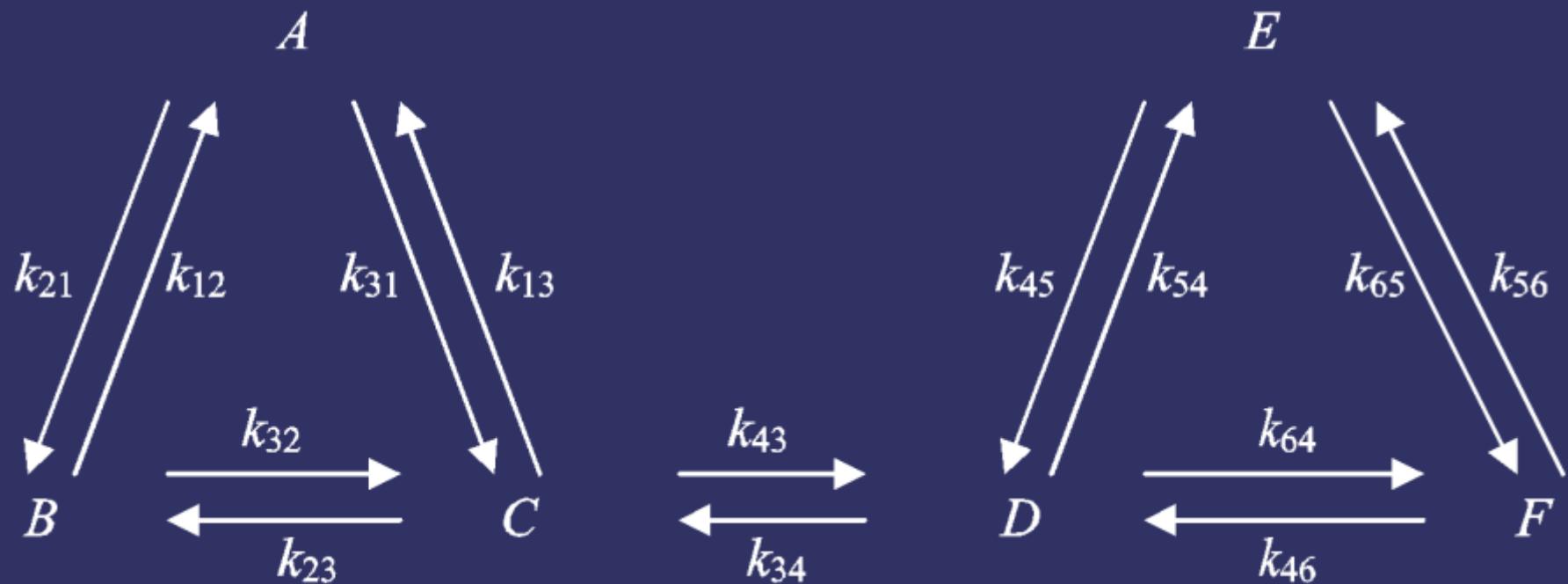
01



Example 1-2- 6

Steady-state analysis of a batch reaction system

Consider a batch reactor in which the following reactions are occurring (*Constantinides and Mostoufi, 1999*):



All the reactions obey the first-order kinetic mechanism, and, under the operating condition of constant pressure and temperature, the reaction rates are listed in the following table:

k_{21}	k_{31}	k_{32}	k_{34}	k_{54}	k_{64}	k_{65}
0.1	0.1	0.1	0.1	0.05	0.2	0.15
k_{12}	k_{13}	k_{23}	k_{43}	k_{45}	k_{46}	k_{56}
0.2	0.05	0.05	0.2	0.1	0.2	0.2

Besides, the initial concentration (mol/L) of each component is as follows:

A_0	B_0	C_0	D_0	E_0	F_0
1.5	0	0	0	1.0	0

Calculate the concentration of each component as a steady state is reached.

Problem formulation and analysis:

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_6] = [A \ B \ C \ D \ E \ F];$$

$$\frac{dx_1}{dt} = -(k_{21} + k_{31})x_1 + k_{12}x_2 + k_{13}x_3$$

$$\frac{dx_2}{dt} = k_{21}x_1 - (k_{12} + k_{32})x_2 + k_{23}x_3$$

$$\frac{dx_3}{dt} = k_{31}x_1 + k_{32}x_2 - (k_{13} + k_{23} + k_{43})x_3 + k_{34}x_4$$

$$\frac{dx_4}{dt} = k_{43}x_3 - (k_{34} + k_{54} + k_{64})x_4 + k_{45}x_5 + k_{46}x_6$$

$$\frac{dx_5}{dt} = k_{54}x_4 - (k_{45} + k_{65})x_5 + k_{56}x_6$$

Problem formulation and analysis:

$$\frac{dx_6}{dt} = k_{64}x_4 + k_{65}x_5 - (k_{46} + k_{56})x_6$$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} -(k_{21} + k_{31}) & k_{12} & k_{13} & 0 & 0 & 0 \\ k_{21} & -(k_{12} + k_{32}) & k_{23} & 0 & 0 & 0 \\ k_{31} & k_{32} & -(k_{13} + k_{23} + k_{43}) & k_{34} & 0 & 0 \\ 0 & 0 & k_{43} & -(k_{34} + k_{54} + k_{64}) & k_{45} & k_{46} \\ 0 & 0 & 0 & k_{54} & -(k_{45} + k_{65}) & k_{56} \\ 0 & 0 & 0 & k_{64} & k_{65} & -(k_{46} + k_{56}) \end{bmatrix}$$

Problem formulation and analysis:

$$\mathbf{x} = \mathbf{V} e^{\lambda t}$$

$$\mathbf{A}\mathbf{V} = \lambda\mathbf{V}$$

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{V} = 0$$

$$|\mathbf{A} - \lambda\mathbf{I}| = 0$$

$$\mathbf{x} = \sum_{i=1}^6 C_i \mathbf{V}_i e^{\lambda_i t}$$

$$\mathbf{x}(0) = \sum_{i=1}^6 C_i \mathbf{V}_i$$

$$= [\mathbf{V}_1 \ \mathbf{V}_2 \ \dots \ \mathbf{V}_6] \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_6 \end{bmatrix} \\ = \mathbf{A}_V \mathbf{C}$$

$$\mathbf{C} = \mathbf{A}_V^{-1} \mathbf{x}(0)$$

MATLAB program design:

— ex1_2_6.m —

```
%  
% Example 1-2-6 Steady-state analysis of a batch reaction system  
%  
clear  
clc  
close all  
%  
% given data  
%  
k21=0.1; k12=0.2; k31=0.1; k13=0.05; k32=0.1; k23=0.05; k34=0.1;  
k43=0.2; k54=0.05; k45=0.1; k64=0.2; k46=0.2; k65=0.15; k56=0.2;  
%  
% initial concentration  
%  
x0=[1.5 0 0 0 1 0]';  
%  
% Coefficient matrix  
%
```

MATLAB program design:

— ex1_2_6.m —

```

A=[-(k21+k31) k12 k13          0 0 0
   k21 -(k12+k32) k23          0 0 0
   k31      k32 -(k13+k23+k43) k34 0 0
   0        0    k43 -(k34+k54+k64) k45 k46
   0        0    0      k54 -(k45+k65) k56
   0        0    0      k64 k65 -(k46+k56)];
%
[m,n]=size(A);
[v,d]=eig(A); % the eigenvalue and eigenvector of A
%
lambda=diag(d); % eigenvalue
%
c=v\x0; % finding the coefficient vector C
%
check=abs(lambda)<=eps; % check if the eigenvalue is close to 0
%
% steady state value computation

```

MATLAB program design:

— ex1_2_6.m —

```
%  
C=c.*check;  
x_final=v*C;  
%  
disp(' ')  
disp('steady-state concentration of each component (mol/L)')  
disp('    A      B      C      D      E      F')  
disp(x_final')
```

Execution results:

```
>> ex1_2_6
```

steady-state concentration of each component (mol/L)

A	B	C	D	E	F
0.2124	0.1274	0.3398	0.6796	0.5825	0.5583

NOTE:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

$$\mathbf{x}(t) = e^{\mathbf{A}t} \mathbf{x}(0)$$

$$\text{expm}(\mathbf{A}^*t)$$

MATLAB program design:

— ex1_2_6b.m —

```
%  
% Solve Example 1-2-6 with the matrix function method  
%  
clear; close all;  
clc  
%  
% given data  
%  
k21=0.1; k12=0.2; k31=0.1; k13=0.05; k32=0.1; k23=0.05; k34=0.1;  
k43=0.2; k54=0.05; k45=0.1; k64=0.2; k46=0.2; k65=0.15; k56=0.2;  
%  
% initial concentration  
%  
x0=[1.5 0 0 0 1 0]';  
%  
% forming the coefficient matrix  
%  
A=[-(k21+k31) k12 k13 0 0 0
```

MATLAB program design:

ex1_2_6b.m

```

k21  -(k12+k32)  k23          0      0      0
k31      k32  -(k13+k23+k43)  k34      0      0
0      0      k43  -(k34+k54+k64)  k45      k46
0      0      0      k54  -(k45+k65)  k56
0      0      0      k64  k65  -(k46+k56)];
%
t=0; %
h=1; % time increment
x_old=x0;
i_check=1;
tout=0;
xout=x0';
while i_check == 1 % continues if not yet converged
    t=t+h;
    x_new=expm(A*t)*x0; % calculating the state value
    if max(x_new-x_old) <= 1.e-6 % check for convergence
        i_check=0;
    end
end

```

MATLAB program design:

- ex1_2_6b.m -

```
tout=[tout;t];
xout=[xout; x_new'];
x_old=x_new;
end
disp(' ')
disp('steady-state concentration of each component (mol/L)')
disp('    A      B      C      D      E      F')
disp(x_new')
%
% plot the results
%
plot(tout,xout')
xlabel('time')
ylabel('states')
legend('A','B','C','D','E','F')
```

Execution results:

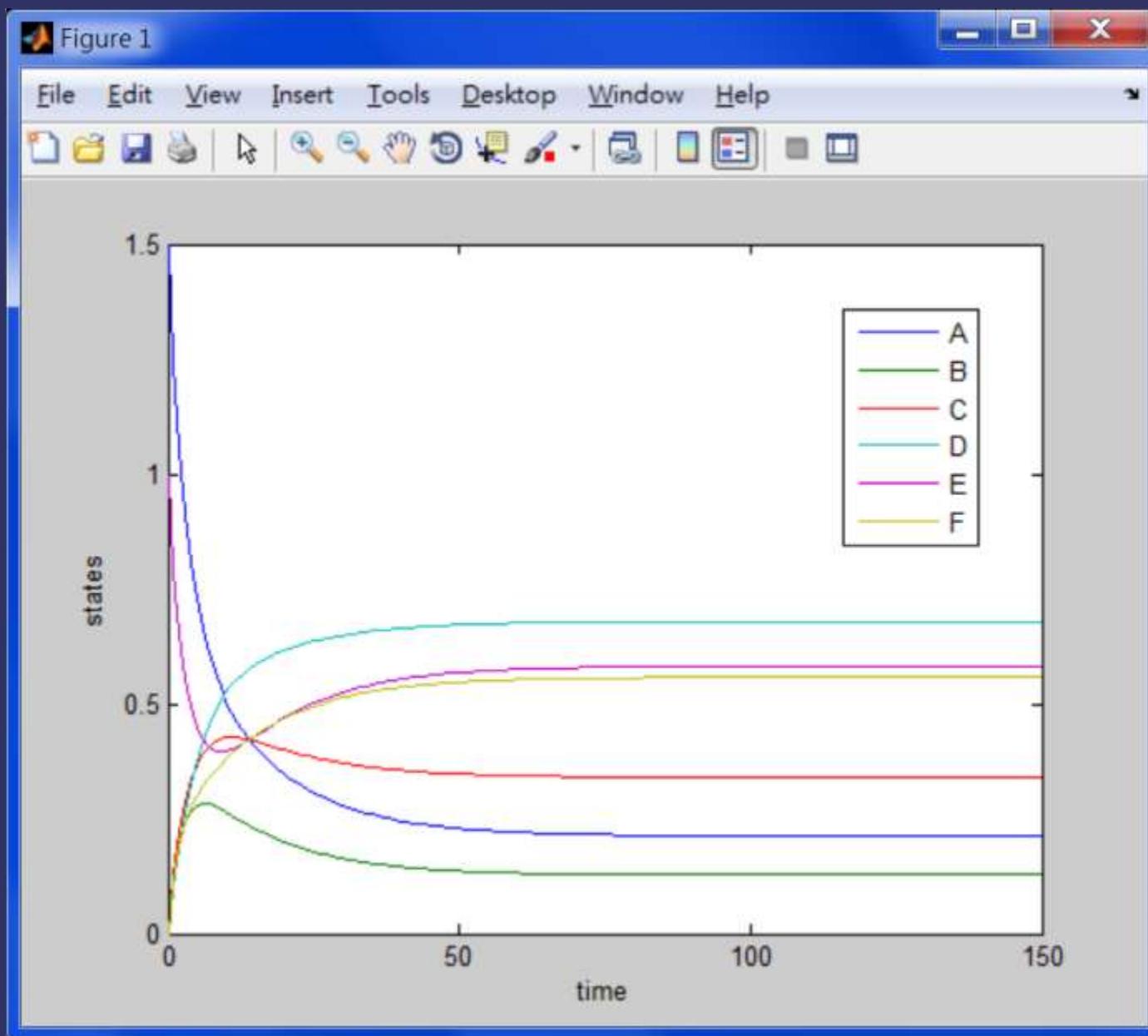
```
>> ex1_2_6b
```

steady-state concentration of each component (mol/L)

A	B	C	D	E	F
0.2124	0.1274	0.3398	0.6796	0.5825	0.5582

Solution of a System of Linear Equations

01



1.4 Summary of the MATLAB commands related to this chapter

Command	Function
det	Matrix determinant
rank	Rank of a matrix
rref	Reduced row Echelon form of a matrix
\	Solution of the linear equation system $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} = ?$
/	Solution of the linear equation system $\mathbf{xA} = \mathbf{b}$, $\mathbf{x} = ?$
inv	Matrix inverse
pinv	The Moore-Penrose pseudo inverse of a matrix



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 2

Solution of Nonlinear Equations

版權所有・請勿翻製

- ① Finding the solution to a nonlinear equation or a system of nonlinear equations is a common problem arising from the analysis and design of chemical processes in steady state.
- ② the relevant MATLAB commands used to deal with this kind of problem are introduced.
- ③ the solution-finding procedure through using the Simulink simulation interface will also be demonstrated.

2.1 Relevant MATLAB commands and the Simulink solution interface

► 2.1.1 Nonlinear equation of a single variable

$$f(x) = 0$$

- For example, one intends to obtain the volume of 1 g-mol propane gas at $P = 10$ atm and $T = 225.46$ K from the following Van der Waals P-V-T equation:

$$f(V) = \left(P + \frac{a}{V^2} \right)(V - b) - RT = 0$$

2.1.1 Nonlinear equation of a single variable

- where $a = 8.664 \text{ atm (L/g-mol)}$, $b = 0.08446 \text{ L/g-mol}$, and the ideal gas constant R is $0.08206 \text{ atm}\cdot\text{L/g-mol}\cdot\text{K}$.

```
x = fzero('filename', x0)
```

2.1.1 Nonlinear equation of a single variable

PVT.m

```
function f=PVT(V)
%
% Finding the solution to the Van der Waals equation (2.1-2)
%
% given data
%
a=8.664; % constant, atm (L/g-mol)^2
b=0.08446; % constant, L/g-mol
R=0.08206; % ideal gas constant, atm-L/g-mol·K
P=10; % pressure, atm
T=225.46; % temperature, K
%
% P-V-T equation
%
f= (P+a/V^2)*(V-b)-R*T;
```

```
>> V=fzero('PVT', 1) % the initial guess value is set to 1
```

V =

1.3204

```
>> V=fzero ('PVT', 0.1) % use 0.1 as the initial guess value
```

V =

0.1099

```
>> P=10;
```

```
>> T=225.46;
```

```
>> R=0.08206;
```

```
>> V0=R*T/P; % determine a proper initial guess value by the  
ideal gas law
```

```
>> V=fzero('PVT', V0) % solving with V0 as the initial guess  
value
```

V=

1.3204

```
>> options= optimset('disp', 'iter'); % display the iteration process
>> V=fzero('PVT', V0, options)
```

Search for an interval around 1.8501 containing a sign change:

Func-count	a	f(a)	b	f(b)	Procedure
1	1.85012	3.62455	1.85012	3.62455	initial interval
3	1.7978	3.22494	1.90245	4.03063	search
5	1.77612	3.06143	1.92413	4.20061	search
7	1.74547	2.83234	1.95478	4.44269	search
9	1.70211	2.51286	1.99813	4.78826	search
11	1.64081	2.07075	2.05944	5.28301	search
13	1.5541	1.46714	2.14614	5.99373	search
15	1.43149	0.66438	2.26876	7.01842	search
16	1.25808	-0.340669	2.26876	7.01842	search

Search for a zero in the interval [1.2581, 2.2688]:

Func-count	X	f(x)	Procedure
16	1.25808	-0.340669	initial
17	1.30487	-0.0871675	interpolation
18	1.32076	0.00213117	interpolation
19	1.32038	-1.80819e-005	interpolation
20	1.32039	-3.64813e-009	interpolation
21	1.32039	0	interpolation

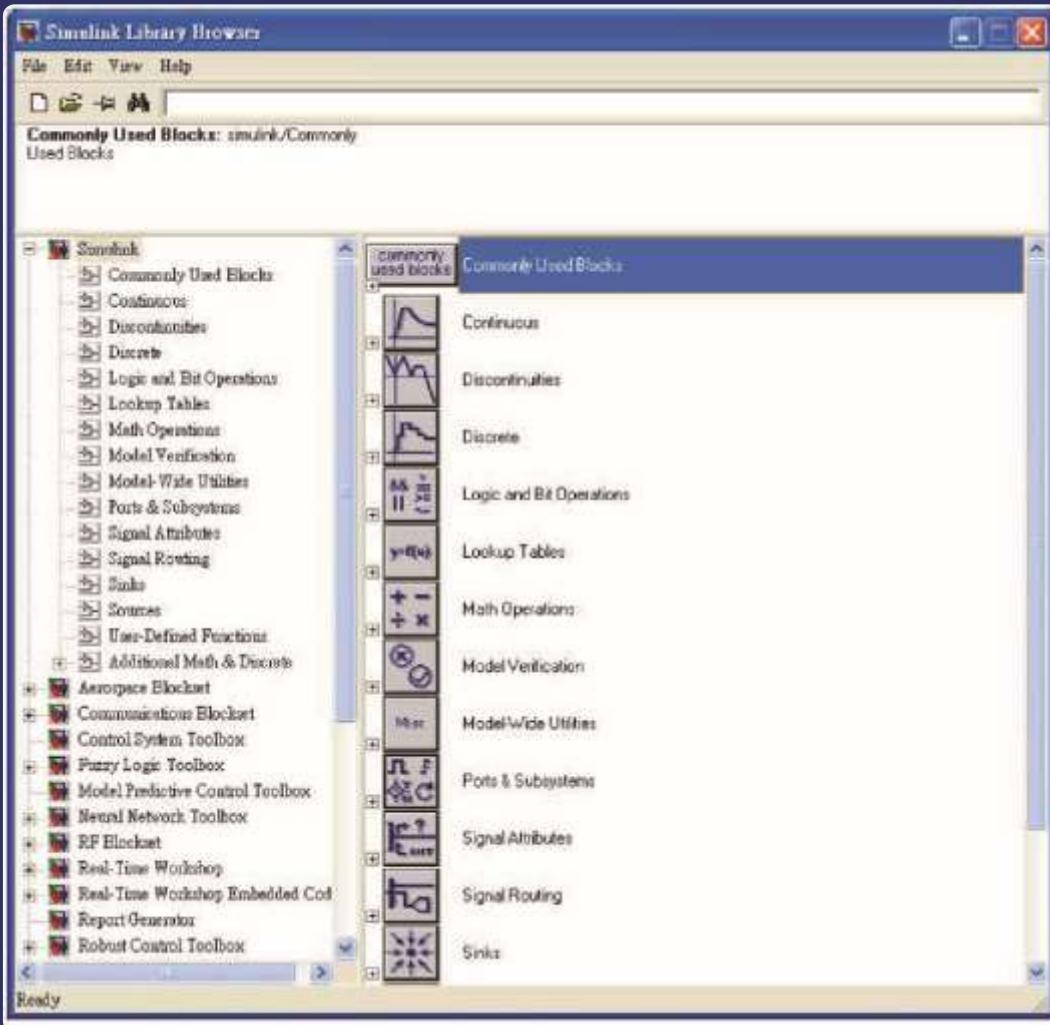
Zero found in the interval [1.25808, 2.26876]

V=

1.3204

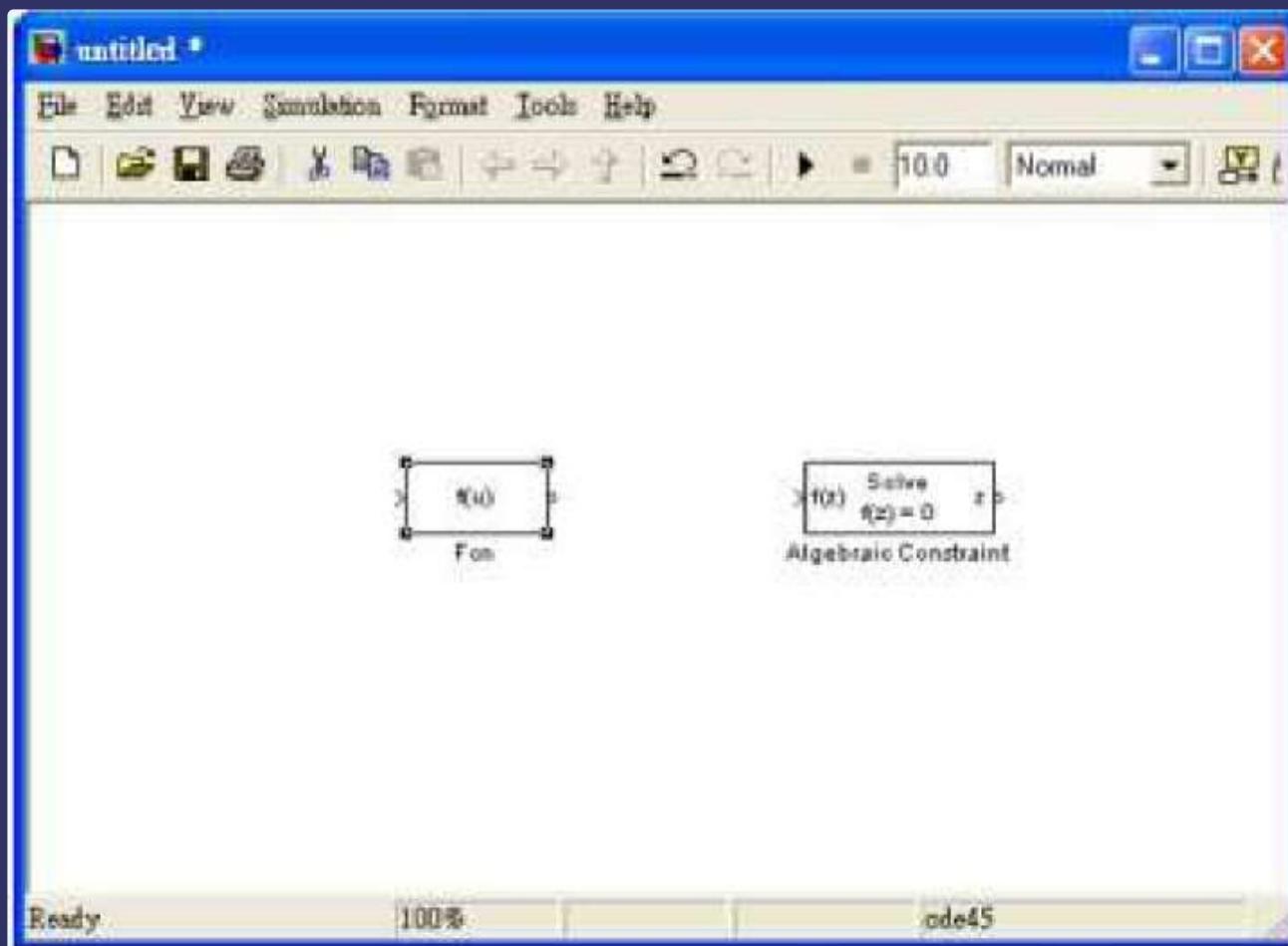
Solution by Simulink:

Step 1: start Simulink



Solution by Simulink:

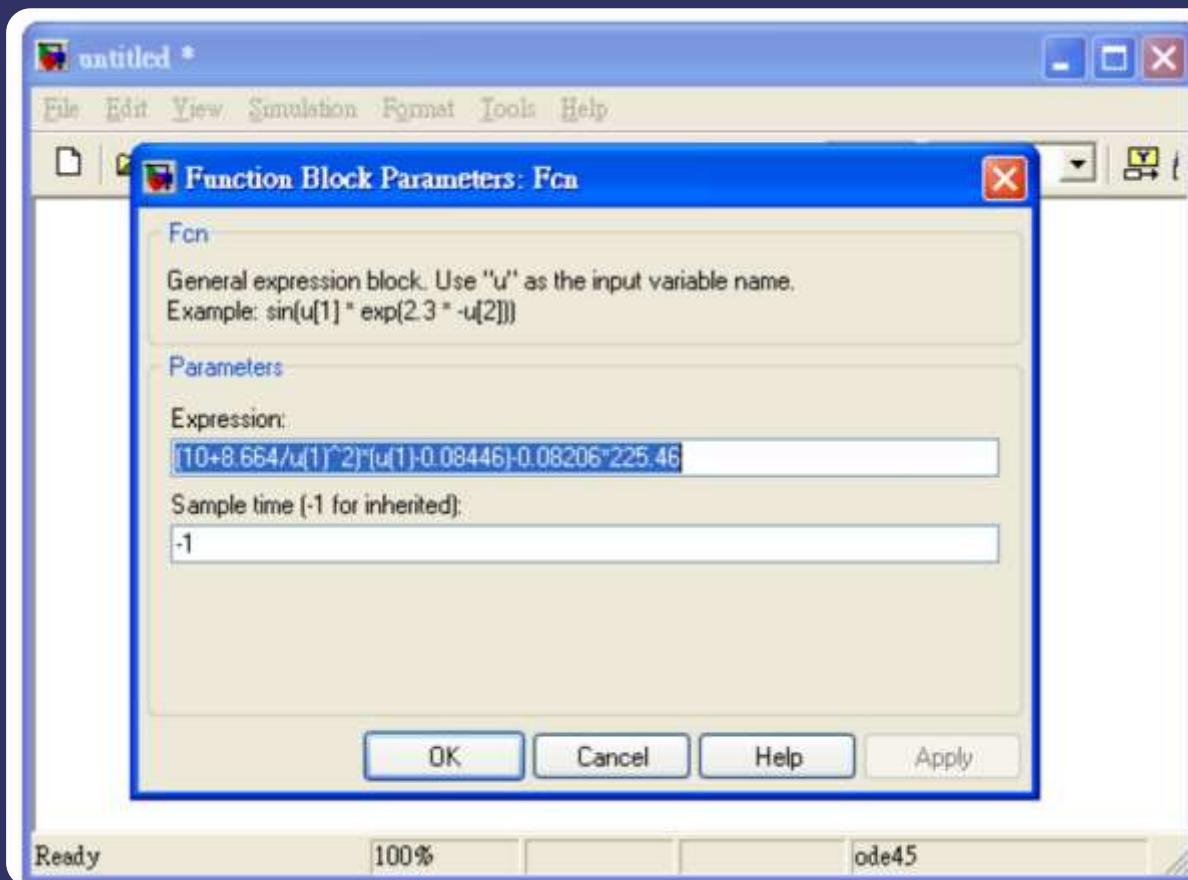
Step 2:



Solution by Simulink:

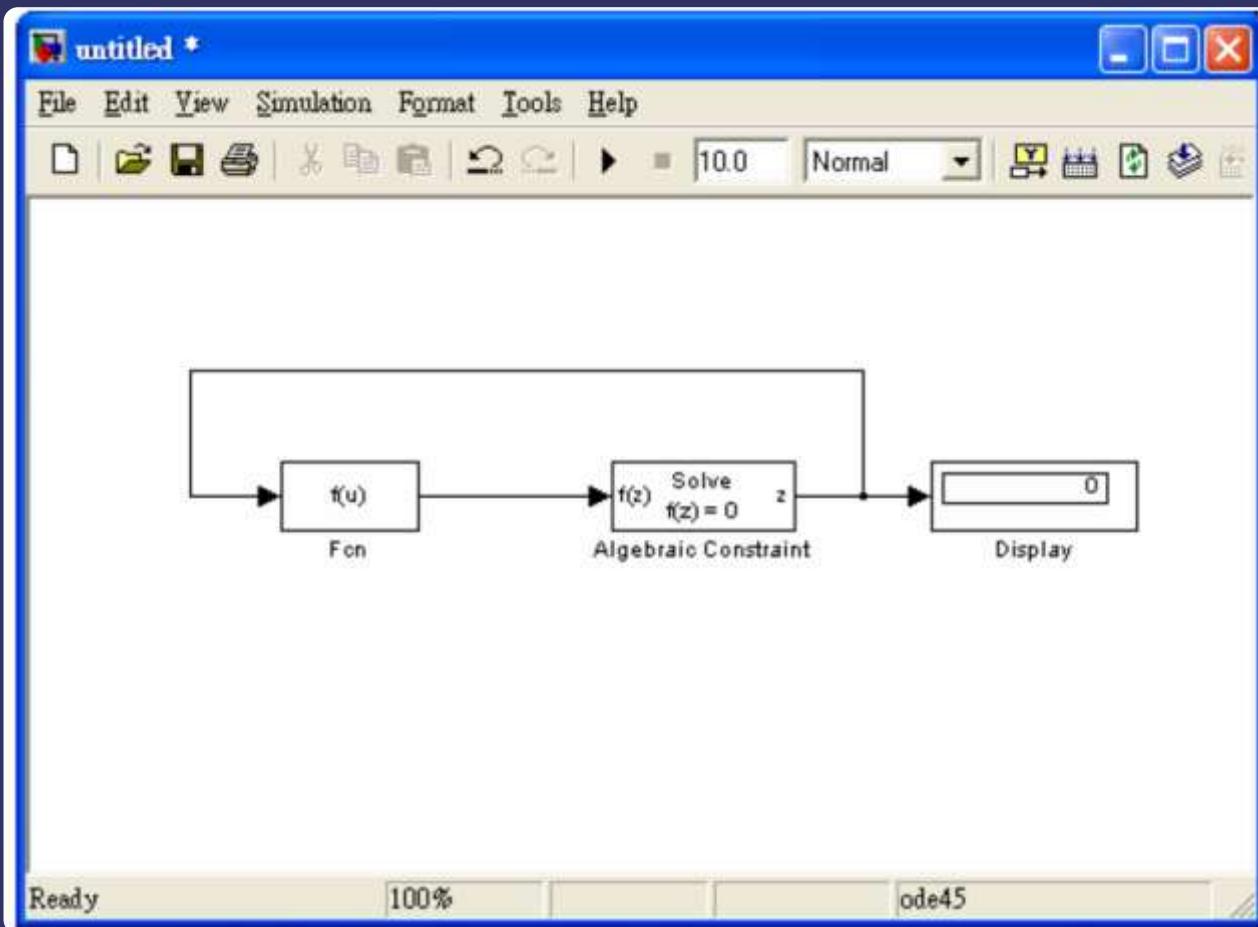
Step 3: Edit Fcn

$$(10+8.664/u(1)^2)*(u(1)-0.08446)-0.08206*225.46$$



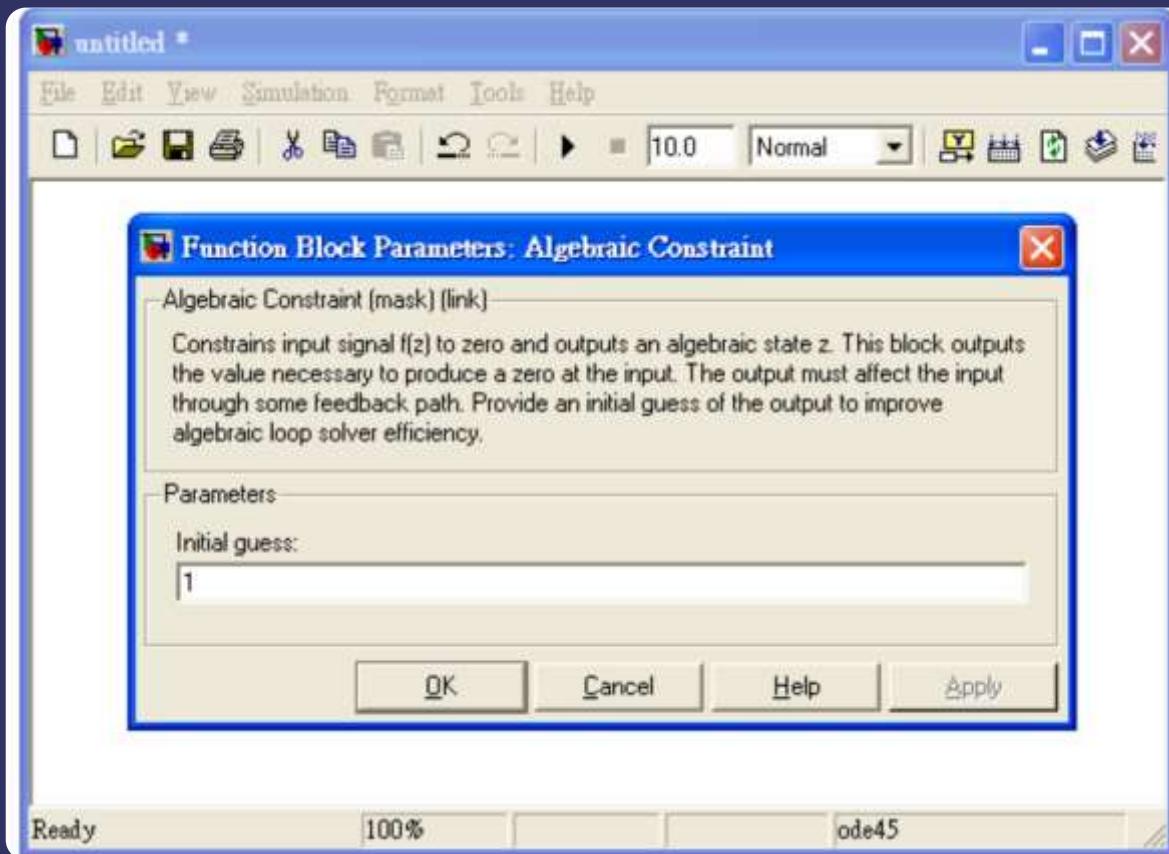
Solution by Simulink:

Step 4: solution display



Solution by Simulink:

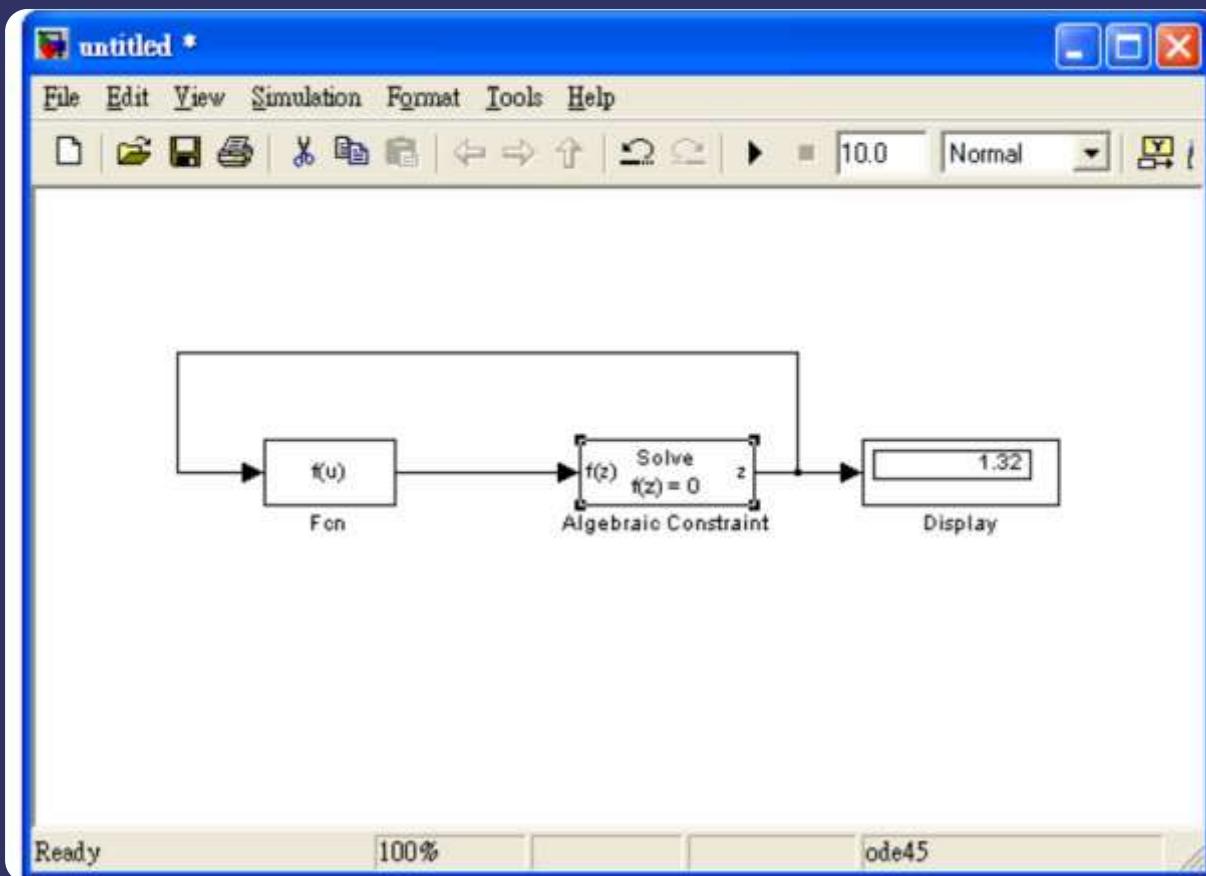
Step 5: Key in the initial guess value



Solution by Simulink:

Step 6: Click 

building a dialogue box

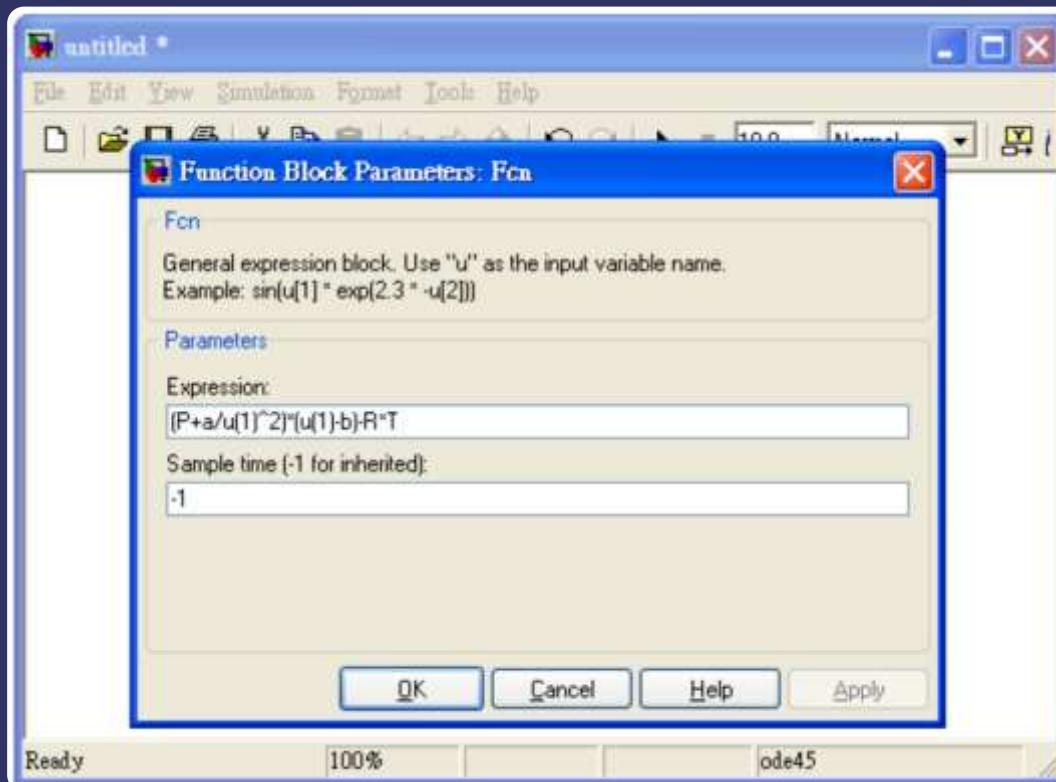


Solution by Simulink:

Step 1-2:

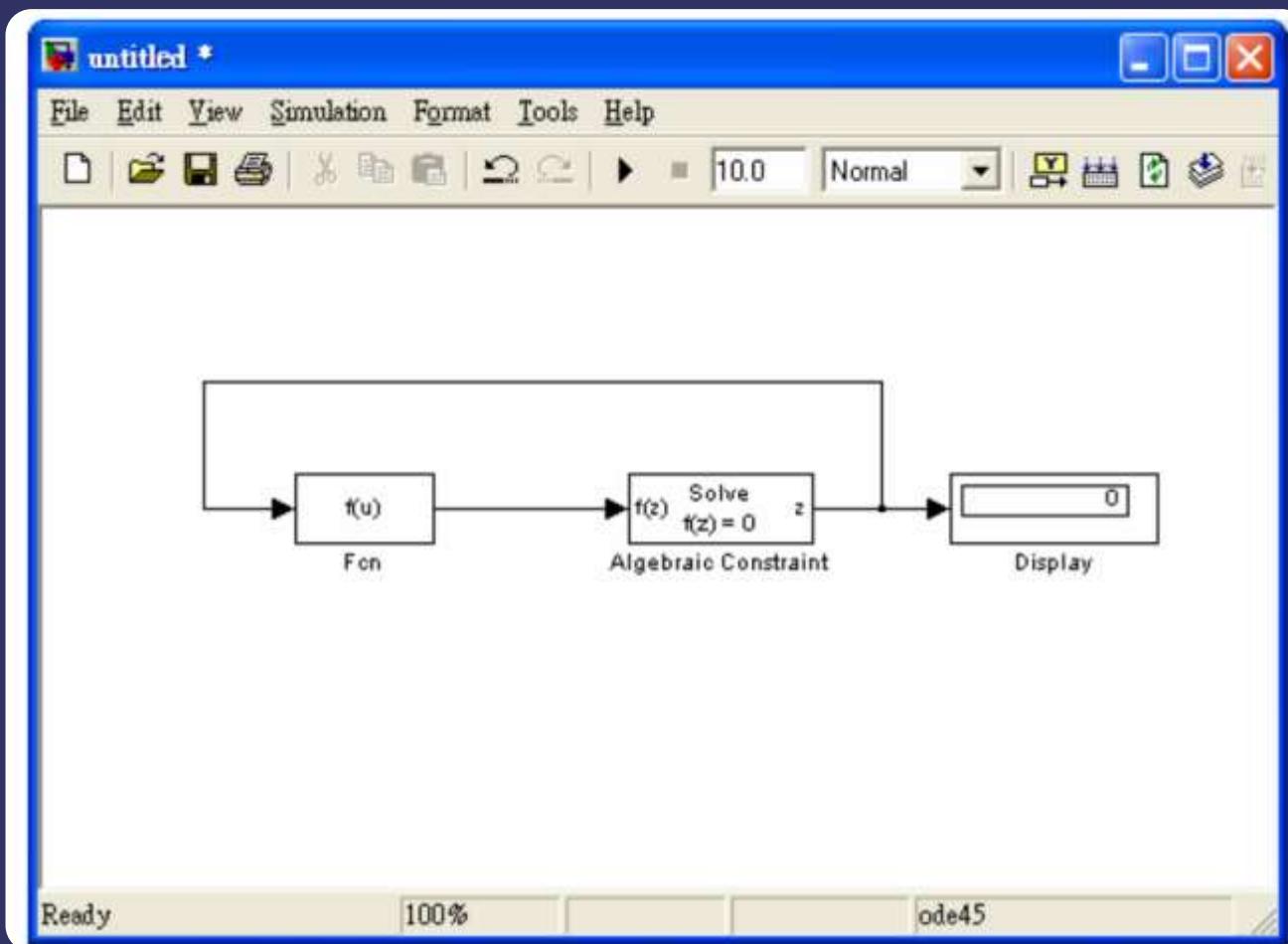
Step 3: Edit Fcn using variable names, P , R , T , a , and b

$$(P + a/u(1)^2)*(u(1)-b) - R*T$$



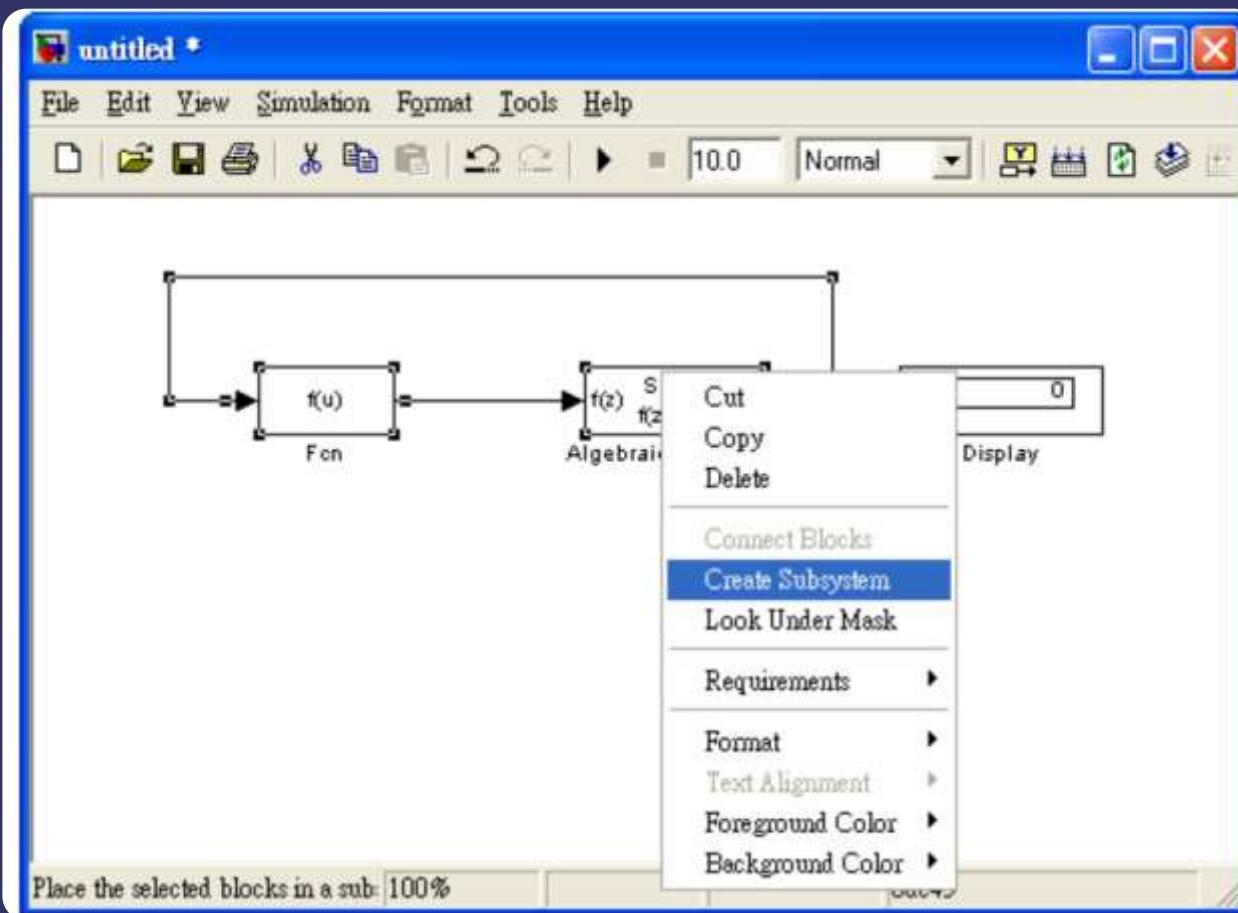
Solution by Simulink:

Step 4-5:



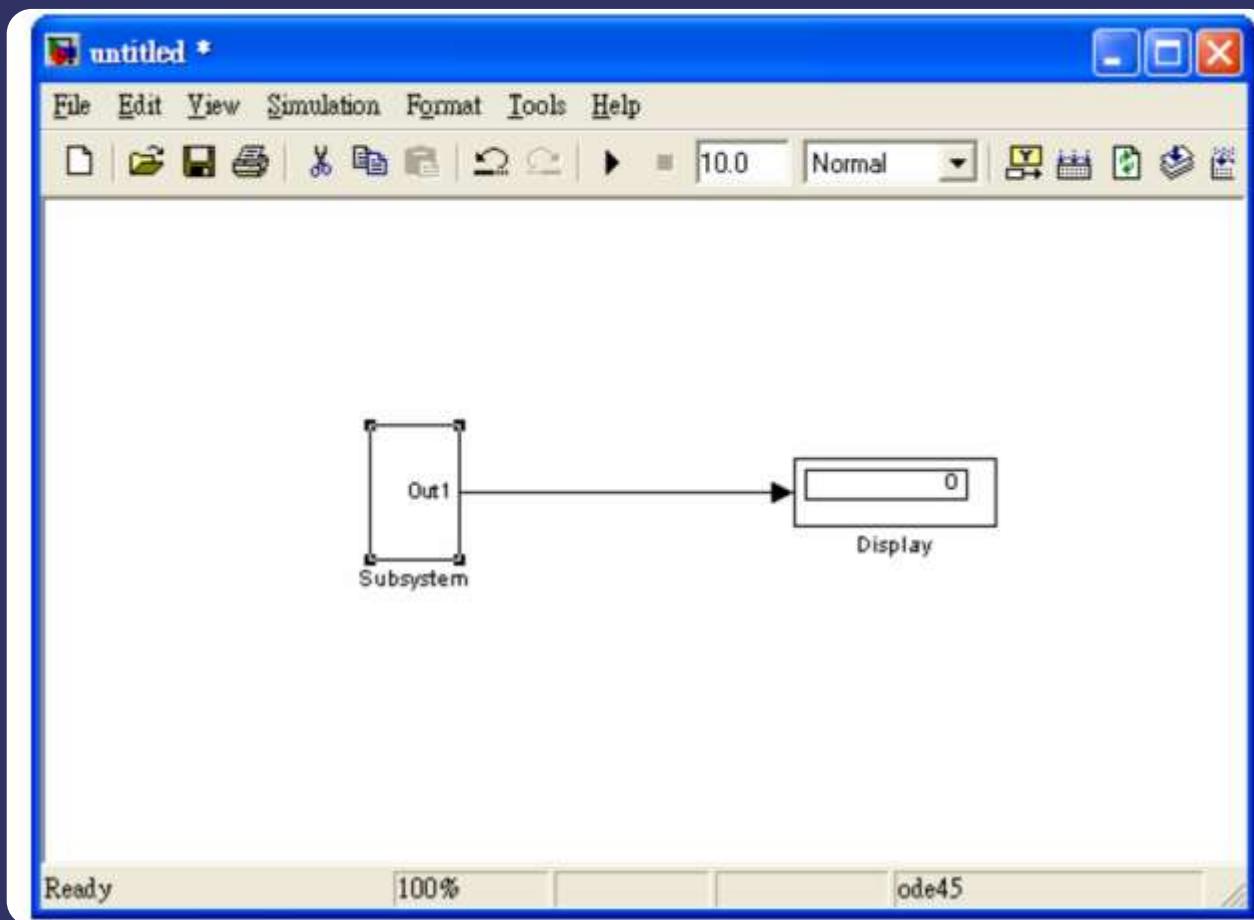
Solution by Simulink:

Step 6:



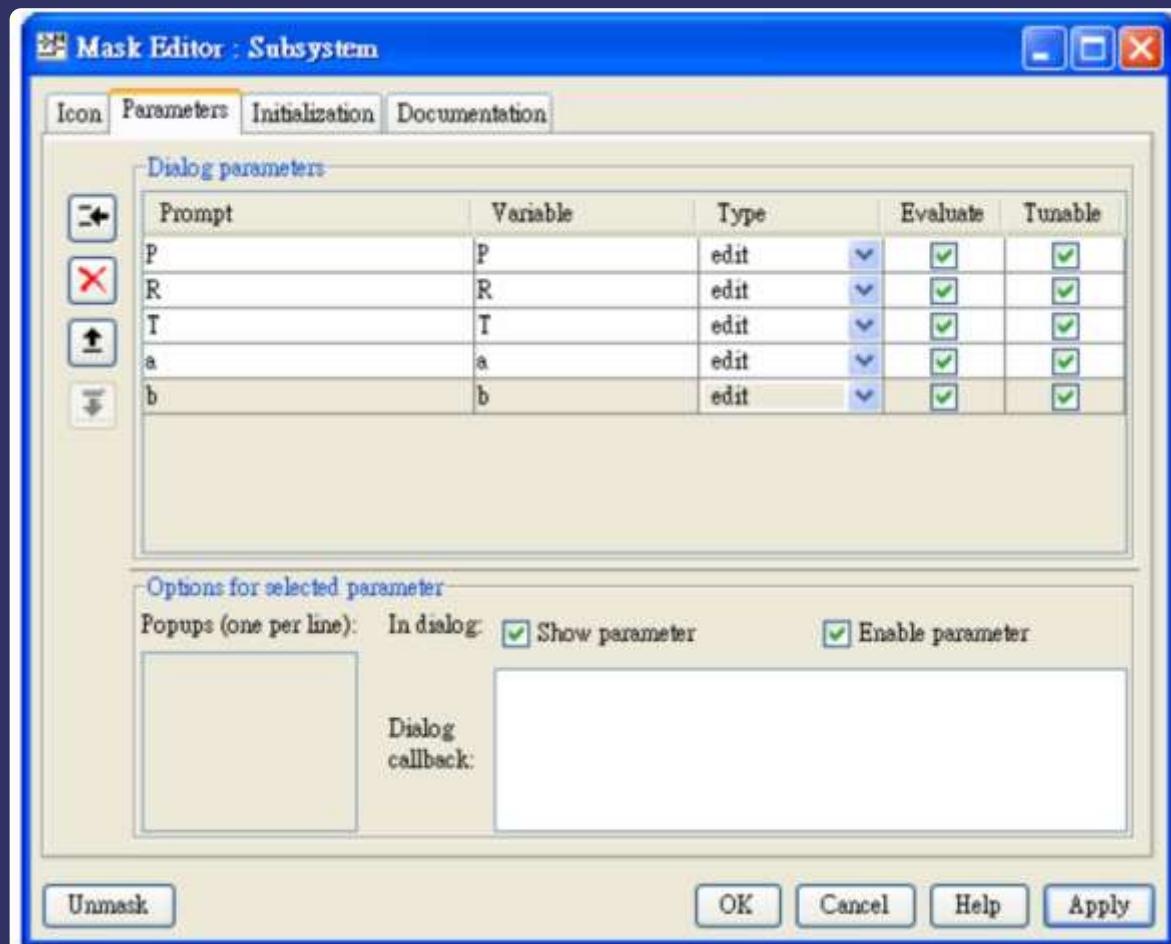
Solution by Simulink:

Step 6:



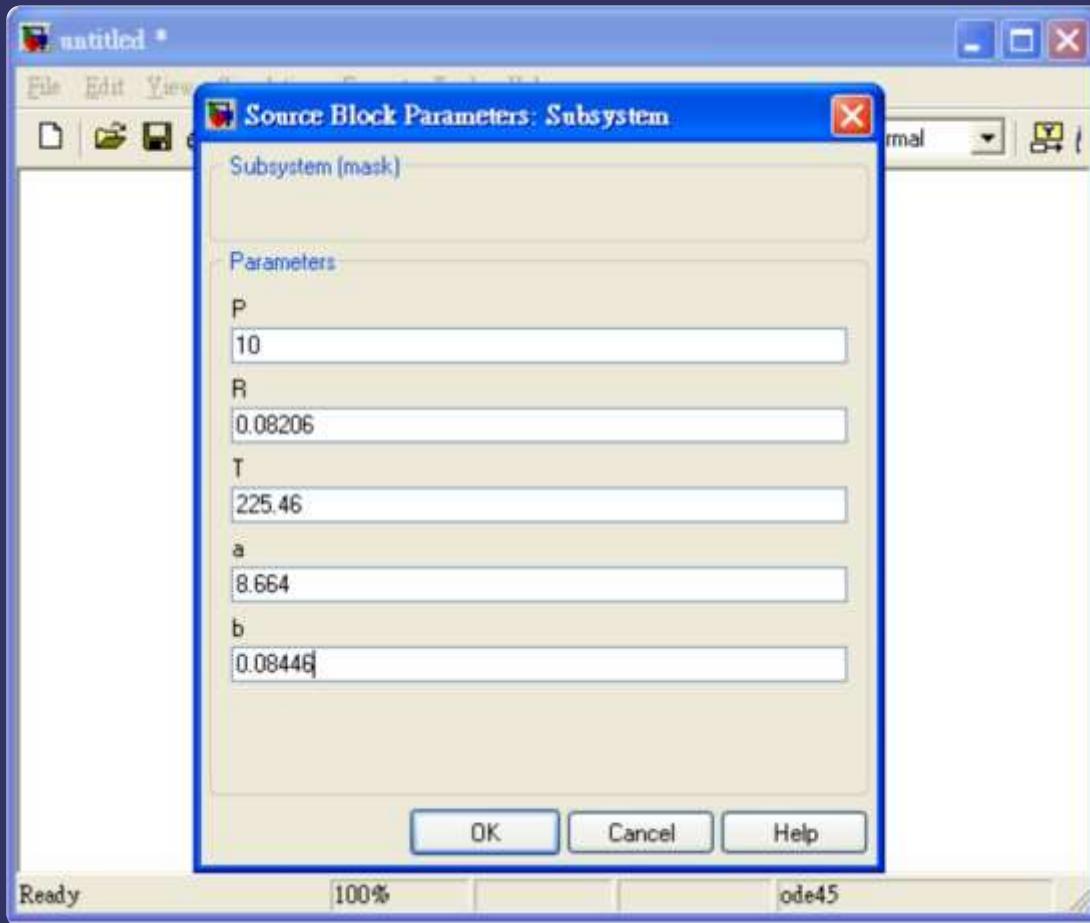
Solution by Simulink:

Step 7: Mask Subsystem



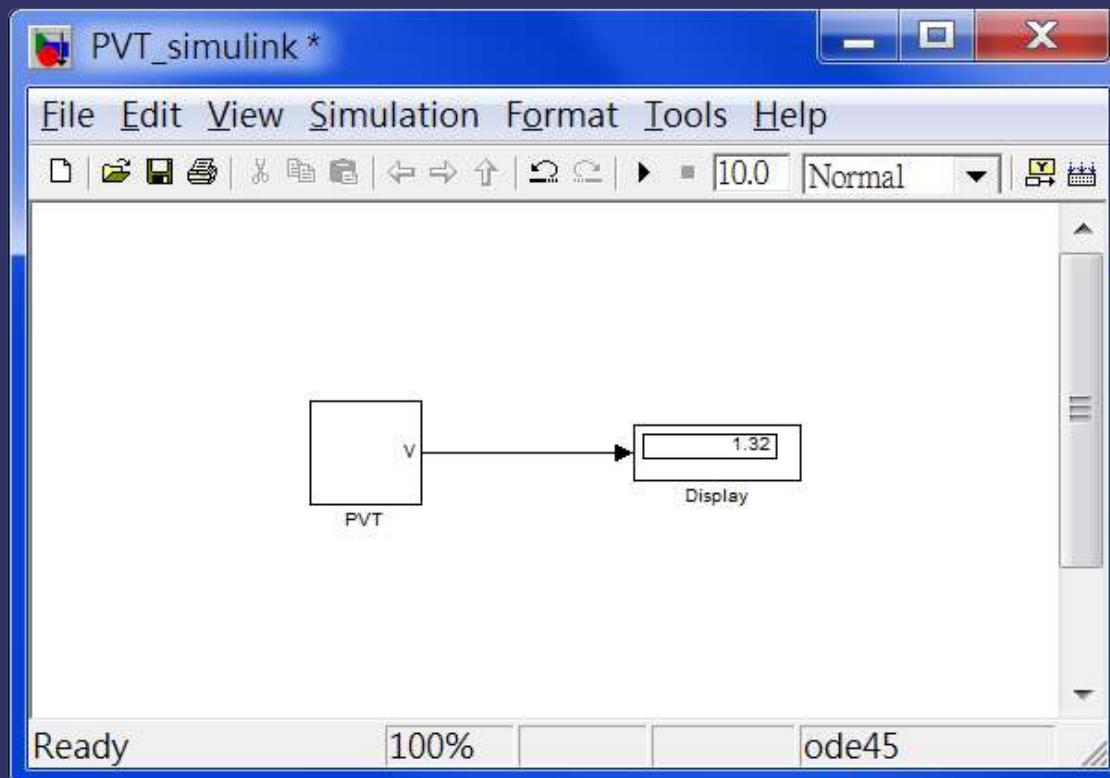
Solution by Simulink:

Step 8: key in the parameter values



Solution by Simulink:

Step 9: Execute the program



2.1.2 Solution of a system of nonlinear equations

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

$$\mathbf{F}(\mathbf{x}) = 0$$

- CSTR

$$\dot{x}_1 = -x_1 + D_a (1 - x_1) \exp\left(\frac{x_2}{1 + x_2 / \phi}\right)$$

$$\dot{x}_2 = -(1 + \beta)x_2 + BD_a (1 - x_1) \exp\left(\frac{x_2}{1 + x_2 / \phi}\right)$$

- The kinetic parameters are as follows: $B = 8$, $D_a = 0.072$, $\phi = 20$, and $\beta = 0.3$.

$$f_1(x_1, x_2) = -x_1 + D_a (1 - x_1) \exp\left(\frac{x_2}{1 + x_2 / \phi}\right) = 0$$

$$f_2(x_1, x_2) = -(1 + \beta)x_2 + BD_a (1 - x_1) \exp\left(\frac{x_2}{1 + x_2 / \phi}\right) = 0$$

`x=fsolve('filename', x0)`

Step 1: Provide the function

cstr.m

```
function f=fun(x)
%
% steady-state equation system (2.1-6) of the CSTR
%
% kinetic parameters
%
B=8;
Da=0.072;
phi=20;
beta=0.3;
%
% nonlinear equations in vector form
%
f= [-x(1)+Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))
    -(1+beta)*x(2)+B*Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))];
```

Step 2:

```
>> x=fsolve('cstr', [0.1 1]) % solve the equations with the initial guess [0.1 1]
```

x=

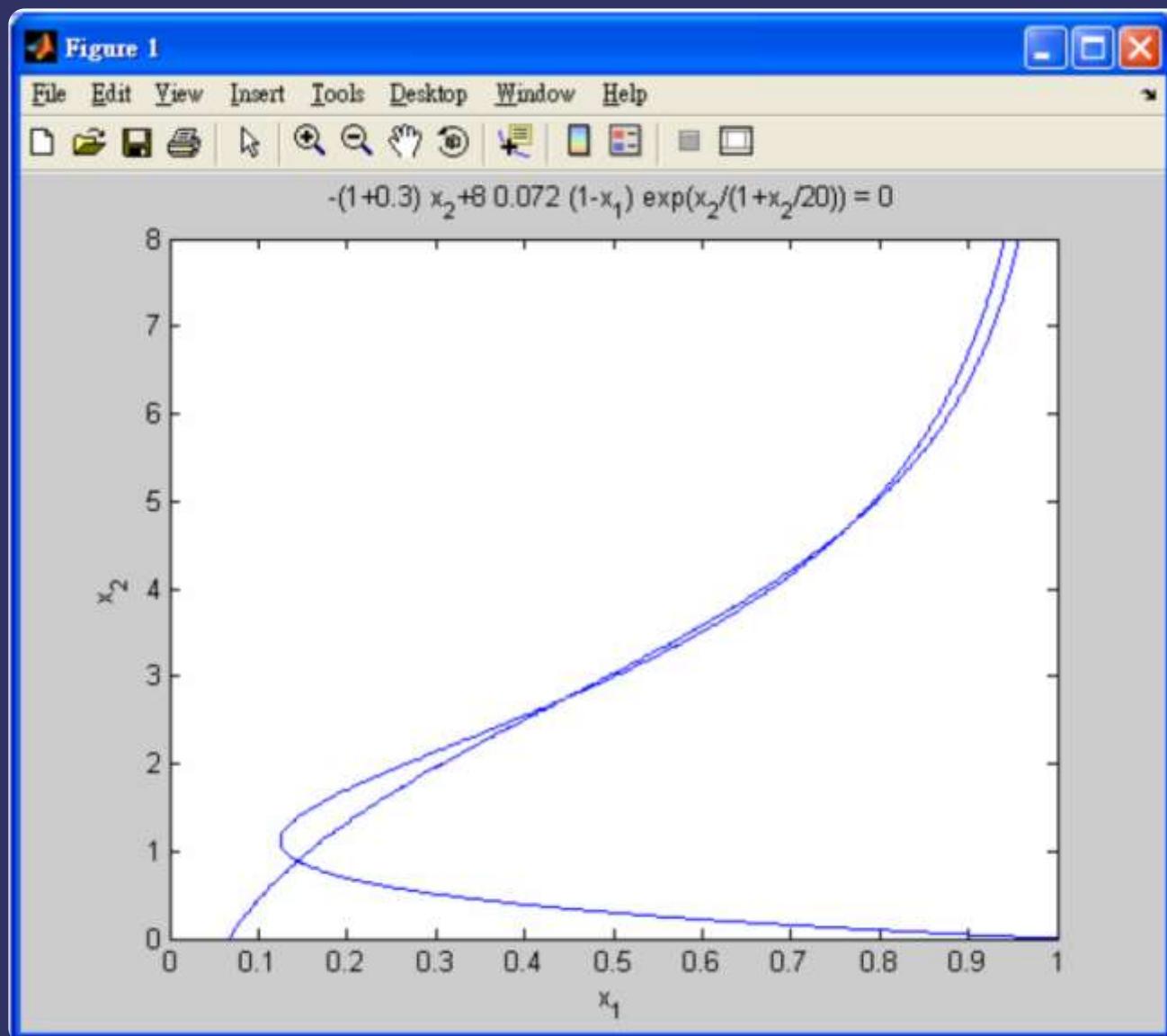
0.1440 0.8860

```
>> ezplot('-x1+0.072*(1-x1)*exp(x2/(1+x2/20))', [0, 1, 0, 8]) % plotting (2.1-6a)
```

```
>> hold on
```

```
>> ezplot('-(1+0.3)*x2+8*0.072*(1-x1)*exp(x2/(1+x2/20))', [0, 1, 0, 8]) % plotting  
(2.1-6b)
```

```
>> hold off
```



- (0.15, 1), (0.45, 3), and (0.75, 5)

```
>> x_1=fsolve('cstr', [0.15 1]) % the first solution
```

x_1=

0.1440 0.8860

```
>> x_2=fsolve('cstr', [0.45 3]) % the second solution
```

x_2=

0.4472 2.7517

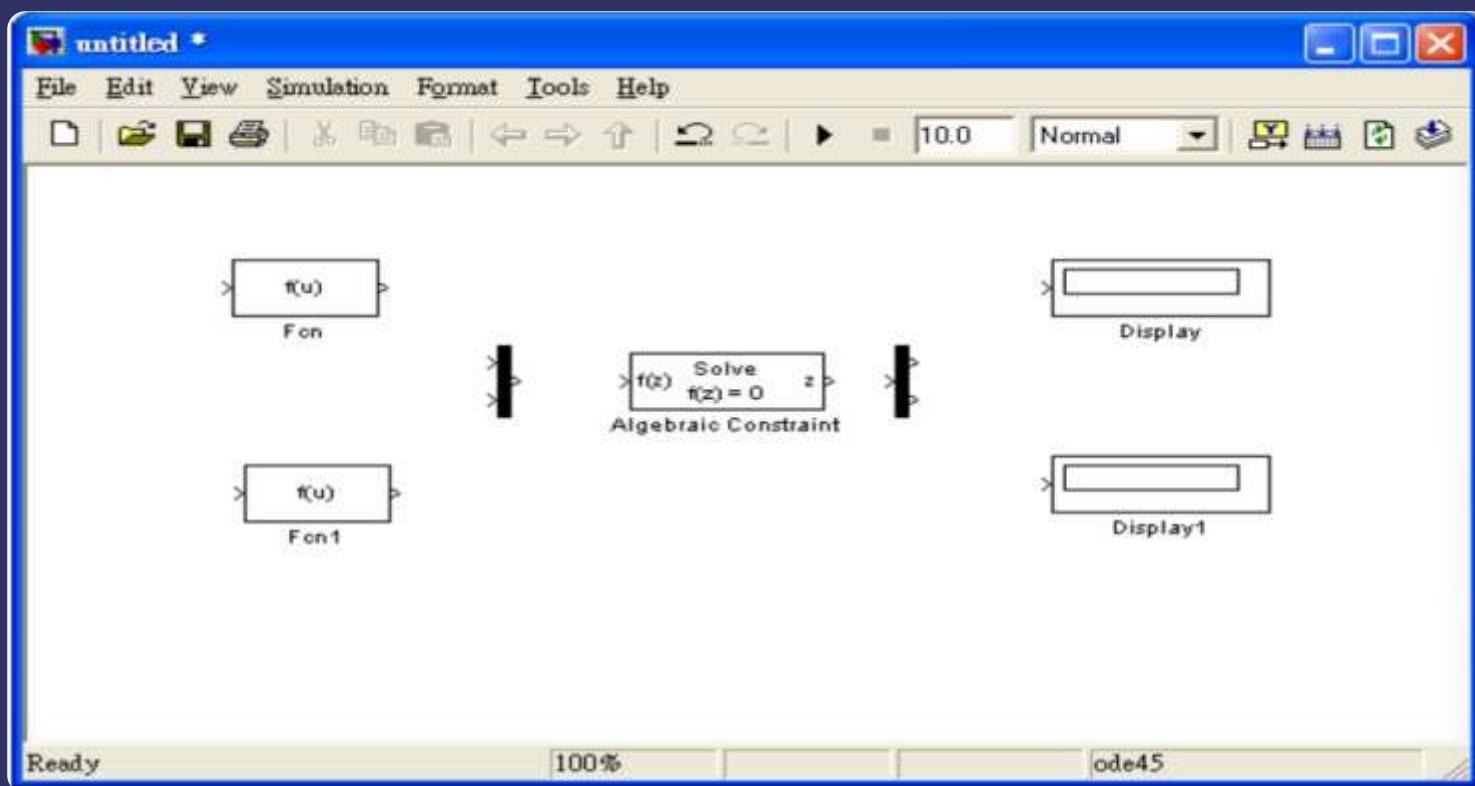
```
>> x_3=fsolve('cstr', [0.75 5]) % the third solution
```

x_3=

0.7646 4.7050

Solution by Simulink:

Step 1:

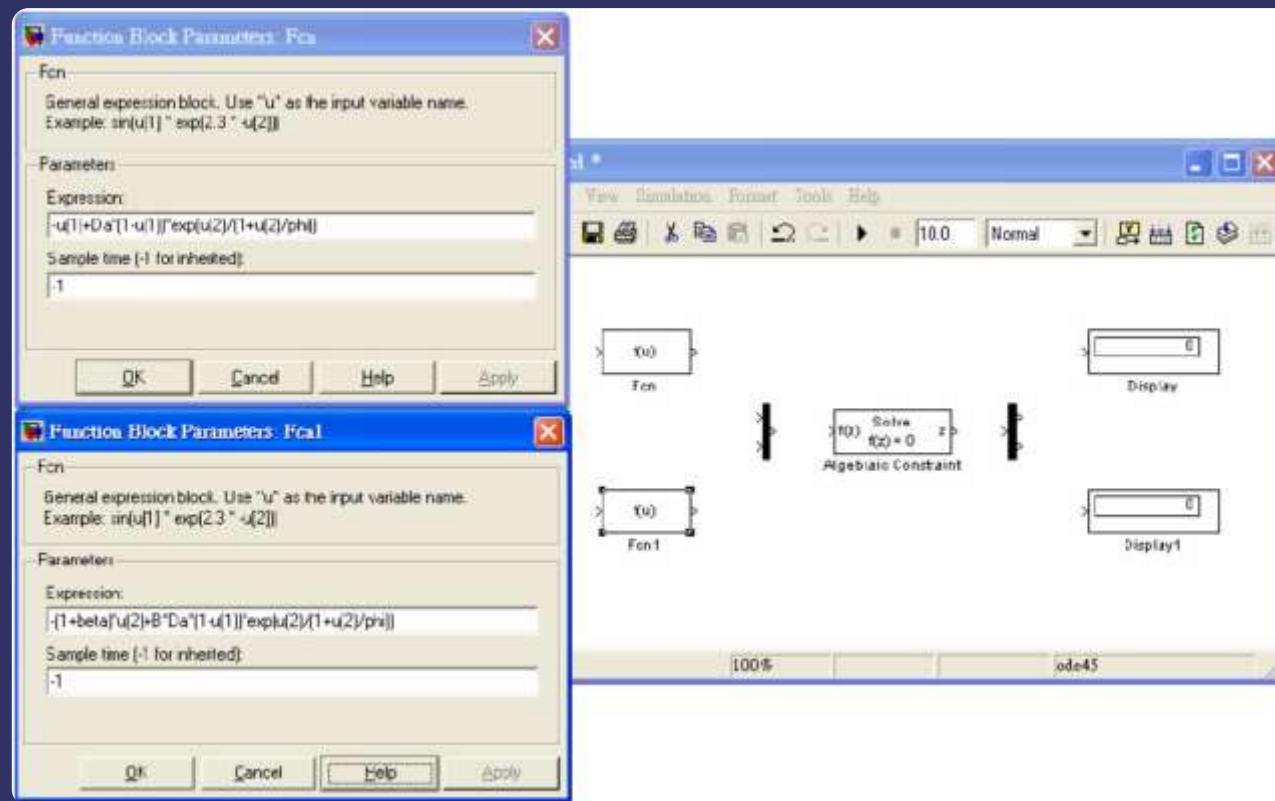


Solution by Simulink:

Step 2:

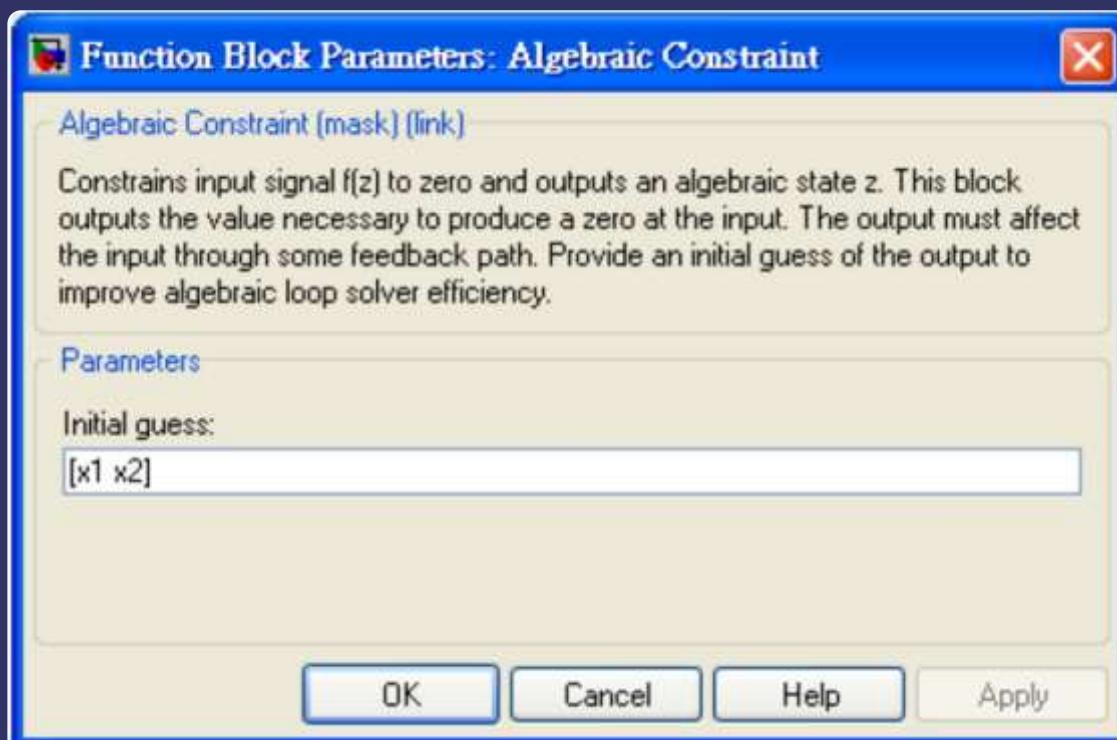
In Fcn: $-u(1) + Da * (1 - u(1)) * \exp(u(2)) / (1 + u(2) / \phi)$

In Fcn 1: $-(1 + \beta) * u(2) + B * Da * (1 - u(1)) * \exp(u(2)) / (1 + u(2) / \phi)$



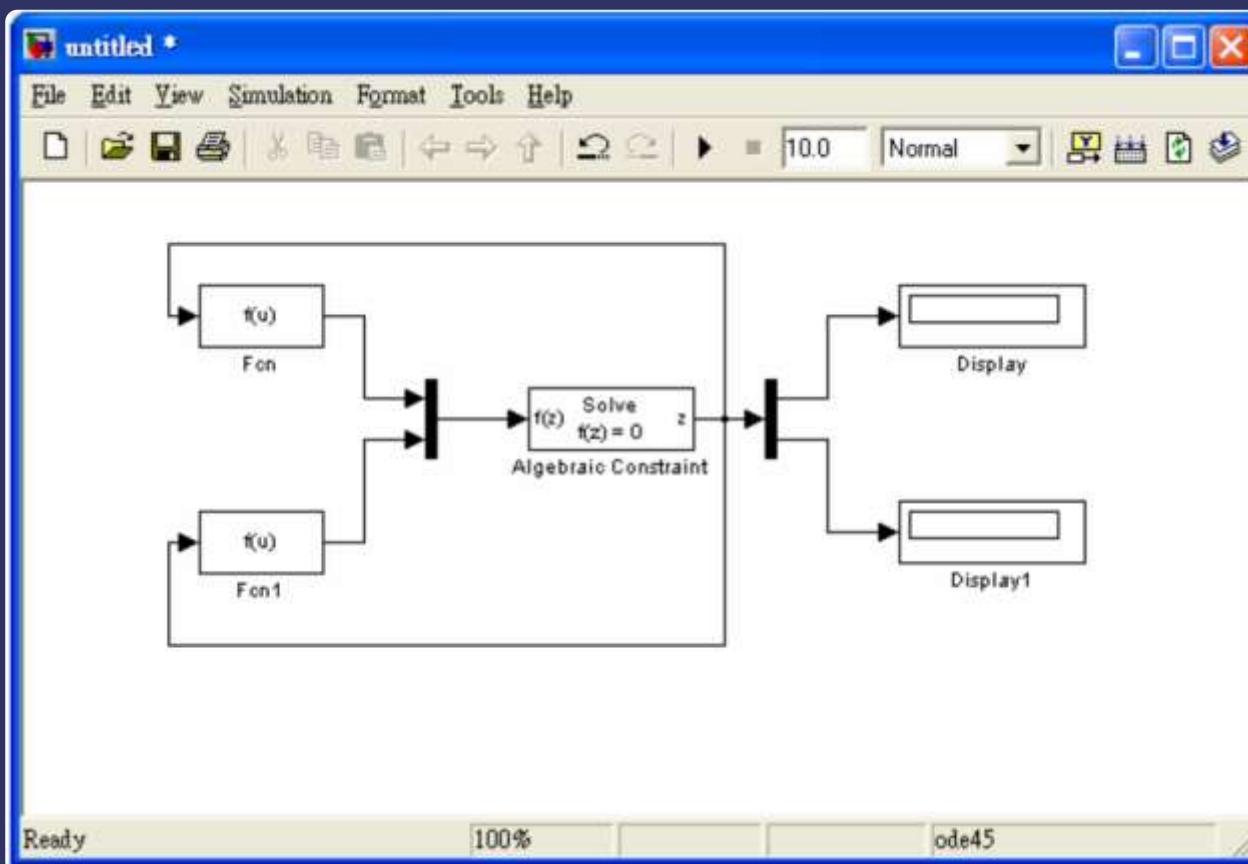
Solution by Simulink:

Step 2:



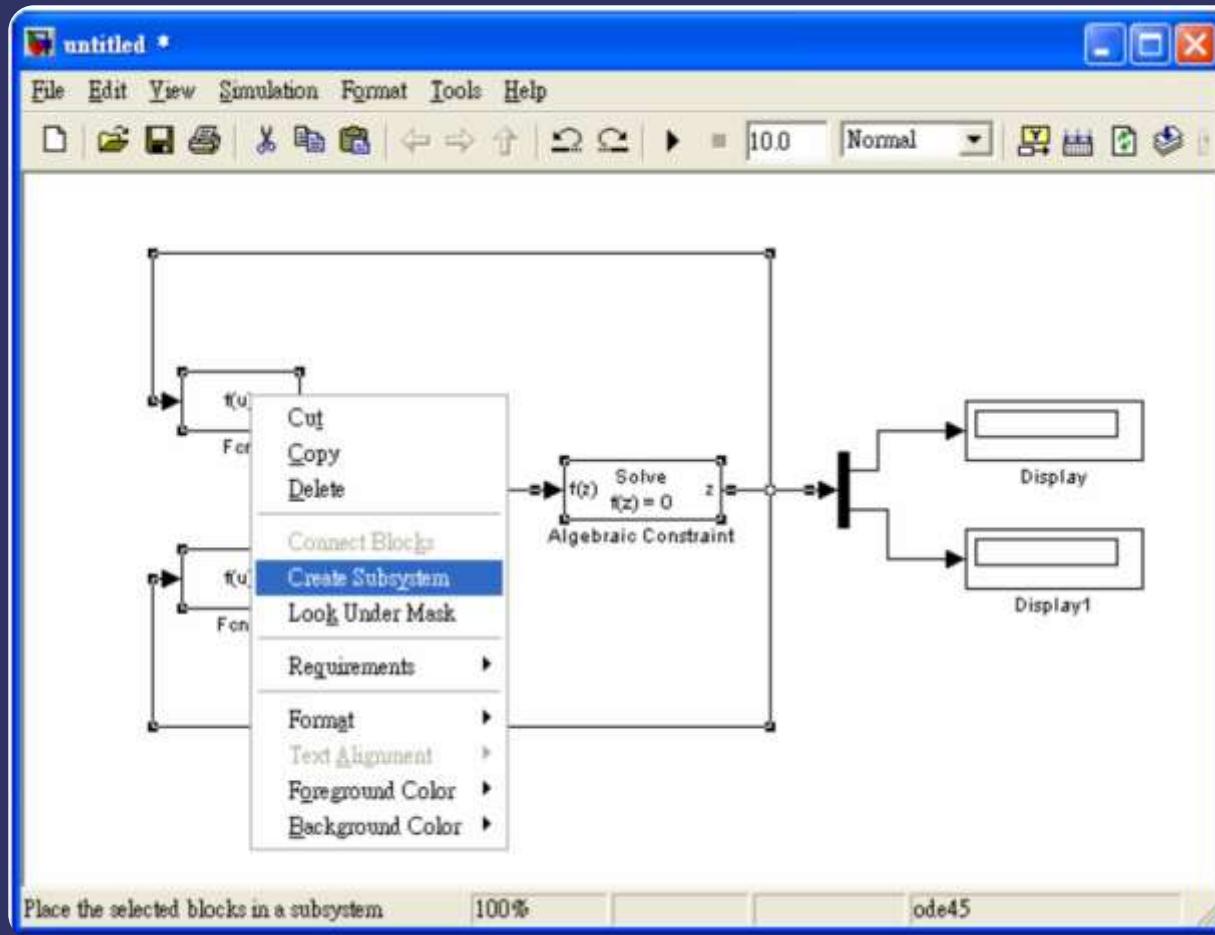
Solution by Simulink:

Step 3:



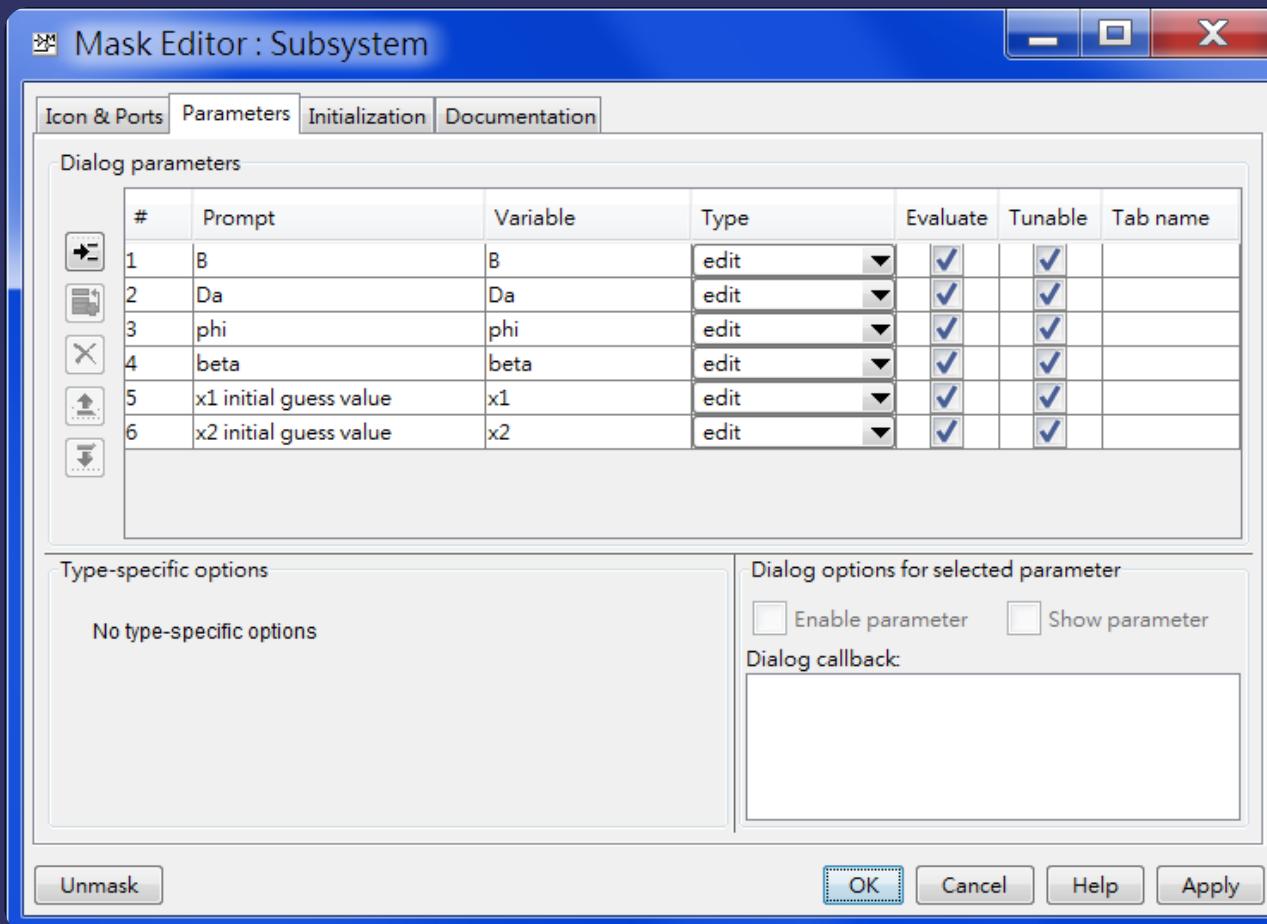
Solution by Simulink:

Step 4: Create a Subsystem for



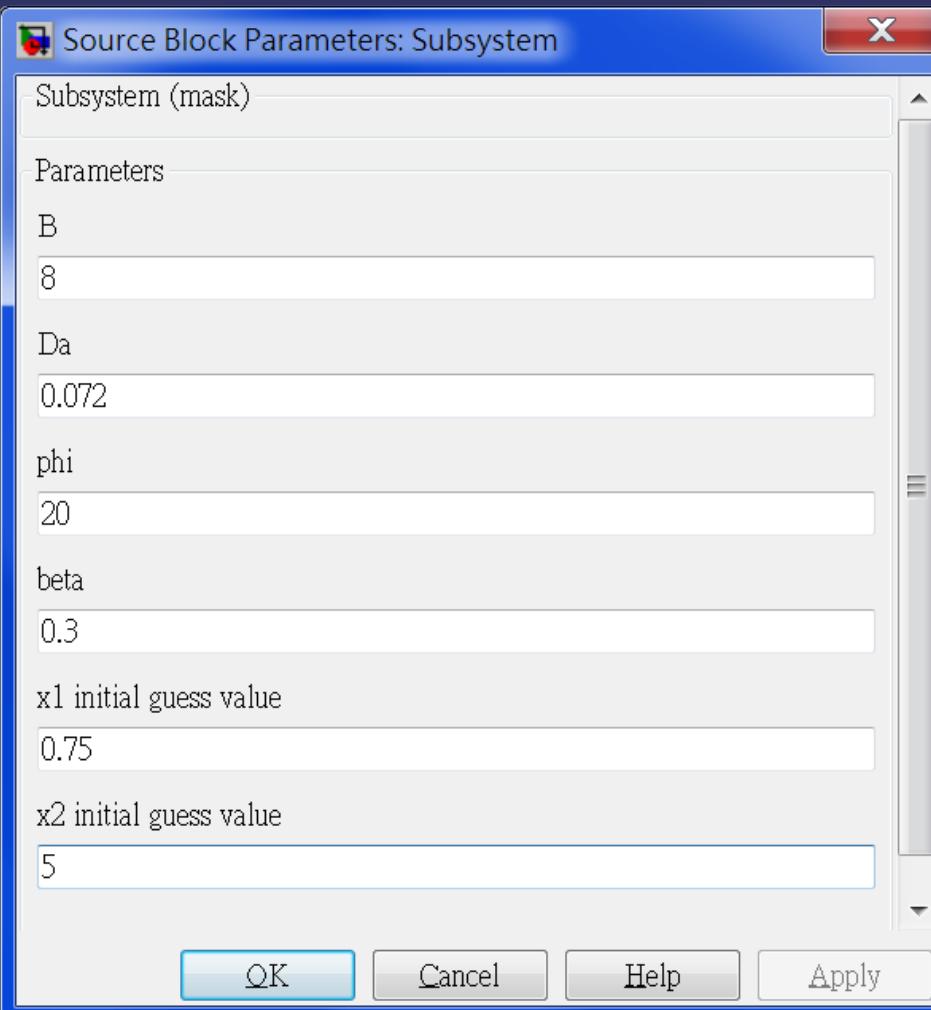
Solution by Simulink:

Step 5: Create a dialogue box



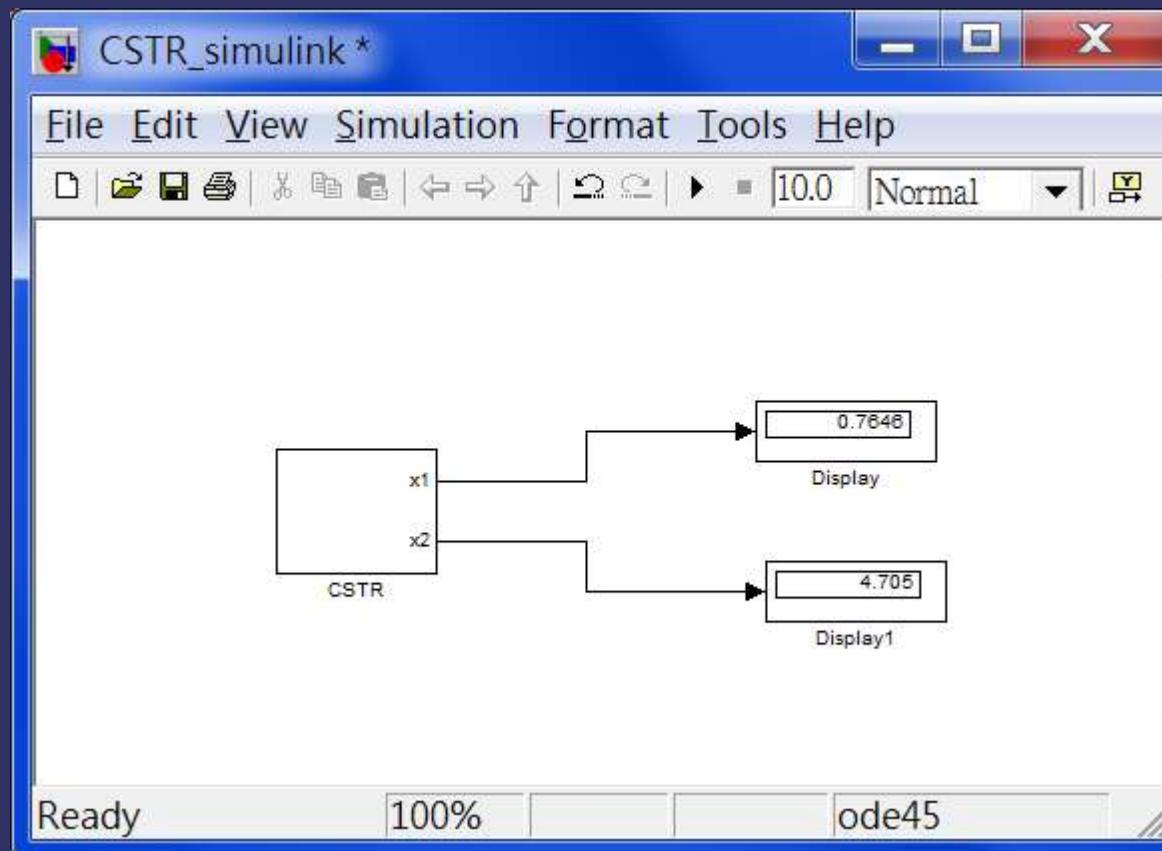
Solution by Simulink:

Step 6: Input system parameters and initial guess values



Solution by Simulink:

Step 7:



2.2 Chemical engineering examples

Example 2-2-1

Boiling point of an ideal solution

Consider a three-component ideal mixture of which the vapor pressure of each component can be expressed by the following Antoine equation:

$$\log P_i^0 = A_i + \frac{B_i}{C_i + T}$$

where T represents the temperature of the solution ($^{\circ}\text{C}$) and P_i^0 denotes the saturated vapor pressure (mmHg). The Antoine coefficients of each component are shown in the following table (Leu, 1985):

Example 2-2-1

Component	A_i	B_i	C_i
1	6.70565	-1,211.033	220.790
2	6.95464	-1,344.800	219.482
3	7.89750	-1,474.080	229.130

Determine the boiling point at one atmospheric pressure for each of the following composition conditions:

Condition	x_1	x_2	x_3
1	0.26	0.38	0.36
2	0.36	0.19	0.45

Problem formulation and analysis:

Raoult's law:

$$P_i = x_i P_i^0$$

$$P = \sum_{i=1}^3 P_i = x_1 P_1^0 + x_2 P_2^0 + x_3 P_3^0$$

$$\begin{aligned} f(T) &= \sum_{i=1}^3 x_i P_i^0 - P = 0 \\ &= \sum_{i=1}^3 x_i \cdot 10^{A_i + B_i / (C_i + T)} - 760 = 0 \end{aligned}$$

MATLAB program design:

— ex2_2_1.m —

```
%  
% Example 2-2-1 boiling point of an ideal solution  
%  
% main program: ex2_2_1.m  
% subroutine (function file): ex2_2_1f.m  
%  
clear  
clc  
%  
global A B C X_M  
%  
% given data  
%  
A= [6.70565 6.95464 7.89750]; % coefficient A  
B= [-1211.033 -1344.800 -1474.080]; % coefficient B  
C= [220.790 219.482 229.130]; % coefficient C  
%
```

MATLAB program design:

— ex2_2_1.m —

```
X=[0.25 0.40 0.35  
     0.35 0.20 0.45]; % component concentrations for each case  
%  
[m, n]=size(X); % n: the number of components  
% m: the number of conditions  
%  
for i=1: m  
X_M=X(i, :);  
T(i) =fzero('ex2_2_1f', 0);  
end  
%  
% results printing  
%  
for i=1: m  
fprintf('condition %d, X(1)=% .3f, X(2)=% .3f, X(3)=% .3f, boiling point=% .3f °C\n', i, X(i,1), X(i,2), X(i,3), T(i))  
end
```

MATLAB program design:

— ex2_2_1f.m —

```
%  
% example 2-2-1 boiling point of an ideal solution  
% subroutine (function file): ex2_2_1f.m  
%  
function f= ex2_2_1f(T)  
global A B C X_M  
Pi=10.^ (A+B./(C+T)); % saturated vapor pressure of each component  
p=sum(X_M.*Pi); % total pressure  
f=p-760;  
end
```

Execution results:

```
>> ex2_2_1
```

condition 1, $X(1)=0.250$, $X(2)=0.400$, $X(3)=0.350$, boiling point=82.103°C

condition 2, $X(1)=0.350$, $X(2)=0.200$, $X(3)=0.450$, boiling point=77.425°C

Example 2-2-2

Equilibrium concentrations in a reaction system



where K_i is the equilibrium constant under the operating condition of 500°C and 1 atmospheric pressure. Assume that 1 mole of CO and 3 moles of H₂ exist initially in the system, determine the equilibrium concentration of each component (mole fraction) when the reaction system reaches an equilibrium.

Analysis of the question:

Let y_1, y_2, \dots, y_6 stand for the mole fractions of H₂, CO, CH₄, H₂O, CO₂, and C₂H₆, respectively. Besides, let x_1, x_2, x_3 , and x_4 be the reaction rates of individual equations in (2.2-5a-d).

$$\frac{y_3 y_4}{y_1^3 y_2} = 69.18$$

$$\frac{y_5 y_1}{y_2 y_4} = 4.68$$

$$\frac{y_2^2}{y_5} = 5.6 \times 10^{-3}$$

$$\frac{y_6 y_4^2}{y_1^5 y_2^2} = 0.14$$

Analysis of the question:

$$y_1 = (3 - 3x_1 + x_2 - 5x_4) / D$$

$$y_2 = (1 - x_1 - x_2 + 2x_3 - 2x_4) / D$$

$$y_3 = x_1 / D$$

$$y_4 = (x_1 - x_2 + 2x_4) / D$$

$$y_5 = (x_2 - x_3) / D$$

$$y_6 = x_4 / D$$

$$D = 4 - 2x_1 + x_3 - 4x_4$$

MATLAB program design:

— ex2_2_2.m —

```
function ex2_2_2 % notice the format of the first line
%
% Example 2-2-2 Equilibrium concentrations in a reaction system
%
clear
clc
%
% initial guess values
%
x0=[0.7 0 -0.1 0];
%
% solving by calling the embedded function file ex2_2_2f
%
options= optimset('disp', 'iter');
x=fsolve(@ex2_2_2f, x0, options); % notice the use of the function handle @
%
% calculate the mole fraction
%
```

MATLAB program design:

— ex2_2_2.m —

```
y=ex2_2_2y(x);
%
% results printing
%
disp(' ')
disp('reaction rate of each equation')
%
for i=1:4
fprintf('x(%d) =%.3e \n', i, x (i))
end
%
disp(' ')
disp('mole fraction of each component')
%
for i=1:6
fprintf('y(%d) =%.3e \n', i, y (i))
end
```

MATLAB program design:

— ex2_2_2.m —

```
%  
% chemical equilibrium equations  
%  
function f=ex2_2_2f(x)  
%  
y= ex2_2_2y(x);  
%  
% the set of nonlinear equations  
%  
f= [y(3)*y(4)/(y(1)^3*y(2))-69.18  
    y(5)*y(1)/(y(2)*y(4))-4.68  
    y(2)^2/y(5)-5.60e-3  
    y(6)*y(4)^2/(y(1)^5*y(2)^2)-0.14];  
%  
% the subroutine for calculating the mole fraction  
%  
function y=ex2_2_2y(x)
```

MATLAB program design:

— ex2_2_2.m —

```
% chemical equilibrium equations  
%  
function f=ex2_2_2f(x)  
%  
y= ex2_2_2y(x);  
%  
% the set of nonlinear equations  
%  
f= [y(3)*y(4)/(y (1)^3*y(2))-69.18  
    y(5)*y(1)/(y(2)*y(4))-4.68  
    y(2)^2/y(5)-5.60e-3  
    y(6)*y(4)^2/(y(1)^5*y(2)^2)-0.14];  
%  
% the subroutine for calculating the mole fraction  
%  
function y=ex2_2_2y(x)  
D=4-2*x(1) +x(3)-4*x(4);
```

MATLAB program design:

— ex2_2_2.m —

```
if D == 0
    disp('D is zero')
    return
end
y(1)=(3-3*x(1) +x(2)-5*x(4))/D;
y(2)=(1-x(1)-x(2) +2*x(3)-2*x(4))/D;
y(3)=x(1)/D;
y(4)=(x(1)-x(2) +2*x(4))/D;
y(5)=(x(2)-x(3))/D;
y(6)=x(4)/D;
```

Execution results:

```
>> ex2_2_2
```

reaction rate of each equation

$x(1) = 6.816\text{e-}001$

$x(2) = 1.590\text{e-}002$

$x(3) = -1.287\text{e-}001$

$x(4) = 1.400\text{e-}005$

Execution results:

```
>> ex2_2_2
```

mole fraction of each component

$y(1) = 3.872e-001$

$y(2) = 1.797e-002$

$y(3) = 2.718e-001$

$y(4) = 2.654e-001$

$y(5) = 5.765e-002$

$y(6) = 5.580e-006$

Example 2-2-3

Analysis of a pipeline network

Figure 2.1 systematically illustrates a pipeline network installed on a horizontal ground surface. In the network, raw materials are fed to eight locations denoted as P_1, P_2, \dots, P_7 , and P_8 , and the flow rate (m^3/min) required at each of the eight locations is listed as follows (Leu, 1985):

$$P_{1Q} = 3.0, \quad P_{2Q} = 2.0, \quad P_{3Q} = 5.0, \quad P_{4Q} = 3.0$$

$$P_{5Q} = 2.0, \quad P_{6Q} = 2.0, \quad P_{7Q} = 5.0, \quad P_{8Q} = 2.0$$

Due to flow frictions, the head loss h (m) for each 1/2 side of the pipeline (such as P_1 to P_5) is given by

$$h = 0.5 |Q|^{0.85} Q$$

Note that the head loss is a function of the flow rate Q (m^3/min) in the pipeline. Based on the required flow rates P_{iQ} determine the corresponding flow rates, Q_1 to Q_9 , in the pipeline and the head at each of the eight locations.

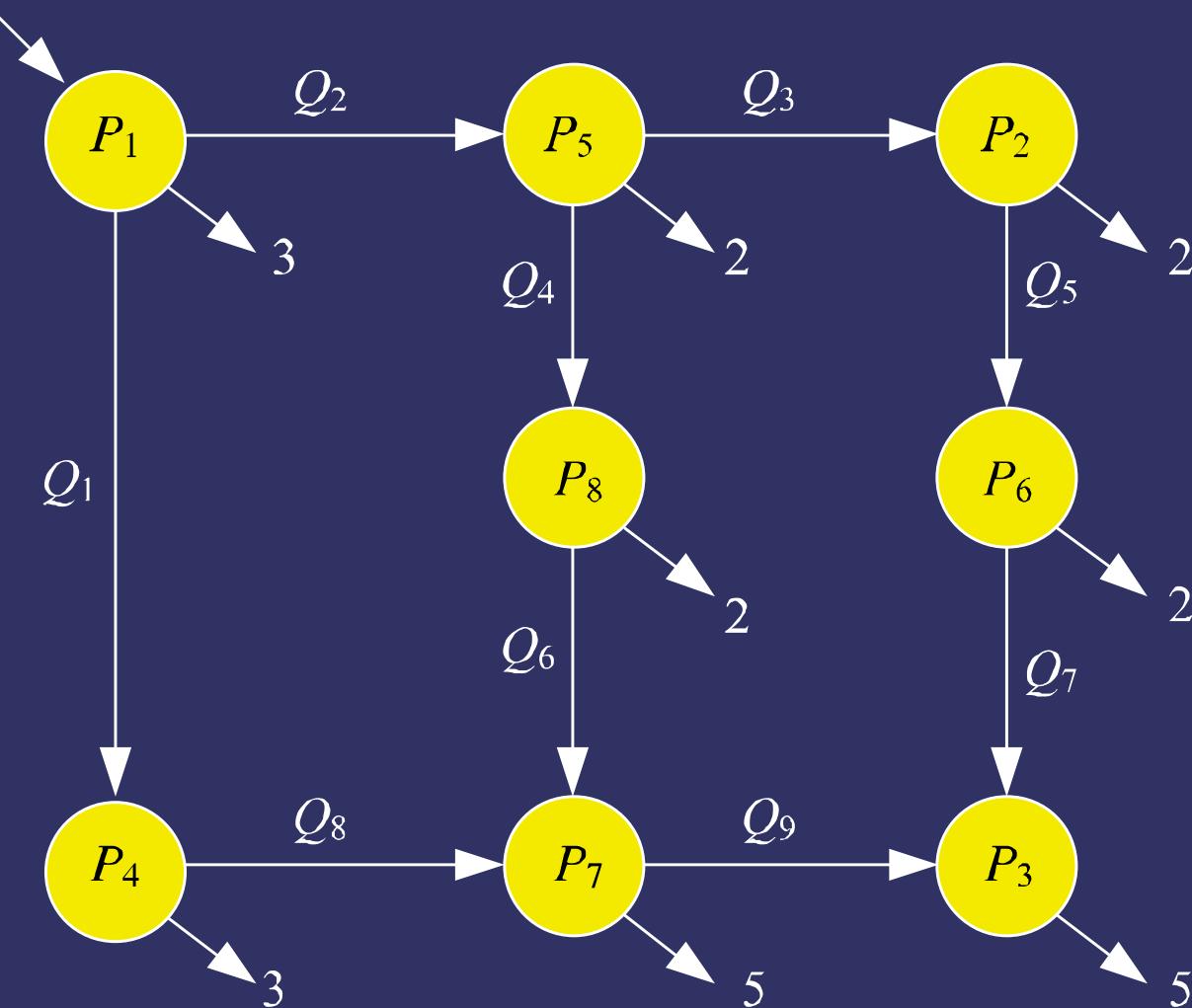


Figure 2.1 Schematic diagram of the pipeline network.

Problem formulation and analysis:

mass balance

$$\text{Mass balance at } P_2: Q_3 = Q_5 + P_{2Q}$$

$$\text{Mass balance at } P_3: Q_7 + Q_9 = P_{3Q}$$

$$\text{Mass balance at } P_1: Q_1 = Q_8 + P_{4Q}$$

$$\text{Mass balance at } P_5: Q_2 = Q_3 + Q_4 + P_{5Q}$$

$$\text{Mass balance at } P_6: Q_5 = Q_7 + P_{6Q}$$

$$\text{Mass balance at } P_7: Q_6 + Q_8 = Q_9 + P_{7Q}$$

$$\text{Mass balance at } P_8: Q_4 = Q_6 + P_{8Q}$$



Problem formulation and analysis:

Head loss balance by considering $h_{P_1P_5} + h_{P_5P_8} + h_{P_8P_7} + h_{P_1P_4} + h_{P_4P_7}$:

$$0.5|Q_2|^{0.85}Q_2 + 0.5|Q_4|^{0.85}Q_4 + 0.5|Q_6|^{0.85}Q_6 \\ = 2 \times 0.5|Q_1|^{0.85}Q_1 + 0.5|Q_8|^{0.85}Q_8$$

Head loss balance by considering

$$h_{P_5P_2} + h_{P_2P_6} + h_{P_6P_6} = h_{P_5P_8} + h_{P_8P_7} + h_{P_7P_3}$$

$$0.5|Q_3|^{0.85}Q_3 + 0.5|Q_5|^{0.85}Q_5 + 0.5|Q_7|^{0.85}Q_7 \\ = 0.5|Q_4|^{0.85}Q_4 + 0.5|Q_6|^{0.85}Q_6 + 0.5|Q_9|^{0.85}Q_9$$

MATLAB program design:

— ex2_2_3.m —

```
function ex2_2_3
%
% Example 2-2-3 pipeline network analysis
%
clear
clc
%
global PQ % declared as a global parameter
%
PQ=[3 2 5 3 2 2 5 2]; % flow rate required at each location
%
Q0=2*ones(1, 9); % initial guess value
%
% solving
%
Q=fsolve(@ex2_2_3f, Q0,1.e-6); % setting the solution accuracy as 1.e-6
%
```

MATLAB program design:

— ex2_2_3.m —

```
h(3)=10;
h(6)=h(3)+0.5*abs(Q(7))^0.85*Q(7);
h(2)=h(6)+0.5*abs(Q(5))^0.85*Q(5);
h(7)=h(3)+0.5*abs(Q(9))^0.85*Q(9);
h(8)=h(7)+0.5*abs(Q(6))^0.85*Q(6);
h(5)=h(2)+0.5*abs(Q(3))^0.85*Q(3);
h(4)=h(7)+0.5*abs(Q(8))^0.85*Q(8);
h(1)=h(5)+0.5*abs(Q(2))^0.85*Q(2);
%
% results printing
%
disp(' ')
disp('flow at each location (m^3/min)')
for i=1:9
fprintf('Q(%i)=%f \n',i,Q(i))
end
disp(' ')
disp('head at each location (m)')
```

MATLAB program design:

— ex2_2_3.m —

```
for i=1:8
fprintf('h(%i)=%7.3f \n',i,h(i))
end
%
% nonlinear simultaneous equations
%
function f=ex2_2_3f(Q)
%
global PQ
%
f=[Q(3)-Q(5)-PQ(2)
Q(7)+Q(9)-PQ(3)
Q(1 )-Q(8)-PQ(4)
Q(2)-Q(3)-Q(4)-PQ(5)
Q(5)-Q(7)-PQ(6)
Q(6)+Q(8)-Q(9)-PQ(7)
Q(4)-Q(6)-PQ(8)]
```

MATLAB program design:

— ex2_2_3.m —

```
0.5*abs(Q(2))^0.85*Q(2)+0.5*abs(Q(4))^0.85*Q(4)+0.5*abs(Q(6))^0.85*Q(6)-...
    abs(Q(1))^0.85*Q(1)-0.5*abs(Q(8))^0.85*Q(8)
0.5*abs(Q(3))^0.85*Q(3)+0.5*abs(Q(5))^0.85*Q(5)+0.5*abs(Q(7))^0.85*Q(7)-...
0.5*abs(Q(4))^0.85*Q(4)-0.5*abs(Q(6))^0.85*Q(6)-0.5*abs(Q(9))^0.85*Q(9)];
```

Execution results:

>> ex2_2_3

flow at each location (m³/min)

$$Q(1) = 8.599$$

$$Q(2) = 12.401$$

$$Q(3) = 5.517$$

$$Q(4) = 4.884$$

$$Q(5) = 3.517$$

$$Q(6) = 2.884$$

$$Q(7) = 1.517$$

$$Q(8) = 5.599$$

$$Q(9) = 3.483$$

head at each location (m)

$$h(1) = 80.685$$

$$h(2) = 16.202$$

$$h(3) = 10.000$$

$$h(4) = 27.137$$

$$h(5) = 27.980$$

$$h(6) = 11.081$$

$$h(7) = 15.031$$

$$h(8) = 18.578$$

Example 2-2-4

A material drying process through heat conduction and forced convection

Figure 2.2 schematically depicts a drying process (*Leu*, 1985), where a wet material with thickness of 6 cm is placed on a hot plate maintained at 100°C.

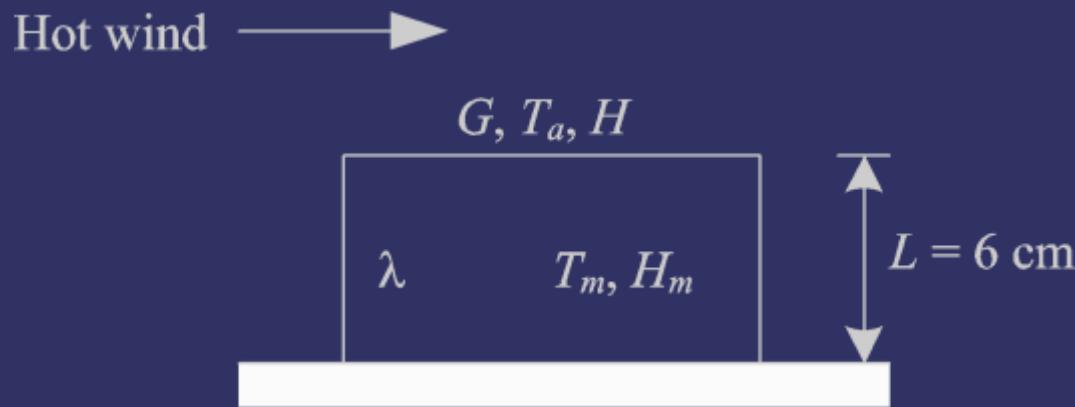


Figure 2.2 A material drying process through heat conduction and forced convection.

To dry the material, the hot wind with a temperature of 85°C and partial vapor pressure of 40 mmHg is blowing over the wet material at a flow rate of 3.5 m/s. The relevant operation conditions and assumptions are stated as follows:

- (1) During the drying process, the heat conduction coefficient of the wet material is kept constant, and its value is measured to be 1.0 kcal/m·hr·°C.
- (2) The convective heat transfer coefficient h (kcal/m·hr·°C) between the hot wind and the material surface is a function of mass flow rate of wet air G (kg/hr), which is given by

$$h = 0.015G^{0.8}$$

- (3) The saturated vapor pressure P_s (mmHg), which is a function of temperature (° C), can be expressed by the Antoine equations as follows:

$$\log P_s = 7.7423 - 1,554.16/(219+T), \quad 35 \leq T < 55 \text{ } ^\circ\text{C}$$

$$\log P_s = 7.8097 - 1,572.53/(219+T), \quad 55 \leq T \leq 85 \text{ } ^\circ\text{C}$$

- (4) The latent heat of water vaporization r_m (kcal/kg) is a function of temperature T ($^\circ\text{C}$), which is given by

$$r_m = 597.65 - 0.575 T$$

- (5) The mass flow rate of wet air G and its mass transfer coefficient M are measured to be the values of 1.225×10^4 kg/hr and $95.25 \text{ kg/m}^2 \cdot \text{hr}$, respectively.

Based on the operating conditions mentioned above, determine the surface temperature of the material T_m and the constant drying rate R_c , which is defined as

$$R_c = M(H_m - H)$$

Problem formulation and analysis:

$$\underbrace{M(H_m - H)r_m}_{\text{Heat of vaporization}} = \underbrace{h(T_a - T_m)}_{\text{Heat convection}} + \underbrace{\frac{k}{L}(T_p - T_m)}_{\text{Heat conduction from hot plate to the wet material}}$$

$$h = 0.015G^{0.8}$$

$$r_m = 597.65 - 0.575T$$

$$T_p = 100 \text{ } ^\circ\text{C}$$

$$L = 0.06 \text{ m}$$

$$k = 1 \text{ kcal/m}\cdot\text{hr}\cdot\text{ }^\circ\text{C}$$

$$T_a = 85 \text{ } ^\circ\text{C}$$

$$G = 1.225 \times 10^4 \text{ kg/hr}$$

$$M = 95.25 \text{ kg/m}^2\cdot\text{ hr}$$



Problem formulation and analysis:

$$H = \frac{18.02P}{28.97(760 - P)}$$

$$H_m = \frac{18.02P_s}{28.97(760 - P_s)}$$

MATLAB program design:

— ex2_2_3.m —

```
function ex2_2_4
%
% Example 2-2-4 A material drying process via heat conduction and forced convection
%
clear
clc
%
global h Ta M k L Tp H
%
G=1.225e4; % mass flow rate of wet air (kg/hr)
M=95.25; % mass transfer coefficient (kg/m^2.hr)
k=1; % heat transfer coefficient of the wet material (kcal/m.hr.°C)
Tp=100; % hot plate temperature (°C)
L=0.06; % material thickness (m)
Ta=85; % hot wind temperature (°C)
P=40; % partial pressure of water vapor (mmHg)
H=18.02*P/(28.97*(760-P)); % humidity of hot air (%)
h=0.015*G^0.8; % heat transfer coefficient (kcal/m^2.hr. °C)
```

MATLAB program design:

— ex2_2_3.m —

```
%  
% solving for Tm  
%  
Tm=fzero(@ex2_2_4f, 50);  
%  
Ps=ex2_2_4Ps(Tm); % eq. (2.2-12)  
%  
% calculating constant drying rate  
%  
Hm=18.02*Ps/(28.97* (760-Ps)); % eq. (2.2-17)  
Rc=M*(Hm-H); % eq. (2.2-14)  
%  
% results printing  
%  
fprintf('the surface temperature of the material =%.3f °C\n', Tm)  
fprintf('the constant drying rate = %.3f kg/m^2.hr\n', Rc)  
%
```

MATLAB program design:

— ex2_2_3.m —

```
% nonlinear function  
%  
function f = ex2_2_4f(Tm)  
%  
global h Ta M k L Tp H  
%  
Ps= ex2_2_4Ps(Tm); % eq.(2.2-12)  
%  
Hm=18.02*Ps/(28.97*(760-Ps)); % eq.(2.2-17)  
rm=597.65-0.575*Tm; % eq.(2.2-13)  
%  
f=h*(Ta-Tm)-M*(Hm-H)*rm-k/L*(Tm-Tp); % eq.(2.2-15)  
%  
% subroutine for calculating the saturated vapor pressure  
%  
function Ps=ex2_2_4Ps(Tm)  
if Tm < 55
```

MATLAB program design:

— ex2_2_3.m —

```
Ps=10^(7.7423-1554.16/(219+Tm));  
elseif Tm >= 55  
    Ps=10^(7.8097-1572.53/(219+Tm));  
end
```

Execution results:

```
>> ex2_2_4
```

the surface temperature of the material = 46.552°C

the constant drying rate = 3.444 kg/m².hr

Example 2-2-5

A mixture of 50 mol% propane and 50 mol% n-butane is to be distilled into 90 mol% propane distillate and 90 mol% n-butane bottoms by a multistage distillation column. The q value of raw materials is 0.5, and the feeding rate is 100 mol/hr. Suppose the binary distillation column is operated under the following operating conditions (*McCabe and Smith*, 1967):

- 1) The Murphree *vapor efficiency* E_{MV} is 0.7;
- 2) The entrainment effect coefficient ε is 0.2;
- 3) The reflux ratio is 2.0;
- 4) The operating pressure in the tower is 220 Psia;

- 5) The equilibrium coefficient ($K = y/x$), which depends on the absolute temperature T (°R), is given by

$$\ln K = A/T + B + CT$$

where the coefficients A, B, and C under 220 Psia for propane and n-butane are listed in the table below.

Component	A	B	C
Propane	-3,987.933	8.63131	-0.00290
n-Butane	-4,760.751	8.50187	-0.00206

Answers to the following questions:

- 1) The theoretical number of plates and the position of the feeding plate.
- 2) The component compositions in the gas phase and the liquid phase at each plate.
- 3) The temperature of the re-boiler and the *bubble point* at each plate.

Problem formulation and analysis:

(1) Overall mass balance:

$$D = \frac{F(z_F - x_W)}{x_D - x_W}$$

$$W = F - D$$

Name of variable	Physical meaning	Name of variable	Physical meaning
D	Flow rate of the distillate	x_D	Composition of the distillate
F	Flow rate of the feed	x_W	Composition of the bottoms
W	Flow rate of the bottoms	z_F	Composition of the feed

Problem formulation and analysis:

(1) Overall mass balance:

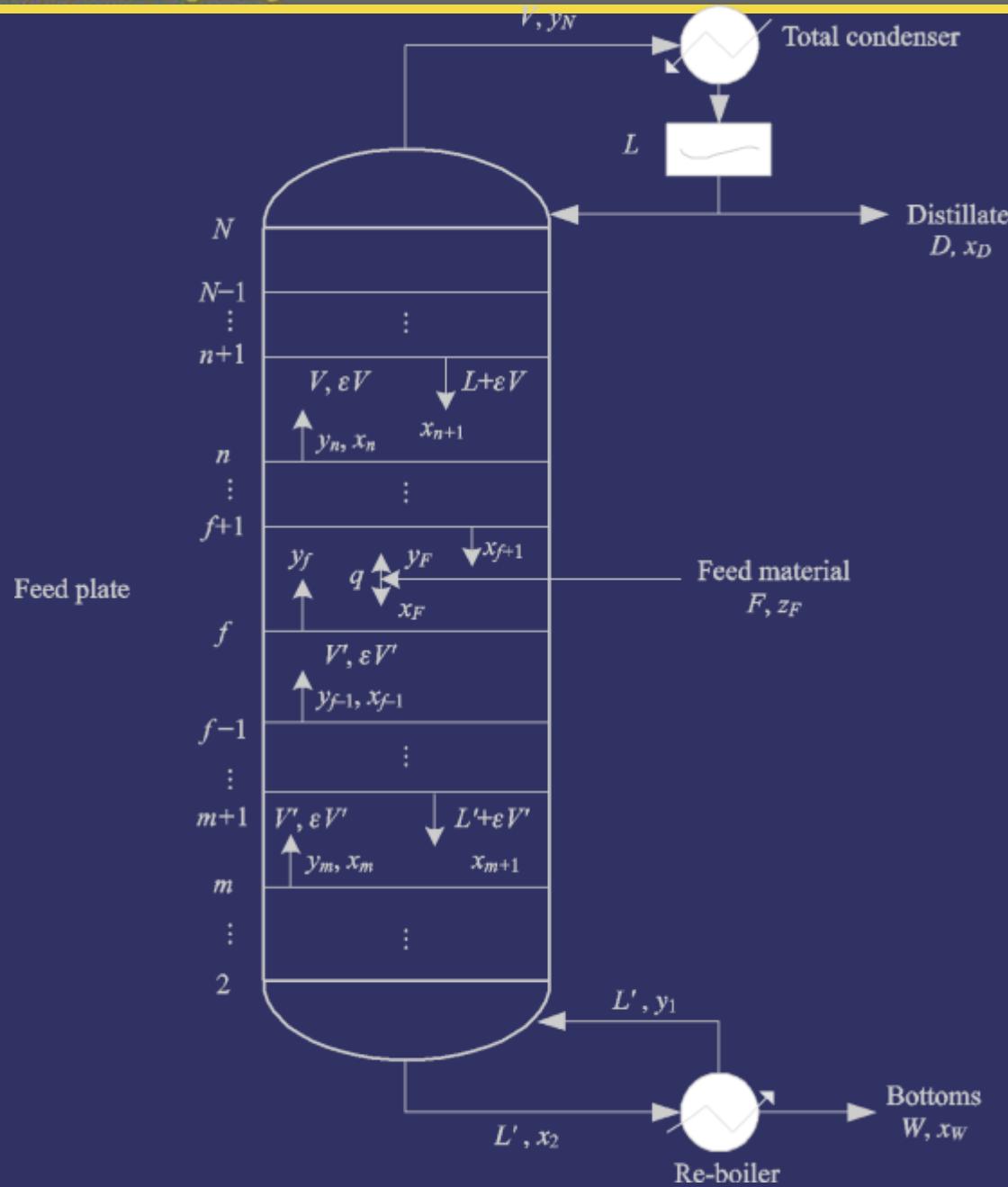
$$R_D = L/D$$

$$L = R_D D$$

$$V = L + D$$

Solution of Nonlinear Equations

02



Problem formulation and analysis:

(2) Material balance on the feed plate:

$$L' = L + qF$$

$$V' = V - (1 - q)F$$

$$q = \frac{\text{heat required to convert one mole of the feed into saturated vapor}}{\text{molar latent heat of vaporization}}$$

q value	Condition of the feed
$q > 1$	The temperature of the feed is lower than the boiling point
$q = 1$	Saturated liquid
$0 < q < 1$	Mixture of liquid and vapor
$q = 0$	Saturated vapor
$q < 0$	Overheated vapor

Problem formulation and analysis:

(3) Material balance made for the re-boiler

$$x_2 = \frac{V'y_1 + Wx_W}{L'}$$

(4) Stripping section

$$x_{m+1} = \frac{V'y_m + \varepsilon V'x_m + Wx_W}{L' + \varepsilon V'}$$

$$y_m = E_{MV} (y_m^* - y_{m-1}) + y_{m-1}$$

Problem formulation and analysis:

(5) Enriching section

$$x_{n+1} = \frac{Ly_n + \varepsilon V x_n - Dx_D}{L + \varepsilon V}$$

(6) Feed plate

$$x_{f+1} = \frac{V'y_f + \varepsilon V' x_f + (1 - q)Fy_F - Dx_D}{L + \varepsilon V'}$$

(A) Calculation of the flash point at the feed plate:

$$y_{Fi} = \frac{z_{Fi}}{1 + q \left(\frac{1}{K_i} - 1 \right)}, \quad i = 1, 2$$

Problem formulation and analysis:

$$x_{Fi} = \frac{y_{Fi}}{K_i}$$

where

$$K_i = \exp (A_i / T + B_i + C_i T) \quad \square$$

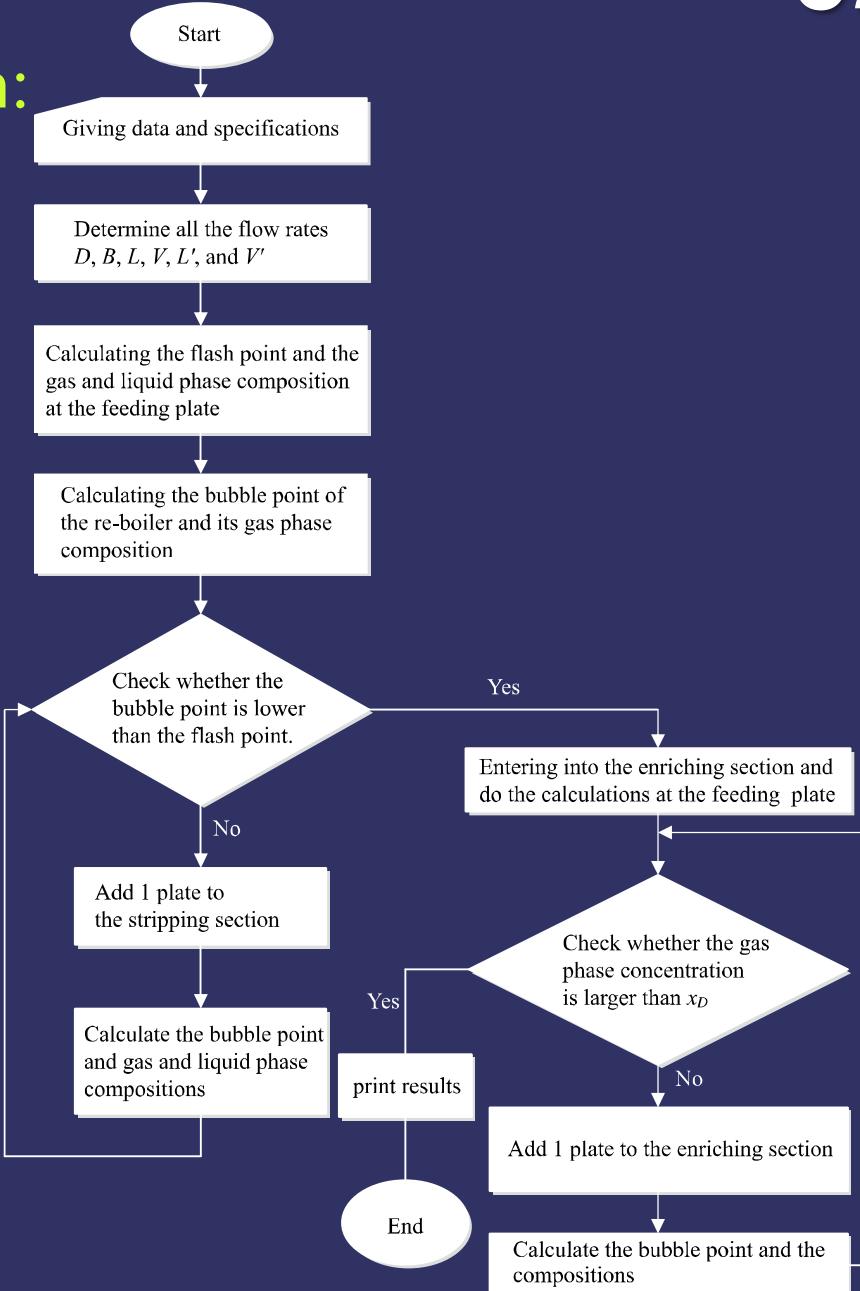
$$\sum_{i=1}^2 y_{Fi} = 1$$

(B) Calculation of the flash point at the feed plate:

$$y_{ni}^* = K_i x_{ni}$$

$$\sum_{i=1}^2 y_{ni}^* = 1$$

MATLAB program design:



MATLAB program design:

— ex2_2_5.m —

```
function ex2_2_5
%
% Example 2-2-5 analysis of a continuous multi-stage binary distillation
%
clear; close all
clc
%
% declare the global variables
%
global A B C zf q
global A B C xc
%
% operation data
%
zf=[0.5 0.5]; % feed composition
xd=[0.9 0.1]; % distillate composition
xw=[0.1 0.9]; % composition of the bottom product
F=100; % feed flow rate (mol/h)
```

MATLAB program design:

— ex2_2_5.m —

```

q=0.5; % q value of the feed
Emv=0.7; % Murphree vapor efficiency
Ee=0.2; % entrainment coefficient
RD=2; % reflux ratio
%
A=[-3987.933 -4760.751]; % parameter A
B=[8.63131 8.50187]; % parameter B
C=[-0.00290 -0.00206]; % parameter C
%
% calculating all flow rates (D,W,L,V, LS, and VS)
%
D=F*(zf(1)-xw(1))/(xd(1)-xw(1)); % distillate flow rate
W=F-D; % flow rate of the bottom product
L=RD*D; % liquid flow rate in the enriching section
V=L+D; % gas flow rate in the enriching section
LS=L+q*F; % liquid flow rate in the stripping section
VS=V-(1-q)*F; % gas flow rate in the stripping section
%

```

MATLAB program design:

— ex2_2_5.m —

```
% calculating the flash point at the feed plate
%
Tf=fzero(@ex2_2_5f1,620); % flash point
K=exp(A/Tf+B+C*Tf);
yf=zf./(1 +q*(1./K-1));    % gas phase composition at the feed plate
xf=yf./K;                  % liquid phase composition at the feed plate
Tf=Tf-459.6;                % converted to °F
%
% equilibrium in the re-boiler
%
xc=xw;
Tw=fzero(@ex2_2_5f2,620);
Kw=exp(A/Tw+B+C*Tw);
yw=Kw.*xw;    % gas phase composition in the re-boiler
Tw=Tw-459.6;   % bubble point (°F) of the re-boiler
%
% calculating the liquid phase composition in the re-boiler
```

MATLAB program design:

— ex2_2_5.m —

```
m=1;
x(m,:)=xw;
y(m,:)=yw;
x(m+1,:)=(VS*y(m,:)+W*xw)/LS;
%
% calculating for the stripping section
%
T(m)=Tw;
while T(m)>Tf
m=m+1;
xc=x(m,:);
T (m)=fzero(@ex2_2_5f2,620);
K(m,:)=exp(A/T(m)+B+C*T(m));
ys(m,:)=K(m,:).*x(m,:);
y(m,:)=Emv*(ys(m,:)-y(m-1,:))+y(m-1,:);
x(m+1,:)=(VS*y(m,:)+Ee*VS*x(m,:)+W*x(m-1,:))/(LS+Ee*VS);
T(m)=T(m)-459.6;
end
```

MATLAB program design:

— ex2_2_5.m —

```
%  
% composition calculation  
%  
n=m+1;  
x(n,:)=(VS*yf+Ee*VS*xf+(1-q)*F*yf-D*xd)/(L+Ee*VS);  
xc=x(n,:);  
T(n)=fzero(@ex2_2_5f2,620);  
K(n,:)=exp(A/T(n)+B+C*T(n));  
ys(n,:)=K(n,:).*x(n,:);  
y(n,:)=Emv*(ys(n,:)-y(n-1,:))+y(n-1,:);  
T(n)=T(n)-459.6;  
x(n+1,:)=(V*y(n,:)+Ee*V*x(n,:)-D*xd)/(L+Ee*V);  
%  
% calculation for the enriching section  
%  
while y(n,1)<xd(1)  
n=n+1;
```

MATLAB program design:

— ex2_2_5.m —

```
xc=x(n,:);  
T(n)=fzero(@ex2_2_5f2,620);  
K(n,: )=exp(A/T(n)+B+C*T(n));  
ys(n,: )=K(n,: ).*x(n,: );  
y(n,: )=Emv*(ys(n,: )-y(n-1,: ))+y(n-1,: );  
x(n+ 1,: )=(V*y(n,: )+Ee*V*x(n,: )-D*xd)/(L+Ee*V);  
T(n)=T(n)-459.6;  
end  
%  
N=n+1; % total number of plates  
%  
% distillate concentration  
%  
x(N,: )=y(n,: );  
%  
% results printing  
%
```

MATLAB program design:

— ex2_2_5.m —

```
disp(' ')
disp(' ')
disp("The bubble point and compositions at each plate.")
disp(' ')
disp('plate      liquid          gas          temperature')
disp('no.       phase          phase')
disp('---- ----- ----- -----')
for i=1:n
fprintf('%2i %10.5f %10.5f %10.5f %10.5f %10.2f\n',i,x(i,1),x(i,2),...
y(i,1), y(i,2),T(i))
end
fprintf('%2i %10.5f %10.5f\n',N,x(N,1),x(N,2))
fprintf('\n total no. of plates= %d; the feed plate no.= %d',N,m)
fprintf('\n distillate concentration = [%7.5f %10.5f]\n', x(N,1),x(N,2))
%
% composition chart plotting
%
```

MATLAB program design:

— ex2_2_5.m —

```
plot(1 :N,x(:, 1),1 :N,x(:,2))
hold on
plot([m m],[0  1], '--')
hold off
xlabel('plate number')
ylabel('liquid phase composition')
title('liquid phase composition chart')
text(3,x(3,1),'\leftarrow \it{x}_1 ','FontSize', 10)
text(3,x(3,2),'\leftarrow \it{x}_2 ','FontSize', 10)
text(m,0.5,' \leftarrow position of the feeding plate ','FontSize',10)
axis([1 N 0 1])
%
% flash point determination function
%
function y=ex2_2_5f1(T)
global A  B  C zf q
K=exp(A./T+B+C*T);
yf=zf./(1 +q*(1./K-1));
```

MATLAB program design:

— ex2_2_5.m —

```
xf=yf./K;
y=sum(yf)-1 ;
%
% bubble point determination function
%
function y=ex2_2_5f2(T)
global A B C xc
K=exp(A./T+B+C.*T);
yc=K.*xc;
y=sum(yc)-1;
```

Execution results:

```
>> ex2_2_5
```

The bubble point and compositions at each plate.

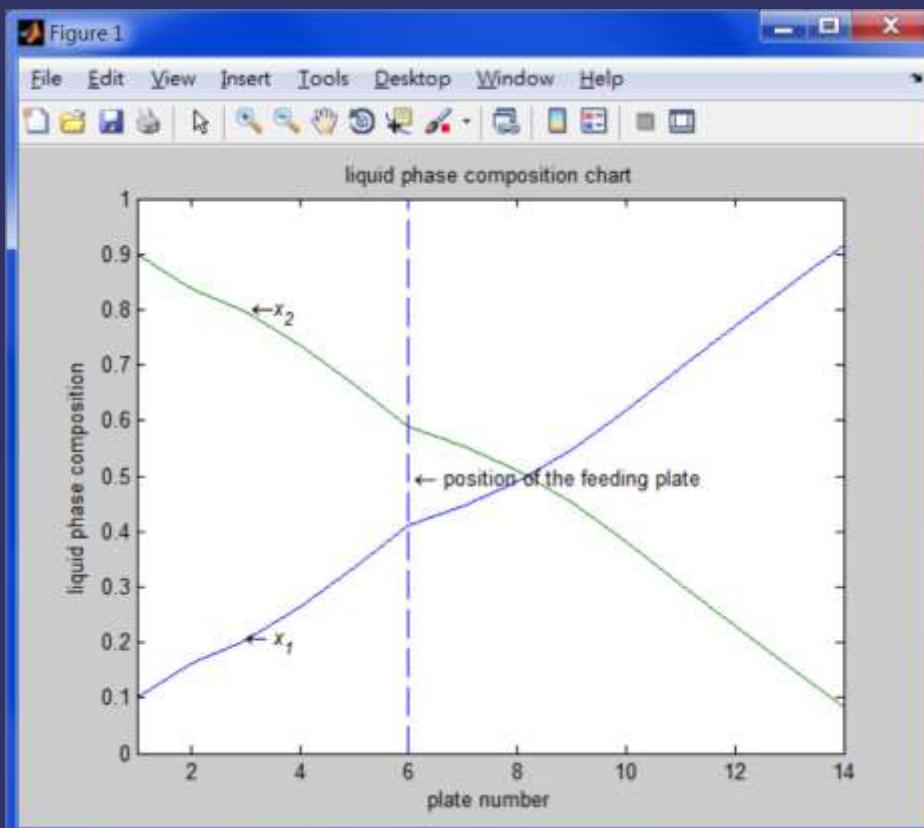
plate no.	liquid phase		gas phase		temperature
1	0.10000	0.90000	0.19145	0.80855	196.35
2	0.16097	0.83903	0.26333	0.73667	189.16
3	0.20325	0.79675	0.33070	0.66930	184.28
4	0.26578	0.73422	0.41292	0.58708	177.25
5	0.33394	0.66606	0.49800	0.50200	169.85
6	0.41040	0.58960	0.58324	0.41676	161.88
7	0.44630	0.55370	0.63418	0.36582	158.27
8	0.48858	0.51142	0.67733	0.32267	154.12
9	0.54813	0.45187	0.72610	0.27390	148.47
10	0.61815	0.38185	0.77825	0.22175	142.10
11	0.69448	0.30552	0.82978	0.17022	135.50
12	0.77156	0.22844	0.87685	0.12315	129.18
13	0.84365	0.15635	0.91688	0.08312	123.57
14	0.91688	0.08312			

Execution results:

```
>> ex2_2_5
```

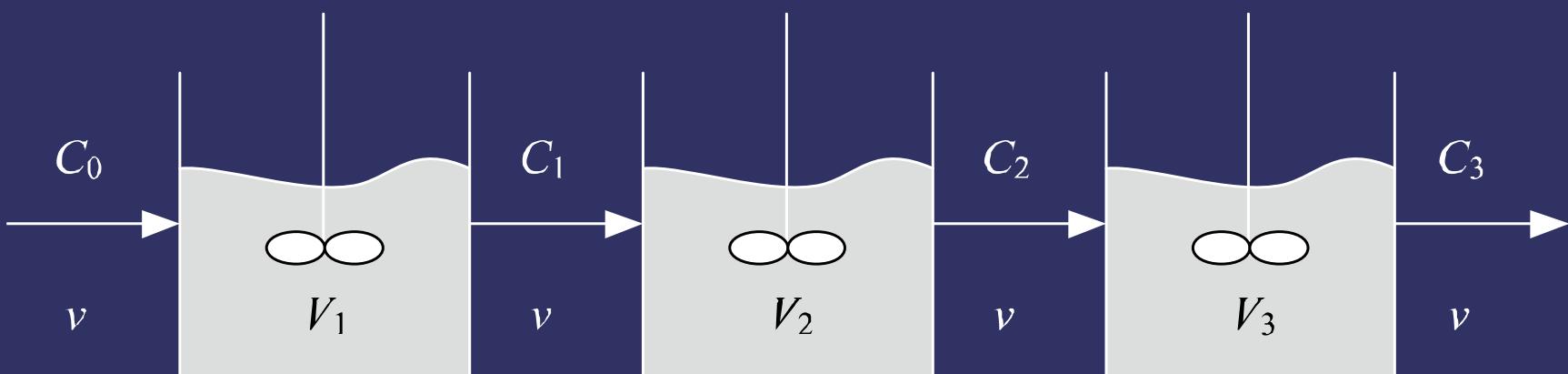
total no. of plates= 14; the feed plate no.= 6

distillate concentration = [0.91688 0.08312]



Example 2-2-6

Analysis of a set of three CSTRs in series.



In each of the reactors, the following liquid phase reaction occurs



The reactor volumes and the reaction rate constants are listed in the following table:

Reactor number	Volume, V_i (L)	Reaction rate constant, K_i (L/g – mol·min)
1	5	0.03
2	2	0.05
3	3	0.10

If the concentration of component A fed into the first reactor is $C_0 = 1$ g-mol/L and the flow rate v is kept constant at the value of 0.5 L/min, determine the steady-state concentration of component A and the conversion rate of each reactor.



Problem formulation and analysis:

steady state, mass

$$vC_{i-1} - vC_i - k_i C_i^{1.5} V_i = 0, \quad i = 1, 2, 3$$

MATLAB program design:

— ex2_2_6.m —

```
function ex2_2_6
%
% Example 2-2-6 Analysis of a set of three CSTRs in series
%
Clear; clc;
%
global v VV kk cc
%
% given data
%
C0=1; % inlet concentration (g-mol/L)
v=0.5; % flow in the reactor (L/min)
V=[5 2 3]; % volume of reactor (L)
k=[0.03 0.05 0.1]; % reaction rate constant (L/g.mol.min)
%
% start calculating
%
```

MATLAB program design:

— ex2_2_6.m —

```
C=zeros(n+1,1); % initialize concentrations
x=zeros(n+1,1); % initialize conversion rates
C(1)=C0; % inlet concentration
x(1)=0; % initial conversion rate
for i=2:n+1
    VV=V(i-1);
    kk=k(i-1);
    cc=C(i-1);
    C(i)=fzero(@ex2_2_6f,cc); % use inlet conc. as the initial guess value
    x(i)=(C0-C(i))/C0; % calculating the conversion rate
end
%
% results printing
%
for i=2:n+1
    fprintf('The exit conc. of reactor %d is %.3f with a conversion rate of... %.3f.\n',i-1,C(i),x(i))
```

MATLAB program design:

— ex2_2_6.m —

```
end
%
% reaction equation
%
function f=ex2_2_6f(C)
global v  VV  kk  cc
f=v*cc-v*C-kk*VV*C^1.5;
```

Execution results:

```
>> ex2_2_6
```

The conc. at the outlet of reactor 1 is 0.790 with a conversion rate of 0.210.

The conc. at the outlet of reactor 2 is 0.678 with a conversion rate of 0.322.

The conc. at the outlet of reactor 3 is 0.479 with a conversion rate of 0.521.

2.4 Summary of MATLAB commands related to this chapter

a. Solution of a single-variable nonlinear equation:

The command format:

`x=fzero(fun, x0)`

`x=fzero(fun, x0, options)`

`x=fzero(fun, x0, options, P1 , P2, ...)`

`[x, fval]=fzero(...)`

`[x, fval, exitflag]=fzero(...)`

`[x, fval, exitflag, output]=fzero(...)`

b. Solution of a set of nonlinear equations

The command format:

`x=fsolve(fun, x0)`

`x=fsolve(fun, x0, options)`

`x=fsolve(fun, x0, options, P1 , P2, ...)`

`[x, fval]=fsolve(...)`

`[x, fval, exitflag]=fsolve(...)`

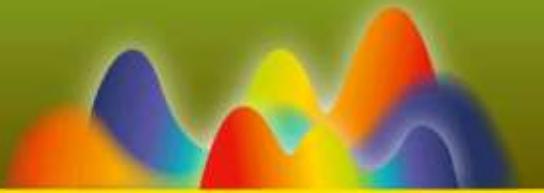
`[x, fval, exitflag, output] =fsolve(...)`

`[x, fval, exitflag, output, Jacobian] =fsolve(...)`



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 3

Interpolation, Differentiation, and Integration

版權所有・請勿翻製

- ① This chapter introduces the MATLAB commands for interpolation, differentiation, and integration.
- ② The relevant applications to chemical engineering problems will also be demonstrated.

3.1 Interpolation commands in MATLAB

► 3.1.1 One-dimensional interpolation

Example 3-1-1

Interpolation of vapor pressure

- experimental

Temperature (°C)	20	40	50	80	100
Vapor pressure (mmHg)	17.5	55.3	92.5	355.0	760.0

Example 3-1-1

- Now, it is required to determine the vapor pressures at those temperatures that are not included in the experiments, say 30, 60, 70, and 90°C, based on the above experimental data.

$$yi = \text{interp1}(x, y, xi, \text{method})$$

Input	Description
x	Data of independent variables
y	Data of dependent variables
xi	The point or vector that needs to get the interpolated output value

Method	Description
‘nearest’	Use the nearest data as the interpolated values
‘linear’	Linear interpolation method (defaulted)
‘spline’	Cubic spline interpolation method
‘cubic’	This method employs the piecewise cubic Hermite interpolation algorithm, which not only preserves the <i>monotonicity</i> of the original data but also maintains the <i>derivative</i> continuity of these data points.

ex3_1_1.m

```
%  
% Example 3-1-1 vapor pressure interpolation using interp1  
%  
% comparisons of interpolation methods  
%  
clear  
clc  
%  
% experimental data  
%  
x=[20 40 50 80 100]; % temperatures (°C)  
y=[17.5 55.3 92.5 355.0 760]; % vapor pressures (mmHg)  
%  
% interpolation with various methods  
%  
xi=[30 60 70 90]; % temperatures of interest  
y1=interp1(x,y,xi,'nearest'); % method of nearest  
y2=interp1(x,y,xi,'linear'); % linear interpolation  
y3=interp1(x,y,xi,'spline'); % cubic spline interpolation  
y4=interp1(x,y,xi,'cubic'); % piecewise cubic Hermite %interpolation
```

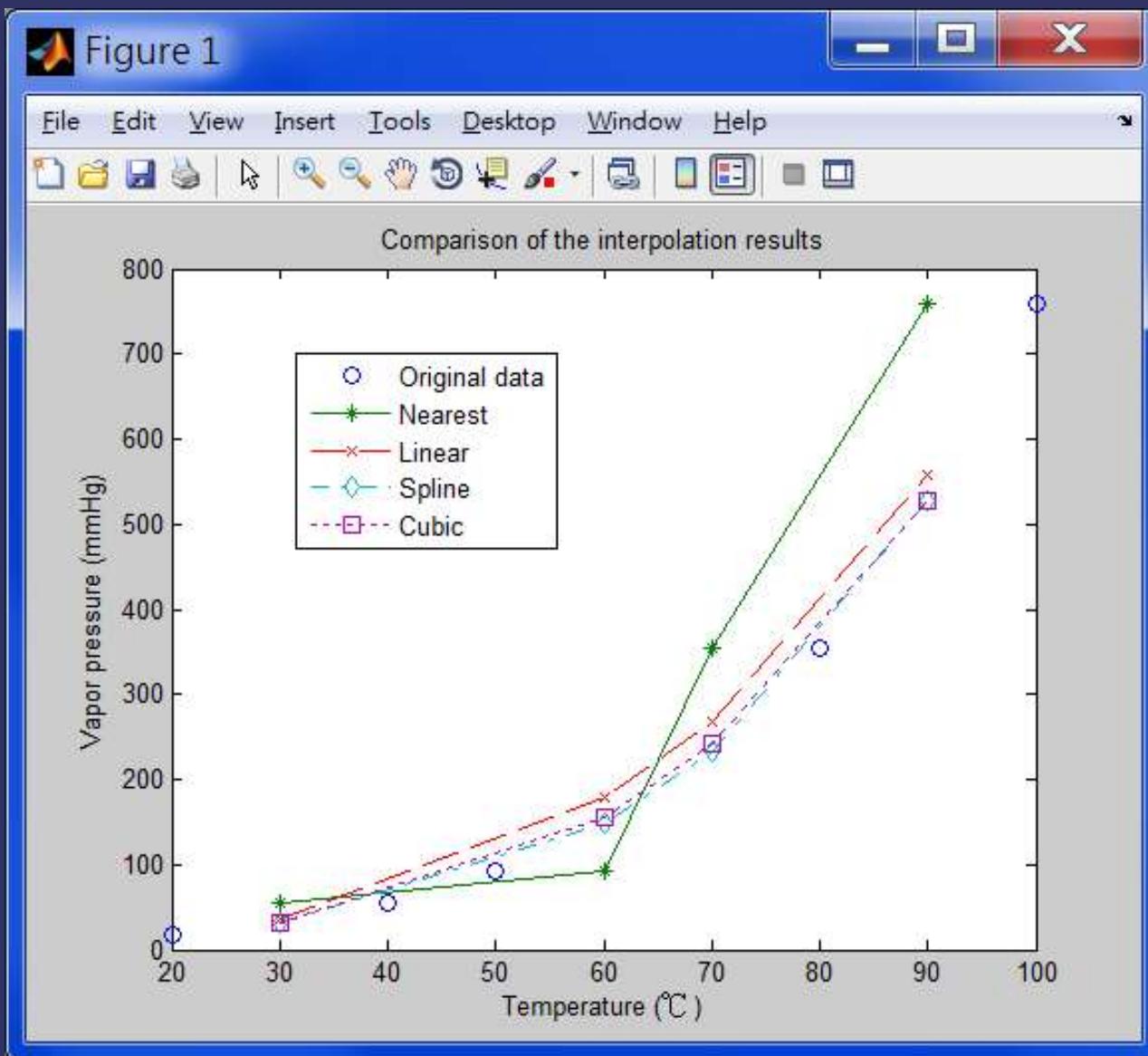
ex3_1_1.m

```
%  
% method comparisons  
%  
disp(' ')  
disp('Vapor pressure interpolation results')  
disp(' ')  
disp("Temp    Nearest   Linear   Spline   Cubic")  
for i=1:length(xi)  
    fprintf('\n %3i  %8.2f  %8.2f  %8.2f  %8.2f,... xi(i),y1(i),y2(i),y3(i),y4(i))  
end  
disp(' ')  
%  
% results plotting  
%  
plot(x,y,'o',xi,y1,'*-',xi,y2,'x--',xi, y3,'d-.',xi, y4,'s:')  
legend('Original data', 'Nearest','Linear','Spline','Cubic')  
xlabel('Temperature (°C)')  
ylabel('Vapor pressure (mmHg)')  
title('Interpolation Results Comparison')
```

```
>> ex3_1_1
```

Vapor pressure interpolation results

Temp	Nearest	Linear	Spline	Cubic
30	55.30	36.40	31.61	31.57
60	92.50	180.00	148.28	154.40
70	355.00	267.50	232.21	242.03
90	760.00	557.50	527.36	526.75



3.1.2 Two-dimensional interpolation

Example 3-1-2

The reaction rate interpolation of a synthetic reaction

r0 Ph	0.7494	0.6742	0.5576	0.5075	0.4256
Pb	0.2182	0.2134	0.2048	0.1883	0.1928
0.2670	0.2213	0.2208	0.2208	0.1695	0.1432
0.3424	0.2516	0.2722	0.2722	0.1732	0.1771
0.4342	0.2832	0.3112	0.3112	0.1892	0.1630
0.5043					

- Estimate the reaction rates when
 - 1) $P_h = 0.6$ and $P_b = 0.4$ atm
 - 2) $P_h = 0.5$ atm and the values of P_b are 0.3, 0.4, and 0.5 atm, respectively.

$zi = \text{interp2}(x, y, z, xi, yi, \text{method})$

Input argument	Description
x	The first independent data vector
y	The second independent data vector
z	Data matrix of the dependent variable corresponding to independent variables x and y
xi, yi	The coordinate points of the independent variables whose interpolated values are to be determined

Method	Description
‘nearest’	Use the nearest data as the interpolation values
‘linear’	Bilinear interpolation (defaulted)
‘cubic’	Bilinear cubic interpolation (bicubic)
‘spline’	Spline interpolation method

```
>> Ph=[0.7494 0.6742 0.5576 0.5075 0.4256]; % Ph data (vector)  
>> Pb=[0.2670 0.3424 0.4342 0.5043]; % Pb data (vector)  
>> r0=[ 0.2182 0.2134 0.2048 0.1883 0.1928  
      0.2213 0.2208 0.2139 0.1695 0.1432  
      0.2516 0.2722 0.2235 0.1732 0.1771  
      0.2832 0.3112 0.3523 0.1892 0.1630]; % reaction rate data
```

```
>> r1=interp2(Ph, Pb, r0, 0.6, 0.4, 'cubic') % using the bicubic method
```

r1 =

0.2333

```
>> r2=interp2(Ph, Pb, r0, 0.5, [0.3 0.4 0.5], 'cubic')
```

r2 =

0.1707 % interpolated reaction rate for Ph=0.5, Pb=0.3

0.1640 % interpolated reaction rate for Ph=0.5 and Pb=0.4

0.1649 % interpolated reaction rate for Ph=0.5 and Pb=0.5

3.2 Numerical differentiation

$$y' = \frac{dy}{dx} = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x}$$

- *Forward difference:*

$$y'_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{\Delta x}, \quad i = 1, 2, \dots, N-1$$

- *Backward difference:*

$$y'_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} = \frac{y_i - y_{i-1}}{\Delta x}, \quad i = 2, 3, \dots, N$$

- *Central difference:*

$$y'_i = \frac{1}{2} \left(\frac{y_i - y_{i-1}}{\Delta x} + \frac{y_{i+1} - y_i}{\Delta x} \right) = \frac{y_{+1i} - y_{i-1}}{2\Delta x}, \quad i = 2, 3, \dots, N-1$$

3.2.1 The application of the MATLAB diff command in numerical differentiation

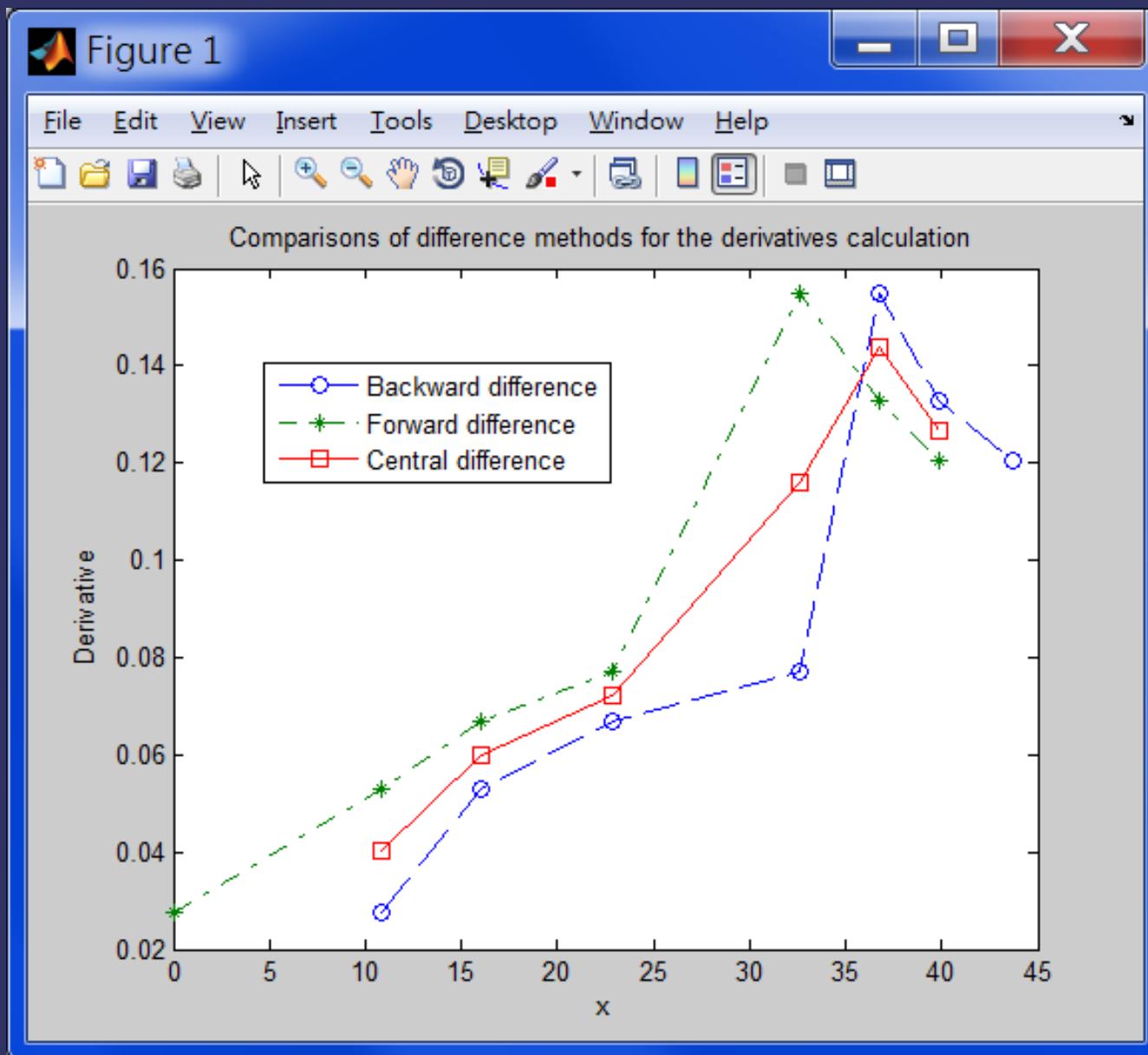
`d=diff(y)`

```
>> y=[1 5 2];
>> d=diff(y)
d=
4    -3
```

Table 3.1 Experimental data on the flow rate and pressure drop

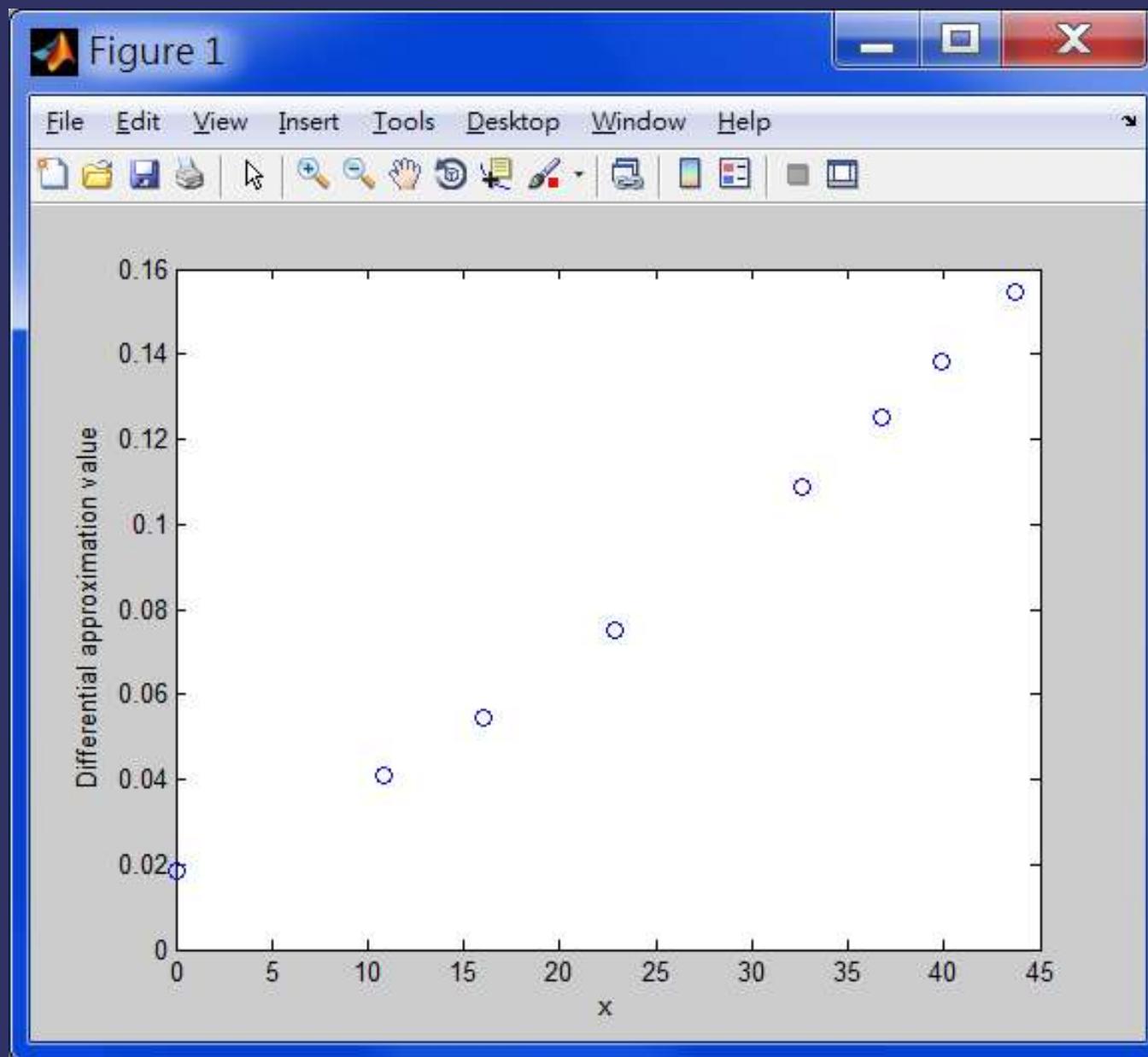
x (flow rate, cm/s)	0	10.80	16.03	22.91	32.56	36.76	39.88	43.68
y (pressure drop, kPa)	0	0.299	0.576	1.036	1.781	2.432	2.846	3.304

```
>> x=[0 10.80 16.03 22.91 32.56 36.76 39.88 43.68]; % flow rate (cm/s)  
>> y=[0 0.299 0.576 1.036 1.781 2.432 2.846 3.304]; % pressure drop (kPa)  
>> n=length(x);  
>> d1=diff(y)./diff(x); % forward and backward differences  
>> d2=0.5*(d1(1:n-2)+d1(2:n-1)); % central difference  
>> plot(x(2: n), d1, 'o --',x(1: n-1), d1,'*-. ',x(2: n-1), d2, 's- ')  
>> xlabel('x')  
>> ylabel('Derivative')  
>> title('Comparisons of difference methods for the derivatives calculation ')  
>> legend('Backward difference', 'Forward difference', 'Central difference')
```



3.2.2 Polynomial fitting and its application to the calculation of derivatives of a data set

```
>> x=[0 10.80 16.03 22.91 32.56 36.76 39.88 43.68]; % flow rate (cm/s)  
>> y=[0 0.299 0.576 1.036 1.781 2.432 2.846 3.304]; % pressure drop (kPa)  
>> p=polyfit(x, y, 3); % fit the data into a polynomial of third order  
>> dp=polyder(p); % derive the first-order derivative of the polynomial p  
>> dydx=polyval(dp, x); % calculating the derivatives based on the derived  
polynomial dp  
>> plot(x, dydx, 'o')  
>> xlabel('x')  
>> ylabel('Differential approximation value')
```



3.2.3 Higher-order derivatives of a data set

I. Derivatives calculation based on forward difference

A. The differential approximation formulae with an error of 0 (h)

$$y'_i = \frac{1}{h} (y_{i+1} - y_i)$$

$$y''_i = \frac{1}{h^2} (y_{i+2} - 2y_{i+1} + y_i)$$

$$y'''_i = \frac{1}{h^3} (y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i)$$

$$y^{(4)}_i = \frac{1}{h^4} (y_{i+4} - 4y_{i+3} + 6y_{i+2} - 4y_{i+1} + y_i)$$

I. Derivatives calculation based on forward difference

B. The differential approximation formulae with an error of 0 (h^2)

$$y'_i = \frac{1}{2h}(-y_{i+2} + 4y_{i+1} - 3y_i)$$

$$y''_i = \frac{1}{h^2}(-y_{i+3} + 4y_{i+2} - 5y_{i+1} + 2y_i)$$

$$y'''_i = \frac{1}{2h^3}(-3y_{i+4} + 14y_{i+3} - 24y_{i+2} + 18y_{i+1} - 5y_i)$$

$$y_i^{(4)} = \frac{1}{h^4}(-2y_{i+5} + 11y_{i+4} - 24y_{i+3} + 26y_{i+2} - 14y_{i+1} + 3y_i)$$

II. Derivatives calculation based on backward difference

A. The differential approximation formulae with an error of $O(h)$

$$y'_i = \frac{1}{h}(y_i - y_{i-1})$$

$$y''_i = \frac{1}{h^2}(y_i - 2y_{i-1} + y_{i-2})$$

$$y'''_i = \frac{1}{h^3}(y_i - 3y_{i-1} + 3y_{i-2} - y_{i-3})$$

$$y^{(4)}_i = \frac{1}{h^4}(y_i - 4y_{i-1} + 6y_{i-2} - 4y_{i-3} + y_{i-4})$$



II. Derivatives calculation based on backward difference

B. The differential approximation formulae with an error of 0 (h^2)

$$y'_i = \frac{1}{2h} (3y_i - 4y_{i-1} + y_{i-2})$$

$$y''_i = \frac{1}{h^2} (2y_i - 5y_{i-1} + 4y_{i-2} - y_{i-3})$$

$$y'''_i = \frac{1}{2h^3} (5y_i - 18y_{i-1} + 24y_{i-2} - 14y_{i-3} + 3y_{i-4})$$

$$y^{(4)}_i = \frac{1}{h^4} (3y_i - 14y_{i-1} + 26y_{i-2} - 24y_{i-3} + 11y_{i-4} - 2y_{i-5})$$

III. Derivatives calculation based on central difference

A. Differential approximation formulae with an error of $O(h^2)$

$$y'_i = \frac{1}{2h} (y_{i+1} - y_{i-1})$$

$$y''_i = \frac{1}{h^2} (y_{i+1} - 2y_i + y_{i-1})$$

$$y'''_i = \frac{1}{2h^3} (y_{i+2} - 2y_{i+1} + 2y_{i-1} - y_{i-2})$$

$$y^{(4)}_i = \frac{1}{h^4} (y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2})$$



III. Derivatives calculation based on central difference

B. Differential approximation formulae with an error of $O(h^4)$

$$y'_i = \frac{1}{12h}(-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2})$$

$$y''_i = \frac{1}{12h^2}(-y_{i-2} + 16y_{i+1} - 30y_i + 16y_{i-1} - y_{i-2})$$

$$y'''_i = \frac{1}{8h^3}(-y_{i+3} + 8y_{i+2} - 13y_{i+1} + 13y_{i-1} - 8y_{i-2} + y_{i-3})$$

$$y^{(4)}_i = \frac{1}{6h^4}(-y_{i+3} + 12y_{i+2} - 39y_{i+1} + 56y_i - 39y_{i-1} + 12y_{i-2} - y_{i-3})$$

`dy=numder(y, h, order, method, error)`

Input	Description
y	The data points of the dependent variable y
h	The interval of the independent variable (default value is 1)
order	Order of derivative (up to 4), default value is 1 (first-order derivative)
method	<p>Method selection (three options):</p> <ul style="list-style-type: none"> 1: Forward difference -1: Backward difference 0: Central difference <p>Defaulted option is central difference.</p>
error	Used to indicate the order of errors in the formulae. For backward and forward differences, the error may be set to 1 or 2, and 2 or 4 for central difference. Default value is 2.

Example 3-2-1

Pressure gradient in a fluidized bed

A set of pressure measurements of a fluidized bed is given as follows (*Constantinides and Mostoufi*, 1999):

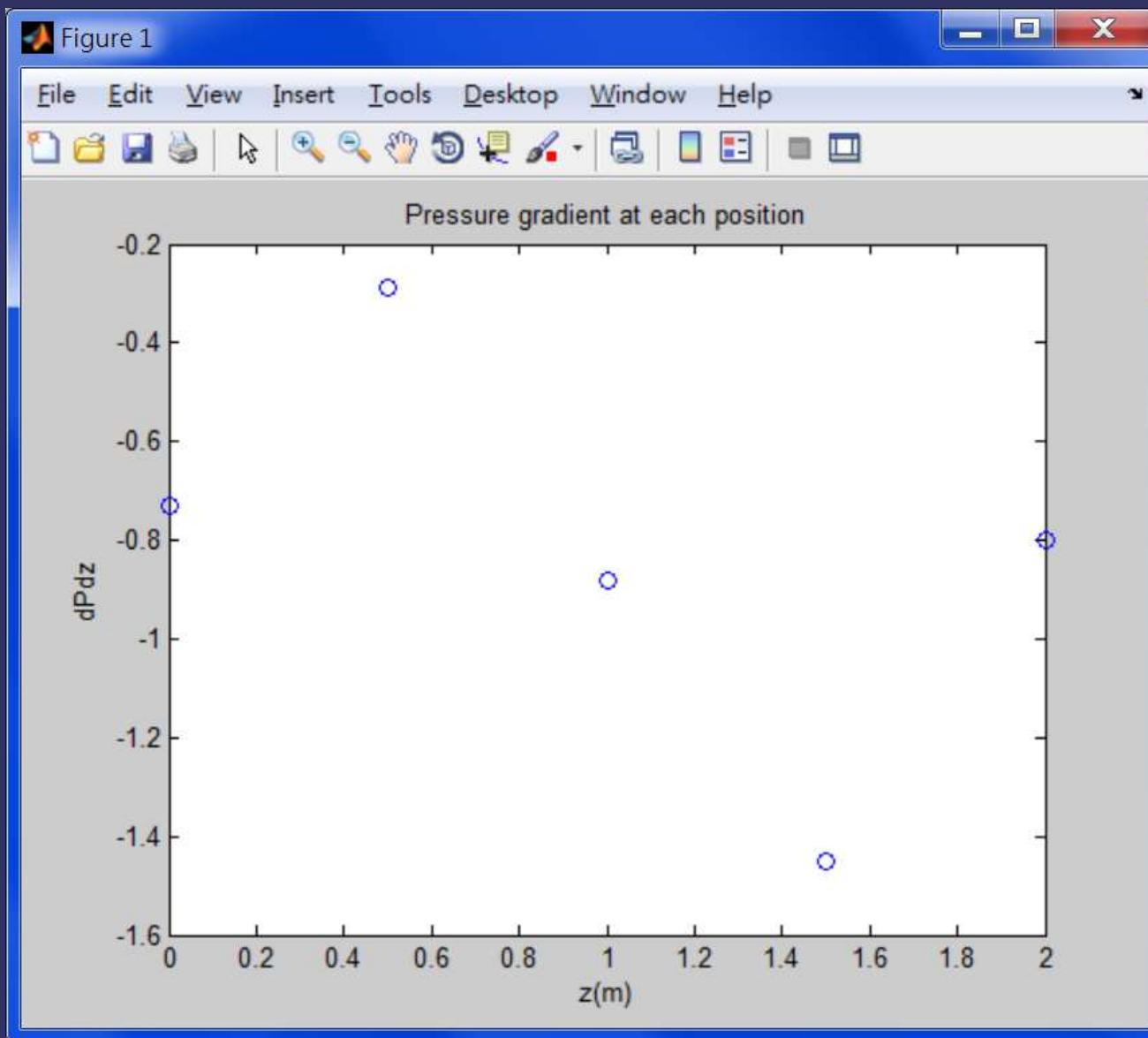
Position z (m)	0	0.5	1.0	1.5	2.0
Pressure P (kPa)	1.82	1.48	1.19	0.61	0.15

Estimate the values of pressure gradient dP/dz at various positions.

Example 3-2-1

Ans:

```
>> P=[1.82      1.48      1.19  0.61  0.15]'; % pressure  
>> h=0.5; % data interval  
>> dPdz=numder(P, h, 1, 0, 4); % central difference with an error of O(h^4)  
>> z=[0:length(P)-1]*h;  
>> plot(z, dPdz, 'o')  
>> xlabel('z(m)')  
>> ylabel('dPdz')  
>> title('Pressure gradient at each position')
```



3.2.4 The derivatives calculation of a known function

`df=numfder('fun',order,x,h,method,parameter)`

Input	Description
'fun'	The format of the function file is as follows: function f=fun(x, parameter) f= ... % the function to be differentiated
order	Order of derivative, up to four (the fourth order)
x	The point where the derivative is to be calculated
h	Step size of x used in the derivatives calculation (default is 10^{-3})
method	Method selection (three options): 1: forward difference -1: Backward difference 0: Central difference NOTE: Default method is the central difference method.
parameter	The parameter to be imported into fun.m.

Example 3-2-2

The higher-order derivatives of a known function

Determine the first to the fourth order derivatives of $f(x) = x^7$ at $x = 1$.

Ans:

Step 1: Create a function file named fun3_2_2.m,

```
function f=fun(x)
```

```
f=x.^7; % notice the usage of .^
```

Example 3-2-2

Ans:

Step 2:

```
>> h=0.005; % set h=0.005
>> x=1; % point of interested to get the derivatives
>> fl=numfder('fun3_2_2', 1, x, h, 0) % the first-order derivative
                                         % using the method of central difference
fl=
    7.0009
>> f2=numfder('fun3_2_2', 2, x, h, 0) % the second-order derivative
                                         % using the method of central difference
f2 =
    42.0018
```

Example 3-2-2

Ans:

```
>> f3=numfder('fun3_2_2', 3, x, h, 0) % the third-order derivative  
% using the method of central difference
```

```
f3 =  
210.0158
```

```
>> f4=numfder('fun3_2_2', 4, x, h, 0) % the fourth-order derivative  
% using the method of central difference
```

```
f4 =  
840.0000
```

Example 3-2-2

Ans:

Symbolic Math Toolbox.

```
>> syms x % define the symbolic variable x
>> f=x^7; % define the f(x) function
>> d4f=diff(f, x, 4); % derive the fourth-order derivative of the
function
>> subs(d4f, 1) % calculate the derivative at the point of interest
ans=
840
```



3.2.5 A method for calculating numerical gradients

$$\nabla F = \frac{\partial F}{\partial x} \vec{i} + \frac{\partial F}{\partial y} \vec{j} + \frac{\partial F}{\partial z} \vec{k} + \dots$$

- $[F_x, F_y, F_z, \dots] = \text{gradient}(F, h1, h2, h3, \dots)$

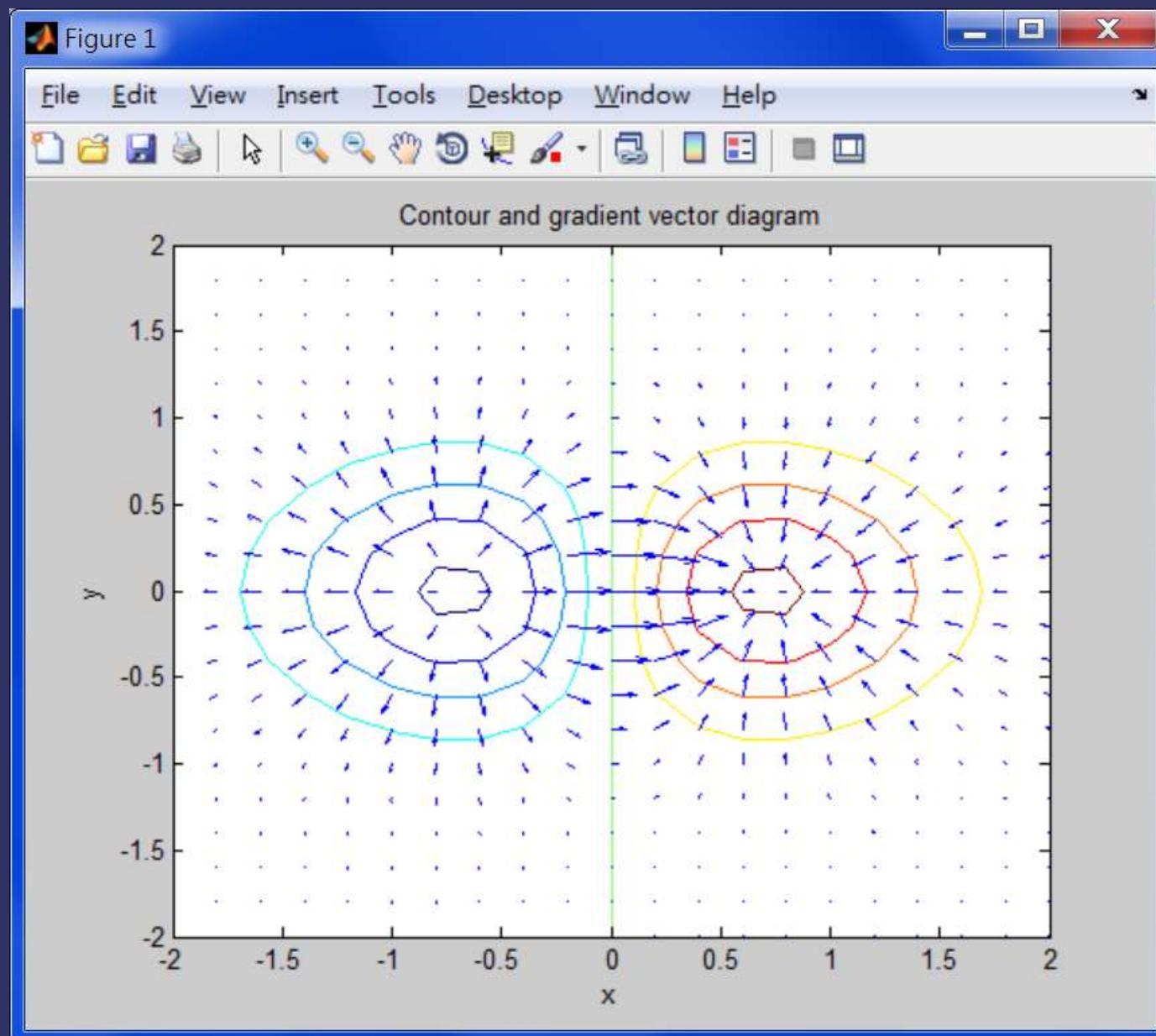
Example 3-2-3

Calculation of function gradients

Determine the gradients of the function $f(x, y) = xe^{-x^2 - 2y^2}$ for $-2 \leq x, y \leq 2$ and plot the gradient vector diagram.

Ans:

```
>> v=-2: 0.2: 2; % mesh points and let the increment be 0.2
>> [x, y]=meshgrid(v); % cross matching the points by meshgrid
>> z=x.*exp(-x.^2-2*y.^2); % calculate the relevant function values
>> [px, py]=gradient(z, 0.2, 0.2); % calculate gradients
>> contour (x, y, z) % plot the contour
>> hold on % turn on the function of figure overlapping
>> quiver(x, y, px, py) % plot the gradient vector diagram
>> hold off % turn off the function of figure overlapping
>> xlabel('x'); ylabel('y'); title('Contour and gradient vector diagram')
```



3.3 Numerical integration

$$I = \int_a^b f(x)dx$$

3.3.1 Integral of numerical data

`area =trapz(x, y)`

Example 3-3-1

Velocity estimation based on acceleration data of an object

Suppose an object is at rest initially, i.e., the initial velocity is zero, determine the velocity in 8 seconds based on the following acceleration data:

T (time, s)	0	1	2	3	4	5	6	7	8
a (acceleration, m/s ²)	0	2	5	8	12	18	24	32	45

Example 3-3-1

Ans:

$$\begin{aligned}v(8) &= \int_0^8 a(t)dt + v(0) \\&= \int_0^8 a(t)dt\end{aligned}$$

```
>> t=0:8; % time
>> a=[0    2    5    8    12   18   24   32   45]; %
acceleration data
>> v=trapz(t, a) % trapezoidal integration
v=
123.5000
```

3.3.2 Integral of a known function

Command	Numerical method used
Quad	Adaptive Simpson method
Quadl	Adaptive Gauss/Lobatto integration

```
area=quad('fun', a, b, tol, trace, parameter)
```

Input argument	Description
----------------	-------------

fun	The integrand function file provided by the user, which has the following format and contents: function f=fun(x, parameter) f= ... % the function to be integrated
a	Lower bound of the independent variable
b	Upper bound of the independent variable
tol	The specified integration accuracy (default 10^{-6})
trace	Display the integration process
parameter	The parameter to be imported into the function file

area= quadl('fun', a, b, tol, trace, parameter)

Example 3-3-2

Definite integral of a known function

Solve $I = \int_0^2 \frac{1}{x^3 - 3x^2 - 2x - 5} dx = ?$

Ans:

Step 1: Create the integrand function file, fun3_3_2.m,

```
function f=fun(x)
```

```
f= 1./(x.^3-3*x.^2-2*x-5); % notice the usage of ./
```

Example 3-3-2

Ans:

Step 2 :

```
>> areal = quad('fun3_3_2', 0, 2) % using the adaptive Simpson  
method
```

areal =

-0.2427

```
>> area2 = quadl('fun3_3_2', 0, 2) % using the adaptive  
Gauss/Lobatto method
```

area2 =

-0.2427

Example 3-3-2

Ans:

```
>> area3=quad('1./(x.^3-3*x.^2-2*x-5)', 0, 2)
```

```
area3 =
```

-0.2427

```
>> f=inline('1./(x.^3-3*x.^2-2*x-5)'); % define the integrand
```

```
>> area4=quad(f, 0, 2)
```

```
area4 =
```

-0.2427

3.3.3 Double integral

$$I = \int_{x\min}^{x\max} \int_{y\min}^{y\max} f(x, y) dy dx$$

```
area=dblquad('fun', xmin, xmax, ymin, ymax, tol, method, parameter)
```

Input argument	Description
	The function file of integrand is in the following format
fun	<pre>function f=fun(x, y, parameter) f= ... % function to be integrated</pre>
xmin	Lower bound of x
xmax	Upper bound of x
ymin	Lower bound of y
ymax	Upper bound of y
tol	Tolerance of integration accuracy (default 10^{-6}).
method	The method of quad or quadl that can be selected; default is quad.
parameter	The parameter value to be imported into the function file

Example 3-3-3

Double integral of a known function

Calculate the following double integral

$$I = \int_{\pi}^{2\pi} \int_0^{\pi} \{2y \sin(x) + 3x \cos(y)\} dy dx$$

Ans:

Step 1: Create the file named fun3_3_3.m

```
function f=fun(x, y)
f=2*y.*sin(x)+3*x.*cos(y);
```

Example 3-3-3

Ans:

Step 2 :

```
>> areal = dblquad('fun3_3_3', pi, 2*pi, 0, pi)
```

area =

-19.7392

```
>> f=inline('2*y.*sin(x)+3*x.*cos(y)');
```

```
>> area2=dblquad(f, pi, 2*pi, 0, pi)
```

area2 =

-19.7392

Example 3-3-3

Ans:

$$I = \int_{x\min}^{x\max} \int_{y\min}^{y\max} \int_{z\min}^{z\max} f(x, y, z) dz dy dx$$

area = triplequad('fun', xmin, xmax, ymin, ymax, zmin, zmax, tol, method)

3.4 Chemical engineering examples

Example 3-4-1

Preparation of a water solution with a required viscosity

To investigate the relationship between the concentration C (weight %; wt%) and the viscosity μ (c.p.) of a substance in a water solution, the following experimental data were obtained at the temperature of 20°C (*Leu, 1985*):

Concentration, C (wt%)	0	20	30	40	45	50
Viscosity, μ (c.p.)	1.005	1.775	2.480	3.655	4.620	5.925

Based on the above data, prepare a table to show the required concentration value when a desired viscosity is assigned. Note that the required value of the viscosity ranges from 1.1 to 5.9 with an interval of 0.3.

MATLAB program design:

— ex3_4_1.m —

```
%  
% Example 3-4-1 Viscosity interpolation of a water solution  
%  
clear  
clc  
%  
% given data  
%  
C=[0 20 30 40 45 50]; % concentration (wt%)  
u=[1.005 1.775 2.480 3.655 4.620 5.925]; % viscosity (c.p.)  
u1=1.1: 0.3: 5.9; % viscosity of interest  
C1=interp1(u, C, u1,'cubic'); % cubic spline interpolation  
%  
% results printing  
%  
disp(' Viscosity Conc.')  
disp([u1' C1'])
```

Execution results:

```
>> ex3_4_1
```

Viscosity	Conc.
-----------	-------

1.1000	2.9960
--------	--------

1.4000	11.5966
--------	---------

1.7000	18.5633
--------	---------

2.0000	23.7799
--------	---------

2.3000	27.9180
--------	---------

2.6000	31.2723
--------	---------

2.9000	34.1872
--------	---------

3.2000	36.7440
--------	---------

3.5000	38.9703
--------	---------

Viscosity	Conc.
-----------	-------

3.8000	40.8916
--------	---------

4.1000	42.5441
--------	---------

4.4000	44.0043
--------	---------

4.7000	45.3529
--------	---------

5.0000	46.6327
--------	---------

5.3000	47.8317
--------	---------

5.6000	48.9339
--------	---------

5.9000	49.9232
--------	---------

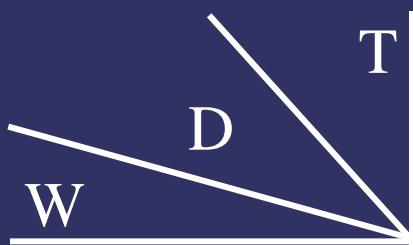
Example 3-4-2

Interpolation of diffusivity coefficients

T (°C)	MDEA weight fraction	Diffusivity coefficient, D×10 ⁵ (cm ² /s)
20	0.30	0.823
20	0.40	0.639
20	0.50	0.430
25	0.30	0.973
25	0.40	0.751
25	0.50	0.506
30	0.30	1.032
30	0.40	0.824
30	0.50	0.561

Based on the above data, estimate the diffusivity coefficient of nitrogen dioxide at the temperature of 27°C, 28°C, and 29°C when the MDEA weight fraction is 0.35 and 0.45, respectively.

Problem formulation and analysis:



	20	25	30
0.3	0.823	0.973	1.032
0.4	0.639	0.751	0.824
0.5	0.430	0.506	0.561

MATLAB program design:

— ex3_4_2.m —

```
%  
% Example 3-4-2 interpolation of diffusivity coefficients  
%  
clear  
clc  
%  
% experimental data  
%  
T=[20 25 30]; % temperatures (°C)  
W=[0.3 0.4 0.5]'; % weight fractions  
D=[0.823 0.973 1.032  
    0.639 0.751 0.824  
    0.430 0.506 0.561]; % diffusivity coefficients (cm^2/s)  
%  
Ti=[27 28 29]; % temperature of interest  
Wi=[0.35 0.45]; % weight fraction of interest
```

MATLAB program design:

— ex3_4_2.m —

```
%  
% calculation and results printing  
%  
for i=1:length(Ti)  
    for j=1:length(Wi)  
        Di=interp2(T,W,D,Ti,Wi(j)); % interpolation  
        fprintf('temp=%i °C, weight fraction=%.\n1f%%, diffusivity coeff.=...  
%.3fe-5 cm^2/s\n',Ti(i),100*Wi(j),Di(i))  
    end  
end
```

Execution results:

```
>> ex3_4_2
```

temp=27 °C, weight fraction=35.0%, diffusivity coeff.=0.888e-5 cm^2/s

temp=27 °C, weight fraction=45.0%, diffusivity coeff.=0.654e-5 cm^2/s

temp=28 °C, weight fraction=35.0%, diffusivity coeff.=0.902e-5 cm^2/s

temp=28 °C, weight fraction=45.0%, diffusivity coeff.=0.667e-5 cm^2/s

temp=29 °C, weight fraction=35.0%, diffusivity coeff.=0.915e-5 cm^2/s

temp=29 °C, weight fraction=45.0%, diffusivity coeff.=0.680e-5 cm^2/s

Example 3-4-3

Reaction rate equation of a batch reactor

Under the reaction temperature of 50°C, the following reaction was occurring in a batch reactor



with the initial concentrations of reactant A and B being both set as 0.1 g-mol/L. The experimental data were obtained, which are listed in the table below (*Leu*, 1985).

Time (sec)/500	0	1	2	3	4	5	6
Product concentration (g-mol/L)	0	0.008	0.014	0.020	0.025	0.029	0.033
Time (sec)/500	7	8	9	10	11	12	13
Product concentration (g-mol/L)	0.036	0.040	0.042	0.045	0.048	0.050	0.052

Based on the obtained experimental data, deduce the reaction rate equation for the reaction system.

Problem formulation and analysis:

$$r_R = \frac{dC_R}{dt} = k C_A^m C_B^n$$

Variable	Physical meaning
k	Reaction rate constant
C_A, C_B	Concentrations of A and B (g-mol/L)
C_R	Product concentration (g-mol/L)
t	Reaction time (s)
m, n	Reaction orders of A and B , respectively

Problem formulation and analysis:

$$C_A = C_B$$

$$C_A = C_{AO} - C_R$$

$$r_R = \frac{dC_R}{dt} = k C_A^N$$

$$\ln r_R = \ln k + N \ln C_A$$

$$= a + b \ln C_A$$

$$N = b$$

$$k = e^a$$

Problem formulation and analysis:

$$\ln r_{R1} = a + b \ln C_{A1}$$

$$\ln r_{R2} = a + b \ln C_{A2}$$

$$\vdots$$

$$\ln r_{RM} = a + b \ln C_{AM}$$

$$\underbrace{\begin{bmatrix} 1 & \ln C_{A1} \\ 1 & \ln C_{A2} \\ \vdots & \vdots \\ 1 & \ln C_{AM} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \ln r_{R1} \\ \ln r_{R2} \\ \vdots \\ \ln r_{RM} \end{bmatrix}}_{\mathbf{B}}$$

Problem formulation and analysis:

$$\mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{A} \setminus \mathbf{B}$$

re-estimate the reaction rate constant \bar{k} based on the obtained integer reaction order \bar{N} .

Step 1: Let \bar{N} be the nearest integer of N

$$r_R = \bar{k} C_A^{\bar{N}}$$

Problem formulation and analysis:

Step 2:

$$\begin{bmatrix} C_{A1}^N \\ C_{A2}^N \\ \vdots \\ C_{AM}^N \end{bmatrix} \bar{k} = \begin{bmatrix} r_{R1} \\ r_{R2} \\ \vdots \\ r_{RM} \end{bmatrix}$$

$\overbrace{\mathbf{A}}$
 $\overbrace{\mathbf{B}}$

Step 3:

$$\bar{k} = \bar{A} \mid \bar{B}$$

MATLAB program design:

1. Calculate r_R by using the method of difference approximation.
2. Form matrix **A** and vector **B** based on the data set and the calculated r_R .
3. Execute **A\ B** to obtain the total reaction order and the reaction rate constant value.
4. Take an integer to be the total reaction order and re-estimate the reaction rate constant value.

MATLAB program design:

— ex3_4_3.m —

```
%  
% Example 3-4-3 Reaction rate equation of a batch reactor  
%  
clear; close all;  
clc  
%  
% given data  
%  
Ca_0=0.1; % initial concentration of reactant (g-mol/L)  
t=[0:13]*500; % sampling time (sec)  
%  
% product R concentration Cr (g-mol/L)  
%  
Cr=[0 0.008 0.014 0.020 0.025 0.0295 0.0330 0.0365 0.0400 0.0425 0.0455...  
0.0480 0.0505 0.0525];  
%  
h=500; % sampling interval
```

MATLAB program design:

— ex3_4_3.m —

```
% use number to estimate the reaction rate
%
rR=numder(Cr,h,1,0,4);
%
% estimate the reaction rate and the reaction order
%
Ca=Ca_0-Cr; % concentration of the reactant
n=length(Cr); % number of the experimental data
A=[ones(n,1) log(Ca')]; % matrix A
B=log(rR); % coefficient vector B
x=A\B;
N=x(2); % reaction order
k=exp(x(1)); % reaction rate constant
%
% take an integer as the reaction order and re-estimate the reaction rate
%
N_bar=round(N); % integer reaction order
```

MATLAB program design:

— ex3_4_3.m —

```
A_bar=Ca'.^N_bar;
B_bar=rR;
k_bar=A_bar\B_bar;
%
% results printing
%
disp(' time prod. conc. react. conc. reaction rate')
for i=1:n
    fprintf('%7.3e %12.3e %12.3e %12.3e\n',t(i),Cr(i),Ca(i),rR(i))
end
%
fprintf('\n estimated reaction order=%5.3f, estimated reaction rate...
constant=%.3e', N, k)
fprintf('\n integer reaction order=%i, re-estimated reaction rate...
constant=%.3e\n ', N_bar, k_bar)
%
% compare the results by plotting
```

MATLAB program design:

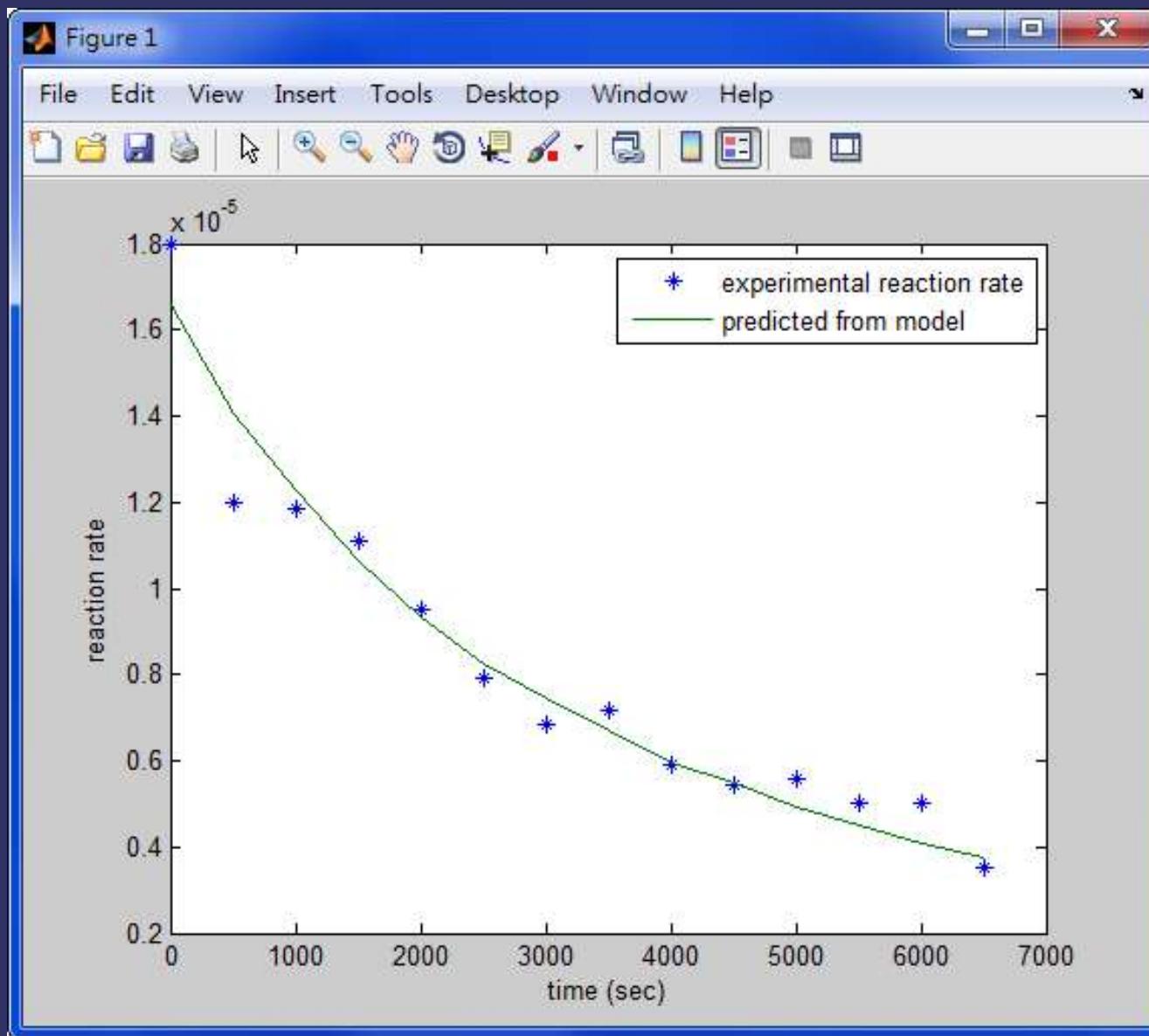
— ex3_4_3.m —

```
%  
plot(t,rR,'*',t,k_bar*Ca.^N_bar)  
xlabel('time (sec)')  
ylabel('reaction rate')  
legend('experimental reaction rate', 'predicted from model')
```

Execution results:

>> ex3_4_3

time	prod. conc.	react. conc.	reaction rate
0.000e+000	0.000e+000	1.000e-001	1.800e-005
5.000e+002	8.000e-003	9.200e-002	1.200e-005
1.000e+003	1.400e-002	8.600e-002	1.183e-005
1.500e+003	2.000e-002	8.000e-002	1.108e-005
2.000e+003	2.500e-002	7.500e-002	9.500e-006
2.500e+003	2.950e-002	7.050e-002	7.917e-006
3.000e+003	3.300e-002	6.700e-002	6.833e-006
3.500e+003	3.650e-002	6.350e-002	7.167e-006
4.000e+003	4.000e-002	6.000e-002	5.917e-006
4.500e+003	4.250e-002	5.750e-002	5.417e-006
5.000e+003	4.550e-002	5.450e-002	5.583e-006
5.500e+003	4.800e-002	5.200e-002	5.000e-006
6.000e+003	5.050e-002	4.950e-002	5.000e-006
6.500e+003	5.250e-002	4.750e-002	3.500e-006



Example 3-4-4

Volume fraction of solid particles in a gas-solid fluidized bed

In a gas-solid two-phase fluidized bed, the pressure measurements at different positions in the axial direction are shown in the following table (*Constantinides and Mostoufi, 1999*):

Axial position z (m)	0	0.5	1.0	1.5	2.0
Pressure value P (kPa)	1.82	1.48	1.19	0.61	0.15

It is assumed that the *stress* between solid particles and the *wall shear* against the bed walls are negligible, and the densities of the gas and the solid in the fluidized bed are 1.5 and 2,550 kg/m³, respectively. Based on the above information, calculate the average *volume fraction* of the solid particles.

Problem formulation and analysis:

$$-\frac{dP}{dz} = [\rho_g(1 - \varepsilon_s) + \rho_s \varepsilon_s]g$$

$$\varepsilon_s = \frac{-\left(\frac{dP}{dz}\right) - \rho_g g}{(\rho_s - \rho_g)g}$$

MATLAB program design:

1. Use the command “numder” to obtain dP/dz .
2. Calculate ϵ_s by using (3.4-6).

ex3_4_4.m

```
%  
% Example 3-4-4 Volume fraction of solid particles in a gas-solid fluidized bed  
%  
clear; close all;  
clc  
%  
% given data  
%  
z=0:0.5:2.0; % axial positions (m)  
P=[1.82 1.48 1.19 0.61 0.15]*1.e3; % pressure (Pa)
```

MATLAB program design:

— ex3_4_4.m —

```
%  
h=0.5; % distance  
rho_g=1.5; % gas phase density (kg/m^3)  
rho_s=2550; % solid phase density (kg/m^3)  
g=9.81; % gravitational acceleration (m/s^2)  
%  
% calculating dP/dz  
%  
dPdz=numder(P, h);  
%  
% calculating the volume fraction  
%  
epson_s=(-dPdz-rho_g*g)/(g*(rho_s-rho_g));  
%  
% results plotting  
%
```

MATLAB program design:

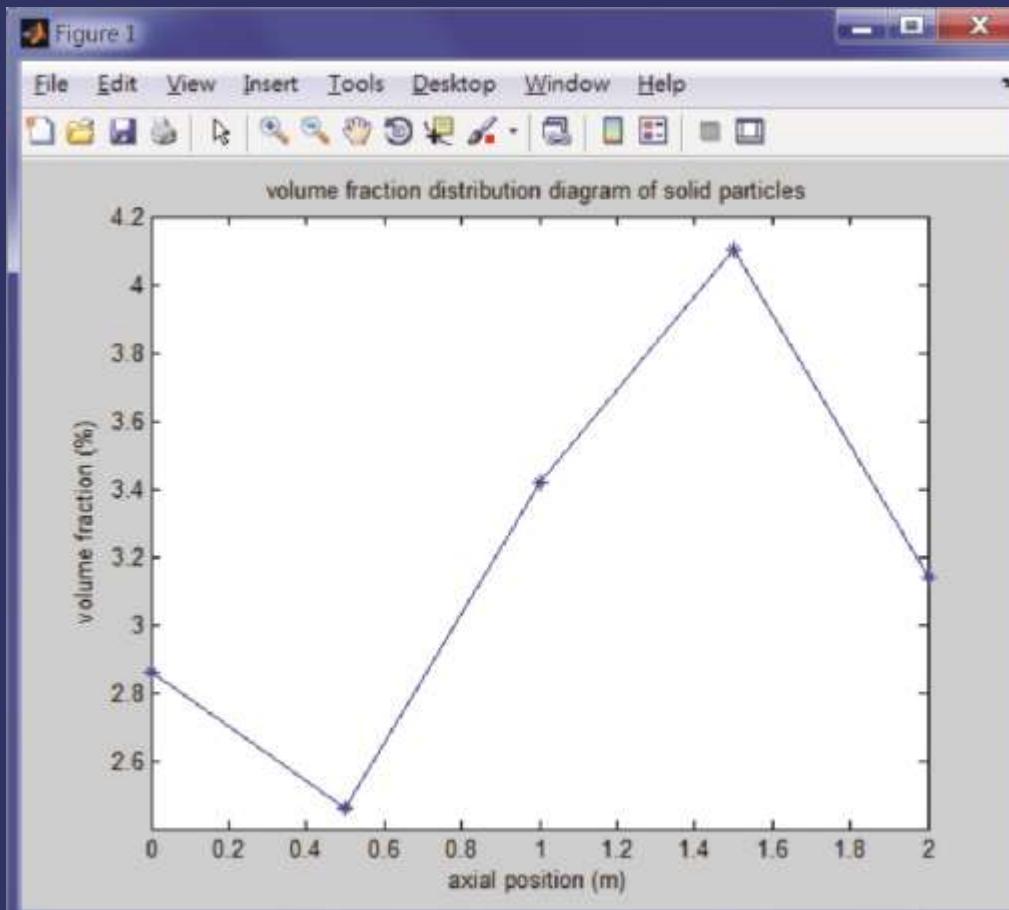
— ex3_4_4.m —

```
plot(z, 100*epson_s, '-*')
title('volume fraction distribution diagram of solid particles')
xlabel('axial position (m)')
ylabel('volume fraction (%)')
%
fprintf('\n average volume fraction=%7.3f %%\n', 100*mean(epson_s))
```

Execution results:

>> ex3_4_4

average volume fraction= 3.197%



Example 3-4-5

Average residence time calculation based on a tracer response

A photoelectric detector is used to measure the concentration of the dye agent flowing out from a measuring device. The following table lists a set of experimental data measured by the detector (*Leu, 1985*):

Time (sec)	0 ¹	1	2	3	4	5	6	7	8
Signal ² (mv)	0	27.5	41.0	46.5	47.0	44.5	42.5	38.5	35.0
Time (sec)	9	10	14	18	22	26	30	34	38
Signal (mv)	31.5	28.0	17.5	10.5	6.5	4.0	2.5	1.5	1.0
Time (sec)	42	46	50	54	58	62	66	70	74
Signal (mv)	0.5	0.3	0.2	0.1	0.1	0.1	0.05	0	0

- 1 The moment the dye agent was injected into the device.
- 2 The size of the signal is proportional to the concentration of the dye agent.

Calculate the average residence time T_m of the dye agent in the device and the secondary T_s^2 torque when the response curve reaches T_m .

Problem formulation and analysis:

$$g(\theta) = \frac{y(\theta)}{a_0}$$

$$a_0 = \int_0^{\infty} y(\theta) d\theta$$

$$T_m = \int_0^{\infty} \theta g(\theta) d\theta = \frac{1}{a_0} \int_0^{\infty} \theta y(\theta) d\theta$$

$$a_1 = \int_0^{\infty} \theta y(\theta) d\theta$$

Problem formulation and analysis:

$$T_m = \frac{a_1}{a_0}$$

$$\begin{aligned} T_s^2 &= \int_0^\infty (\theta - T_m)^2 g(\theta) d\theta \\ &= \int_0^\infty \theta^2 g(\theta) d\theta - 2T_m \int_0^\infty \theta g(\theta) d\theta + T_m^2 \int_0^\infty g(\theta) d\theta \end{aligned}$$

$$\int_0^\infty g(\theta) d\theta = 1$$

$$T_s^2 = \frac{1}{a_0} \int_0^\infty \theta^2 y(\theta) d\theta - T_m^2$$

Problem formulation and analysis:

$$a_2 = \int_0^{\infty} \theta^2 y(\theta) d\theta$$

$$T_s^2 = \frac{a_2}{a_0} - T_m^2$$

MATLAB program design:

— ex3_4_5.m —

```
%  
% Example 3-4-5 Average residence time calculation based on a tracer response  
%  
clear  
clc  
%  
% given data  
%  
t=[0:10 14:4:74]; % time  
y=[0 27.5 41.0 46.5 47.0 44.5 42.5 38.5 35.0 31.5 28.0 17.5...  
    10.5 6.5 4.0 2.5 1.5 1.0 0.5 0.3 0.2 0.1 0.1 0.1...  
0.05 0 0]; % signal (mv)  
%  
a0=trapz(t, y);  
a1=trapz(t, t.*y);  
a2=trapz(t, t.^2.*y);
```

MATLAB program design:

— ex3_4_5.m —

```
%  
Tm=a1/a0; % average residence time  
Ts2=a2/a0-Tm^2; % the secondary torque  
%  
% results printing  
%  
fprintf('\n average residence time=%7.3f (sec)', Tm)  
fprintf('\n secondary torque=%7.3f\n', Ts2)
```

Execution results:

>> ex3_4_5

average residence time= 10.092 (sec)

secondary torque= 69.034

Example 3-4-6

Design of an absorption tower

In an absorption tower operated at the temperature of 20°C and under the pressure of 1 atm, the polluted air containing 5 mol% SO₂ is being washed with water to reduce its concentration to 0.5 mol% SO₂. Determine the overall number of transfer units N_{0G} on the basis of the gas phase molar concentration when the liquid/gas ratio is 45. In addition, determine the height of the absorption tower when the *overall height of a transfer unit* (H.T.U) H_{0G} is 0.75 m.

NOTE: A set of equilibrium data of the solubility of SO₂ in water is listed as follows (*Leu, 1985*):

$y^* \times 10^3$	51.3	34.2	18.6	11.2	7.63	4.21	1.58	0.658
$x \times 10^3$	1.96	1.40	0.843	0.562	0.422	0.281	0.141	0.056

In the above table, x and y^* , respectively, indicate the mole fractions of SO_2 in the liquid phase and its equilibrium mole fraction in the gas phase.



Problem formulation and analysis:

overall number of transfer units

$$N_{0G} = \int_{y_2}^{y_1} \frac{1}{y - y^*} dy$$

$$G_M \left(\frac{y}{1-y} - \frac{y_2}{1-y_2} \right) = L_M \left(\frac{x}{1-x} - \frac{x_2}{1-x_2} \right)$$

$$H = N_{0G} H_{0G}$$

MATLAB program design:

The calculation steps are summarized as follows:

1. Fit the solubility data of x and y^* into a polynomial of third order.
2. The results of the polynomial are imported into the integrand file through the use of the global command.
3. Use quadl to integrate (3.4-14) for the value of N0G.
4. Calculate the height of the absorption tower by using (3.4-16).

$$x = \frac{\frac{G_M}{L_M} \left(\frac{y}{1-y} - \frac{y_2}{1-y_2} \right) + \frac{x_2}{1-x_2}}{\frac{G_M}{L_M} \left(\frac{y}{1-y} - \frac{y_2}{1-y_2} \right) + \frac{x_2}{1-x_2} + 1}$$

MATLAB program design:

— ex3_4_6.m —

```
function ex3_4_6
%
% Example 3-4-6 design of an absorption tower
%
clear
clc
%
global P GL y2 x2
%
% given data
%
% the equilibrium data
%
x=[1.96 1.40 0.843 0.562 0.422 0.281 0.141 0.056]*1.e-3;
yeg=[51.3 34.2 18.6 11.2 7.63 4.21 1.58 0.658]*1.e-3;
%
```

MATLAB program design:

— ex3_4_6.m —

```
y2=0.005; % desired mole fraction of SO2
```

```
y1=0.05; % mole fraction of SO2 in the pollutant entering the tower
```

```
HOG=0.75; % H.T.U.
```

```
x2=0; % mole fraction of SO2 in the liquid phase at the top
```

```
GL=45; % LM/GM
```

```
%
```

```
% fitting the equilibrium data into a polynomial of 3rd order
```

```
%
```

```
P=polyfit(x, yeg, 3);
```

```
%
```

```
% calculate NOG and H
```

```
%
```

```
NOG=quadl(@ex3_4_6b, y2, y1);
```

```
H=NOG*HOG;
```

```
%
```

```
% results printing
```

MATLAB program design:

— ex3_4_6.m —

```
%  
fprintf('\n The overall number of transfer units NOG=%7.3f,...  
height of the absorption tower=%.3f m\n', NOG, H)  
%  
% subroutine  
%  
function f=ex3_4_6b(y)  
global P GL y2 x2  
R=(y./(1-y)-y2./(1-y2))/GL+x2./(1-x2);  
x=R./(1+R); % calculating x  
yeg=polyval(P, x); % calculating y*  
f=1./(y-yeg);
```

Execution results:

```
>> ex3_4_6
```

The overall number of transfer units NOG=3.105, height of the absorption tower=2.329 m

Example 3-4-7

Reaction time of an adiabatic batch reactor

In an adiabatic batch reactor, the following liquid phase reaction occurs

$$A \rightarrow B, \quad -r_A = k_0 \exp\left(-\frac{E}{RT}\right) C_A^n$$

The significance of each variable in the above rate equation is listed in the table below:

Variable	Physical meaning
r_A	Reaction rate (g-mol/L·hr)
k_0	Reaction rate constant, $\frac{\text{L}}{\text{hr}} \left(\frac{\text{g} \cdot \text{mol}}{\text{L}} \right)^{1-n}$
E	Activation energy (cal/g-mol)
R	Ideal gas constant (1.987 cal /g-mol·K)
T	Reaction temperature (K)
C_A	Concentration of reactant A (g-mol/L)
n	Reaction order

The reaction conditions and the relevant physical properties are as follows (*Leu*, 1985): the initial reaction temperature $T_0 = 298$ K, initial concentration $C_{A0} = 0.1$ g-mol/L, heat capacity $C_p = 0.95$ cal/g-mol, density of the solution $\rho = 1.1$ g/cm³, heat of reaction $-\Delta H_r = 35,000$ cal/g-mol, activation energy $E = 15,300$ cal/g-mol, reaction order $n = 1.5$ and the reaction rate constant $k_0 = 1.5 \times 10^{11} \frac{\text{L}}{\text{hr}} \left(\frac{\text{g-mol}}{\text{L}} \right)^{1-n}$. Based on the above, determine the required reaction time when the conversion rate of reactant A reaches 85%.

Problem formulation and analysis:

$$\frac{dC_A}{dt} = r_A = -k_0 \exp\left(-\frac{E}{RT}\right) C_A^n$$

$$x_A = \frac{C_{A0} - C_A}{C_{A0}} = 1 - \frac{C_A}{C_{A0}}$$

$$\frac{dx_A}{dt} = -\frac{1}{C_{A0}} \frac{dC_A}{dt} = \frac{1}{C_{A0}} k_0 \exp\left(-\frac{E}{RT}\right) [C_{A0}(1-x_A)]^n$$

Problem formulation and analysis:

$$dt = - \frac{dx_A}{k_0 C_{A0}^{n-1} \exp\left(-\frac{E}{RT}\right) (1-x_A)^n}$$

$$t = - \frac{1}{k_0 C_{A0}^{n-1}} \int_0^{x_{Af}} \frac{dx_A}{\exp\left(-\frac{E}{RT}\right) (1-x_A)^n}$$

$$T = T_0 + \frac{(-\Delta H_r) C_{A0}}{\rho C_p} x_A$$

MATLAB program design:

— ex3_4_7.m —

```
function ex3_4_7
%
% Example 3-4-7 design of an adiabatic batch reactor
%
clear
clc
%
global T0 Hr CA0 rho Cp n E R
%
% given data
%
CA0=0.1; % initial concentration of the reactant (g-mol/L)
k0=1.5e11; % reaction rate constant
E=15300; % activation energy (cal/g-mol)
R=1.987; % ideal gas constant (cal/g-mol.K)
n=1.5; % reaction order
xf=0.85; % the desired reaction rate
```

MATLAB program design:

— ex3_4_7.m —

```
T0=298; % reaction temperature (K)
```

```
Hr=-35000; % reaction heat (cal/g-mol)
```

```
Cp=0.95; % specific heat capacity (cal/g.K)
```

```
rho=1.1e3; % density of the reactant (g/L)
```

```
%
```

```
% calculating the reaction time
```

```
%
```

```
t=quadl(@ex3_4_7b,0,xf)/(CA0^(n-1)*k0);
```

```
%
```

```
% results printing
```

```
%
```

```
fprintf('\n reaction time=% .3f hr\n', t)
```

```
%
```

```
% subroutine
```

```
%
```

MATLAB program design:

— ex3_4_7.m —

```
function y=ex3_4_7b(x)
global T0 Hr CA0 rho Cp n E R
T=T0+(-Hr)*CA0*x./(rho*Cp);
y=1./(exp(-E./(R*T)).*(1-x).^n);
```

Execution results:

```
>> ex3_4_7
```

reaction time=9.335 hr

Example 3-4-8

Breakthrough time determination for an absorption tower

An absorption tower packed with activated carbon in its fixed-layer is used to treat a water solution containing 2 ppm ABS. The equilibrium ABS absorption can be expressed by the Freundlich equation as follows:

$$q^* = \alpha C^\beta$$

where q^* (mg-ABS/mg-activated carbon) is the equilibrium absorption amount and C denotes the ABS concentration (ppm) in the water solution. Note that the equation constants for the ABS absorption system are estimated to be $\alpha = 2$ and $\beta = 1/3$. Besides, the operational conditions and physical properties of the ABS absorption system are listed in the table below (*Leu, 1985*).

Physical parameter	Value	Physical parameter	Value
Surface velocity, u	10 m/hr ⁻¹	The average particle diameter of the activated carbon, d_p	0.95 mm
Depth of the fixed layer, Z	0.5 m	Density of the solution, ρ	1,000 kg/m ³
Void fraction, ε_b	0.5	The viscosity of the solution, μ	3.6 pa·s
Density of the activated carbon, ρ_s	700 kg/m ³	The diffusivity coefficient of the ABS, D_{AB}	1.35×10^{-6} cm ² /s

In addition, the intragranular diffusivity coefficient in the activated carbon particles D_{iq} is 2.15×10^{-10} cm²/s. Determine the *breakthrough time* when the ABS concentration at the outlet of the absorption tower reaches 0.2 ppm.

Problem formulation and analysis:

breakthrough time

$$t_b = \frac{q_0 \rho_b}{C_0 u} \left(Z - \frac{Z_a}{2} \right)$$

layer density ρ_b

$$\rho_b = \rho_s (1 - \varepsilon_b)$$

length of the absorption layer Z_a

$$Z_a = \frac{u}{K_f a_v} \int_{C_B}^{C_0 - C_B} \frac{dC}{C - C^*}$$

Problem formulation and analysis:

$$q = q_0 C/C_0$$

$$C^* = \left(\frac{q_0 C}{\alpha C_0} \right)^{1/\beta}$$

overall mass transfer coefficient K_f

$$\frac{1}{K_f} = \frac{1}{k_f} + \frac{1}{H k_s}$$

Henry constant

$$H \cong \alpha \beta \left(\frac{C_0}{2} \right)^{\beta-1}$$

Problem formulation and analysis:

film coefficient, k_f ,

$$\frac{k_f}{u/\varepsilon_b} \left(\frac{\mu}{\rho D_{AB}} \right)^{2/3} = 1.15 \left(\frac{d_p u \rho}{\mu \varepsilon_b} \right)^{-0.5}$$

surface area

$$a_v = \frac{6(1 - \varepsilon_b)}{d_p}$$

solid film coefficient k_s ,

$$k_s = \frac{60D_{iq}\rho_b}{d_p^2 a_v}$$

MATLAB program design:

1. Calculate q^* using (3.4-25).
2. Calculate ρ_b using (3.4-27).
3. Calculate a_v using (3.4-33).
4. Calculate H using (3.4-31).
5. Calculate k_f and k_s using (3.4-32) and (3.4-34), respectively.
6. Obtain Z_a by integrating (3.4-28).
7. Calculate the breakthrough time t_b using (3.4-26).

MATLAB program design:

— ex3_4_7.m —

```
function ex3_4_8
%
% Example 3-4-8 breakthrough time of an absorption tower
%
clear
clc
%
global q0 alpha c0 beta
%
% given data
%
Dab=1.35e-6; % molecular diffusivity coefficient (cm^2/s)
Diq=2.15e-10; % particle diffusivity coefficient (cm^2/s))
alpha=2; % constant
beta=1/3; % constant
c0=2e-3; % concentration at the inlet (kg/m^3)
cB=2e-4; %: desired concentration at the outlet (kg/m^3)
```

MATLAB program design:

— ex3_4_7.m —

```
Z=0.5; % depth of the fixed layer (m)
```

```
dp=9.5e-4; % average granular diameter (m)
```

```
rho_s=700; % surface density of activated carbon kg/m^3)
```

```
eb=0.5; % void fraction
```

```
mu=3.6; % viscosity (pa.s)
```

```
rho_L=1000; % liquid density (kg/m^3)
```

```
u=10; % surface velocity (m/hr)
```

```
%
```

```
% calculating those relevant constant values
```

```
%
```

```
q0= alpha *c0^beta; % balance absorption quantity at the inlet
```

```
rho_b=rho_s*(1-eb); % surface layer density
```

```
av=6*(1-eb)/dp; % surface area
```

```
H=alpha*beta*(c0/2)^(beta-1); % Henry constant
```

```
ks=60*Diq*rho_b/dp^2/av; % solid film coefficient
```

```
kf=u/eb*(mu/(rho_L*Dab))^-2/3*1.15*((dp*u*rho_L)/(mu*eb))^-0.5;
```

```
Kf=1/(1/kf+1/(H*ks)); % overall mass transfer coefficient
```

MATLAB program design:

— ex3_4_7.m —

```
%  
% calculate the Za value  
%  
area=quadl(@ex3_4_8b, cB, c0-cB); % integration  
Za=u*area/Kf/av; % length of the absorption layer  
tb=q0*rho_b*(Z-Za/2)/(c0*u); % breakthrough time  
%  
% results printing  
%  
disp(sprintf('\n the breakthrough time=%7.3f hr', tb))  
%  
% subroutine  
%  
function y=ex3_4_8b(c)
```

MATLAB program design:

— ex3_4_7.m —

```
global q0 alpha c0 beta
%
c_star=(q0*c/( alpha *c0)).^(1/beta);
y=1./(c-c_star);
```

Execution results:

```
>> ex3_4_8
```

the breakthrough time = 1599.687 hr

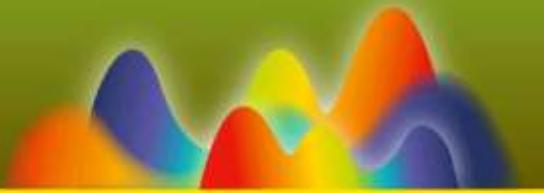
3.6 Summary of the MATLAB commands related to this chapter

Function	Command	Description
Interpolation	interp1	One-dimensional interpolation
	interp2	Two-dimensional interpolation
	interp3	Three-dimensional interpolation
	interpn	The n-dimensional interpolation
Differentiation	diff	Difference and differentiation
	numder	Numerical difference approximation
	numfder	Numerical derivative of a known function
	gradient	Gradient calculation
Integration	trapz	Trapezoidal integration
	quad	Function integration using the adaptive Simpson method
	quadl	Function integration using Gauss/Lobatto method
	dblquad	Double integral of a known function



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 4

Numerical Solution of Ordinary Differential Equations

版權所有・請勿翻製

- ① *initial value problems* (IVPs)
- ② *boundary value problems* (BVPs)
- ③ numerical solution of *ordinary differential equations* (ODEs)
- ④ *differential-algebraic equations* (DAEs)

the relevant Simulink tools and the *differential equation editor* (DEE) simulation interface will be presented in this chapter.

4.1 Initial value problems for ordinary differential equations

► 4.1.1 The standard problem

$$\frac{dx_1}{dt} = f_1(x_1, x_2, \dots, x_n, u, t), \quad x_1(0) = x_{10}$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, \dots, x_n, u, t), \quad x_2(0) = x_{20}$$

$$\vdots$$

$$\frac{dx_n}{dt} = f_n(x_1, x_2, \dots, x_n, u, t), \quad x_n(0) = x_{n0}$$

continuous stirred-tank reactor

$$\frac{dx_1}{dt} = -x_1 + D_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right), \quad x_1(0) = x_{10}$$

$$\frac{dx_2}{dt} = -(1+\beta)x_2 + BD_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right) + \beta u, \quad x_2(0) = x_{20}$$

- x_1 and x_2 represent the *dimensionless reactant concentration* and reactor temperature, respectively, while the forcing function (input variable) u is the cooling water temperature,
- u is the cooling water temperature,
- The CSTR model parameters D_a , ϕ , B , and β represent the *Damköhler number*, *activation energy*, *heat of reaction*, and the *heat transfer coefficient*, respectively.

4.1.2 The MATLAB ODE solvers

Problem types	Commands	Algorithms (Shampine and Reichelt, 1997)
Non-stiff ODE	ode45 ode23 ode113	Explicit Runge-Kutta (4, 5) pair of Dormand-Prince Explicit Runge-Kutta (2, 3) pair of Bogacki and Shampine Variable order Adam-Basforth-Moulton PECE solver
Stiff ODE	ode15s ode23s ode23t ode23tb	Numerical differentiation formulas of order 1-5 Modified Rosenbrock (2, 3) pair with a free interpolant Trapezoidal rule with a free interpolant Implicit Runge-Kutta formula with trapezoidal rule and a backward differentiation formula of order 2.

[t, x]=ode45('odefun', tspan, xo)

Output	Description
t	The position vector of the independent variable
x	The system state output, which is a matrix of n columns. Its row number equals to the length of independent variable t and the i-th column contains the evolutionary data of the i-th state with respect to the independent variable t .

Input	Description
odefun	The file name of the ordinary differential equation, which is in the format of <pre>function xdot=fun(t, x) xdot= [...]; % list the ordinary differential equations one by one</pre>
tspan	The position of the independent variable t . If tspan is set to [to tf], indicating that only the beginning and the end of the independent variable t are assigned, the middle points in between are automatically determined by the program. Furthermore, if users intend to acquire the state values at some specific values of the independent variable, tspan can be specified as $[t_0 \ t_1 \ \dots \ t_m]$, where t_i is the desired inner location of the independent variable satisfying the relation of $t_0 < t_1 < \dots < t_m (= t_f)$.
x0	The vector of initial state values.

Example 4-1-1

Dynamic simulation of a CSTR

$D_a = 0.072$, $\phi = 20$, $B = 8$, and $\beta = 0.3$

$u = 0$,

$x_1(0) = 0.1$ and $x_2(0) = 1$.

Step 1: Create an ODE file named ex4_1fun.

ex4_1_1fun.m

```
function xdot=fun(t, x)
% subroutine: ODE model of the CSTR
global Da phi B beta % declared as global variables
xdot=[-x(1)+Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))
      -(1+beta)*x(2)+B*Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))]; % u=0
```

Step 2: Create the main program to solve the IVP ODE

ex4_1_1.m

```
%  
% Example 4-1-1 Main program for solving CSTR dynamics  
%  
clear; close all; clc;  
%  
global Da phi B beta % declared as the global variables  
%  
Da=0.072; % Damköhler number  
phi=20; % activation energy  
B=8; % heat of reaction  
beta=0.3; % heat transfer coefficient  
x0=[0.1 1]'; % initial value of the system state  
%  
% Solving with ode45  
%  
[t, x]=ode45('ex4_1_1fun', [0 20], x0);  
%
```

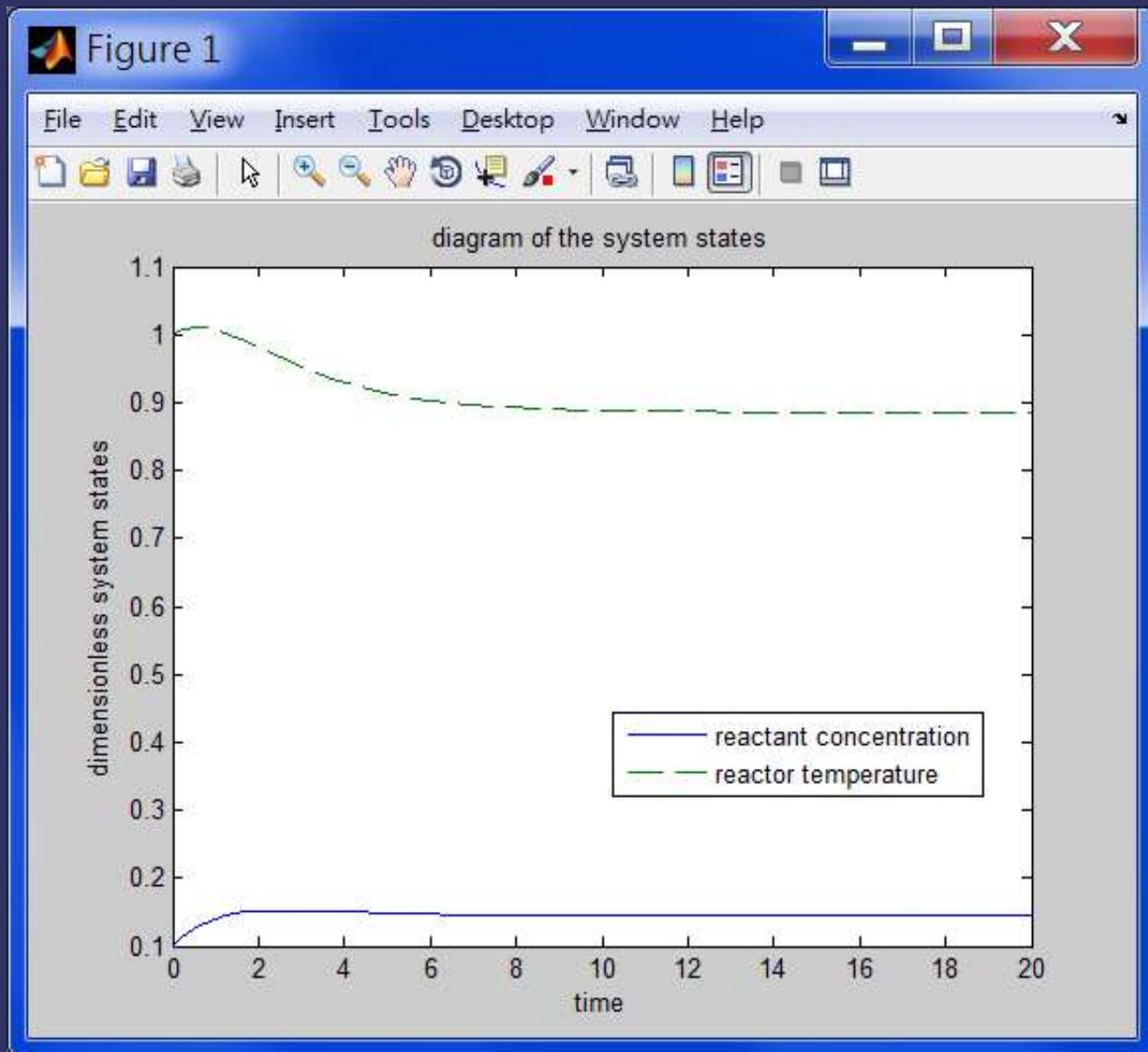
Step 2: Create the main program to solve the IVP ODE

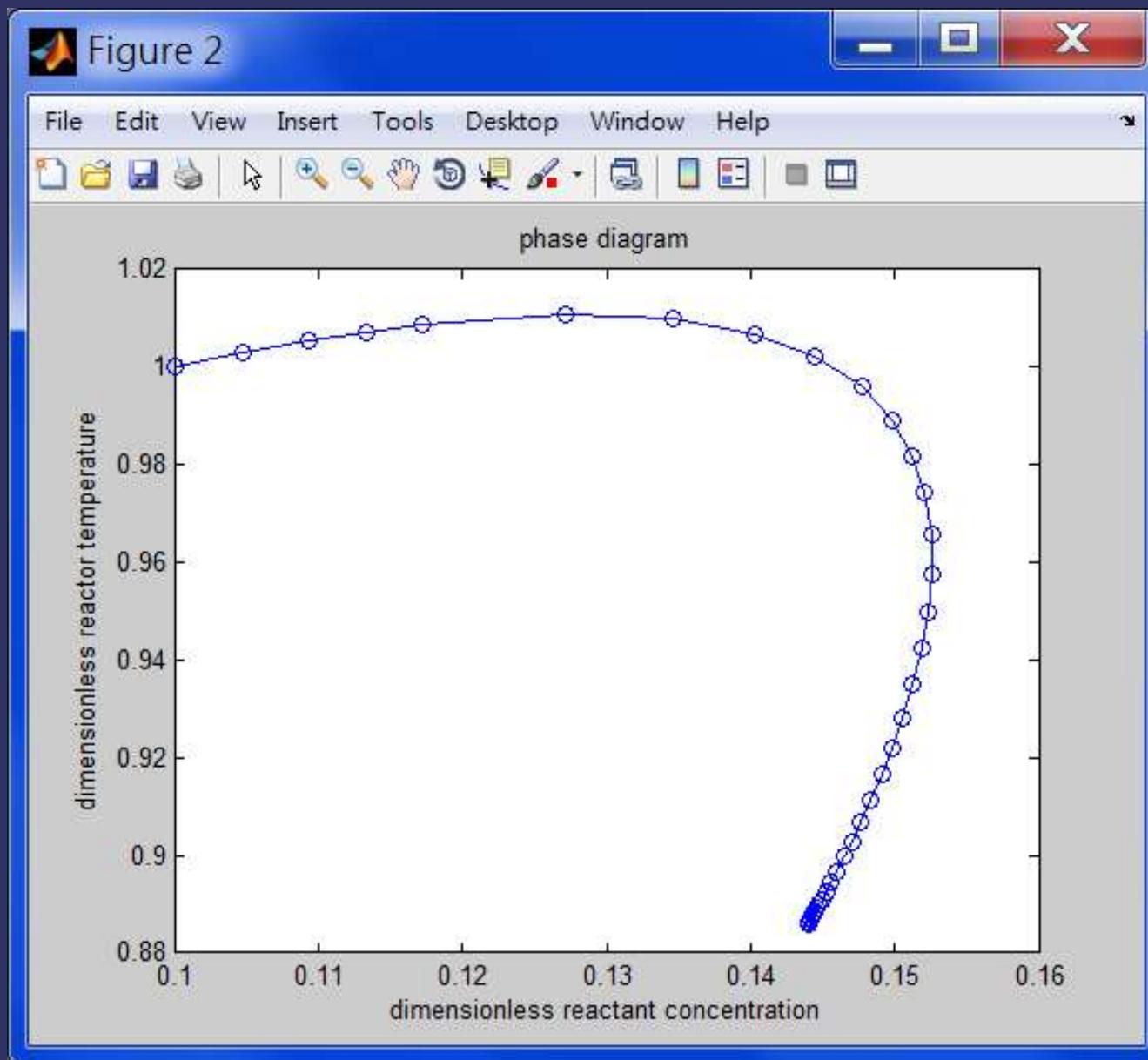
ex4_1_1.m

```
% Results plotting
%
figure(1) % diagram of the system states
plot(t,x(:,1),t,x(:,2),'--')
%
title('diagram of the system states')
xlabel('time')
ylabel('dimensionless system states')
legend('reactant concentration', 'reactor temperature')
%
figure(2) % phase diagram
plot(x(:,1), x(:,2), '-o')
xlabel('dimensionless reactant concentration')
ylabel('dimensionless reactor temperature')
title('phase diagram')
```

Step 3:

```
>> ex4_1_1
```





ex4_1_1b.m

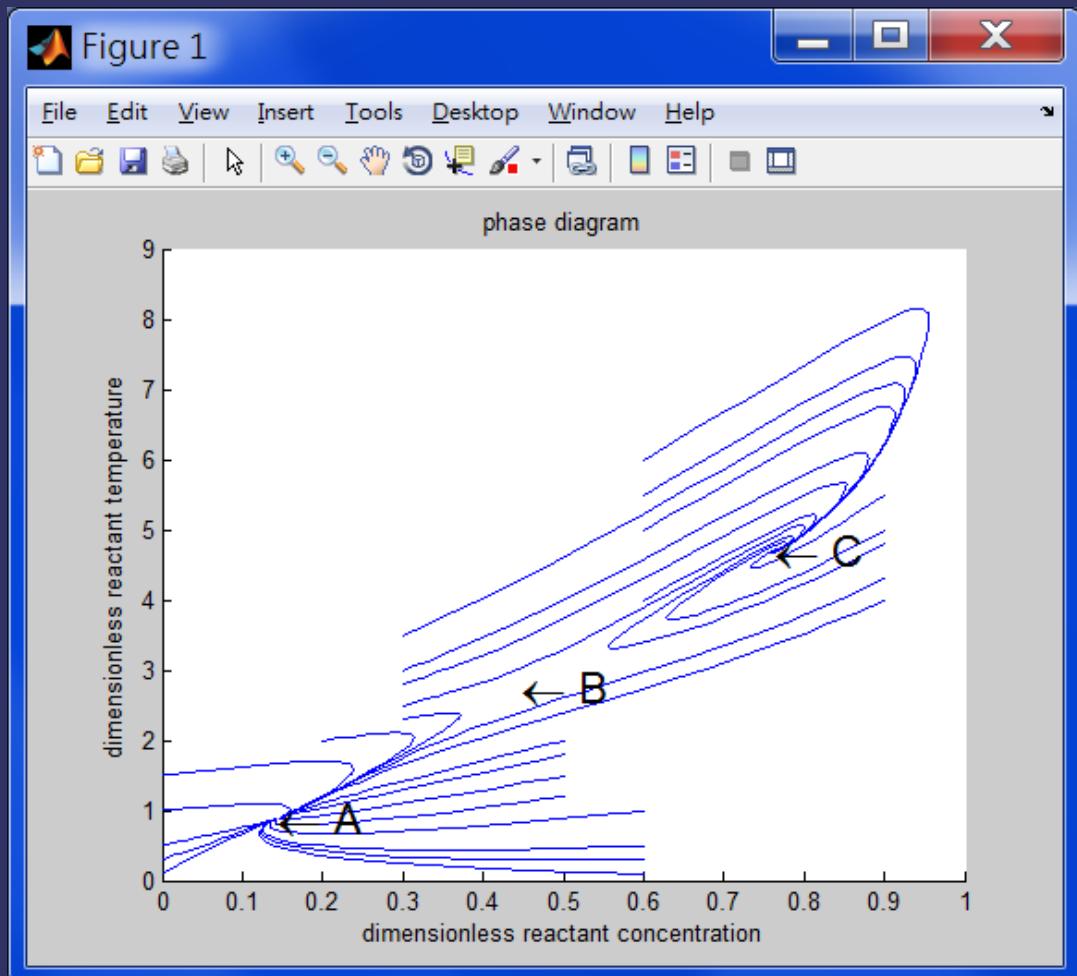
```
function ex4_1_1b
%
% Example 4-1-1 CSTR dynamics - the effect of the initial values
%
clear; close all;
clc
%
global Da phi B beta % declared as global variables
%
Da=0.072; % Damköhler number
phi=20; % activation energy
B=8; % heat of reaction
beta=0.3; % heat transfer coefficient
%
% different initial values
%
x1=[0 0 0 0 0 0.6 0.6 0.6 0.6 0.5 0.5 0.5 0.5 0.5 0.2 0.3 0.3 0.3 ...
0.3 0.3 0.9 0.9 0.9 0.9 0.9 0.6 0.6 0.6 0.6];
```

ex4_1_1b.m

```
x2=[0.1 0.3 0.5 1 1.5 0.1 0.3 0.5 1 1.5 1.2 1.8 2 2 2.3 2.5 2.8 ...
3 3.5 4 4.3 4.8 5 5.5 4 5 5.5 6];
N=length(x1);
%
figure(1) % phase diagram
hold on
%
for i=1:N
    x0=[x1(i) x2(i)]'; % initial value
    %
    % ODE solutions
    %
    [t, x]=ode45(@fun, [0 20], x0);
    %
    % phase diagram plotting
    %
    plot(x(:,1), x(:,2))
end
hold off
```

ex4_1_1b.m

```
xlabel('dimensionless reactant concentration')
ylabel('dimensionless reactant temperature')
title('phase diagram')
text(0.144,0.886, '\leftarrow A', 'FontSize', 18)
text(0.4472,2.7517, '\leftarrow B', 'FontSize', 18)
text(0.7646,4.7050, '\leftarrow C', 'FontSize', 18)
%
% ODE model of the CSTR
%
function xdot=fun(t, x)
global Da phi B beta % declared as global variables
xdot=[-x(1)+Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))
      -(1+beta)*x(2)+B*Da*(1-x(1))*exp(x(2)/(1+x(2)/phi))];
```



- The whole phase diagram shows that the CSTR has three equilibrium points of which A (0.1440, 0.8860), and C (0.7646, 4.7050) are *stable* equilibrium points and B (0.4472, 2.7517) is the *unstable* one.

4.1.3 Solving ODEs with MATLAB Simulink

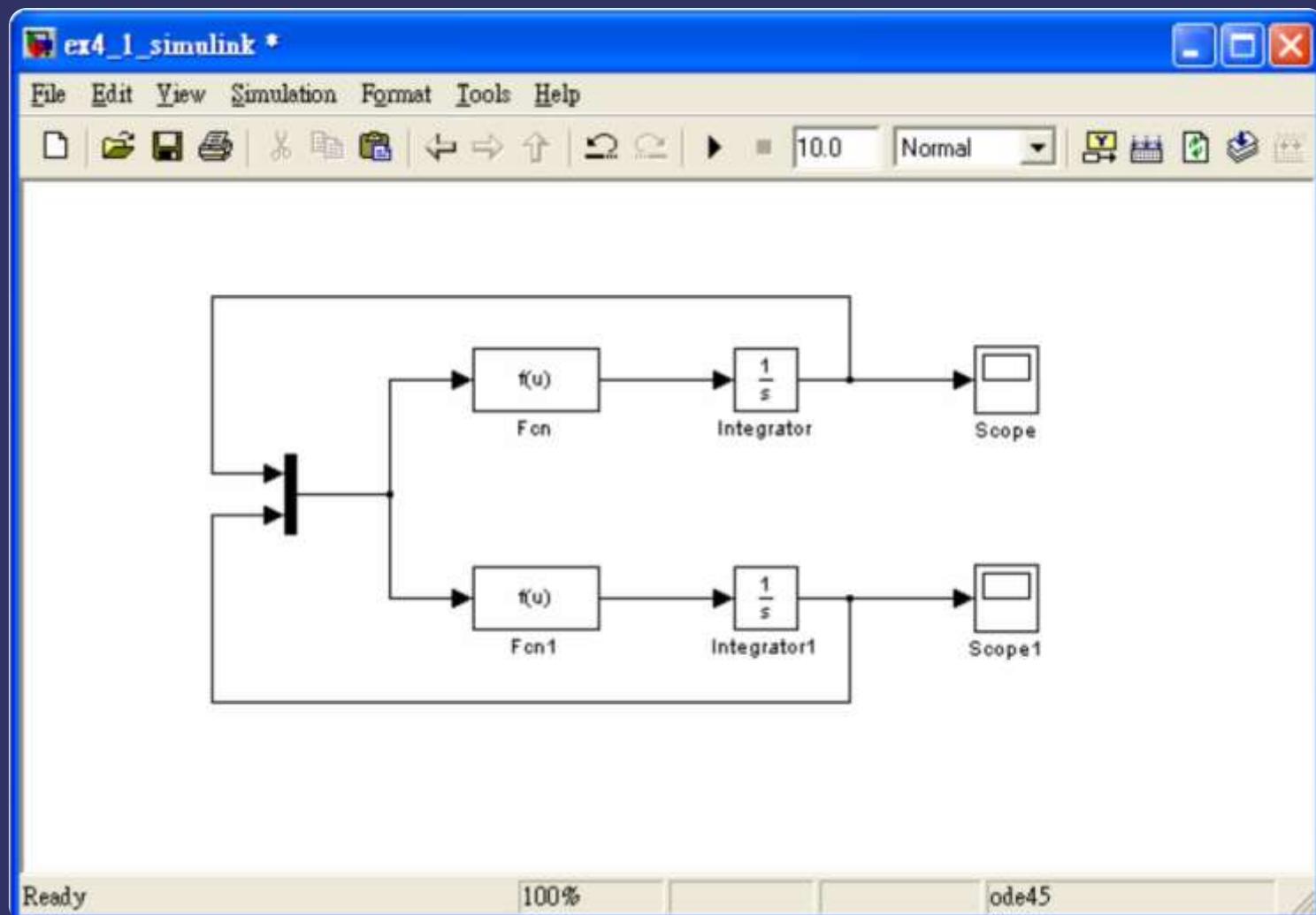
Step 1: Enter the Simulink environment and open a new model file.

Step 2: In Fcn,

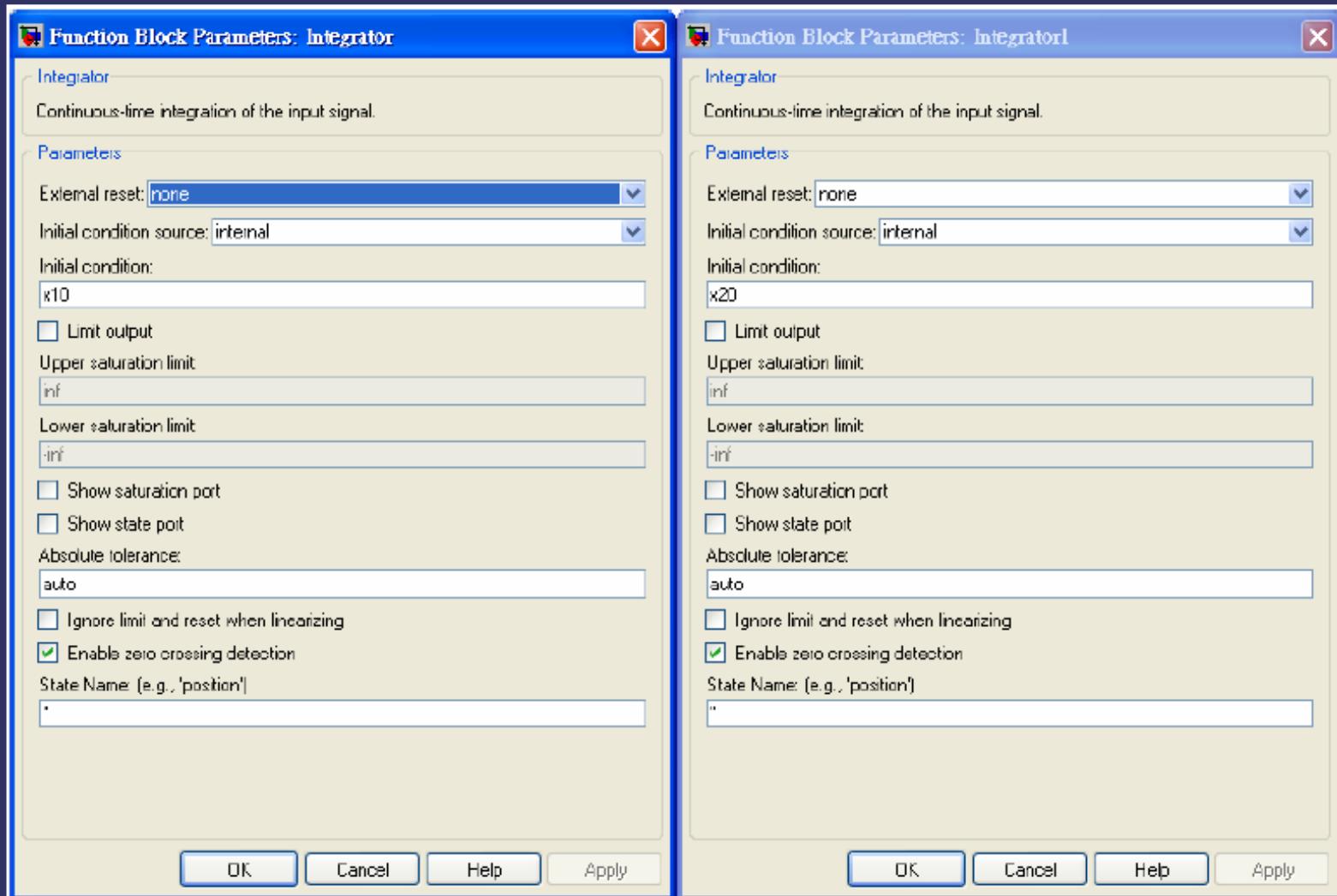
$$-u(1)+Da*(1-u(1))*\exp(u(2)/(1+u(2)/phi))$$

Fcn1:

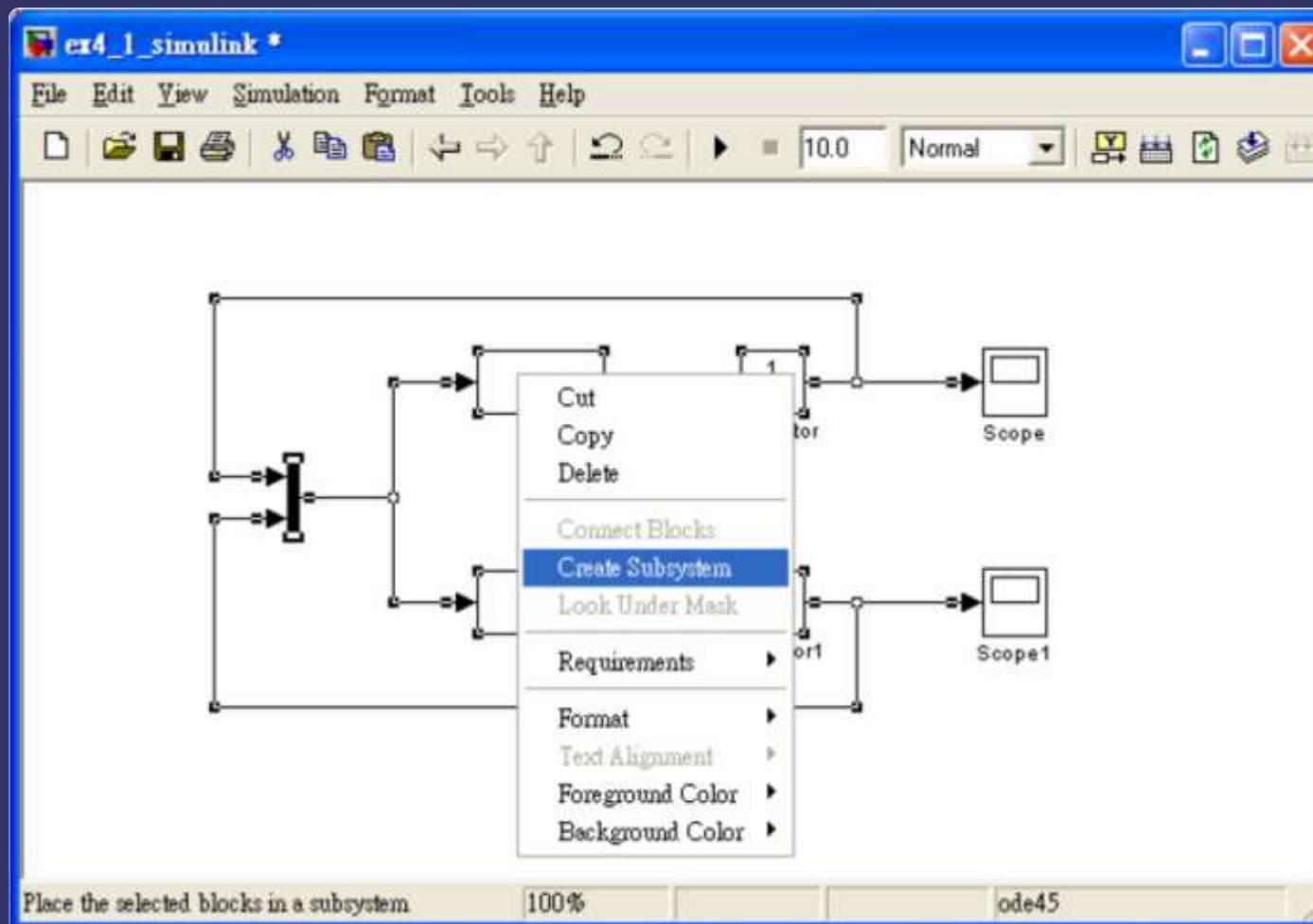
$$-(1+beta)*u(2)+B*Da*(1-u(1))*\exp(u(2)/(1+u(2)/phi))$$



Step 3: Open $\frac{1}{s}$ to key in the initial condition value.

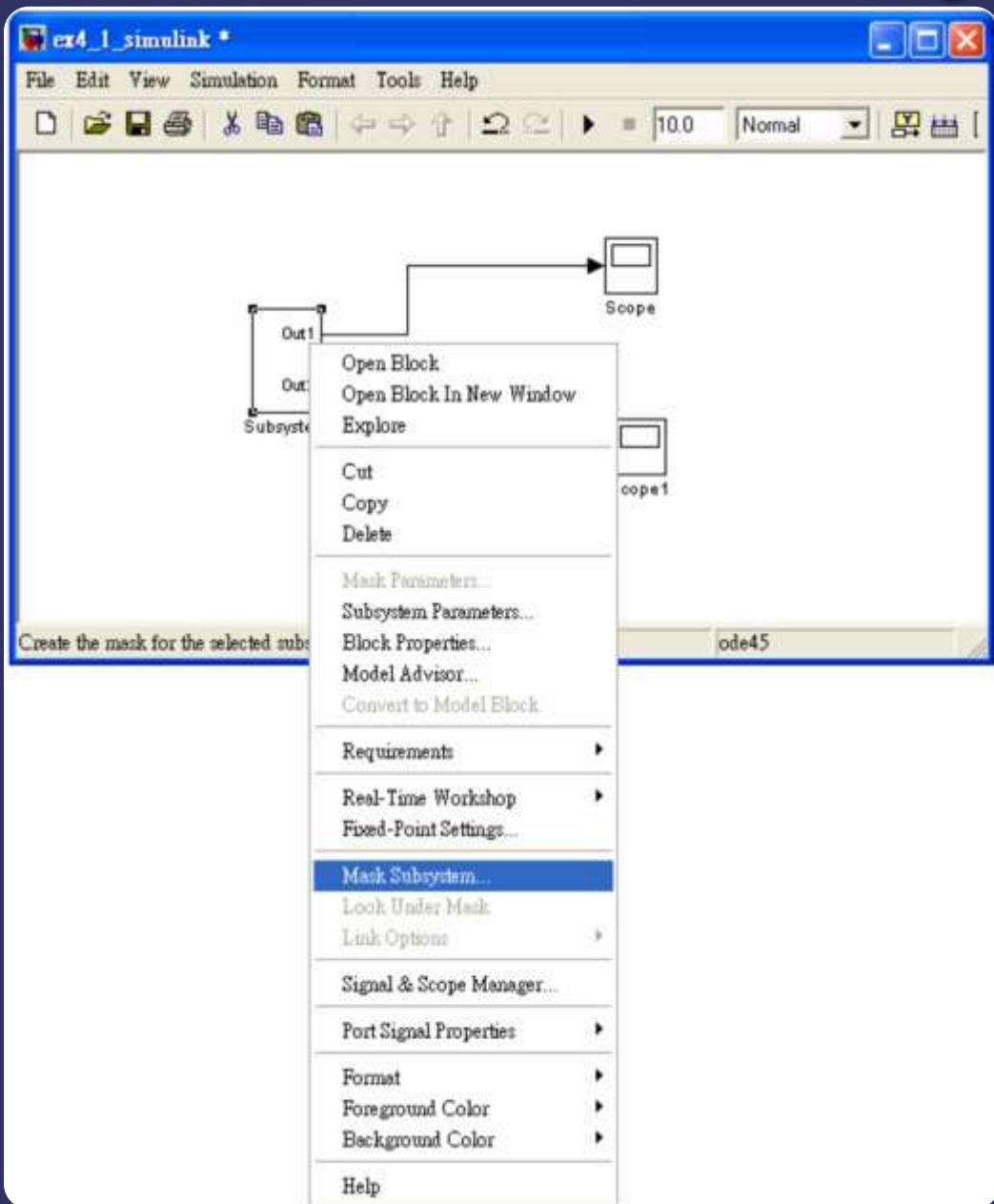


Step 4: Create a subsystem for the constructed module.



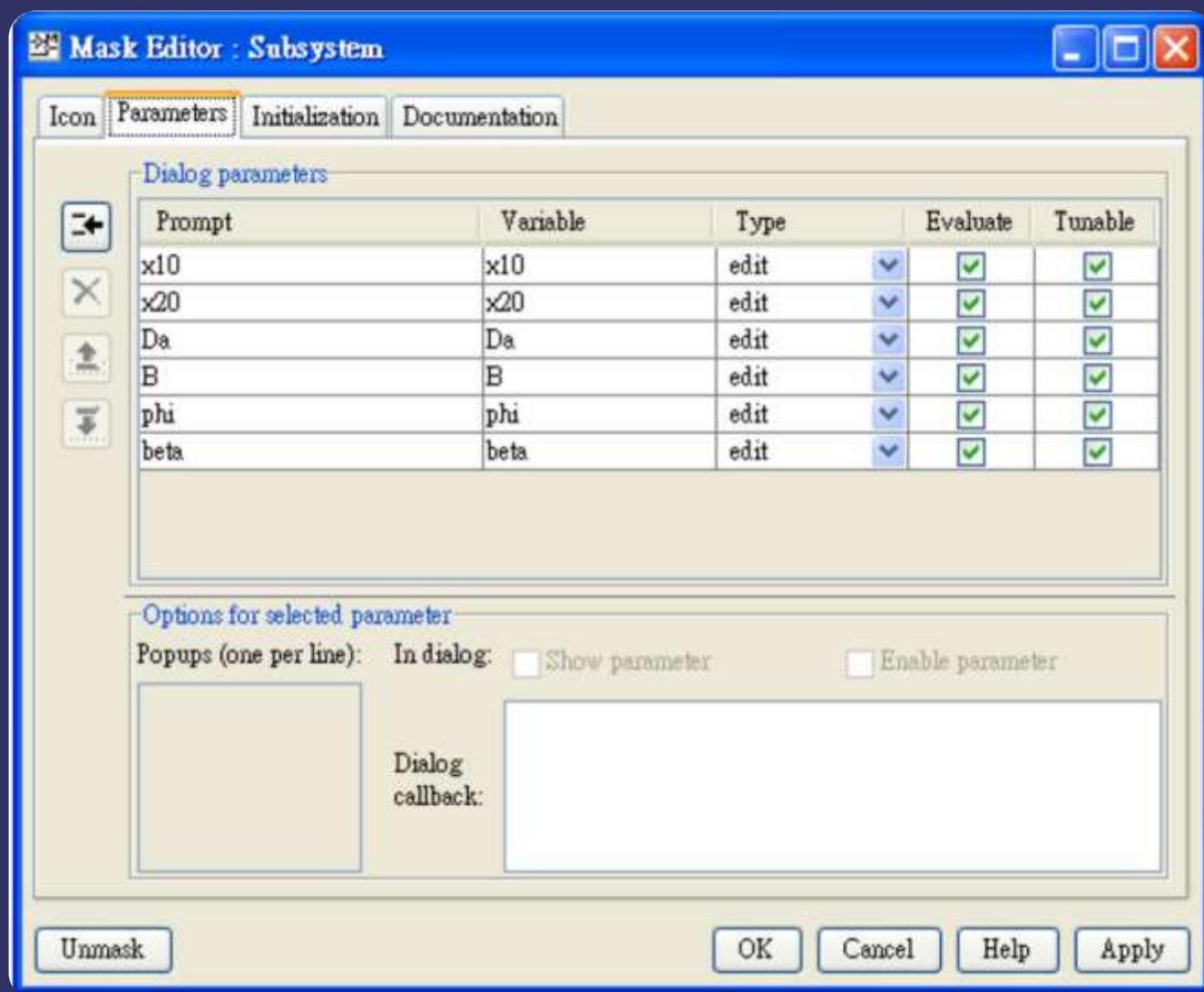
Step 5:

Create the dialogue box.



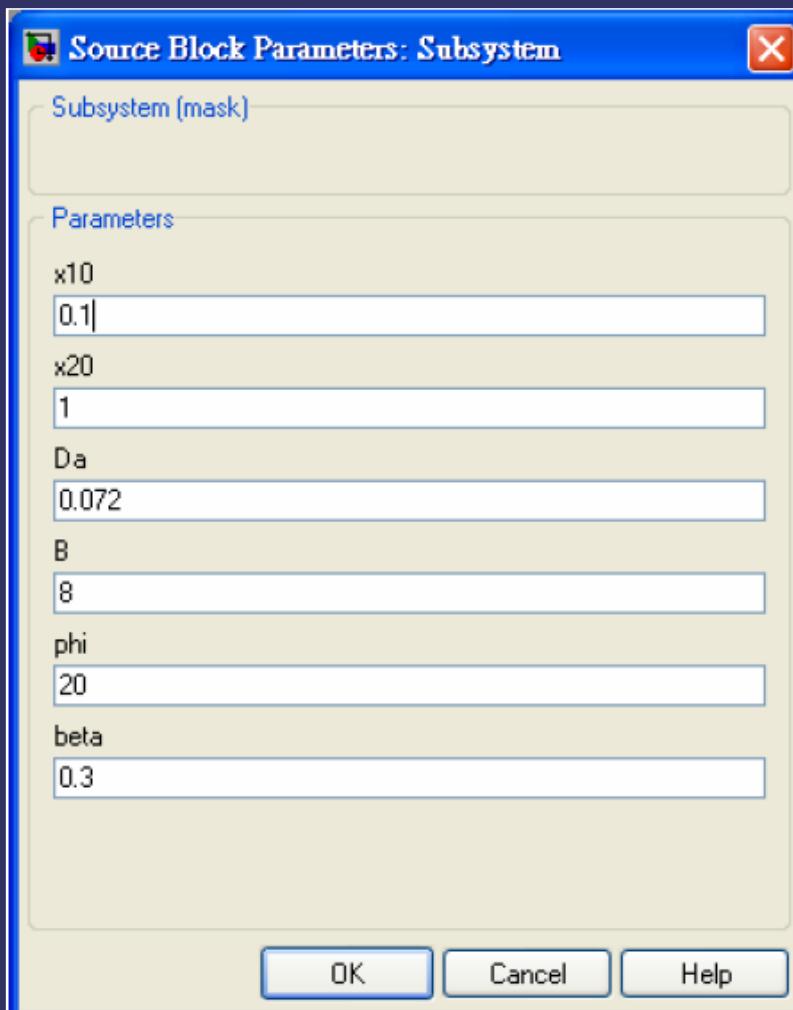
Step 5:

Mask Editor



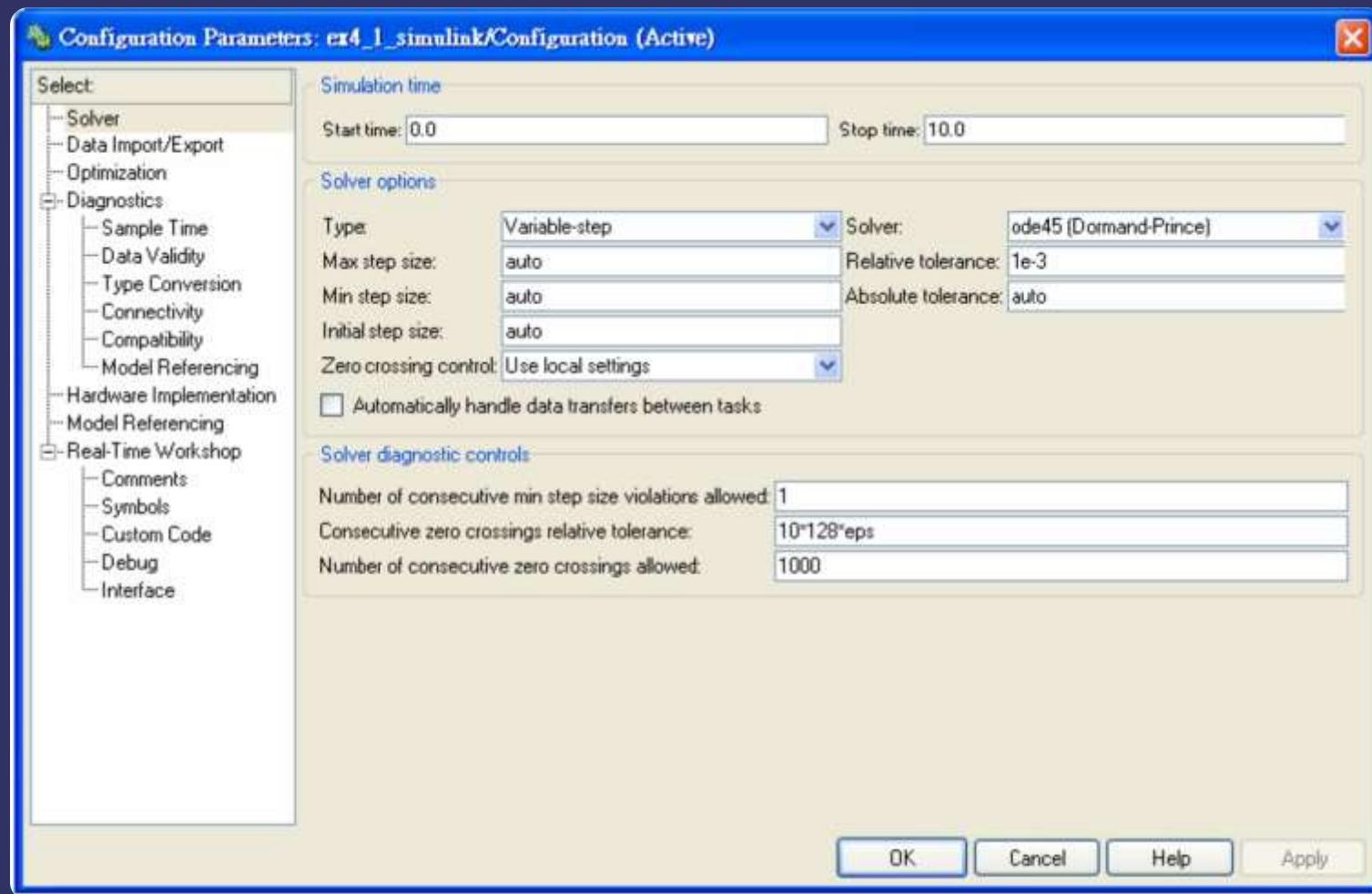
Step 6: Execute the simulation.

- open the dialogue box



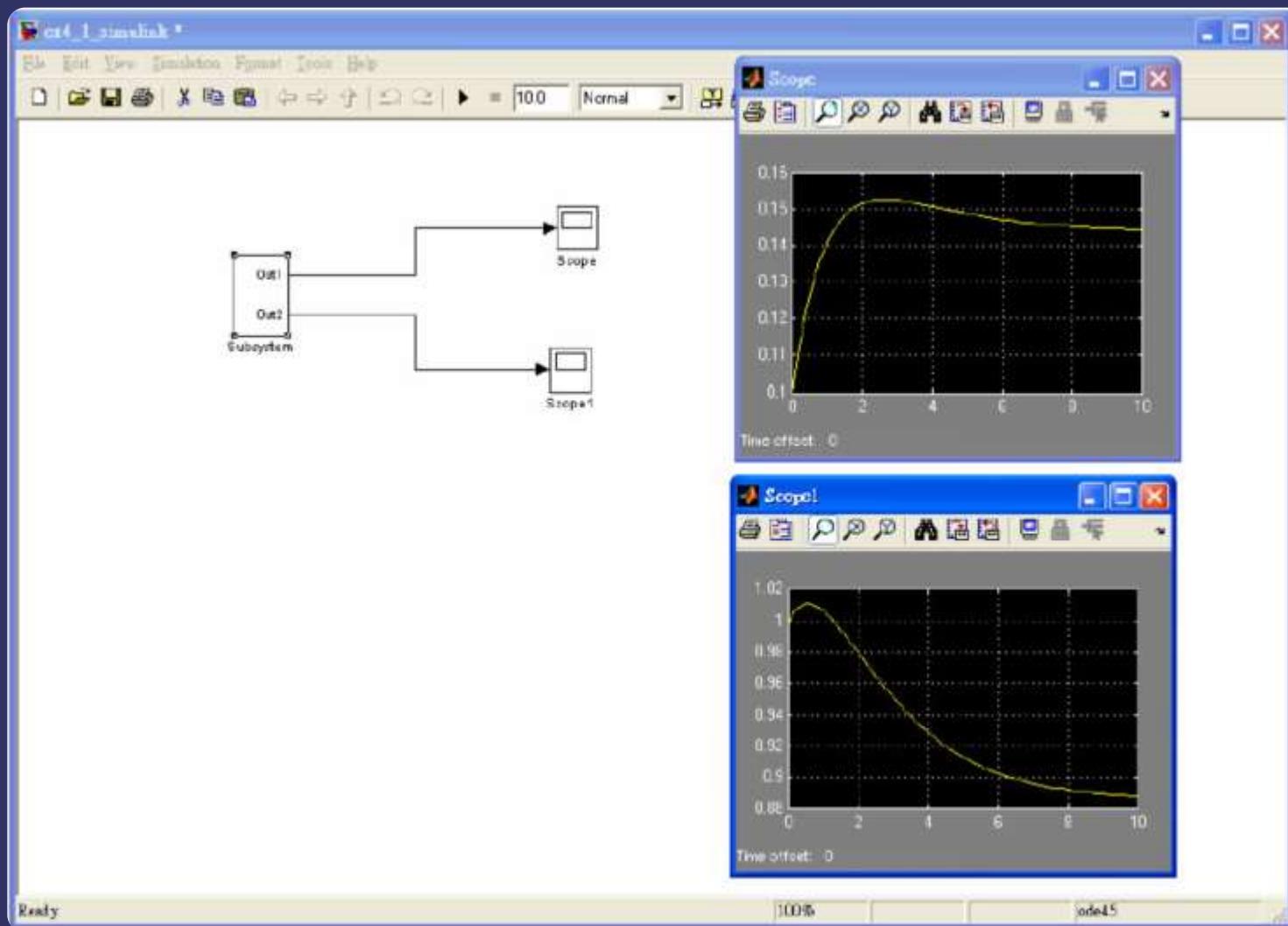
Step 6: Execute the simulation.

- (ii) Input the simulation time and select the numerical method



Step 6: Execute the simulation.

(iii) Press  to simulate



Example 4-1-2

Dynamic simulation of a CSTR with Simulink

Revisit the same CSTR stated in Example 4-1-1 by introducing the following three kinds of forcing functions:

(1) Unit step function

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

(2) Sine function with delay

$$u(t) = \begin{cases} \sin(t - 2), & t \geq 2 \\ 0, & t < 2 \end{cases}$$

(3) Proportional- integral (PI) control law for temperature regulation

$$u(t) = K_c \left[(x_{2s} - x_2) + \frac{1}{T_I} \int_0^t (x_{2s} - x_2) dt \right]$$

$$x_{2s} = 3$$

$$K_c = 5 \text{ and } T_I = 10$$

Ans:

Fcn

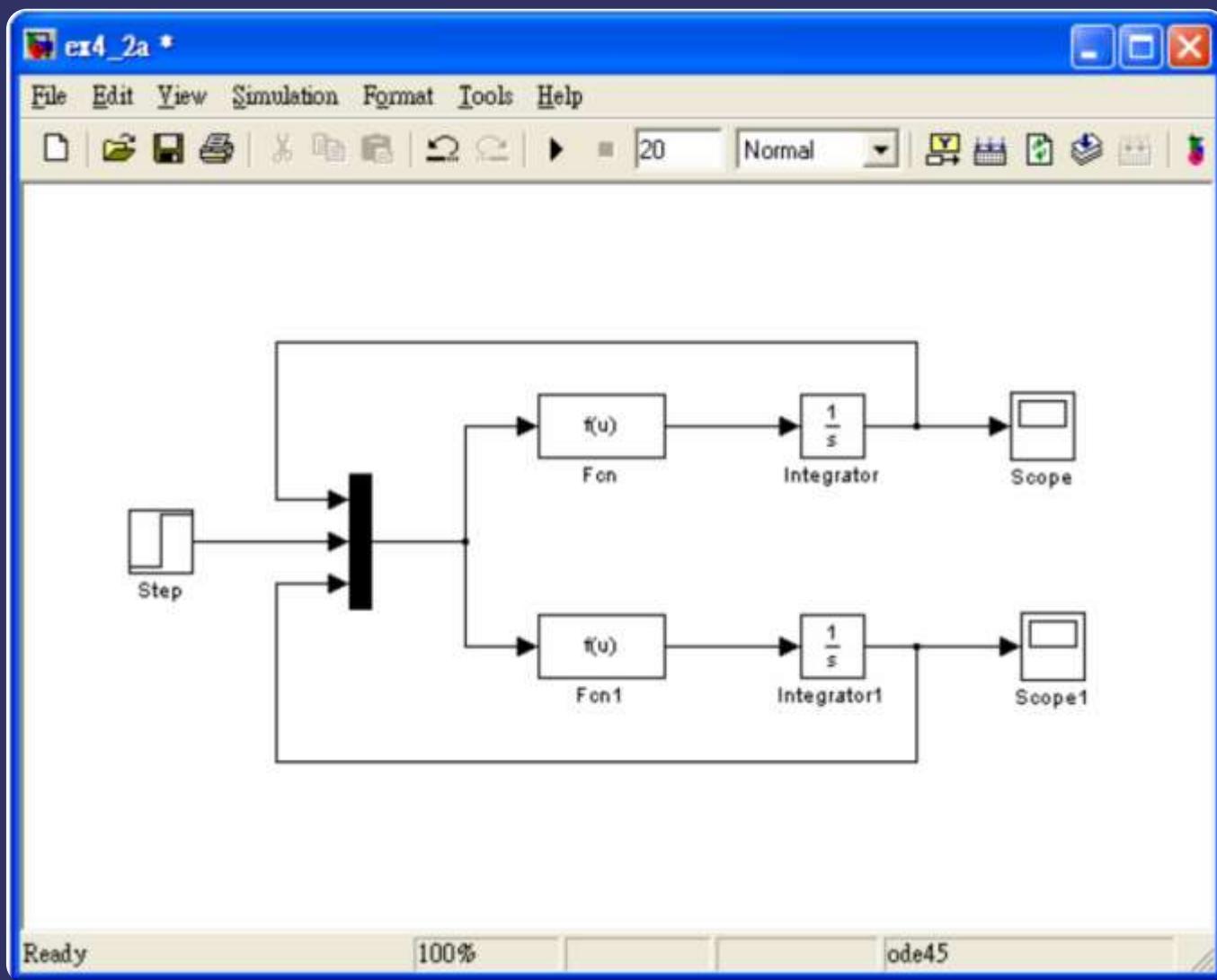
$$-u(1) + Da * (1 - u(1)) * \exp(u(3) / (1 + u(3) / \phi))$$

Fcn1

$$-(1 + \beta) * u(3) + B * Da * (1 - u(1)) * \exp(u(3) / (1 + u(3) / \phi)) + \beta * u(2)$$

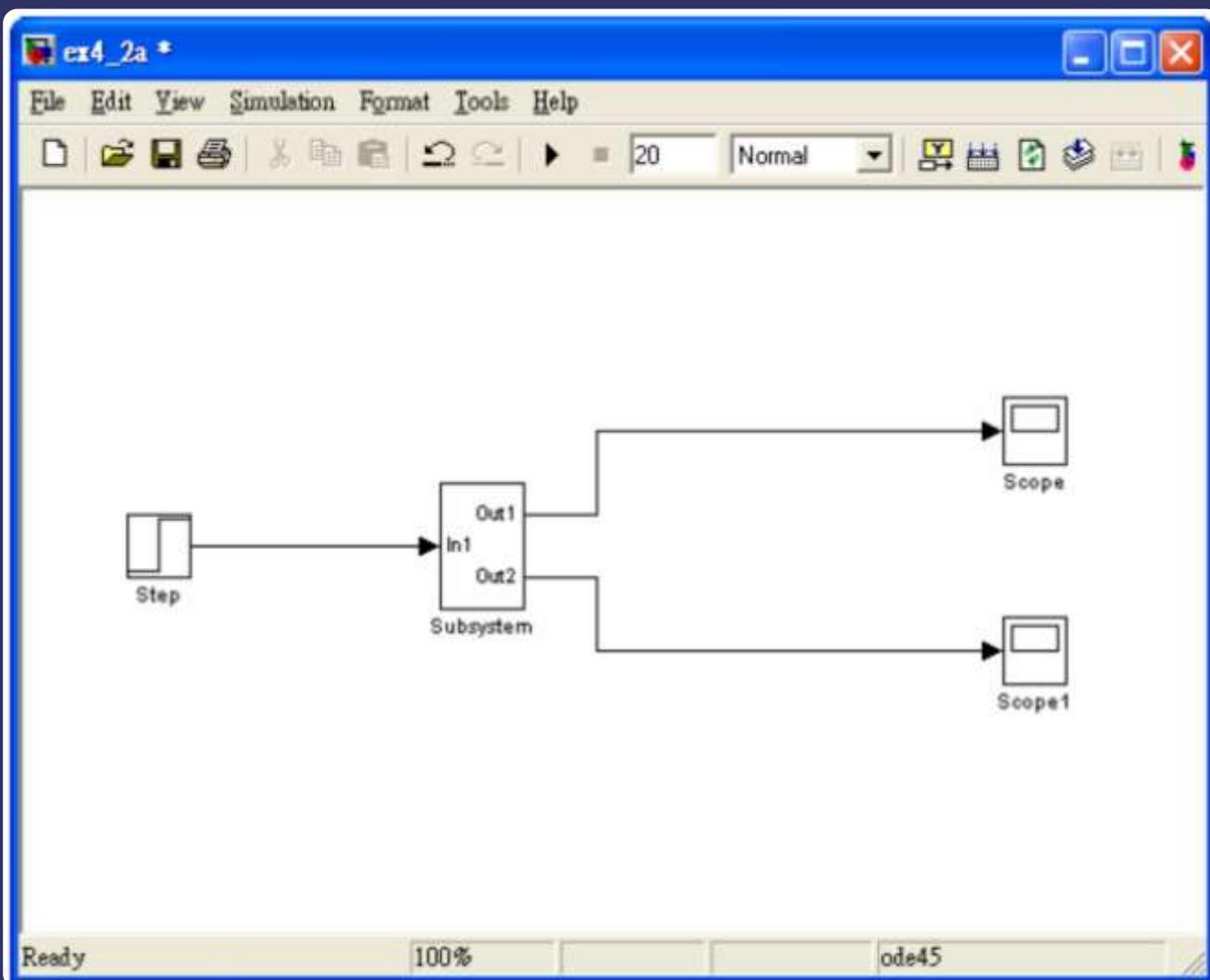
(1) Case 1: unit step function

Ans:



Ans:

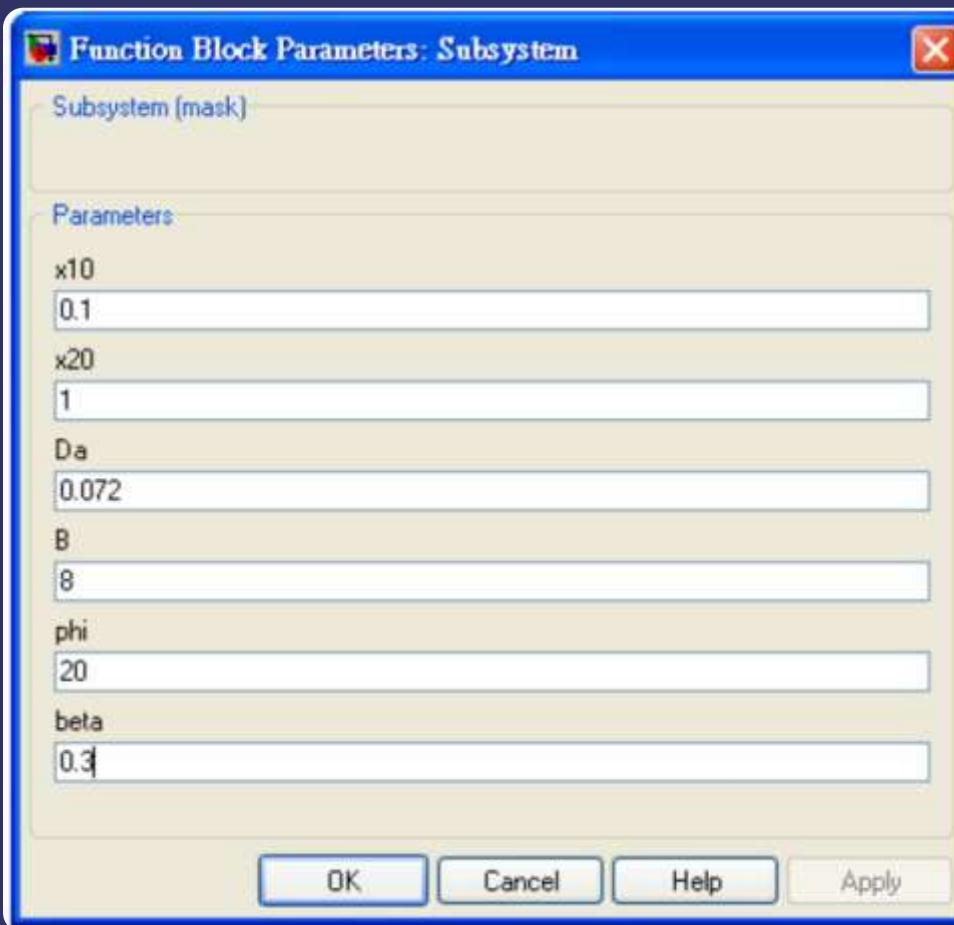
Steps 3-5



Ans:

Step 6: Execute the simulation.

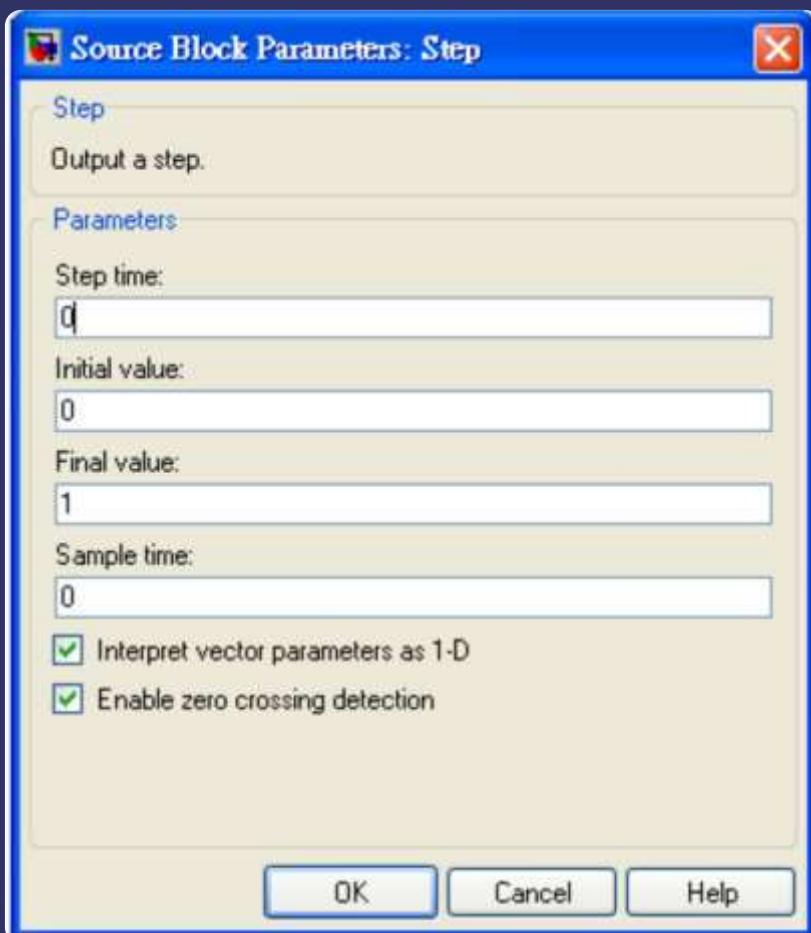
(i)



Ans:

Step 6: Execute the simulation.

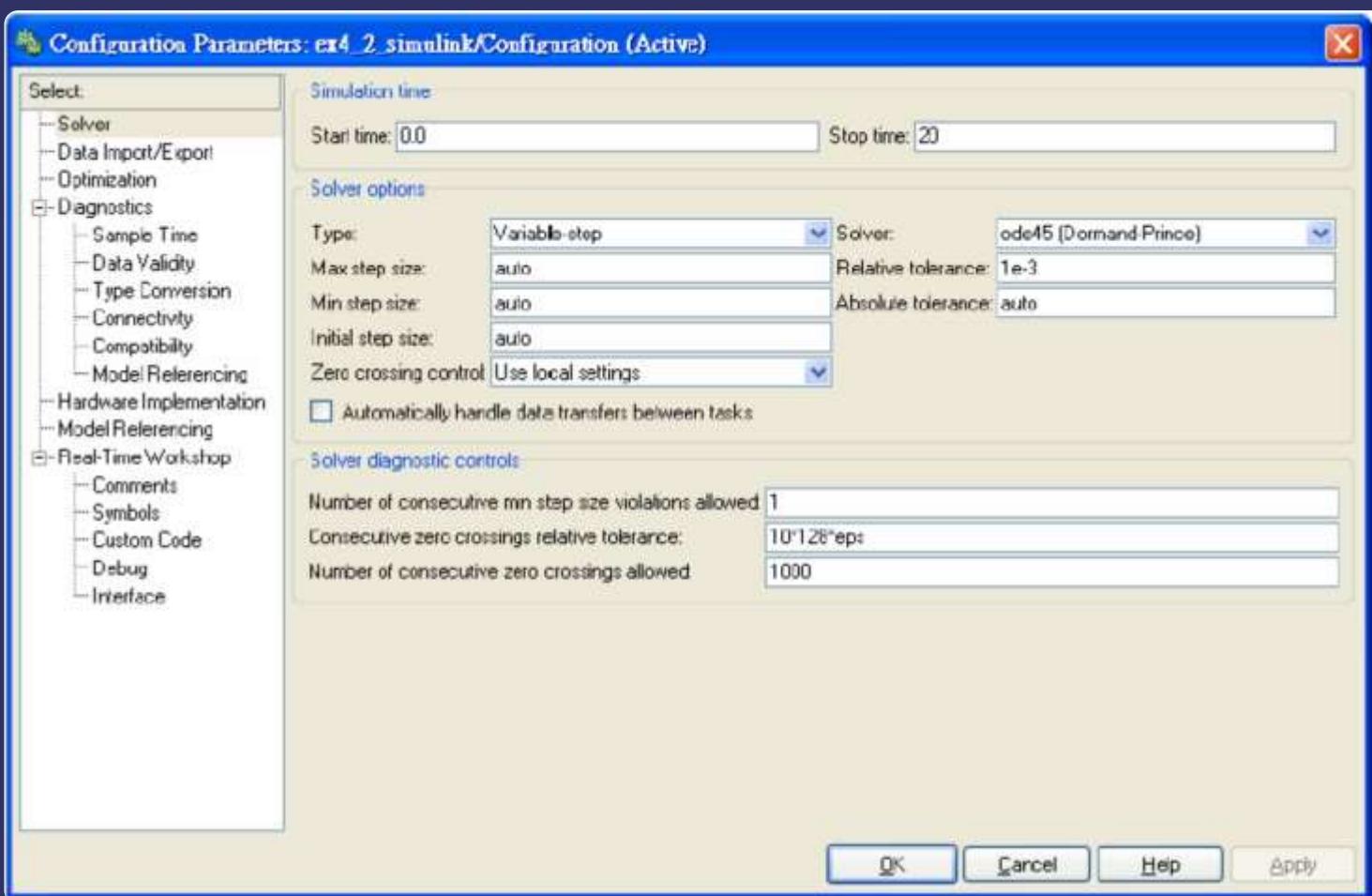
(ii)



Ans:

Step 6: Execute the simulation.

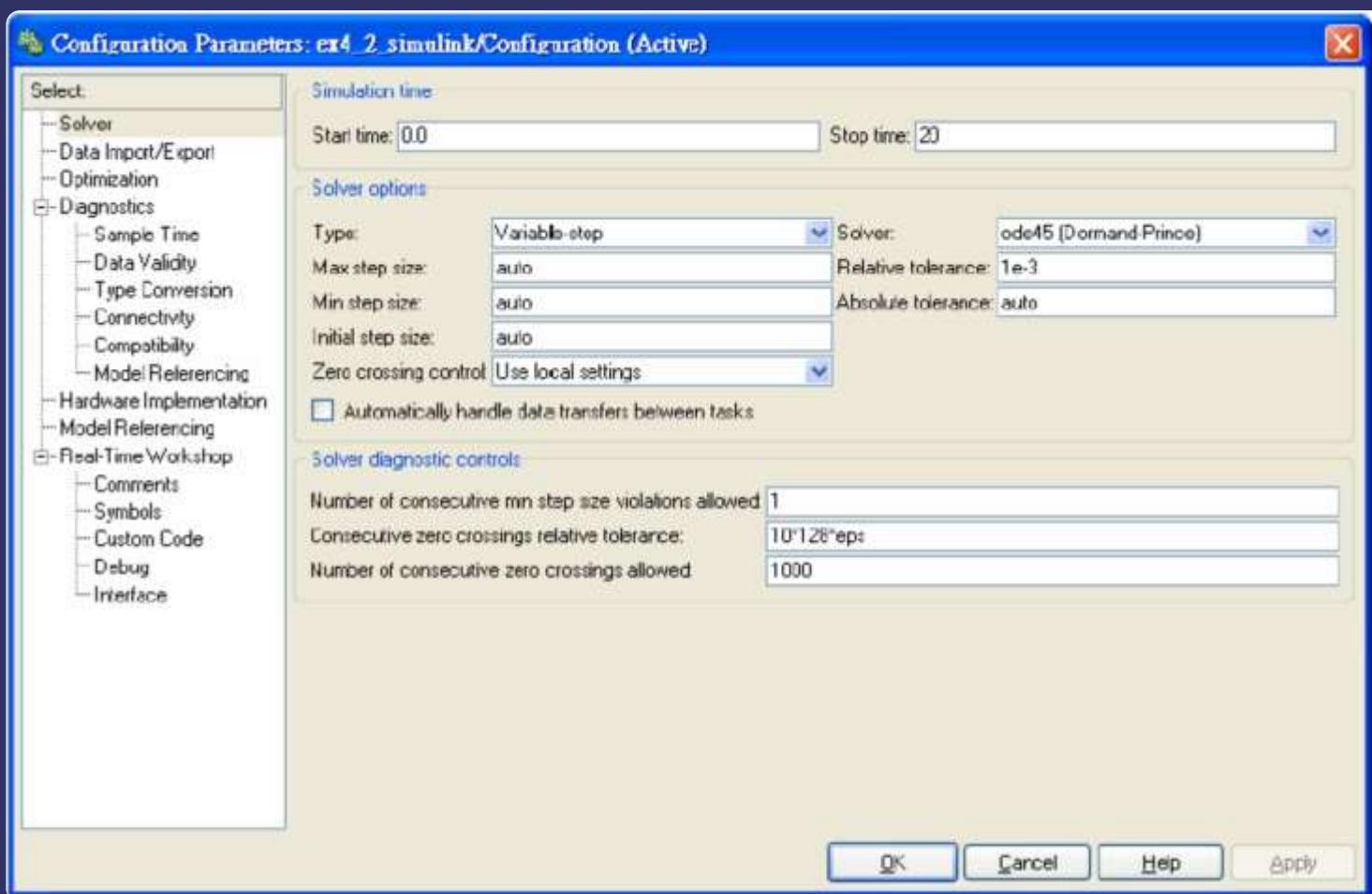
(iii)



Ans:

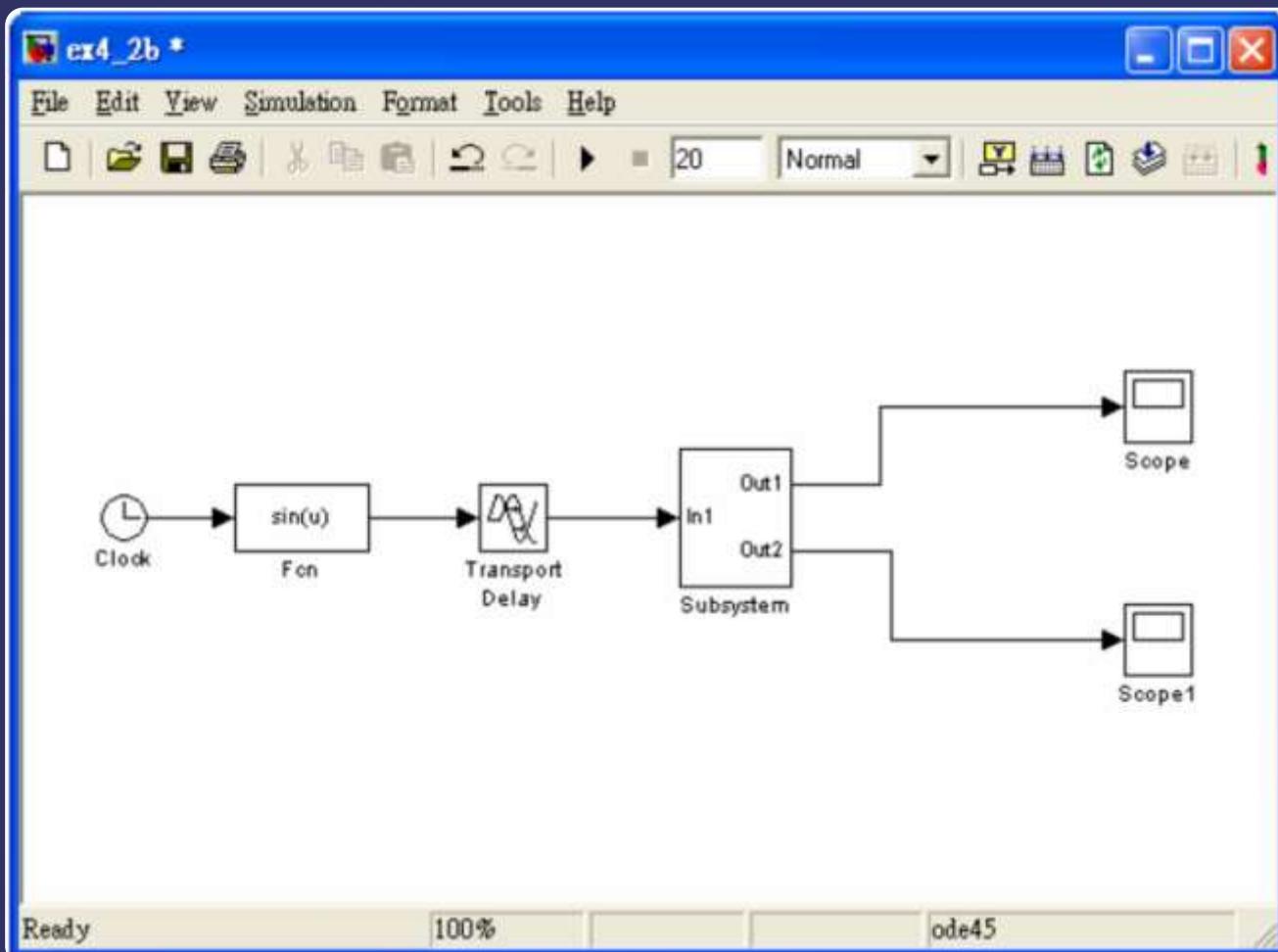
Step 6: Execute the simulation.

(iii)



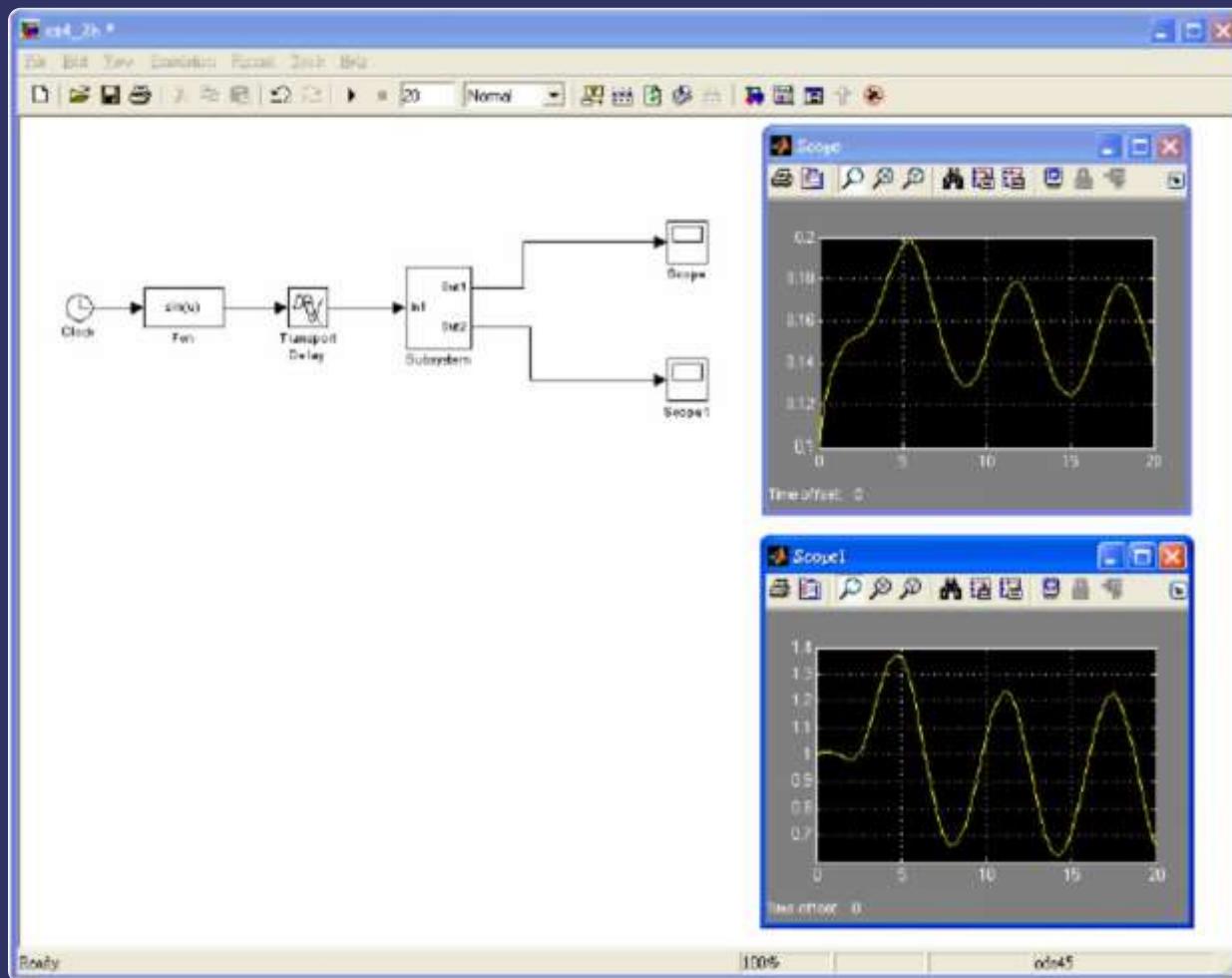
Ans:

(2) Case 2: Sine function with delay



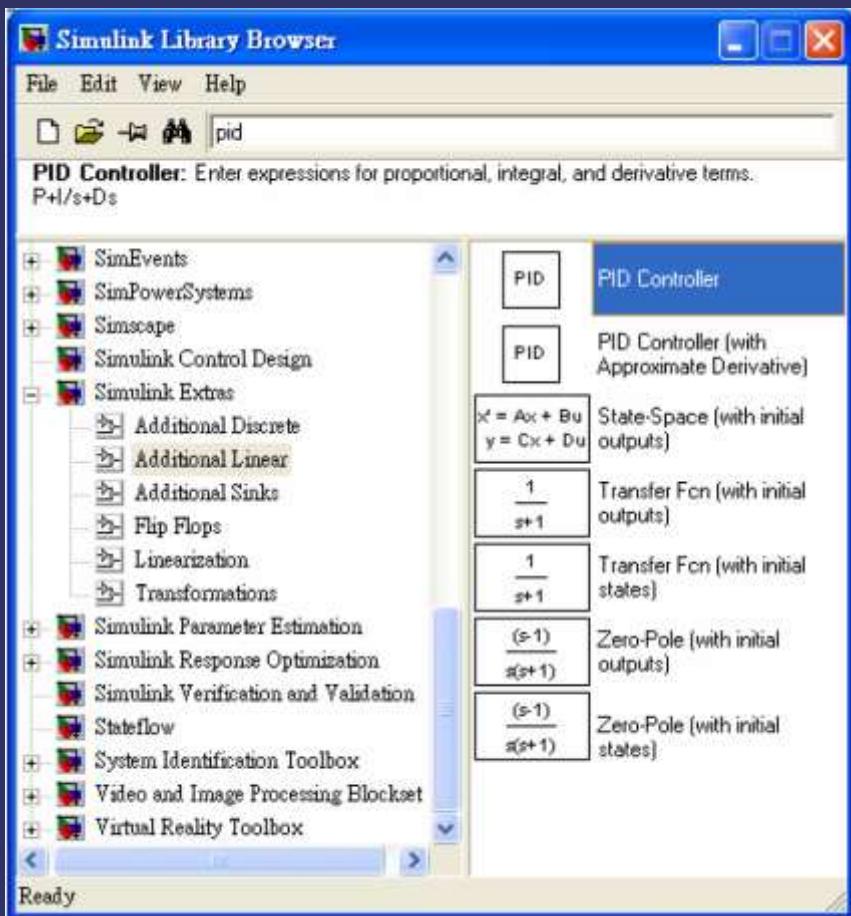
Ans:

(2) Case 2: Sine function with delay



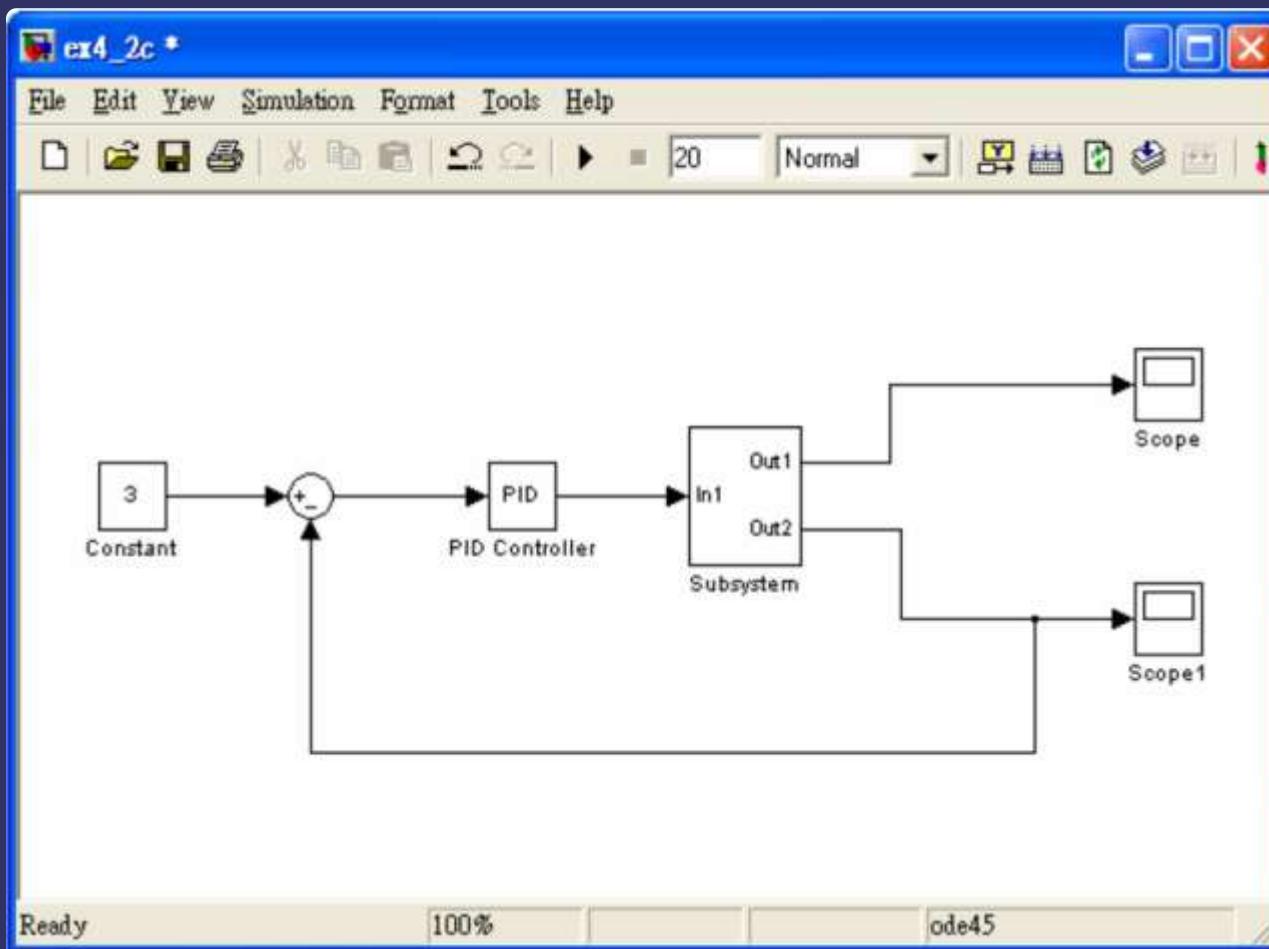
Ans:

(3) Case 3: proportional-integral control locate PID Controller



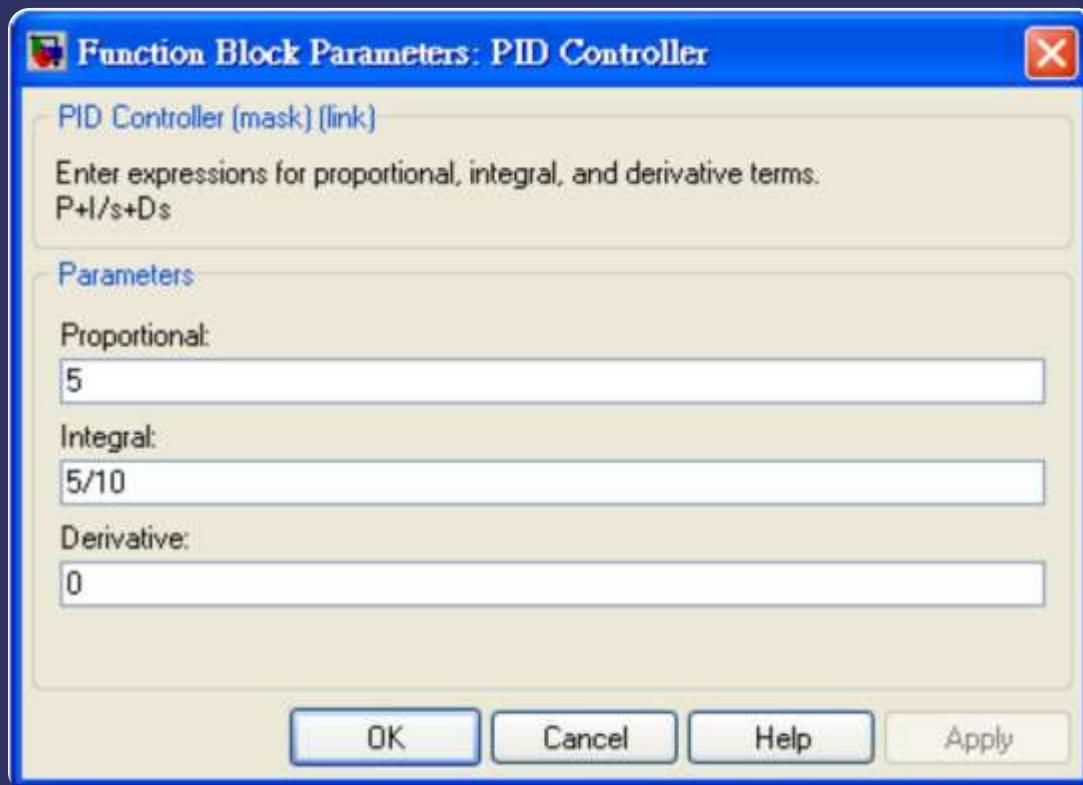
Ans:

(3) Case 3: proportional-integral control



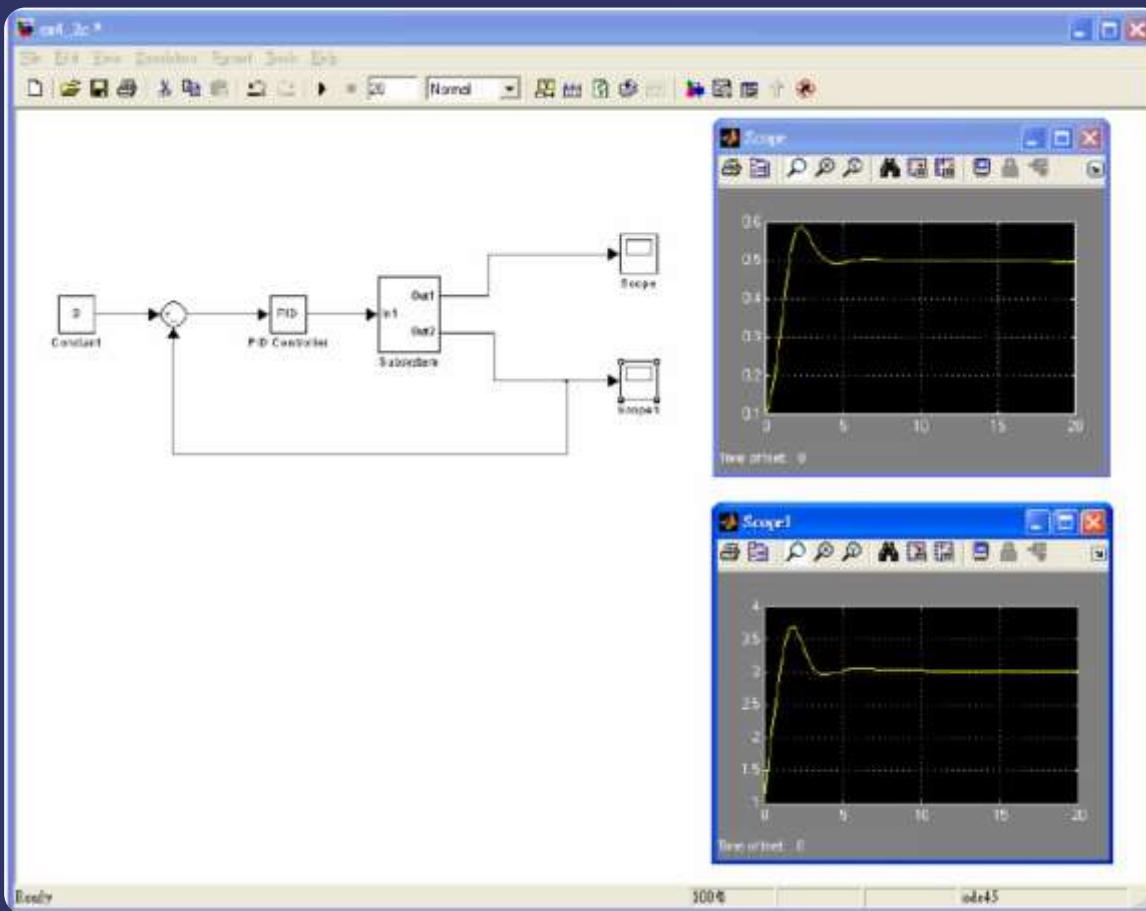
Ans:

(3) Case 3: proportional-integral control



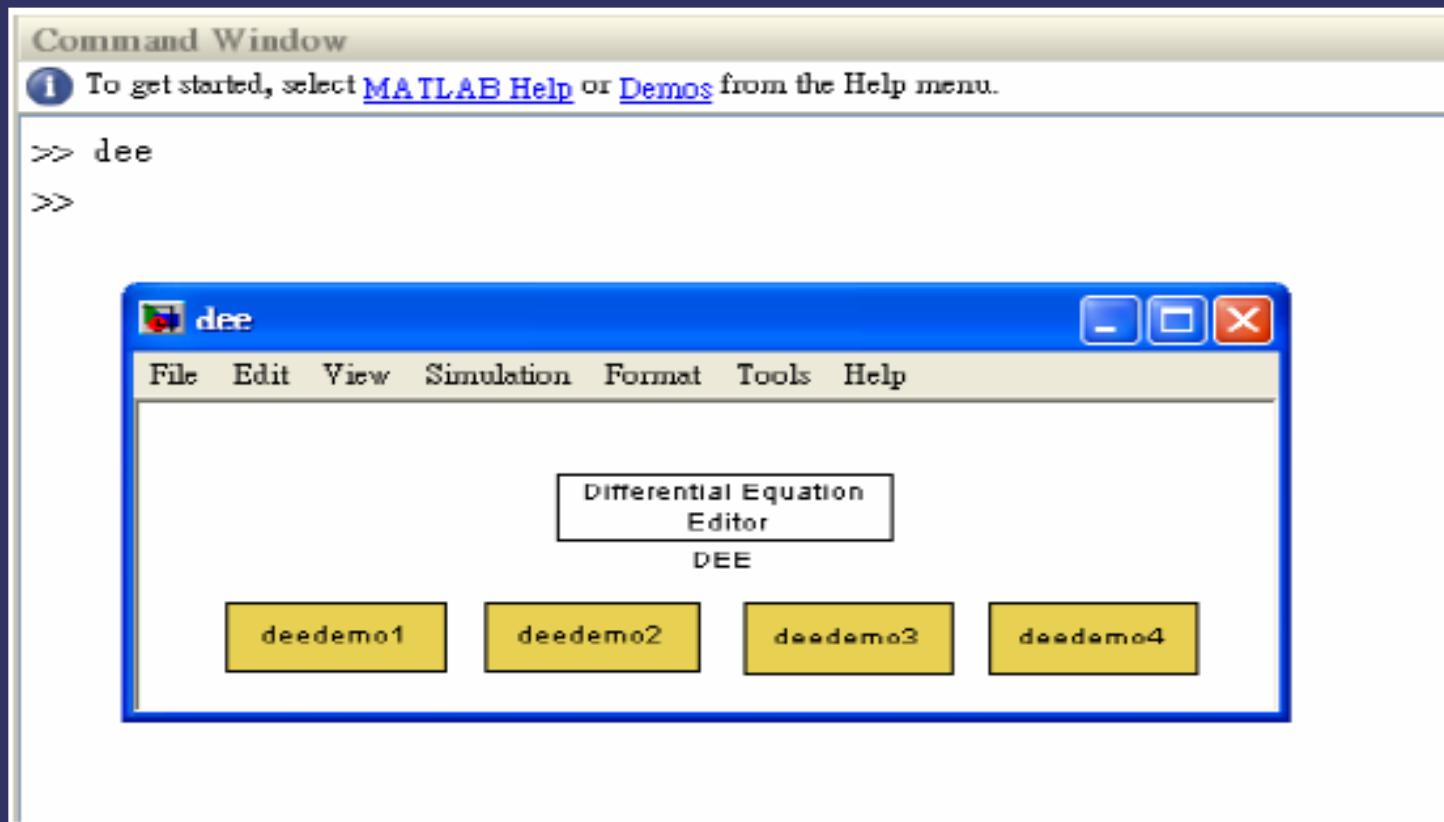
Ans:

(3) Case 3: proportional-integral control

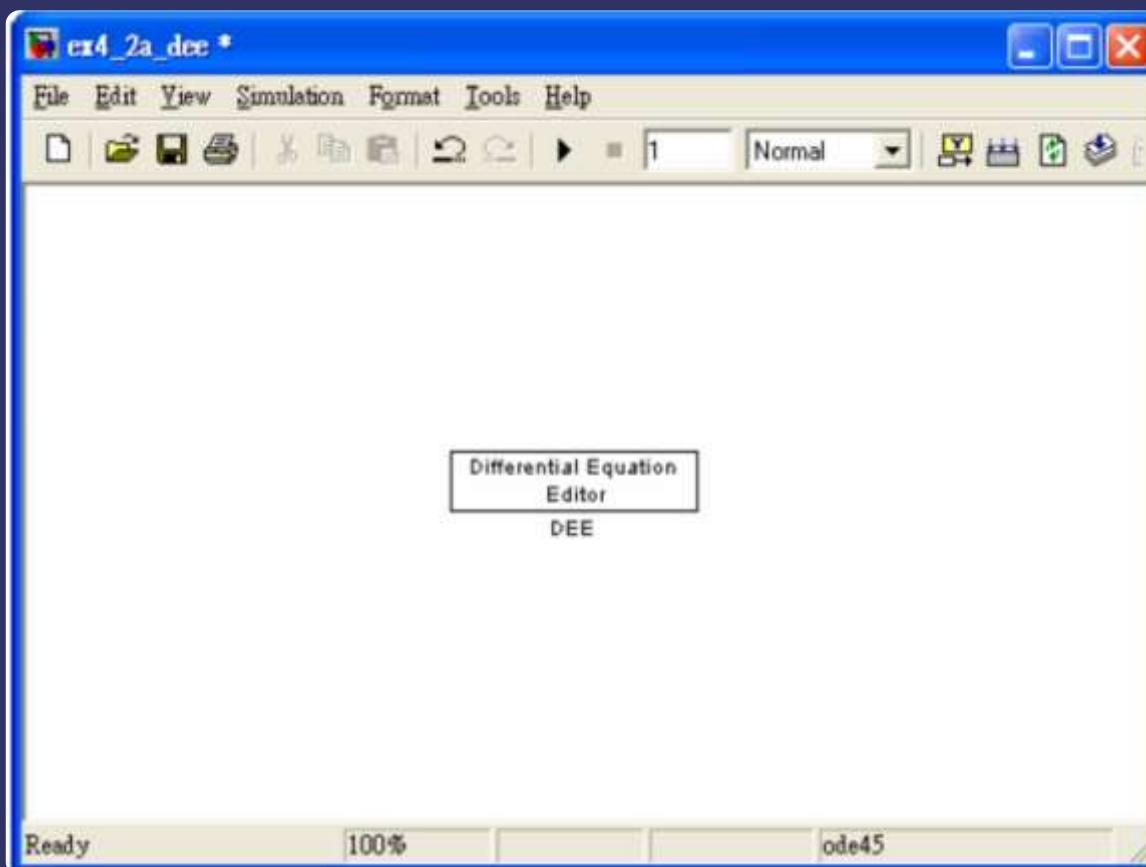


4.1.4 The DEE solution interface

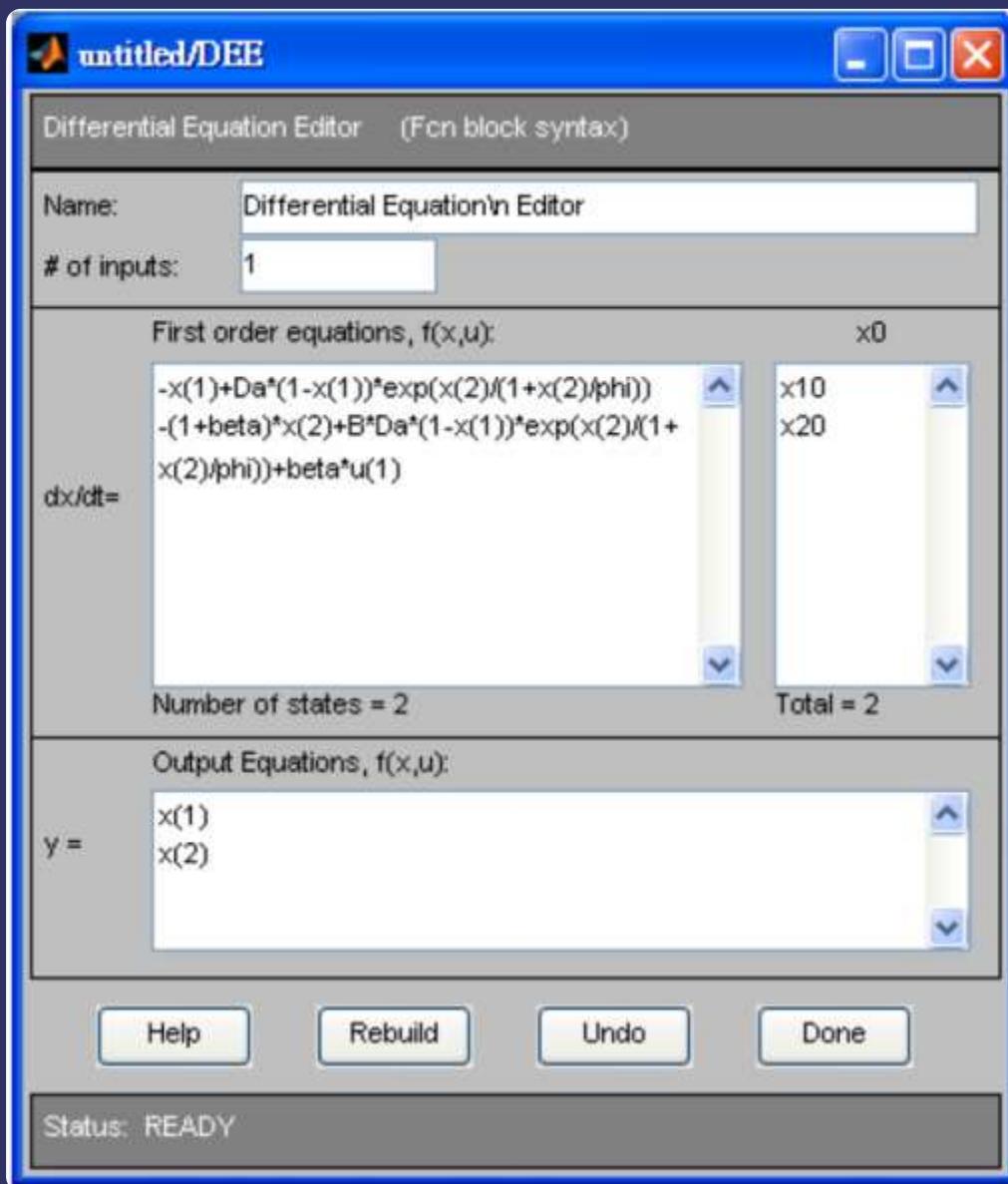
Step 1: Key in dee in the Command Window to launch the DEE editor as follows:



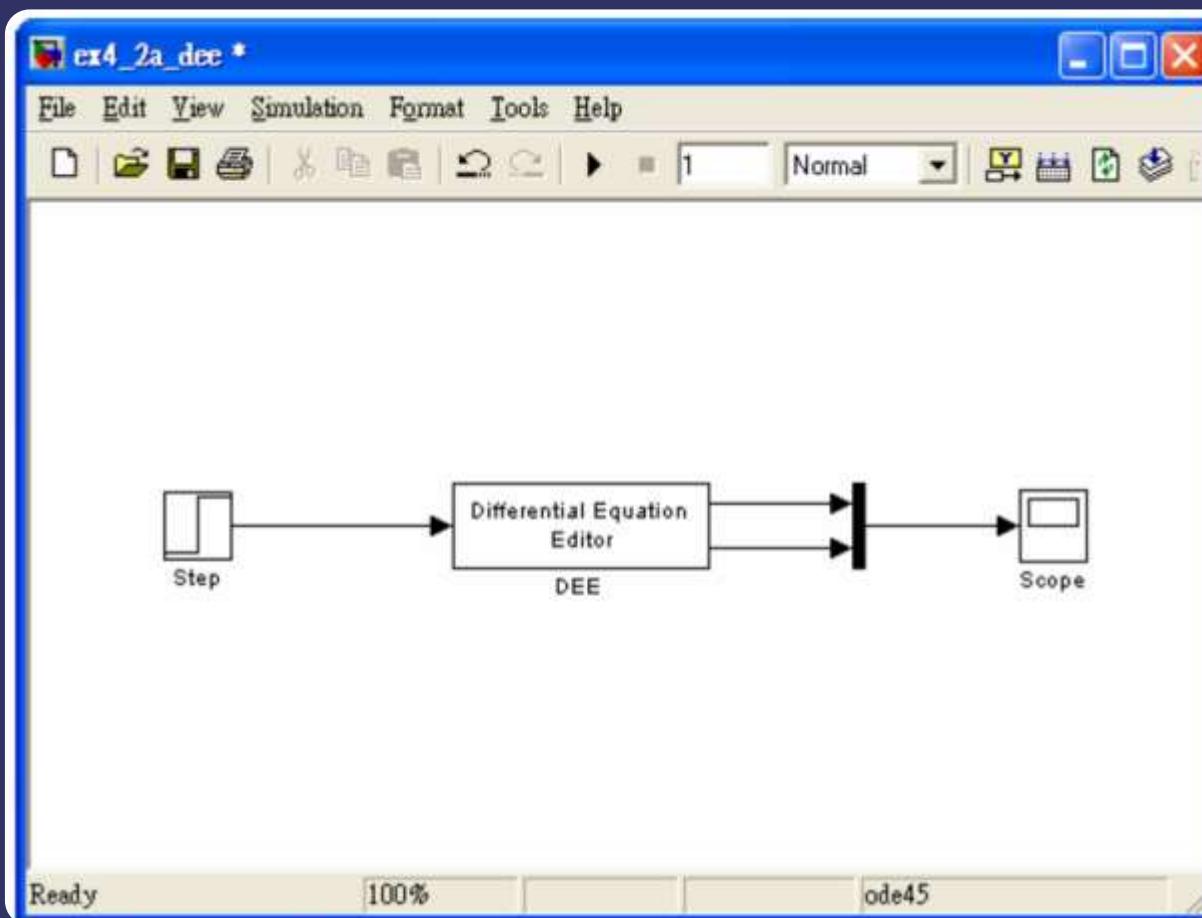
Step 2: After entering Simulink, open a new model file and copy one DEE into it.



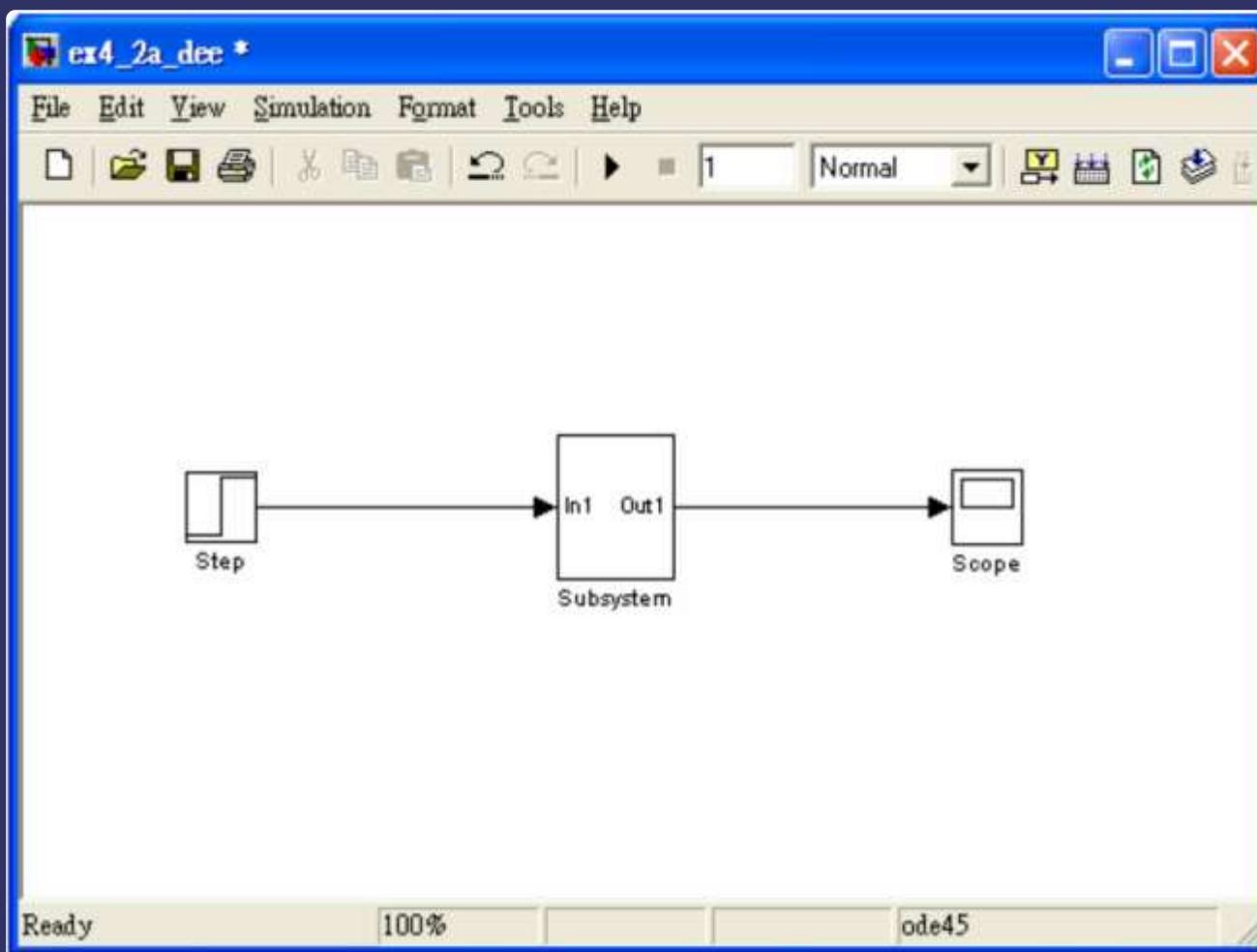
Step 3:



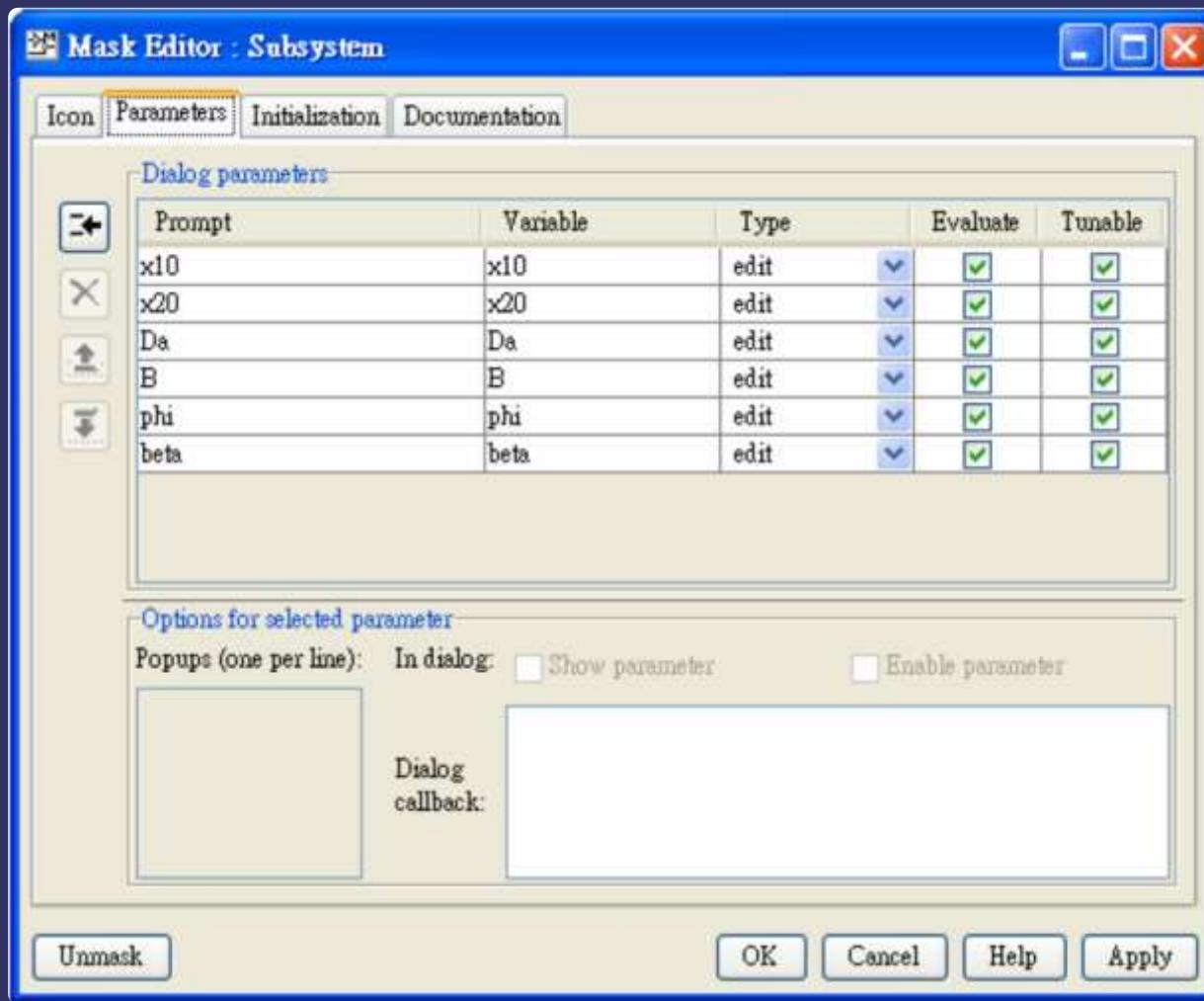
Step 4:



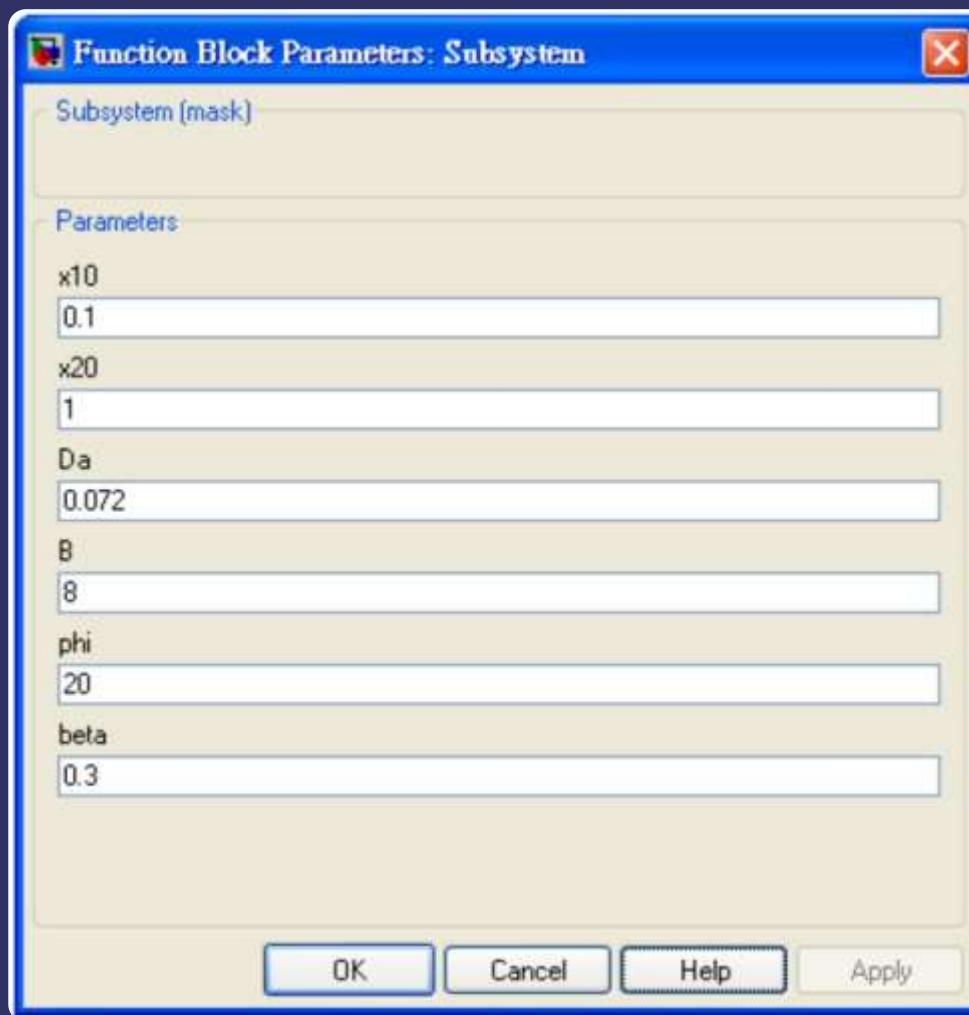
Step 5: Create the CSTR Subsystem and a dialogue box.



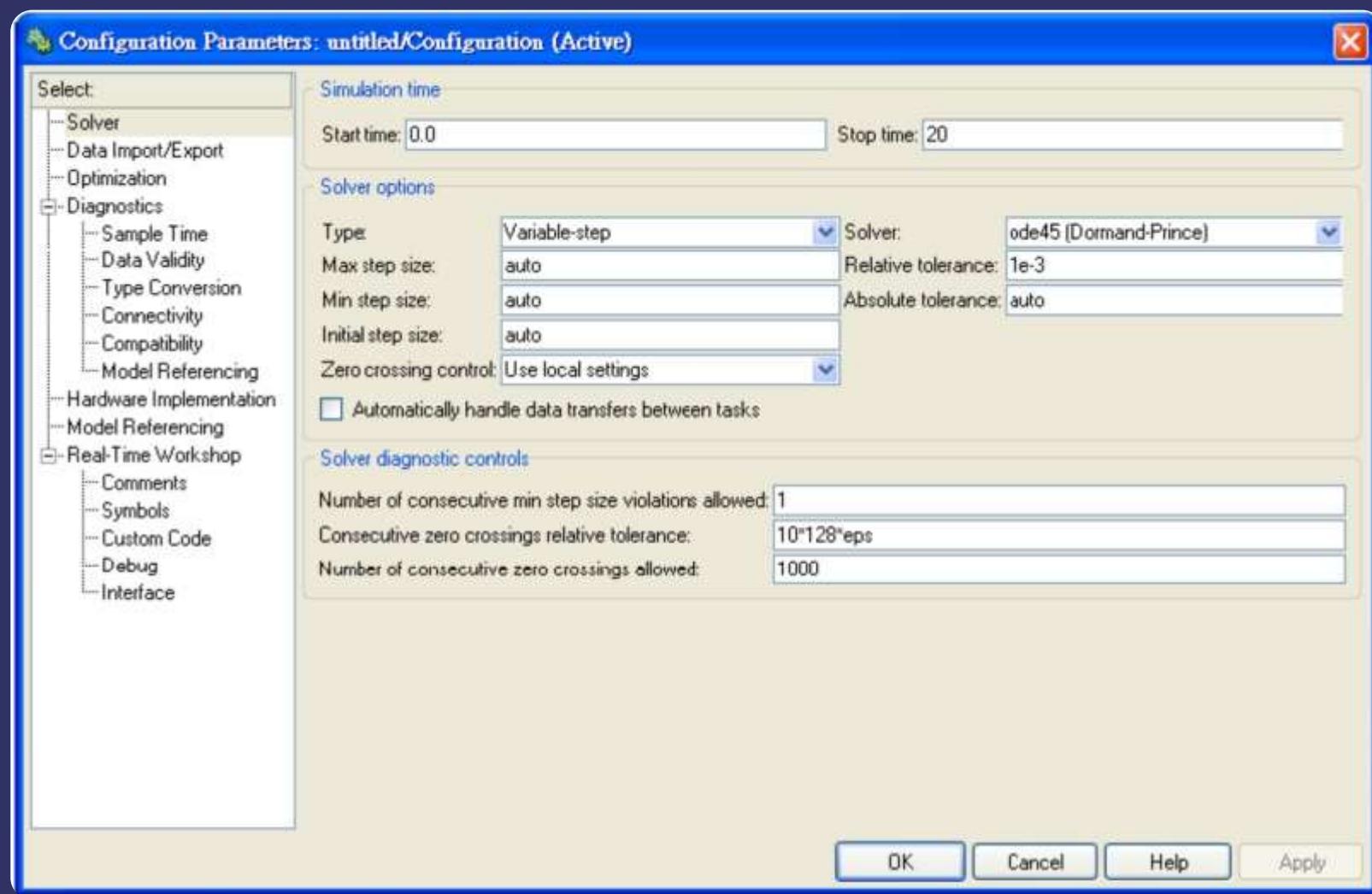
Step 5: Create the CSTR Subsystem and a dialogue box.



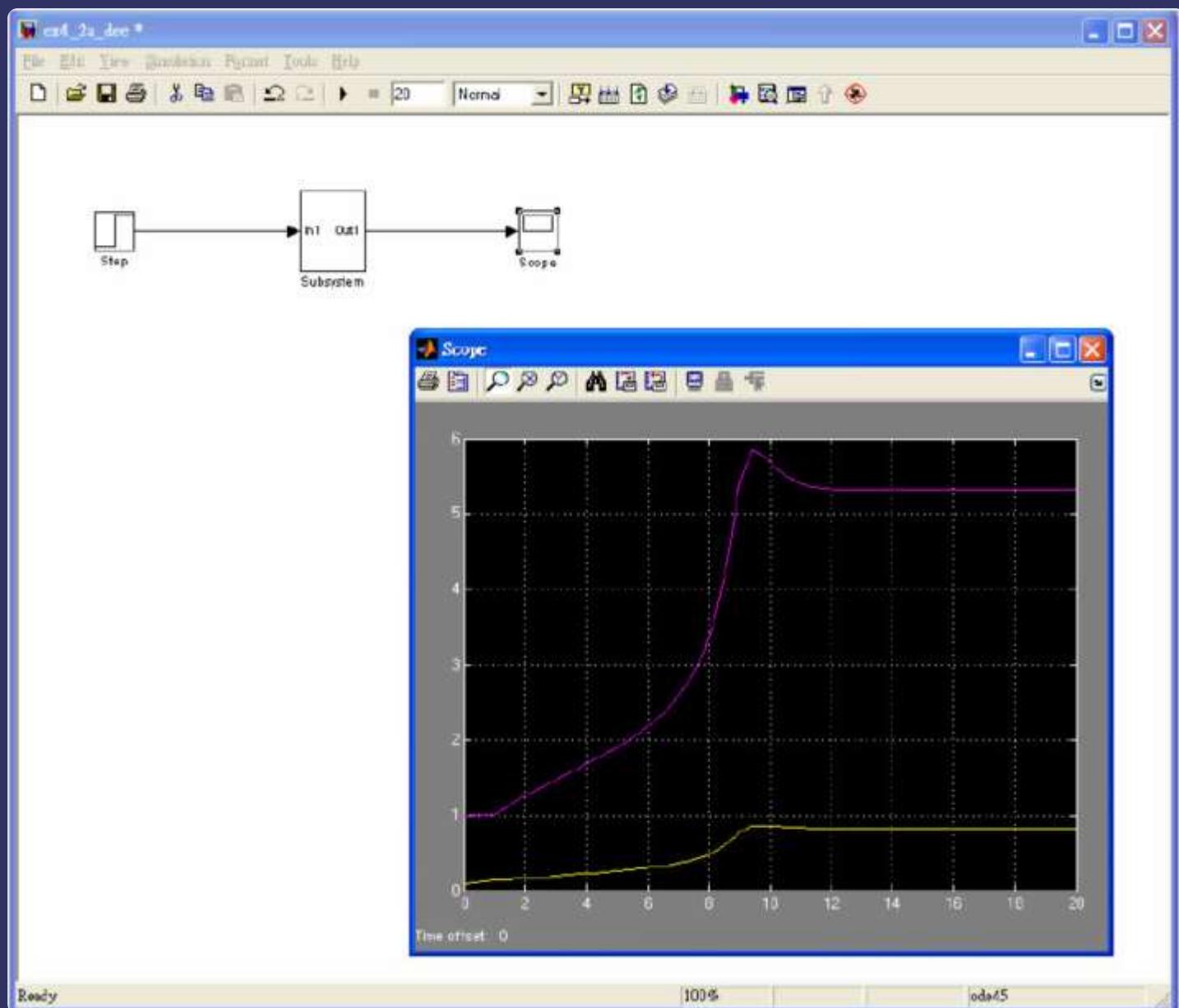
Step 5: Create the CSTR Subsystem and a dialogue box.



Step 6:



Step 6:





4.2 Higher-order ordinary differential equations

$$y^{(n)} + a_{n-1}y^{(n-1)} + a_{n-2}y^{(n-2)} + \dots + a_1y' + a_0y = u$$

$$y^{(i)}(0) = y_{i0}, \quad i = 0, 1, \dots, n-1$$

$$x_1 = y$$

$$x_2 = y'$$

⋮

$$x_n = y^{(n-1)}$$



$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = x_3$$

$$\vdots$$

$$\dot{x}_{n-1} = x_n$$

$$\dot{x}_n = -a_{n-1}x_n - a_{n-2}x_{n-1} - \dots - a_1x_2 - a_0x_1 + u$$

$$x_1(0) = y_{00}$$

$$x_2(0) = y_{10}$$

$$\vdots$$

$$x_n(0) = y_{n-1,0}$$

Example 4-2-1

Dynamic behavior of a cone-and-plate viscometer

Figure 4.1 schematically depicts a cone-and-plate viscometer that is used to measure the viscosity of a fluid.

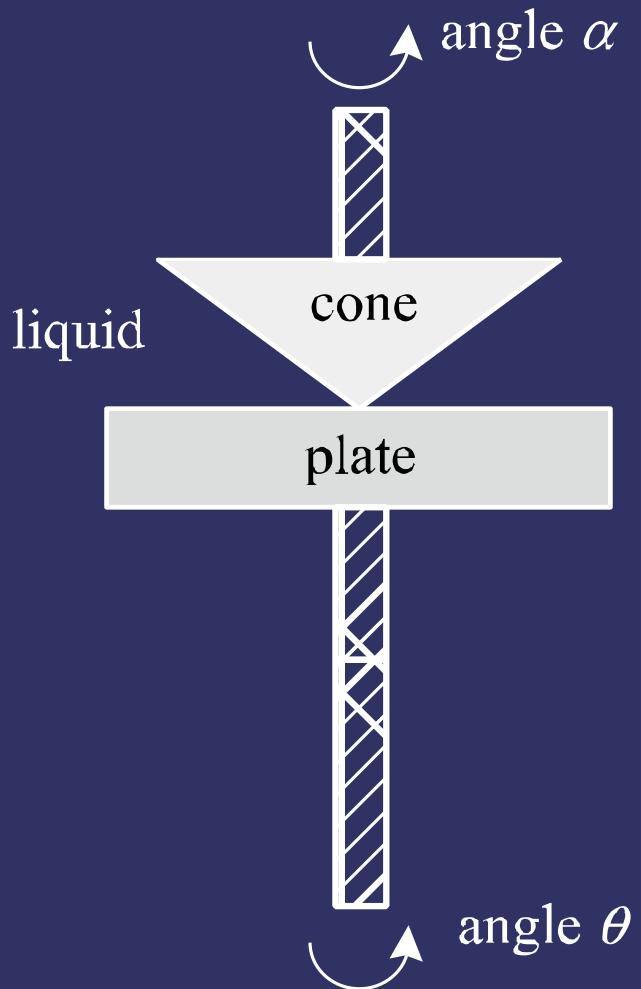


Figure 4.1 Schematic diagram of a cone-and-plate viscometer with a 2D view.

The mathematical model of the viscometer is described as follows (Friedly, 1972):

$$I \frac{d^2\alpha}{dt^2} + K\alpha = \mu \left(\tilde{\theta} - \frac{d\alpha}{dt} \right)$$

where I represents the dimensionless moment of inertia, K is the torsion constant of the Hook's law, $\tilde{\theta}$ is the rate change of the angle θ , i.e., $\tilde{\theta}(t) = d\theta/dt$, μ is the dimensionless viscosity, and α is the deflection of the cone. If the model parameters are estimated to be $I = 2$, $K = 1$, and $\mu = 0.5$, and the forcing function imposed on the plate is given by $\tilde{\theta}(t) = 0.5\sin(5t)$ determine the dynamic deflection of the cone.

Ans:

Since this viscometer is static initially, the initial conditions are given by $\alpha(0) = 0$ and $\alpha'(0) = 0$.

$$x_1 = \alpha$$

$$x_2 = \alpha'$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{\mu}{I}(\tilde{\theta} - x_2) - \frac{K}{I}x_1$$

$$x_1(0) = 0$$

$$x_2(0) = 0.$$

Ans:

ex4_2_1.m

```
function ex4_2_1
%
% Example 4-2-1 Dynamics of a cone-and-plate viscometer
%
clear; close all;
clc
%
global mu I K
%
% model parameters
%
mu=0.5; % dimensionless viscosity
K=1; % torsion constant
I=2; % dimensionless moment of inertia
%
% solving with ode45
%
```

Ans:

ex4_2_1.m

```
[t, x]=ode45(@ex4_2_1f, [0 10], [0 0]);
%
% Results plotting
%
alpha=x(:, 1); % cone deflection angle
d_alpha=x(:, 2); % rate change of the deflection angle
figure(1)
plot(t, alpha)
xlabel('time')
ylabel('deflection angle, \alpha')
figure(2)
plot(t, d_alpha)
xlabel('time')
ylabel('rate change of the deflection angle, d\alpha /dt')
%
% Dynamic equations
```

Ans:

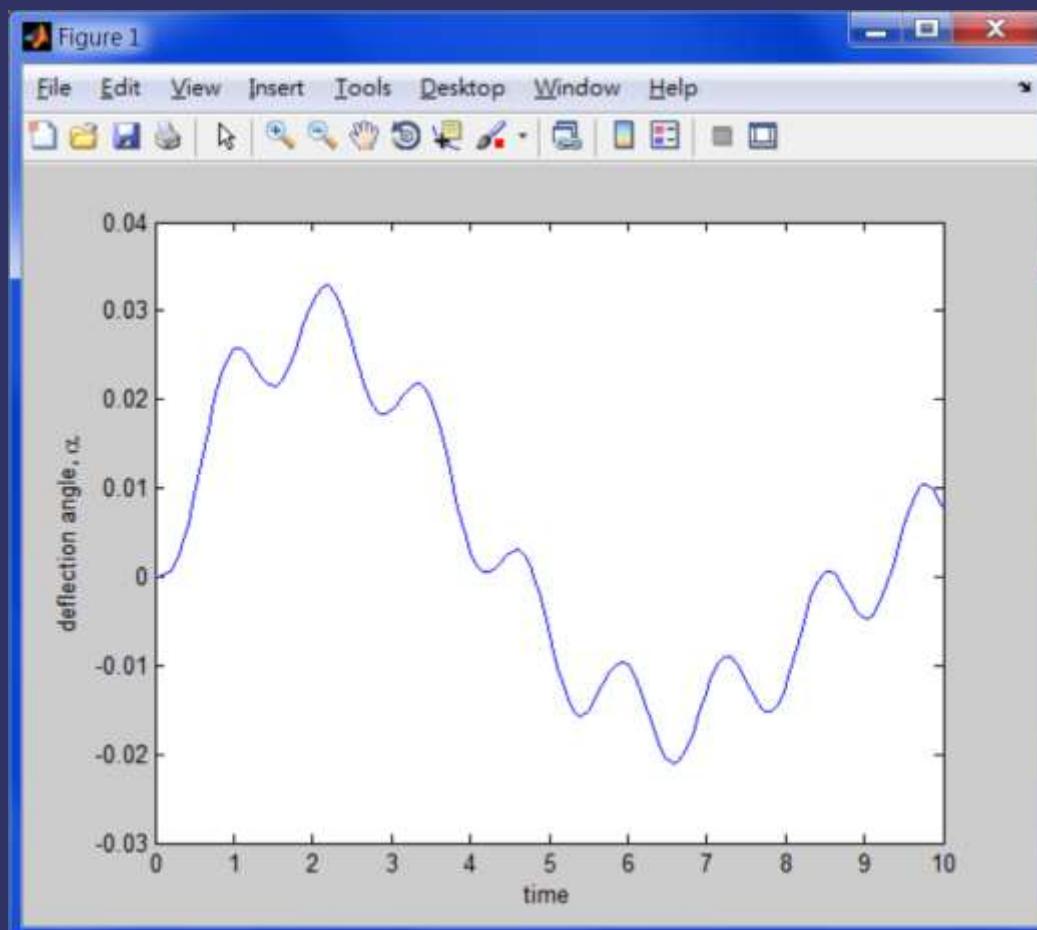
ex4_2_1.m

```
%  
function xdot=ex4_2_1f(t, x)  
global mu I K  
xdot=[x(2)  
mu*(0.5*sin(5*t)-x(2))/I-K*x(1)/I];
```

Ans:

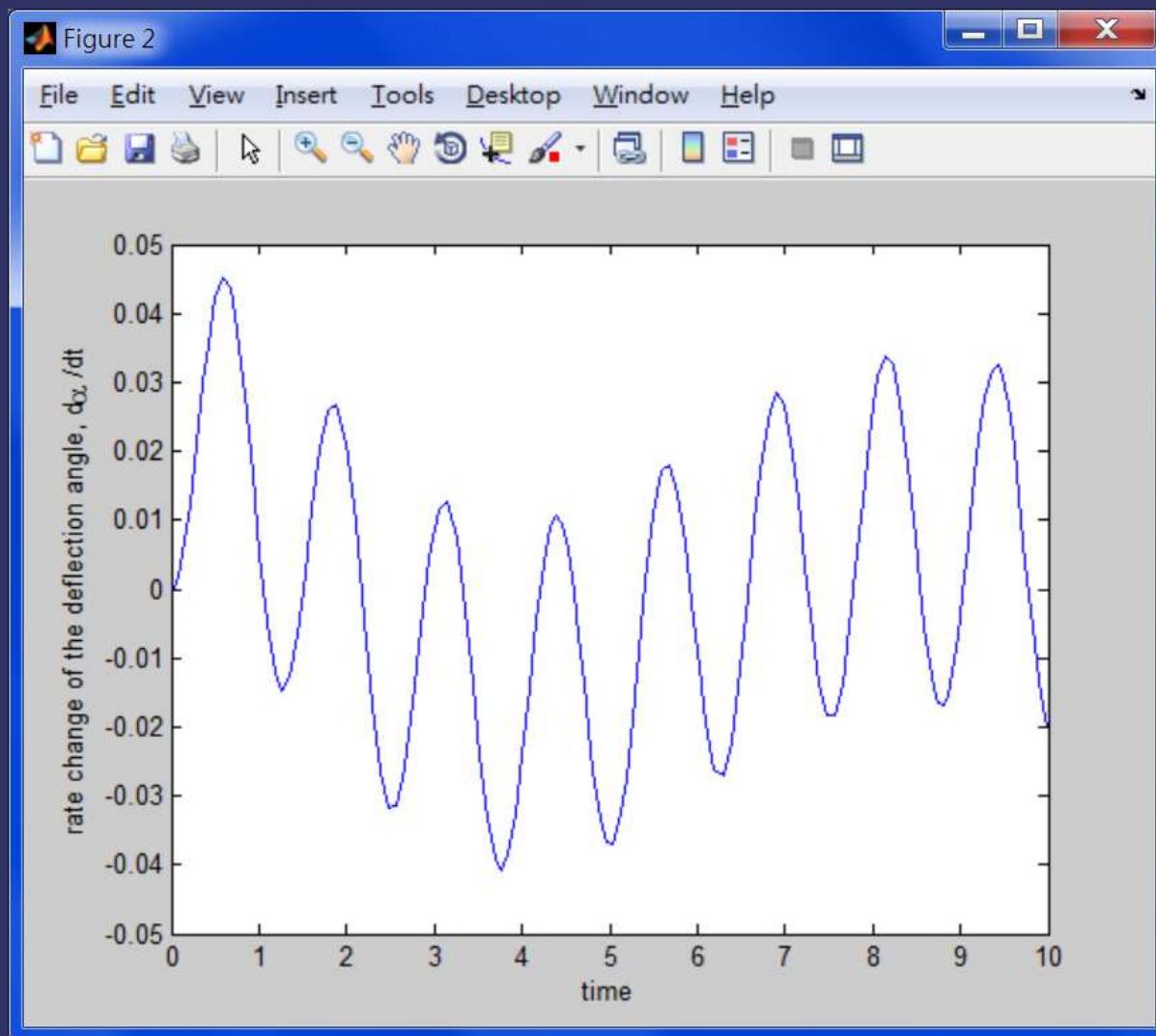
Execution results:

```
>> ex4_2_1
```



Numerical Solution of Ordinary Differential Equations

04



4.3 Stiff differential equations

$$\frac{dx_1}{dt} = f_1(t, x_1, x_2, \dots, x_n)$$

$$\frac{dx_2}{dt} = f_2(t, x_1, x_2, \dots, x_n)$$

$$\vdots$$

$$\frac{dx_n}{dt} = f_n(t, x_1, x_2, \dots, x_n)$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

- Let $\lambda_i = 1, 2, \dots, n$, be the eigenvalues of the Jacobian matrix; then, the *stiffness ratio* (SR) of the system is defined by

$$\text{SR} = \frac{\max_{1 \leq i \leq n} |\text{Re}(\lambda_i)|}{\min_{1 \leq i \leq n} |\text{Re}(\lambda_i)|}$$

- example

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1,000 & -1,001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ -1,000 & -1,001 \end{bmatrix}$$

- The eigenvalues of this Jacobian matrix are calculated to be -1 and -1,000.
- SR is 1,000,
- $x_1(0) = 1$
- $x_2(0) = 0$

$$x_1(t) = \frac{1}{999}(1,000e^{-t} - e^{-1,000t})$$

$$x_2(t) = \frac{-1,000}{999}(e^{-t} - e^{-1,000t})$$

Example 4-3-1

Dynamic simulation of a stiff ODE system

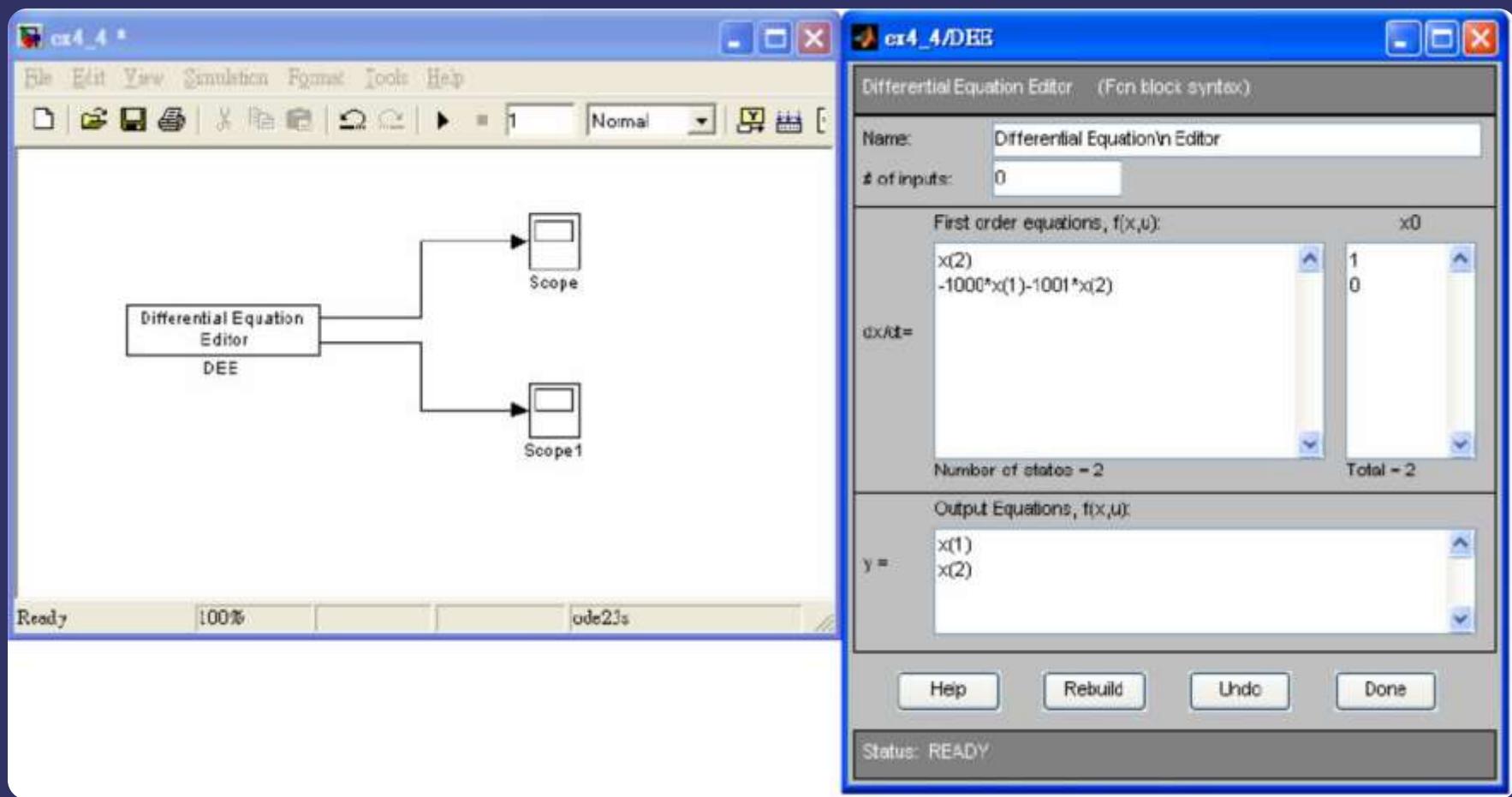
Solve the ODE system (4.3-4) with the nonstiff ODE solver ode23 and the stiff ODE solver ode23s and compare their solution stabilities. Note that the initial values are $x_1(0) = 1$ and $x_2(0) = 0$.

Ans:

The comparison will be performed using MATLAB Simulink.

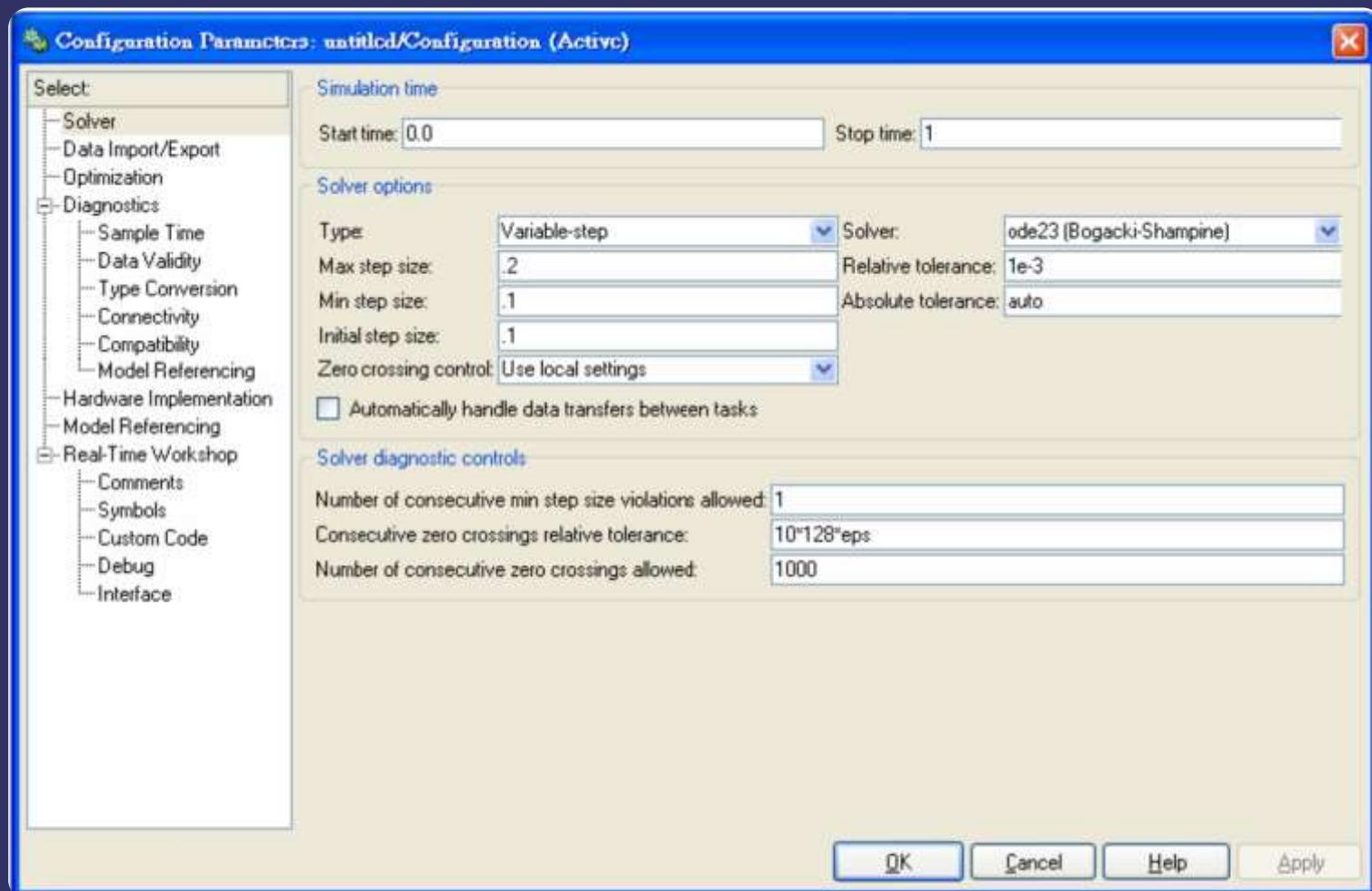
Ans:

1.



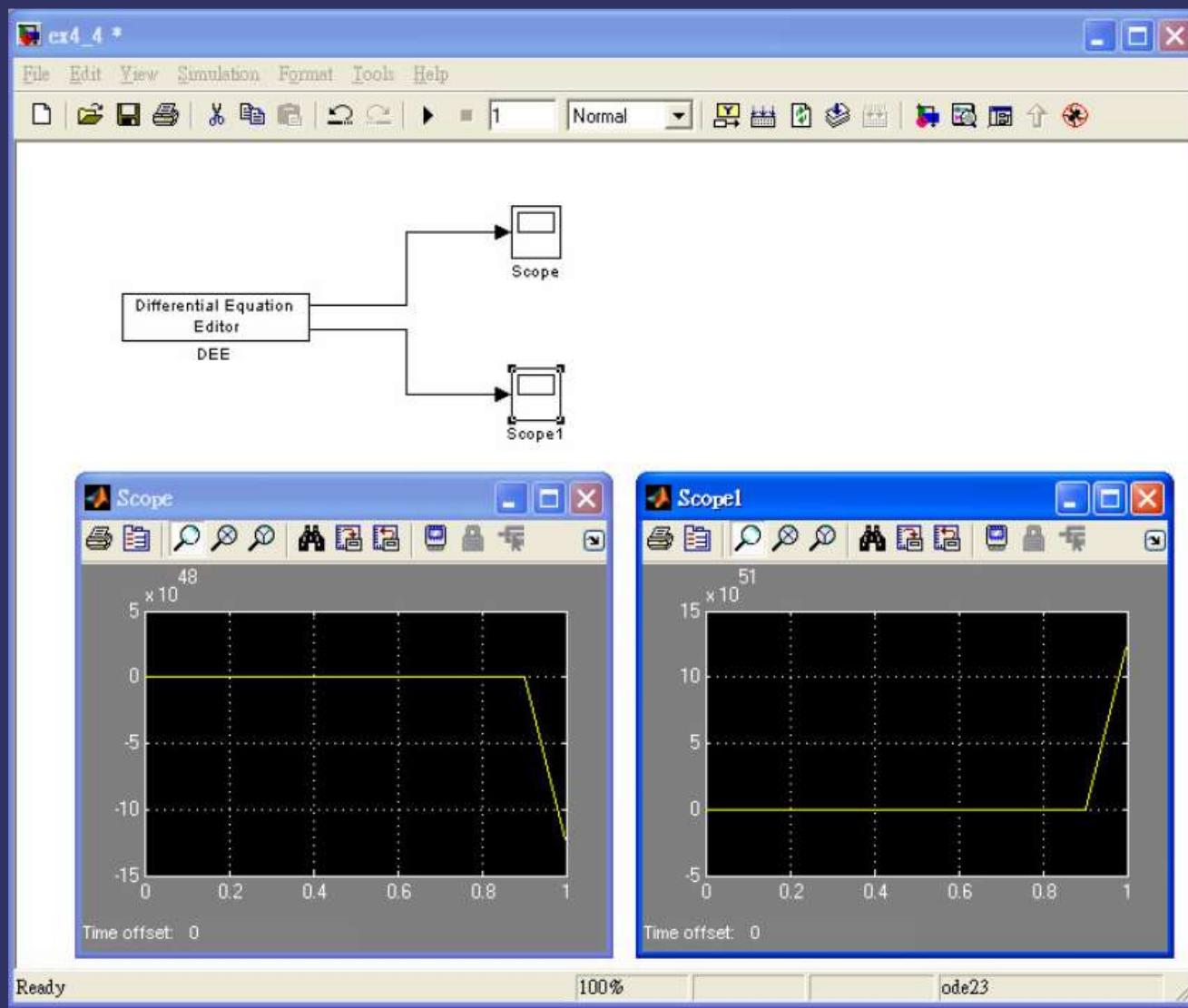
Ans:

2. Select the nonstiff ODE solver ode23 to start the simulation.



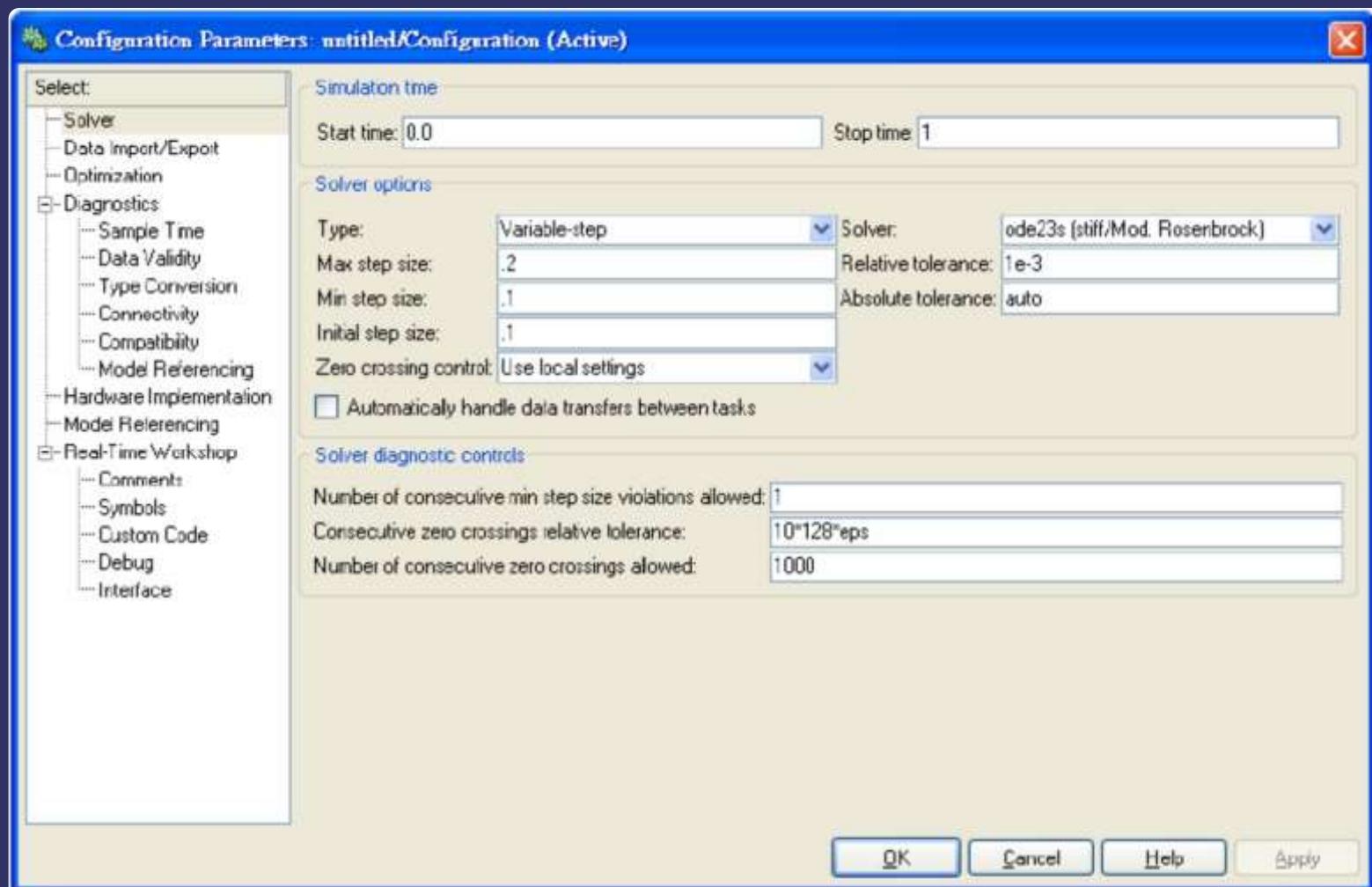
Ans:

2.



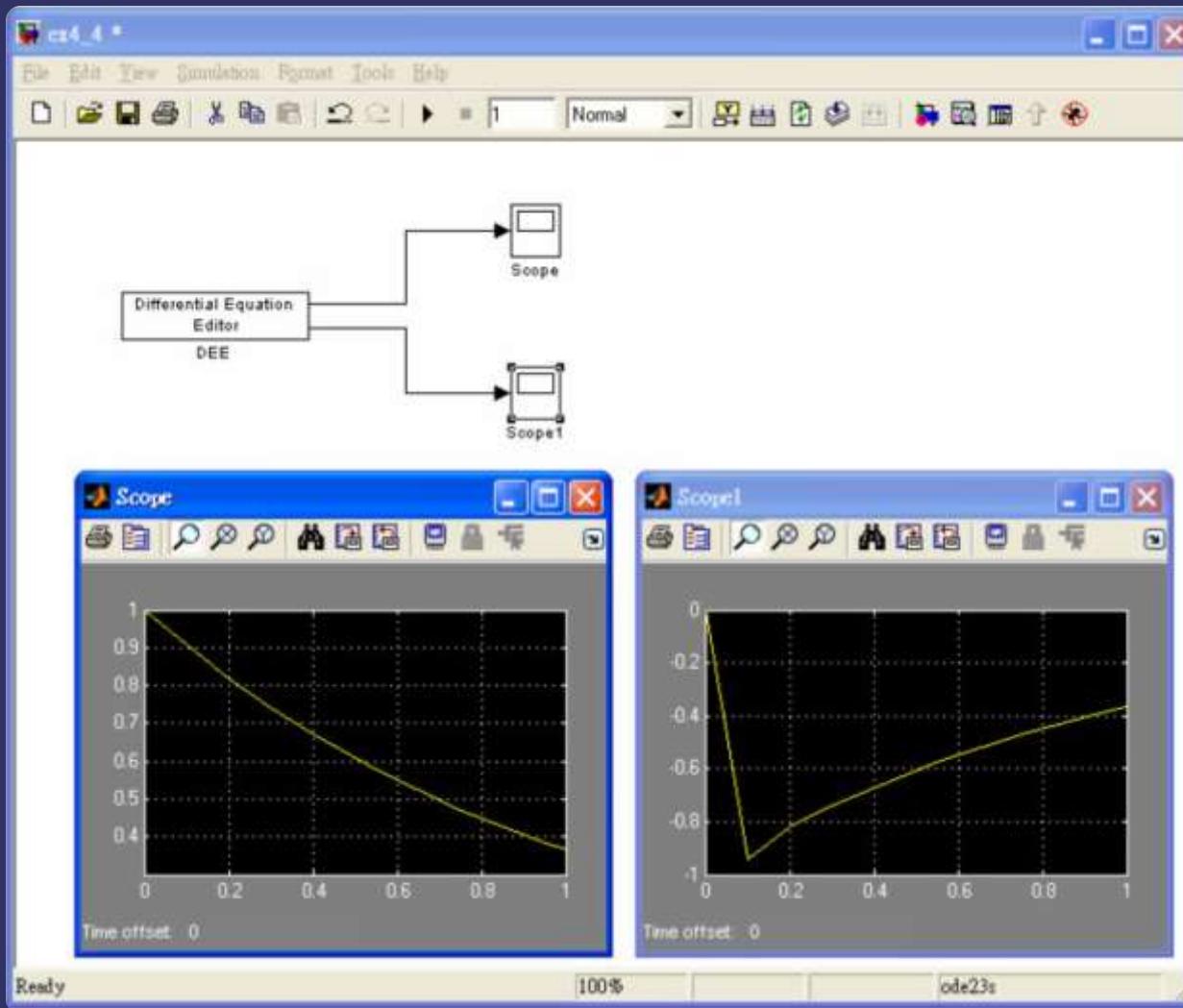
Ans:

3. Select ode23s to restart the simulation.



Ans:

3.



4.4 Differential-algebraic equation system

$$\dot{y}_1 = f_1(x, y_1, y_2, \dots, y_N)$$

$$\dot{y}_2 = f_2(x, y_1, y_2, \dots, y_N)$$

 \vdots

$$\dot{y}_n = f_n(x, y_1, y_2, \dots, y_N)$$

$$0 = f_{n+1}(x, y_1, y_2, \dots, y_N)$$

$$0 = f_{n+2}(x, y_1, y_2, \dots, y_N)$$

 \vdots

$$0 = f_{n+m}(x, y_1, y_2, \dots, y_N)$$

$$\mathbf{M}\dot{\mathbf{y}} = \mathbf{F}(x, \mathbf{y})$$

$$\mathbf{M} = \begin{bmatrix} I_{n \times n} & 0 \\ 0 & 0 \end{bmatrix}$$

- MATLAB stiff ODE solvers such as ode15s or ode23t can be used.

Example 4-4-1

Solution of a DAE system

Solve the following DAE system:

$$\frac{dy_1}{dt} = -0.04y_1 + 10^4 y_2 y_3$$

$$\frac{dy_2}{dt} = 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2$$

$$0 = y_1 + y_2 + y_3 - 1$$

where the initial conditions are given by $y_1(0) = 1$, $y_2(0) = 0$, and $y_3(0) = 0$.

Ans:

ex4_4_1.m

```
function ex4_4_1
%
% Example 4-4-1 Solution of a DAE system
%
clear; close all;
clc
%
% Set the mass matrix of the DAE
%
M=[1 0 0
    0 1 0
    0 0 0]; % two differential eqs and one algebraic eq.
%
% Initial values and the simulation time
%
y0=[0.5 0 0.5]';
tspan=[0 logspace(-6,6)];
```

Ans:

ex4_4_1.m

```
%  
% Solution parameter settings for the DAE  
%  
options=odeset('Mass', M, 'MassSingular', 'yes',...  
'RelTol', 1e-4, 'AbsTol', [1e-6 1e-10 1e-6]);  
%  
% Solving the DAE with ode15s  
%  
[t, y]=ode15s(@ex4_5f, tspan, y0, options);  
%  
% Results output and plotting  
%  
subplot(3, 1, 1)  
semilogx(t, y(:,1))  
ylabel('y_1')  
subplot(3, 1, 2)  
semilogx(t, y(:,2))
```

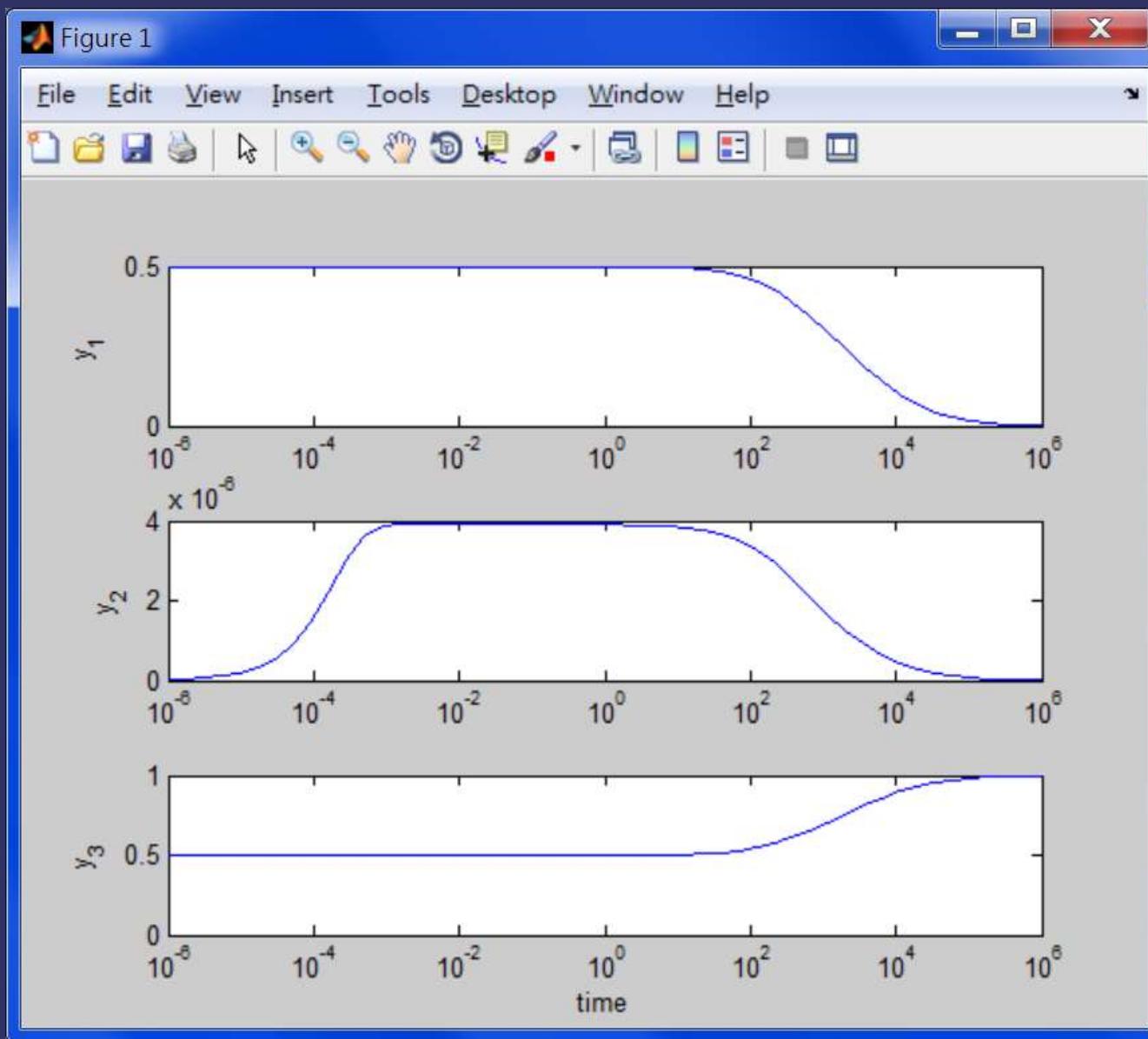
Ans:

ex4_4_1.m

```
ylabel('y_2')
subplot(3, 1, 3)
semilogx(t, y(:,3))
xlabel('time')
ylabel('y_3')
%
% DAE equations
%
function out=ex4_5f(t, y)
out=[-0.04*y(1)+1e4*y(2)*y(3)
     0.04*y(1)-1e4*y(2)*y(3)-3e7*y(2)^2
     y(1)+y(2)+y(3)-1];
```

Numerical Solution of Ordinary Differential Equations

04



NOTES:

1. If the initial values fail to satisfy these the equality constraints, they become inconsistent and would cause *the jumping behavior* at the very instant when the system states are fast approaching some certain values that meet the equality conditions.
2.

```
options=odeset('Mass', M, 'MassSingular', 'yes', 'RelTol', 1e-4,...  
'AbsTol', [1e-6 1e-10 1e-6]);
```

Property	Value	Description
Mass	M	The mass matrix of the DAE
MassSingular	Yes	To indicate M is a singular matrix
RelTol	1e-4	The <i>relative error tolerance</i>
AbsTol	[1e-6 1e-10 1e-6]	The <i>absolute error tolerance</i>

4.5 Boundary-valued ordinary differential equations

► 4.5.1 Problem patterns

$$\dot{y}_1 = f_1(x, y_1, y_2, \dots, y_n)$$

$$\dot{y}_2 = f_2(x, y_1, y_2, \dots, y_n)$$

$$\vdots$$

$$\dot{y}_n = f_n(x, y_1, y_2, \dots, y_n)$$

$$y_i(x_0) = y_{i,0}, \quad i = 1, 2, \dots, m$$

$$y_i(x_f) = y_{i,f}, \quad i = m+1, m+2, \dots, n$$

- The two-point BVP in a *compact* form is described by

$$\mathbf{y}' = \mathbf{F}(x, \mathbf{y}), \quad x_0 \leq x \leq x_f$$

- where the boundary conditions are expressed in a vector form as follows:

$$\mathbf{g}(\mathbf{y}(x_0), \mathbf{y}(x_f)) = 0$$

- For example, a second-order BVP

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right), \quad x_0 \leq x \leq x_f$$

$$g_1(y(x_0), y(x_f)) = a_0 y(x_0) + b_0 y'(x_0) - c_0 = 0$$

$$g_2(y(x_0), y(x_f)) = a_f y(x_f) + b_f y'(x_f) - c_f = 0$$

4.5.2 MATLAB commands for solving two-point boundary value problems

- *Problem type I:*

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \quad a \leq x \leq b$$

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = 0$$

- *Problem type II:*

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}, \mathbf{p}), \quad a \leq x \leq b$$

- where \mathbf{p} is the parameter vector to be determined.

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b), \mathbf{p}) = 0$$

`sol=bvp4c(odefun, bcfun, solinit)`

Arguments	Description
sol	<p>The solution output, which is with the following structure:</p> <p>sol.x mesh position (collocation) of the independent variable</p> <p>sol.y the y value associated with the mesh sol.x</p> <p>sol.yp the value of \dot{y} on the mesh sol.x</p> <p>sol.parameters the value of the unknown parameter p</p> <p>NOTE: With the command deval, the solution at some other points in the interval [a b] can be obtained through the use of the solution structure sol; for example, $yi=deval(sol, xi)$ is used to interpolate the value yi at the position of xi through sol.</p>
odefun	<p>The ode function file should be provided in the format of</p> <pre>function dydx=odefun(x, y, p) dydx=[...];</pre> <p>NOTE: x is a scalar, y and dydx are the column vectors, and p, the system parameter to be determined, can be a vector or a scalar.</p>

Arguments

Description

bcfun

The boundary conditions file provided by users, which is of the format:

```
function res=bcfun(ya, yb, p)
res=[ ...];
```

NOTE: ya and yb, which respectively represent the value of the left boundary and the right boundary in the boundary conditions, are column vectors, res is the residue of the boundary conditions, and p the parameter to be solved along with the boundary conditions.

solinit

The initial guess values provided by the users. This can be done with the command bvpinit. The syntax is solinit=bvpinit(x, yinit, p), where x, yinit and p are, respectively, the initial guess values for the independent variable, system states and the system parameter. For detailed usages, refer to the illustrative examples.

Example 4-5-1

Solution of a BVP with one decision parameter

Solve the following BVP of which the ODE is given by

$$y'' + (\lambda - 5 \cos(2x)) y = 0, \quad 0 \leq x \leq \pi \quad \square$$

associated with the boundary conditions: $y' = 0$, $y'(\pi) = 0$, and $y'(0) = 1$, where λ is the system parameter to be determined.

Ans:**Step 1:** Let $y_1 = y$ and $y_2 = y'$

$$y'_1 = y_2$$

$$y'_2 = (5\cos(2x) - \lambda)y_1$$

$$y_2(0) = 0, \quad y_2(\pi) = 0, \quad \text{and} \quad y_1(0) = 1.$$

Step 2: Prepare the ODE function file

```
function dydx = ex4_5_1ode(x, y, lambda)
```

```
dydx = [y(2)
```

```
(5*cos(2*x)-lambda)*y(1)];
```

Ans:

Step 3: Prepare the boundary conditions file, which is given by

```
function res = ex4_5_1bc(ya, yb, lambda)
```

```
res = [ya(2)
```

```
yb(2)
```

```
ya(1)-1] ;
```

NOTE: In the above expressions, $ya(2)$ and $yb(2)$ represent the value of y_2 on the left and right boundary, respectively. Likewise, $ya(1)$ is the value of y_1 on the left boundary.

Ans:

Step 4: Create a file to assign the initial guess value

```
function yinit = ex4_5_1init(x)  
yinit=[cos(4*x)  
       -4*sin(4*x)];
```

unknown parameter,

```
lambda = 15; % guessed value
```

```
solinit = bvpinit(linspace(0, pi, 10), @ex4_5_1init, lambda);
```

Ans:

Step 4: Find solution with bvp4c

```
sol = bvp4c(@ex4_5_1ode, @ex4_5_1bc, solinit)
```

Step 6: Extract the parameter value that has been determined

```
lambda = sol.parameters % parameter value
```

ex4_5_1.m

```
function ex4_5_1
%
% Example 4-5-1 Solution of the BVP with one decision parameter
%
clear; close all;
clc
%
lambda=15; % Initial guess value of the decision parameter
%
% Initial guess values
%
solinit = bvpinit(linspace(0, pi, 10), @ex4_5_1init, lambda);
%
options=bvpset('stats', 'on', 'RelTol', 1e-4, 'AbsTol', [1e-5 1e-6]);
%
% Solving with bvp4c
%
sol = bvp4c(@ex4_5_1ode, @ex4_5_1bc, solinit, options);
```

ex4_5_1.m

```
% Results output
%
lambda=sol.parameters;
fprintf('\n The estimated value of lambda =%7.3f.\n', lambda)
%
% Solution plotting
%
x = linspace(0, pi);
Y = deval(sol, x);
plot(x, Y(1,:), x, Y(2,:), '--')
legend('y', 'y'' ')
axis([0 pi -4 4])
xlabel('x')
ylabel('y and y'' ')
legend('y', 'y'' ')
%
% Differential equations
%
function dydx = ex4_5_1ode(x, y, lambda)
```

 ex4_5_1.m

```
dydx = [ y(2)
          (5*cos(2*x)-lambda)*y(1) ];
%
% Boundary conditions
%
function res = ex4_5_1bc(ya, yb, lambda)
res = [ya(2)
       yb(2)
       ya(1) - 1];
%
% Initial guess values of the states
%
function v = ex4_5_1init(x)
v = [ cos(4*x)
      -4*sin(4*x) ];
```

Execution results:

```
>> ex4_5_1
```

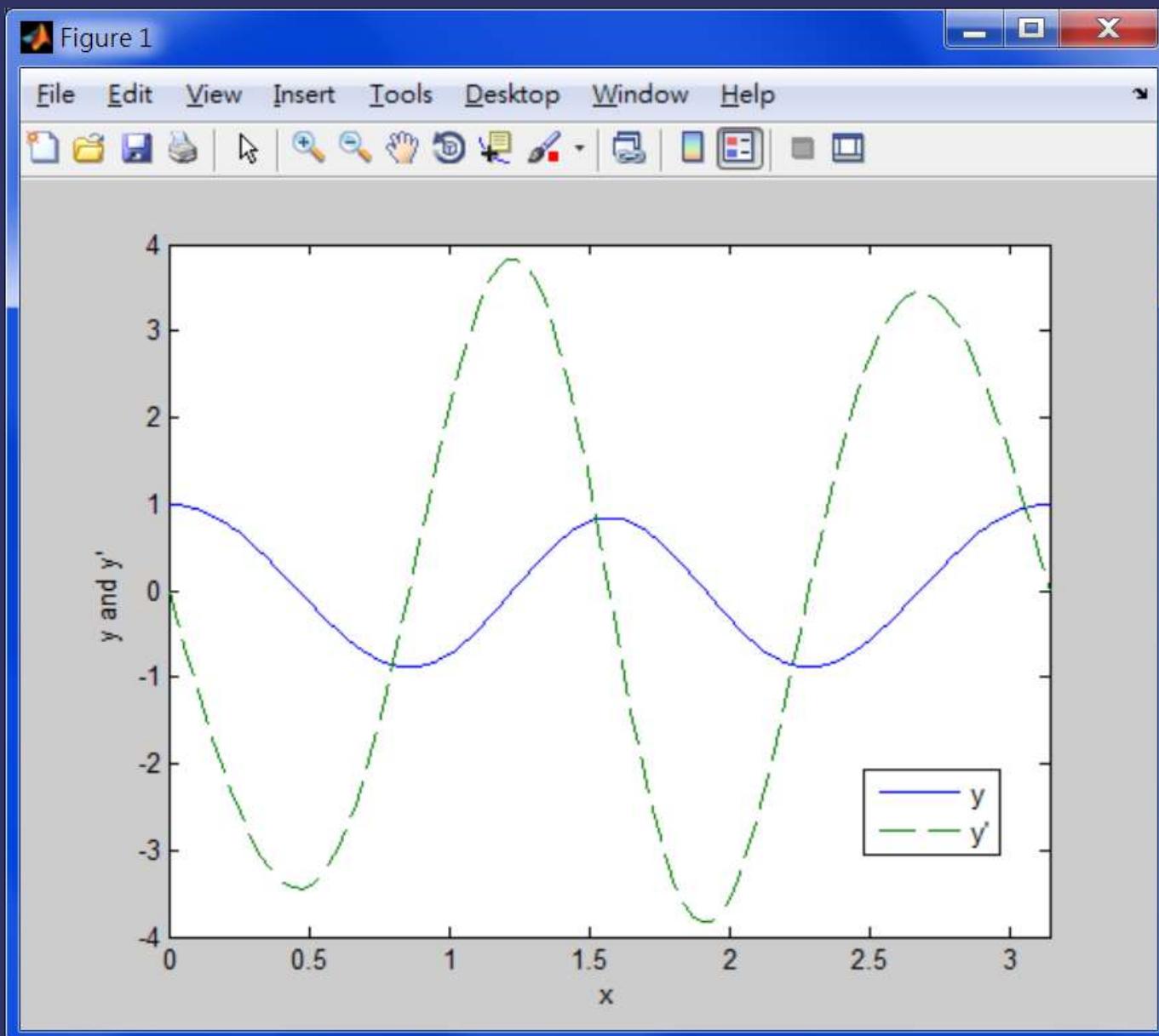
The solution was obtained on a mesh of 73 points.

The maximum residual is 8.767e-005.

There were 2,641 calls to the ODE function.

There were 72 calls to the BC function.

The estimated value of lambda = 16.227.



Example 4-5-2

A BVP with an unknown period

Solve the following BVP whose ODE system equations are described by (*Seydel, 1988*)

$$\dot{y}_1 = 3 \left(y_1 + y_2 - \frac{1}{3} y_1^3 - 1 \right)$$

$$\dot{y}_2 = -\frac{1}{3} \left(y_1 + 0.5 y_2 - 1 \right)$$

and the boundary conditions are given as follows: $y_1(0) = y_1(T)$, $y_2(0) = y_2(T)$, and $\dot{y}_2(0) = 1/T$ where the period T needs to be determined.

Ans:

$$\tau = t/T$$

$$\frac{dy_1}{d\tau} = 3T \left(y_1 + y_2 - \frac{1}{3} y_1^3 - 1 \right)$$

$$\frac{dy_2}{d\tau} = \frac{-T(y_1 + 0.5y_2 - 1)}{3}$$

$$\tau \in [0, 1],$$

$$y_1(0) = y_1(1), \quad y_2(0) = y_2(1) \quad \text{and} \quad \frac{dy_2}{d\tau}(0) = 1$$

 ex4_5_2.m

```
function ex4_5_2
%
% Example 4-5-2 BVP with an unknown period to be determined
%
clear; close all;
clc
%
% Initial guess values
%
T=2*pi;
solinit = bvpinit(linspace(0,1,5), @ex4_5_2init, T);
%
options = bvpset('stats', 'on', 'RelTol', 1e-4, 'AbsTol', [1e-5 1e-6]);
%
% Solving with bvp4c
%
sol = bvp4c(@ex4_5_2ode, @ex4_5_2bc, solinit, options);
%
% Results output
```

 ex4_5_2.m

```
%  
T = sol.parameters;  
fprintf('\n The estimated period T=%7.3f.\n', T)  
t=T*sol.x;  
%  
% Results plotting  
%  
figure(1)  
plot(t, sol.y(1,:), T*solinit.x, solinit.y(1,:), 'o')  
legend('solutions', 'initial guess values');  
axis([0 T -2.2 3]);  
ylabel('solution y_1(t)');  
xlabel('t (time)');  
shg  
%  
figure(2)  
plot(t, sol.y(2,:), T*solinit.x, solinit.y(2,:), 'o')  
legend('solutions', 'initial guess values');  
axis([0 T -1.5 3]);
```

ex4_5_2.m

```
ylabel('solution y_2(t)');
xlabel('t (time)');
%
% Differential equations
%
function dydt = ex4_5_2ode(t, y, T);
dydt = [ 3*T*(y(1) + y(2) - 1/3*y(1)^3 - 1);
          -1/3*T*(y(1) + 0.5*y(2)- 1) ];
%
% initial guesses of system states
%
function v = ex4_5_2init(x)
v = [ sin(2*pi*x)
      cos(2*pi*x)];
%
% Boundary conditions
%
function res = ex4_5_2bc(ya, yb, T)
```

ex4_5_2.m

```
res = [ya(1) - yb(1)
       ya(2) - yb(2)
       -1/3*T *(ya(1) + 0.5*ya(2) - 1) - 1];
```

Execution results:

```
>> ex4_5_2
```

The solution was obtained on a mesh of 76 points.

The maximum residual is 9.410e-005.

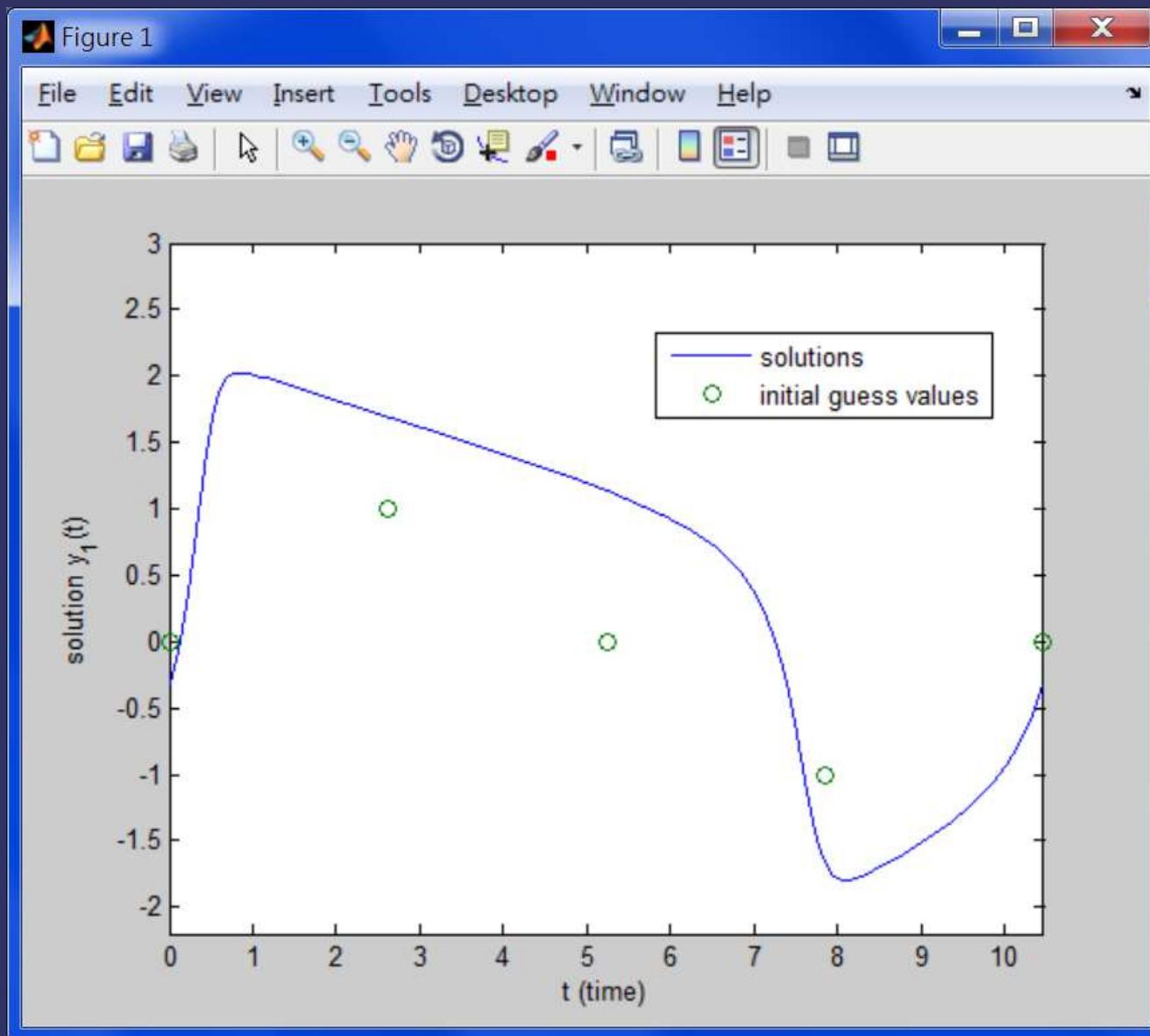
There were 3,267 calls to the ODE function.

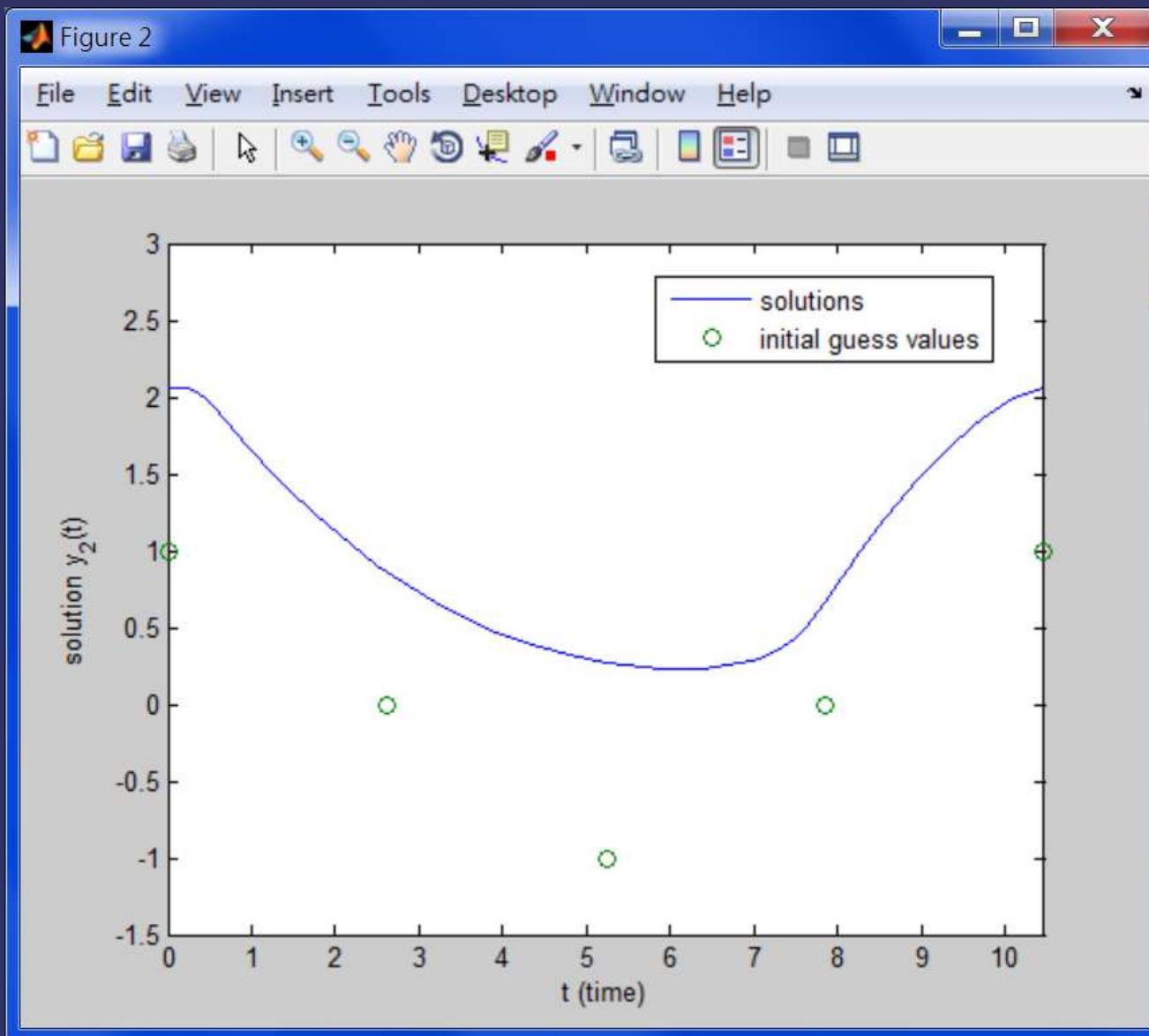
There were 111 calls to the BC function.

The estimated period $T = 10.460$.

Numerical Solution of Ordinary Differential Equations

04





4.5.3 Multipoint BVP

ODE system model: $a \leq x \leq b$

$$\mathbf{y}' = f_i(x, \mathbf{y}, \mathbf{p}), \quad s_{i-1} \leq x \leq s_i, \quad i = 1, 2, \dots, N + 1$$

where S_i is the interior point satisfying

$$a \equiv s_0 < s_1 < s_2 < \dots < s_N < s_{N+1} \equiv b.$$

Constraints:

- (i) Two-point boundary value conditions specified at the two endpoints:

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = 0,$$

- (ii) Constraints imposed on internal points:

$$h_i(s_i, \mathbf{y}(s_i)) = 0, \quad i = 1, 2, \dots, N$$

$$t = \begin{cases} \frac{x - s_{i-1}}{s_i - s_{i-1}}, & s_{i-1} \leq x \leq s_i, \quad i \text{ is an odd number} \\ \frac{s_i - x}{s_i - s_{i-1}}, & s_{i-1} \leq x \leq s_i, \quad i \text{ is an even number} \end{cases}$$

- let $y_i(t)$ denote the solution when the independent variable x lies in the interval $s_{i-1} \leq x \leq s_i$ two-point BVP:
- ODE equations: $0 \leq t \leq 1$

$$\mathbf{y}'_i(t) = (s_i - s_{i-1})\mathbf{f}_i(s_{i-1} + (s_i - s_{i-1})t, \mathbf{y}_i(t), \mathbf{p}),$$

$i \text{ is an odd number, } 1 \leq i \leq N + 1$

$$\mathbf{y}'_i(t) = (s_{i-1} - s_i)\mathbf{f}_i(s_i + (s_{i-1} - s_i)t, \mathbf{y}_i(t), \mathbf{p}),$$

$i \text{ is an even number, } 1 \leq i \leq N + 1$

$$s'_i = 0, \quad i = 1, 2, \dots, N$$

- Two-point boundary value conditions:

$$g(\cdot) = \begin{cases} g(y_1(0), y_{N+1}(0)) = 0, & N \text{ is an odd number} \\ g(y_1(0), y_{N+1}(1)) = 0, & N \text{ is an even number} \end{cases}$$

$$h_i(s_i, y_i(1)) = 0, \quad i \text{ is an odd number}, \quad 1 \leq i \leq N$$

$$h_i(s_i, y_i(0)) = 0, \quad i \text{ is an even number}, \quad 1 \leq i \leq N$$

- Continuity of states:

$$\mathbf{y}_i(1) = \mathbf{y}_{i+1}(1), \quad i \text{ is an odd number}, \quad 1 \leq i \leq N$$

$$\mathbf{y}_i(0) = \mathbf{y}_{i+1}(0), \quad i \text{ is an even number}, \quad 1 \leq i \leq N$$

Example 4-5-3

The solution of a multipoint BVP with bvp4c

Solve the following multipoint BVP (*Shampine et al.*, 2000):

System model:

(i) When $0 \leq x \leq 1$,

$$\frac{dv}{dx} = \frac{(c-1)}{n}$$

$$\frac{dc}{dx} = \frac{(vc - x)}{\eta}$$



(ii) When $1 \leq x \leq \lambda$,

$$\frac{dv}{dx} = \frac{(c-1)}{n}$$

$$\frac{dc}{dx} = \frac{(vc-1)}{\eta}$$

Boundary conditions:

$$v(0) = 0$$

$$c(\lambda) = 1$$

Ans:

$$y_1(x) = v(x) \quad y_2(x) = c(x) \quad 0 \leq x \leq 1$$

$$y_3(x) = v(x) \quad y_4(x) = c(x) \quad 1 \leq x \leq \lambda$$

ODE model:

(i) When $0 \leq x \leq 1$,

$$\frac{dy_1}{dx} = \frac{y_2 - 1}{n}$$

$$\frac{dy_2}{dx} = \frac{y_1 y_2 - x}{\eta}$$

Ans:

(ii) When $1 \leq x \leq \lambda$

$$\frac{dy_3}{dx} = \frac{y_4 - 1}{n}$$

$$\frac{dy_4}{dx} = \frac{y_3 y_4 - 1}{\eta}$$

Ans:

(i) The original boundary conditions

$$y_1(0) = 0$$

$$y_4(\lambda) = 1$$

(ii) Continuity of states

$$y_1(1) = y_3(1)$$

$$y_2(0) = y_4(1)$$

$$0 \leq \tau \leq 1 \quad 1 \leq x \leq \lambda$$

Ans:

$$\frac{dy_3}{d\tau} = \frac{(\lambda - 1)(y_4 - 1)}{n}$$

$$\frac{dy_4}{d\tau} = \frac{(\lambda - 1)(y_3 y_4 - 1)}{\eta}$$

two-point BVP

ODE model: $0 \leq t \leq 1$

$$\frac{dy_1}{dt} = \frac{y_2 - 1}{n}$$

$$\frac{dy_2}{dt} = \frac{y_1 y_2 - t}{\eta}$$

Ans:

$$\frac{dy_3}{dt} = \frac{(\lambda - 1)(y_4 - 1)}{n}$$

$$\frac{dy_4}{dt} = \frac{(\lambda - 1)(y_3 y_4 - 1)}{\eta}$$

Boundary conditions:

$$y_1(0) = 0$$

$$y_4(1) = 1$$

$$y_1(1) = y_3(0)$$

$$y_2(0) = y_4(0)$$

ex4_5_3.m

```
function ex4_5_3
%
% Example 4-5-3 Solution of a multipoint BVP
%
clear; close all;
clc
%
% Given data
%
n = 5e-2;
lambda = 2;
eta = 3.2;
%
options = [ ]; % No options specified
%
% Initial guess value
%
sol = bvpinit(linspace(0, 1, 5), [1 1 1 1]);
%
```

ex4_5_3.m

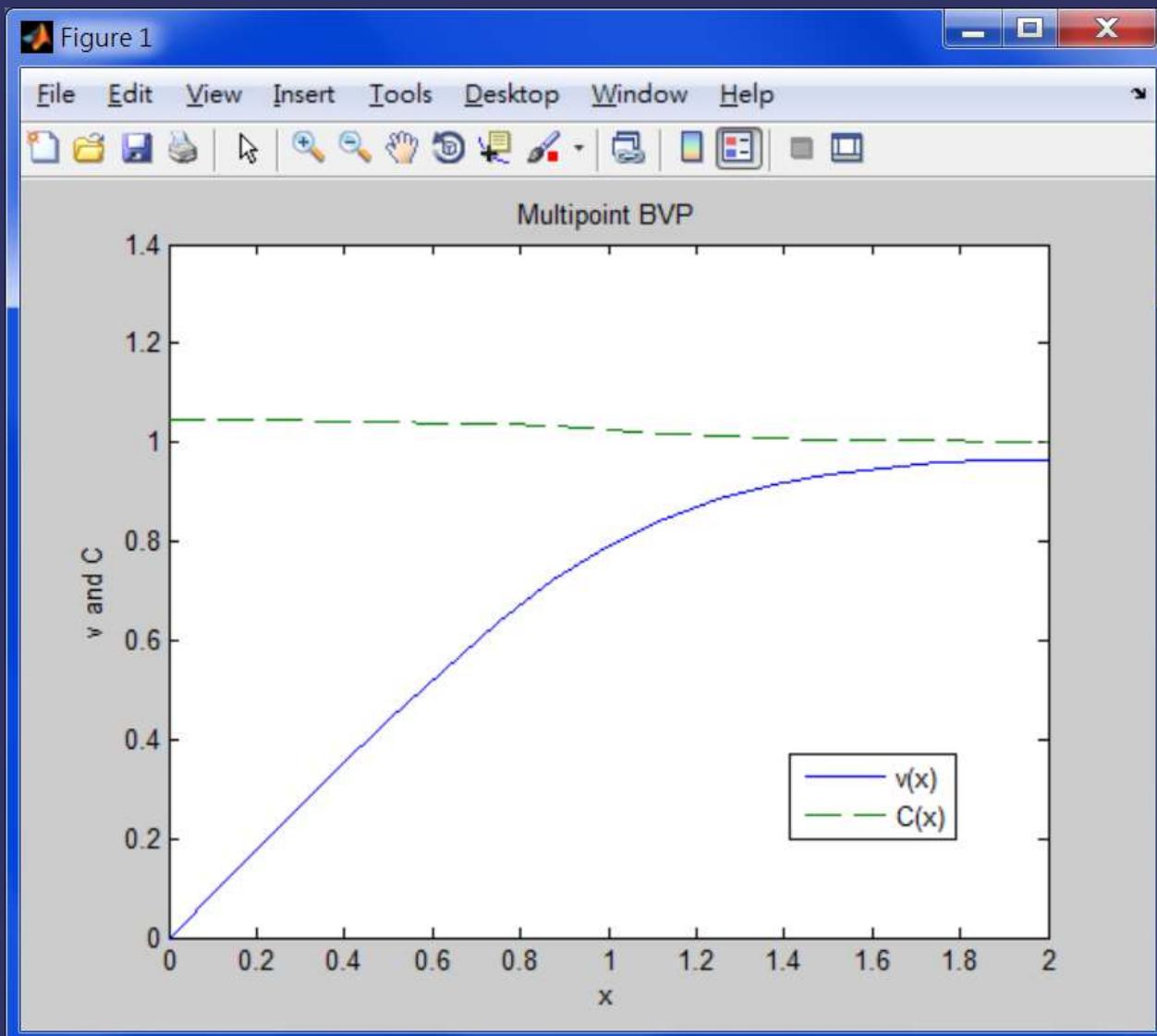
```
% Solving with bvp4c
%
sol = bvp4c(@ex4_5_3ode, @ex4_5_3bc, sol, options, n, lambda, eta);
%
x = [sol.x  sol.x*(lambda-1)+1];
y = [sol.y(1:2,:); sol.y(3:4,:)];
%
% Results plotting
%
plot(x, y(1,:), x, y(2,:), '--')
legend('v(x)', 'C(x)')
title('Multiple point BVP')
xlabel('x')
ylabel('v and C')
shg
%
% Differential equations
%
```

 ex4_5_3.m

```
function dydt = ex4_5_3ode(t, y, n, lambda, eta)
dydt = [ (y(2) - 1)/n
          (y(1)*y(2) - t)/eta
          (lambda - 1)*(y(4) - 1)/n
          (lambda - 1)*(y(3)*y(4) - 1)/eta ];
%
% Boundary conditions
%
function res = ex4_5_3bc(ya, yb, n, lambda, eta)
res = [ ya(1)
        yb(4) - 1
        yb(1) - ya(3)
        yb(2) - ya(4)];
```

Execution results:

```
>> ex4_5_3
```

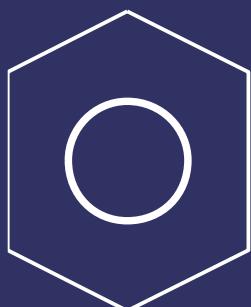


4.6 Chemical engineering examples

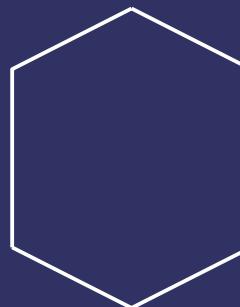
Example 4-6-1

Temperature and conversion distributions in a catalyzed tubular reactor

In a catalyzed tubular reactor, the following chemical reaction is taking place



Benzene



cyclohexane

$$r = \frac{k K_H^3 K_B P_H^3 P_B}{(1 + K_H P_H + K_B P_B + K_C P_C)^4}$$

$$\ln k = -12,100 / (RT) + 32.2 / R$$

$$\ln K_H = 15,500 / (RT) - 31.9 / R$$

$$\ln K_B = 11,200 / (RT) - 23.1 / R$$

$$\ln K_C = 8,900 / (RT) - 19.4 / R$$

Gas pressure, $P_t = 1.25$ atm

Diameter of the reaction tube, $R_0 = 2.5$ cm

Length of the tube, $L = 1$ m

Wall temperature, $T_w = 100^\circ\text{C}$

Wall temperature, $T_w = 100^\circ\text{C}$

Mass velocity, $G = 630 \text{ kg/m}^2 \cdot \text{hr}$

The ration of the molar flow rate of hydrogen to that of benzene,
 $m = 30$

Mole fraction of benzene at the inlet of the reaction tube, $y_0 = 0.0325$

Average molecular weight of reaction gas, $M_{av} = 4.45$

The density of the catalysis, $\rho_B = 1,200 \text{ kg/m}^3$

Average specific heat of the fluid, $C_P = 1.75 \text{ kcal/kg} \cdot {}^\circ\text{C}$

Heat of reaction, $\Delta H_r = -49,250 \text{ kcal/kg-mol}$

Overall heat transfer coefficient, $h_0 = 65.5 \text{ kcal/ m}^2 \cdot \text{hr} \cdot {}^\circ\text{C}$

The temperature of the feed, $T(0) = 125^\circ\text{C}$



Under the assumption that the temperature and reaction rate distributions in the radial direction are uniform, determine the temperature and conversion distributions along the axis of the tubular reactor.

Problem formulation and analysis:

Mass balance equation:

$$Gy_0 \frac{dx}{dL} = r\rho_B M_{av}$$

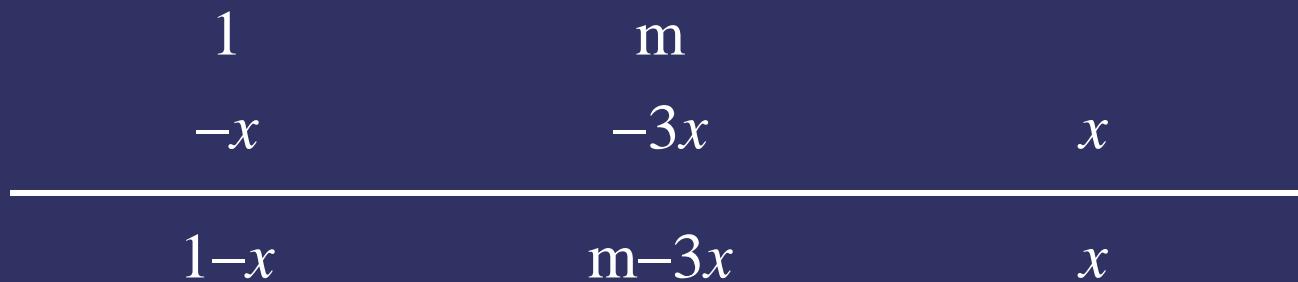
Energy balance equation:

$$GC_P \frac{dT}{dL} = \frac{-2h_0}{R_0}(T - T_w) + r\rho_B(-\Delta H_r)$$

$$\frac{dx}{dL} = \frac{\rho_B M_{av}}{Gy_0} r$$

Problem formulation and analysis:

$$\frac{dT}{dL} = \frac{-2h_0}{GC_P R_0} (T - T_w) + \frac{(-\Delta H_r) \rho_B}{GC_P} r$$





total mole number of the system is $1 + m - 3x$,

$$P_H = P_t \frac{m-3x}{1+m-3x}$$

$$P_B = P_t \frac{1-x}{1+m-3x}$$

$$P_C = P_t \frac{x}{1+m-3x}$$

MATLAB program design:

Let $\alpha = \frac{\rho_B M_{av}}{Gy_0}$, $\beta = \frac{-2h_0}{GC_P R_0}$, and $\mu = \frac{(-\Delta H_r) \rho_B}{GC_P}$

$$\frac{dx}{dL} = \alpha r$$

$$\frac{dT}{dL} = \beta(T - T_w) + \mu r$$

MATLAB program design:

— ex4_6_1.m —

```
function ex4_6_1
%
% Example 4-6-1 Temperature and conversion distributions
%
clear; close all;
clc;
%
global alpha beta mu Pt m Tw R
%
% Given data
%
Pt=1.25; % Gas pressure (atm)
Ro=0.025; % Diameter of the reaction tube (m)
rho_b=1200; % Density of the catalyst (kg/m^3)
Cp=1.75; % Average specific heat of the fluid (kcal/kg.oC)
Tw=100+273; % Wall temperature (K)
T0=125+273; % Feeding temperature (K)
```

MATLAB program design:

— ex4_6_1.m —

G=630; % Mass velocity rate (kg/m².hr)

m=30; % Mole flow ratio of hydrogen to benzene

Hr=-49250; % Reaction heat (kcal/kg-mol)

yo=0.0325; % Mole fraction of benzene at the inlet of the reaction tube

ho=65.5; % Overall heat transfer coefficient (kcal/m².hr.oC)

Mav=4.45; % Average molecular weight of the reaction gas

R=1.987; % Ideal gas constant (cal/mol.K)

%

```
alpha=rho_b*Mav/(G*yo);
```

```
beta=-2*ho/(G*Cp*Ro);
```

```
mu=-Hr*rho_b/(G*Cp);
```

%

```
x0=[0 T0]';
```

```
[L, X]=ode45(@ex4_6_1f, [0 1], x0);
```

%

% Results plotting

%

MATLAB program design:

— ex4_6_1.m —

```
figure(1)
plot(L, X(:,1))
xlabel('Reactor length (m)')
ylabel('Conversion rate')
axis([0 1 0 1.2])
%
figure(2)
plot(L, X(:,2))
xlabel('Reactor length (m)')
ylabel('Temperature (K)')
%
% System equations
%
function y=ex4_6_1f(L, f)
%
global alpha beta mu Pt m Tw R
%
```

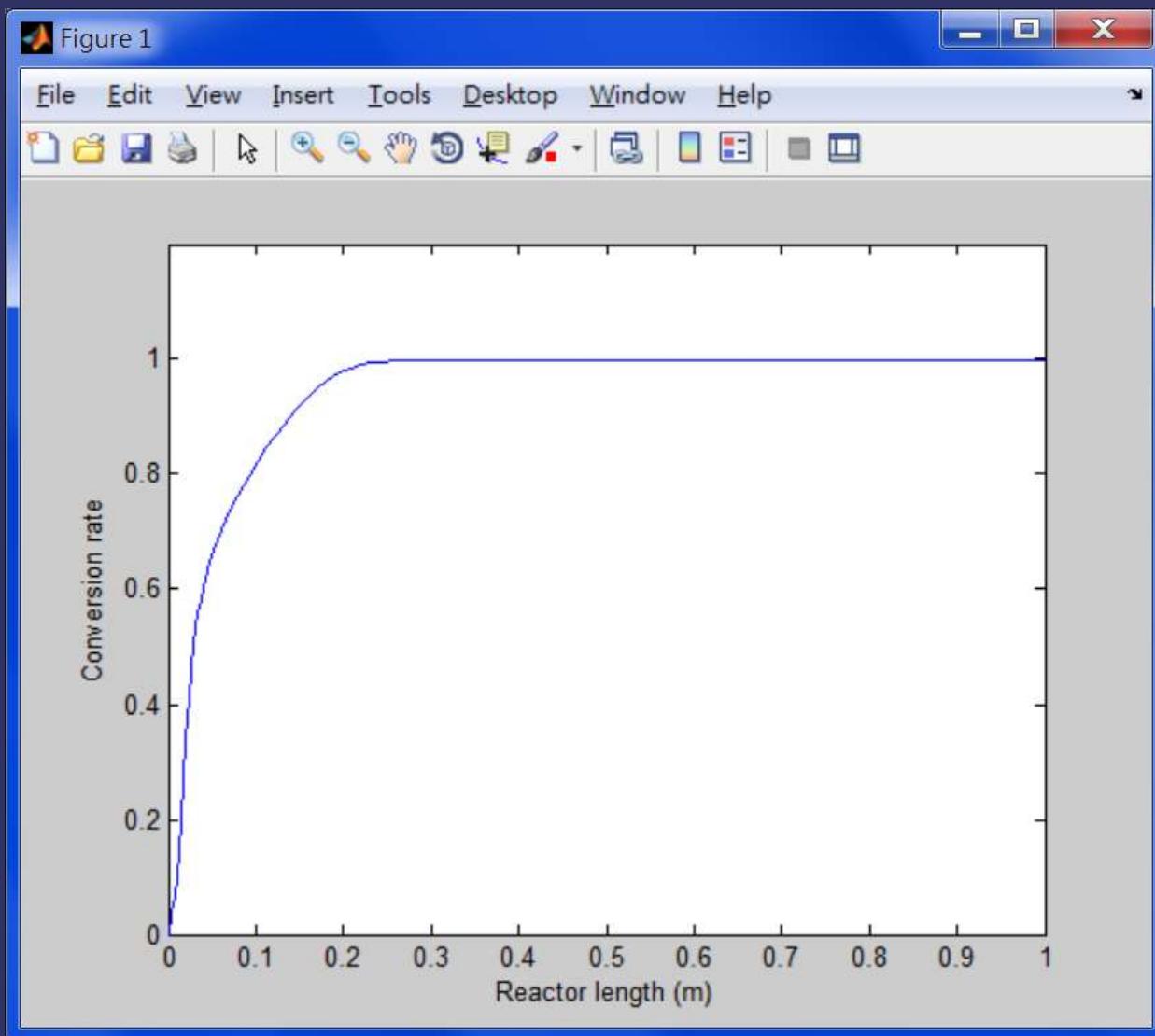
MATLAB program design:

— ex4_6_1.m —

```
x=f(1);
T=f(2);
%
k=exp(-12100/(R*T)+32.3/R);
Kh=exp(15500/(R*T)-31.9/R);
Kb=exp(11200/(R*T)-23.1/R);
Kc=exp(8900/(R*T)-19.4/R);
%
a=1+m-3*x;
ph=Pt*(m-3*x)/a;
pb=Pt*(1-x)/a;
pc=Pt*x/a;
%
r=k*Kh^3*Kb*ph^3*pb/(1+Kh*ph+Kb*pb+Kc*pc)^4;
%
y=[alpha*r
  beta*(T-Tw)+mu*r];
```

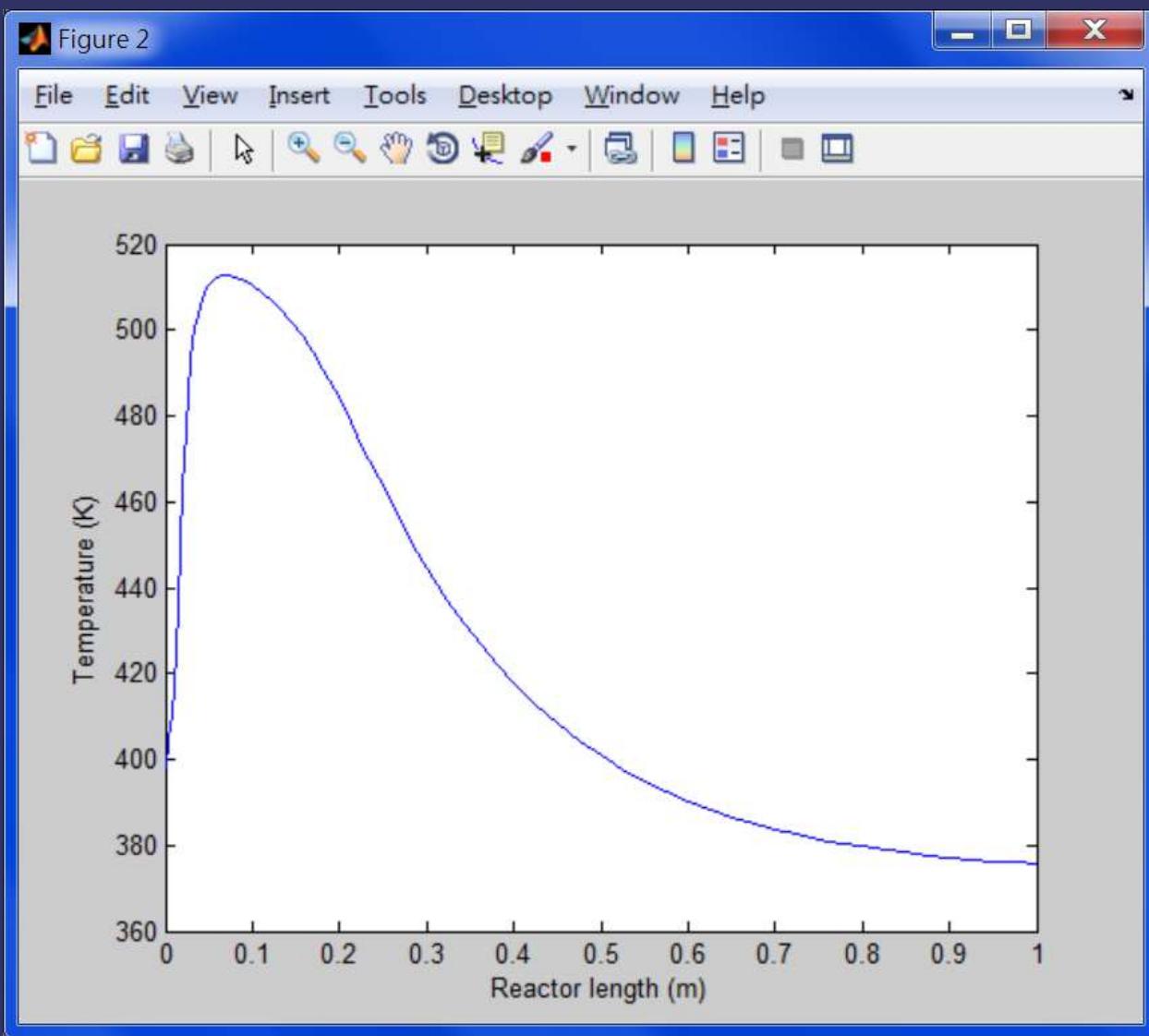
Execution results:

```
>> ex4_6_1
```



Execution results:

```
>> ex4_6_1
```



Example 4-6-2

Temperature and conversion distribution in a plug-flow reactor

In a *plug-flow* reactor (PFR), the following acetone cracking reaction occurs:



$$T_0 = 1,030 \text{ K},$$

$$P_0 = 165 \text{ kPa}.$$

$$T_a = 1,175 \text{ K}.$$

Volumetric flow of the feed (pure acetone), $V_0 = 0.002 \text{ m}^3/\text{s}$

Reactor volume, $V_R = 0.001 \text{ m}^3/\text{s}$

Overall heat transfer coefficient, $U = 110 \text{ W/m}^2 \cdot \text{K}$

Heat transfer area, $A = 150 \text{ m}^2/\text{m}^3$ reactor

Reaction rate constant, $k = 3.58 \exp\left[34,222\left(\frac{1}{T_0} - \frac{1}{T}\right)\right] \text{s}^{-1}$

$$\Delta H_R = 80,770 + 6.8(T - 298) - 5.75 \times 10^{-3}(T^2 - 298^2) - 1.27 \times 10^{-6}(T^3 - 298^3)$$

$$C_{PA} = 26.63 + 0.1830T - 45.86 \times 10^{-6}T^2 \quad \text{J/mol} \cdot \text{K}$$

$$C_{PB} = 20.04 + 0.0945T - 30.95 \times 10^{-6}T^2 \quad \text{J/mol} \cdot \text{K}$$

$$C_{PC} = 13.39 + 0.0770T - 18.71 \times 10^{-6}T^2 \quad \text{J/mol} \cdot \text{K}$$

$$-r_A = k C_{A0} \frac{1-x}{1+x} \frac{T_0}{T}$$



Problem formulation and analysis:

$$F_{A0} \frac{dx}{dV} = -r_A$$

$$F_{A0}(C_{PA} + x\Delta C_P) \frac{dT}{dV} = UA(T_a - T) + r_A \Delta H_R$$

$$\Delta C_P = C_{PB} + C_{PC} - C_{PA}$$

MATLAB program design:

$$\frac{dx}{dV} = \frac{-r_A}{F_{A0}}$$

$$\frac{dT}{dV} = \frac{UA(T_a - T) + r_A \Delta H_R}{F_{A0} (C_{PA} + x \Delta C_P)}$$

$$0 \leq V \leq V_R$$

$$x(0) = 0$$

$$T(0) = T_0 = 1,035.$$

MATLAB program design:

— ex4_6_2.m —

```
function ex4_6_2
%
% Example 4-6-2 Temperature and conversion distributions in a plug-flow reactor
%
clear; close all;
clc
%
global V0 FA0 U A Ta CA0 T0
%
% Given data
%
V0=0.002; % Volumetric flow rate of the feeding materials (m^3/s)
VR=0.001; % Total volume of the reactor (m^3)
U=110; % Overall heat transfer coefficient (W/m^2.K)
A=150; % Heat transfer area (m^2/m^3)
T0=1030; % Feeding temperature (K)
Ta=1175; % exterior reactor surface temperature (K)
```

MATLAB program design:

— ex4_6_2.m —

```
X0=0; % zero conversion at the reactor inlet
```

```
P0=165e3; % Reactor pressure (Pa)
```

```
R=8.314; % Ideal gas constant (J/g-mol.K)
```

```
%
```

```
CA0=P0*(1-X0)/(R*T0); % Acetone concentration of the feed (mol/m^3)
```

```
FA0=V0*CA0; % Inlet molar flow rate (mol/s)
```

```
%
```

```
% Solving with ode45
```

```
%
```

```
[t, x]=ode45(@ex4_6_2ode, [0 VR],[X0 T0]');
```

```
%
```

```
% Results plotting
```

```
%
```

```
subplot(2,1,1)
```

```
plot(t, x(:,1))
```

```
ylabel('Conversion rate')
```

```
subplot(2,1,2)
```

MATLAB program design:

— ex4_6_2.m —

```
plot(t, x(:,2))
ylabel('Reactor temperature (K)')
xlabel('Reactor volume (m^3)')
%
% System equations
%
function odefun=ex4_6_2ode(t, y)
global V0 FA0 U A Ta CA0 T0
%
x=y(1);
T=y(2);
%
dHR=80770+6.8*(T-298)-5.75e-3*(T^2-298^2)-1.27e-6*(T^3-298^3);
k=3.58*exp(34222*(1/T0-1/T));
ra=-k*CA0*((1-x)/(1+x))*T0/T;
```

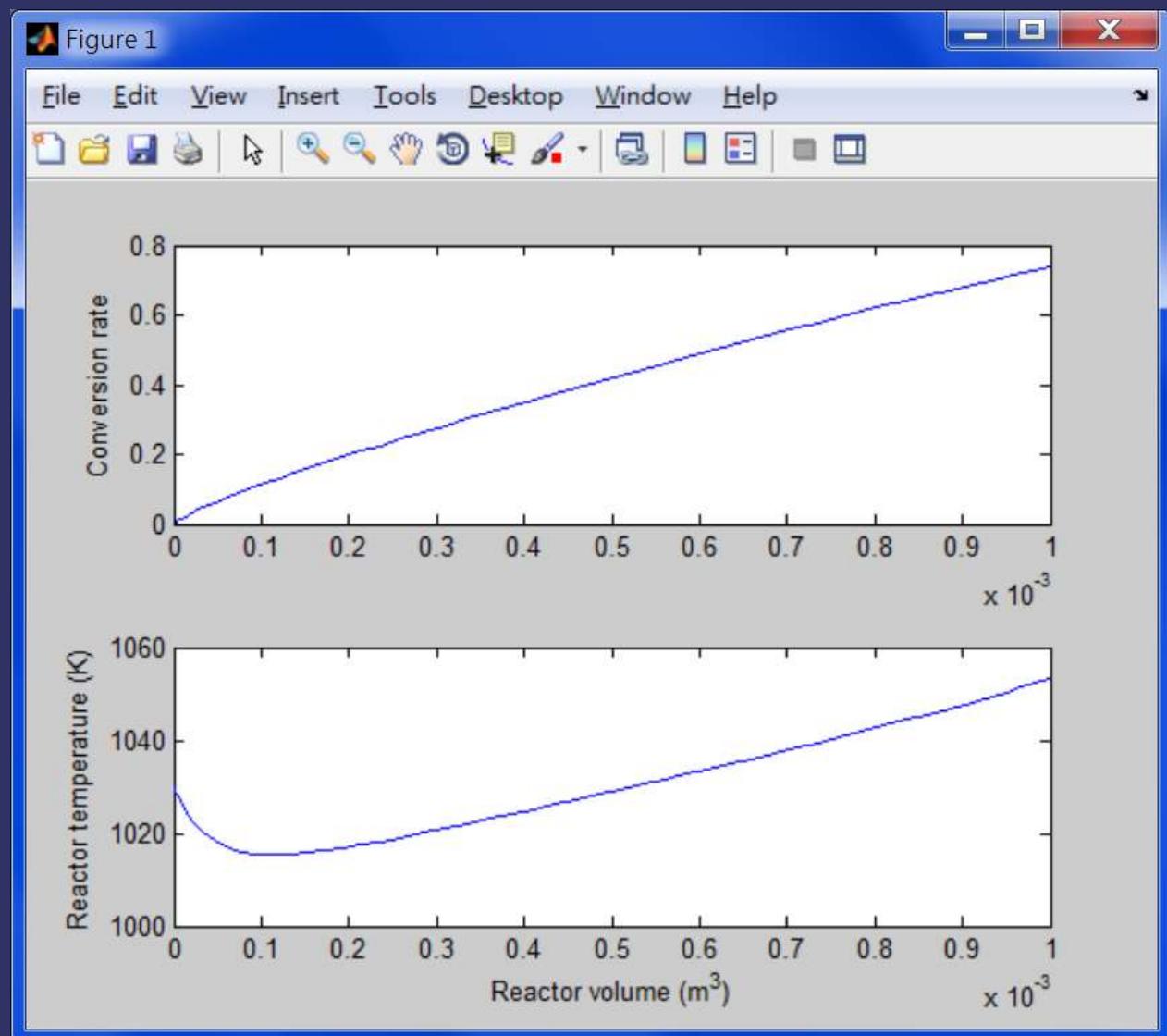
MATLAB program design:

— ex4_6_2.m —

```
Cpa=26.63+0.1830*T-45.86e-6*T^2;  
Cpb=20.04+0.0945*T-30.95e-6*T^2;  
Cpc=13.39+0.0770*T-18.71e-6*T^2;  
dCp=Cpb+Cpc-Cpa;  
%  
odefun=[-ra/FA0  
        (U*A*(Ta-T)+ra*dHR)/(Cpa+x*dCp)/FA0];
```

Execution results:

```
>> ex4_6_2
```



Example 4-6-3

Biochemical process dynamics in a batch reactor

$$\frac{dB}{dt} = \frac{kBS}{K + S}$$

$$\frac{dS}{dt} = \frac{-0.75kBS}{K + S}$$

where B and S represent, respectively, the concentration of biomass and substrate. Under the assumption that the initial concentrations of the biomass and the substrate are 0.5 and 5 mol/L, respectively, plot a diagram to show the changes of biomass and substrate concentration with respect to reaction time.

NOTE:

The kinetic constants for this reaction are $k = 0.3$ and $K = 10^{-6}$.

Problem formulation and analysis:

$$f_1(B, S) = \frac{0.3BS}{10^{-6} + S} = 0$$

$$f_2(B, S) = -\frac{0.225BS}{10^{-6} + S} = 0$$

$$\mathbf{J} \equiv \begin{bmatrix} \frac{\partial f_1}{\partial B} & \frac{\partial f_1}{\partial S} \\ \frac{\partial f_2}{\partial B} & \frac{\partial f_2}{\partial S} \end{bmatrix} = \begin{bmatrix} \frac{0.3S}{10^{-6} + S} & \frac{0.3 \times 10^{-6} B}{(10^{-6} + S)^2} \\ -\frac{0.225S}{10^{-6} + S} & -\frac{0.225 \times 10^{-6} B}{(10^{-6} + S)^2} \end{bmatrix}$$

Problem formulation and analysis:

If $S = 0$ and $B \neq 0$,

$$\mathbf{J} \Big|_{S=0} = \begin{bmatrix} 0 & 0.3 \times 10^{-6} B \\ 0 & -0.225 \times 10^{-6} B \end{bmatrix}$$

two eigenvalues are 0 and -0.225×10^6 ,

SR value is infinite;

To solve this problem, the MATLAB stiff ODE solvers, such as ode15s, ode23s, ode23t, and ode23tb, can be applied.

MATLAB program design:

— ex4_6_3.m —

```
function ex4_6_3
%
% Example 4-6-3 Biochemical process dynamics in a batch reactor
%
clear; close all;
clc
%
global k K
%
% Given data
%
k=0.3;
K=1.e-6;
%
% Solving with ode15s
%
```

MATLAB program design:

— ex4_6_3.m —

```
%  
% Results plotting  
%  
plot(t, x(:,1), t, x(:,2), '-.')  
legend('Biomass concentration', 'Substrate concentration')  
xlabel('Time')  
ylabel('System states')  
%  
% System equations  
%  
function y=ex4_6_3f(t, x)  
global k K  
B=x(1);  
S=x(2);  
%  
% Avoid unreasonable situation of S<0  
%
```

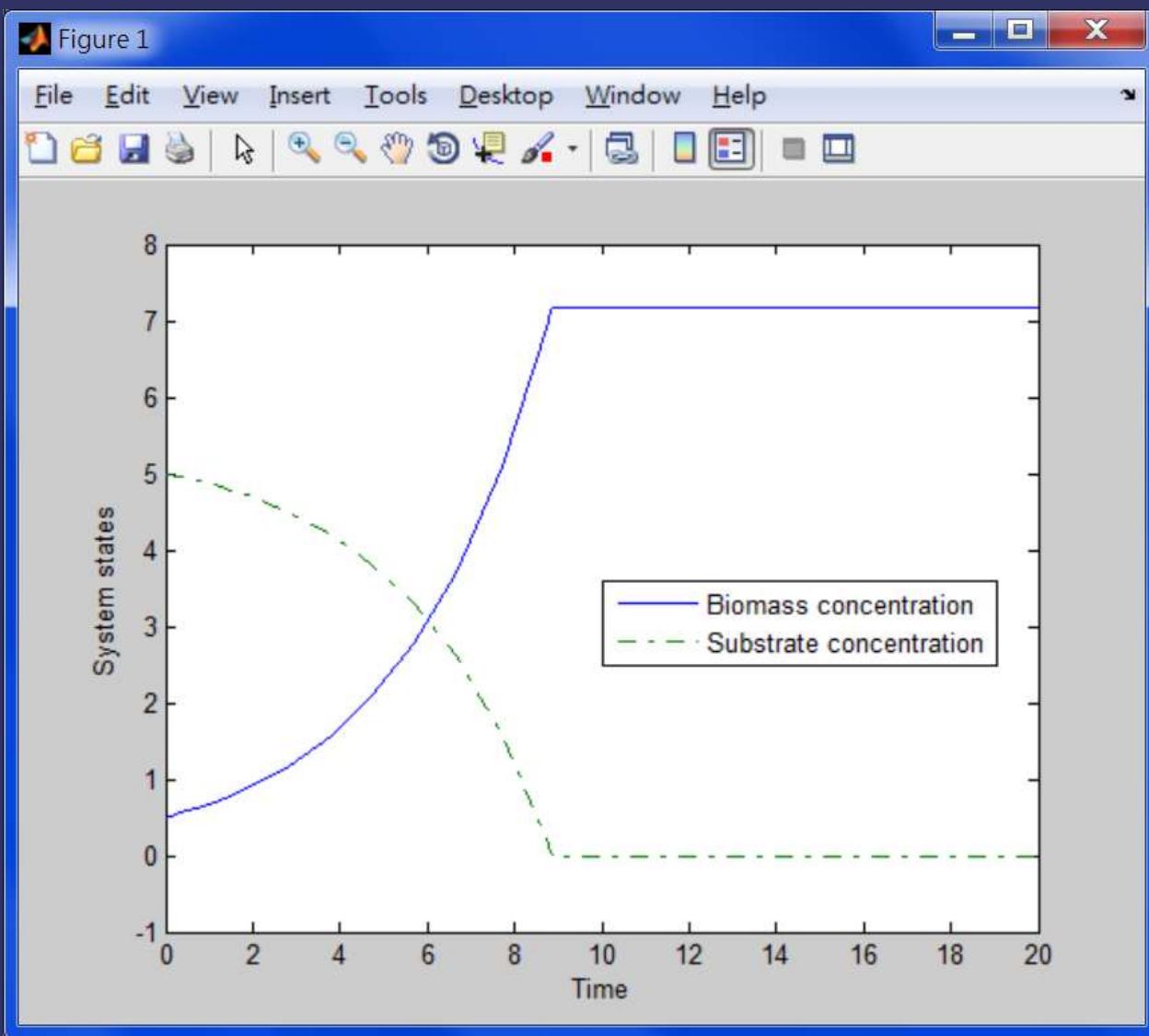
MATLAB program design:

— ex4_6_3.m —

```
if S<0  
    S=0;  
end  
y=[k*B*S/(K+S)  
   -0.75*k*B*S/(K+S)];
```

Execution results:

```
>> ex4_6_3
```



Execution results:

```
>> ex4_6_3
```

It should be noted that the physical constraint $S \geq 0$ was added into the system equation to prevent substrate concentration from reaching an unreasonable condition of negative values.

Example 4-6-4

Temperature distribution on a flat panel with heat conduction and radiation

Figure 4.2 schematically depicts a flat panel whose temperature at left-hand side is maintained at T_0 , while the right-end temperature T_s is affected by the ambient black-body temperature T_B . The thickness of the panel is L , and g_x is the amount of heat (W) transferring in the x direction. Besides, the heat transfer coefficient of the panel is a function of temperature, which is expressed by $k = 30(1+0.005T)$ (*Cutlip and Shacham, 1999*). If $T_0 = 300$ K, $T_B = 1,273$ K, and $L = 0.5$ m, determine the temperature distribution on the flat panel in the x direction and the right-hand side temperature, T_s .

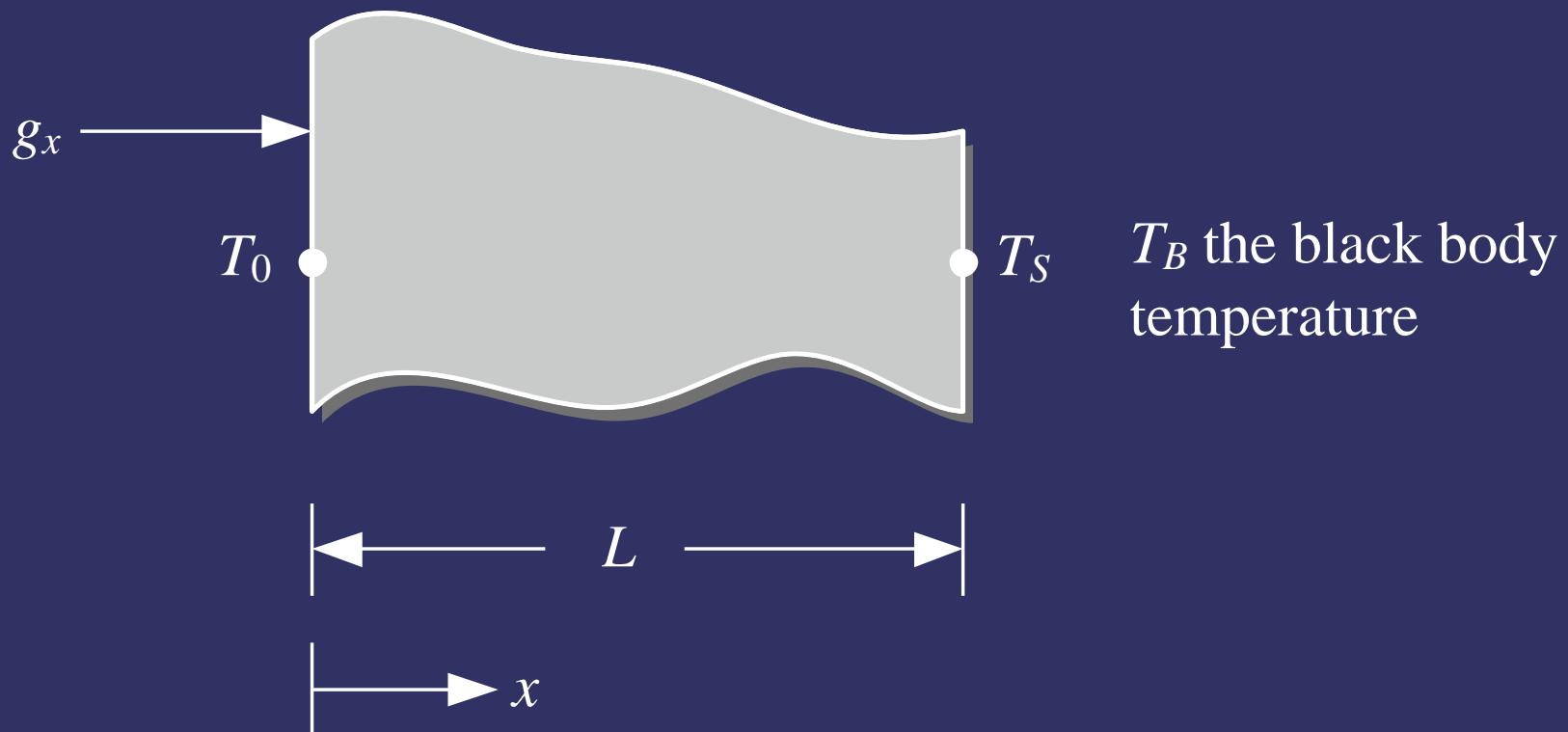


Figure 4.2 A flat panel with heat conduction and radiation.

Problem formulation and analysis:

Fourier's law,

$$k \frac{dT}{dx} = -\frac{g_x}{A}$$

Stefan-Boltzmann's law

$$\left. \frac{g_x}{A} \right|_{x=L} = \sigma (T_S^4 - T_B^4) \Big|_{x=L}$$

where σ is the Stefan-Boltzmann constant whose value is $5.676 \times 10^{-8} \text{ W/m}^2 \cdot \text{K}^4$.

$$T(0) = T_0 = 300$$

$$T(L) = T_S.$$

MATLAB program design:

— ex4_6_4.m —

```
function ex4_6_4
%
% Example 4-6-4 temperature distribution on a flat panel
%
clear; close all;
clc
%
global sigma TB
%
% Given data
%
L=0.5; % Length of the flat panel (m)
T0=300; % Temperature at the left-hand side of the panel (K)
TB=1273; % Black-body temperature (K)
sigma=5.676e-8; % Stefan-Boltzmann constant
%
% Initial guess value of the parameter
```

MATLAB program design:

— ex4_6_4.m —

```
%  
TS=500; % K  
solinit=bvpinit(linspace(0,L), @ex4_6_4init, TS);  
%  
% Solving with bvp4c  
%  
sol=bvp4c(@ex4_6_4ode, @ex4_6_4bc, solinit);  
%  
TS=sol.parameters;  
fprintf('\n TS=%7.3f K\n', TS)  
x=linspace(0, L, 100);  
T=deval(sol, x); % obtaining the temperature value by deval  
%  
% Results plotting  
%  
plot(x, T)
```

MATLAB program design:

— ex4_6_4.m —

```
xlabel('Flat panel length (m)')  
ylabel('Temperature (K)')  
%  
% ODE equation  
%  
function y=ex4_6_4ode(L, T, TS)  
global sigma TB  
k=30*(1+0.005*T);  
qxA=sigma*(TS^4-TB^4);  
y=-qxA/k;  
%  
% Boundary conditions  
%  
function res=ex4_6_4bc(ya, yb, TS)  
res=[ya(1)-300  
yb(1)-TS];
```

MATLAB program design:

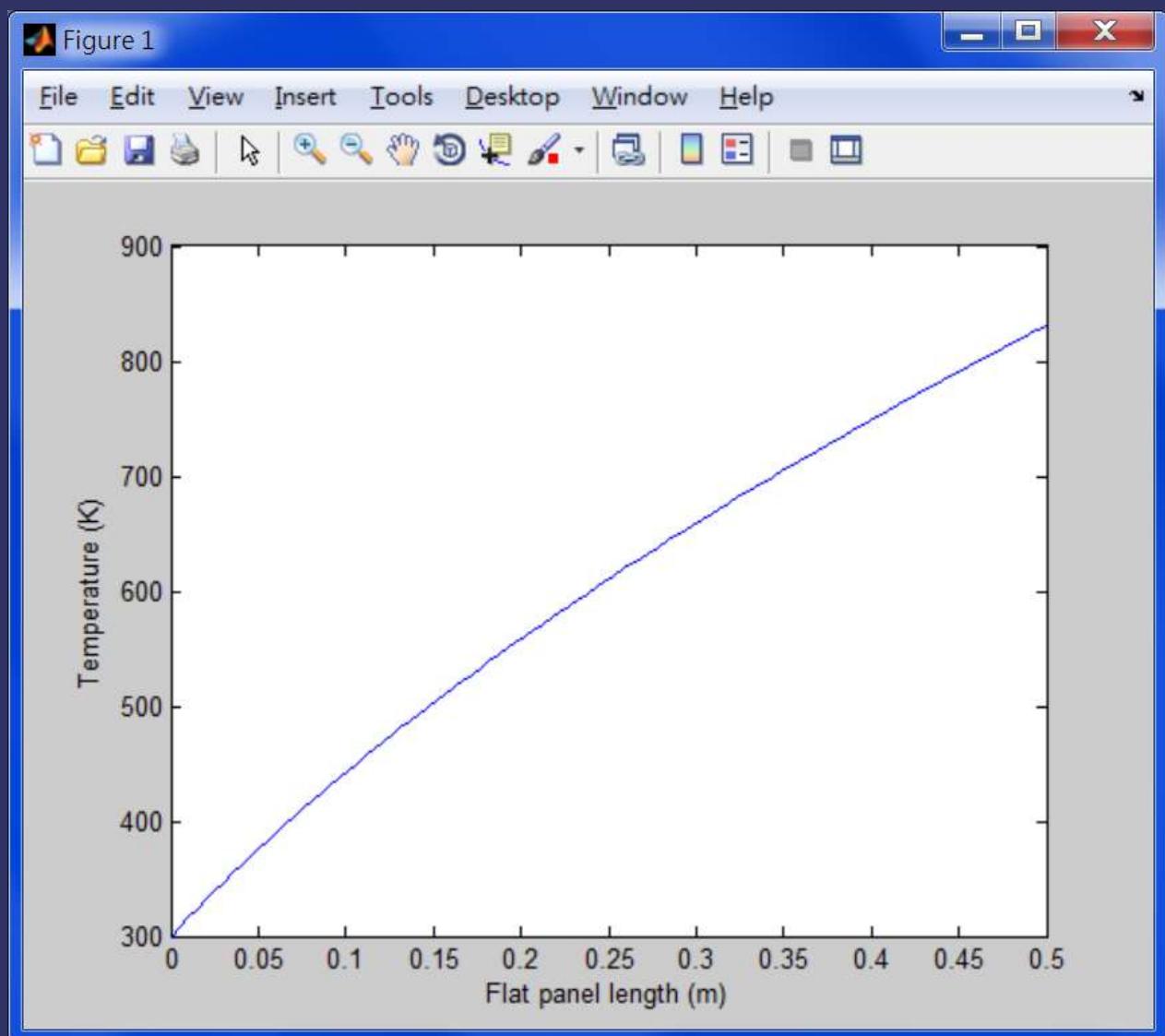
— ex4_6_4.m —

```
%  
% Initial guess value  
%  
function yinit=ex4_6_4init(L)  
yinit=300*(1+L);
```

Execution results:

```
>> ex4_6_4
```

TS=831.106 K



Example 4-6-5

Flow dynamics of a non-Newtonian fluid

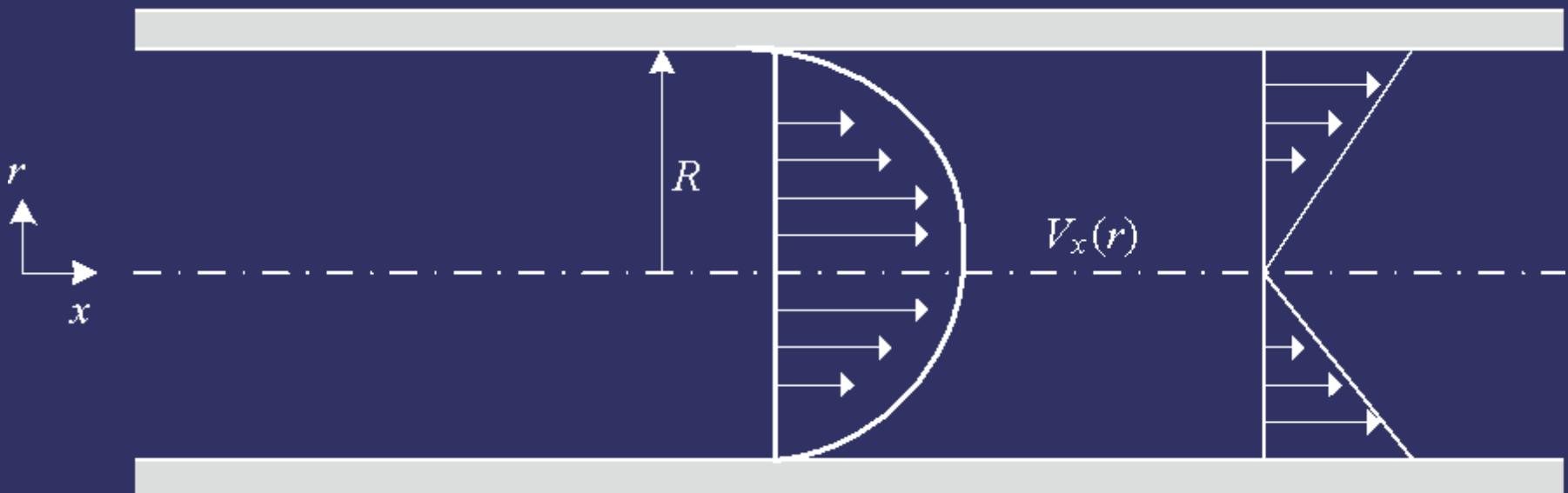


Figure 4.3 The flow of a non-Newtonian fluid in an infinitely long tube.

Carreau model

$$\frac{\mu}{\mu_0} = \left[1 + (t_1 \dot{r})^2 \right]^{(n-1)/2}$$

where n is a constant, and μ , and t_1 denote, respectively, the viscosity, the *shear rate*, the viscosity when the shear rate is zero, and the characteristic time.

Determine the flow velocity in the tube $V_x(r)$ and calculate the *volumetric flow rate* under the following conditions:

$$\mu_0 = 105 \text{ Pa} \cdot \text{s}, \quad t_1 = 4.35 \text{ s}, \quad n = 0.25,$$

$$-\Delta P / L = 10 \text{ kPa/m}, \quad R = 0.1 \text{ m}$$

Problem formulation and analysis:

momentum balance

$$\frac{d}{dr}(r\tau_{rx}) = -\frac{\Delta P}{L} r$$

$$\tau_{rx} = -\mu \dot{r} = -\mu \frac{dV_x(r)}{dr}$$

$$V_x = 0, \quad r = R$$

$$\frac{dV_x}{dr} = 0, \quad r = 0$$

Problem formulation and analysis:

$$t = r / R$$

$$\xi = V_x / V_0$$

$$V_0 = \frac{(-\Delta P)R^2}{L\mu_0}.$$

$$\frac{d^2\xi}{dt^2} = -\frac{\frac{1}{t} \frac{d\xi}{dt} + \left[1 + \lambda^2 \left(\frac{d\xi}{dt} \right)^2 \right]^{(1-n)/2}}{1 - \left[1 + \lambda^2 \left(\frac{d\xi}{dt} \right)^2 \right]^{(1-n)/2}}$$

Problem formulation and analysis:

$$\lambda = t_1 V_0 / R.$$

$$y_1 = \xi$$

$$y_2 = d\xi / dt$$

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = -\frac{\frac{1}{t}y_2 + \left[1 + \lambda^2 y_2^2\right]^{(1-n)/2}}{1 - \frac{(1-n)\lambda^2 y_2}{\left[1 + \lambda^2 y_2^2\right]}}$$

Problem formulation and analysis:

$$0 \leq t \leq 1$$

$$y_1(1) = \frac{1}{V_0} V_x \Big|_{r=R} = 0$$

$$y_2(0) = \frac{1}{V_0} \frac{dV_x}{dt} \Big|_{r=0} = 0$$

$$Q = \int_0^R 2\pi r V_x dr$$

MATLAB program design:

Step 1: Solve the BVP formed by the differential equations (4.6-24) and the boundary conditions (4.6-25).

Step 2: Perform the numerical integration for (4.6-26) to obtain the volumetric flow rate.

MATLAB program design:

— ex4_6_5.m —

```
function ex4_6_5
%
% Example 4-6-5 Flow dynamics of a non-Newtonian fluid
%
clear; close all;
clc
%
global n lambda
%
% Given data
%
mu_0=105; % Viscosity when the shear rate is zero (Pa.s)
t1=4.35; % Characteristic time (s)
n=0.25; % Dimensionless constant
PL=10e3; % -dP/L (Pa/m)
R=0.1; % radius (m)
%
```

MATLAB program design:

— ex4_6_5.m —

```
v0=PL*R^2/mu_0;
lambda=t1*v0/R;
%
% Initial guess values
%
solinit=bvpinit(linspace(0,1), @ex4_6_5init);
%
% Solving with bvp4c
%
sol=bvp4c(@ex4_6_5ode, @ex4_6_5bc, solinit);
%
t=linspace(0,1,100);
y=deval(sol, t);
vx=y(1,:)*v0;
dvxdr=y(2,:)*v0/R;
r=t*R;
```

MATLAB program design:

— ex4_6_5.m —

```
% Results plotting
%
subplot(211)
plot(r, vx)
xlabel('Radial length, r (m)')
ylabel('Velocity (m/s)')
subplot(212)
plot(r, dvxdr)
xlabel('Radial length, r (m)')
ylabel('Acceleration (1/s)')
%
% Calculating the volumetric flow rate
%
Q=2*pi*trapz(r, r.*vx);
%
% Results printing
%
```

MATLAB program design:

— ex4_6_5.m —

```
fprintf('\n Volumetric flow rate = %5.3f L/s. \n', Q*1000)
%
% System equations
%
function f=ex4_6_5ode(t, y)
global n lambda
if t==0
f=[y(2)
-((1+lambda^2*y(2)^2)^((1-n)/2))/(1-(1-n)*lambda^2*y(2)/...
(1+lambda^2*y(2)^2));
else
f=[y(2)
-(y(2)/t+(1+lambda^2*y(2)^2)^((1-n)/2))/(1-(1-n)*lambda^2*y(2)/...
(1+lambda^2*y(2)^2));
end
%
% Boundary conditions
```

MATLAB program design:

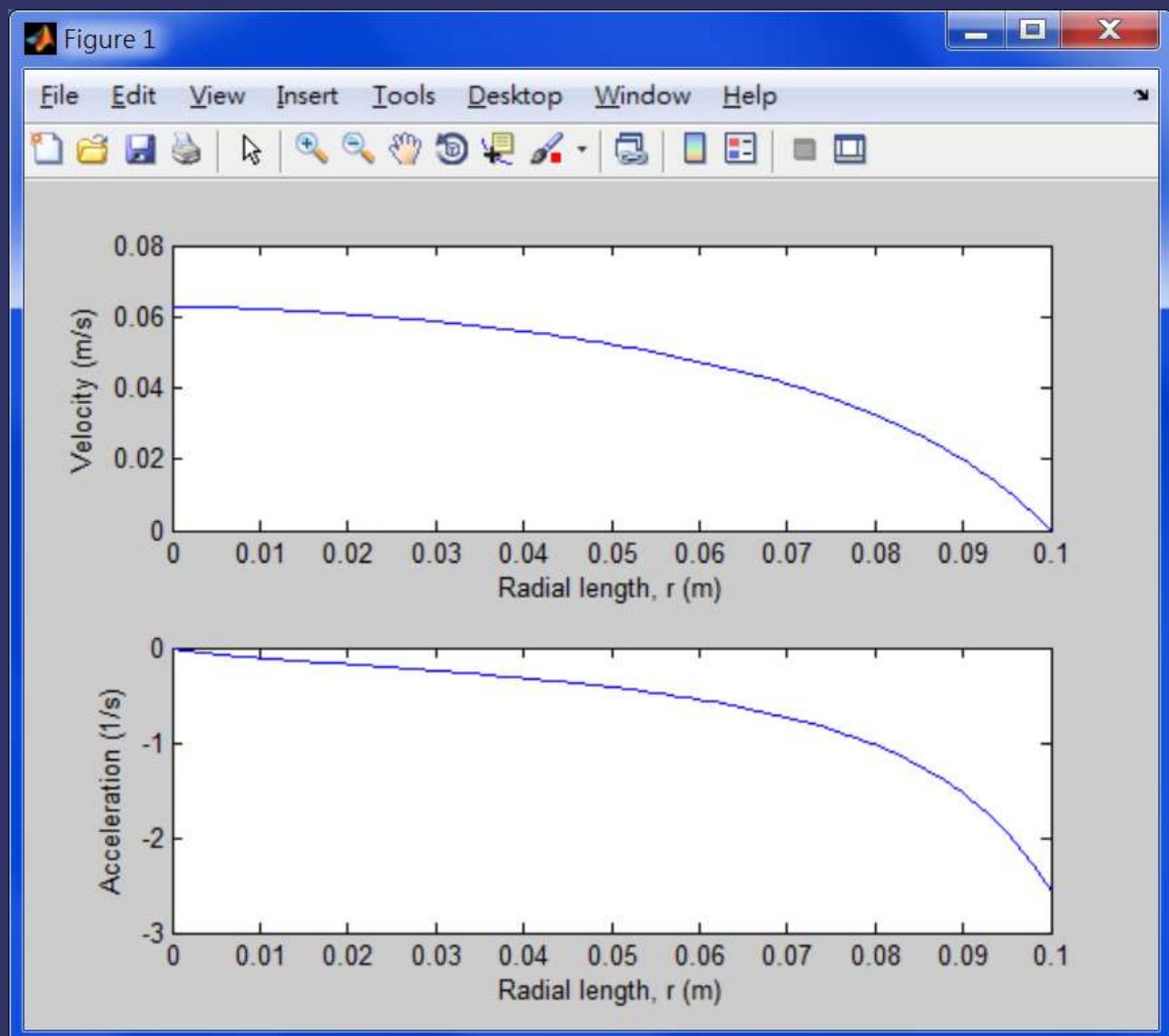
— ex4_6_5.m —

```
%  
function res=ex4_6_5bc(ya, yb)  
res=[ yb(1)  
      ya(2)];  
%  
% Initial guess values  
%  
function yinit=ex4_6_5init(t)  
yinit=[0.25*(1-t.^2)  
      -0.5*t];
```

Execution results:

>> ex4_6_5

Volumetric flow
rate = 1.180 L/s.



Example 4-6-6

Optimal operation temperature for penicillin fermentation

$$\frac{dy_1}{dt} = b_1 y_1 - \frac{b_1}{b_2} y_1^2, \quad y_1(0) = 0.05$$

$$\frac{dy_2}{dt} = b_3 y_1, \quad y_2(0) = 0$$

$$b_1 = 14.9714 - 0.0749(T - 30)^2$$

$$b_2 = 1.0743 - 0.0054(T - 30)^2$$

$$b_3 = 1.9886 - 0.01(T - 20)^2$$



If the admissible temperature operating range is $20 \leq T \leq 30$ °C, determine the time-dependent optimal temperature profile such that the concentration of penicillin at the end of the reaction is maximized.

Problem formulation and analysis:

A dynamic optimization problem:

Find $\mathbf{u}(t)$ such that the objective function

$$J = \Phi\left(\mathbf{x}(t_f), t_f\right) + \int_0^{t_f} \phi(t, \mathbf{x}, \mathbf{u}) dt$$

is maximized subject to the following IVP ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}, \mathbf{u})$$

$$\mathbf{x}|_{t=0} = \mathbf{x}(0)$$

Problem formulation and analysis:

Hamiltonian function

$$H = \phi + \boldsymbol{\lambda}^T \mathbf{f} = \phi + \sum_{i=1}^n \lambda_i f_i$$

λ is the *adjoint vector*

$$\frac{d\boldsymbol{\lambda}}{dt} = \frac{\partial \phi}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$$

$$\boldsymbol{\lambda}|_{t=t_f} = \left. \frac{\partial \Phi}{\partial \mathbf{x}} \right|_{t=t_f}$$

at the optimality condition

$$\frac{\partial H(t, \mathbf{x}^*, \mathbf{u}^*, \boldsymbol{\lambda})}{\partial \mathbf{u}} = 0$$

Problem formulation and analysis:

a Hamiltonian function

$$H = \lambda_1 \left(b_1 y_1 - \frac{b_1}{b_2} y_1^2 \right) + \lambda_2 (b_3 y_1)$$

$$\frac{dy_1}{dt} = b_1 y_1 - \frac{b_1}{b_2} y_1^2$$

$$\frac{dy_2}{dt} = b_3 y_1$$

$$\frac{d\lambda_1}{dt} = -\lambda_1 \left(b_1 - 2 \frac{b_1}{b_2} y_1 \right) - b_3 \lambda_2$$

$$\frac{d\lambda_2}{dt} = 0$$

Problem formulation and analysis:

$$y_1|_{t=0} = y_1(0)$$

$$y_2|_{t=0} = y_2(0)$$

$$\lambda_1|_{t=t_f} = 0$$

$$\lambda_2|_{t=t_f} = 1$$

$$\frac{\partial H}{dT} = \lambda_1 \left(y_1 \frac{\partial b_1}{\partial T} - y_1^2 \frac{\partial(b_1/b_2)}{\partial T} \right) + \lambda_2 y_1 \frac{\partial b_3}{\partial T} = 0$$

MATLAB program design:

Step 1: Divide the interval $[0 \ 1]$ into N sections, where N is a specified integer, and assign initial guess value for the temperature at each interval.

Step 2: Obtain y_{ij} and λ_{ij} , $i = 1, 2, j = 1, 2, \dots, N + 1$, at each specified time point by solving the BVP (4.6-35).

Step 3: Solve for $T_j, j = 1, 2, \dots, N + 1$, by inserting the value of y_{ij} and λ_{ij} at each point into the algebraic equation (4.6-36); then repeat steps 2 and 3 until T_j converges.

MATLAB program design:

— ex4_6_6.m —

```
function ex4_6_6
%
% Example 4-6-6 Optimal operating temperature for penicillin fermentation
%
clear; close all;
clc
%
global T0 tspan dt
global x0
global x
%
% Given data and the solution parameters
%
N=20; % Number of divided sections
n=N+1; % Number of division points
tspan=linspace(0,1,n); % Time points
```

MATLAB program design:

— ex4_6_6.m —

```

T0=25*ones(1,n); % Initial guess values
y0=[0.05 0]; % Initial concentration
lambda0=[0 1]; % Initial value of the adjoint vector
x0=[y0 lambda0]; % Initial state value
check=1; % Program running flag (1=continue, 0=stop)
TOL=1e-5; % Convergence tolerance
%
while check == 1
    solinit=bvpinit(tspan, [1 1 0 1]); % Initial guess value for states
    sol=bvp4c(@ex4_6_6ode, @ex4_6_6bc, solinit); % Solving with bvp4c
    x=deval(sol, tspan); % State values on time points
    T1=fsolve(@ex4_6_fun, T0, [20 30]); % Solve for T when dH/dT=0
    error=sum(abs(T1-T0)); % Convergence error of the value of T
    if error <= TOL
        check=0; % Converged and change the flag
    end
end

```

MATLAB program design:

— ex4_6_6.m —

```
else
```

```
    T0=T1; % if not converged, an iteration is conducted
```

```
end
```

```
end
```

```
%
```

```
% Results plotting
```

```
%
```

```
solinit=bvpinit(tspan, [1 1 0 1]); % Initial guess value for BVP state
```

```
sol=bvp4c(@ex4_6_6ode, @ex4_6_6bc, solinit);
```

```
x=deval(sol, tspan); % State value on time points
```

```
y1=x(1,:);
```

```
y2=x(2,:);
```

```
lambda1=x(3,:);
```

```
lambda2=x(4,:);
```

```
b1=14.9714-0.0749*(T0-30).^2;
```

```
b2=1.0743-0.0054*(T0-30).^2;
```

```
b3=1.9886-0.01*(T0-20).^2;
```

MATLAB program design:

— ex4_6_6.m —

```
%  
subplot(2,2,1)  
plot(tspan, y1, tspan, y2)  
legend('y1', 'y2')  
xlabel('time')  
ylabel('concentrations')  
axis([0 1 0 1.5])  
subplot(2,2,2)  
plot(tspan,lambda1,tspan,lambda2)  
xlabel('time')  
ylabel('\lambda')  
legend('\lambda_1', '\lambda_2')  
axis([0 1 -0.5 5])  
%  
dHdT=ex4_6fun(T0);  
H=lambda1.*(b1.*y1-b1./b2.*y1.^2)+lambda2.*b3.*y1;  
subplot(2,2,3)
```

MATLAB program design:

— ex4_6_6.m —

```
plot(tspan, dHdT, tspan, H)
xlabel('time')
ylabel('dHdT & H values')
legend('dHdT', 'H')
axis([0 1 -1 2])
subplot(2,2,4)
stairs(tspan, T0)
xlabel('time')
ylabel('optimal temperature')
axis([0 1 15 30])
%
% Results output
%
y2f=y2(:, n);
fprintf('\n The optimal penicillin concentration value = %.3f.\n', y2f)
%
% Differential equations (kinetic model and the adjoint equation)
%
```

MATLAB program design:

— ex4_6_6.m —

```
function f=ex4_6_6ode(t, x)
global T0 tspan dt
y1=x(1);
y2=x(2);
lambda1=x(3);
lambda2=x(4);
T=interp1(tspan, T0, t, 'spline');
b1=14.9714-0.0749*(T-30)^2;
b2=1.0743-0.0054*(T-30)^2;
b3=1.9886-0.01*(T-20)^2;
f=[b1*y1-b1/b2*y1^2
   b3*y1;
   -lambda1*(b1-2*b1/b2*y1)-b3*lambda2
   0];
%
% Boundary conditions
%
```

MATLAB program design:

— ex4_6_6.m —

```
function res=ex4_6_6bc(ya, yb)
global x0
y10=x0(1);
y20=x0(2);
lambda10=x0(3);
lambda20=x0(4);
res=[ya(1)-y10
     ya(2)-y20
     yb(3)-lambda10
     yb(4)-lambda20];
%
% Partial derivative of the Hamiltonian with respect to T
%
function HT=ex4_6_6fun(T)
global x
y1=x(1,:);
y2=x(2,:);
```

MATLAB program design:

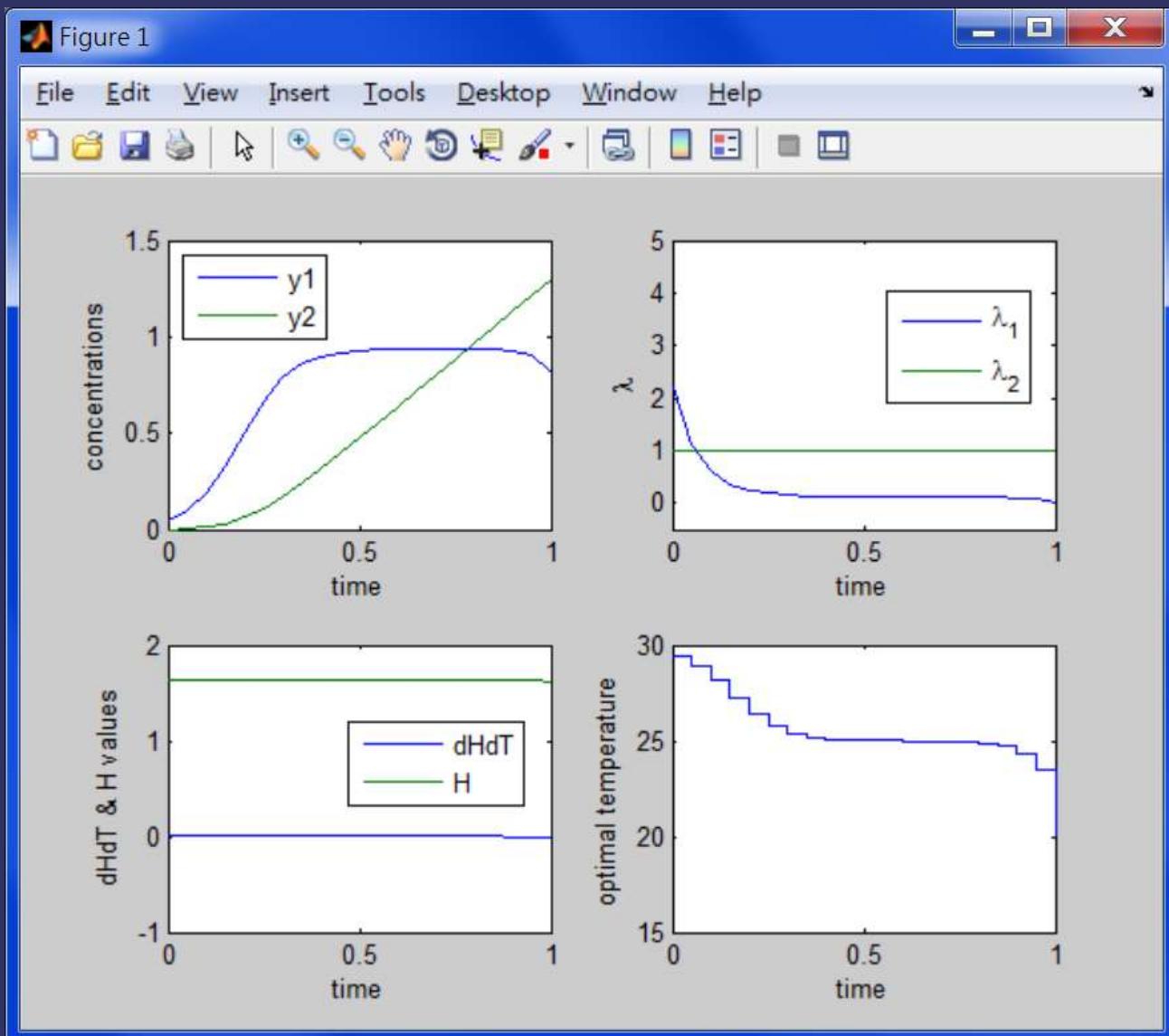
— ex4_6_6.m —

```
lambda1=x(3,:);  
lambda2=x(4,:);  
b1=14.9714-0.0749*(T-30).^2;  
b2=1.0743-0.0054*(T-30).^2;  
b3=1.9886-0.01*(T-20).^2;  
db1=-2*0.0749*(T-30);  
db2=-2*0.0054*(T-30);  
db3=-2*0.01*(T-20);  
db1b2=db1./b2-b1.*b2.^(-2).*db2;  
HT=lambda1.*(y1.*db1-y1.^2.*db1b2)+lambda2.*y1.*db3;
```

Execution results:

`>> ex4_6_6`

The optimal penicillin concentration value = 1.299.



4.8 Summary of the MATLAB commands related to this chapter

(I) IVP ODE commands

Command	Description
ode45	
ode23	Nonstiff IVP ODE solvers
ode113	
ode15s	
ode23s	Stiff IVP ODE and DAE solvers
ode23t	
ode23tb	

(II) BVP ODE problem

Command	Description
bvp4c	The solver for two-point BVPs.
bvpinit	The command for initializing the independent variable, system states, and the unknown parameters to be determined.
bvpset	Create/alter the BVP options structure.
deval	Interpolate solution to any point through sol (the solution structure).

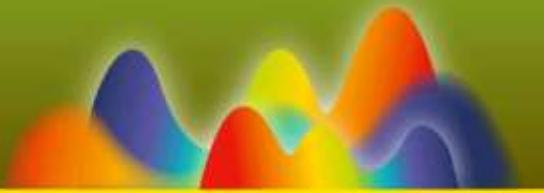
(III) Simulink ODE solution editor

Command	Description
dee	The differential equation editor



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 5

Numerical Solution of Partial Differential Equations

版權所有・請勿翻製

- ① solve PDEs with the MATLAB PDE toolbox.
- ② The PDE solver and the graphical interface will be demonstrated through the solution of some typical PDEs.
- ③ The relevant applications to chemical engineering problems will also be presented.

5.1 Classifications of PDEs

► 5.1.1 The order of a PDE

- the first-order PDE

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

- second-order PDE:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \left(\frac{\partial u}{\partial x} \right)^3 + \frac{\partial u}{\partial y} = 0$$

- third-order PDE:

$$\left(\frac{\partial^3 u}{\partial x^3} \right)^2 + \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

5.1.2 Nonlinearity of a PDE

- second-order PDE

$$a(\cdot) \frac{\partial^2 u}{\partial y^2} + b(\cdot) \frac{\partial^2 u}{\partial x \partial y} + c(\cdot) \frac{\partial^2 u}{\partial x^2} + d(\cdot) = 0$$

Category	Properties of the coefficient
Linear	The coefficient (\cdot) is a constant or a function of the independent variables (x, y).
Quasilinear	(\cdot) is a function of the <i>dependent variable</i> u , or a function of lower order terms of the partial derivatives in the PDE, for example (\cdot) = ($x, y, u, \partial u / \partial x, \partial u / \partial y$).
Nonlinear	The coefficient (\cdot) is a function with a term equal to the order of the PDE, for example (\cdot) = ($x, y, u, \partial^2 u / \partial x^2, \partial^2 u / \partial y^2, \partial^2 u / \partial x \partial y$).

$$a \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} + d \frac{\partial u}{\partial x} + e \frac{\partial u}{\partial y} + fu + g = 0$$

Category	Predicate	Examples
Elliptic	$b^2 - 4ac < 0$	Laplace equation, $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ Poisson equation, $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y);$ $-\nabla \cdot (\tilde{c} \nabla u) + \tilde{a}u = \tilde{f}.$
Parabolic	$b^2 - 4ac = 0$	Heat transfer or diffusion equation, $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2};$ $\tilde{d} \frac{\partial u}{\partial t} - \nabla \cdot (\tilde{c} \nabla u) + \tilde{a}u = \tilde{f}.$
Hyperbolic	$b^2 - 4ac > 0$	Wave equation, $\frac{\partial^2 u}{\partial t^2} = \alpha^2 \frac{\partial^2 u}{\partial x^2};$ $\tilde{d} \frac{\partial^2 u}{\partial t^2} - \nabla \cdot (\tilde{c} \nabla u) + \tilde{a}u = \tilde{f}.$

5.1.3 Categories of initial conditions and boundary conditions

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

(i) Dirichlet condition

If the dependent variable (T) is specified at a certain value of the independent variable,

$$\begin{cases} T|_{x=0} = f(t), & t > 0 \\ T|_{x=1} = T_1, & t > 0 \\ T|_{t=0} = T_0, & 0 \leq x \leq 1 \end{cases}$$

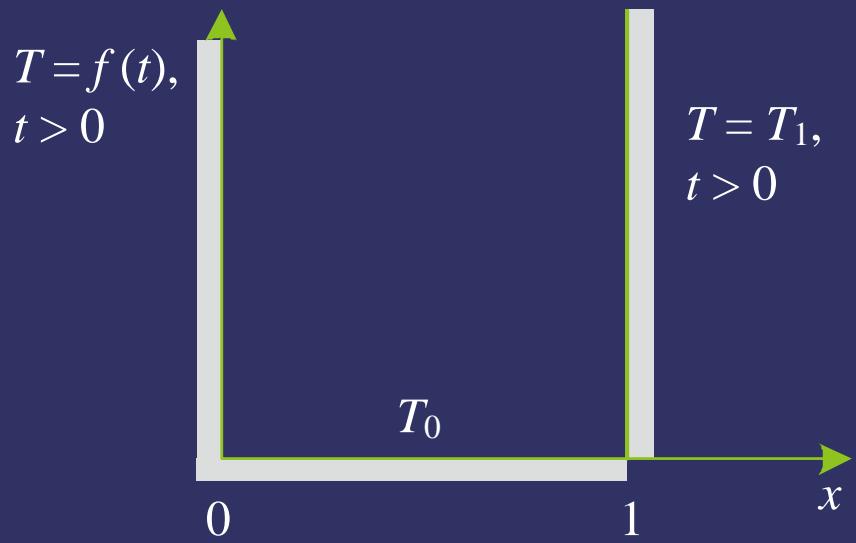


Figure 5.1
A schematic diagram of Dirichlet boundary conditions on a flat panel.

Besides, if the initial temperature distribution is a function of position, it also belongs to the category of Dirichlet boundary conditions. For example,

$$T|_{t=0} = f(x), \quad 0 \leq x \leq 1$$

(ii) Neumann condition

If the rate of change of the dependent variable on the boundary is a fixed value or a function of independent variable,

$$\left. \frac{\partial T}{\partial x} \right|_{x=1} = 0, \quad t \geq 0$$

or

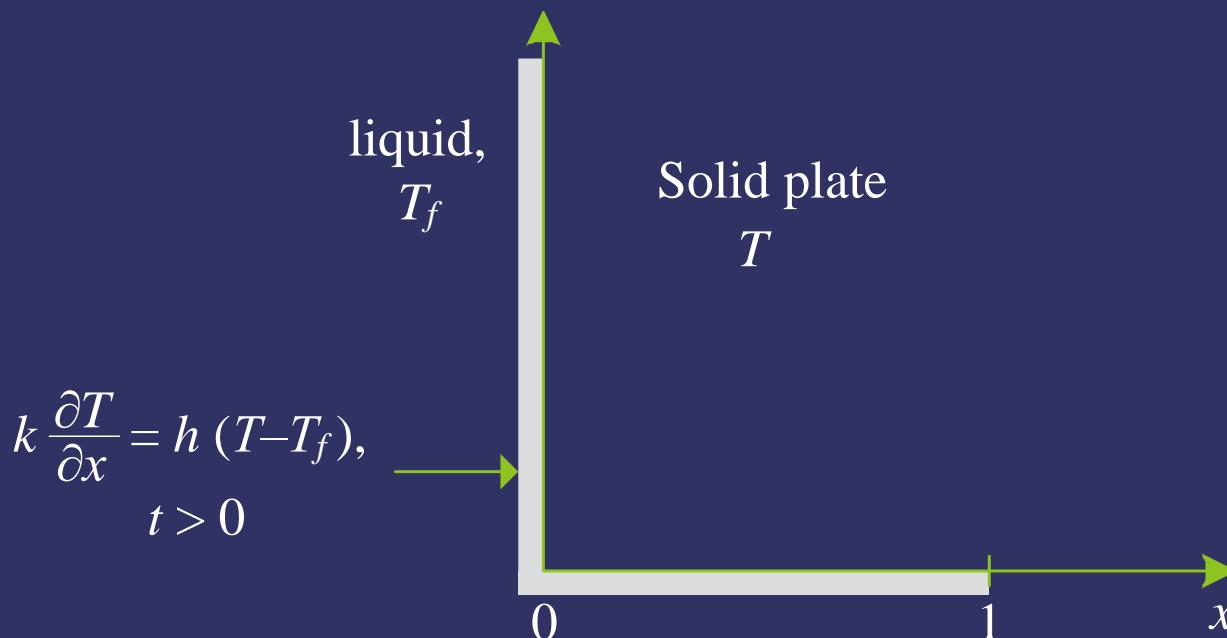
$$\left. \frac{\partial T}{\partial x} \right|_{t=0} = f(x), \quad 0 \leq x \leq 1$$

It is noted that the Neumann boundary condition is also called the natural boundary condition.

(iii) Robbins condition

If the rate of change of the dependent variable on the boundary is a function of the dependent variable itself,

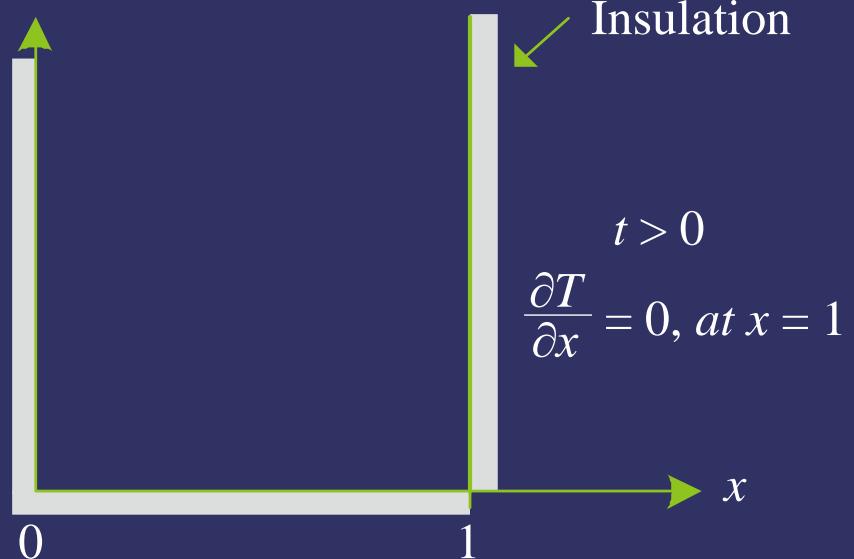
$$k \frac{\partial T}{\partial x} \Big|_{x=0} = h(T - T_f), \quad t \geq 0$$



(iv) Cauchy condition

The Cauchy condition means both Dirichlet and Neumann boundary conditions exist in the system.

$$\begin{aligned} t &> 0 \\ T &= f(t), \\ \text{at } x &= 0 \end{aligned}$$



$$T|_{x=0} = f(t), \quad t > 0 \quad \text{“Dirichlet condition”}$$

$$\left. \frac{\partial T}{\partial x} \right|_{x=1} = 0, \quad t > 0 \quad \text{“Neumann condition”}$$

5.2 The MATLAB PDE toolbox

► 5.2.1 The MATLAB PDE solver

$$c\left(x, t, u, \frac{\partial u}{\partial x}\right) \frac{\partial u}{\partial t} = x^{-m} \frac{\partial}{\partial x} \left(x^m f\left(x, t, u, \frac{\partial u}{\partial x}\right) \right) + s\left(x, t, u, \frac{\partial u}{\partial x}\right)$$

- $t_0 \leq t \leq t_f$
- $x \in [a \ b]$. The m value could be 0, 1, or 2
- initial values

$$u(x, t_0) = v_0(x)$$

- boundary conditions (BCs)

$$p(x, t, u) + q(x, t) f\left(x, t, u, \frac{\partial u}{\partial x}\right) = 0$$

```
sol = pdepe(m, 'pdefun', 'icfun', 'bcfun', xmesh, tspan, options)
```

- sol is a three-dimensional output matrix in which $\text{sol}(:, :, i)$ is the output of u_i , the i -th element of the vector of the dependent variables; in other words, $u_i = \text{sol}(:, :, i)$. Besides, the element which means denotes the obtained simulation results of u_i at $t = \text{tspan}(j)$ and $x = \text{xmesh}(k)$.

Input argument

Description

m

The parameter to specify the symmetric property of the PDE, whose value can be 0, 1, or 2.

xmesh

The mesh position vector of the independent variable x ; that is, $\text{xmesh} = [x_0 \ x_1 \ \dots \ x_N]$, where $x_0 = a$ (left end point) and $x_N = b$ (right end point).

tspan

The vector of the independent variable t (time) to be specified by the user; that is, $\text{tspan} = [t_0 \ t_1 \ \dots \ t_f]$, where t_0 is the initial time and t_f is the final time.

The PDE function file with its format as
function [c, f, s]=pdefun(x, t, u, dudx)

c=...

f=...

s=...

pdefun

Note that PDE function file only contains the coefficients c, f , and s in a vector format. For detailed usage of pdefun, refer to the illustrative examples provided in this subsection.

Input argument

Description

The file of initial values of u is in the following format

icfun
function $u=icfun(x)$
 $u=...$

Note that u is a row vector.

The file of boundary conditions is in the following format

bcfun
function $[pl, ql, pr, qr]=bcfun(xl, ul, xr, ur, t)$
 $pl=...$
 $ql=...$
 $pr=...$
 $qr=...$

In the above, ul and ur represent the solutions at the left boundary ($xl = a$) and right boundary ($xr=b$), respectively. In the output arguments, pl and ql denote, respectively, the row vector of p and q at the left boundary, whereas pr and qr are, respectively, the row vector of p and q at the right boundary.

options

The relevant solution parameters used in the solver; refer to “odeset” for its detailed usage.

- $[u_{\text{out}}, du_{\text{out}}dx] = \text{pdeval}(m, x_{\text{mesh}}, u_i, x_{\text{out}})$

Input
argument

Description

m

The symmetry of the problem; $m=0$: indicates a slab; $m=1$: cylindrical; $m=2$: spherical symmetry.

x_{mesh}

Mesh points of the independent variable x

u_i

$u_i = \text{sol}(j, :, i)$, the solution of the output variable u_i at the x_{mesh} position for $t_j = \text{tspan}(j)$

x_{out}

The position vector of the output points to be interpolated.

Output argument	Description
<code>uout</code>	The relevant output based on the designated position of <code>xout</code> when $t_j = \text{tspan}(j)$
<code>duoutdx</code>	The corresponding output value of du/dx .

Example 5-2-1

Solution of a univariate PDE using pdepe

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

$$0 \leq x \leq 1$$

(i) Initial value condition

$$u(x, 0) = \sin(\pi x)$$

(ii) Boundary conditions

$$\text{BC1} : u(t, 0) = 0$$

$$\text{BC2} : \pi e^{-1} + \frac{\partial u(1, t)}{\partial x} = 0$$

NOTE: The analytical solution to this problem is

$$u(x, t) = e^{-t} \sin(\pi x)$$

Example 5-2-1

Ans:

Step 1: Rewrite the PDE into the standard equation form as in (5.2-1):

$$\pi^2 \frac{\partial u}{\partial t} = x^0 \frac{\partial}{\partial x} \left(x^0 \frac{\partial u}{\partial x} \right) + 0$$

$$m = 0$$

$$c(x, t, u, \partial u / \partial x) = \pi^2$$

$$f(x, t, u, \partial u / \partial x) = \frac{\partial u}{\partial x}$$

and

$$s(x, t, u, \partial u / \partial x) = 0$$

Ans:

Step 2: Edit the PDE function file using the coefficient vector

function [c, f, s]=ex5_2_1pdefun(x, t, u, dudx)

c=pi^2;

f=dudx;

s=0;

Step 3: Edit the initial condition file for the dependent variable

function u0=ex5_2_1ic(x)

u0=sin(pi*x);

Ans:

Step 4:

$$\text{BC1: } u(0, t) + 0 \cdot \frac{\partial u}{\partial x}(0, t) = 0 \quad \text{at } x = 0$$

$$\text{BC2: } \pi e^{-t} + 1 \cdot \frac{\partial u}{\partial x}(1, t) = 0 \quad \text{at } x = 1$$

$$pl = u(0, t), \quad ql = 0$$

$$pr = \pi e^{-t}, \quad qr = 1$$

function [pl, ql, pr, qr]=ex5_2_1 bc(xl, ul, xr, ur, t)

pl=ul; % NOTE: ul indicates the u on the left boundary

ql=0;

pr=pi*exp(-t);

qr=1;

Ans:

Step 5: Take mesh points

```
x=linspace(0, 1, 20); % taking 20 meshes for x (xmesh)
```

```
t= linspace(0, 2, 20); % taking 20 time points (tspan)
```

Step 6: Solve with pdepe

```
m= 0; % based on the result from Step 1
```

```
sol= pdepe(m, @ex5_2_1pedefun, @ex5_2_1lic,  
@ex5_2_1bc, x, t);
```

Ans:

Step 7: Results plotting

```
u=sol(:, :, 1); % Extract the solution
```

```
surf(x, t, u) % 3D plot of solution
```

```
title('PDE numerical solution')
```

```
xlabel('position x')
```

```
ylabel('time t' )
```

```
zlabel('u')
```

ex5_2_1.m

```
function ex5_2_1
%
% Example 5-2-1 Solution of a PDE using pdepe
%
clear; close all;
clc
%
m=0;
x=linspace(0,1,20); % Take 20 mesh points at x (xmesh)
t=linspace(0,2,20); %Take 20 time points for output (tspan)
%
% Find PDE solution with pdepe
%
sol=pdepe(m, @ex5_2_1pdefun, @ex5_2_1ic, @ex5_2_1bc, x, t);
u=sol(:, :, 1); % Extract the solution
%
% Results plotting
%
figure(1)
surf(x, t, u)
```

ex5_2_1.m

```
title('PDE numerical solution')
xlabel('position x')
ylabel('time t' )
zlabel(' u')
axis([0 1 0 2 0 1])
%
% Comparison with the analytical solution
%
figure(2)
surf(x, t, exp(-t)*sin(pi*x));
title('Analytical solution')
xlabel('position x')
ylabel('time t' )
zlabel(' u')
axis([0 1 0 2 0 1])
%
% Solution at each position when t=tf=2
%
figure(3)
```

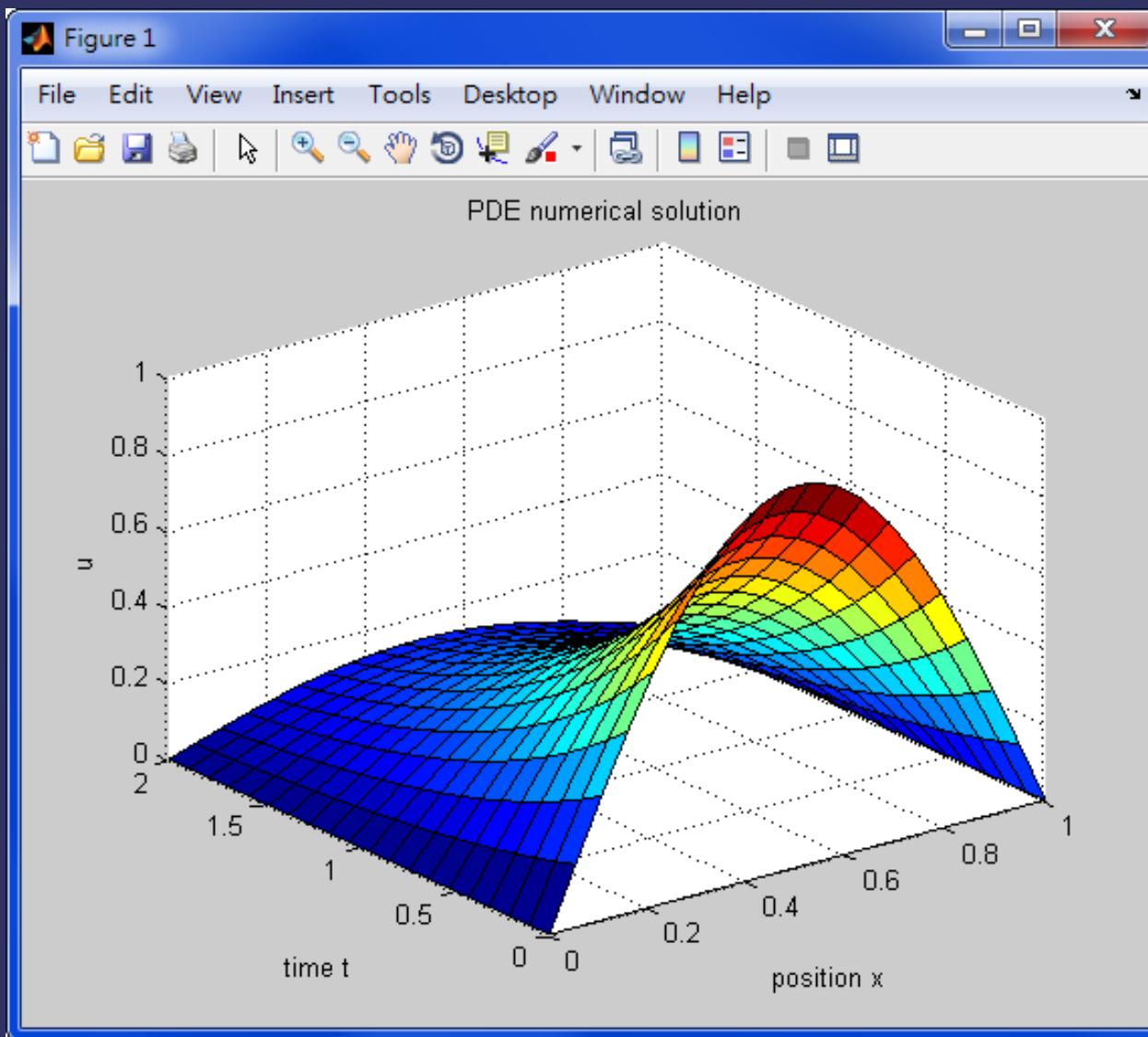
 ex5_2_1.m

```
xout=linspace(0, 1, 100); % position of the output point  
[uout, dudx]=pdeval(m, x, u(end,:), xout); % take the values at end time  
plot(xout, uout); % results plotting  
title('solution at each position when time tf=2')  
xlabel('x')  
ylabel('u')  
%  
% PDE function file  
%  
function [c, f, s]=ex5_2_1pdefun(x, t, u, dudx)  
c=pi^2;  
f=dudx;  
s=0;  
%  
% initial condition  
%  
function u0=ex5_2_1ic(x)  
u0=sin(pi*x);
```

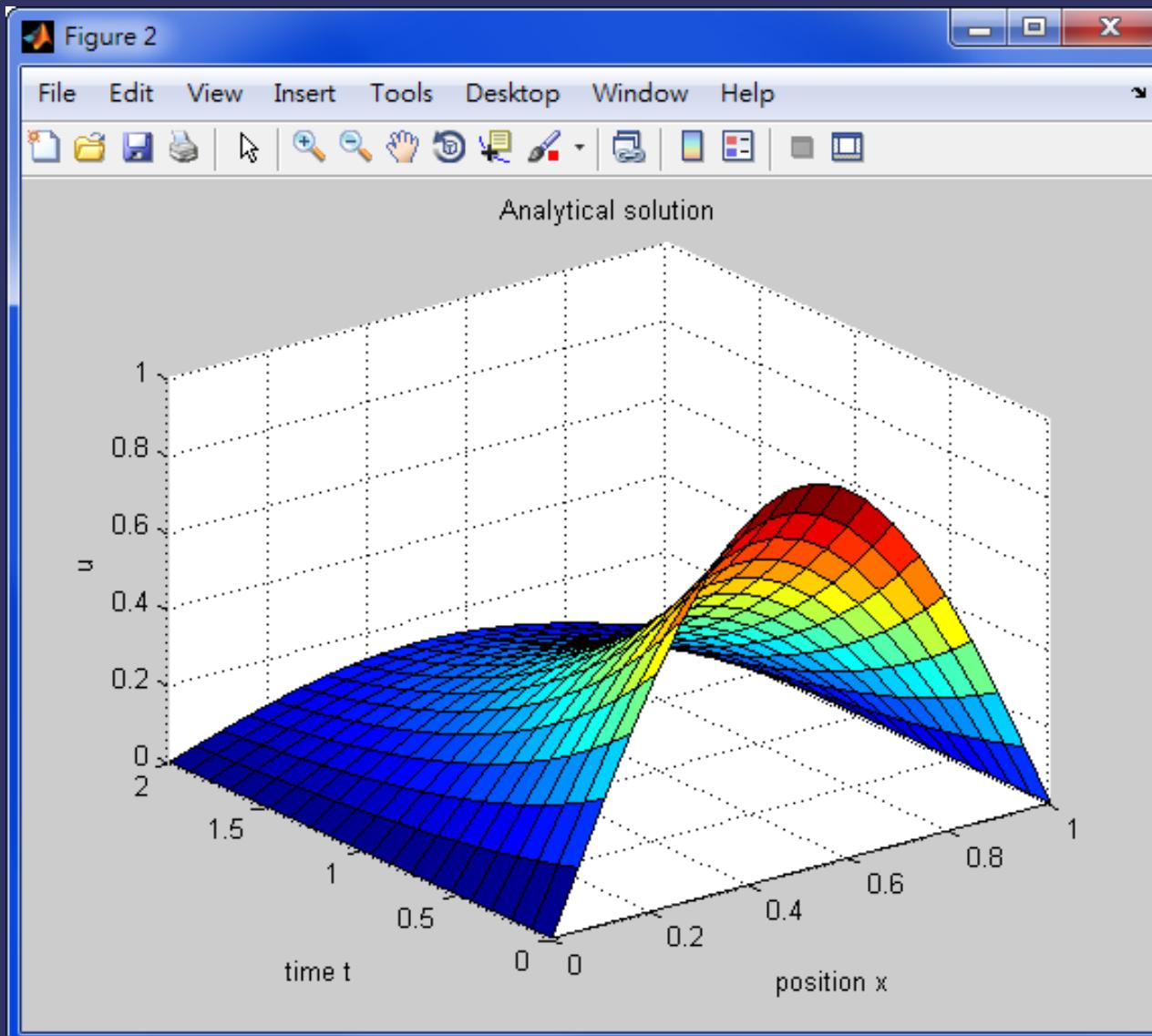
ex5_2_1.m

```
%  
% Boundary conditions  
%  
function [pl, ql, pr, qr]=ex5_2_1bc(xl, ul, xr, ur, t)  
pl=ul; % NOTE: ul indicates the u at the left boundary  
ql=0;  
pr=pi*exp(-t);  
qr=1;
```

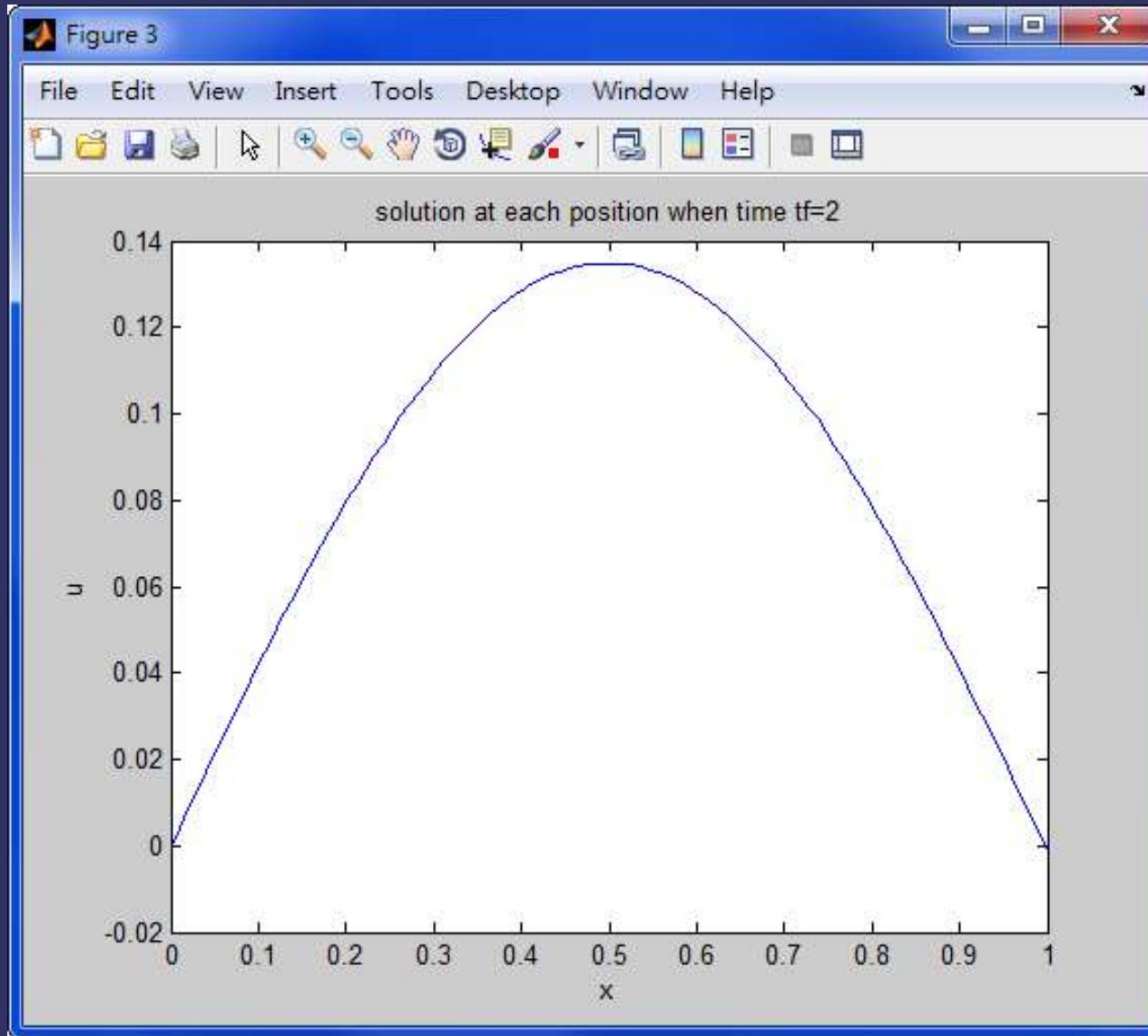
Execution results:



Execution results:



Execution results:



Example 5-2-2

Solution of a multivariate PDE

$$\frac{\partial u_1}{\partial t} = \frac{\partial^2 u_1}{\partial x^2} - G(u_1 - u_2)$$

$$\frac{\partial u_2}{\partial t} = 2 \frac{\partial^2 u_2}{\partial x^2} + G(u_1 - u_2)$$

where

$$G(u_1 - u_2) = \exp(5(u_1 - u_2)) - \exp(-10(u_1 - u_2))$$

$$0 \leq x \leq 1$$

$$t \geq 0$$

(i) initial conditions

$$u_1(x, 0) = 1$$

$$u_2(x, 0) = 0$$

(ii) boundary conditions

$$\frac{\partial u_1}{\partial x}(0, t) = 0$$

$$u_1(t, 0) = 0$$

$$u_2(1, t) = 1$$

$$\frac{\partial u_2}{\partial x}(1, t) = 0$$

Example 5-2-2

Ans:

Step 1: Rewrite the PDE into the standard form:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}_* * \frac{\partial_1}{\partial t} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \frac{\partial}{\partial x} \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ 2 \frac{\partial u_2}{\partial x} \end{bmatrix} + \begin{bmatrix} -G(u_1 - u_2) \\ G(u_1 - u_2) \end{bmatrix}$$

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Ans:

$$f = \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ 2 \frac{\partial u_2}{\partial x} \end{bmatrix}$$

$$s = \begin{bmatrix} -G(u_1 - u_2) \\ G(u_1 - u_2) \end{bmatrix}$$

$$m = 0$$

$$\begin{bmatrix} 0 \\ u_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}_* \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ 2 \frac{\partial u_2}{\partial x} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Ans:

$$pl = \begin{bmatrix} 0 \\ u_2 \end{bmatrix}$$

$$ql = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} u_1 - 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ 2 \frac{\partial u_2}{\partial x} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$pr = \begin{bmatrix} u_1 - 1 \\ 0 \end{bmatrix}$$

$$qr = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Ans:

Step 2: Edit the PDE function file:

```
function [c, f, s]=ex5_2_2pdefun(x, t, u, dudx)
c=[1 1]';
f=[1 2]'.*dudx;
y=u(1)-u(2);
G=exp(5*y)-exp(-10*y);
s=[-G G] ';
```

Step 3: Edit the initial condition file:

```
function u0=ex5_2_2ic(x)
u0=[1 0]';
```

Ans:

Step 4: Edit the boundary condition file:

```
function [pl, ql, pr, qr]=ex5_2_2 bc(xl, ul, xr, ur, t)
pl=[0 u1(2)]'; % NOTE: u1(2) represents u2 at the left
boundary.
ql=[1 0]';
pr=[ur(1)-1 0]'; % NOTE: ur(1) represents u1 at the
right boundary.
qr=[0 1] ';
```

Step 5:Take mesh points:

```
x=[0 .005 .01 .05 .1 .2 .5 .7 .9 .95 .99 .995 1];
t=[0 .005 .01 .05 .1 .5 1 1.5 2];
```

ex5_2_2.m

```
function ex5_2_2
%
% Example 5-2-2 Solution of simultaneous PDEs
%
clear; close all;
clc
%
m=0;
x=[0 .005 .01 .05 .1 .2 .5 .7 .9 .95 .99 .995 1];
t=[0 .005 .01 .05 .1 .5 1 1.5 2];
%
% PDE solution with pdepe
%
sol=pdepe(m, @ex5_2_2pdefun, @ex5_2_2ic, @ex5_2_2bc, x, t);
u1=sol(:, :, 1); % u1
u2=sol(:, :, 2); % u2
%
% Results plotting
%
```

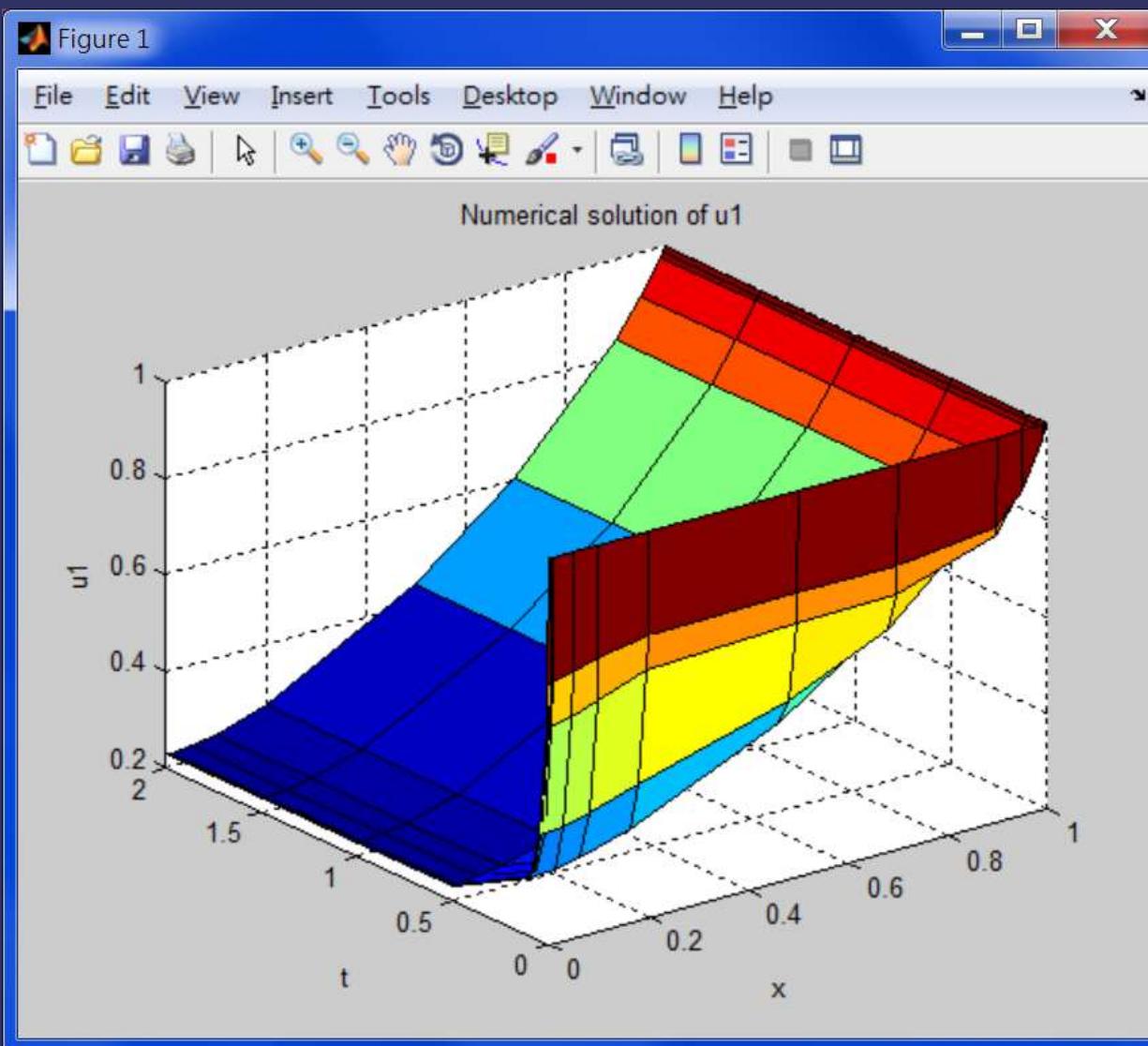
ex5_2_2.m

```
figure(1)
surf(x, t, u1)
title('Numerical solution of u1')
xlabel('x')
ylabel('t')
zlabel('u1')
%
figure(2)
surf(x, t, u2)
title('Numerical solution of u2')
xlabel('x')
ylabel('t')
zlabel('u2')
%
% PDE equations
%
function [c, f, s]=ex5_2_2pdefun(x, t, u, dudx)
c=[1 1]';
f=[1 2].*dudx;
```

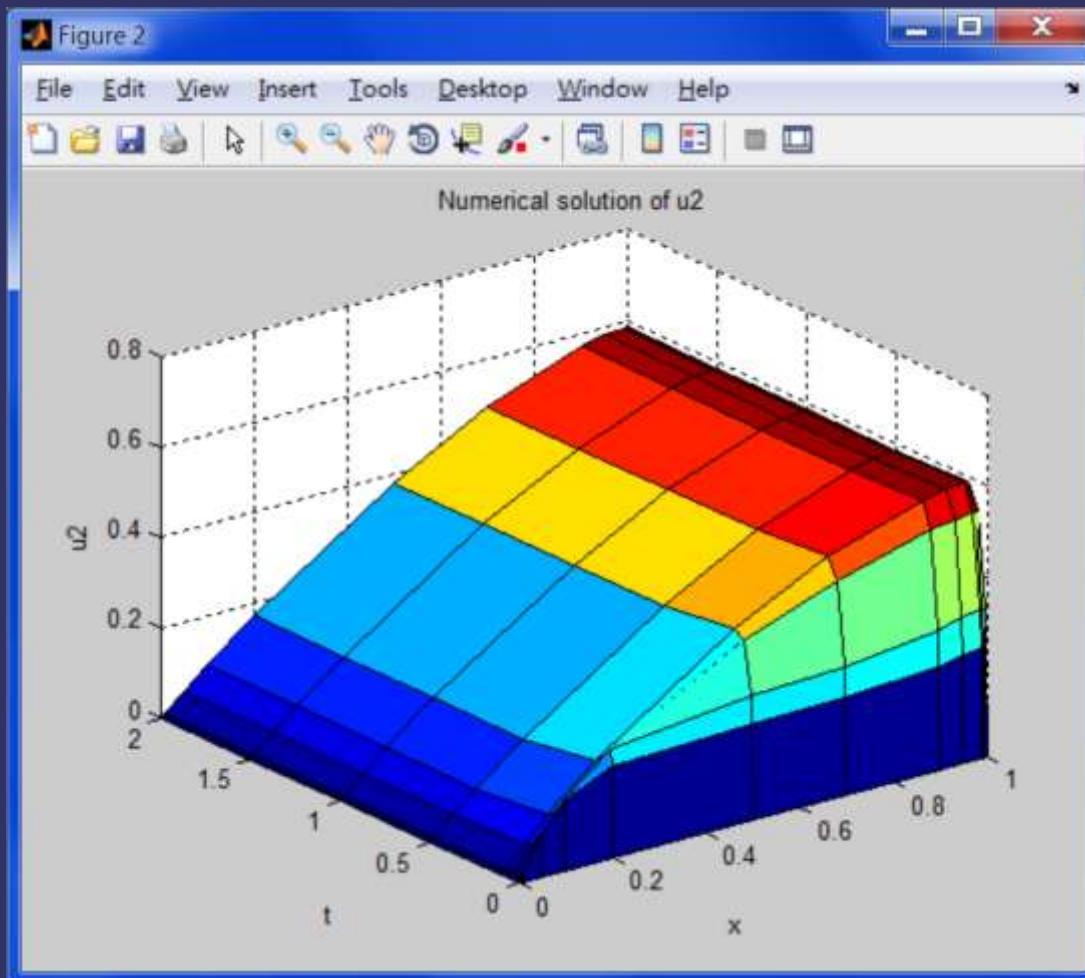
ex5_2_2.m

```
y=u(1)-u(2);
G=exp(5*y)-exp(-10*y);
s=[-G G]';
%
% Initial condition
%
function u0=ex5_2_2ic(x)
u0=[1 0]';
%
% Boundary value conditions
%
function [pl, ql, pr, qr]=ex5_2_2bc(xl, ul, xr, ur, t)
pl=[0 ul(2)]'; % NOTE: u1 (2) represents u2 at the left boundary
ql=[1 0]';
pr=[ur(1)-1 0]'; % NOTE: ur(1) represents ul at the right boundary
qr=[0 1]';
```

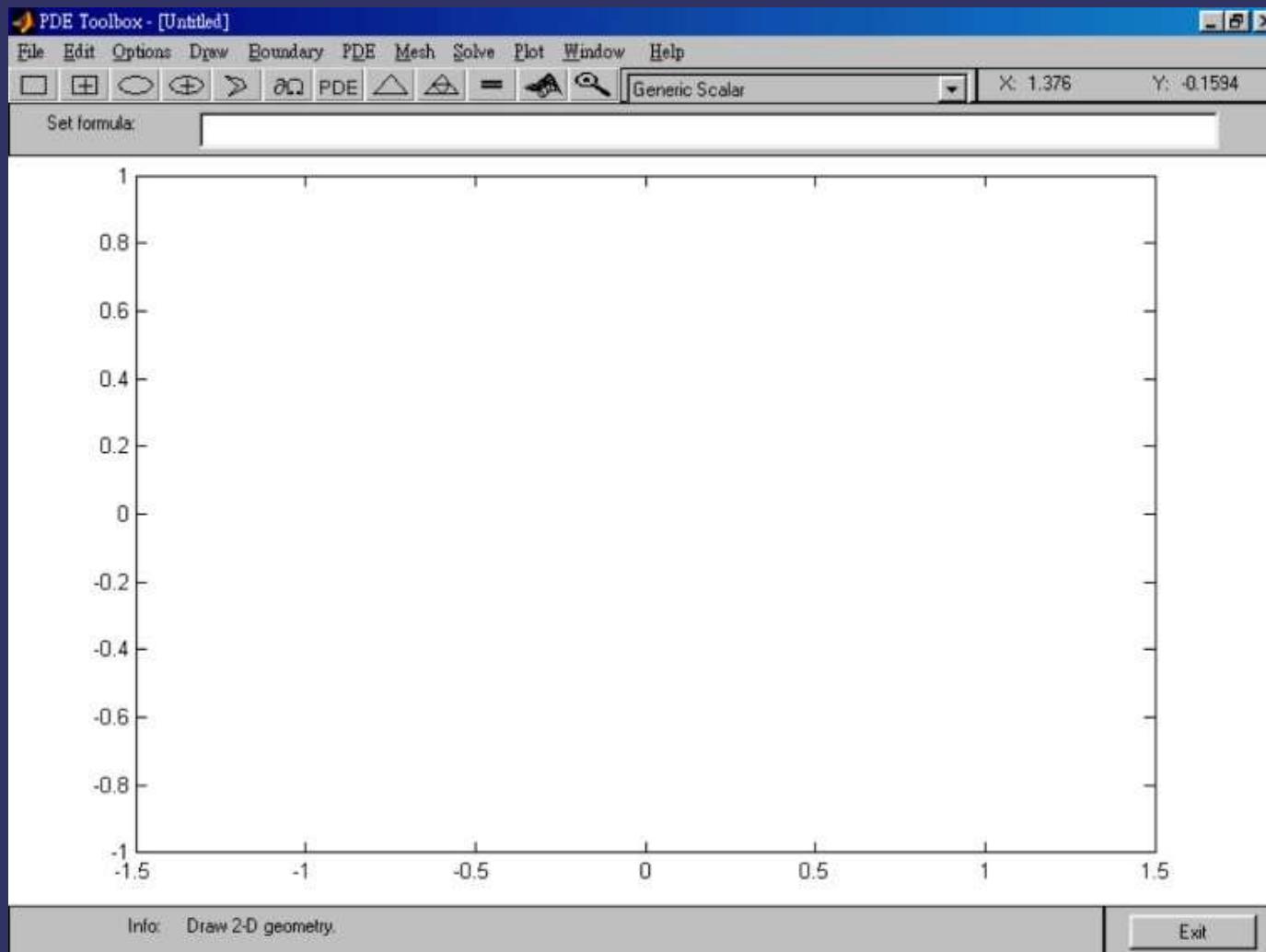
Execution results:



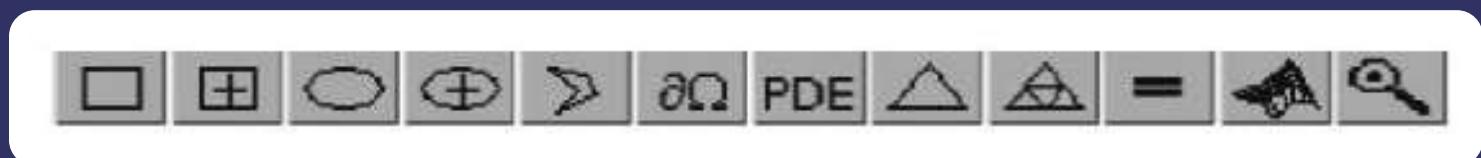
Execution results:



5.2.2 The PDE graphical interface toolbox



- three main steps:
 - 1) Define the PDE problem, including the two-dimensional domain, boundary conditions, and the PDE coefficients.
 - 2) Discretize the solution domain, which produces the discretized meshes for the PDE.
 - 3) Solve the PDE by *finite element methods* (FEMs) and display the solutions.



Icons

Function description



Construct a rectangle or a square through diagonal plotting.
Hold the left button on the mouse to plot a rectangle and hold the right button to plot a square.



Plot a rectangle or a square starting from the central point to a corner. Likewise, hold the left button on the mouse to plot a rectangle and hold the right button to plot a square.



Plot an ellipse or a circular area by means of boundary lines. Using the left mouse button to plot an ellipse and the right mouse button to plot a circle.



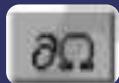
Plot an ellipse or a circle from the central point to the outwards of the shape. Likewise, using the left mouse button to plot an ellipse and the right mouse button to plot a circle.



Used for plotting polygonal or irregular areas; right click the mouse to close this function.

Icons

Function description



Used to give boundary conditions. After this icon is selected, users can left click twice at the boundary of the domain and then set the boundary conditions in the pop-up dialogue box.



Used to designate the PDE equation and the relevant parameters.



Used to produce the discretized meshes in the solution domain.



Used for further refining the discretized meshes.



After the domain, the PDE equation, the boundary conditions, and the meshes are assigned, click this icon to start solving the PDE.



Used to designate how the results are plotted and displayed.



Zoom in and out to facilitate results plotting and display.

5.2.2.1 The solvable PDE problem patterns

$$-\nabla \cdot (c\nabla u) + au = f$$

- here $\nabla \equiv \partial/\partial x \vec{i} + \partial/\partial y \vec{j}$ is the *Laplacian operator*, and c, a, f , and u are scalars or *complex-valued functions* defined in the solution domain.

Problem description	Governing equation	Parameter settings			
		<i>u</i>	<i>c</i>	<i>a</i>	<i>f</i>
Flow in porous medium	$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = 0$	<i>P</i>	1	0	0
Potential flow	$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0$	<i>ψ</i>	1	0	0
Two-dimensional steady-state heat transfer	$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$	<i>T</i>	1	0	0
Steady-state mass transfer	$-\nabla \cdot (D \nabla C) = Q_m$	<i>C</i>	<i>D</i>	0	<i>Q_m</i>
Steady-state heat transfer	$-\nabla \cdot (k \nabla T) = Q + h(T_{sur} - T)$	<i>T</i>	<i>k</i>	<i>h</i>	<i>Q</i> + <i>hT_{sur}</i>

(1) Parabolic PDE

$$d \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u) + au = f$$

Problem description	Governing equation	Parameter settings				
		u	d	c	a	f
Non-steady-state heat transfer	$\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q + h(T_{sur} - T)$	T	ρC_p	k	h	$Q + hT_{sur}$
Two-dimensional non-steady-state mass transfer	$\frac{\partial C}{\partial t} - \nabla \cdot (D \nabla C) = Q_m$	C	I	D	0	Q _m
One-dimensional non-steady-state mass transfer	$\frac{\partial C}{\partial t} = D_{AB} \frac{\partial^2 C}{\partial x^2}$	C	I	D_{AB}	0	0

(2) Hyperbolic PDE

$$d \frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c \nabla u) + au = f$$

$$\rho \frac{\partial^2 u}{\partial t^2} = \tau \frac{\partial^2 u}{\partial x^2} + f(x, t)$$

(3) Eigenvalue problem

$$-\nabla \cdot (c \nabla u) + au = \lambda du$$

where λ is the unknown eigenvalue to be determined.

(i) Dirichlet condition

$$hu = r$$

(ii) Generalized Neumann condition

$$\vec{n} \cdot (c \nabla u) + qu = g$$

5.2.2.2 Solution of PDE problems with the pdetool interface

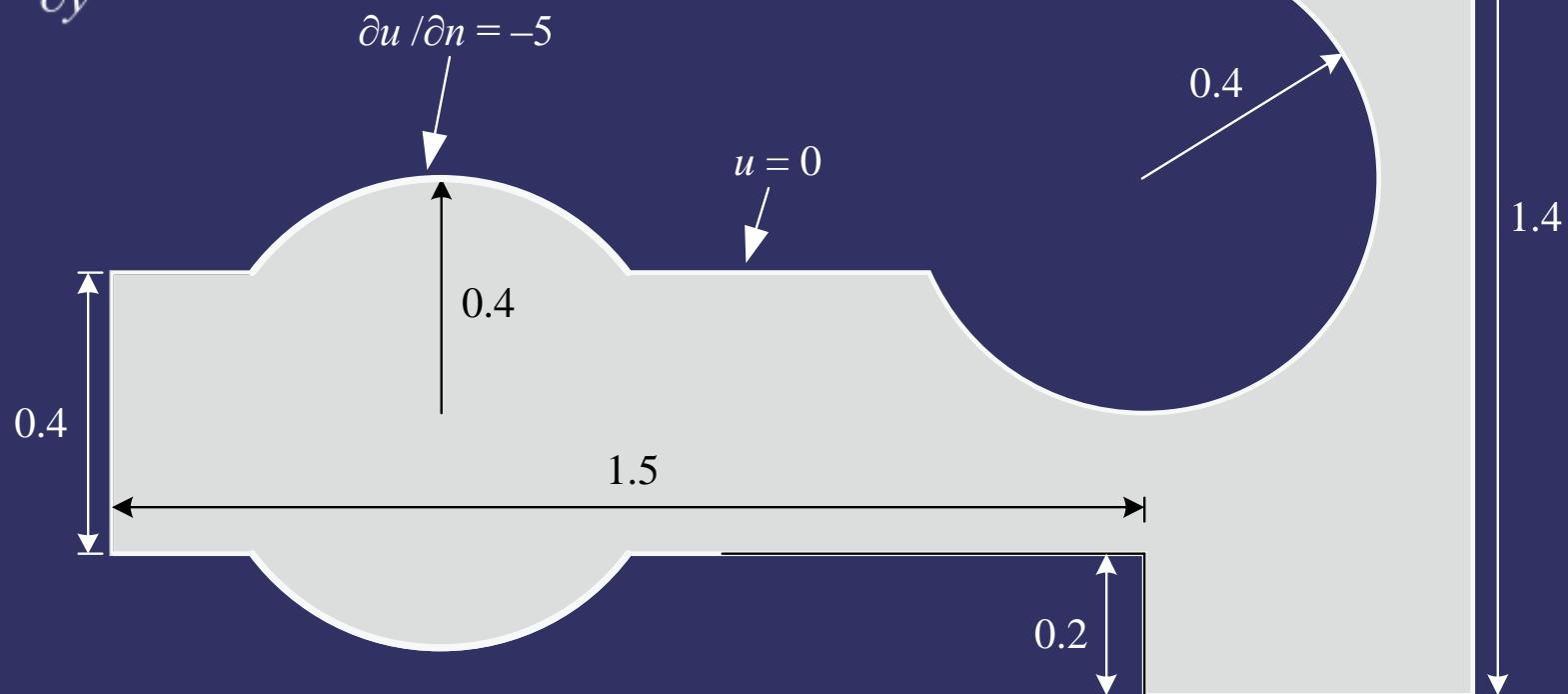
1. Use the *Draw* mode to define the problem domain.
 2. Use the PDE mode to assign the PDE coefficients; that is, one needs to assign the coefficients, such as c , a , f , and d , in the PDE mode.
 3. Use the Boundary mode to set up the boundary conditions.
- After defining the PDE problem, one can solve it through the following steps:
 - 1) In Mesh mode, produce meshes to discretize the solution domain.
 - 2) In Solve mode, obtain the PDE solutions.
 - 3) Finally, display the solutions in the Plot mode.

Example 5-2-3

Solution of a two-dimensional elliptic PDE with pdetool

two-dimensional steady-state elliptic PDE

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -10.$$

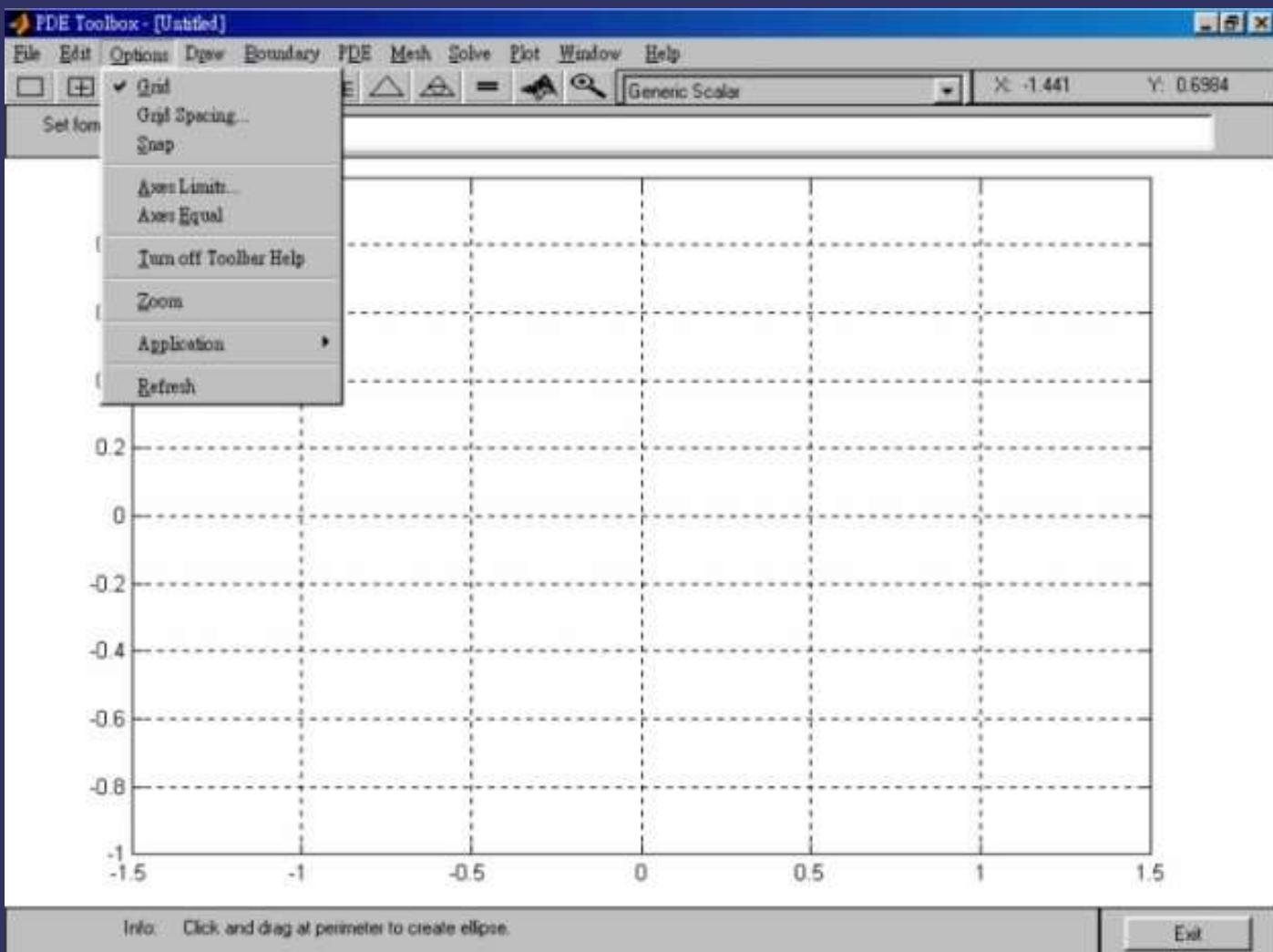


It is assumed that, at the curved portion (arc sections), the boundary conditions are of the Neumann type with $\frac{\partial u}{\partial n} = 0$. Besides, the boundary conditions located in the straight line sections are of the Dirichlet type; that is, $u = 0$. Now, use the pdetool graphical interface to solve the PDE equation and display the results by plotting a 3D diagram.

Example 5-2-3

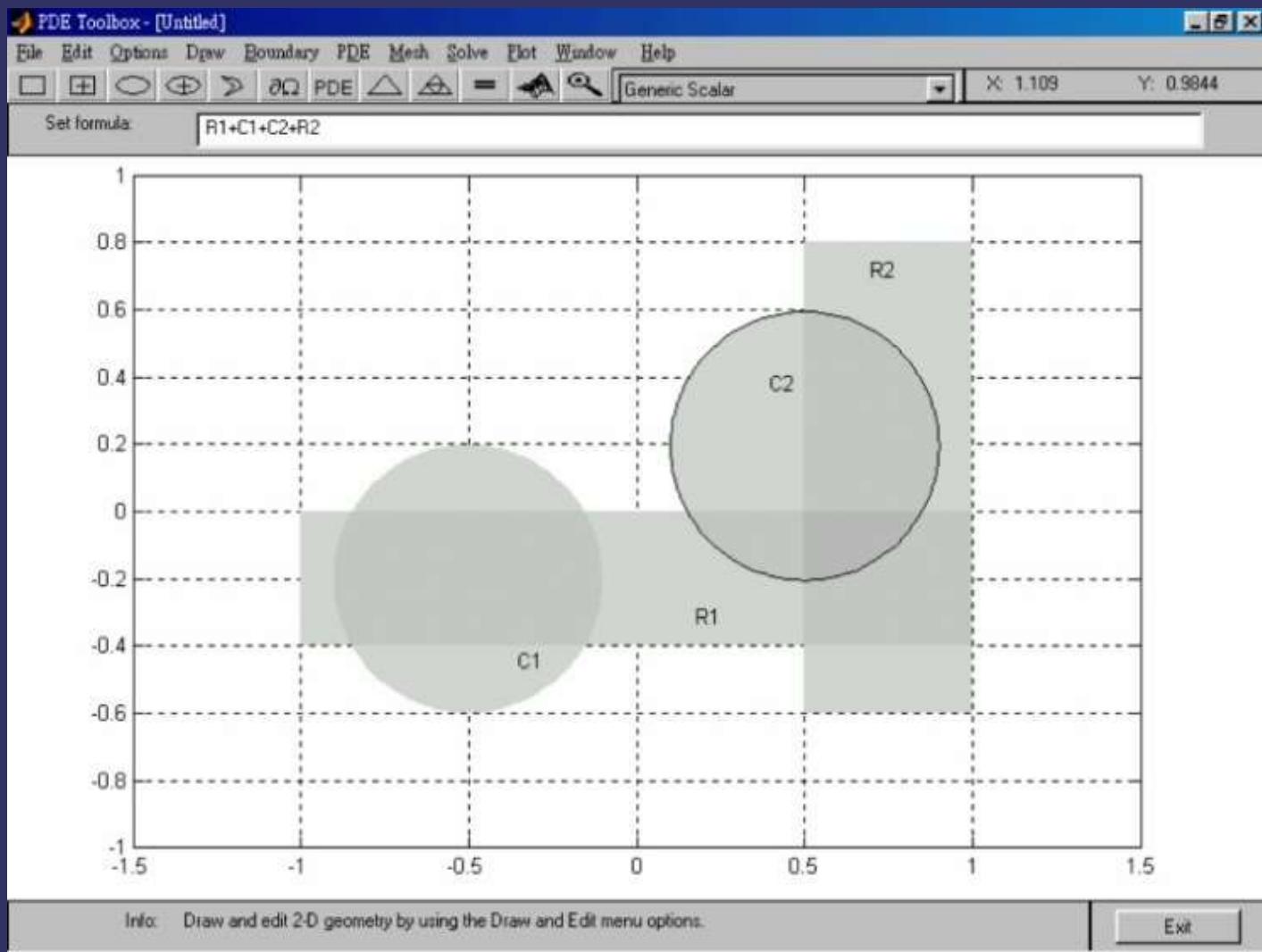
Ans:

Step 1:



Ans:

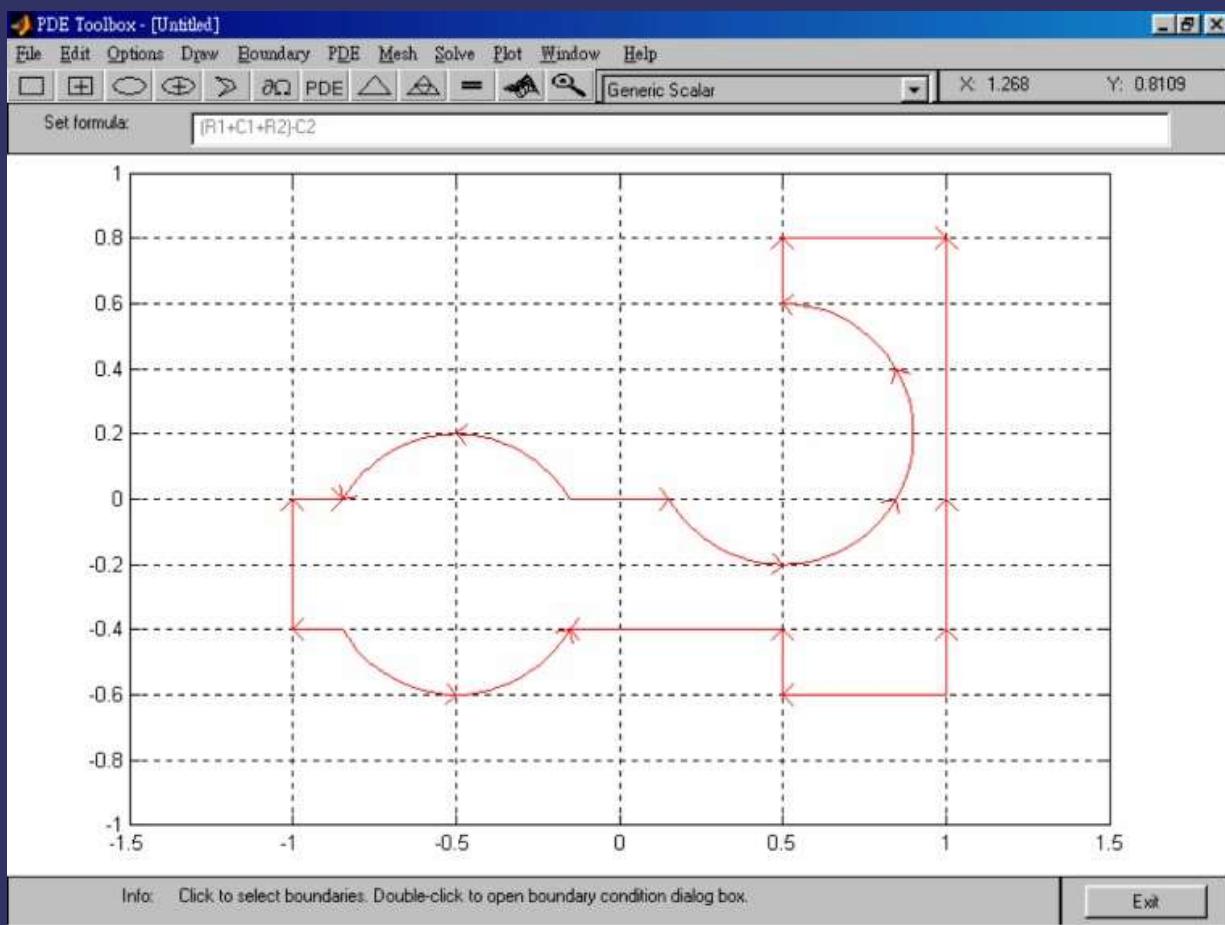
Step 2:



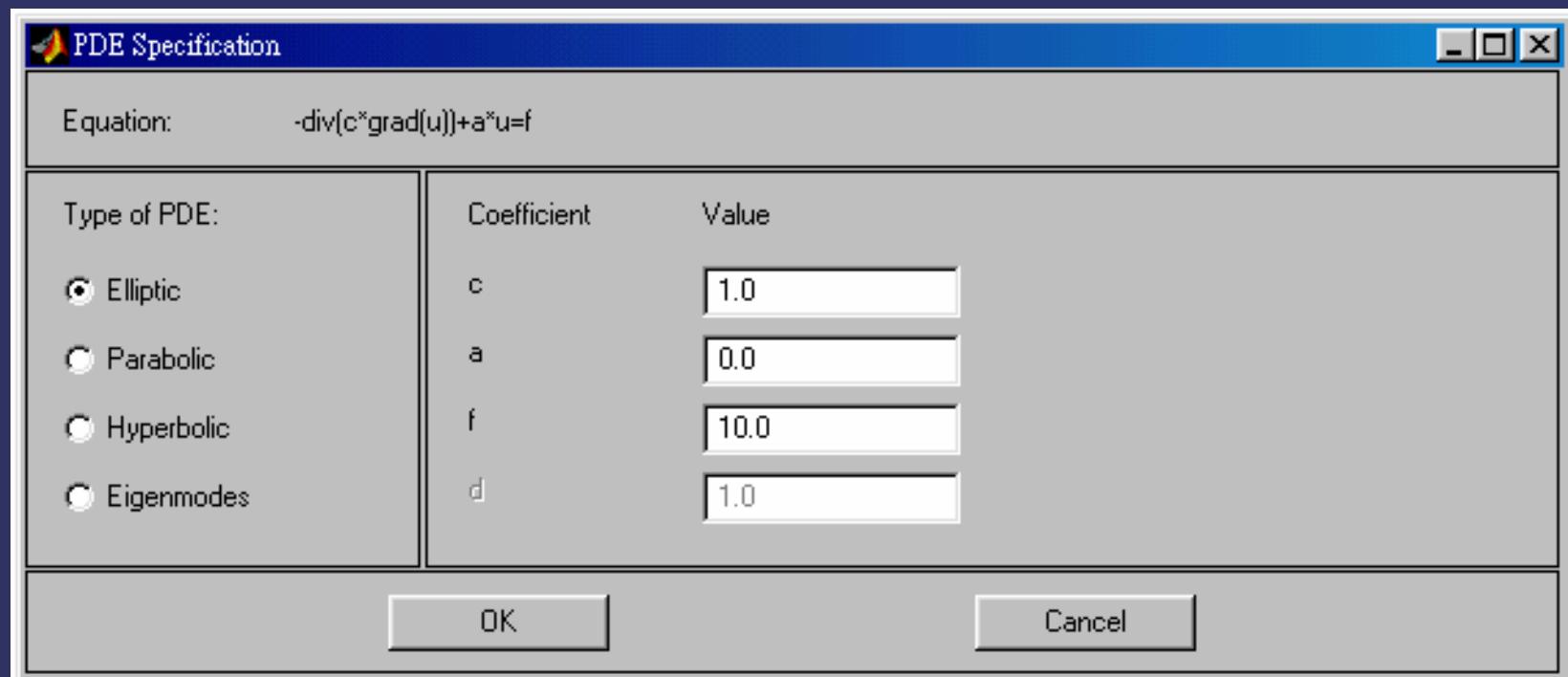
Ans:

Step 2:

type the algebraic formula $R1 + C1 + R2 - C2$ into the Set formula box.

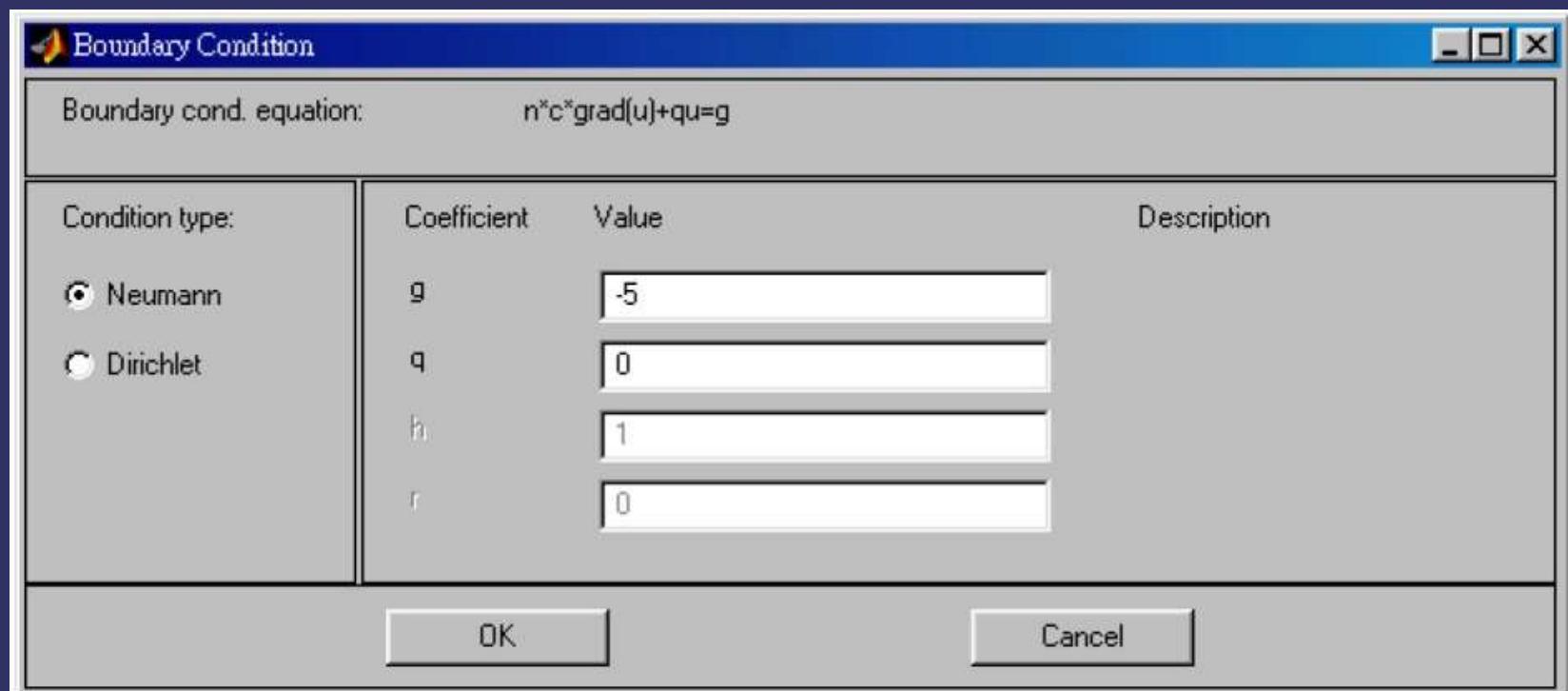


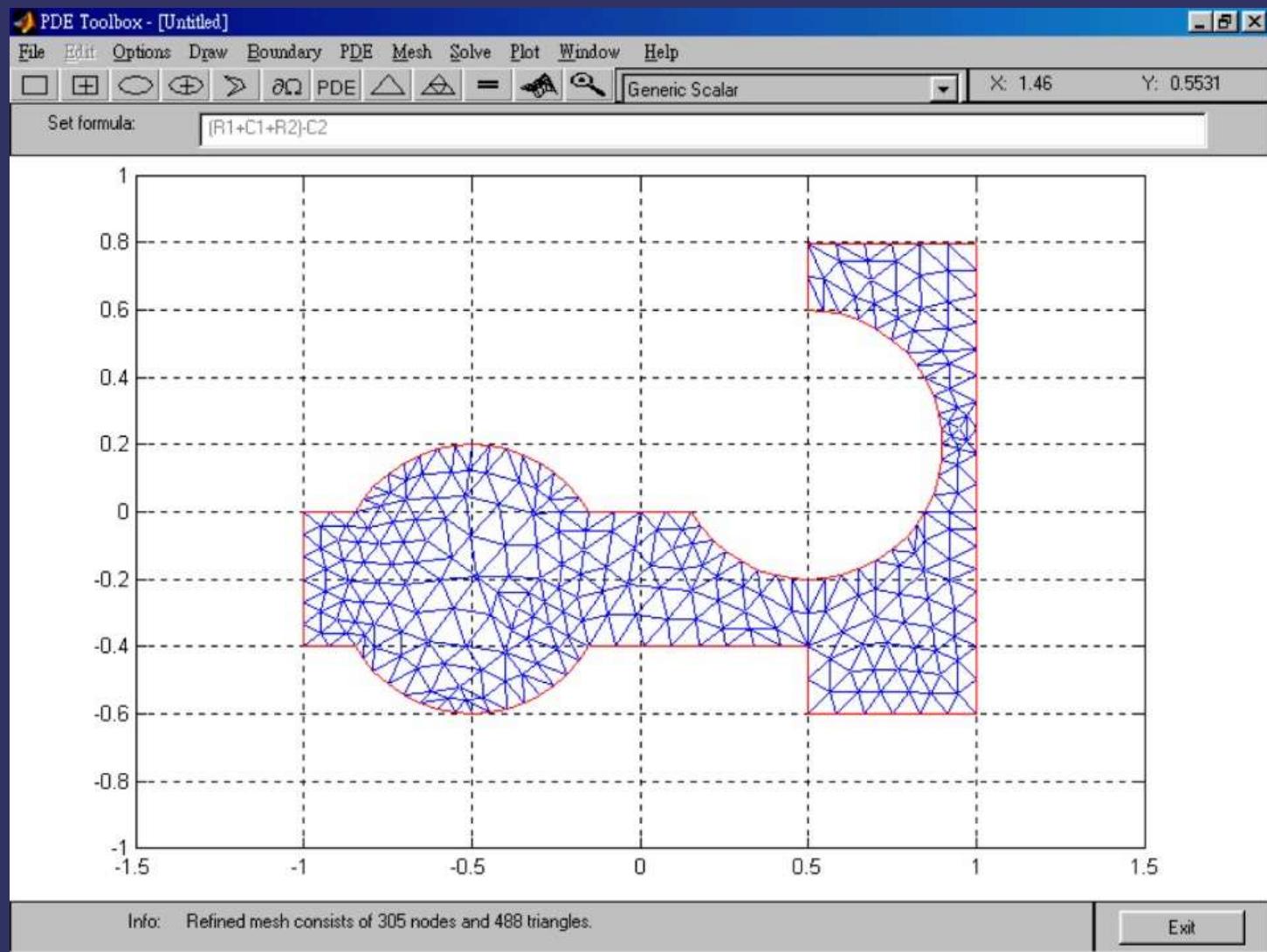
Ans:



Ans:

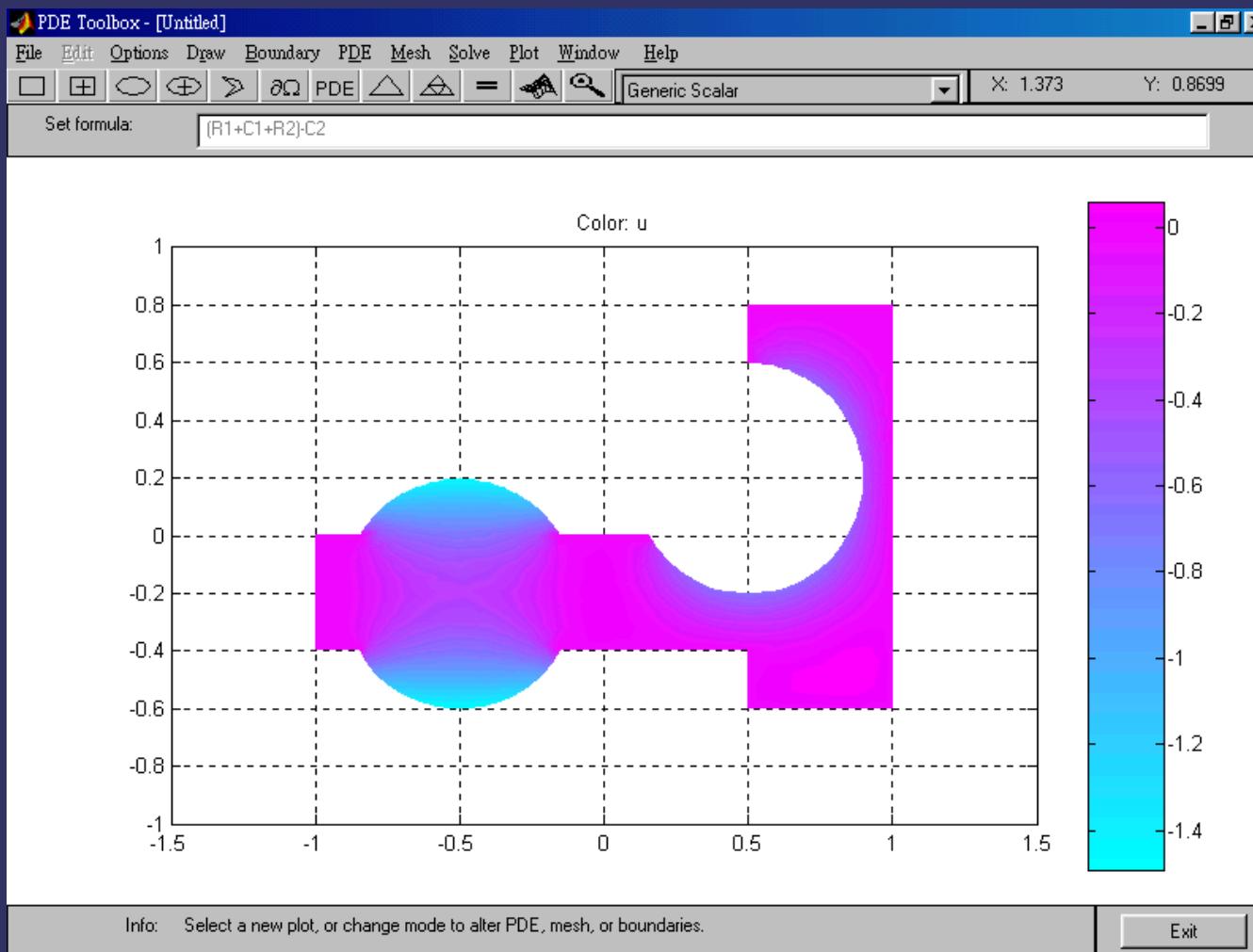
Step 4:

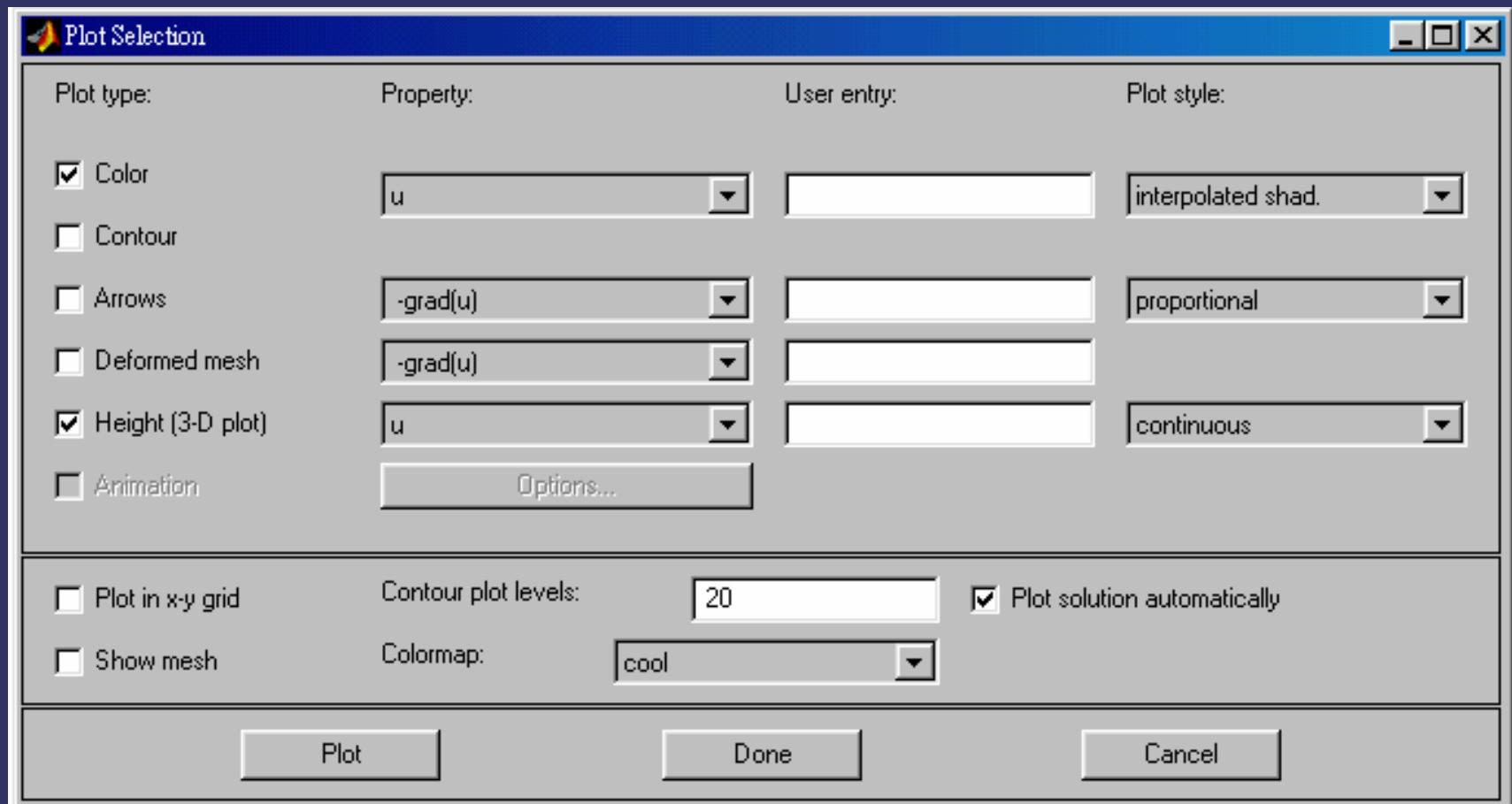


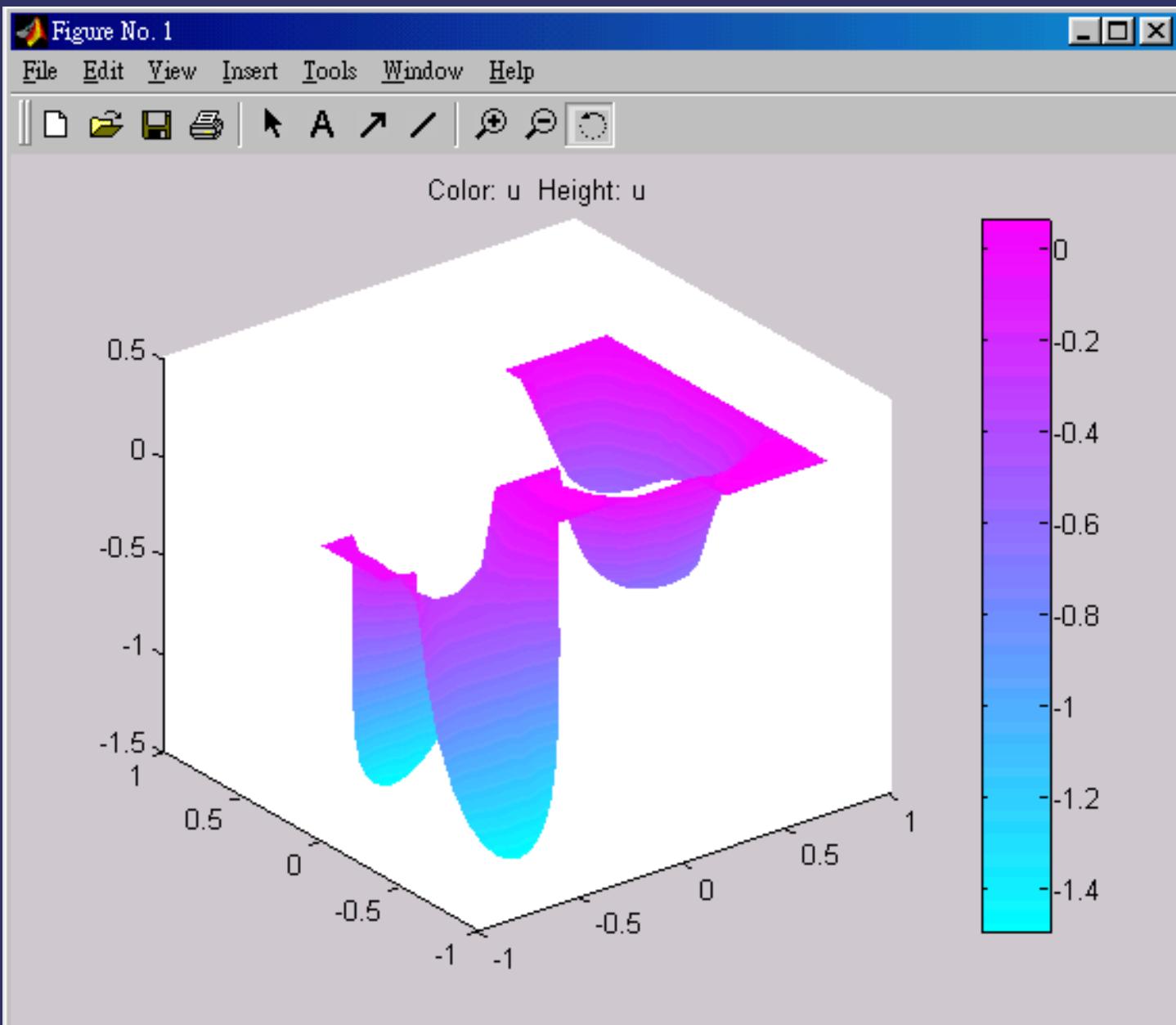
Ans:**Step 5:**

Ans:

Step 6: Select  to solve the PDE.



Ans:**Step 6:**

Ans:**Step 6:**



5.3 Chemical engineering examples

Example 5-3-1

Temperature and reaction rate distributions in a catalyzed reactor

$$-\frac{\partial T}{\partial L} + \frac{k_e}{GC_p} \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right) + \frac{r_A \rho_B (-\Delta H_r)}{GC_p} = 0$$

$$-\frac{\partial x}{\partial L} + \frac{D_e}{u} \left(\frac{\partial^2 x}{\partial r^2} + \frac{1}{r} \frac{\partial x}{\partial r} \right) + \frac{r_A \rho_B M_{av}}{Gy_0} = 0$$

$$\left\{ \begin{array}{l} T|_{L=0} = T_0(r), \quad x|_{L=0} = x_0(r) \\ \frac{\partial T}{\partial r}|_{r=0} = \frac{\partial x}{\partial r}|_{r=0} = 0 \\ -k_e \frac{\partial T}{\partial r}|_{r=r_w} = h_w(T - T_w) \\ \frac{\partial x}{\partial r}|_{r=r_w} = 0 \end{array} \right.$$

(i) The reaction rate equation



Benzene hydrogen cyclohexane



$$r_A = \frac{k K_H^3 K_B P_H^3 P_B}{(1 + K_H P_H + K_B P_B + K_C P_C)^4}$$

$$\ln k = 12,100 / (RT) + 32.3 / R$$

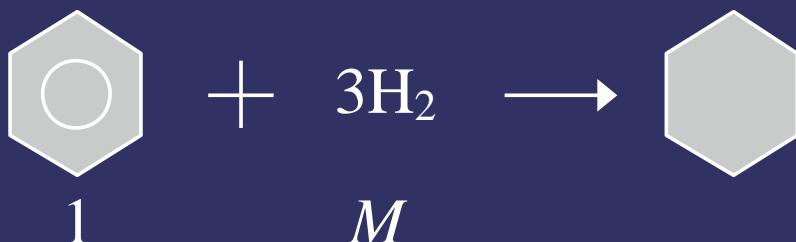
$$\ln K_H = 15,500 / (RT) - 31.9 / R$$

$$\ln K_B = 11,200 / (RT) - 23.1 / R$$

$$\ln K_C = 8,900 / (RT) - 19.4 / R$$

Parameter	Physical meaning	Value	Unit
P_t	Total pressure	1.25	atm
r_w	Diameter of the reaction tube	2.5	cm
T_w	Wall temperature	100	°C
G	Mass flow rate	630	kg/m ² .hr
M	The ratio of the molar flow rate of hydrogen to that of benzene	30	
y_0	Molar fraction of benzene at the inlet of the reaction tube	0.0325	
M_{av}	Average molecular weight of the reaction gas	4.45	
ρ_B	Density of the catalyst	1,200	kg/m ³
C_P	Average specific heat of the liquid	1.75	kcal/kg-mol
ΔH_r	Heat of reaction	-49,250	kcal/kg-mol
h_0	Overall heat transfer coefficient	65.5	kcal/m ² .hr.°C
$T(0)$	The temperature of the feed	125	°C
L_w	Reaction tube length	1	m
u	Flow rate	8.05	m/hr
k_e	Effective heat transfer coefficient	0.65	kcal/m ² .hr.°C
h_w	Heat transfer coefficient of wall film	112	kcal/m ² .hr.°C
D_e	Effective diffusivity	0.755	m ² /hr

Problem formulation and analysis:



$$\frac{-x}{1-x} \quad \frac{-3x}{M-3x} \quad \frac{x}{x}$$

$$P_H = P_t \frac{M - 3x}{1 + M - 3x}$$

$$P_B = P_t \frac{1 - x}{1 + M - 3x}$$

$$P_C = P_t \frac{x}{1 + M - 3x}$$

MATLAB program design:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}_* \begin{bmatrix} \frac{\partial T}{\partial L} \\ \frac{\partial x}{\partial L} \end{bmatrix} = r^{-1} \frac{\partial}{\partial r} \begin{bmatrix} r \frac{k_e}{GC_P} \frac{\partial T}{\partial r} \\ r \frac{D_e}{u} \frac{\partial x}{\partial r} \end{bmatrix} + \begin{bmatrix} \frac{r_A \rho_B (-\Delta H_r)}{GC_P} \\ \frac{r_A \rho_B M_{av}}{Gy_0} \end{bmatrix}$$

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$f = \begin{bmatrix} \frac{k_e}{GC_P} \frac{\partial T}{\partial r} \\ \frac{D_e}{u} \frac{\partial x}{\partial r} \end{bmatrix}$$

MATLAB program design:

$$S = \begin{bmatrix} \frac{r_A \rho_B (-\Delta H_r)}{G C_p} \\ \frac{r_A \rho_B M_{av}}{G y_0} \end{bmatrix}$$

and $m = 1$ (cylinder). (at $r = 0$)

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}_* f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

MATLAB program design:

$$pl = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$ql = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(at $r = r_w$)

$$\begin{bmatrix} h_w(T - T_w) \\ 0 \end{bmatrix} + \begin{bmatrix} GC_P \\ 1 \end{bmatrix} * f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$pr = \begin{bmatrix} h_w(T - T_w) \\ 0 \end{bmatrix}$$

$$qr = \begin{bmatrix} GC_P \\ 1 \end{bmatrix}$$

MATLAB program design:

— ex5_3_1.m —

```
function ex5_3_1
%
% Example 5-3-1 Temperature and reaction rate distributions in a catalyzed reactor
%
clear; close all;
clc
%
global Pt rw Tw G M y0 Mav rho_B Cp dHr h0 u R ke hw De
%
% Given data
%
Pt=1.25; % Total pressure (atm)
rw=0.025; % Tube diameter (m)
Tw=100+273; % Wall temperature (°C)
G=630; % Mass flow rate (kg/m^2.hr)
M=30; % Molar flow ratio between hydrogen and benzene
y0=0.0325; % Molar fraction of benzene at the inlet of the reaction tube
Mav=4.45; % Average molecular weight of the reaction gas
```

MATLAB program design:

— ex5_3_1.m —

```
rho_B=1200; % Density at the catalyst (kg/m^3)
```

```
Cp=1.75; % Average specific heat capacity of the fluid (kcal/kg-mol)
```

```
dHr=-49250; % Reaction heat (kcal/kg-mol)
```

```
h0=65.5; % Overall heat transfer coefficient (kcal/m^2.hr.°C)
```

```
T0=125+273; % Feeding temperature (K)
```

```
Lw=1; % Reaction tube length (m)
```

```
u=8.05; % Flow rate (m/hr)
```

```
R=1.987; % Ideal gas constant (cal/mol.K)
```

```
ke=0.65; % Effective heat transfer coefficient (kcal/m.hr.°C)
```

```
hw=112; % Wall film heat transfer coefficient (kcal/m^2.hr.°C)
```

```
De=0.755; % Effective diffusivity coefficient (m^2/hr)
```

```
%
```

```
m=1; % Cylindrical symmetry
```

```
%
```

```
% Mesh
```

```
%
```

```
r=linspace(0, rw, 10);
```

```
L=linspace(0, Lw, 10);
```

MATLAB program design:

— ex5_3_1.m —

```
%  
% Solving through pdepe  
%  
sol=pdepe(m, @ex5_3_1pdefun, @ex5_3_1ic, @ex5_3_1bc, r, L);  
T=sol(:, :, 1); % Temperature  
x=sol(:, :, 2); % Conversion rate  
%  
% Results plotting  
%  
figure(1)  
surf(L, r, T'-273)  
title('Temperature distribution in the reactor')  
xlabel('Tube length (m)')  
ylabel('Radial position (m)')  
zlabel('Temperature (0C)')  
axis([0 1 0 0.025 100 280])  
%
```

MATLAB program design:

— ex5_3_1.m —

```
figure(2)
surf(L, r, x')
title('Conversion rate distribution')
xlabel('Tube length (m)')
ylabel('Radial position (m)')
zlabel('Conversion rate')
axis([0 1 0 0.025 0 1])
%
% PDE equation
%
function [c, f, s]=ex5_3_1pdefun(r, L, u1, dudr)
global Pt rw Tw G M y0 Mav rho_B Cp dHr h0 u R ke hw De
T=u1(1);
x=u1(2);
%
k=exp(-12100/(R*T)+32.3/R);
KH=exp(15500/(R*T)-31.9/R);
```

MATLAB program design:

— ex5_3_1.m —

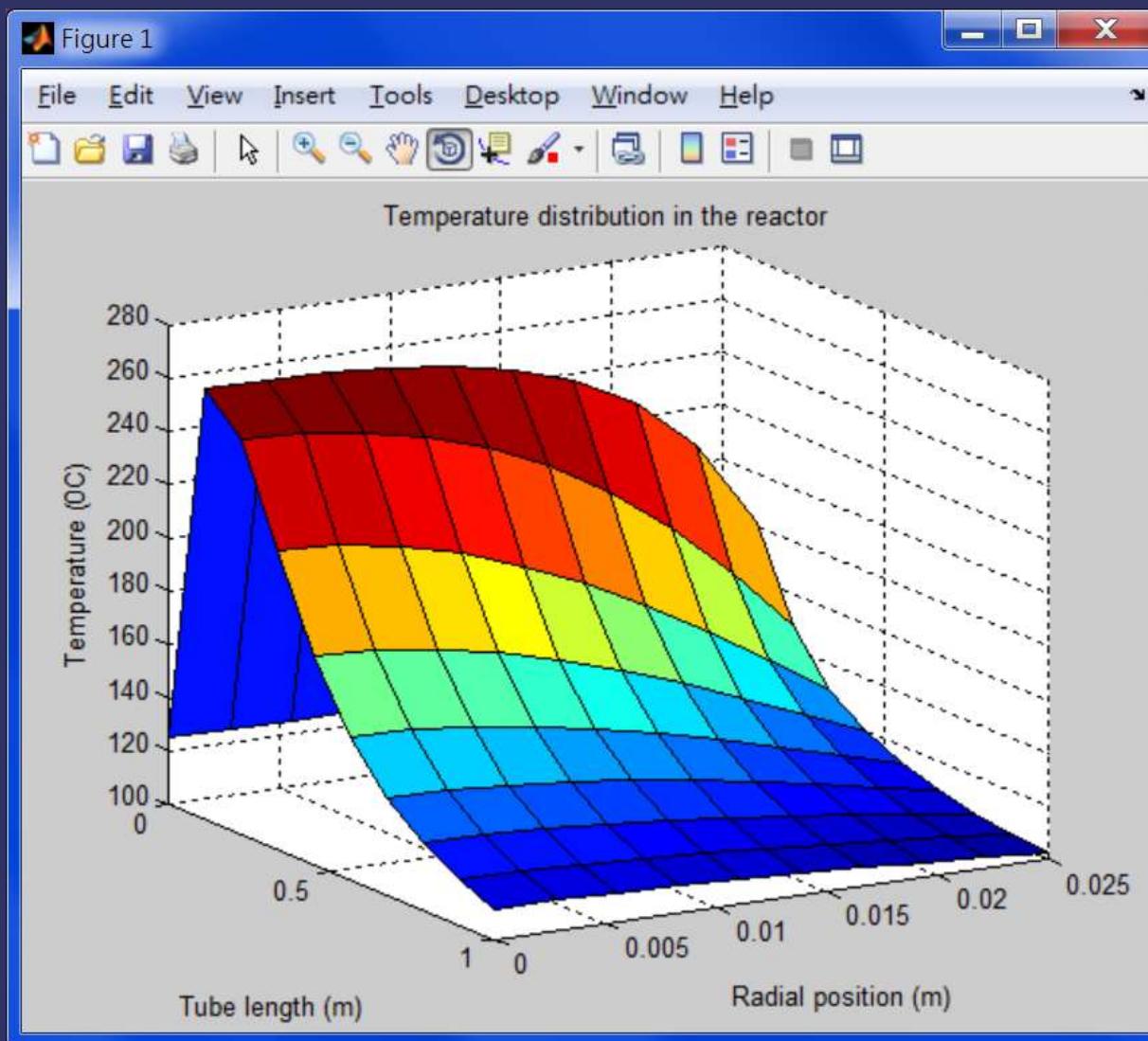
```
KB=exp(11200/(R*T)-23.1/R);
KC=exp(8900/(R*T)-19.4/R);
%
a=1+M-3*x;
PH=Pt*(M-3*x)/a;
PB=Pt*(1-x)/a;
PC=Pt*x/a;
%
rA=k*KH^3*KB*PH^3*PB/(1+KH*PH+KB*PB+KC*PC)^4;
%
c=[1 1]';
f=[ke/(G*Cp) De/u].*dudr;
s=[rA*rho_B*(-dHr)/(G*Cp)
    rA*rho_B*Mav/(G*y0)];
%
% Initial condition
%
```

MATLAB program design:

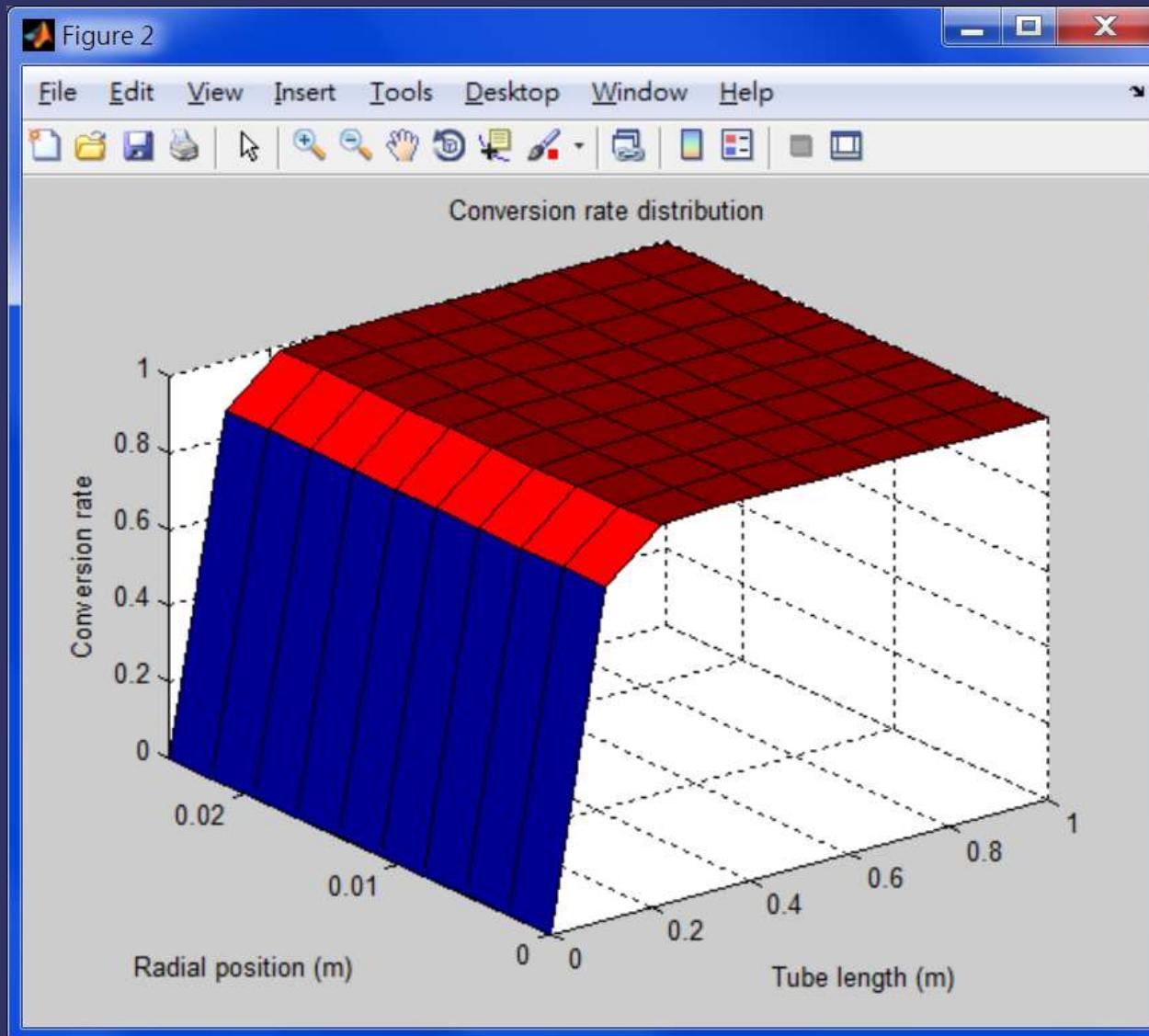
— ex5_3_1.m —

```
function u0=ex5_3_1ic(r)
u0=[125+273  0]';
%
% Boundary conditions
%
function [pl, ql, pr, qr]=ex5_3_1bc(rl, ul, rr, ur, L)
global Pt rw Tw G M y0 Mav rho_B Cp dHr h0 u R ke hw De
pl=[0  0]';
ql=[1  1]';
pr=[hw*(ur(1)-Tw)  0]';
qr=[G*Cp  1]';
```

Execution results:



Execution results:



Example 5-3-2

Concentration distribution in a diffusive reaction system

Consider the device as shown in Figure 5.2 (*Constantinides and Mostoufi*, 1999), where the stationary liquid *B* stored in the tube is exposed to the surroundings filled with gas *A*. The concentration of gas *A* is $C_{A0} = 0.1 \text{ mol/m}^3$ and its value remains unchanged during the operation time of 10 days. Besides, the diffusivity of gas *A* in liquid *B* is estimated to be $D_{AB} = 2.5 \times 10^{-9} \text{ m}^2/\text{s}$, and the liquid level of the tube is as high as $L = 10 \text{ cm}$. Based on the above, determine the *flux* of gas *A* diffused in liquid *B* under the following two conditions:

- (1) *A* does not react with *B*.

(2) A and B reacts and obeys the first-order reaction kinetics as follows:



where the reaction rate constant is $k = 3.6 \times 10^{-6} \text{ s}^{-1}$.

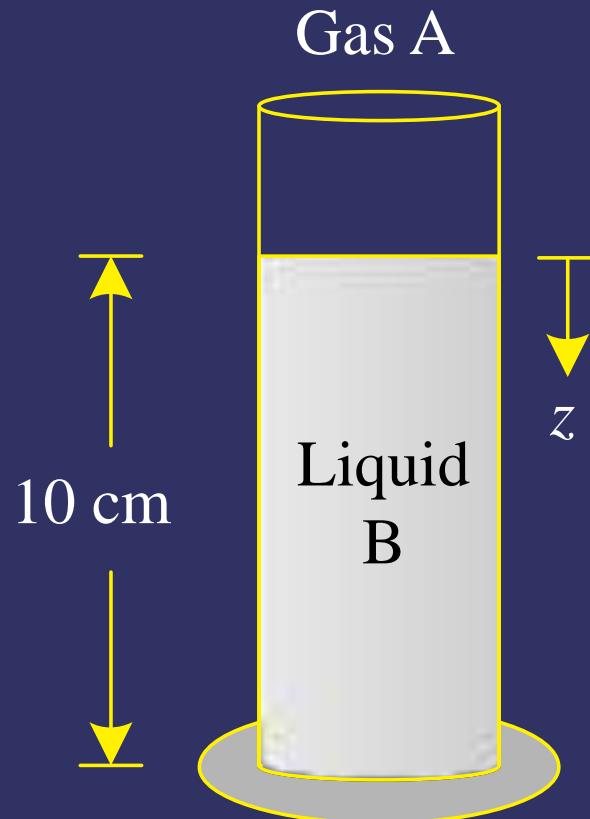


Figure 5.2
The diffusion of gas A in a stationary liquid B.

Problem formulation and analysis:

(1) Since the gas A does not react with the liquid B ,

$$\frac{\partial C_A}{\partial t} = D_{AB} \frac{\partial^2 C_A}{\partial z^2}$$

Initial condition: $C_A|_{t=0} = 0, \quad z > 0$

Boundary conditions: $\begin{cases} C_A|_{z=0} = C_{A0}, & t \geq 0 \\ \frac{\partial C_A}{\partial z}|_{z=L} = 0, & t \geq 0 \end{cases}$

Problem formulation and analysis:

(2) If the first-order reaction occurs,

$$\frac{\partial C_A}{\partial t} = D_{AB} \frac{\partial^2 C_A}{\partial z^2} - k C_A$$

Fick's law

$$N_{Az}(t) = -D_{AB} \frac{\partial C_A}{\partial z} \Big|_{z=0}$$

MATLAB program design:

(1) By comparing (5.3-17) with the standard form (5.2-1), one has $c = 1$, $s = 0$, $f = D_{AB} \partial C_A / \partial z$, and $m = 0$.

(i) at the left boundary ($z = 0$)

$$pl = C_A(0, t) - CA_0$$

$$ql = 0$$

(ii) at the right boundary ($z = L$)

$$pr = 0$$

$$qr = \frac{1}{D_{AB}}$$

MATLAB program design:

- (2) The comparison of (5.3-20) with the standard form (5.2-1) yields the PDE coefficients as follows: $c = 1$, $f = D_{AB} \partial C_A / \partial z$, $s = -kC_A$, and $m = 0$.
the boundary conditions are the same as those in the case (1).

ex5_3_2.m

```
function ex5_3_2
%
% Example 5-3-2 Concentration distribution in a diffusive reaction
%
clear; close all;
clc
%
global DAB k CA0
%
```

MATLAB program design:

— ex5_3_2.m —

```
%  
% Given data  
%  
CA0=0.1; % Concentration at the interface (mol/m^3)  
L=0.1; % Liquid height (m)  
DAB=2.5e-9; % Diffusivity (m^2/s)  
k=3.6e-6; % Reaction rate constant (s^-1)  
h=10*24*3600; % Reaction time (s)  
%  
% Meshes  
%  
t=linspace(0, h, 101);  
z=linspace(0, L, 10);  
%  
% Case 1: no reaction  
%  
m=0;
```

MATLAB program design:

— ex5_3_2.m —

```
sol=pdepe(m, @ex5_3_2pdefuna, @ex5_3_2ic, @ex5_3_2bc, z, t);
CA1=sol(:, :, 1);
for i=1:length(t)
    [CA_i, dCAdz_i]=pdeval(m, z, CA1(i,:), 0);
    NAz(i)=-dCAdz_i*DAB;
end
%
figure(1)
subplot(211)
surf(z, (t(11:101))/(24*3600),CA1(11:101,:))
title('Case 1: no reaction')
xlabel('length (m)')
ylabel('time (day)')
zlabel('conc. (mol/m^3)')
subplot(212)
plot(t/(24*3600), NAz'*24*3600)
xlabel('time (day)')
```

MATLAB program design:

— ex5_3_2.m —

```
ylabel('flux (mol/m^2.day)')  
%  
figure(2)  
plot(z, CA1(11:10:101,:))  
title('Case 1: no reaction')  
xlabel('length (m)')  
ylabel('conc. (mol/m^3)')  
legend('Day1', 'Day2', 'Day3', 'Day4', 'Day5', 'Day6',...  
'Day7', 'Day8', 'Day9', 'Day10')  
grid on  
%  
% Case 2: first-order reaction  
%  
m=0;  
sol=pdepe(m, @ex5_3_2pdefunb, @ex5_3_2ic, @ex5_3_2bc, z, t);  
CA2=sol(:, :, 1);  
for i=1:length(t)
```

MATLAB program design:

— ex5_3_2.m —

```
[CA_i, dCAdz_i]=pdeval(m, z, CA2(i,:), 0);
NAz(i)=-dCAdz_i*DAB;
end
%
figure(3)
subplot(211)
surf(z, (t(11:101))/(24*3600),CA2(11:101,:))
title('Case 2: first-order reaction')
xlabel('length (m)')
ylabel('time (day)')
zlabel('conc. (mol/m^3)')
subplot(212)
plot(t/(24*3600), NAz'*24*3600)
xlabel('time (day)')
ylabel('flux (mol/m^2.day)')
%
figure(4)
```

MATLAB program design:

— ex5_3_2.m —

```
plot(z, CA2(11:10:101,:))  
title('Case 2: first-order reaction')  
xlabel('length (m)')  
ylabel('conc. (mol/m^3)')  
legend('Day1', 'Day2', 'Day3', 'Day4', 'Day5', 'Day6',...  
'Day7', 'Day8', 'Day9', 'Day10')  
grid on  
%  
% PDE equation  
%  
% Case 1: no reaction  
%  
function [c, f, s]=ex5_3_2pdefuna(z, t, CA, dCAdz)  
global DAB k CA0  
c=1;  
f=DAB*dCAdz;  
s=0;
```

MATLAB program design:

— ex5_3_2.m —

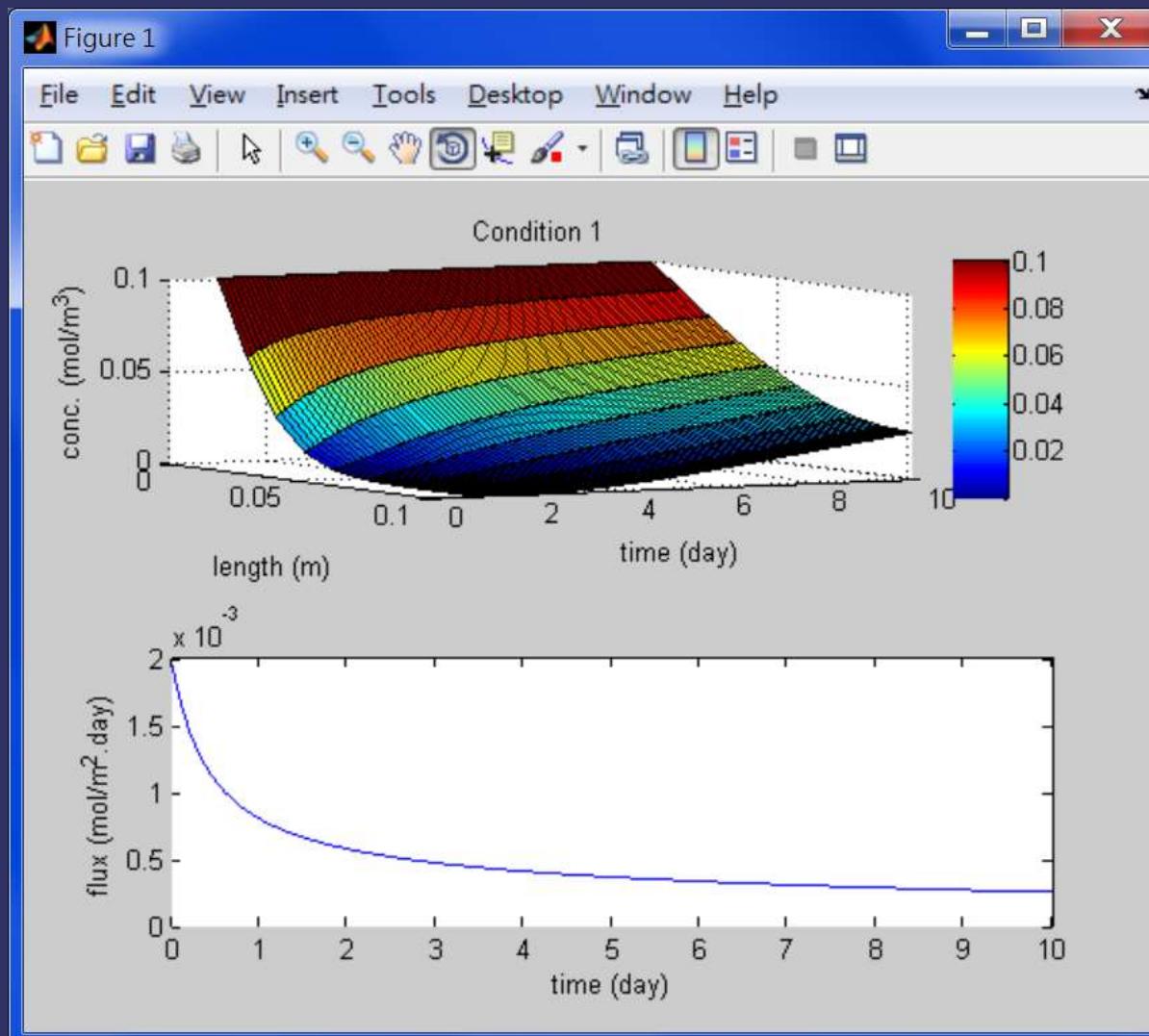
```
%  
% Case 2: first-order reaction  
%  
function [c, f, s]=ex5_3_2pdefunb(z, t, CA, dCAdz)  
global DAB k CA0  
c=1;  
f=DAB*dCAdz;  
s=-k*CA;  
%  
% Initial condition  
%  
function CA_i=ex5_3_2ic(z)  
CA_i=0;  
%  
% Boundary conditions  
%  
function [pl, ql, pr, qr]=ex5_3_2bc(zl, CA_l, zr, CA_r, t)
```

MATLAB program design:

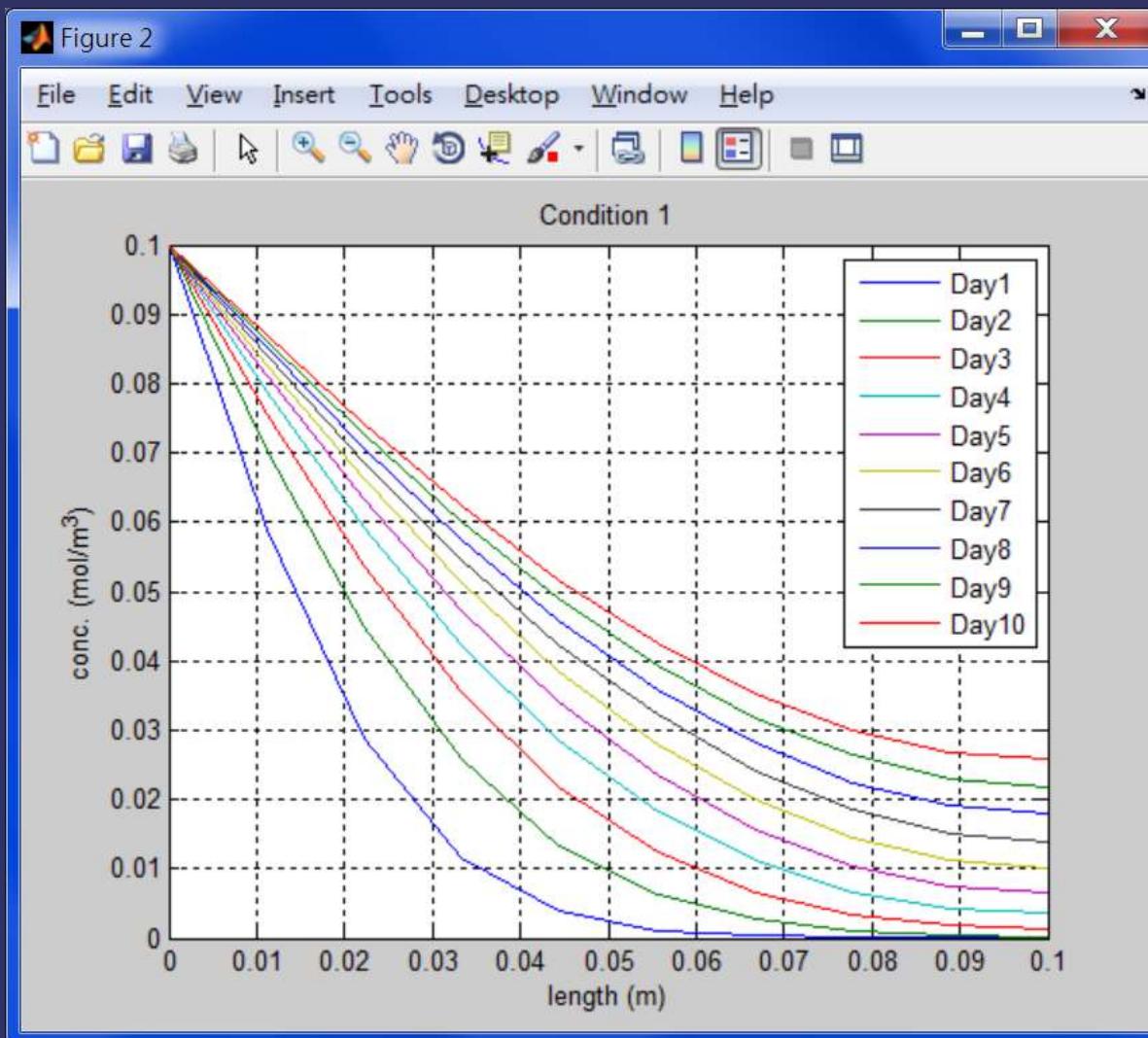
— ex5_3_2.m —

```
global DAB k CA0
pl=CAl-CA0;
ql=0;
pr=0;
qr=1/DAB;
```

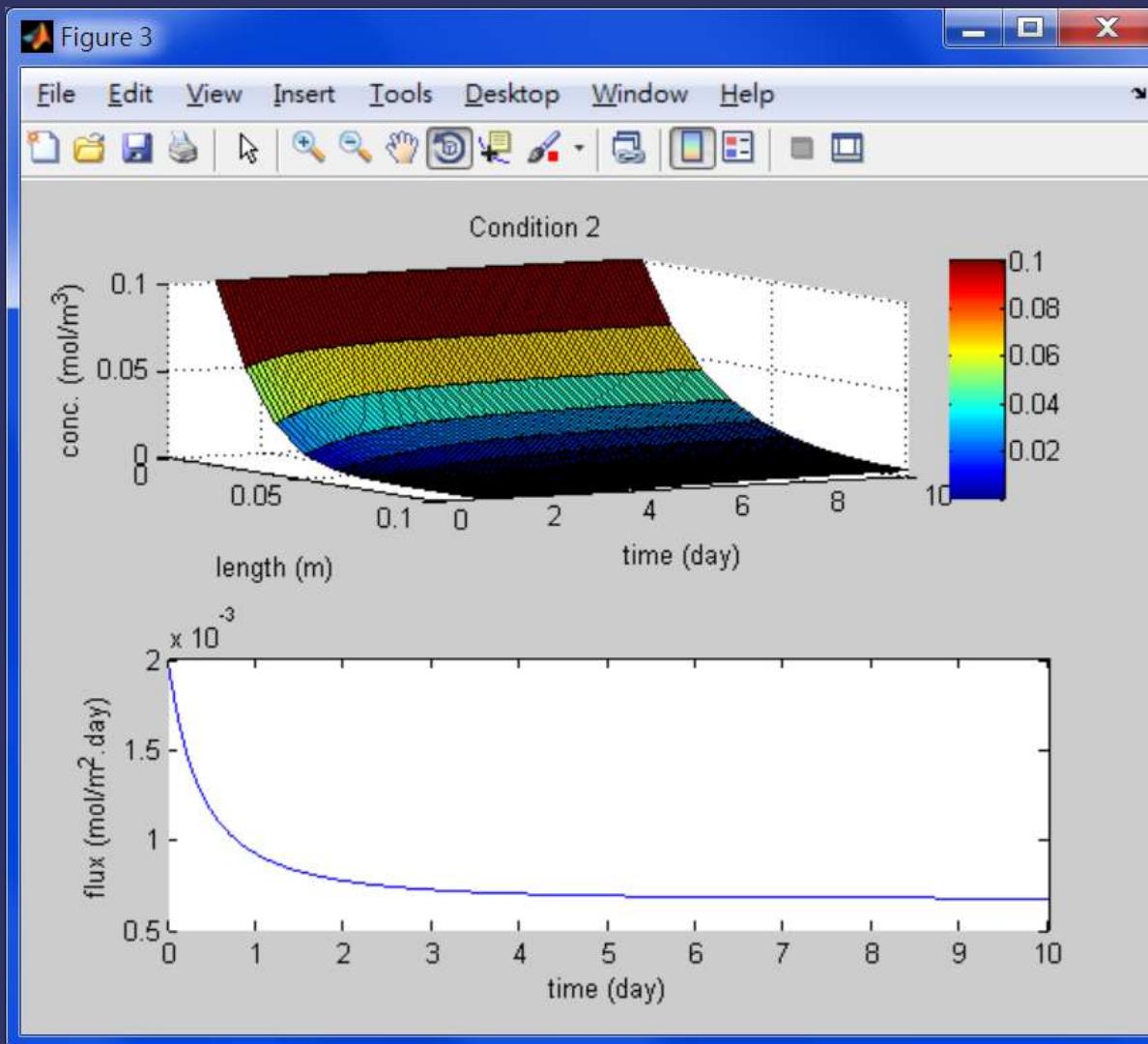
Execution results:



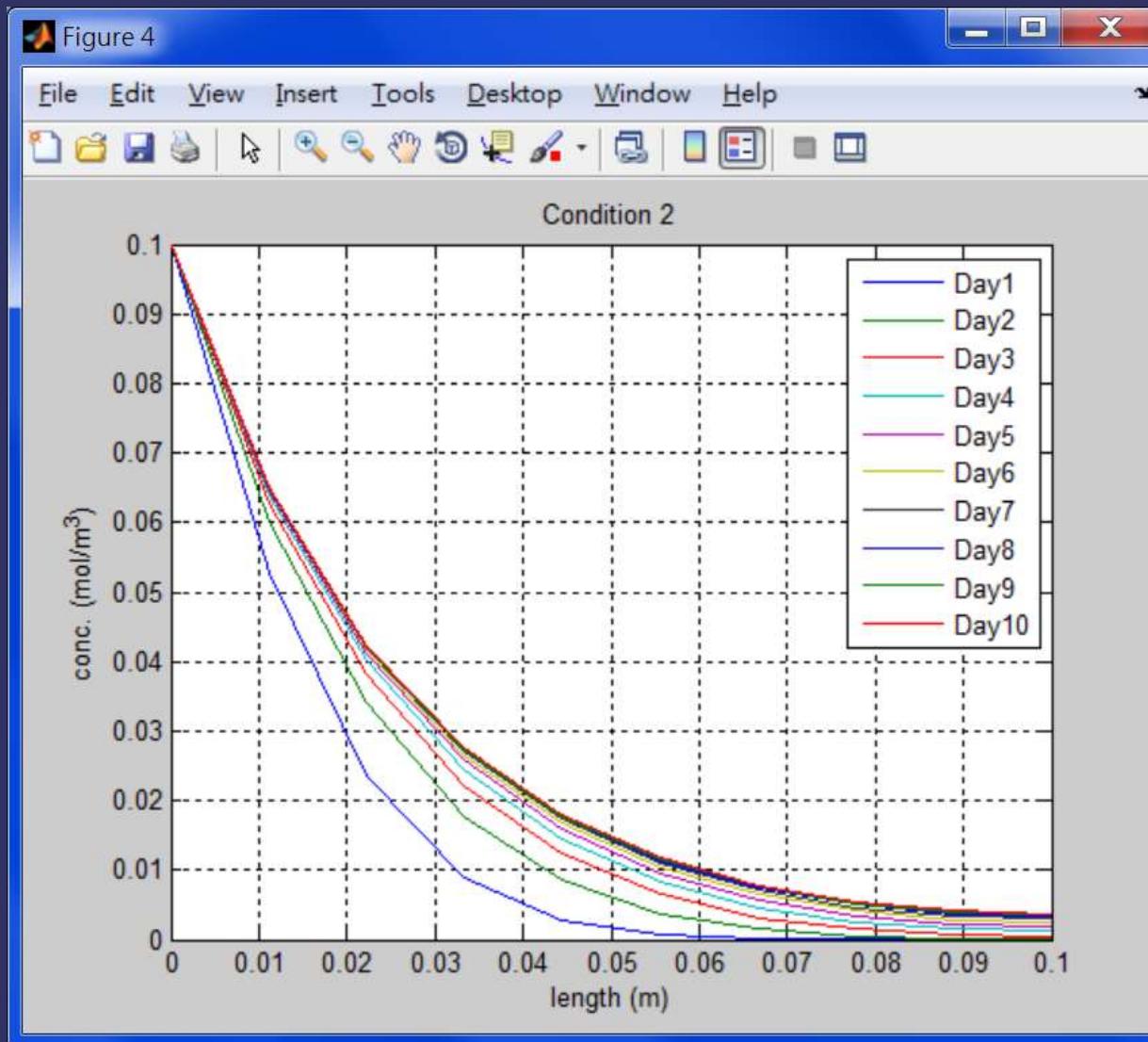
Execution results:



Execution results:



Execution results:



Example 5-3-3

Rapid cooling of a hot solid object

In the system, a 350°C hot spherical object whose diameter is $D = 0.15 \text{ m}$ is quenched into a large amount of water for rapid cooling. The temperature of the cold water is $T_w = 25^{\circ}\text{C}$, which is fixed and unchanged during the quenching process because of the large amount of water. The conductive heat transfer coefficient,

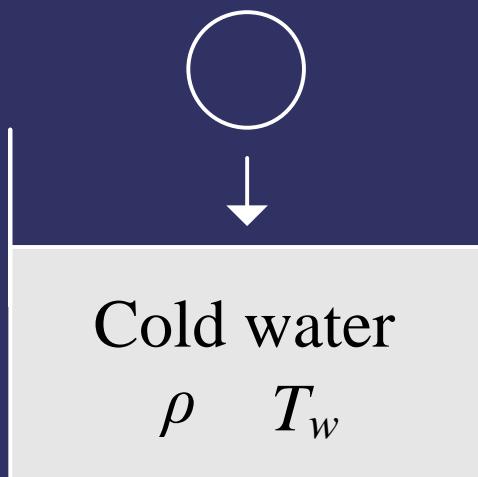


Figure 5.3
Quenching of a hot solid object in cold water.

Example 5-3-3

Rapid cooling of a hot solid object

the density, and the heat capacity of the object is $k = 10 \text{ W/m}\cdot\text{°C}$, $\rho = 8,200 \text{ kg/m}^3$, and $C_P = 0.410 \text{ kJ/kg}\cdot\text{°C}$, respectively. Besides, the average convective heat transfer coefficient between the spherical object surface and the water is $h = 200 \text{ W/m}^2\cdot\text{°C}$. Based on the above-mentioned operating conditions, determine the temperature distribution profile inside the hot spherical object in this quenching process.

Problem formulation and analysis:

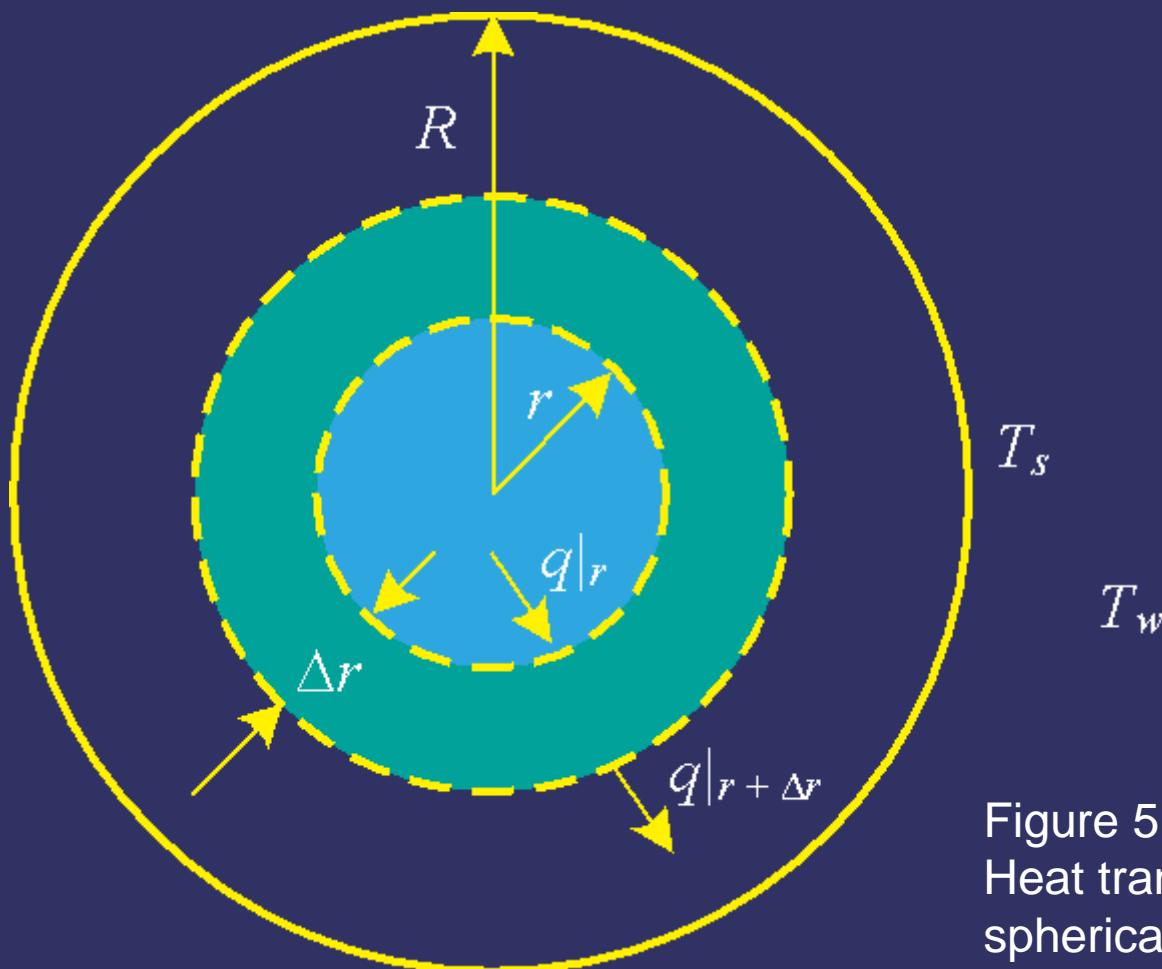


Figure 5.4
Heat transfer inside a solid spherical object.



Problem formulation and analysis:

$$q|_r \cdot \Delta t = q|_{r+\Delta r} \cdot \Delta t + 4\pi r^2 \Delta r \rho C_p (T|_{t+\Delta t} - T|_t)$$

$\Delta t \rightarrow 0$ and $\Delta r \rightarrow 0$,

$$\frac{\partial T}{\partial t} = - \left(\frac{1}{4\pi r^2 \rho C_p} \right) \frac{\partial q}{\partial r}$$

$$q = -4k\pi r^2 \frac{\partial T}{\partial r}$$

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C_p r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial T}{\partial r} \right)$$



Problem formulation and analysis:

$$T(r, 0) = 350 \text{ }^{\circ}\text{C}, \quad 0 \leq r \leq R$$

(i) The center of the spherical object

$$\frac{\partial T}{\partial r} \Big|_{r=0} = 0$$

(ii) The spherical surface of the object

$$-k \frac{\partial T}{\partial r} \Big|_{r=R} = h(T_s - T_w)$$

MATLAB program design:

By rewriting (5.3-25) into the standard form, one has $m = 2$ (sphere), $c = \rho C_p / k$, $f = \partial T / \partial r$, and $s = 0$. Besides, from the boundary conditions (5.3-27) and (5.3-28), the following parameters are obtained: $pl = 0$ and $ql = 1$, $pr = h(T_s - T_w)$, and $qr = k$.

MATLAB program design:

— ex5_3_3.m —

```
function ex5_3_3
%
% Example 5-3-3 Quenching of a spherical hot solid object
%
clear; close all;
clc
%
global k h Ti Tw rho Cp
%
% Given data
%
D=0.15; % Diameter of the object (m)
R=D/2; % Radius of the object (m)
Tw=25; % Cooling water temperature (°C)
Ti=350; % Initial temperature of the hot object (°C)
k=10; % Conductive heat transfer coefficient (W/m.°C)
rho=8200; % Density of the object (kg/m^3)
```

MATLAB program design:

— ex5_3_3.m —

```
Cp=0.41; % Heat capacity of the object (kJ/kg.°C)
h=200; % Average convective heat transfer coefficient (W/m^2.°C)
%
% Solve the PDE with pdepe
%
t=linspace(0, 2, 10);
r=linspace(0, R, 10);
m=2; % Sphere
sol=pdepe(m, @ex5_3_3pdefun, @ex5_3_3ic, @ex5_3_3bc, r, t);
T=sol(:, :, 1);
surf(r, t, T)
xlabel('Radial position')
ylabel('Time')
zlabel('Temperature')
%
% PDE equation
%
```

MATLAB program design:

— ex5_3_3.m —

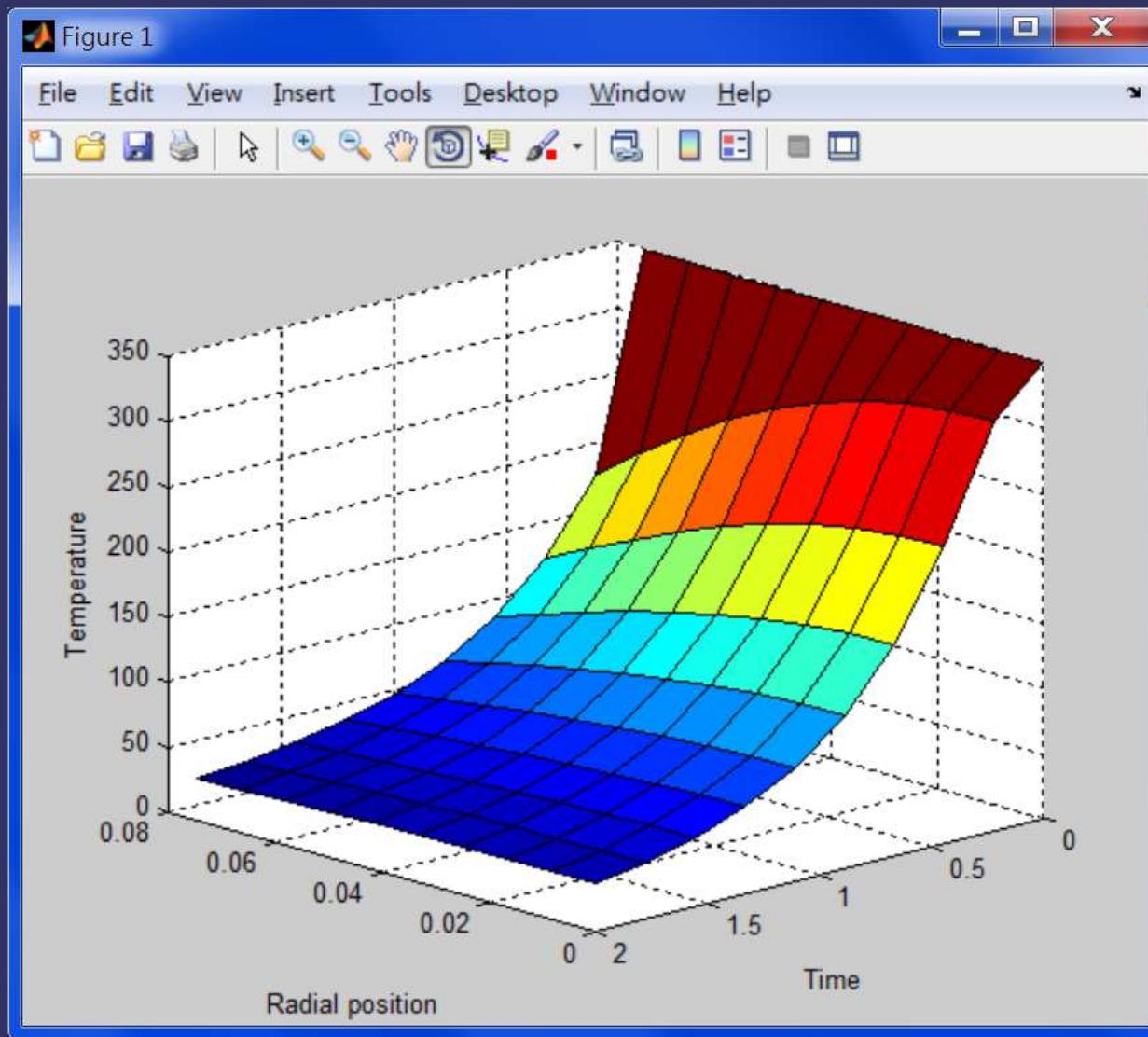
```
function [c, f, s]=ex5_3_3pdefun(r, t, T, dTdr)
global k h Ti Tw rho Cp
c=rho*Cp/k;
f=dTdr;
s=0;
%
% Initial condition
%
function T0=ex5_3_3ic(r)
global k h Ti Tw rho Cp
T0=Ti;
%
% Boundary conditions
%
function [pl, ql, pr, qr]=ex5_3_3bc(rl, Tl, rr, Tr, t)
global k h Ti Tw rho Cp
```

MATLAB program design:

— ex5_3_3.m —

```
pl=0;  
ql=1;  
pr=h*(Tr-Tw);  
qr=k;
```

Execution results:



Example 5-3-4

Two-dimensional heat transfer

Figure 5.5 illustrates a planar object where the edge temperature is maintained at either 100°C or 50 °C (Leu, 1985). The heat transfer coefficient, the density, and the heat capacity of the object are, respectively, given as $k = 10 \text{ W/m}\cdot\text{°C}$, $\rho = 10 \text{ kg/m}^3$, and $C_p = 0.50 \text{ kJ/kg}\cdot\text{°C}$. Because the thickness of the plate is considerably smaller than its width and length, the temperature distribution in the z -axis presumes to be uniform and thus can be ignored. Based on the above-mentioned operating conditions, answer the following questions:

- (a) What is the steady-state temperature distribution on the planar object?
- (b) What is the non-steady-state temperature distribution on the planar object?
- (c) If a circle whose radius is 2 is cut off from the center of the planar object (see the figure below) and the temperature around the circle is kept at 0°C , determine the temperature distribution of the new object.

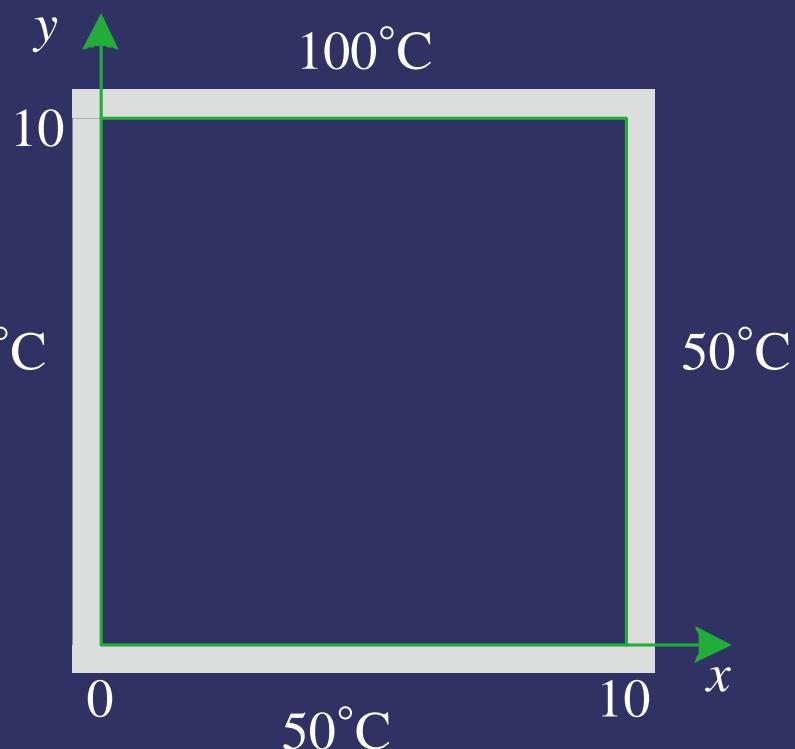
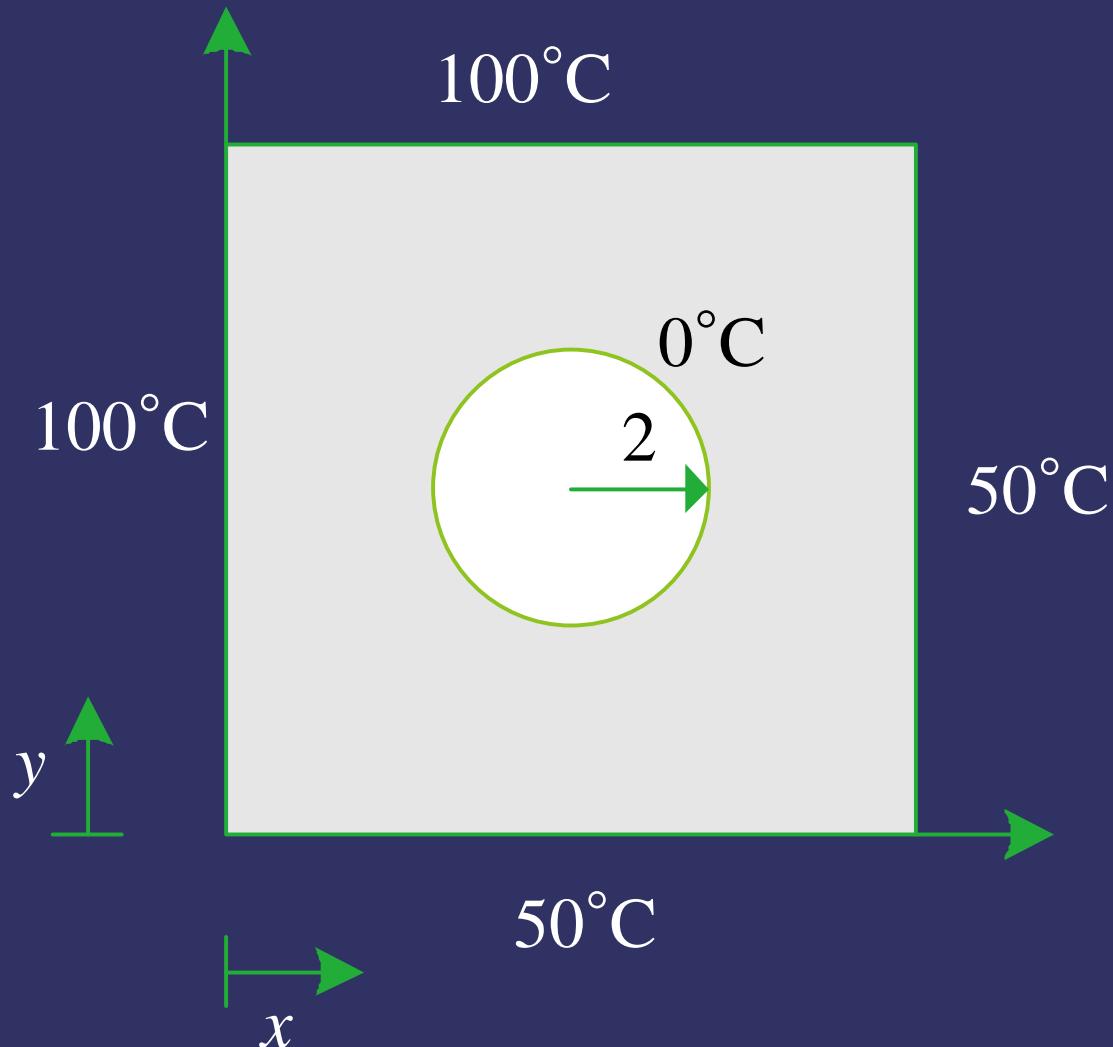


Figure 5.5 Boundary conditions on the planar object.





Analysis of the question:

$$\rho C_p \frac{\partial T}{\partial t} = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

case (a) and case (b)

$$\begin{cases} T(0, y) = 100^\circ\text{C} \\ T(10, y) = 50^\circ\text{C} \end{cases}, \quad 0 \leq y \leq 10$$

$$\begin{cases} T(x, 0) = 50^\circ\text{C} \\ T(x, 10) = 100^\circ\text{C} \end{cases}, \quad 0 \leq x \leq 10$$

Analysis of the question:

the case (c),

$$T(x, y) = 0^\circ\text{C}, \quad x, y \in \Omega_C$$

where Ω_C is defined by

$$\Omega_C = \left\{ (x, y) \mid (x - 5)^2 + (y - 5)^2 = 4 \right\}$$

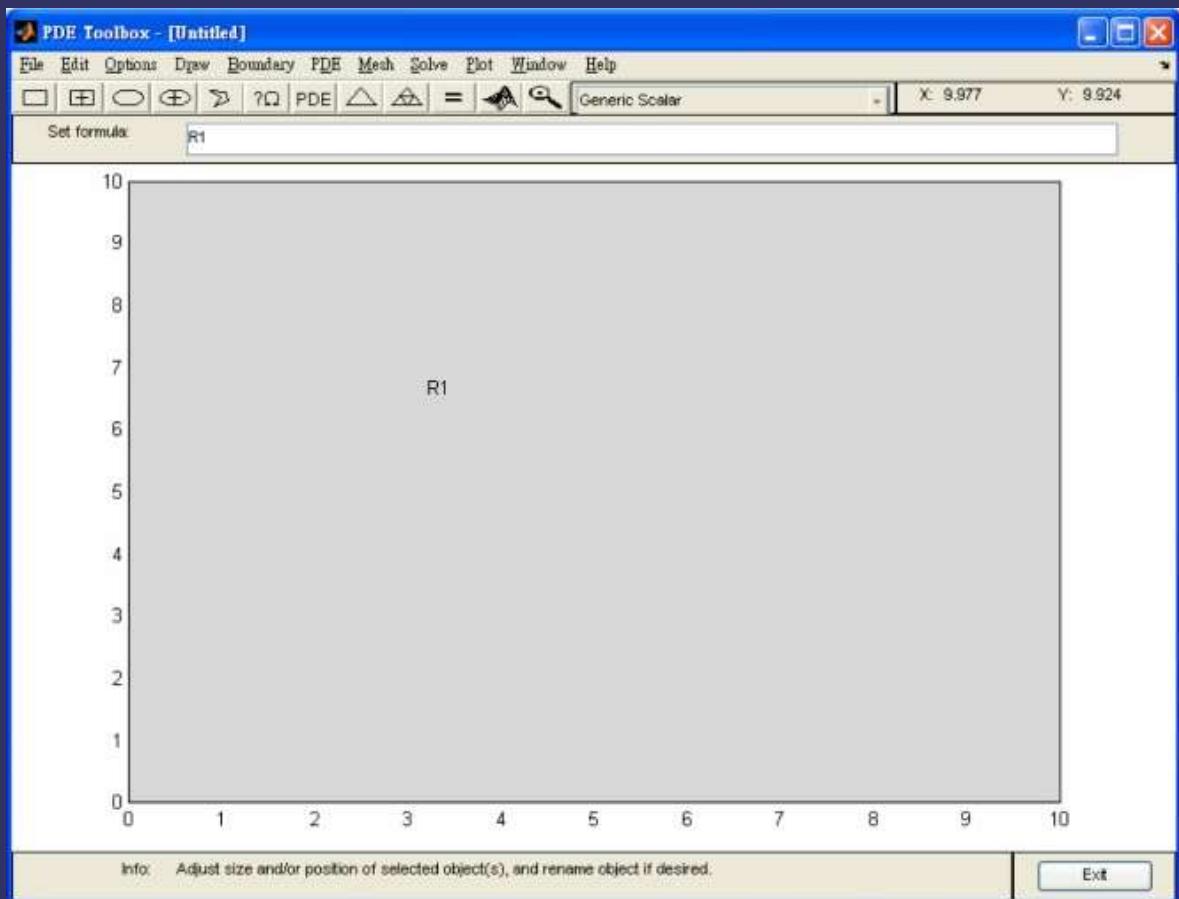
Solution by MATLAB pdetool:

(a) Solution by MATLAB pdetool

Step 1:

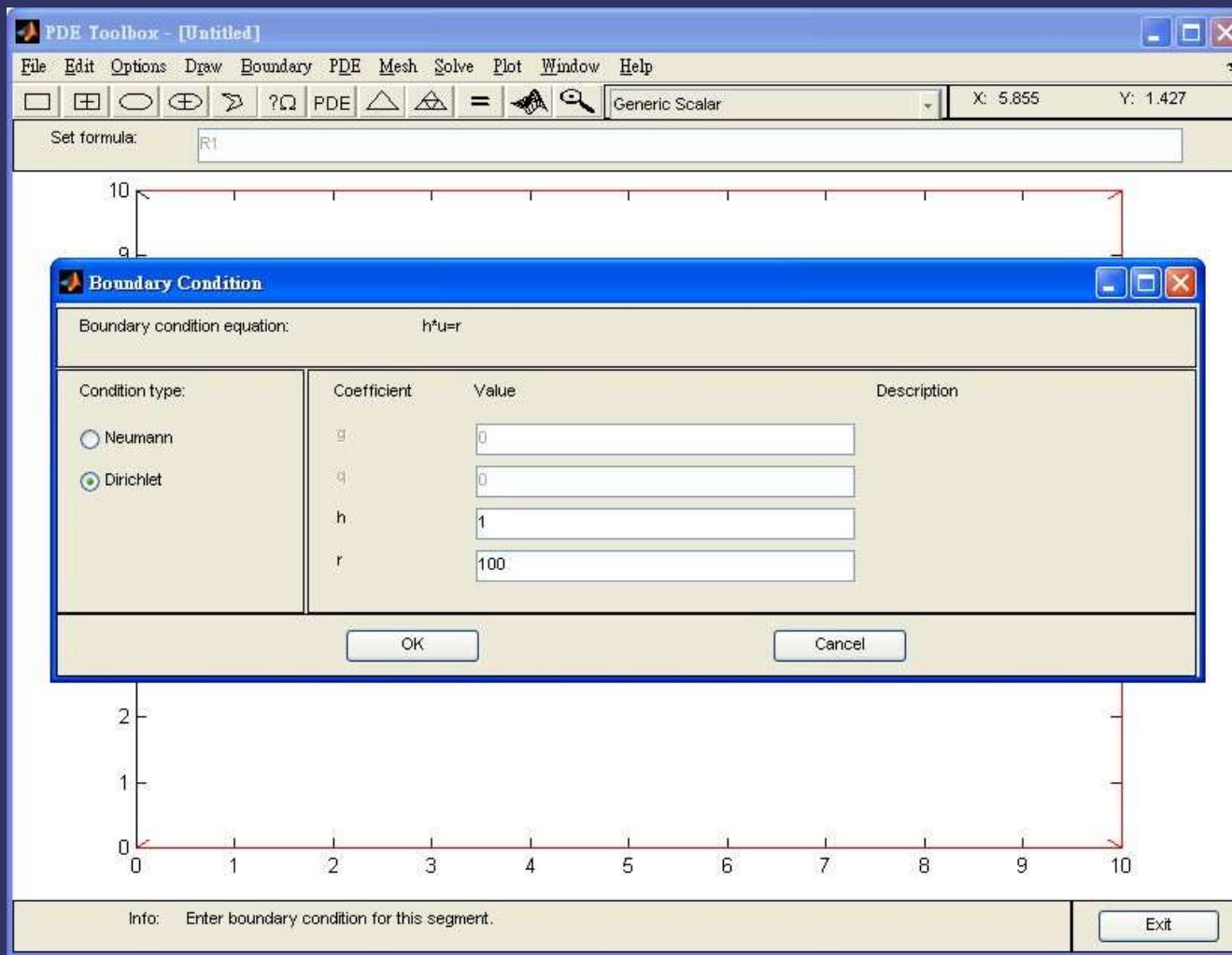
```
>> pdetool
```

Step 2:



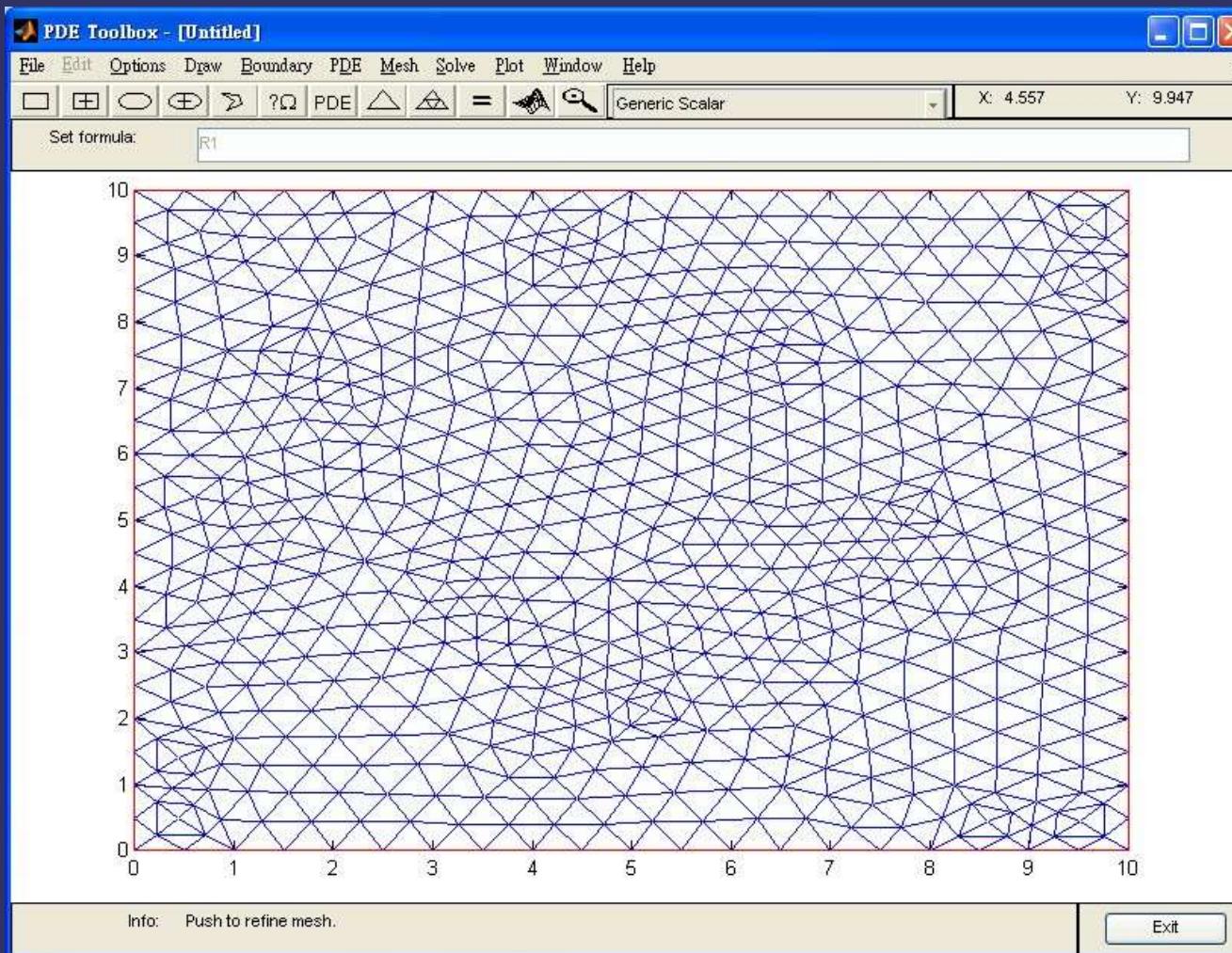
Solution by MATLAB pdetool:

Step 3: Use the icon  to set up the boundary conditions.



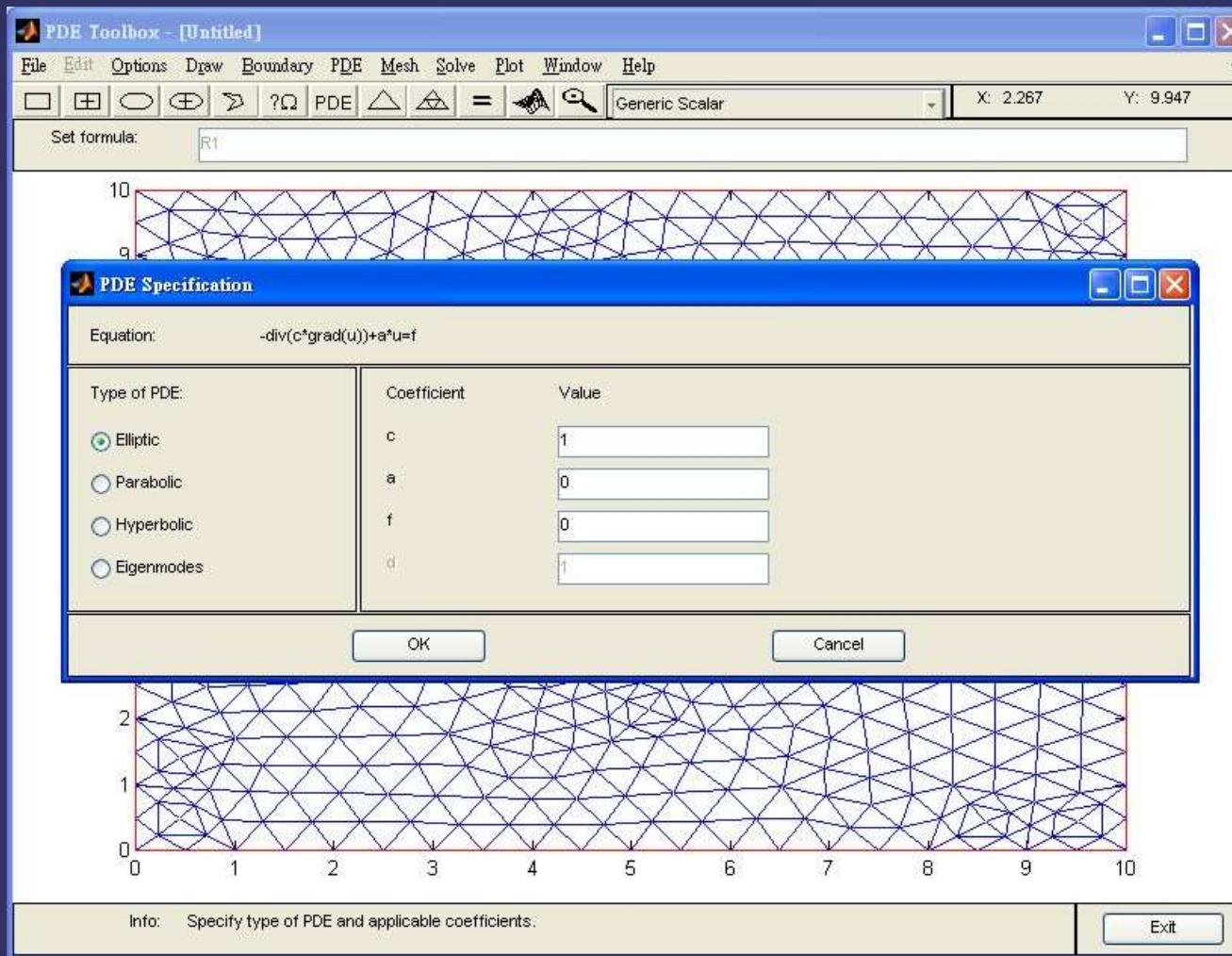
Solution by MATLAB pdetool:

Step 4: Click  to take the meshes



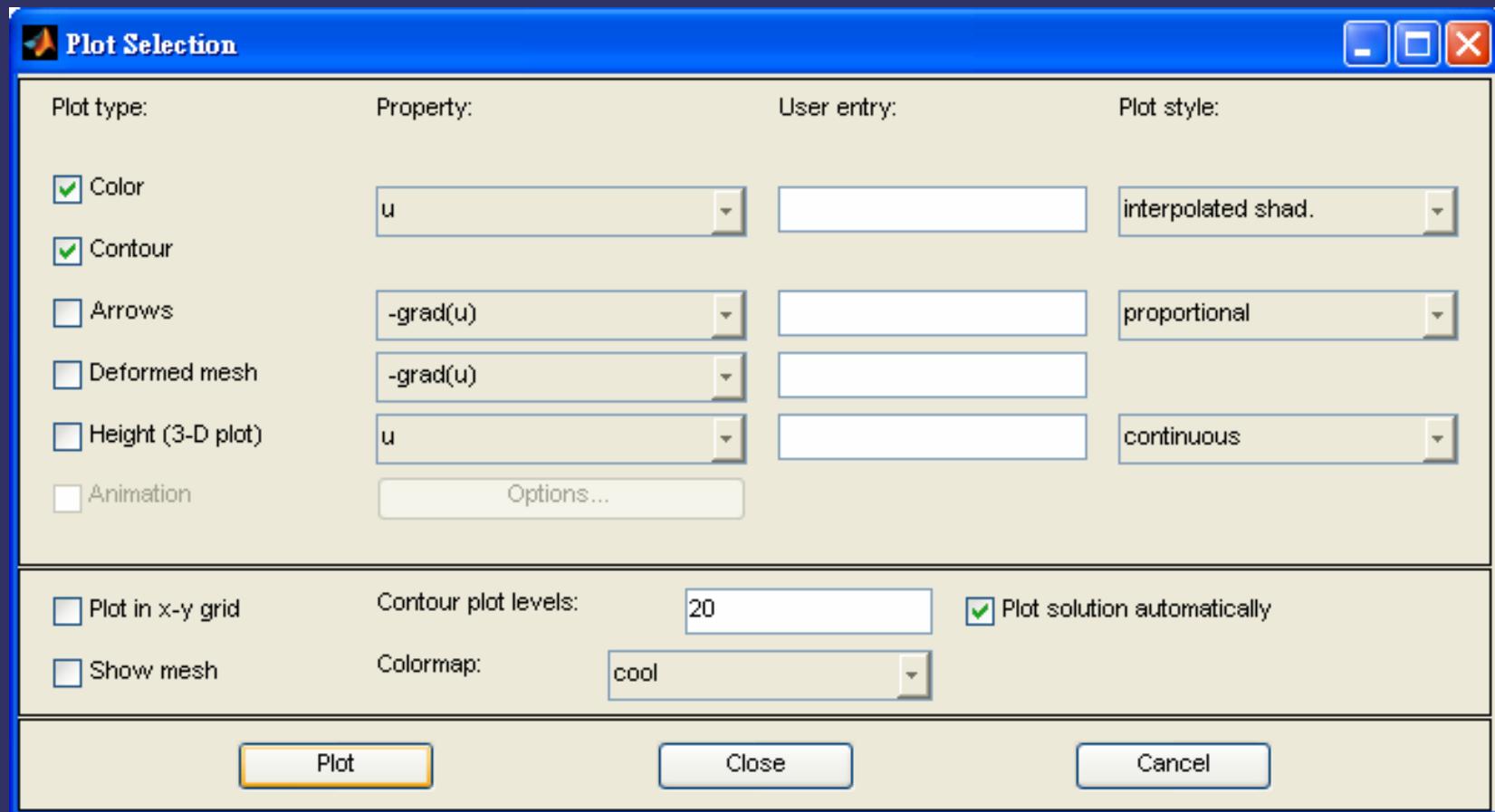
Solution by MATLAB pdetool:

Step 5: Click the icon  to specify the PDE equation



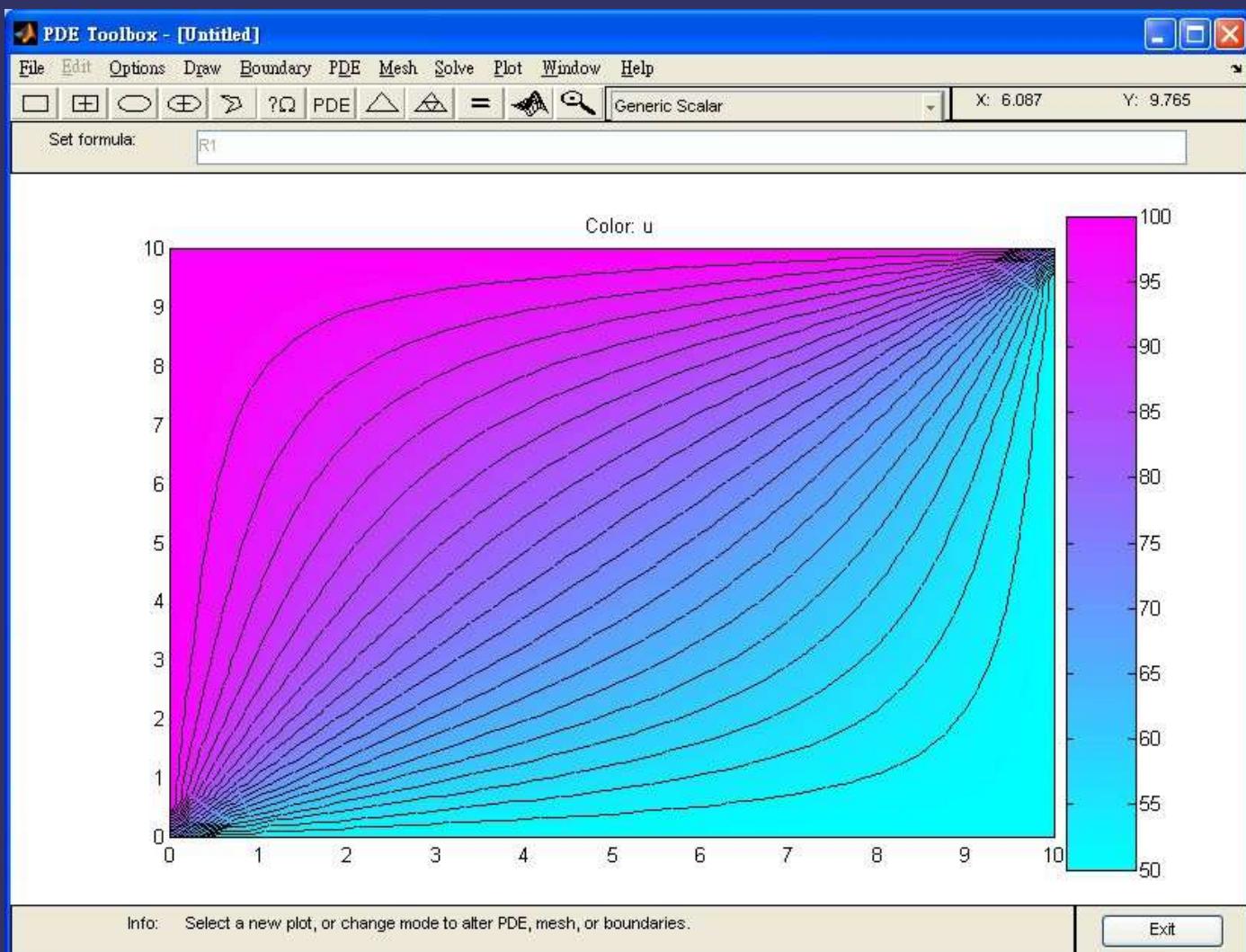
Solution by MATLAB pdetool:

Step 6: Click the icon  to solve the PDE



Solution by MATLAB pdetool:

Step 6:



Solution by MATLAB pdetool:

Step 6:

the results are output as the variable names p, t, and u,

```
>> uxy=tri2grid(p,t,u,5,5)
```

```
uxy =
```

```
75.0031
```

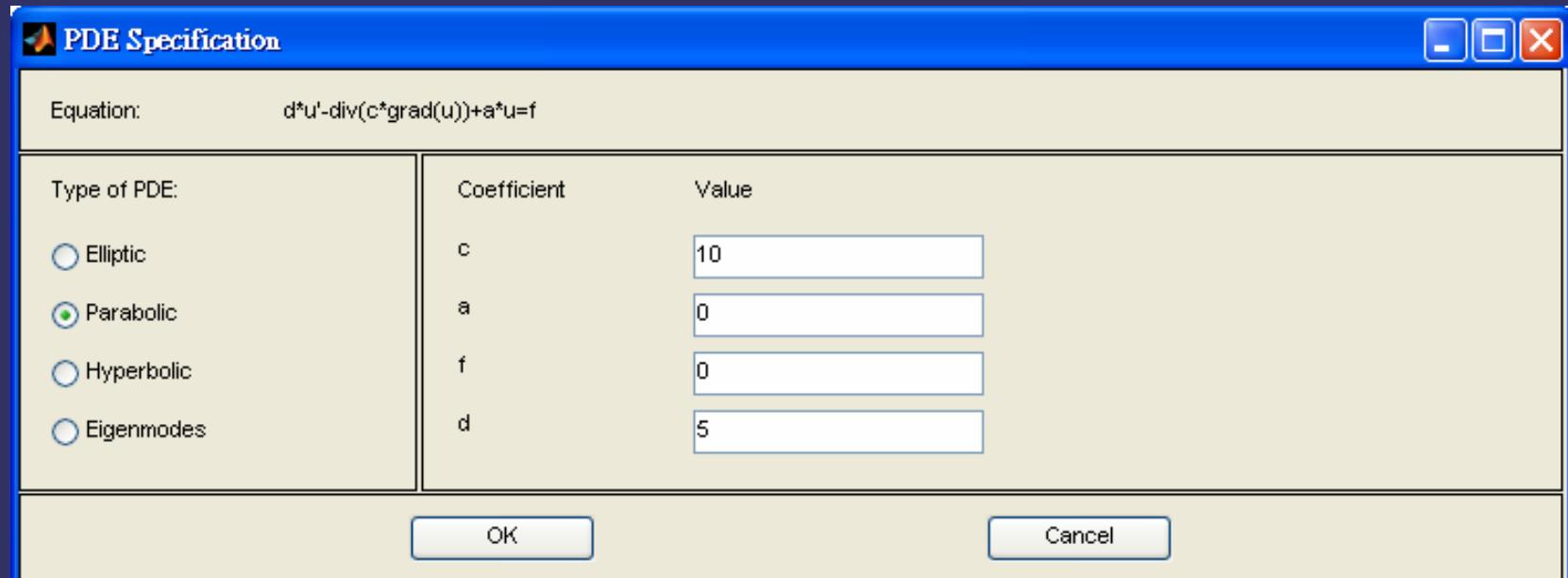
That is, the temperature at the point (5, 5) is 75.0031°C.

Solution by MATLAB pdetool:

(b) Non-steady-state heat transfer

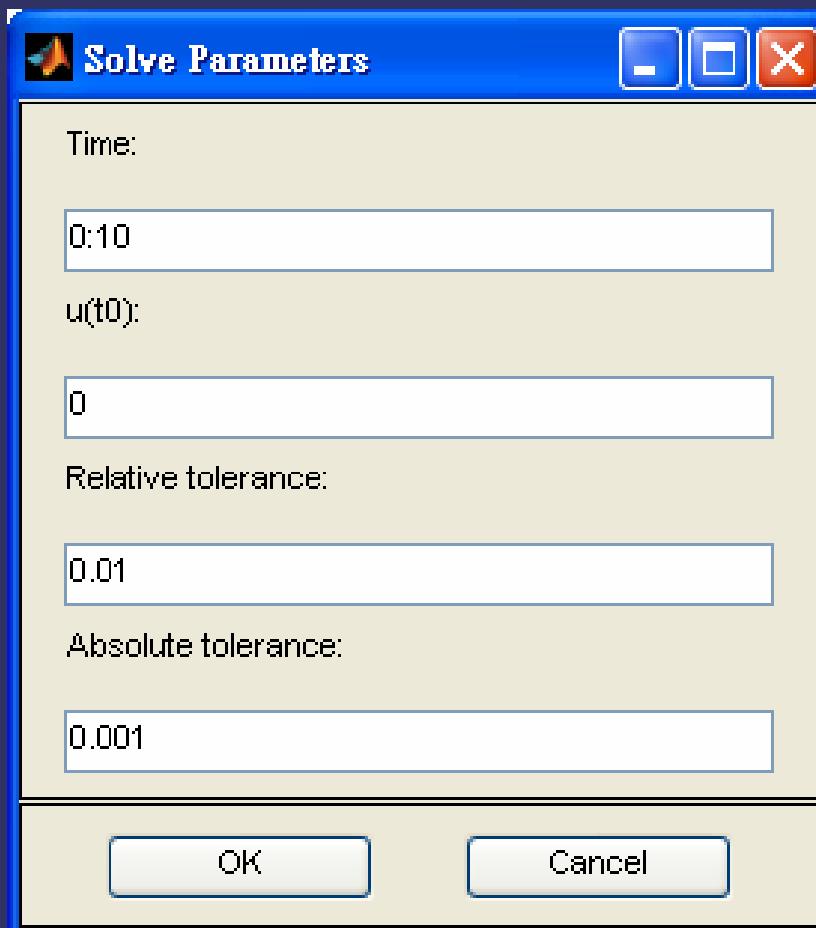
the solution steps 1-4 are identical

Step 5: Click the icon  to specify the PDE equation.



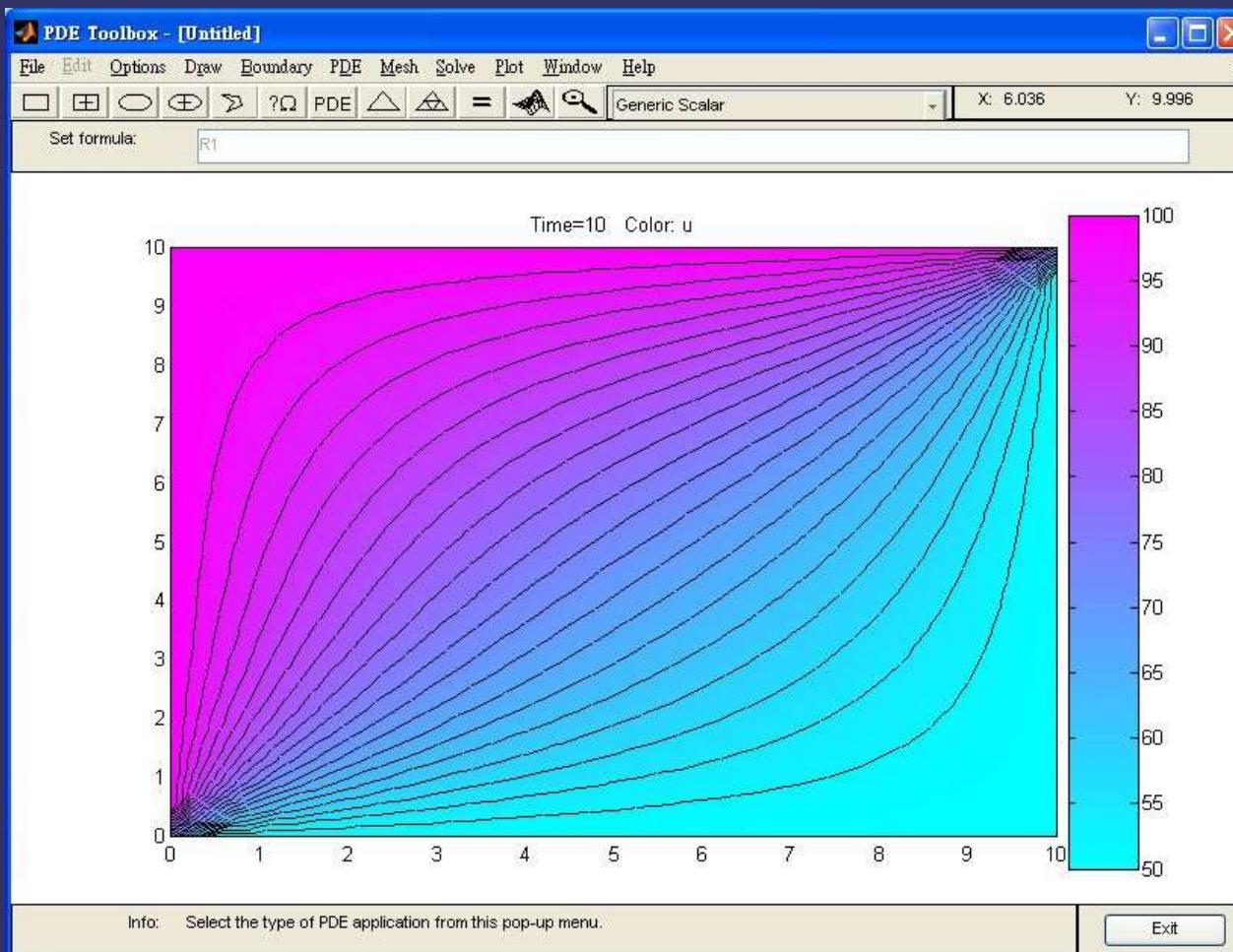
Solution by MATLAB pdetool:

Step 6:



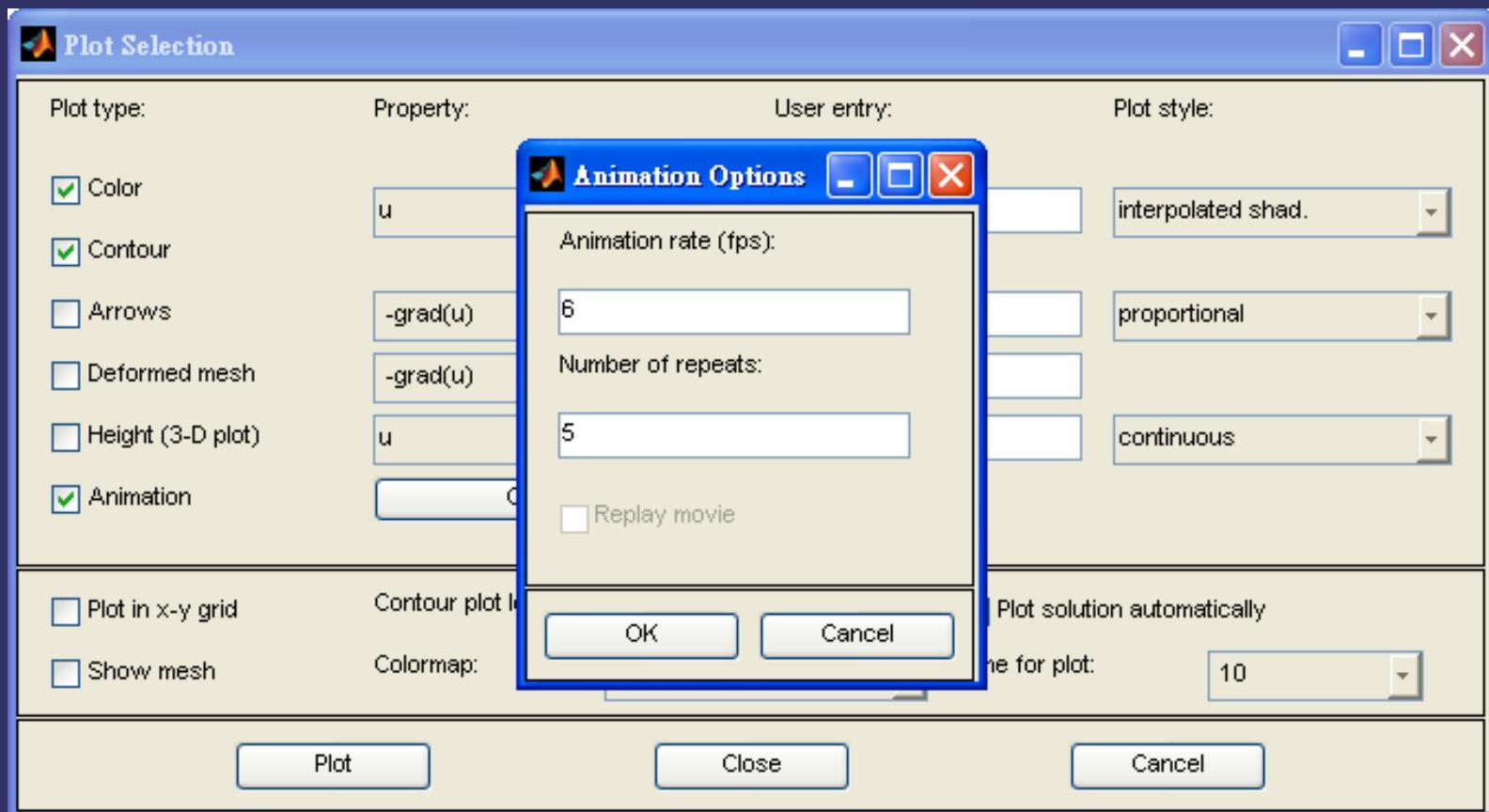
Solution by MATLAB pdetool:

Step 7: Press the icon  to solve the PDE.



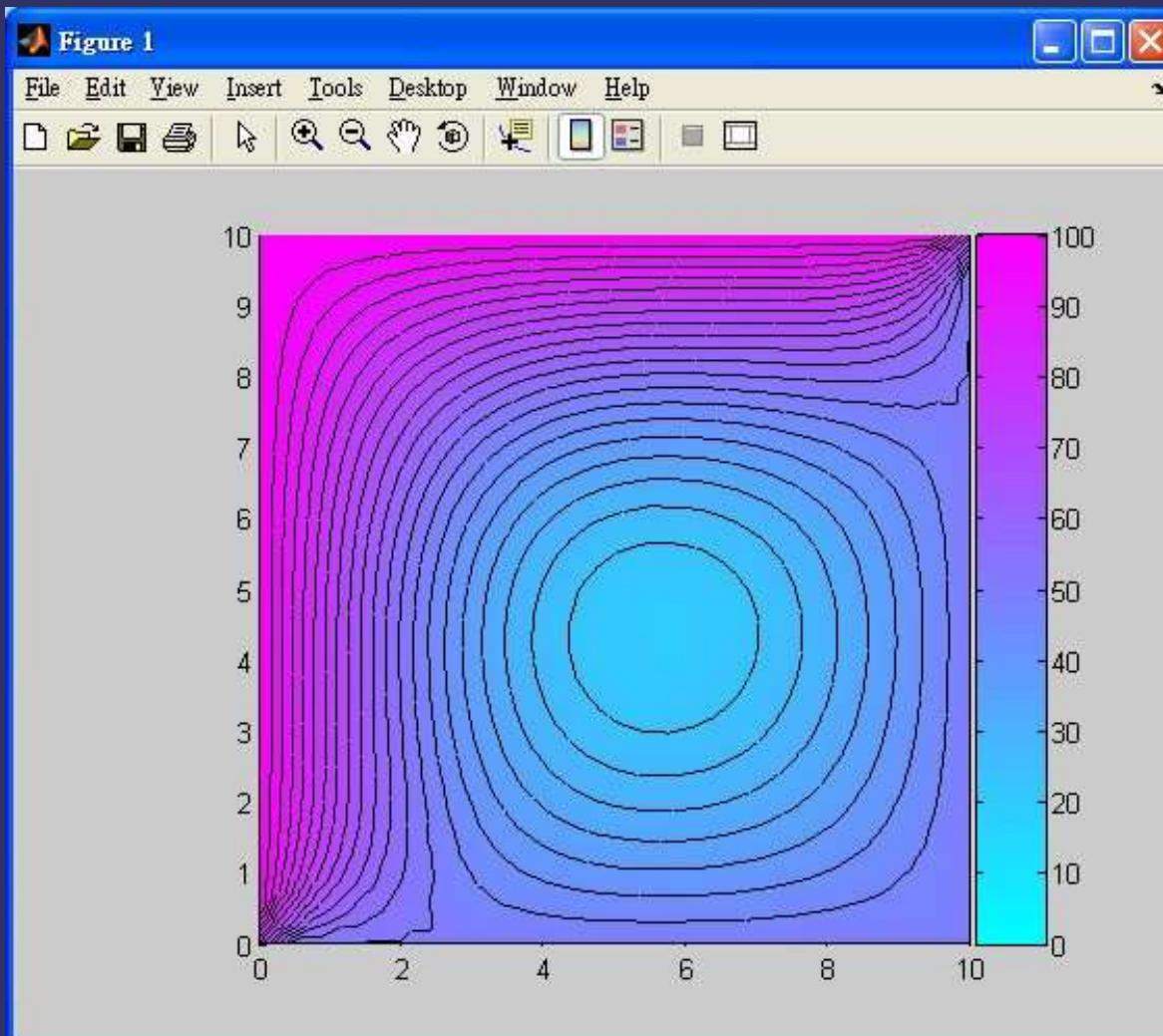
Solution by MATLAB pdetool:

Step 8:



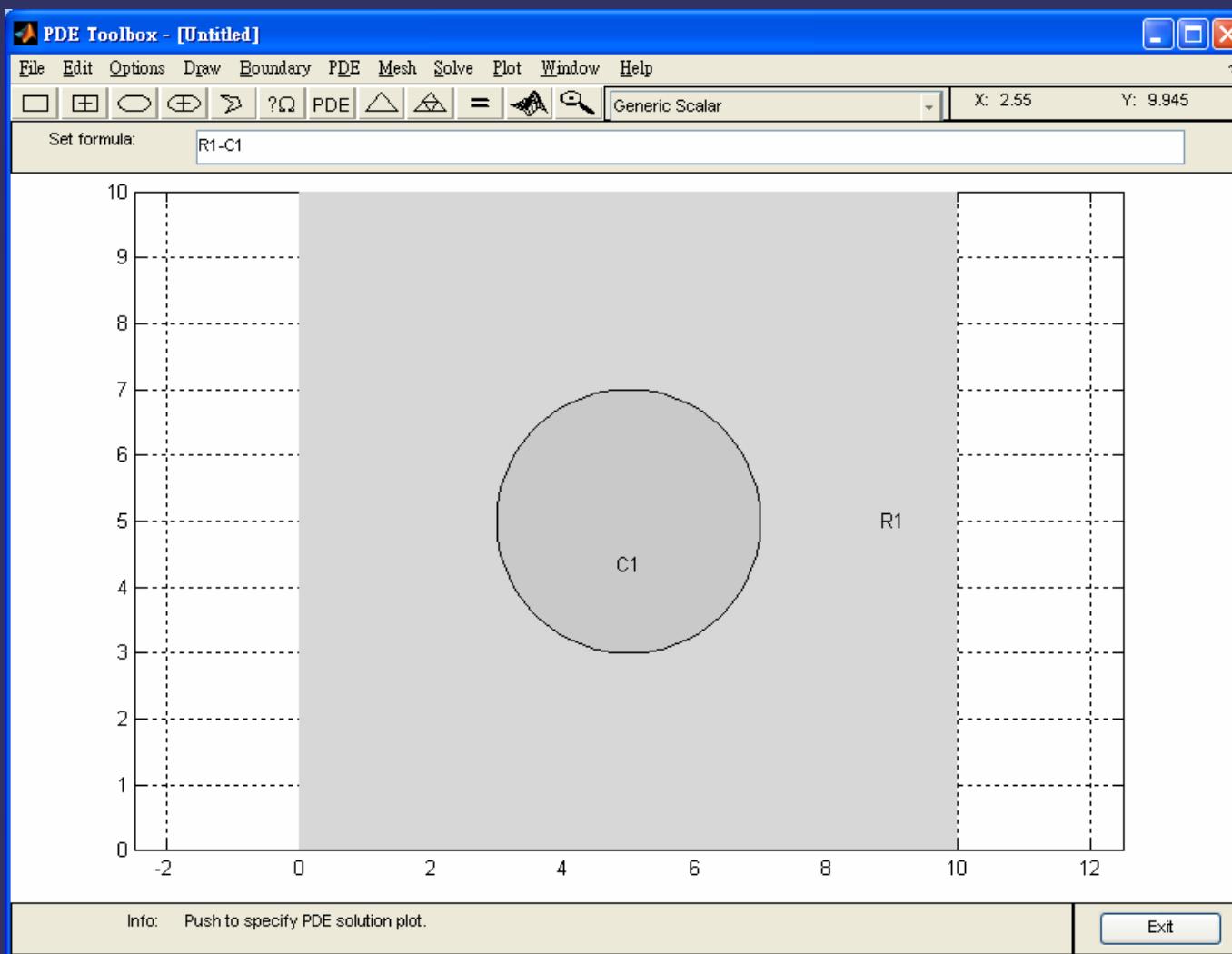
Solution by MATLAB pdetool:

Step 8:



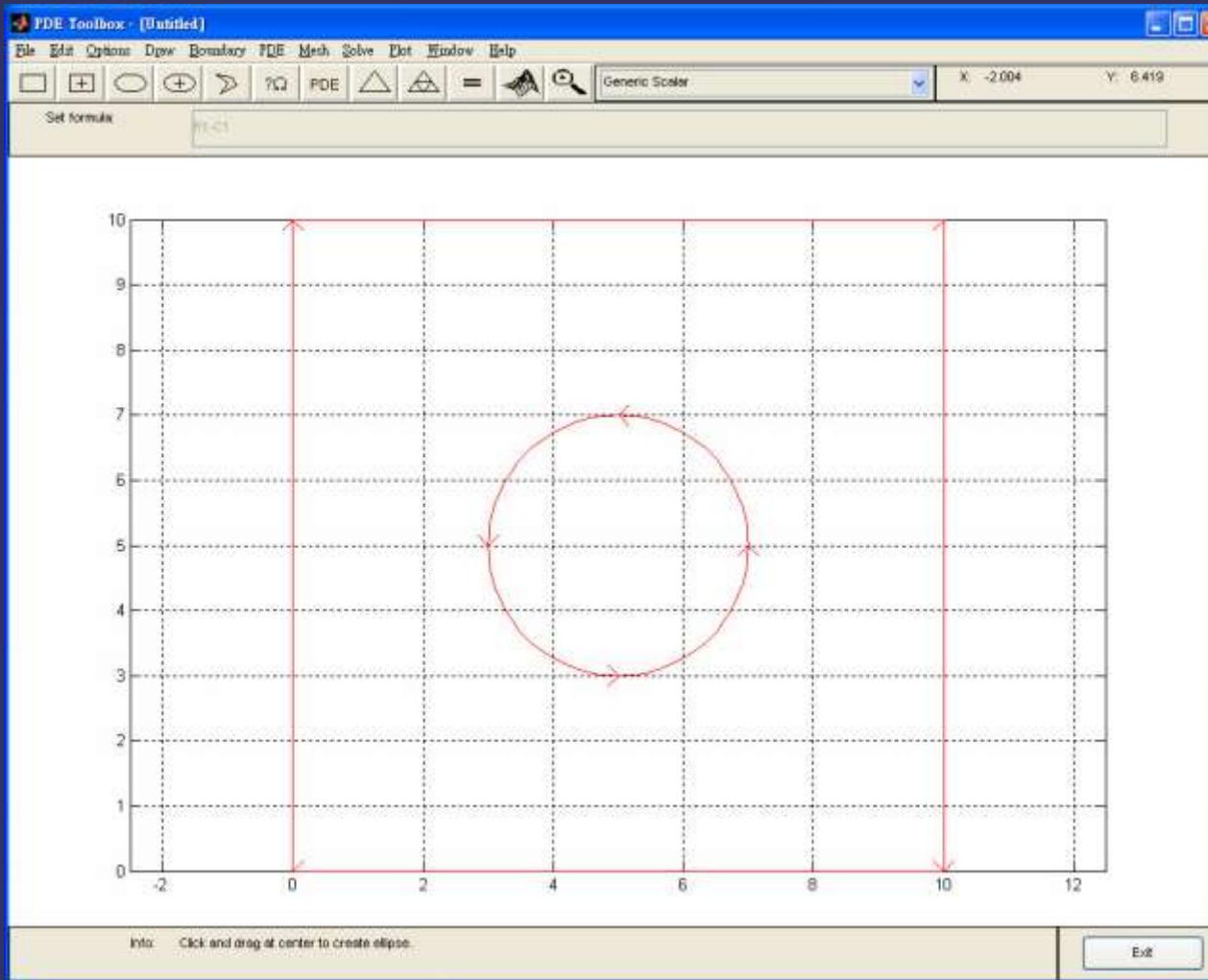
Solution by MATLAB pdetool:

(c)



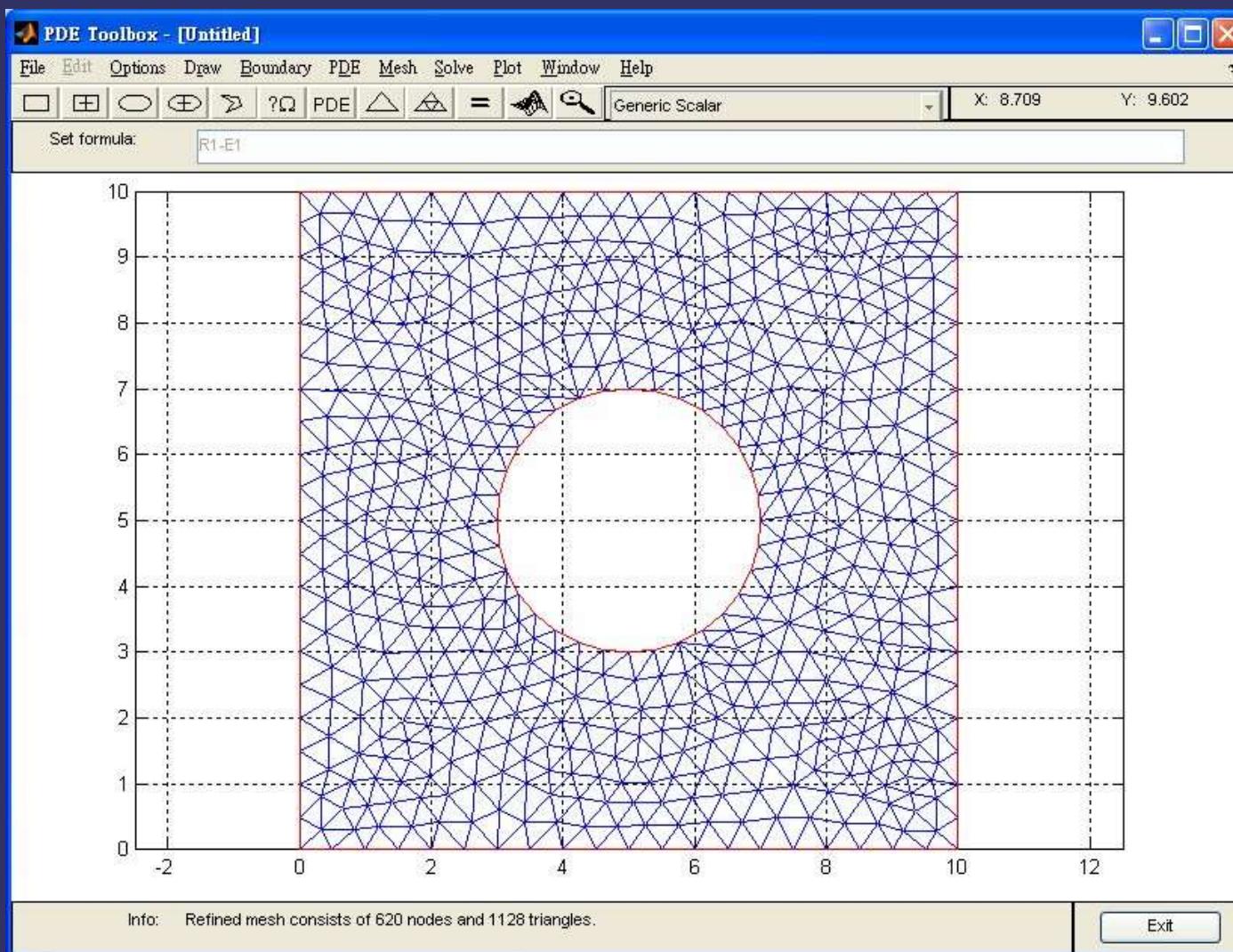
Solution by MATLAB pdetool:

(c) Set formula by typing R1-C1



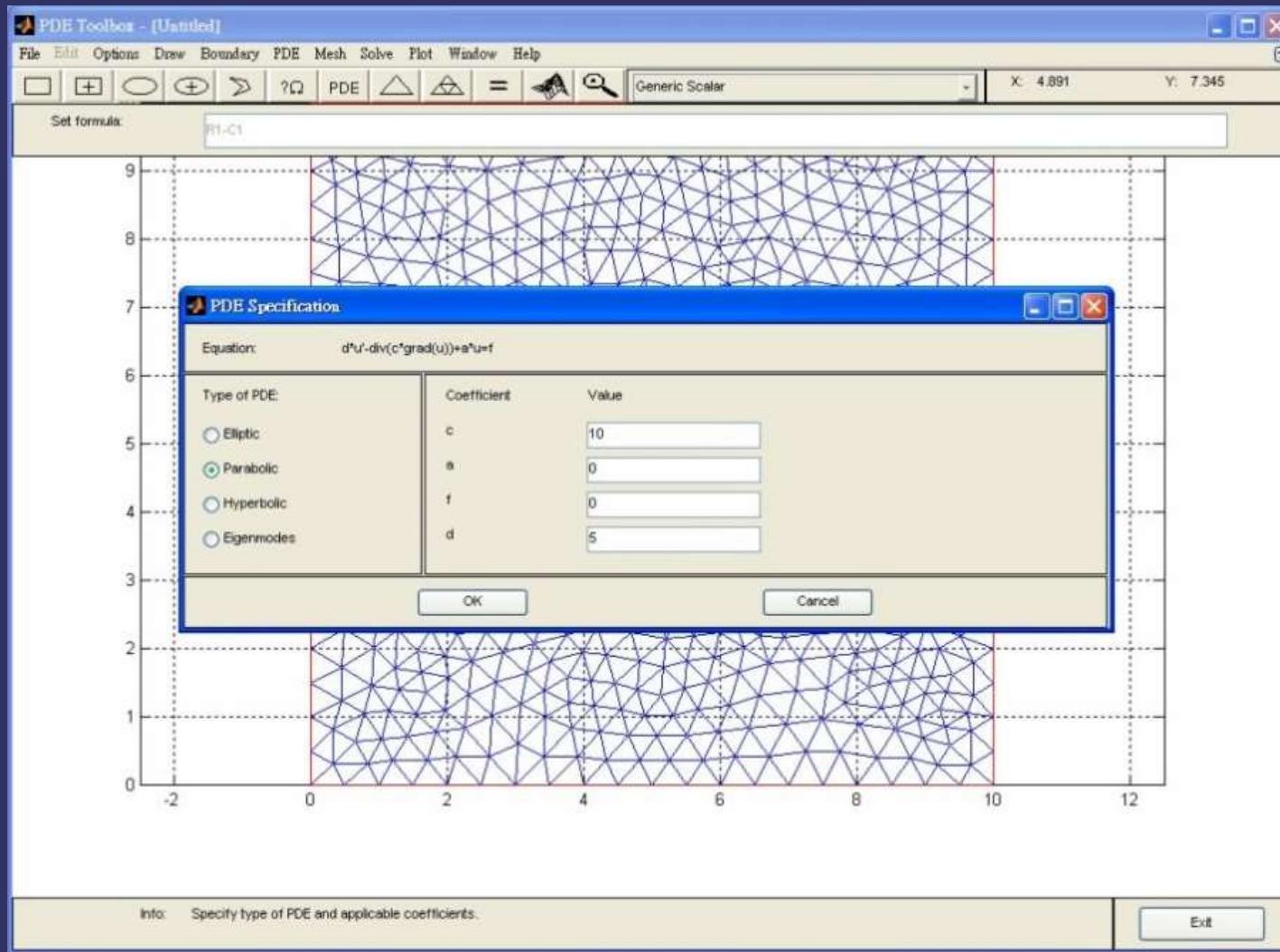
Solution by MATLAB pdetool:

(c)



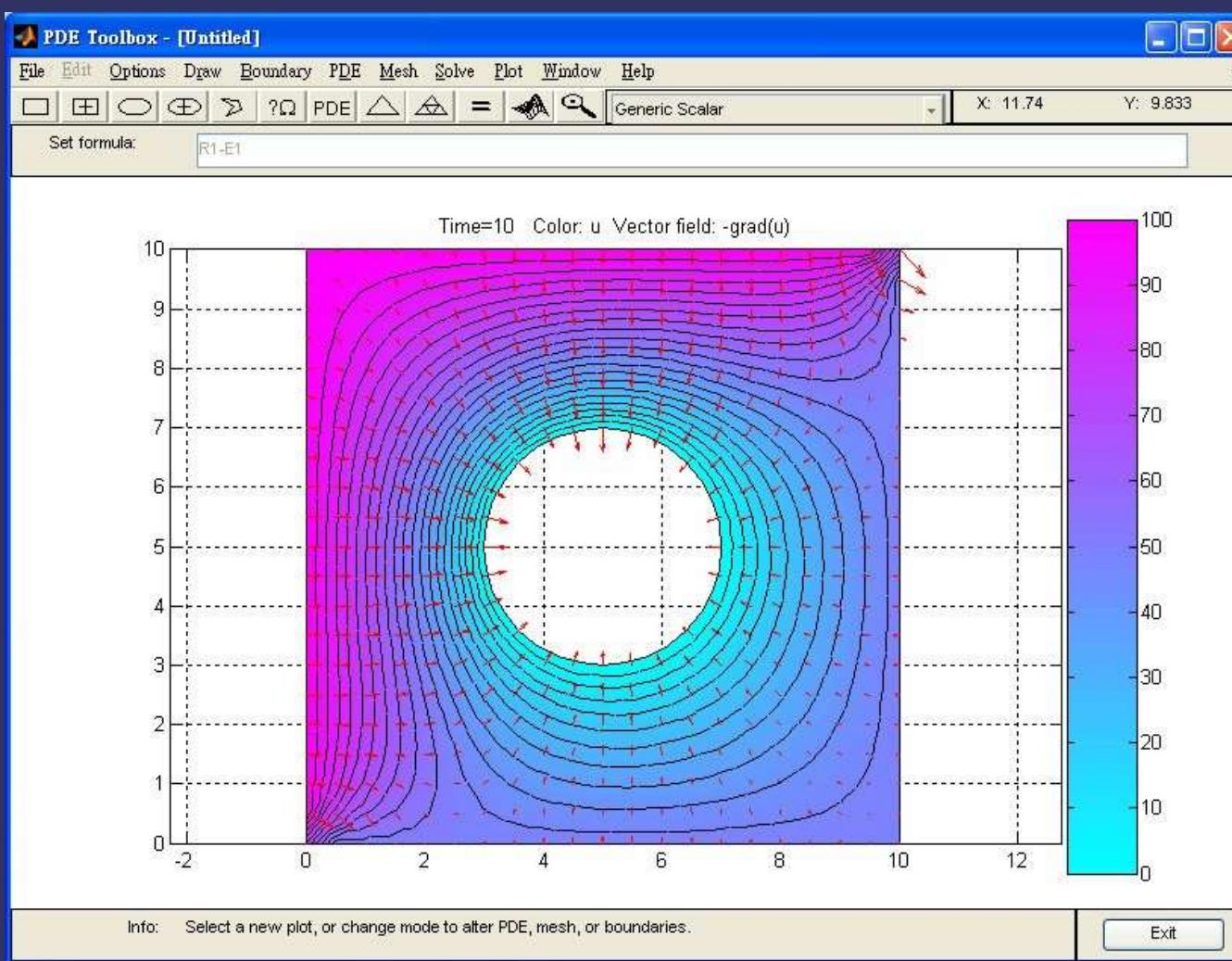
Solution by MATLAB pdetool:

(c)



Solution by MATLAB pdetool:

(c)





Example 5-3-5

The permeation of gaseous solute into a liquid film

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} \right), \quad 0 < x < L$$

$$c \Big|_{t=0} = 0, \quad 0 < x < L$$

$$\begin{cases} c \Big|_{x=0} = c^* \\ c \Big|_{x=L} = 0 \end{cases}$$



Based on the above, answer the following questions:

- 1) What is the solute concentration distribution of the film?
- 2) Determine the variations of the dimensionless mass transfer coefficient, the so-called Sherwood number, with respect to time.



Problem formulation and analysis:

$$\theta = \frac{Dt}{L^2}, \quad X = \frac{x}{L}, \quad C = \frac{c}{c^*},$$

$$\frac{\partial C}{\partial \theta} = \frac{\partial^2 C}{\partial X^2}, \quad 0 < X < 1$$

$$\begin{cases} C|_{\theta=0} = c^*, \quad 0 < X < 1 \\ C|_{X=0} = 1, \quad T > 0 \\ C|_{X=1} = 0, \quad T \geq 0 \end{cases}$$

Problem formulation and analysis:

$$Sh \equiv \frac{k_A L}{D} = \frac{LN_A}{Dc^*} = \frac{L}{Dc^*} \left(-D \frac{\partial c}{\partial x} \right)_{x=0} = - \left(\frac{\partial C}{\partial X} \right)_{x=0}$$

$$f(h) = f(0) + hf'(0) + \left(\frac{h^2}{2} \right) f''(0) + \left(\frac{h^3}{3!} \right) f'''(0) + \dots$$

and

$$f(2h) = f(0) + 2hf'(0) + \left(\frac{4h^2}{2} \right) f''(0) + \left(\frac{8h^3}{3!} \right) f'''(0) + \dots$$

$$f'(0) = - \left(\frac{1}{2h} \right) [3f(0) - 4f(h) + f(2h)]$$

$$Sh = \left(\frac{1}{2h} \right) [3C(0) - 4C(h) + C(2h)]$$

Solution by MATLAB:

The PDE solution steps

Step 1: After launching the PDE toolbox, first change the coordinate of x axis to $[0 \ 1]$, and draw a rectangle with the length of 1. Note that any width is acceptable since only the x direction is considered for the problem under study and there is no concern with the y direction.

Step 2: Enter Boundary mode, change the left-hand side boundary to Dirichlet and set $h = 1$ and $r = 1$. The boundary condition on the right-hand side is also specified as Dirichlet with $h = 1$ and $r = 0$. In addition, the top and bottom boundaries that have no effects on the results should be set as Neumann with $g = 0$ and $q = 0$.

Solution by MATLAB:

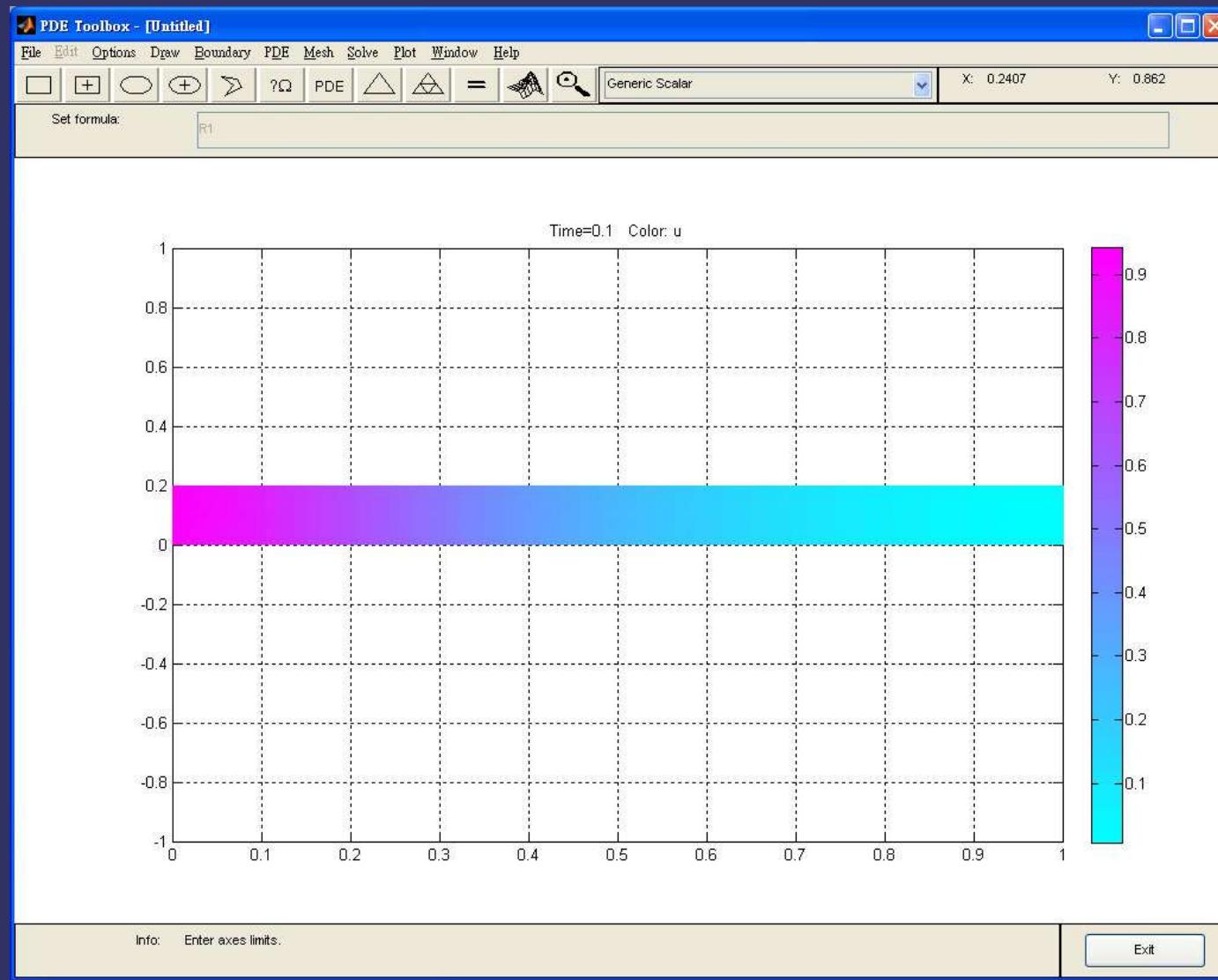
Step 3: Specify the PDE type as parabolic and set the PDE coefficients to be $c = d = 1$ and $a = f = 0$.

Step 4: Set the time in Solve Parameters to $0:0.01:0.1$, i.e., a total of 11 points from 0 to 0.1 with an interval of 0.01, and then solve the PDE.

Step 5: Export the Solution and the Mesh to the MATLAB Workspace for subsequent data manipulation. Note that the solution (mesh) export can be done under the menu of Solve (Mesh), where there is a function named “Export Solution” (“Export Mesh”).

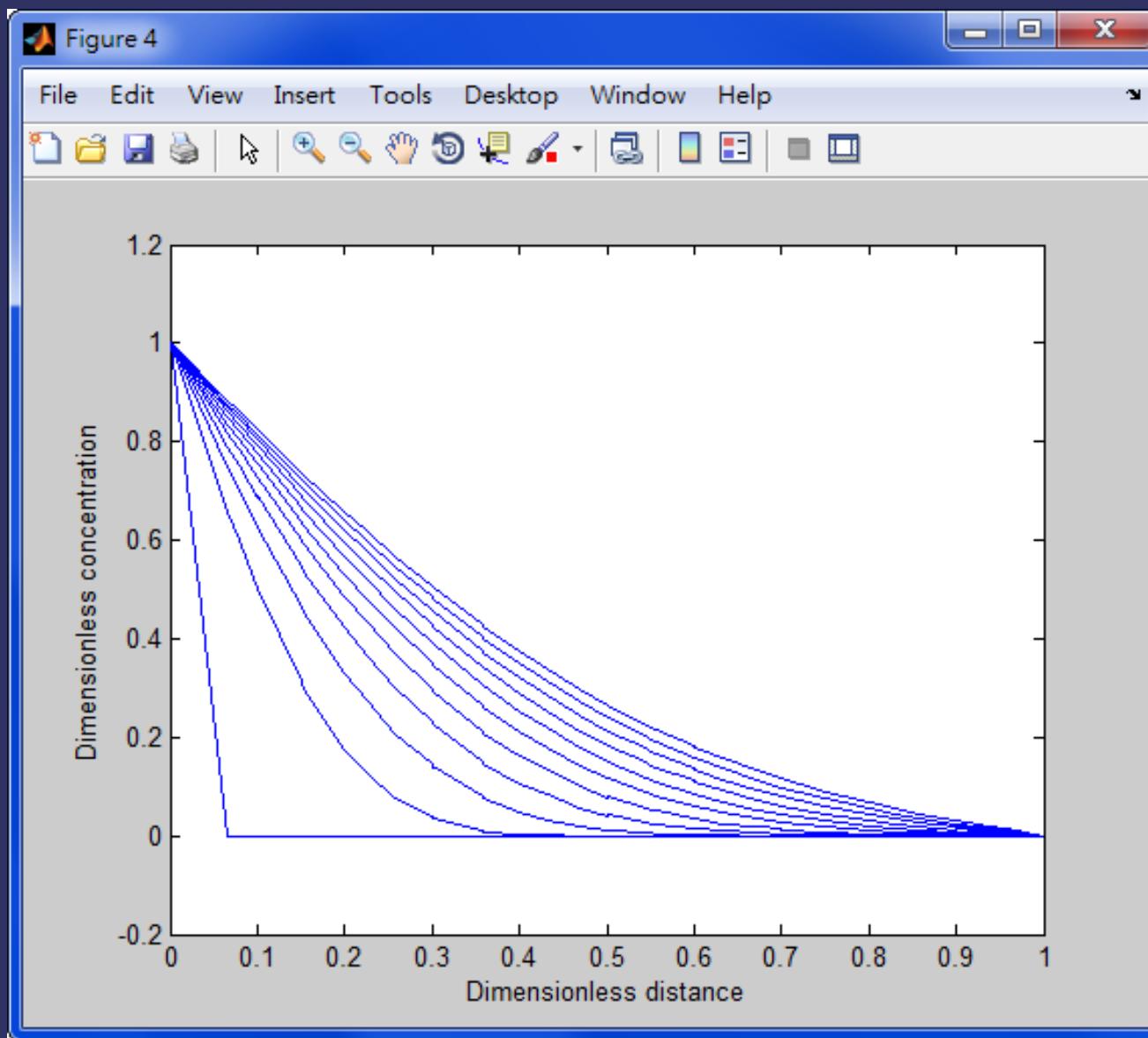
Numerical Solution of Partial Differential Equations

05



ex5_3_5b.m

```
[xs, I]=sort(xx);
for i=1:11 % Plot a graph at each time point
    ui=u(:, i);
    usi=ui(I); % Concentration value at each position at the time of t(i)
    plot(xs, usi)
    hold on
end
xlabel('Dimensionless distance')
ylabel('Dimensionless concentration')
%
% Calculating the Sherwood number
%
h=0.0005;
for i=1:11
    Sh(i)=(1/(2*h))*(3*tri2grid(p,t,u(:,i),0,0)-4*tri2grid(p,t,u(:,i),h,0)+...
    tri2grid(p,t,u(:,i),2*h,0));
    fprintf('Dimensionless time=% .3f, Sh=% .3f\n', (i-1)*0.01, Sh(i))
end
```



Dimensionless time=0.000, Sh=10.000

Dimensionless time=0.010, Sh=4.984

Dimensionless time=0.020, Sh=3.732

Dimensionless time=0.030, Sh=3.113

Dimensionless time=0.040, Sh=2.724

Dimensionless time=0.050, Sh=2.455

Dimensionless time=0.060, Sh=2.251

Dimensionless time=0.070, Sh=2.090

Dimensionless time=0.080, Sh=1.959

Dimensionless time=0.090, Sh=1.851

Dimensionless time=0.100, Sh=1.759

Example 5-3-6

Concentration distribution of ethanol in a tube

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2}$$

$$c(x, 0) = 2,$$

$$c(0, t) = 0$$

$$\text{and } c(20, t) = 10.$$

Solution by MATLAB pdetool:

Step 1: After launching the PDE toolbox, first choose Grid and Snap in the Options menu. In the Axes limits dialogue box, the coordinate along the x axis is specified to [0 20], whereas the y axis is set as the option of Auto. Draw a rectangle with the length of 20 in the x direction and any width in the y direction. Note that only x direction is discussed here, regardless of the y direction.

Step 2: Press $\partial\Omega$ to activate the Boundary mode, and then change the left-hand side of the rectangle to Dirichlet and set $h = 1$ and $r = 0$. The right-hand side is set as Dirichlet also, whereas with $h = 1$ and $r = 10$. Finally, set the top and bottom boundaries to Neumann $g = q = 0$ with.

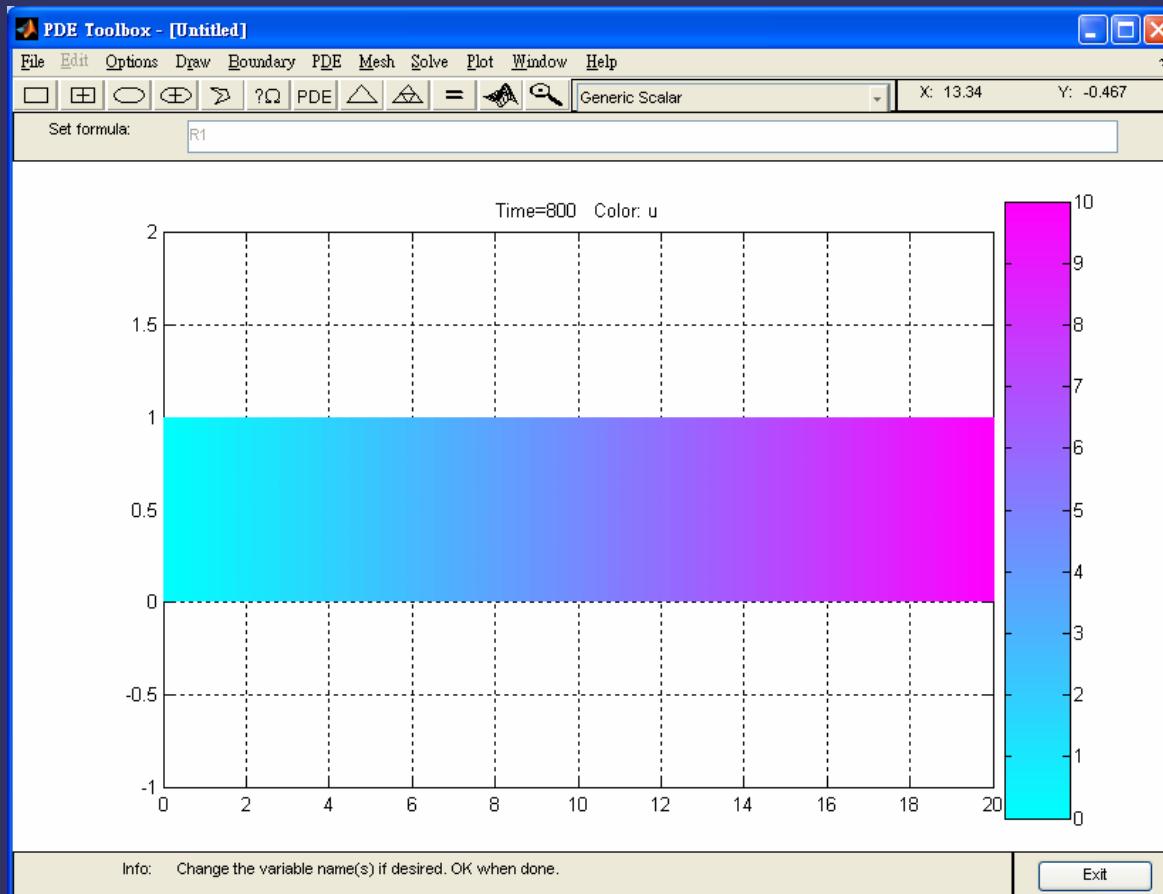
Solution by MATLAB pdetool:

Step 3: Press  to specify the PDE type as parabolic of which the coefficients are set to be $c = 0.119$, $d = 1$, and $a = f = 0$.

Step 4: Assign the time in Solve Parameters to 0: 50: 800; that is, the time from 0 to 800 is equally spaced with an interval of 50. Note that other interval sizes are also acceptable. Next, click the Parameter options under the Solve menu, and set the initial value $u(t0)$ as 2. Refine meshes by clicking  and press  to solve the PDE.

Solution by MATLAB pdetool:

Step 1: Export the Solution and the Mesh to the MATLAB Workspace.



ex5_3_6b.m

```
time=0: 50: 800;
for i=1: length(time)
    uxy(i,:)=tri2grid(p,t,u(:,i), 0: 4: 20, 0);
end
c=[time' uxy];
disp([' time      x=0      x=4      x=8      x=12      x=16      x=20'])
disp(c)
```

Execution results:

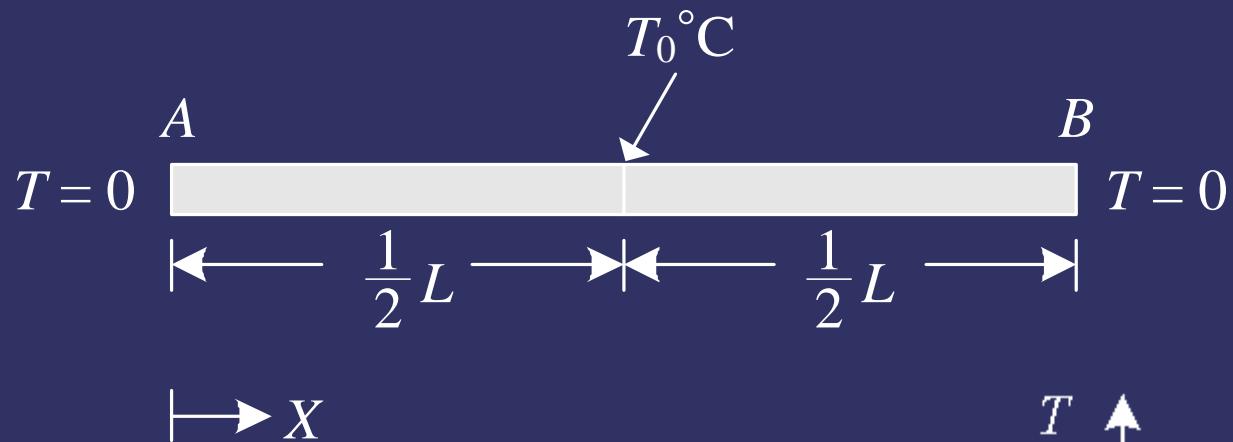
time	x=0	x=4	x=8	x=12	x=16	x=20
0	0	2.0000	2.0000	2.0000	2.0000	10.0000
50.0000	0	1.5003	1.9633	2.1544	3.9990	10.0000
100.0000	0	1.1794	1.9036	2.7832	5.3101	10.0000
150.0000	0	1.0484	1.9928	3.3586	6.0202	10.0000
200.0000	0	1.0334	2.1641	3.8083	6.4595	10.0000
250.0000	0	1.0836	2.3610	4.1596	6.7561	10.0000
300.0000	0	1.1628	2.5544	4.4394	6.9761	10.0000
350.0000	0	1.2504	2.7338	4.6689	7.1465	10.0000
400.0000	0	1.3369	2.8964	4.8608	7.2803	10.0000
450.0000	0	1.4179	3.0418	5.0239	7.3892	10.0000
500.0000	0	1.4925	3.1705	5.1618	7.4786	10.0000
550.0000	0	1.5596	3.2835	5.2798	7.5537	10.0000
600.0000	0	1.6191	3.3814	5.3796	7.6161	10.0000
650.0000	0	1.6711	3.4666	5.4659	7.6699	10.0000
700.0000	0	1.7163	3.5401	5.5397	7.7156	10.0000
750.0000	0	1.7552	3.6032	5.6029	7.7547	10.0000
800.0000	0	1.7887	3.6576	5.6573	7.7884	10.0000

Example 5-3-7

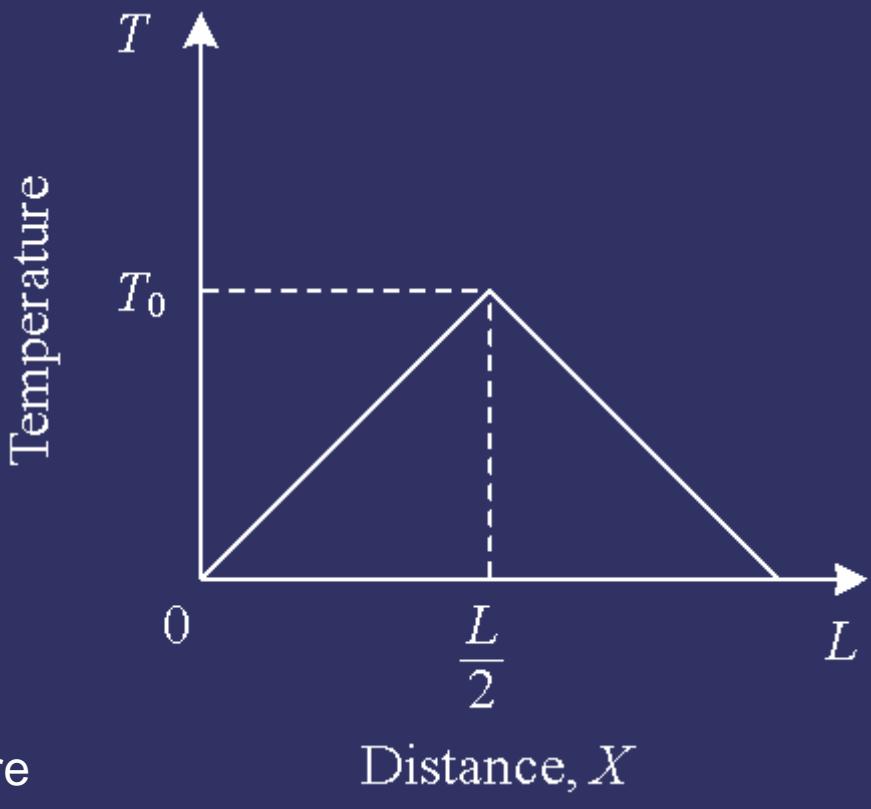
Heat conduction of a long rod

Figure 5.6 depicts a rod whose length L is considerably longer than its radius (*Leu*, 1985). The temperatures at both ends, A and B , of the rod are fixed and both kept at 0°C . The center of the rod is heated up to $T_0^\circ\text{C}$ ($T_0 > 0$) and then let the initial temperature distribution of the rod from the center to both ends follows a linear function, as shown in Figure 5.7. If the heating is ceased suddenly, determine the dimensionless temperature distribution u ($u = T/T_0$) of the rod with respect to time.

NOTE: The heat losses at both ends of the rod are negligible.



▲Figure 5.6 Heat conduction of a long rod.



►Figure 5.7 The initial temperature distribution.

Problem formulation and analysis:

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial X^2}$$

Initial condition:

$$T(X, 0) = \begin{cases} 2T_0(X / L), & 0 \leq X \leq 0.5L \\ 2T_0(1 - X / L), & 0.5L \leq X \leq L \end{cases}$$

Boundary condition:

$$T(0, t) = T(L, t) = 0, \quad \text{for } t \geq 0$$

$$x = X / L, u = T / T_0, \theta = (k / L^2)t,$$

$$\frac{\partial u}{\partial \theta} = \frac{\partial^2 u}{\partial x^2}$$

Problem formulation and analysis:

$$u(x, 0) = \begin{cases} 2x, & 0 \leq x \leq 0.5 \\ 2(1 - x), & 0.5 \leq x \leq 1 \end{cases}$$

analytical solution

$$u(0, \theta) = u(1, \theta) = 0 \quad \text{for } \theta \geq 0$$

Solution by MATLAB pdetool:

In what follows, the solution steps using pdetool are demonstrated.

Step 1: After launching the PDE toolbox, draw a rectangle from $(0, 0)$ with the length of 1 and any width. It is noted that, since the rod is considerably long, only x direction is discussed for heat conduction and there is no concern with the y direction.

Step 2: Press $\partial\Omega$ to activate the Boundary mode, and then specify the boundary conditions on both ends to Dirichlet with $h = 1$ and $r = 0$. The top and bottom boundaries, where there is no heat flux, are set as Neumann with .

Solution by MATLAB pdetool:

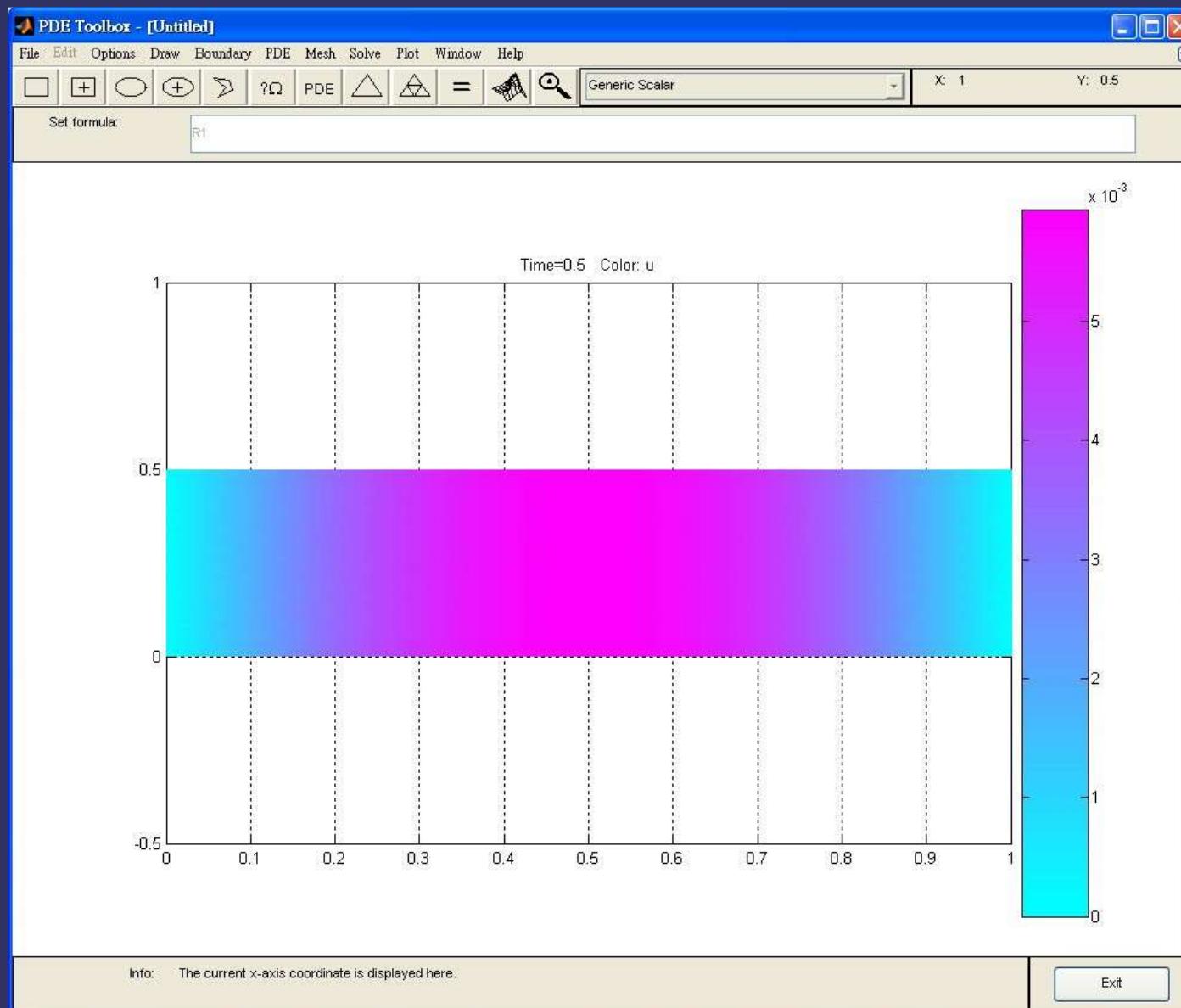
Step 3: Press **PDE** to specify the PDE type as parabolic and set the PDE coefficients as follows: $c = d = 1$ and $a = f = 0$.

Step 4: Enter Mesh mode to refine the meshes once. Change the simulation time in Solve Parameters to 0:0.01:0.5 (from 0 to 0.5 with an interval of 0.01). The initial value is set to be $2 * (0.5 - \text{abs}(x - 0.5))$ based on (5.3-49) and then solve the PDE.

Step 5: Export the Solution and Mesh to the MATLAB Workspace.

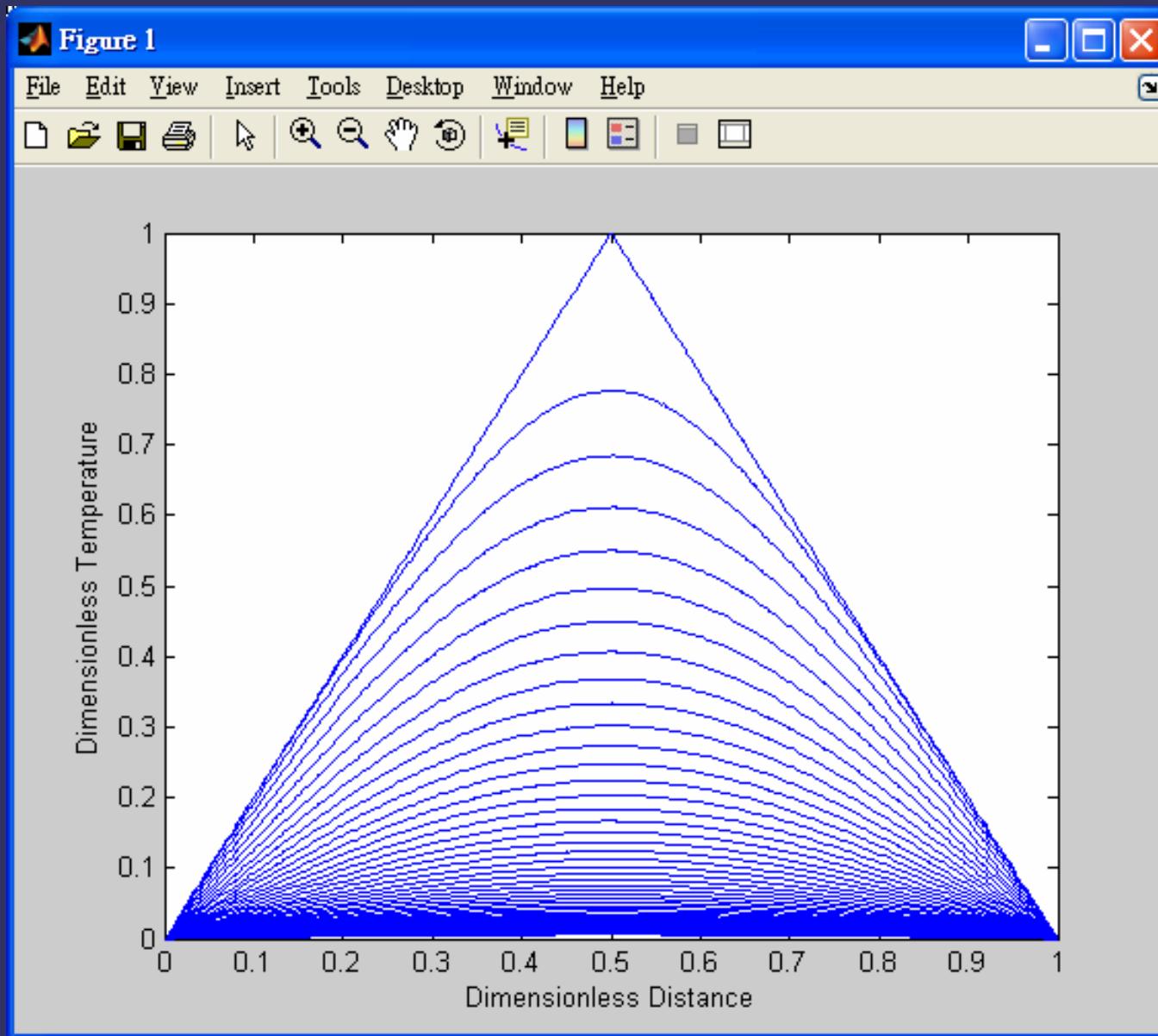
Numerical Solution of Partial Differential Equations

05



ex5_3_7b.m

```
xx=p(1,:);
[xs, I]=sort(xx);
for i=1:51
ui=u(:, i);
usi=ui(I);
plot(xs, usi)
hold on
end
xlabel('Dimensionless Distance')
ylabel('Dimensionless Temperature')
```



Example 5-3-8

Unsteady-state heat conduction in a flat panel

An ethylene chloride panel, whose width is considerably larger than its thickness, is immersed in a water bath for cooling. The thickness and the initial temperature of the panel are and respectively. The water temperature is kept constant at during the cooling process. Besides, the *thermal diffusivity* of the ethylene chloride panel is measured to be Based on the above-mentioned operating condition, determine the temperature distribution along the thickness of the panel with respect to time.



Problem formulation and analysis:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

$$T(x, 0) = T_0 \quad \text{for } 0 \leq x \leq L$$

$$T(0, t) = T_1 \quad \text{for } t \geq 0$$

$$\left. \frac{\partial T}{\partial x} \right|_{x=L} = 0 \quad \text{for } t \geq 0$$

Solution by MATLAB pdetool:

Step 1: After launching the PDE toolbox GUI, first specify the coordinate along the x axis to $[0 \ 0.012]$ and that of y axis to $[0 \ 0.04]$. Then, draw a rectangle with the length of 0.012 (half of its thickness) and a width of 0.04. Note that the width is arbitrary since only x direction is discussed, regardless of the y direction.

Step 2: Enter Boundary mode, set the left-hand side to Dirichlet with $h = 1$ and $r = 20$. The boundary conditions at the right-hand side, where $x = L$, are assigned to be Neumann with $g = q = 0$. On the other hand, the top and bottom boundaries are also specified as Neumann with $g = q = 0$.

Solution by MATLAB pdetool:

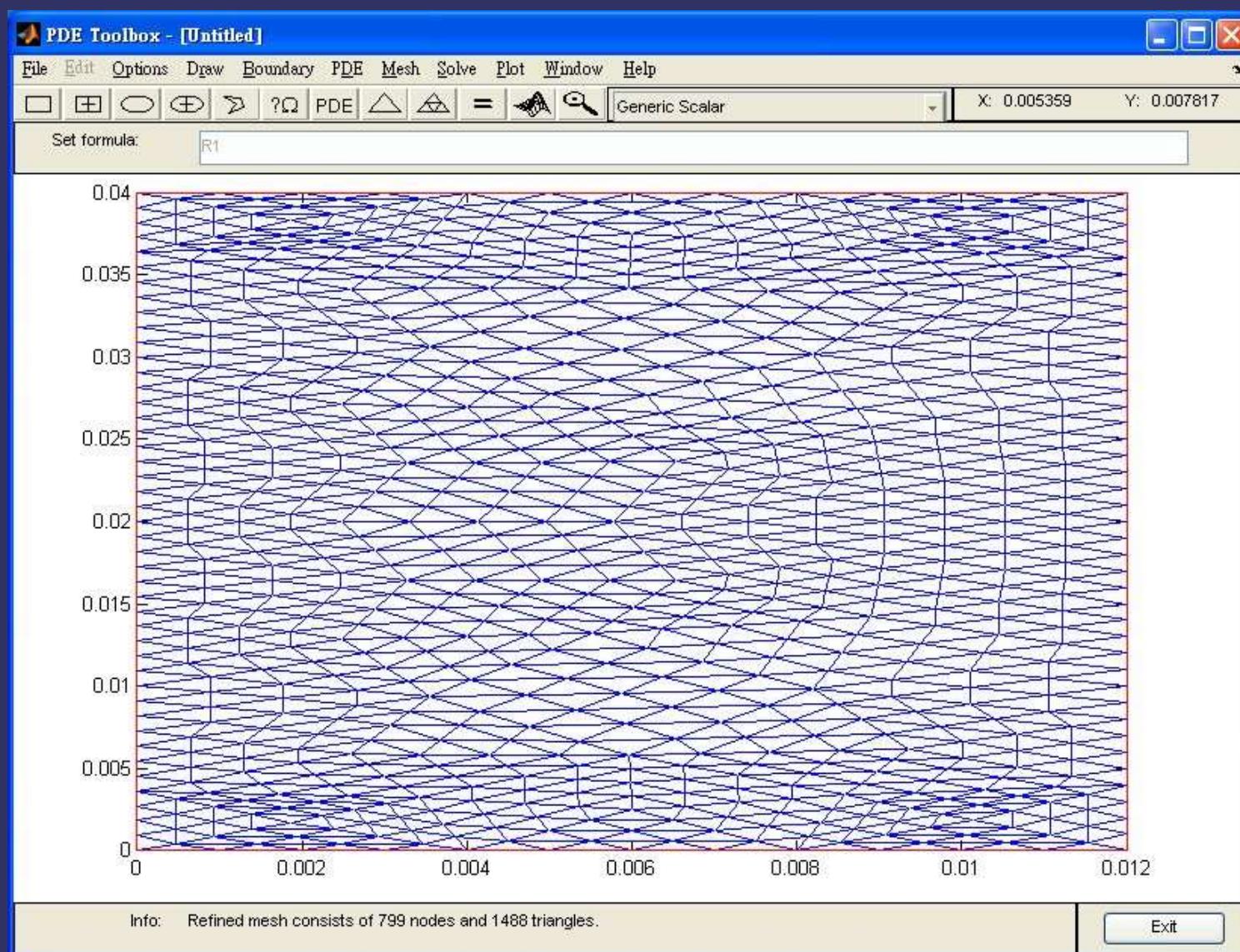
Step 3: Specify the PDE type as parabolic, and the coefficients are set as $c = 0.0005$, $d = 1$ and $a = f = 0$.

Step 4: Initialize the meshes once and refine them twice to ensure the solution accuracy. It is noted that, if the meshes are not refined twice, significant error may occur because of the huge temperature difference within the considerably small thickness of the panel. Then, assign the simulation time in Solve Parameters to $0:0.006:0.252$, i.e., from 0 to 0.252 with an interval of 0.006. Set initial value $u(t0)$ at 70 and solve this PDE.

Step 5: Export the Solution and Mesh to the MATLAB Workspace.

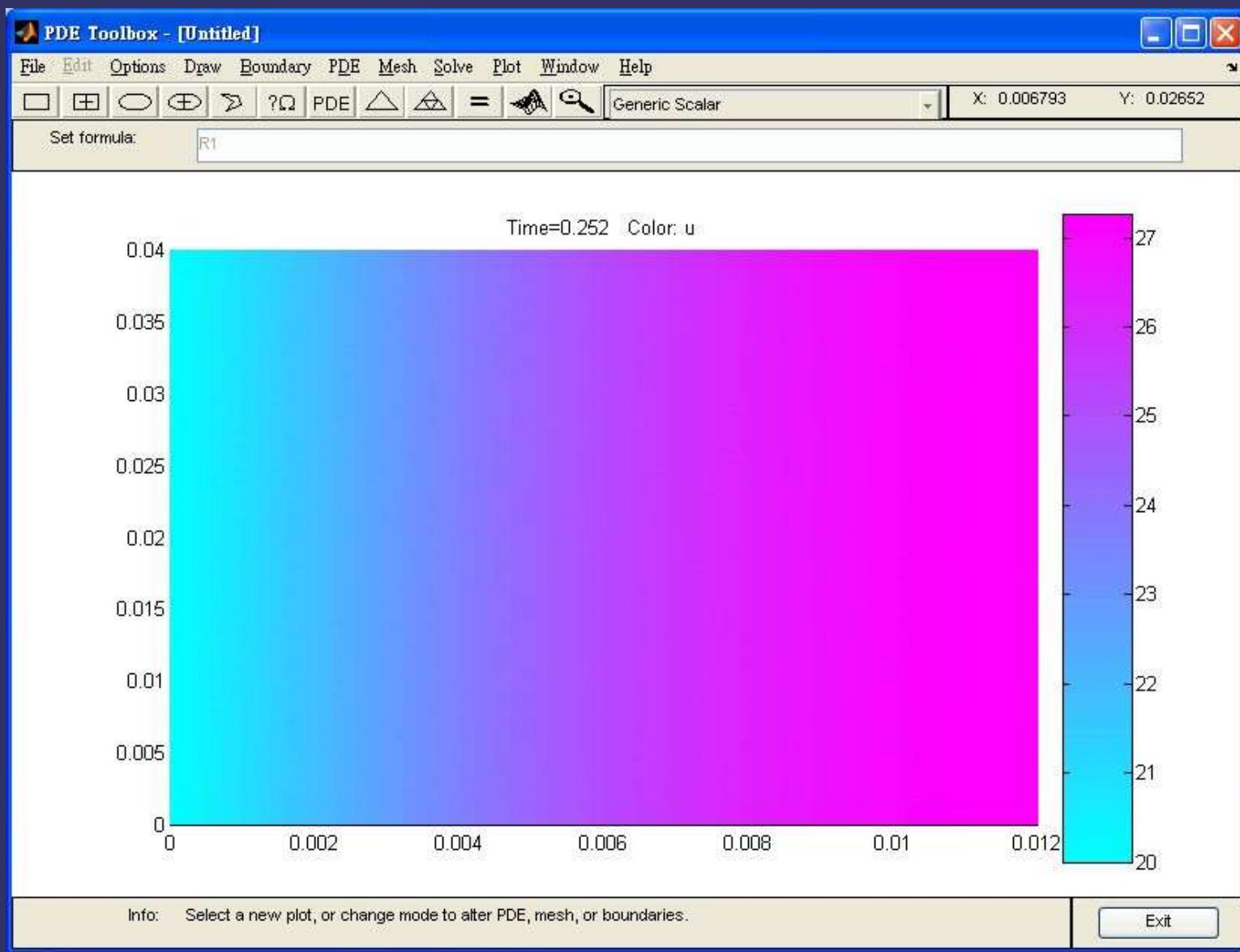
Numerical Solution of Partial Differential Equations

05



Numerical Solution of Partial Differential Equations

05



ex5_3_8b.m

```
disp('time x=0.000 x=0.003 x=0.006 x=0.009 x=0.012')
```

```
for i=1:43
```

```
uxy=tri2grid(p, t, u(:,i), 0:0.003:0.012, 0);
```

```
disp(sprintf(' %.4f ',(i-1)*0.006, uxy))
```

```
end
```

Execution results:

time	x=0.000	x=0.003	x=0.006	x=0.009	x=0.012
0.0000	20.0000	70.0000	70.0000	70.0000	70.0000
0.0060	20.0000	58.7539	69.3959	69.9925	69.9999
0.0120	20.0000	50.4702	65.8711	69.5706	69.9556
0.0180	20.0000	45.8484	62.1027	68.3331	69.5764
0.0240	20.0000	42.8565	58.8873	66.6102	68.6490
0.0300	20.0000	40.6566	56.1905	64.7037	67.2663
0.0360	20.0000	38.9872	53.8967	62.7190	65.5730
0.0420	20.0000	37.6590	51.9030	60.7283	63.7011
0.0480	20.0000	36.5469	50.1260	58.7766	61.7570
0.0540	20.0000	35.5540	48.4874	56.8928	59.8331
0.0600	20.0000	34.6666	46.9741	55.0799	57.9431
0.0660	20.0000	33.8824	45.5833	53.3365	56.0865
0.0720	20.0000	33.1446	44.2657	51.6738	54.3111
0.0780	20.0000	32.4793	43.0438	50.0862	52.5944
0.0840	20.0000	31.8397	41.8748	48.5751	50.9641
0.0900	20.0000	31.2450	40.7760	47.1399	49.4090
0.0960	20.0000	30.6794	39.7309	45.7744	47.9293
0.1020	20.0000	30.1425	38.7389	44.4785	46.5250

Execution results:

time	x=0.000	x=0.003	x=0.006	x=0.009	x=0.012
0.1080	20.0000	29.6312	37.7937	43.2436	45.1867
0.1140	20.0000	29.1410	36.8891	42.0630	43.9080
0.1200	20.0000	28.6772	36.0326	40.9448	42.6965
0.1260	20.0000	28.2398	35.2243	39.8888	41.5523
0.1320	20.0000	27.8254	34.4585	38.8882	40.4679
0.1380	20.0000	27.4266	33.7218	37.9261	39.4255
0.1440	20.0000	27.0492	33.0248	37.0157	38.4390
0.1500	20.0000	26.6934	32.3674	36.1570	37.5085
0.1560	20.0000	26.3582	31.7481	35.3480	36.6318
0.1620	20.0000	26.0341	31.1494	34.5660	35.7845
0.1680	20.0000	25.7274	30.5827	33.8258	34.9825
0.1740	20.0000	25.4380	30.0481	33.1274	34.2257
0.1800	20.0000	25.1659	29.5454	32.4708	33.5142
0.1860	20.0000	24.9034	29.0605	31.8373	32.8277
0.1920	20.0000	24.6541	28.5999	31.2356	32.1757
0.1980	20.0000	24.4187	28.1649	30.6674	31.5600
0.2040	20.0000	24.1973	27.7557	30.1328	30.9807
0.2100	20.0000	23.9865	27.3664	29.6242	30.4295

Execution results:

time	x=0.000	x=0.003	x=0.006	x=0.009	x=0.012
0.2160	20.0000	23.7858	26.9954	29.1395	29.9042
0.2220	20.0000	23.5953	26.6434	28.6796	29.4059
0.2280	20.0000	23.4144	26.3093	28.2431	28.9329
0.2340	20.0000	23.2429	25.9923	27.8289	28.4840
0.2400	20.0000	23.0802	25.6917	27.4362	28.0584
0.2460	20.0000	22.9257	25.4062	27.0632	27.6543
0.2520	20.0000	22.7790	25.1350	26.7089	27.2703

5.5 Summary of the MATLAB commands related to this chapter

Command	Function
pdepe	Find numerical solutions to one-dimensional PDEs
pdeval	Evaluate the solution at some other specified points based on using the results obtained from pdepe
pdetool	The graphical interface solution tool for two-dimensional PDEs
tri2grid	Interpolate the solution at a specific point based on the simulation results obtained from pdetool



Applications in Chemical Engineering

Chyi-Tsong Chen



Chapter 6

Process Optimization

版權所有・請勿翻製

- ① to introduce the optimization commands
- ② describe how to use them to solve diversified optimization problems that arose from the field of chemical engineering.
introduce a population-based real-coded genetic algorithm (RCGA)

6.1 The optimization problem and the relevant MATLAB commands

► 6.1.1 Optimization problems of a single decision variable

$$\min_{a \leq x \leq b} f(x)$$

[x, fval]= fminbnd('fun', a, b, options)

Output argument	Description
x	The obtained optimal solution in the feasible range [a b]
fval	The objective function value corresponding to the obtained optimal point x

Input argument

Description

The name of the function file provided by the user, and the format of the function is as follows:

function f=fun(x)

f= ... % objective function

fun

Besides, if the objective function is not complicated it can be directly given by the inline command, see Example 6-1-1 for detailed usage.

a

Lower limit of the decision variable

b

Upper limit of the decision variable

options

Solution optional parameters provided by the users, such as the maximal iteration number, the accuracy level and the gradient information, and so on. Refer to the command ‘optimset’ for its usage.

Example 6-1-1

Solve the following optimization problem

$$\min_{0 \leq x \leq 2} x^3 + x^2 - 2x - 5$$

Ans:

```
>> f=inline('x.^3+x.^2-2*x-5'); % use inline to provide the objective  
function  
>> x=fminbnd(f, 0, 2)  
x=  
0.5486  
>> y=f(x)  
y=  
-5.6311
```

6.1.2 Multivariate optimization problems without constraints

$$\min_{\mathbf{x}} f(\mathbf{x})$$

```
[x, fval]= fminunc('fun', x0, options)
```

Example 6-1-2

Solve the following unconstrained optimization problem of two variables

$$\min_{x_1, x_2} 3x_1^2 + 2x_1 x_2 + 5x_2^2$$

Ans:

Step 1: Provide the objective function file

Ans:

ex6_1_2f.m

```
function f=fun(x)
x1=x(1);
x2=x(2);
f=3*x1^2+2*x1*x2+5*x2^2; % objective function
```

Step 2: Provide the objective function file

>> x0=[1 1]; % initial guess value

>> [x, fval]=fminunc('ex6_1_2f', x0)

x=

1.0e-006 *
-0.2047 0.2144

fval =

2.6778e-013

Ans:

ex6_1_2f2.m

```
function [f, G]=fun(x)
x1=x(1);
x2=x(2);
f=3*x1^2+2*x1*x2+5*x2^2; % objective function
if nargout > 1 % the number of the output arguments is more than 1
    G=[6*x1+2*x2 % partial differentiation with respect to x1
        2*x1+10*x2]; % partial differentiation with respect to x2
end
```

Ans:

```
>> options=optimset('GradObj', 'on'); % declare a gradient has been provided  
>> x0=[1 1]; % initial guess  
>> [x, fval]=fminunc('ex6_1_2f2', x0, options)
```

```
x =  
1.0e-015 *  
-0.2220 0.1110
```

```
fval =  
1.6024e-031
```

```
[x, fval]=fminsearch('fun', x0, options)
```

6.1.3 Linear programming problems

$$\min_{\mathbf{x}} \quad \mathbf{f}^T \mathbf{x}$$

s.t.

$$\mathbf{A} \mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq} \mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

[x, fval]=linprog(f, A, b, Aeq, beq, xL, xU, x0, options)

Example 6-1-3

Solve the following linear programming problem

$$\min_x -5x_1 - 4x_2 - 3x_3$$

s.t.

$$x_1 - x_2 + 2x_3 \leq 30$$

$$3x_1 + 2x_2 + x_3 \leq 40$$

$$2x_1 + 3x_2 \leq 20$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_3 \geq 0$$

Example 6-1-3

Ans:

$$\mathbf{f} = [-5 \quad -4 \quad -3]^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 1 \\ 2 & 3 & 0 \end{bmatrix}$$

$$\mathbf{b} = [30 \quad 40 \quad 20]^T$$

and

$$\mathbf{x}_L = [0 \quad 0 \quad 0]^T$$

Ans:

```
>> f=[-5 -4 -3]';  
>> A=[1 -1 2; 3 2 1; 2 3 0];  
>> b=[30 40 20]';  
>> Aeq=[ ];  
>> beq=[ ];  
>> xL=[0 0 0]';  
>> xU=[ ];  
>> [x, fval]=linprog(f, A, b, Aeq, beq, xL, xU)
```

x=

0.0000
6.6667
18.3333

fval =

-81.6667

6.1.4 Quadratic programming problem

$$\min_{\mathbf{x}} \quad \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x}$$

s.t.

$$\mathbf{A} \mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq} \mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

`[x, fval] =quadprog(H, f, A, b, Aeq, beq, xL, xU, x0, options)`

Example 6-1-4

$$\min_{\mathbf{x}} \quad \frac{1}{2}x_1^2 + x_2^2 - x_1 x_2 - 6x_1 - 2x_2$$

s.t.

$$x_1 - x_2 \leq 1$$

$$-x_1 + 2x_2 \leq 2$$

$$2x_1 + x_2 \leq 3$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Ans:

$$\frac{1}{2}x_1^2 + x_2^2 - x_1 x_2 - 6x_1 - 2x_2 = \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x}$$

where

$$\mathbf{x} = [x_1 \quad x_2]^T$$

$$\mathbf{H} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$$

$$\mathbf{f} = [-6 \quad -2]^T$$

$\mathbf{A} = [1 \ -1 ; -1 \ 2; 2 \ 1]$, $\mathbf{b} = [1 \ 2 \ 3]'$, $\mathbf{A}\mathbf{eq} = []$ and $\mathbf{beq} = []$.

Ans:

```
>> H=[1 -1 ; -1 2];
>> f=[-6 -2]';
>> A=[1 -1; -1 2; 2 1]; % inequality coefficient matrix
>> b=[1 2 3]'; % inequality vector
>> xL=[0 0]'; % lower limit of the variable
>> [x, fval]=quadprog(H, f, A, b, [ ], [ ], xL, [ ])
```

x=

1.3077

0.3846

fval =

-8.1154

Ans:

NOTES:

The determination of the **H** matrix of the quadratic objective function (6.1-11) can be easily accomplished by using the Symbolic Math operations as follows.

```
>> syms x1 x2 % declare x1 and x2 as symbolic variables  
>> f=1/2*x1^2+x2^2-x1*x2; % define the quadratic function  
>> H=jacobian(jacobian(f)) % calculate the H matrix
```

H =

$$\begin{bmatrix} 1, & \square 1 \\ \square 1, & 2 \end{bmatrix}$$

6.1.5 The constrained nonlinear optimization problems

$$\min_{\mathbf{x}} f(\mathbf{x})$$

s.t.

$$\mathbf{c}(\mathbf{x}) \leq 0$$

$$\mathbf{c}_{eq}(\mathbf{x}) = 0$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

```
[x, fval]= fmincon('fun', x0, A, b, Aeq, beq, xL, xU, 'nonlcon', options)
```

- the objective function,

```
function f=fun(x)
f= ... % the objective function
```

- The nonlinear constraints, $c(x)$ and $c_{eq}(x)$,

```
function [c, ceq]=nonlcon(x)
c=... % the nonlinear inequality constraints
ceq=... % the nonlinear equality constraints
```

- the gradient information

```
options=optimset('Gradconstr', 'on')
```

```
function [c, ceq, Gc, Gceq]=nonlcon(x)
c=... % nonlinear inequality constraints
ceq=... % nonlinear equality constraints
if nargout > 2 % since gradients are provided
    Gc=... % the gradients of the inequality constraints (in vector form)
    Gceq=... % the gradient of the equality constraints (in vector form)
end
```

Example 6-1-5

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = -x_1 x_2^2 x_3$$

s.t.

$$-x_1 - 2x_2 - 2x_3 \leq 0$$

$$x_1 + 2x_2 + 2x_3 \leq 72$$

$$x_2 x_3 = 144$$

$$x_1 x_3 - \frac{1}{2} x_2^2 \leq 380$$

Ans:

ex6_1_5.m

```
function ex6_1_5
%
% Example 6-1-5 Solve an optimization problem with fmincon
%
clear
clc
%
% initial guess values
%
x0=[10 10 10]';
%
% coefficients for the inequality constraints
%
A=[-1 -2 -2; 1 2 2];
b=[0 72]';
%
% solving the optimization problem with fmincon
```

Ans:

— ex6_1_5.m —

```
%  
[x, fval]=fmincon(@objfun, x0, A, b, [ ], [ ], [ ], [ ], @nonlcon);  
x1=x(1);  
x2=x(2);  
x3=x(3);  
fprintf('\n The optimal solution is x1=%.3f, x2=%.3f, x3=%.3f,...  
,\n and the minimum objective function is f=%.3f\n', x1, x2, x3, fval)  
%  
% objective function  
%  
function f=objfun(x)  
x1=x(1);  
x2=x(2);  
x3=x(3);  
f=-x1*x2^2*x3;  
%  
% nonlinear constraints
```

Ans:

ex6_1_5.m

```
%  
function [c, ceq]=nonlcon(x)  
x1=x(1);  
x2=x(2);  
x3=x(3);  
c=x1*x3-0.5*x2^2-380;  
ceq=x2*x3-144;
```

Execution results:

```
>> ex6_1_5
```

The optimal solution is $x_1=20.000$, $x_2=18.000$, $x_3=8.000$,
and the minimum objective function is $f=-51840.000$

6.1.6 The multi-objective goal attainment problem

$$\min_{\mathbf{x}, r} \quad r$$

s.t.

$$\mathbf{f}(\mathbf{x}) - \mathbf{w}r \leq \text{goal}$$

$$\mathbf{c}(\mathbf{x}) \leq 0$$

$$\mathbf{c}_{eq}(\mathbf{x}) = 0$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

[x, fval, r]=fgoalattain('fun', x0, goal, w, A, b, Aeq, beq, xL, xU, 'nonlcon', options)

Example 6-1-6

Design of a pole placement controller

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

$$\mathbf{y} = \mathbf{Cx}$$

$$\mathbf{A} = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

feedback control law $\mathbf{u} = \mathbf{K}\mathbf{y}$

the closed-loop system

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{B}\mathbf{K}\mathbf{C})\mathbf{x}$$

To avoid aggressive control inputs, the absolute value of each element in the gain matrix \mathbf{K} is required to be less than 4.

Determine the controller's gain matrix \mathbf{K} if the desired closed-loop poles, i.e., the *eigenvalues* of the matrix $\mathbf{A} + \mathbf{B}\mathbf{K}\mathbf{C}$, are located at -5 , -3 , and -1 .

Problem formulation and MATLAB program design:

goal = [-5, -3, -1]

F=sort(eig(A+B*K*C))

w=abs(goal)

and

$$\mathbf{K} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$$

ex6_1_6

```
function ex6_1_6
%
% Example 6-1-6 Design of a pole placement controller
%
clear
clc
%
global A B C % declared as the global variables
%
% system matrices
%
A=[-0.5 0 0 ;0 -2 10 ;0 1 -2];
B=[1 0 ;-2 2 ;0 1 ];
C=[1 0 0 ;0 0 1 ];
%
% initial guess values of the controller parameter
%
K0=[-1 -1 ;-1 -1 ];
```

ex6_1_6

```
% the desired poles and the weighting vector
%
goal=[-5 -3 -1]; % vector of the desired poles
w=abs(goal); % weighting vector
%
% the lower and upper bounds of the decision parameters
%
xL=-4*ones(size(K0)); % lower limit of the controller parameters
xU=4*ones(size(K0)); % upper limit of the controller parameters
%
% solving the multi-objective goal attainment problem
%
% options=optimset('GoalsExactAchieve',3);
% [K,fval,r]=fgoalattain(@eigfun,K0,goal,w,[ ],[ ],[ ],xL,xU,[ ],options)
[K,fval,r]=fgoalattain(@eigfun,K0,goal,w,[ ],[ ],[ ],xL,xU)
%
% multi-objective function vector
%
```

ex6_1_6

```
function f=eigfun(K)
global A B C
f=sort(eig(A+B*K*C));
```

Execution results:

```
>> ex6_1_6
```

K=

```
-4.0000 -0.2564  
-4.0000 -4.0000
```

fval =

```
-6.9313  
-4.1588  
-1.4099
```

r=

```
-0.3863
```

Execution results:

```
>> ex6_1_6
```

replace the original command of fgoalattain with the following two lines:

```
options=optimset ('GoalsExactAchieve', 3);  
[K, fval, r]=fgoalattain(@eigfun, K0, goal, W, [ ], [ ], [ ], [ ], xL, xU, [ ], options)
```

Execution results:

```
>> ex6_1_6
```

```
K =
```

```
-1.5953  1.2040  
-0.4201 -2.9047
```

```
fval =
```

```
-5.0000  
-3.0000  
-1.0000
```

```
r =
```

```
4.1222e-021
```

6.1.7 Semi-infinitely constrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x})$$

s.t.

$$\mathbf{c}(\mathbf{x}) = 0$$

$$\mathbf{c}_{eq}(\mathbf{x}) = 0$$

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{Aeq}\mathbf{x} = \mathbf{beq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

$$k_1(\mathbf{x}, w_1) \leq 0$$

$$k_2(\mathbf{x}, w_2) \leq 0$$

⋮

$$k_n(\mathbf{x}, w_n) \leq 0$$

```
[x, fval]=fseminf('fun' , x0, n, 'seminfcon' , A, b, Aeq, beq, xL, xU, options)
```

```
function f = fun(x)
f = ... % objective function
```

```
options=optimset('GradObj' , 'on ')
```

```
function [f, G]=fun(x)
f= ... % the objective function
if nargout>1 % due to the gradient information is provided
G =... % the gradient formula
end
```

```
function [c, ceq, k1, k2, ... , kn, S]=seminfcon(x, S)
%
% given S matrix
%
if isnan(S(1, 1))
S =... % S is a matrix with n rows and 2 columns
end
%
% mesh points of wi based on S matrix
%
w1=... ; %
w2= ... ; %
...
```

```
wn= ... ; %  
%  
% calculate the semi-infinite inequalities  
%  
k1=...%  
k2=...%  
...  
kn=...%  
%  
% other nonlinear constraints  
%  
c = ... % inequality constraints  
ceq= ...% equality constraints
```

Example 6-1-7

Solve the following semi-infinite optimization problem
(MATLAB User's Guide, 1997)

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2$$

s.t.

$$k_1(x, w_1) = \sin(w_1 x_1) \cos(w_1 x_2) - \frac{1}{1000} (w_1 - 50)^2 - \sin(w_1 x_3) - x_3 - 1 \leq 0$$

$$k_2(x, w_2) = \sin(w_2 x_2) \cos(w_2 x_1) - \frac{1}{1000} (w_2 - 50)^2 - \sin(w_2 x_3) - x_3 - 1 \leq 0$$

where the values of w_1 and w_2 both lie in the range of 1 and 100.

Example 6-1-7

Ans:

Step 1: Prepare the objective function file:

```
function f=fun6_1_7(x)
f=sum((x-0.5).^2);
```

Step 2: Prepare a function file that contains the semi-infinite constraints:

```
function [c, ceq, k1 , k2, S]=seminfcon6_1_7(x, S)
%
% S matrix: distance between meshes
%
if isnan(S(1, 1))
```

Ans:

```
S=[0.1  0  
    0.1  0]; % NOTE: the second column is a zero  
vector  
end  
%  
% meshes for wi  
%  
w1=1:S(1,1):100;  
w2=1:S(2,1):100;  
%  
% semi-infinite constraints  
%  
k1=sin(w1*x(1)).*cos(w1*x(2))-1/1000*(wl-50).^2-  
sin(w1*x(3))-x(3)-1;
```

Ans:

```
k2=sin(w2*x(2)).*cos(w2*x(1))-1/1000*(w2-50).^2-
sin(w2*x(3))-x(3)-1;
%
% other nonlinear constraints
%
c=[ ]; % no other nonlinear constraints
ceq=[ ];
```

Step 3: Combine the subroutines

 ex6_7.m

```
function ex6_1_7
%
% Example 6-1-7 Semi-infinite optimization
%
clear
clc
%
x0=[0.5 0.2 0.3]'; % initial guess values
%
% solve the semi-infinite optimization problem
%
[x, fval]=fseminf(@fun6_1_7, x0, 2, @seminfcon6_1_7)
%
% objective function
%
function f=fun6_1_7(x)
f=sum((x-0.5).^2);
%
% constraints
```

 ex6_7.m

```
%  
function [c, ceq, k1, k2, S]=seminfcon6_1_7(x, S)  
%  
% S matrix: distance between meshes  
%  
if isnan(S(1,1))  
S=[0.1 0  
    0.1 0]; % NOTE: the second column is a zero vector  
end  
%  
% meshes for wi  
%  
w1=1:S(1,1):100;  
w2=1:S(2,1):100;  
%  
% semi-infinite constraints  
%  
k1=sin(w1*x(1)).*cos(w1*x(2))-1/1000*(wl-50).^2-sin(w1*x(3))-x(3)-1;  
k2=sin(w2*x(2)).*cos(w2*x(1))-1/1000*(w2-50).^2-sin(w2*x(3))-x(3)-1;
```

ex6_7.m

```
%  
% other nonlinear constraints  
%  
c=[ ]; % no other nonlinear constraints  
ceq=[ ];
```

Ans:

Step 4: Execute the program to get the optimal solution

```
>> ex6_1_7
```

X =

0.6581

0.2972

0.3962

fval =

0.0769

Example 6-1-8

Solve the following two-dimensional semi-infinite optimization problem (*MATLAB User's Guide*, 1997)

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2$$

s.t.

$$k(x, \mathbf{w}) = \sin(w_1 x_1) \cos(w_1 x_2) - \frac{1}{1000} (w_1 - 50)^2 - \sin(w_1 x_3)$$

$$+ \sin(w_2 x_2) \cos(w_1 x_1) - \frac{1}{1000} (w_2 - 50)^2 \\ - \sin(w_2 x_3) - 2x_3 - 1.5 \leq 0$$

where $\mathbf{w} = [w_1 \ w_2]$ with $1 \leq w_1 \leq 100$ and $1 \leq w_2 \leq 100$.

Example 6-1-8

Ans:

ex6_1_8.m

```
function ex6_1_8
%
% Example 6-1-8 Two-dimensional semi-infinite optimization
%
clear; close all;
clc
%
x0=[0.5 0.5 0.5]'; % initial guess values
%
% solve the 2D semi-infinite optimization problem
%
[x, f]=fseminf(@fun6_1_8, x0, 1, @seminfcon6_1_8);
%
[c, ceq, k]=seminfcon6_1_8(x, [0.2 0.2]);
kmax=max(max(k)); % the maximum value in the constraint
```

Ans:

ex6_1_8.m

```
function ex6_1_8
%
% Example 6-1-8 Two-dimensional semi-infinite optimization
%
clear; close all;
clc
%
x0=[0.5 0.5 0.5]'; % initial guess values
%
% solve the 2D semi-infinite optimization problem
%
[x, f]=fseminf(@fun6_1_8, x0, 1, @seminfcon6_1_8);
%
[c, ceq, k]=seminfcon6_1_8(x, [0.2 0.2]);
kmax=max(max(k)); % the maximum value in the constraint
%
% Results printing
%
```

Ans:

ex6_1_8.m

```
fprintf('\n if x1=% .3f, x2=% .3f, x3=% .3f, minimal value is...')
```

```
f= % .3e',x(1),x(2),x(3),f)
```

```
fprintf ('\n the maximal value in the constraint is %.3f\n', kmax)
```

```
%
```

```
% objective function
```

```
%
```

```
function f=fun6_1_8(x)
```

```
f=sum((x-0.5).^2);
```

```
%
```

```
% constraints
```

```
%
```

```
function [c, ceq, k, S]=seminfcon6_1_8(x, S)
```

```
%
```

```
% S matrix: distance between meshes
```

```
%
```

```
if isnan(S(1, 1))
```

```
S=[0.2 0.2];
```

Ans:

ex6_1_8.m

```
end
%
% meshes
%
w1=1:S(1,1):100;
w2=1:S(1,2):100;
[W1, W2]=meshgrid(w1, w2); % meshgrid for interlacing meshes in 2D
%
% semi-infinite constraints
%
k=sin(W1*x(1)).*cos(W1*x(2))-1/1000*(W1-50).^2-sin(W1*x(3))+...
    sin(W2*x(2)).*cos(W1*x(1))-1/1000*(W2-50).^2-sin(W2*x(3))-2*x(3)-1.5;
%
c=[ ]; % no other nonlinear constraints
ceq=[ ];
%
% plotting during execution
%
```

Ans:

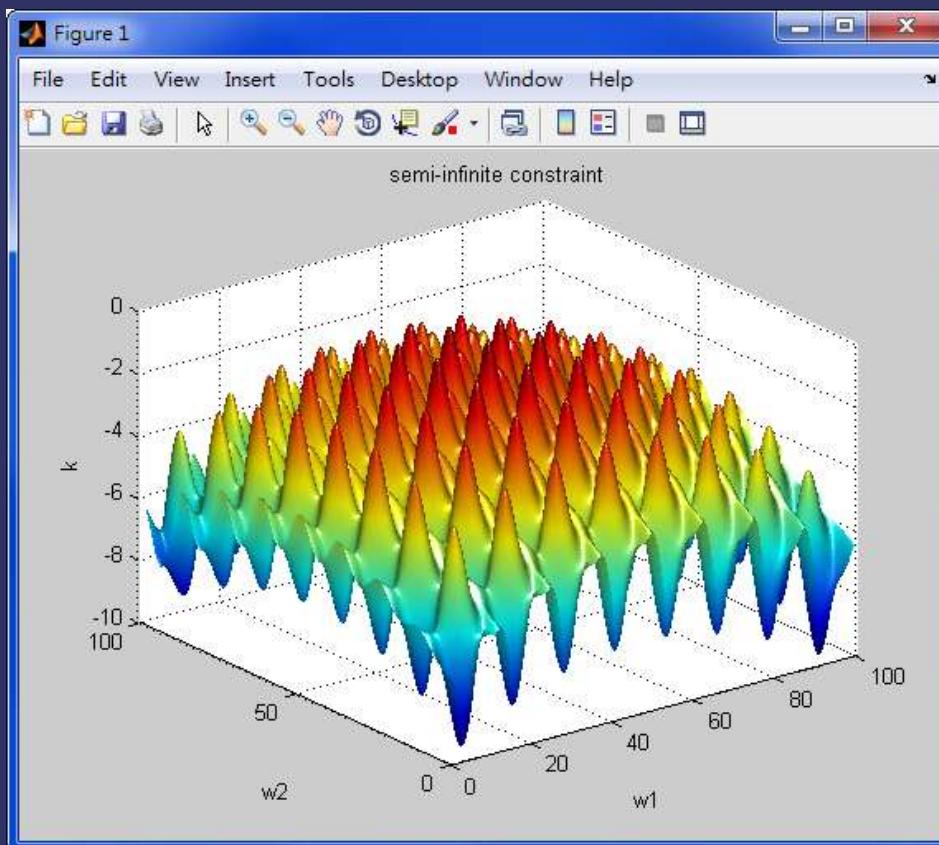
— ex6_1_8.m —

```
m=surf(W1, W2, k, 'edgecolor', 'non', 'facecolor', 'interp');  
camlight headlight  
xlabel('w1')  
ylabel('w2')  
zlabel('k')  
title('semi-infinite constraint')
```

Execution results:

```
>> ex6_1_8
```

if $x_1=0.503$, $x_2=0.506$, $x_3=0.492$, minimal value is $f=9.803e-005$
the maximal value in the constraint is -0.147



6.1.8 The minimax problem

$$\min_{\mathbf{x}} \max_i \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})\}$$

s.t.

$$\mathbf{c}(\mathbf{x}) \leq 0$$

$$\mathbf{c}_{eq}(\mathbf{x}) = 0$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

```
[x, f, maxf] =fminimax('fun', x0, A, b, Aeq, beq, xL, xU, 'nonlcon', options)
```

- objective functions

```
function f=fun(x)
f(1) =... % the first objective function
f(2) =... % the second objective function
⋮
f(n) =... % the n-th objective function
```

```
options = optimset('GradObj', 'on', 'GradConstr', 'on')
```

```
function [f, G] = fun(x)
f=[ ... % the first objective function
    ... % the second objective function
    ⋮
    ... % the n-th objective function
]
```

```

if nargout > 1
    G=[ ... %  $\partial f_1 / \partial \mathbf{x}$ 
        ... %  $\partial f_2 / \partial \mathbf{x}$ 
        ...
        ... %  $\partial f_n / \partial \mathbf{x}$ 
    ]
end

```

```

function [c, ceq, Gc, Gceq]=nonlcon(x)
c =... % the nonlinear inequality constraints
ceq =... % the nonlinear equality constraints
if nargout > 2
    Gc=... % gradients for inequality constraints
    Gceq=... % gradients for equality constraints
end

```

Example 6-1-9

$$\min_{\mathbf{x}} \max_i f_i(\mathbf{x})$$

where

$$f_1 = x_1^2 + x_2^4$$

$$f_2 = (2 - x_1)^2 + (2 - x_2)^2$$

$$f_3 = 2e^{x_2 - x_1}$$

Example 6-1-9

Ans:

ex6_1_9.m

```
function ex6_1_9
%
% Example 6-1-9 Minimax optimization
%
clear
clc
%
% initial guess values
%
x0=[2 2];
%
% solving the minimax optimization problem
%
[x, fval, fmax]=fminimax(@objfun, x0)
```

Ans:

ex6_1_9.m

```
%  
% objective functions  
%  
function f=objfun(x)  
x1=x(1);  
x2=x(2);  
f=[x1^2+x2^4  
    (2-x1)^2+(2-x2)^2  
    2*exp(x2-x1)];
```

Execution results:

```
>> ex6_1_9
```

x =

1.1390 0.8996

fval =

1.9522

1.9522

1.5741

fmax =

1.9522

6.1.9 Binary integer programming problem

$$\min_{\mathbf{x}} \mathbf{f}^T \mathbf{x}$$

s.t.

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x} \in \{0, 1\}$$

```
[x, fval]=bintprog(f, A, b, Aeq, beq, x0, options)
```



Example 6-1-10

$$\min_{\mathbf{x}} J = -9x_1 - 5x_2 - 6x_3 - 4x_4$$

s.t.

$$6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 9$$

$$x_3 + x_4 \leq 1$$

$$-x_2 + x_4 \leq 0$$

$$x_1 + x_2 + x_3 + x_4 = 2$$

$$x_i = 0 \text{ or } 1, i = 1, 2, 3, 4$$

Example 6-1-10

Ans:

$$\mathbf{f} = [-9 \ -5 \ -6 \ -4]^T$$

$$\mathbf{A} = \begin{bmatrix} 6 & 3 & 5 & 2 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{f} = [9 \ 1 \ 0 \ 0]^T$$

$$\mathbf{A}_{eq} = [1 \ 1 \ 1 \ 1]^T$$

Ans:

and

$$\mathbf{b}_{eq} = 2$$

```
>> f=[-9 -5 -6 -4]';
>> A=[6 3 5 2
      0 0 1 1
      -1 0 1 0
      0 -1 0 1];
>> b=[9 1 0 0]';
>> Aeq=[1 1 1 1];
>> beq=2;
>> [x, fval]=bintprog(f, A, b, Aeq, beq)
```

Optimization terminated.

Ans:

```
x =  
1  
1  
0  
0
```

```
fval =  
-14
```

6.1.10 A real-coded genetic algorithm for optimization

- ▶ 6.1.10.1 Fundamental principles of the real-coded genetic algorithm
 - Genetic algorithm (GA) is a kind of population-based stochastic approach that mimics the natural selection and survival of the fittest in the biological world.
 - Let $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_n]$ be a set of possible solutions
 - The feasible solution Ω_{θ} is expressed as follows:

$$\Omega_{\theta} \equiv \{ \boldsymbol{\theta} \in R^n \mid \theta_{i,\min} \leq \theta_i \leq \theta_{i,\max}, i=1, 2, \dots, n \}$$

6.1.10.1 Fundamental principles of the real-coded genetic algorithm

I. Reproduction

- the reproduction is an evolutionary process to determine which chromosome should be eliminated or reproduced in the population.
- Using the scheme, $p_r \times N$ chromosomes having worst objective function values will be discarded and at the same time other $r_p \times N$ chromosomes that have better objective functions are reproduced.

II. Crossover

- Crossover operation, which acts as a scheme to blend the parents' information for generating the offspring chromosomes, is well recognized as the most important and effective searching mechanism in RCGAs.
- if $obj(\theta_1) < obj(\theta_2)$

$$\theta_1 \leftarrow \theta_1 + r (\theta_1 - \theta_2)$$

$$\theta_2 \leftarrow \theta_2 + r (\theta_1 - \theta_2)$$

else

$$\theta_1 \leftarrow \theta_1 + r (\theta_2 - \theta_1)$$

$$\theta_2 \leftarrow \theta_2 + r (\theta_2 - \theta_1)$$

end

III. Mutation

- Mutation serves as a process to change the gene of a chromosome (solution) randomly in order to increase the population diversity and thus prevent the population from premature convergence to a suboptimal solution.

$$\theta \leftarrow \theta + s \times \Phi$$

- The operational procedure of the RCGA

Step 1: Set the number of chromosomes in the population N , and the relevant probability values p_r , p_c , and p_m . Meanwhile, the maximum number of generations G is also assigned.

Step 2: Randomly produce N chromosomes from the feasible domain Ω_θ .

Step 3: Evaluate the corresponding objective function value for each chromosome in the population.

III. Mutation

Step 4: The genetic evolution terminates once the computation process has reached the specified maximum number of generations or the objective function has attained an optimal value.

Step 5: Implement the operations of reproduction, crossover, and mutation in the genetic evolution. Note that if the chromosomes of the offspring produced exceed the feasible range Ω_θ , the chromosomes before the evolution are retained.

Step 6: Go back to Step 3.

6.1.10.2 Application of the real-coded genetic algorithm to solve optimization problems

$$\min_{\mathbf{x}} f(\mathbf{x})$$

s.t.

$$\mathbf{c}(\mathbf{x}) \leq 0$$

$$\mathbf{c}_{eq}(\mathbf{x}) = 0$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$$

```
[x , fval]=rcga('fun', A, b, Aeq, beq, xL, xU, 'nonlcon', param)
```

Param	Description
param(1)	Number of chromosomes in the population, N
param(2)	Reproduction probability, p_r
param(3)	Crossover probability threshold, p_c
param(4)	Mutation probability, p_m
param(5)	Maximum number of generations, G

Example 6-1-11

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

s.t.

$$0 \leq 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \leq 92$$

$$90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110$$

$$20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25$$

$$78 \leq x_1 \leq 102$$

$$33 \leq x_2 \leq 45$$

$$27 \leq x_3, x_4, x_5 \leq 45$$

MATLAB program design:

$N = 200, P_r = 0.2, P_c = 0.3, P_m = 0.7$ and $G = 500$.

ex6_1_11

```
function ex6_1_11
%
% Example 6-1-11 Optimization with the real-coded genetic algorithm
%
clear
clc
%
% the RCGA parameters
%
N=200; % Number of populations
Pr=0.2; % Reproduction probability
Pc=0.3; % Crossover probability threshold
Pm=0.7; % Mutation probability
G=500; % Maximum generation numbers
```

MATLAB program design:

— ex6_1_11 —

```
% the linear constraints  
%  
A=[ ];  
b=[ ];  
Aeq=[ ];  
beq=[ ];  
%  
% the upper and lower bounds of the decision variables  
%  
xL=[78 33 27 27 27]; % lower bound  
xU=[102 45 45 45 45]; % upper bound  
%  
% rearrange the RCGA parameters into param  
%  
param=[N Pr Pc Pm G]; % RCGA parameter vector  
%  
% solving with the real-coded genetic algorithm
```

MATLAB program design:

— ex6_1_11 —

```
%  
[x, f]=rcga(@fun, A, b, Aeq, beq, xL, xU, @nonlcon, param);  
[c, ceq]=nonlcon(x) % check the satisfaction of the constraints  
%  
% results printing  
%  
fprintf('\n The optimal solution is x_%i=%.\n3f\n',[1:length(x); x])  
fprintf('\n The minimum objective function = %.\n3f\n', f)  
%  
% Objective function  
%  
function f=fun(x)  
x1=x(1); x2=x(2); x3=x(3); x4=x(4); x5=x(5);  
f=5.3578547*x3^2+0.8356891*x1*x5+37.293239*x1-40792.141;  
%  
% nonlinear constraints  
%
```

MATLAB program design:

— ex6_1_11 —

```
function [c, ceq]=nonlcon(x)
x1=x(1); x2=x(2); x3=x(3); x4=x(4); x5=x(5);
f1=85.334407+0.0056858*x2*x5+0.00026*x1*x4-0.0022053*x3*x5;
f2=80.51249+0.0071317*x2*x5+0.0029955*x1*x2+0.0021813*x3^2;
f3=9.300961+0.0047026*x3*x5+0.0012547*x1*x3+0.0019085*x3*x4;
c=[-f1; f1-92; 90-f2; f2-110; 20-f3; f3-25];
ceq=[ ];
```

Execution results:

```
>> ex1_1_11
```

```
c =
-92.0000
      0
-10.4048
-9.5952
      0
-5.0000
```

```
ceq =
```

```
[]
```

The optimal solution is $x_1=78.000$
 The optimal solution is $x_2=33.000$
 The optimal solution is $x_3=27.071$
 The optimal solution is $x_4=45.000$
 The optimal solution is $x_5=44.969$

The minimum objective function = -31025.560

(Gen and Cheng, 1997), which is

$f^* = -30665.5$ at \mathbf{x}^*

$= [78 \ 33 \ 29.995 \ 45 \ 36.776]$.

Example 6-1-12

Resolve the problem in Example 6-1-5 with the real-coded genetic algorithm

MATLAB program design:

$$0 \leq x_i \leq 25, i = 1, 2, 3.$$

MATLAB program design:

— ex6_1_12.m —

```
function ex6_1_12
%
% Example 6-1-12 Optimization with RCGA
%
clear
clc
%
% the RCGA parameters
%
N=500; % Population number
Pr=0.2; % Reproduction probability
Pc=0.3; % Crossover probability threshold
Pm=0.7; % Mutation probability
G=1000; % Maximum generation number
%
% linear equality and inequality restraints
%
```

MATLAB program design:

— ex6_1_12.m —

```
A=[-1 -2 -2; 1 2 2];
b=[0 72]';
Aeq=[];
beq=[];
%
% the upper and lower bounds
%
xL=[0 0 0]; % lower bound
xU=[25 25 25]; % upper bound
%
% rearrange the RCGA parameters into param
%
param=[N Pr Pc Pm G]; % the RCGA parameters
%
% Solving with RCGA
%
[x, f]=rcga(@fun, A, b, Aeq, beq, xL, xU, @nonlcon, param);
```

MATLAB program design:

— ex6_1_12.m —

```
A*x'-b;
f=fun(x);
[c, ceq]=nonlcon(x);
%
% result printing
%
fprintf('\n The optimal solution is x_%i=%.\n', [1:length(x); x])
fprintf('\n The minimum objective function = %.\n', f)
%
% objective function
%
function f=fun(x)
x1=x(1);
x2=x(2);
x3=x(3);
f=-x1*x2^2*x3;
%
```

MATLAB program design:

— ex6_1_12.m —

```
% nonlinear constraints  
%  
function [c, ceq]=nonlcon(x)  
x1=x(1);  
x2=x(2);  
x3=x(3);  
c=x1*x3-0.5*x2^2-380;  
ceq=x2*x3-144;
```

Execution results:

```
>> ex6_1_12
```

The optimal solution is $x_1=20.000$

The optimal solution is $x_2=18.000$

The optimal solution is $x_3=8.000$

The minimum objective function = -51840.00

6.2 Chemical engineering examples

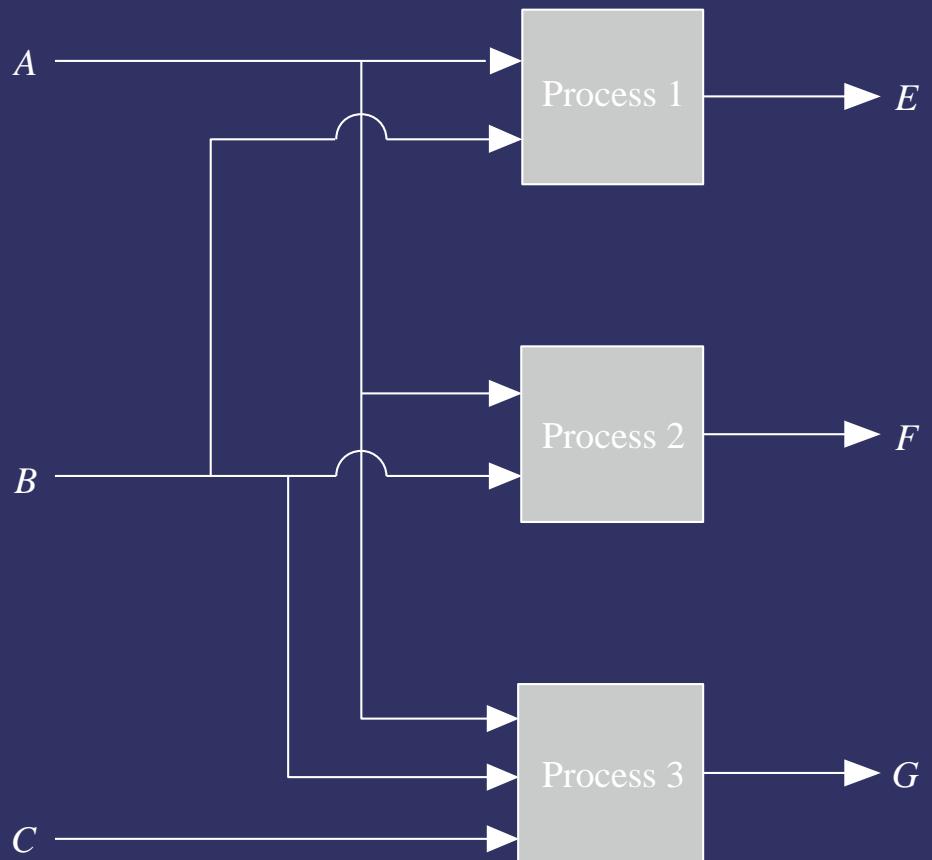
Example 6-2-1

Maximizing the profit of a production system

Process 1: $A+B \rightarrow E$

Process 2: $A+B \rightarrow F$

Process 3: $3A+2B+C \rightarrow G$



Raw material	Daily maximum supply (lb/day)	Unit price (\$/lb)
A	45,000	0.015
B	35,000	0.020
C	25,000	0.025

Process	Product	The raw materials required per pound of product (lb)	Operation costs	Price of product
1	E	$\frac{2}{3}A, \frac{1}{3}B$	0.015\$/lb E	0.040\$/lb E
2	F	$\frac{2}{3}A, \frac{1}{3}B$	0.005\$/lb F	0.033\$/lb F
3	G	$\frac{1}{2}A, \frac{1}{6}B, \frac{1}{3}C$	0.010\$/lb G	0.038\$/lb G

Determine the operating condition such that the daily profit is maximized. And how much is the maximum profit?

Problem formulation and analysis:

Let $x_1 \sim x_6$ denote the daily consumption of the raw materials A, B, and C and the production quantities of the products E, F, and G, respectively.

$$IC = 0.04x_4 + 0.033x_5 + 0.038x_6$$

Cost of raw materials: $C1 = 0.015x_1 + 0.02x_2 + 0.025x_3$

Operation cost: $C2 = 0.015x_4 + 0.005x_5 + 0.01x_6$

daily total cost: $C = C1 + C2$

$$F = IC - C$$

$$= 0.025x_4 + 0.028x_5 + 0.028x_6 - 0.015x_1 - 0.02x_2 - 0.025x_3$$

Problem formulation and analysis:

$$x_1 = 0.667x_4 + 0.667x_5 + 0.5x_6$$

$$x_2 = 0.333x_4 + 0.333x_5 + 0.167x_6$$

$$x_3 = 0.333x_6$$

$$0 \leq x_1 \leq 45,000$$

$$0 \leq x_2 \leq 35,000$$

$$0 \leq x_3 \leq 25,000$$

Problem formulation and analysis:

$$\mathbf{f} = [-0.015 \quad -0.02 \quad -0.025 \quad 0.025 \quad 0.028 \quad 0.028]^T$$

$$\mathbf{A}_{eq} = \begin{bmatrix} 1 & 0 & 0 & -0.667 & -0.667 & -0.5 \\ 0 & 1 & 0 & -0.333 & -0.333 & -0.167 \\ 0 & 0 & 1 & 0 & 0 & -0.333 \end{bmatrix}$$

$$\mathbf{b}_{eq} = [0 \quad 0 \quad 0]^T$$

$$\mathbf{x}_L = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\mathbf{x}_U = [45000 \quad 35000 \quad 25000 \quad \text{inf} \quad \text{inf} \quad \text{inf}]^T$$

MATLAB program design:

— ex6_2_1.m —

```
function ex6_2_1
%
% Example 6-2-1 Maximal profit of a production system
%
clear
clc
%
% Relevant parameters in a linear programming problem
%
n=6; % Number of variables
f=[ -0.015 -0.02 -0.025 0.025 0.028 0.028]';
Aeq=[1 0 0 -0.667 -0.667 -0.5
      0 1 0 -0.333 -0.333 -0.167
      0 0 1 0     0    -0.333];
beq=zeros(3,1);
xL=zeros(1,n);
xU=[45000 35000 25000 inf inf inf];
```

MATLAB program design:

— ex6_2_1.m —

```
A=[ ];  
b=[ ];  
%  
% Solve the linear programming problem with linprog  
%  
[x, fval]=linprog(-f, A, b, Aeq, beq, xL, xU); % Note that -f is used  
%  
% Results printing  
%  
for i=l: n  
    fprintf('\n The optimal solution x(%d) is %.3f', i, x(i))  
end  
fprintf('\n The maximum profits are =%.3f.\n', -fval) % NOTE: -fval
```

Execution results:

```
>> ex6_2_1
```

The optimal solution is $x(1)=45000.000$

The optimal solution is $x(2)=16263.175$

The optimal solution is $x(3)=25000.000$

The optimal solution is $x(4)=0.000$

The optimal solution is $x(5)=11188.100$

The optimal solution is $x(6)=75075.075$

The maximum profit is $f=790.105.$

Example 6-2-2

The optimal photoresist film thickness in a wafer production process

According to practical operation experiences, one can summarize the factors that affect the quality of printed circuits in a wafer production process as follows (*Edgar and Himmelblau, 1989*):

1. Inadequately lower photoresist film thickness can inevitably increase the defect ratio. The correlation between the defect ratio and the film thickness is expressed by

$$D_0 = 1.5 d^{-3}$$

where D_0 is the defect ratio and d is the film thickness (micrometers).

2. The percentage of perfect chips in the circuit layer can be estimated by

$$\eta = \frac{1}{(1+\beta D_0 a)^4}$$

where a is the activated area of the chips and β is the percentage of severe defects.

3. The increment of the photoresist film thickness tends to reduce the wafer production capacity; the relationship is given as follows:

$$V = 125 - 50d + 5d^2$$

Note that the unit of V is wafers/h. Besides, the following process information and statistical data are known: each wafer contains 100 chips, each chip has an activated area of 0.25 and the percentage of severe defects is 30% (i.e., $\beta = 0.3$). Based on the above, determine the optimal photoresist film thickness that maximizes the production amount of perfect chips.

NOTE:

the admissible photoresist film thickness lies in the range of $0.5 \leq d \leq 2.5$.

Problem formulation and analysis:

the hourly production number of perfect chips,

$$f = 100 V \eta = (125 - 50d + 5d^2) \frac{100}{(1 + \beta D_0 a)^4}$$

where $\beta = 0.3$, $a = 0.25$, and $0.5 \leq d \leq 2.5$.

Ans:

```
>> f=inline(' - 100*(125-50*d+5*d^2)/(1+0.3*0.25*1.5*d^(-3))^4');
```

```
>> [d, fval]=fminbnd(f, 0.5, 2.5)
```



```
d=
```



```
    1.2441
```



```
fval =
```



```
-5.6203e+003
```

Example 6-2-3

Minimal energy of a chemical equilibrium system

Find the mole fraction of each component such that the energy function

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i \left(w_i + \ln P + \ln \frac{x_i}{\sum_{i=1}^{10} x_i} \right)$$

is minimized subject to following equality constraints derived from material balances

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} = 2$$

$$x_4 + 2x_5 + x_6 + x_7 = 1$$

$$x_3 + x_7 + x_8 + 2x_9 + x_{10} = 1$$

The values of the coefficient in Equation (6.2-15) are listed in the following table:

i	1	2	3	4	5
w_i	-10.021	-21.096	-37.986	-9.846	-28.653
i	6	7	8	9	10
w_i	-18.918	-28.032	-14.640	-30.594	-26.111

Note that the system is operated under the total pressure of $P = 760 \text{ mmHg}$.

MATLAB program design:

Because the mole fraction x_i must satisfy $0 \leq x_i \leq 1$, these additional constraints should be added into the optimization problem.

— ex6_2_3.m —

```
function ex6_2_3
%
% Example 6-2-3 Energy minimization of a chemical equilibrium system
%
clear
clc
%
global w P
%
% System data: w values
%
w=[-10.021 -21.096 -37.986 -9.846 -28.653 -18.918 -28.032 -14.640... -30.594 -
26.111]; %
```

MATLAB program design:

— ex6_2_3.m —

```
P=760; % Pressure (mmHg)
%
% Initial guess values
%
x0=0.5*ones(1, 10);
%
% Constraints
%
Aeq=[1 2 2 0 0 1 0 0 0 1
      0 0 0 1 2 1 1 0 0 0
      0 0 1 0 0 0 1 1 2 1];
beq=[2 1 1]';
xL=eps*ones(1,10); % Lower bounds of the decision variables
xU=ones(1,10); % Upper bounds of the decision variables
%
% Solving the optimization problem with fmincon
%
```

MATLAB program design:

— ex6_2_3.m —

```
[x, fval]=fmincon(@fun, x0, [], [], Aeq, beq, xL, xU);  
%  
% Results printing  
%  
disp('The mole fraction of each component is')  
x  
disp('The minimal energy function value is')  
fval  
%  
% Objective function  
%  
function f=fun(x)  
global w P  
%  
f=sum(x.*(w+log(P)+log(x./sum(x))));
```

Execution results:

```
>> ex6_2_3
```

The mole fraction of each component is

x =

Columns 1 through 6

0.0070 0.0678 0.9076 0.0004 0.4909 0.0005

Columns 7 through 10

0.0173 0.0029 0.0152 0.0418

The minimal energy function value is

fval =

-43.4740

Execution results:

```
>> ex6_2_3
```

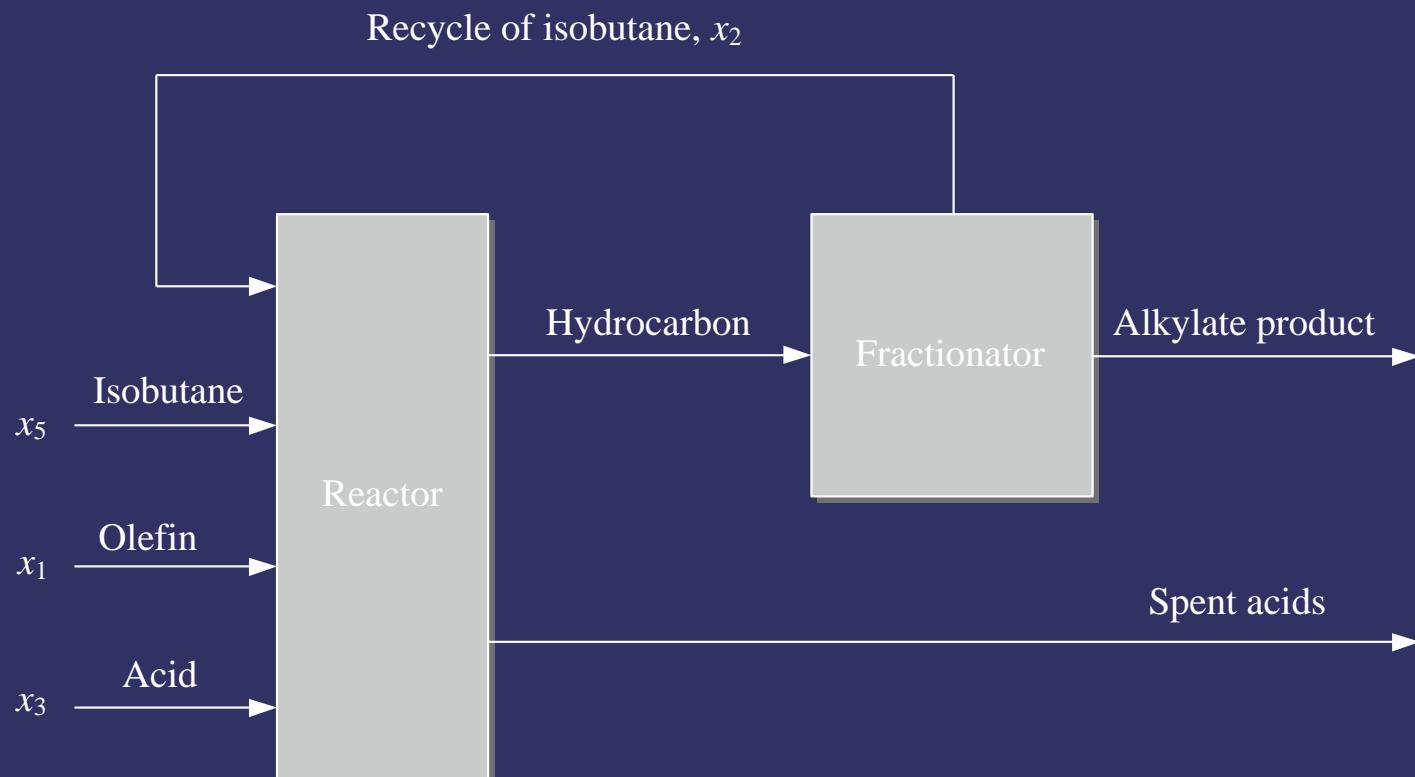
NOTE:

In the MATLAB program, the lower bound 0 is replaced by eps (2.2204×10^{-16}) to avoid the numerical problem arising from the calculation of natural logarithm for the objective function.

Example 6-2-4

Maximal profit of an alkylation process

Consider the alkylation process shown in the following figure (*Edgar and Himmelblau, 1989*).



Variables	Physical significance	Unit	Lower bound	Upper bound
x_1	Olefin feed	Barrel/day	0	2,000
x_2	Isobutane recycled	Barrel/day	0	16,000
x_3	Feed of fresh acid	Thousands of pounds per day	0	120
x_4	Alkylate yield	Barrel/day	0	5,000
x_5	Isobutane makeup	Barrel/day	0	2,000
x_6	Acid strength	Weight percentage	85	93
x_7	Motor octane number	-	90	95
x_8	External isobutene-to-olefin ratio	-	3	12
x_9	Acid dilution factor	-	1.2	4
x_{10}	F-4 performance number	-	145	162

The objective function,

$$f = c_1 x_4 x_7 - c_2 x_1 - c_3 x_2 - c_4 x_3 - c_5 x_5$$

Parameter	Physical meaning	Value	Unit
c_1	Alkylate product value	0.063	\$/barrel
c_2	Olefin feed cost	5.04	\$/barrel
c_3	Isobutane recycle cost	0.035	\$/barrel
c_4	Acid addition cost	10	\$/thousand lbs
c_5	Isobutane makeup cost	3.36	\$/barrel



$$x_4 = x_1(1.12 + 0.13167x_8 - 0.00667x_8^2)$$

$$1.22x_4 - x_1 - x_5 = 0$$

$$x_6 = \frac{98,000x_3}{x_4x_9 + 1,000x_3}$$

$$x_7 = 86.35 + 1.098x_8 - 0.038x_8^2 + 0.325(x_6 - 89)$$

$$x_8 = \frac{x_2 + x_5}{x_1}$$

$$x_9 = 35.82 - 0.222x_{10}$$



$$x_{10} = -133 + 3x_7$$

To include the process uncertainties in the formulation,

$$-x_1(1.12 + 0.13167x_8 - 0.00667x_8^2) + d_4^- x_4 \leq 0$$

$$x_1(1.12 + 0.13167x_8 - 0.00667x_8^2) - d_4^+ x_4 \leq 0$$

$$-[86.35 + 1.098x_8 - 0.038x_8^2 + 0.325(x_6 - 89)] + d_7^- x_7 \leq 0$$

$$86.35 + 1.098x_8 - 0.038x_8^2 + 0.325(x_6 - 89) - d_7^+ x_7 \leq 0$$

$$-(35.82 - 0.222x_{10}) + d_9^- x_9 \leq 0$$

$$-(35.82 - 0.222x_{10}) + d_9^- x_9 \leq 0$$

$$35.82 - 0.222x_{10} - d_9^+ x_9 \leq 0$$

$$133 - 3x_7 + d_{10}^- x_{10} \leq 0$$

$$-133 + 3x_7 - d_{10}^+ x_{10} \leq 0$$

where the values of d_i^- and d_i^+ are listed in the following table:

d_4^-	d_4^+	d_7^-	d_7^+	d_9^-	d_9^+	d_{10}^-	d_{10}^+
$\frac{95}{100}$	$\frac{100}{95}$	$\frac{95}{100}$	$\frac{100}{95}$	$\frac{95}{100}$	$\frac{100}{95}$	$\frac{95}{100}$	$\frac{100}{95}$

MATLAB program design:

— ex6_2_4.m —

```
function ex6_2_4
%
% Example 6-2-4 Maximum profit in an alkylation process
%
clear
clc
%
% Upper and lower bounds of the decision variables
%
xL=[0 0 0 0 0 85 90 3 1.2 145]';
xU=[2000 16000 120 5000 2000 93 95 12 4 162]';
%
% Initial guess values
%
x0=[1745 12000 110 3048 1974 89.2 92.8 8 3.6 145]';
%
% Linear constraints
```

MATLAB program design:

— ex6_2_4.m —

```
%  
Aeq=[-1 0 0 1.22 -1 0 0 0 0 0]; % Eq. (6.2-20)  
beq=0;  
%  
A=[0 0 0 0 0 0 0 95/100 0.222      % Eq. (6.2-30)  
  0 0 0 0 0 0 0 -100/95 -0.222      % Eq. (6.2-31)  
  0 0 0 0 0 0 -3 0 0 95/100        % Eq. (6.2-32)  
  0 0 0 0 0 0 3 0 0 -100/95];       % Eq. (6.2-33)  
b=[35.82 -35.82 -133 133]';  
%  
% Solving with fmincon  
%  
[x, f]=fmincon(@fun, x0, A, b, Aeq, beq, xL, xU, @nonlcon);  
%  
% Results printing  
%  
disp('The optimal solution is:')
```

MATLAB program design:

— ex6_2_4.m —

```
fprintf('\n The maximum profit is %.3f\n.', -f)
%
% Objective function
%
function f=fun(x)
x1=x(1);x2=x(2);x3=x(3);x4=x(4);x5=x(5);
x6=x(6);x7=x(7);x8=x(8);x9=x(9);x10=x(10);
f=0.063*x4*x7-5.04*x1-0.035*x2-10*x3-3.36*x5;
f=-f;
%
% Nonlinear constraints
%
function [c, ceq]=nonlcon(x)
x1=x(1);x2=x(2);x3=x(3);x4=x(4);x5=x(5);
x6=x(6);x7=x(7);x8=x(8);x9=x(9);x10=x(10);
c=[-x1*(1.12+0.13167*x8-0.00667*x8^2)+95/100*x4 % (6.2-26)
```

MATLAB program design:

— ex6_2_4.m —

```
x1*(1.12+0.13167*x8-0.00667*x8^2)-100/95*x4 % Eq. (6.2-27)
```

```
-(86.35+1.098*x8-0.038*x8^2+0.325*(x6-89))+95/100*x7 % (6.2-28)
```

```
86.35+1.098*x8-0.038*x8^2+0.325*(x6-89)-100/95*x7]; % Eq. (6.2-29)
```

```
ceq=[x6*(x4*x9+1000*x3)-98000*x3 % Eq. (6.2-21)
```

```
 x8*x1-x2-x5]; % Eq. (6.2-23)
```

Execution results:

```
>> ex6_2_4
```

The optimal solution is:

```
x =  
1.0e+003 *  
1.7506  
8.4495  
0.0241  
3.0743  
2.0000  
0.0850  
0.0950  
0.0060  
0.0012  
0.1562
```

The maximum profit is 2319.484.

Example 6-2-5

Maximum separation efficiency in a single-effect distillatory

Consider a two-component single-effect distillation tower operated at one atmospheric pressure. The relationship between the liquid-phase mole fraction x and the vapor-phase mole fraction y of the light key component is shown in the following table (Leu, 1985):

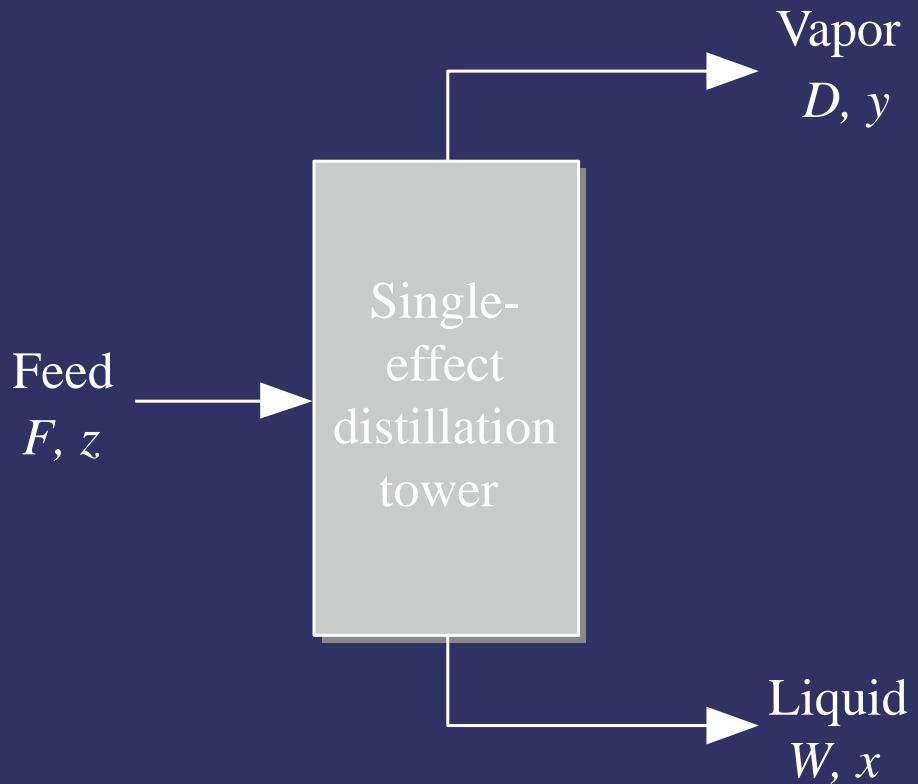
x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	0	0.191	0.339	0.480	0.618	0.730	0.812	0.864	0.913	0.959	1.0

Let z and F denote, respectively, the mole fraction of the light key component in the feed and the feeding rate. Then, the separation efficiency of the single-effect distillation tower can be defined as

$$\eta \equiv \frac{D}{F} \frac{y - z}{z(1 - z)} = d \frac{y - z}{z(1 - z)}$$

where $d = D/F$ is the ratio of the flow rate of distillate to the feeding rate. Based on the above-mentioned operating conditions, determine the values of d and x such that the separation efficiency is maximized under each of the feed condition of z ranging from 0.1 to 0.9 with an interval of 0.1.

Problem formulation and analysis:



Problem formulation and analysis:

$$F = D + W$$

$$Fz = Dy + Wx$$

Let $w = W/F$,

$$d = w + 1$$

$$z = dy + wx$$

$$y = f(x)$$

MATLAB program design:

— ex6_2_5.m —

```
function ex6_2_5
%
% Example 6-2-5 Maximum separation efficiency of a single-effect distillatory
%
clear; close all;
clc
%
global z xe ye
%
% equilibrium data
%
xe=0:0.1:1;
ye=[0 0.191 0.339 0.480 0.618 0.730 0.812 0.864 0.913 0.959 1.0];
%
% initial guess values
%
p0=[0.5 0.5];
```

MATLAB program design:

— ex6_2_5.m —

```
function ex6_2_5
%
% Example 6-2-5 Maximum separation efficiency of a single-effect distillatory
%
clear; close all;
clc
%
global z xe ye
%
% equilibrium data
%
xe=0:0.1:1;
ye=[0 0.191 0.339 0.480 0.618 0.730 0.812 0.864 0.913 0.959 1.0];
%
% initial guess values
%
p0=[0.5 0.5];
```

MATLAB program design:

— ex6_2_5.m —

```
%  
% Upper and lower bounds of the decision variables  
%  
xL=[0 0];  
xU=[1 1];  
%  
% finding the optimal solution to each case of z values  
%  
for i=1:9  
    z=0.1*i;  
    [p, f]=fmincon(@fun, p0, [], [], [], [], xL, xU, @nonlcon);  
    dopt(i)=p(1);  
    xopt(i)=p(2);  
    eta(i)=-f;  
end  
%  
% Results printing
```

MATLAB program design:

— ex6_2_5.m —

```
%  
for i=1:9  
    z=0.1*i;  
    fprintf('\n When z=% .1f, d=% .3f, x=% .3f, the optimal efficiency is =...  
% .3f.', z, dopt(i), xopt(i), eta(i))  
end  
%  
% Plot a figure to show the variations of the optimal results  
%  
z=0.1:0.1:0.9;  
plot(z, eta, z, eta, '*')  
title('Variations of the optimial results')  
xlabel('Mole fraction of the light key component in the feeding stream, z')  
ylabel('Maximum separation efficiency')  
%  
% Objective function  
%
```

MATLAB program design:

— ex6_2_5.m —

```
function f=fun(p)
global z xe ye
d=p(1);
x=p(2);
y=interp1(xe, ye, x, 'cubic');
f=-d*(y-z)/(z*(1-z));
%
% Constraints
%
function [c, ceq]=nonlcon(p)
global z xe ye
d=p(1);
x=p(2);
w=1-d;
y=interp1(xe, ye, x, 'cubic');
c=[ ];
ceq=z-d*y-w*x;
```

Execution results:

>> ex6_2_5

When $z=0.1$, $d=0.426$, $x=0.071$, the optimal efficiency is = 0.188.

When $z=0.2$, $d=0.454$, $x=0.147$, the optimal efficiency is = 0.180.

When $z=0.3$, $d=0.457$, $x=0.231$, the optimal efficiency is = 0.180.

When $z=0.4$, $d=0.456$, $x=0.315$, the optimal efficiency is = 0.192.

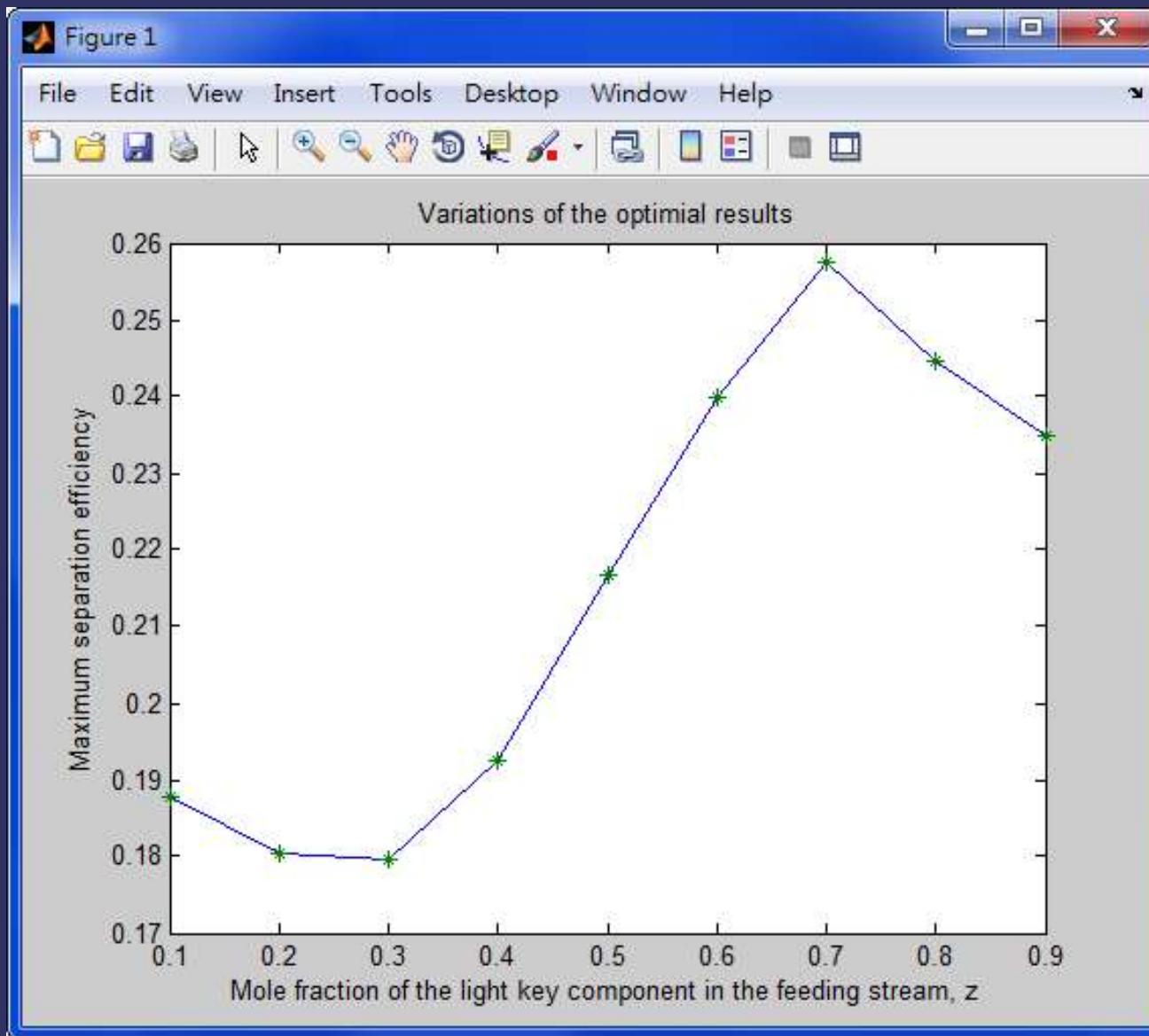
When $z=0.5$, $d=0.472$, $x=0.397$, the optimal efficiency is = 0.217.

When $z=0.6$, $d=0.499$, $x=0.485$, the optimal efficiency is = 0.240.

When $z=0.7$, $d=0.541$, $x=0.582$, the optimal efficiency is = 0.257.

When $z=0.8$, $d=0.584$, $x=0.706$, the optimal efficiency is = 0.245.

When $z=0.9$, $d=0.595$, $x=0.848$, the optimal efficiency is = 0.235.



Example 6-2-6

Dynamic optimization of an ammonia synthesis process

Haber-Bosch synthesis process



$$\begin{aligned} \max_L f = & 1.198 \times 10^7 - 1.710 \times 10^4 N_{N_2} |_L + 704.4 T_g |_L - 699.3 T_f |_L \\ & - (3.456 \times 10^7 + 2.101 \times 10^9 L)^{0.5} \end{aligned}$$

s.t.

$$\frac{dT_f}{dx} = -\frac{US_1}{WC_{pf}}(T_g - T_f)$$

$$\frac{dT_g}{dx} = -\frac{US_1}{WC_{pg}}(T_g - T_f) + \frac{(-\Delta H)S_2}{WC_{pg}} r \left(\frac{1.5k_1 P_{N_2} P_{H_2}}{P_{NH_3}} - \frac{k_2 P_{NH_3}}{1.5 P_{H_2}} \right)$$



$$\frac{dN_{N_2}}{dx} = -r \left(\frac{1.5k_1 P_{N_2} P_{H_2}}{P_{NH_3}} - \frac{k_2 P_{NH_3}}{1.5 P_{H_2}} \right)$$

$$T_f \Big|_{x=L} = 693 \text{ K}$$

$$T_g \Big|_{x=0} = 693 \text{ K}$$

$$N_{N_2} \Big|_{x=0} = 701.2 \text{ kg} \cdot \text{mol}/(\text{m}^2 \cdot \text{h})$$

$$0 \leq x \leq L$$



$$P_{\text{N}_2} = P_T \frac{N_{\text{N}_2}}{N - 2B}$$

$$P_{\text{H}_2} = P_T \frac{3N_{\text{N}_2}}{N - 2B}$$

$$P_{\text{NH}_3} = P_T \frac{A}{N - 2B}$$

where

$$B = N_{\text{N}_2}^0 - N_{\text{N}_2}$$

$$N = \frac{100N_{\text{N}_2}^0}{21.75}$$

$$A = \frac{5N}{100} + 2B$$



$$k_1 = 1.7895 \times 10^4 \exp\left(-\frac{20,800}{RT_g}\right)$$

$$k_2 = 2.5714 \times 10^{16} \exp\left(-\frac{47,400}{RT_g}\right)$$

Parameter	Physical meaning	Value	Unit
C_{pf}	Heat capacity of the feed	0.707	kcal/(kg·K)
C_{pg}	Heat capacity of the reaction gas	0.719	kcal/(kg·K)
r	Activity of the catalyst	0.95	-
ΔH	Heat of reaction	-26,600	kcal / (kg-mol N ₂)
$N_{N_2}^0$	The molar flow rate of N ₂ in the feed	701.2	kg-mol/(m ² ·hr)
P_T	Total pressure	268	psi
R	Ideal gas constant	1.987	kcal/(kg-mol·K)
S_1	Heat transfer area per unit of reactor length	10	m
S_2	Cross-sectional area of the catalyst section	0.78	m ²
U	Overall heat conduction coefficient	500	kcal/(hr·m ² · K)
W	Overall mass transport rate	26,400	kg/hr

Solve the dynamic optimization problem under the above-mentioned operating condition.

Analysis of the question:

$$\frac{dT_f}{dz} = L \left[-\frac{US_1}{WC_{pf}} (T_g - T_f) \right]$$

$$\frac{dT_g}{dz} = L \left[\frac{-US_1}{WC_{pg}} (T_g - T_f) + \frac{(-\Delta H)S_2}{WC_{pg}} r \left(\frac{1.5k_1 P_{N_2} P_{H_2}}{P_{NH_3}} - \frac{k_2 P_{NH_3}}{1.5 P_{H_2}} \right) \right]$$

$$\frac{dN_{N_2}}{dz} = -rL \left(\frac{1.5k_1 P_{N_2} P_{H_2}}{P_{NH_3}} - \frac{k_2 P_{NH_3}}{1.5 P_{H_2}} \right)$$

$$T_f \Big|_{z=1} = 694$$

$$T_g \Big|_{z=0} = 694$$

Analysis of the question:

$$N_{\text{N}_2} \Big|_{z=0} = 701.2$$

$$0 \leq z \leq L$$

where $z = x/L$ is the defined dimensionless variable and the decision variable L (the reactor length) is in the range of $0 \leq L \leq 20$.

MATLAB program design:

— ex6_2_6.m —

```
function ex6_2_6
%
% Example 6-2-6 Optimization of an ammonia synthesis process
%
clear; close all;
clc
%
global PT U S1 S2 W Cpf Cpg dH R r NN20 L
%
% Model parameters
%
PT=286; % Total pressure (psi)
Cpf=0.707; % Heat capacity of the feed, kcal/(kg.K)
Cpg=0.719; % Heat capacity of the reaction gas, kcal/(kg.K)
r=0.95; % Catalyst activity
dH=-26600; % Heat of reaction, kcal/(kg-mol N2)
R=1.987; % Ideal gas constant, kcal/(kg-mol.K)
```

MATLAB program design:

— ex6_2_6.m —

```
S1=10; % Heat transfer area per unit of reactor length, m  
S2=0.78; % cross-sectional area of the catalyst section, m^2  
U=500; % Overall heat conduction coefficient, kcal/(h.m^2.K)  
W=26400; % Overall mass transport rate, kg/h  
NN20=701.2; % Mole flow rate of ammonia in the feed, kg-mol/(m^2.h)  
%  
% Upper and lower bounds of the reactor length (the decision variable)  
%  
pL=1;  
pU=20;  
%  
% Solve the optimization problem with fminbnd  
%  
[L, fval]=fminbnd(@objfun, pL, pU);  
%  
% Results printing  
%
```

MATLAB program design:

— ex6_2_6.m —

```
fprintf('\n The optimal reactor length is %.3f m, and the maximum profit is...
%.3e.', L, -fval)
%
% Results plotting
%
solinit=bvpinit(linspace(0,1,20), @init);
sol=bvp4c(@ode, @bc, solinit);
xint=linspace(0,1);
yint=deval(sol, xint);
plot(xint, yint(1,:), 'r', xint, yint(2,:), '-.b', xint, yint(3,:), '--c')
legend('Tf', 'Tg', 'NN2')
xlabel('The dimensionless reactor length (z)')
ylabel('System states')
axis([0 1 670 735])
%
% objective function
%
```

MATLAB program design:

— ex6_2_6.m —

```
function f=objfun(p)
global PT U S1 S2 W Cpf Cpg dH R r NN20 L
L=p;
%
% Solve for the BVP ODE model
%
xint=linspace(0,1,20);
solinit=bvpinit(xint, @init);
sol=bvp4c(@ode, @bc, solinit);
yint=deval(sol, xint);
index=length(xint);
Tf=yint(1,index);
Tg=yint(2,index);
NN2=yint(3,index);
%
% Calculate the objective function value
%
```

MATLAB program design:

— ex6_2_6.m —

```
f=1.198e7-1.710e4*NN2+704.4*Tg-699.3*Tf-(3.456e7+2.101e9*L)^0.5;
f=-f;
%
% The ODE model
%
function dydt=ode(t, y)
global PT U S1 S2 W Cpf Cpg dH R r NN20 L
Tf=y(1);
Tg=y(2);
NN2=y(3);
k1=1.7895e4*exp(-20800/(R*Tg));
k2=2.5714e16*exp(-47400/(R*Tg));
B=NN20-NN2;
N=100*NN20/21.75;
A=5/100*N+2*B;
PN2=PT*NN2/(N-2*B);
PH2=PT*(3*NN2)/(N-2*B);
```

MATLAB program design:

— ex6_2_6.m —

```
PNH3=PT*A/(N-2*B);
Q=r*(1.5*k1*PN2*PH2/PNH3-k2*PNH3/(1.5*PH2));
dydt=L*[-U*S1*(Tg-Tf)/(W*Cpf)
-U*S1*(Tg-Tf)/(W*Cpg)+(-dH)*S2*Q/(W*Cpg)
-Q];
%
% Boundary conditions
%
function res=bc(ya, yb)
res=[yb(1)-693
ya(2)-693
ya(3)-701.2];
%
% Initial guess values
%
function yinit=init(x)
yinit=[693
```

MATLAB program design:

— ex6_2_6.m —

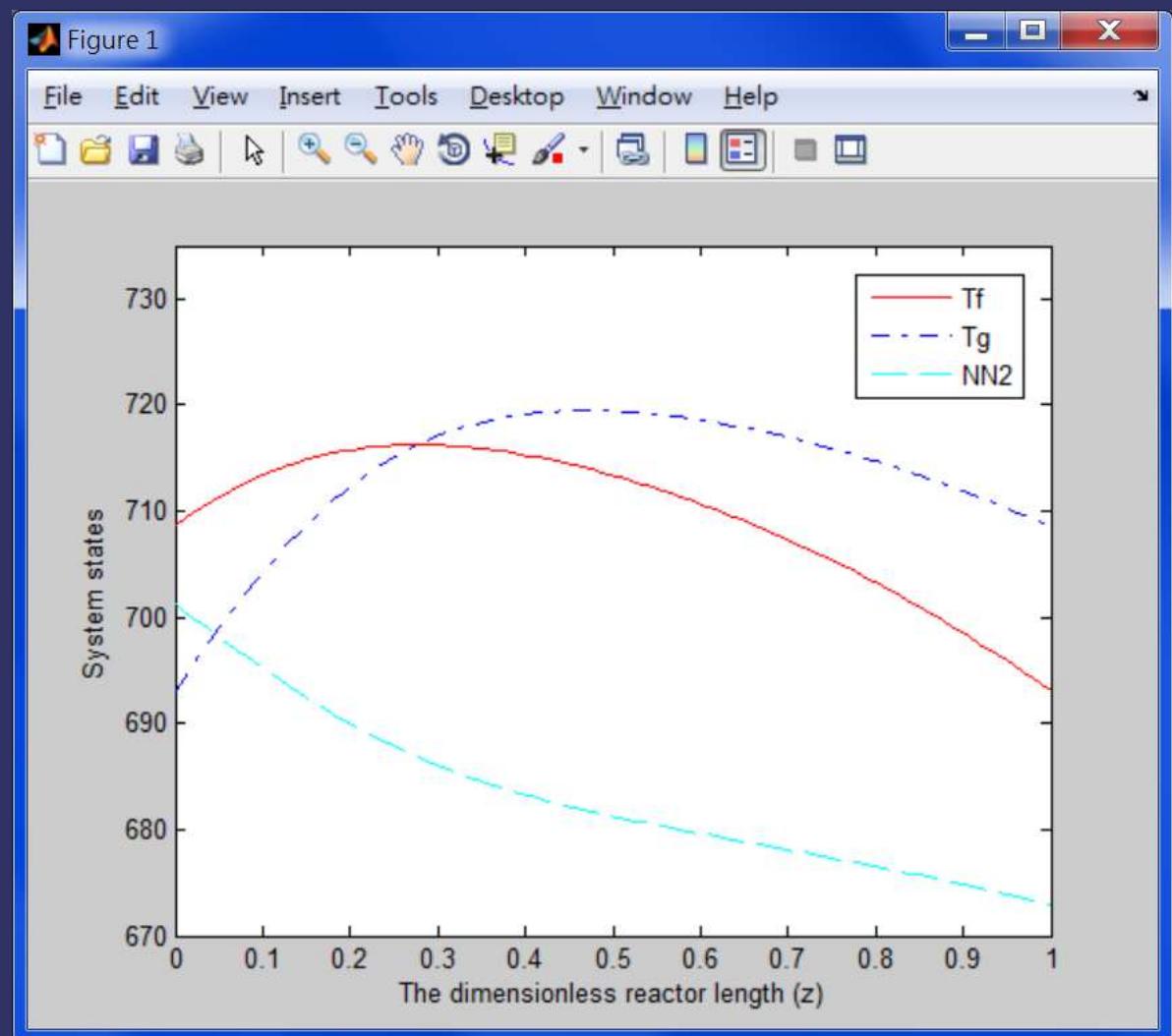
693

701.2];

Execution results:

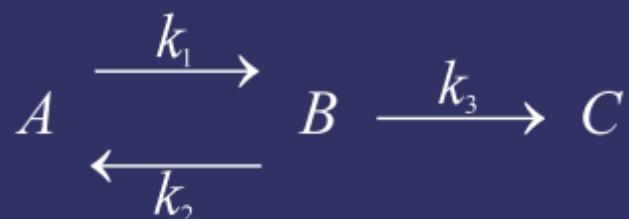
```
>> ex6_2_6
```

The optimal reactor length is 14.313 m, and the maximum profit is 3.125e+005.



Example 6-2-7

Optimal operating temperature for a tubular reactor



$$\frac{dx_1}{dt} = -k_1 x_1^2 + k_2 x_2, \quad x_1(0) = 0.95$$

$$\frac{dx_2}{dt} = k_1 x_1^2 - (k_2 + k_3) x_2, \quad x_2(0) = 0.05$$

$$k_i = k_{i0} \exp\left(\frac{-E_i}{RT}\right), \quad i = 1, 2, 3$$

I	$k_i 0$	E_i
1	5.35×10^5	1.82×10^4
2	4.61×10^9	3.16×10^4
3	2.24×10^8	2.87×10^4

Suppose the reaction time is 10 hours, determine the optimal profile of the operating temperature such that the concentration of the product B (x_2) is maximized at the end of the reaction.

Problem formulation and analysis:

$$\min_T -x_2 \Big|_{\tau=1}$$

s.t.

$$\frac{dx_1}{d\tau} = t_f (-k_1 x_1^2 + k_2 x_2), \quad x_1(0) = 0.95$$

$$\frac{dx_2}{d\tau} = t_f (k_1 x_1^2 - (k_2 + k_3)x_2), \quad x_2(0) = 0.05$$

where $\tau = t/t_f$ ($0 \leq \tau \leq 1$) is the dimensionless time and the reaction time is $t_f = 10$ hr.

MATLAB program design:

— ex6_2_7.m —

```
function ex6_2_7
%
% Example 6-2-7 Optimal operating temperature in a tubular reactor
%
clear; close all;
clc
%
global x0 tspan k0 ER tf dt T
%
% System operation data
%
x0=[0.95  0.05]'; % Initial state values
k0=[5.35e5  4.61e9  2.24e8]; % ko value
E=[1.82e4  3.16e4  2.87e4]; % E value
R=1.987; % Ideal gas constant (kcal/kg-mol.K)
ER=E/R; % E/R value
tf=10; % reaction time (hr)
```

MATLAB program design:

— ex6_2_7.m —

```
%  
n=10; % number of sections (intervals) N=n+1;  
N=n+1; % % number of points  
dt=1/n; % time interval  
tspan=linspace(0, 1, N); % dimensionless time  
%  
% Upper and lower bounds and the initial guess values  
%  
pL=[273*ones(1,N)]/100; % Lower bounds  
pU=[873*ones(1,N)]/100; % Upper bounds  
p0=(pL+pU)/2; % Initial guess value  
%  
% Solve the dynamic optimization problem  
%  
[P, fval]=fmincon(@objfun, p0, [ ], [ ], [ ], [ ], pL, pU);  
%  
% Results printing
```

MATLAB program design:

— ex6_2_7.m —

```
%  
T=P*100;  
fprintf('\n At the end of the reaction, the maximum value of x2 is %.3f.\n', -fval)  
%  
% Numerical solution of the ODE  
%  
[s, x]=ode45(@ode, tspan, x0);  
%  
% Plot of the system states  
%  
figure(1)  
plot(s,x(:,1), 'r', s, x(:,2), '-.b')  
xlabel('Dimensionless time, t/tf')  
ylabel('System states')  
legend('x1', 'x2')  
%  
% Optimal temperature distribution
```

MATLAB program design:

— ex6_2_7.m —

```
%  
figure(2)  
stairs(s, T)  
xlabel('Dimensionless time, t/tf')  
ylabel('Temperature (K)')  
title('Optimal temperature distribution')  
%  
% Objective function  
%  
function f=objfun(P)  
global x0 tspan k0 ER tf dt T  
%  
T=P*100;  
%  
% Solving the ODE  
%  
[s, x]=ode45(@ode, tspan, x0);
```

MATLAB program design:

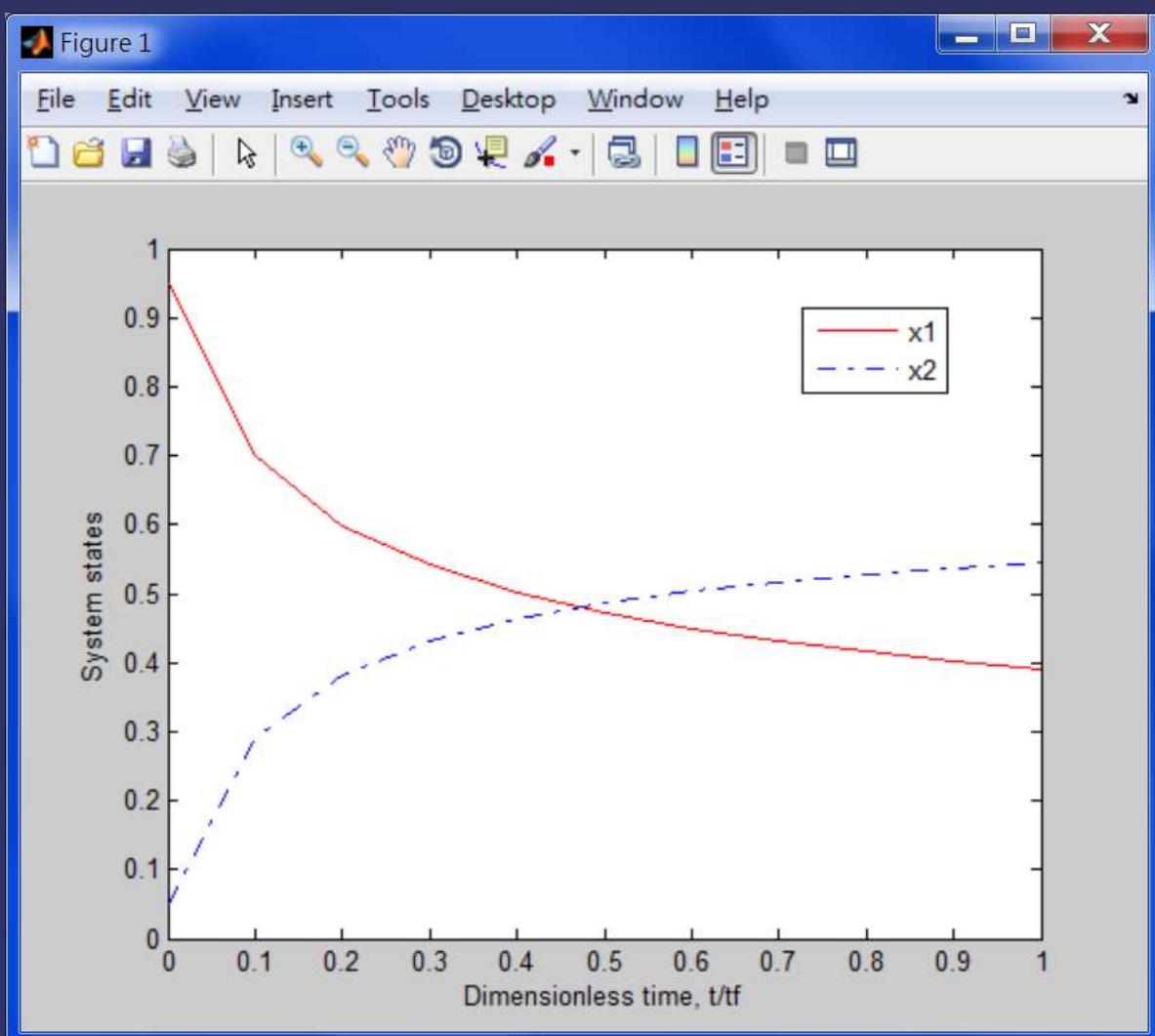
— ex6_2_7.m —

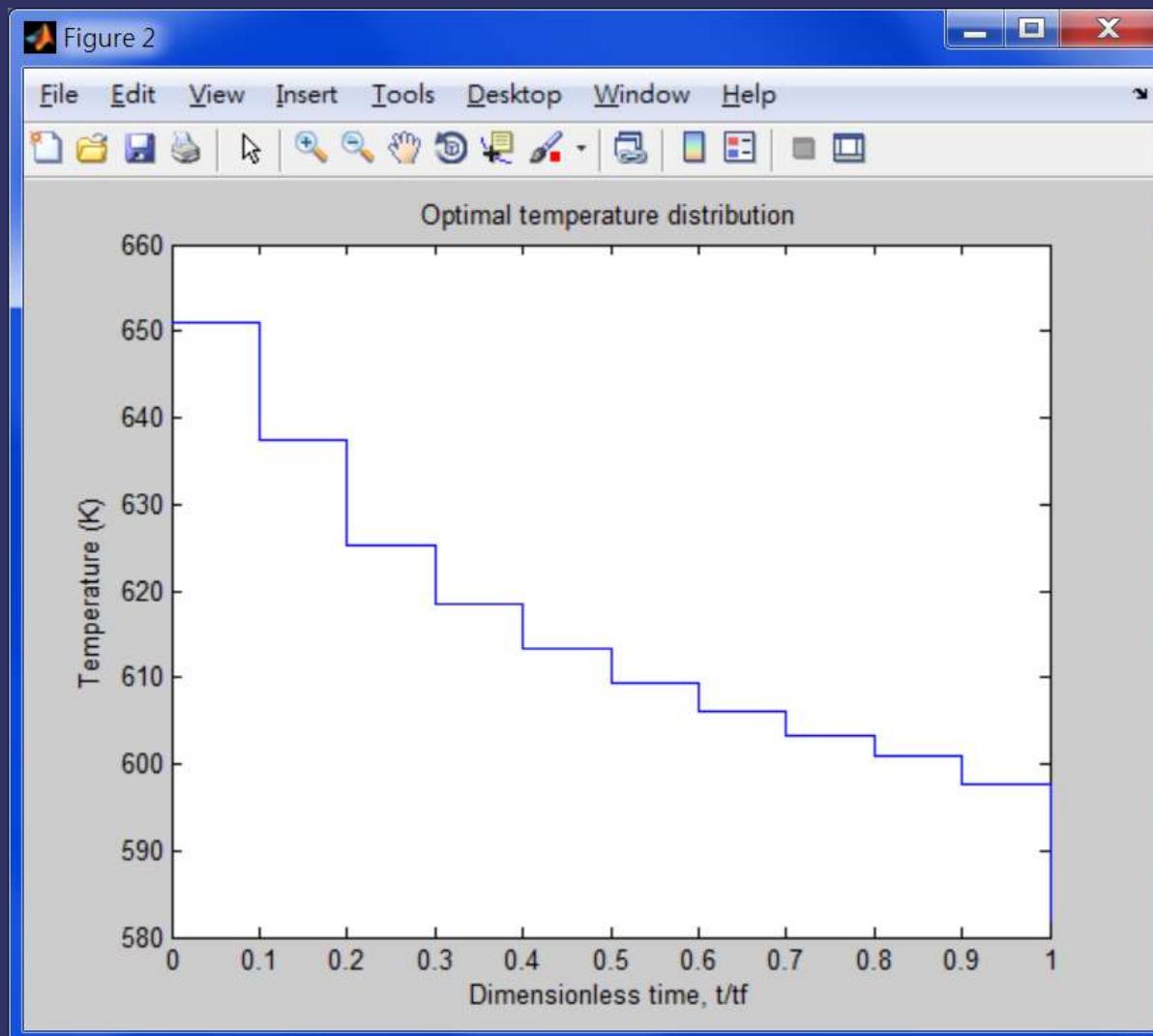
```
%  
% Calculating the objective function value  
%  
f=x(end, 2);  
f=-f;  
%  
% Kinetic equations  
%  
function dxdttau=ode(tau, x)  
global x0 tspan k0 ER tf dt T  
i=floor(tau/dt)+1;  
Ti=T(i);  
k=k0.*exp(-ER/Ti);  
k1=k(1);k2=k(2);k3=k(3);  
dxdttau=tf*[-k1*x(1)^2+k2*x(2)  
k1*x(1)^2-(k2+k3)*x(2)];
```

Execution results:

```
>> ex6_2_7
```

At the end of the reaction, the maximum value of x_2 is 0.545.





6.4 Summary of the MATLAB commands related to this chapter

Command	Function
fminbnd	The solver for the solution of univariate optimization problems
fminunc	The solver for multivariate optimization problems without constraints
fminsearch	The solver using Simplex method for the multivariate optimal problem without constraints
linprog	The solver for linear programming problems
quadprog	The solver for quadratic programming problems
fmincon	The solver for nonlinear optimization problems
fgoalattain	The solver for multi-objective goal attainment optimization problems
fseminf	The solver for semi-infinite optimization problems
fminimax	The solver for minimax problems
bintprog	The solver for binary integer programming problems
rcga	A real-coded genetic algorithm for process optimization



Chyi-Tsong Chen

Applications in Chemical Engineering



Chapter 7

Parameter Estimation

版權所有・請勿翻製

- ① the relevant MATLAB built-in commands for parameter estimation are introduced,
- ② the presentation of the concepts of least-squares and the 95% confidence interval of the parameter estimates.
- ③ a variety of parameter-estimation applications in the field of chemical engineering have been performed.

7.1 Parameter estimation using the least-squares method

- Let $(x_i, y_i), i = 1, 2, \dots, n$ be a set of experimental data,
- the process model $y = f(\mathbf{p}, x)$

$$J = \sum_{i=1}^n e_i^2$$

- is minimized, □

$$e_i = y_i - y_i^M, \quad i = 1, 2, \dots, n$$

7.1.1 Linear least-squares method

$$Y^M = a + bx$$

$$an + b \sum_{i=1}^n x_i = \sum_{i=1}^n y_i$$

$$J(a, b) = \sum_{i=1}^n (y_i - a - bx_i)^2$$

$$a \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i$$

$$\frac{\partial J}{\partial a} = 0 = -2 \sum_{j=1}^n (y_j - a - bx_j)$$

$$b = \frac{\sum_{i=1}^n x_i y_i - n \bar{x} \bar{y}}{\sum_{i=1}^n x_i^2 - n \bar{x}^2}$$

$$\frac{\partial J}{\partial b} = 0 = -2 \sum_{j=1}^n x_j (y_j - a - bx_j)$$

$$a = \bar{y} - b \bar{x}$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

$$\mathbf{Y}^M = \begin{bmatrix} y_1^M & y_2^M & \cdots & y_n^M \end{bmatrix}^T$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

$$\begin{bmatrix} y_1^M \\ y_2^M \\ \vdots \\ y_n^M \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\boldsymbol{\theta} = [a \quad b]^T;$$

$$\mathbf{Y}^M = \mathbf{X}\boldsymbol{\theta}$$

$$J(\boldsymbol{\theta}) = (\mathbf{Y} - \mathbf{Y}^M)^T (\mathbf{Y} - \mathbf{Y}^M)$$

- constrained optimization problem

$$\min_{\theta} J(\theta) = (\mathbf{Y} - \mathbf{Y}^M)^T (\mathbf{Y} - \mathbf{Y}^M)$$

s.t.

$$\mathbf{Y}^M = \mathbf{X} \theta$$

$$J(\theta) = (\mathbf{Y} - \mathbf{X}\theta)^T (\mathbf{Y} - \mathbf{X}\theta)$$

- Set $\frac{\partial J(\theta)}{\partial \theta} = 0$ for optimality,

$$\frac{\partial J(\theta)}{\partial \theta} = 0 = -\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\theta) - (\mathbf{Y} - \mathbf{X}\theta)^T \mathbf{X}$$

$$2\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\theta) = 0$$

$$\theta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

$$\theta = \mathbf{X} \setminus \mathbf{Y}$$

Example 7-1-1

Let the process model be $y = a + be^{3x} + ce^{-3x}$. Perform the least-squares method to estimate the model parameters, a , b , and c , based on the following experimental data:

x	0.5	1	2	3
y	1	2	2	1

Example 6-1-3

Ans:

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}}_{\mathbf{B}} = \underbrace{\begin{bmatrix} 1 & e^{3x_1} & e^{-3x_1} \\ 1 & e^{3x_2} & e^{-3x_2} \\ 1 & e^{3x_3} & e^{-3x_3} \\ 1 & e^{3x_4} & e^{-3x_4} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

B **A**

$$\mathbf{A} = \begin{bmatrix} 1 & e^{1.5} & e^{-1.5} \\ 1 & e^3 & e^{-3} \\ 1 & e^6 & e^{-6} \\ 1 & e^9 & e^{-9} \end{bmatrix}$$

Ans:

$$\mathbf{B} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{B}$$

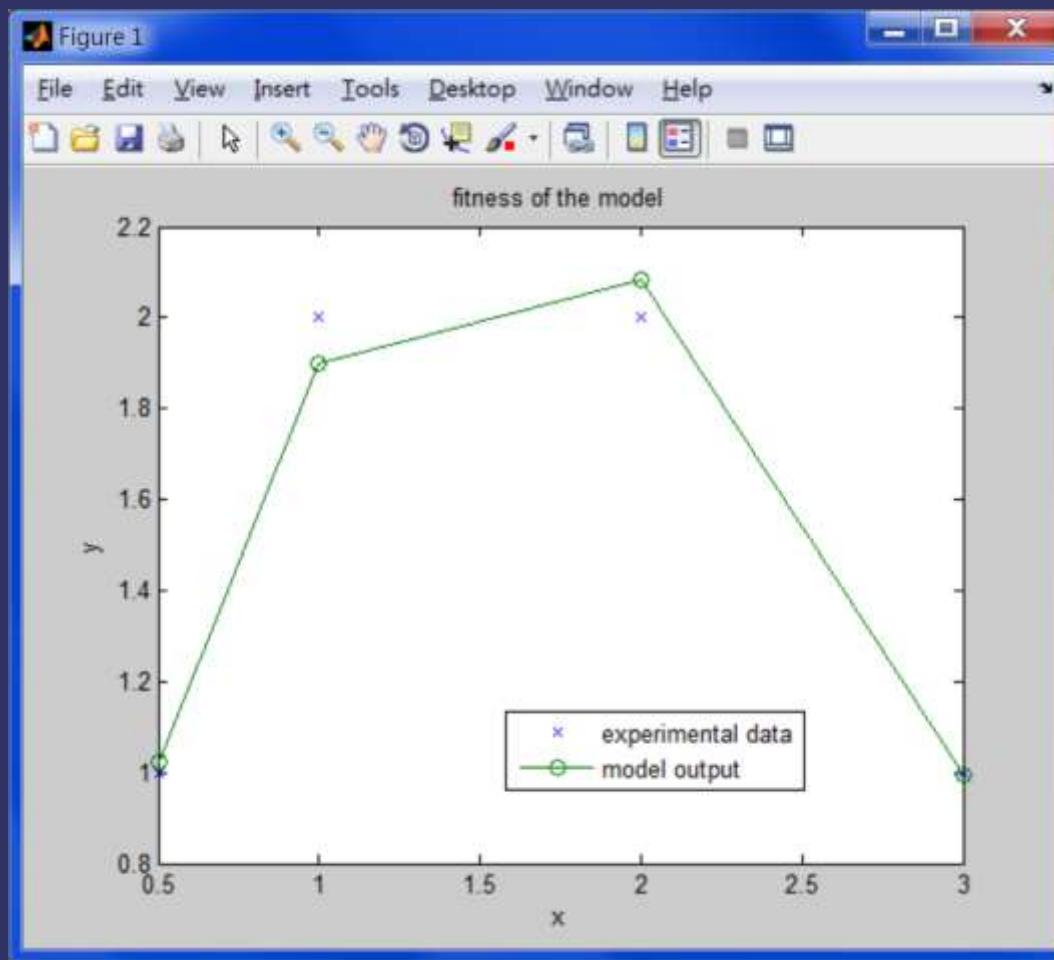
```
>> x = [0.5 1 2 3];
>> y = [1 2 2 1];
>> B = y';
>> A = [ones(4, 1) exp(3*x') exp(-3*x')];
>> p = inv(A'*A)*A'*B
p =
2.1539
-0.0001
-5.0711
```

Ans:

```
>> A\B  
ans =  
2.1539  
-0.0001  
-5.0711
```

```
>> y_model = A*p; % model outputs  
>> plot (x, y, 'x', x, y_model, '-o');  
>> xlabel('x'); ylabel('y');  
>> title('fitness of the model')  
>> legend('experimental data', 'model output')
```

Ans:



```
>> J=sum((y'-y_model).^2)  
J=  
0.0178
```

Example 7-1-2

Reconsider Example 7-1-1 with the model of Estimate the model parameters, a and b , based on the experimental data and verify the validity of this new candidate model.

Ans:

$$\ln y = \ln a + \ln x - bx$$

$$\alpha = \ln a$$

$$\underbrace{\begin{bmatrix} 1 & -x_1 \\ 1 & -x_2 \\ 1 & -x_3 \\ 1 & -x_4 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha \\ b \end{bmatrix}}_{\mathbf{B}} = \underbrace{\begin{bmatrix} \ln y_1 - \ln x_1 \\ \ln y_2 - \ln x_2 \\ \ln y_3 - \ln x_3 \\ \ln y_4 - \ln x_4 \end{bmatrix}}_{\mathbf{B}}$$

A

B

Ans:

```
>> x=[0.5 1 2 3];  
>> y=[1 2 2 1];  
>> B=log(y')-log(x');  
>> A=[ones(4,1) -x'];  
>> p=A\B  
p =  
    1.2722  
    0.7386
```

$$a = \exp(p(1))$$

$$b = p(2).$$

Ans:

```
>> a=exp(p(1)); b=p(2);
>> y_model=a*x.*exp(-b*x) % model prediction
y_model =
    1.2333   1.7050   1.6292   1.1676

>> error=y-y_model      % model errors
error =
   -0.2333   0.2950   0.3708  -0.1676

>> J=sum(error.^2)       % sum of squared errors
J =
    0.3071
```

7.1.2 Nonlinear least-squares methods

- general nonlinear model,

$$y = f(x, \mathbf{p})$$

- experimental data
- given by $(x_i, y_i), i = 1, 2, \dots, n,$
- search for the model parameter vector \mathbf{p} such that the sum of the squared errors $J = \sum_{i=1}^n e_i^2(\mathbf{p})$ (the objective function) is minimized where $e_i(\mathbf{p}) = y_i - f(x_i, \mathbf{p}).$

[p, resnorm, residual]=lsqnonlin('fun', p0, lb, ub, options)

Output argument

Description

p

The vector of the model parameter estimates

resnorm

The objective function value, J

residual

The vector of the value of $y_i - f(x_i, \mathbf{p})$, i.e., the value of the error e_i at each data point

Input argument	Description
p0	Initial guess values of the model parameters (vector)
lb	The lower bound of the model parameters (vector)
ub	The upper bound of the model parameters (vector)
options	The settings of the options required for the parameter estimation. It can be assigned with the command optimset. Refer to Chapter 6 for its detailed usage.
fun	<p>Name of the error function file whose contents should be coded as follows:</p> <pre>function E = fun(p) global x y % import the data of x_i and y_i E =... % compute the value of e_i. Note that E is a vector.</pre>

```
[p, resnorm, residualJ = lsqcurvefit( 'fun', p0, x, y,lb, ub, options)
```

```
function y_model=fun(p, x)
```

```
y_model=... % model output
```

```
[p, residual, jacobian]=nlinfit(x, y, 'fun', p0)
```

$\gamma \cong 0.5$.

Example 7-1-3

Nonlinear model parameter estimation

Assume the relationship between the input x and the output y of a system obeys the following model equation:

$$y = \alpha x^2 + \beta \sin(x) + \gamma x^3$$

Estimate the model parameters based on the following experimental data:

x	3.6	7.7	9.3	4.1	8.6	2.8	1.3	7.9	10.0	5.4
y	16.5	150.6	263.1	24.7	208.5	9.9	2.7	163.9	325.0	54.3

Note that the approximate model parameter values are $\alpha \cong 0.5$, $\beta \cong 0.5$, $\gamma \cong 0.5$,

Solution :

— ex7_1_3.m —

```
function ex7_1_3
%
% Example 7-1-3 Nonlinear model parameter estimation using lsqcurvefit
%
clear; close all;
clc
%
% Experimental data
%
xdata = [3.6 7.7 9.3 4.1 8.6 2.8 1.3 7.9 10.0 5.4];
ydata = [16.5 150.6 263.1 24.7 208.5 9.9 2.7 163.9 325.0 54.3];
%
% Initial guess values
%
p0 = [0.5 0.5 0.5];
%
```

Solution :

ex7_1_3.m

```
% Parameter estimation using lsqcurvefit
%
[p, resnorm, residual] = lsqcurvefit(@fun7_1_3, p0, xdata, ydata);
alpha=p(1); beta=p(2); gamma=p(3);
%
% Results printing
%
fprintf('\n The estimated parameter values are:')
fprintf('\n alpha=% .4f, beta=% .4f, gamma=% .4 f.', alpha, beta, gamma)
fprintf('\n Sum of the squared errors = %.4f.\n', resnorm)
%
% Plotting a figure to show the fitness of the model
%
x = sort(xdata);
y_model = fun7_1_3(p, x);
plot(xdata, ydata, '*', x, y_model, '-')
```

Solution :

— ex7_1_3.m —

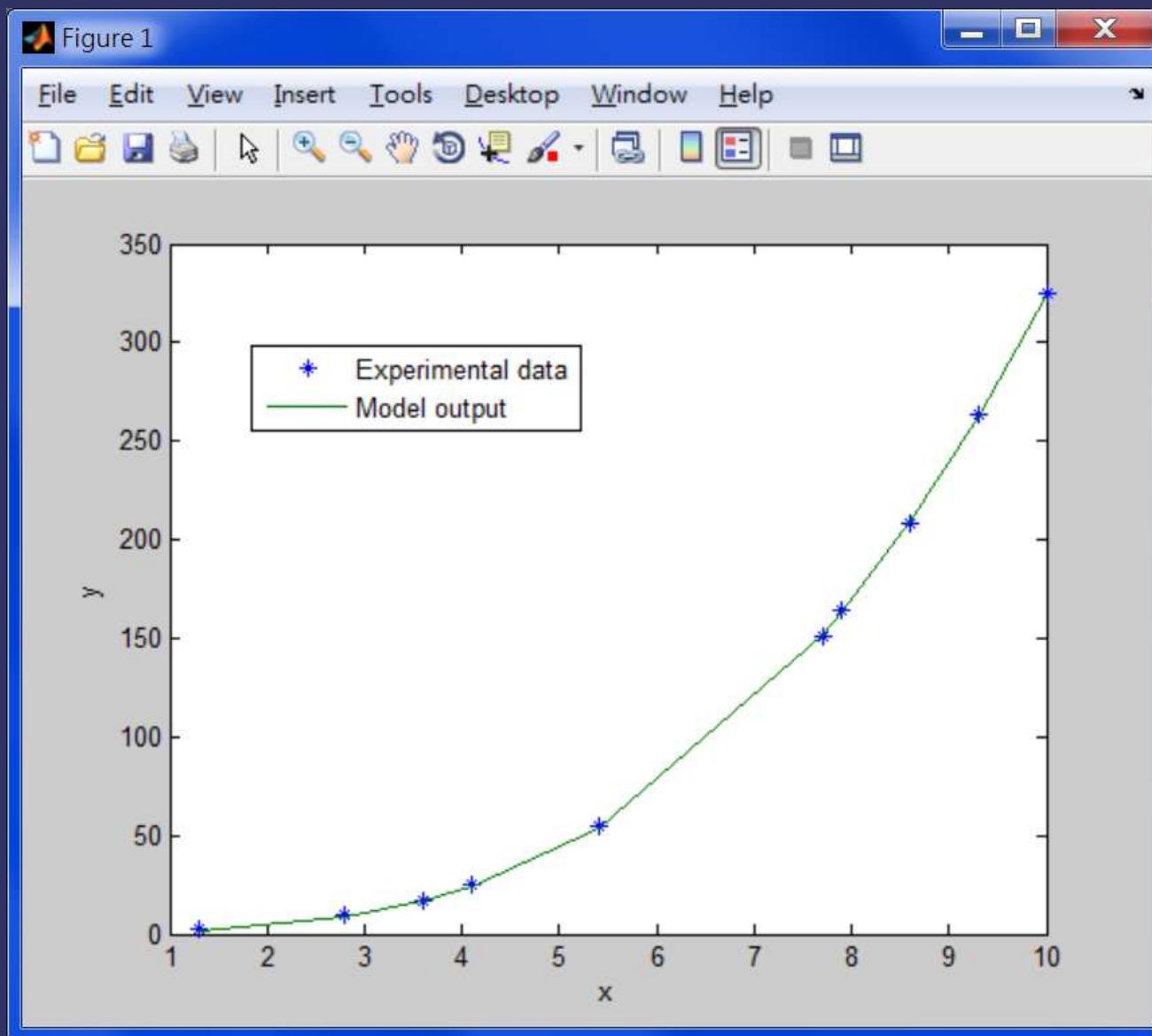
```
xlabel('x')
ylabel('y')
legend('Experimental data', 'Model output')
%
% Process model
%
function y_model = fun7_1_3(p, xdata)
alpha = p(1); beta = p(2); gamma = p(3);
y_model=alpha*xdata.^2+beta*sin(xdata)+gamma*xdata.^3;
```

Execution results:

The estimated parameter values are:

alpha = 0.2269, beta = 0.3385, gamma = 0.3022.

Sum of the squared errors = 6.2950.



7.1.3 The confidence interval of parameter estimation

$$P\left(\theta^* - z \frac{\sigma}{\sqrt{n}} \leq \theta \leq \theta^* + z \frac{\sigma}{\sqrt{n}}\right) = r$$

- For 95% confidence interval, one has $r = 0.95$ and $z = 1.96$ according to the normal distribution.

`ci = nlparci(p, residual, jacobian)`

Example 7-1-4

Confidence interval of the estimated parameter values

Deduce the 95% confidence interval of the estimated parameter values for the parameter estimation problem stated in Example 7-1-3.

Solution :

— ex7_1_4.m —

```
function ex7_1_4
%
% Example 7-1-4 The 95% confidence interval of the estimated parameter values
%
clear
clc
%
% Experimental data
%
xdata = [3.6 7.7 9.3 4.1 8.6 2.8 1.3 7.9 10.0 5.4];
ydata = [16.5 150.6 263.1 24.7 208.5 9.9 2.7 163.9 325.0 54.3];
%
% Initial guess values of the parameters
%
p0 = [0.5 0.5 0.5];
%
% Parameter estimation using nlinfit
```

Solution :

— ex7_1_4.m —

```
%  
[p, res, jacob] = nlinfit(xdata, ydata, @fun7_1_4, p0);  
ci = nlparci (p, res, jacob); % Calculate the 95% confidence interval  
alpha = p (1); beta = p (2); gamma = p(3);  
%  
% Results printing  
%  
fprintf('\n The estimated parameter values are:')  
fprintf('\n alpha = %.4f, beta=% .4f, gamma=% .4f\n', alpha, beta, gamma)  
fprintf('\n The 95%% confidence interval of the model parameters:')  
fprintf('\n %.4f <= alpha <= %.4f', ci(1,1), ci(1,2))  
fprintf('\n %.4f <= beta<= %.4f', ci(2,1),ci(2,2))  
fprintf('\n %.4f <= gamma <= %.4f\n', ci(3,1),ci(3,2))  
%  
% Process model  
%
```

Solution :

— ex7_1_4.m —

```
function y_model=fun7_1_4(p, xdata)
alpha=p(1); beta=p(2); gamma=p(3);
y_model = alpha*xdata.^2+beta*sin(xdata)+gamma*xdata.^3;
```

Execution results:

The estimated parameter values are:

$$\text{alpha} = 0.2269, \text{beta} = 0.3385, \text{gamma} = 0.3022$$

The 95% confidence interval of the model parameters:

$$0.1333 \leq \text{alpha} \leq 0.3205$$

$$-0.6654 \leq \text{beta} \leq 1.3424$$

$$0.2917 \leq \text{gamma} \leq 0.3126$$

7.2 Chemical engineering examples

Example 7-2-1

The solubility model of sulfur dioxide

The relationship between the solubility of sulfur dioxide (SO_2) in water and its partial pressure in the gas phase can be modeled as (Leu, 1985):

$$x = ap + b\sqrt{p}$$

where x denotes the mole fraction of SO_2 in the water solution, p is the partial pressure (atm) of SO_2 in the gas phase. Under 15°C, the following experimental data were obtained:

P _{SO₂} (mmHg)	0.3	0.8	2.2	3.8	5.7	10.0	19.3	28.0	44.0
C _w (g-SO ₂ /100g-H ₂ O)	0.02	0.05	0.10	0.15	0.20	0.30	0.50	0.70	1.00

Estimate the model parameters, a and b , based on the experimental data.

Problem formulation and analysis:

$$x = \frac{C_w / M_{\text{SO}_2}}{C_w / M_{\text{SO}_2} + 100 / M_{\text{H}_2\text{O}}}$$

$$p = \frac{P_{\text{SO}_2}}{760}$$

$$\underbrace{\begin{bmatrix} p_1 & \sqrt{p_1} \\ p_2 & \sqrt{p_2} \\ \vdots & \vdots \\ p_n & \sqrt{p_n} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} a \\ b \end{bmatrix} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\mathbf{B}}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{B} = \mathbf{A} \setminus \mathbf{B}$$

MATLAB program design:

— ex7_2_1.m —

```
%  
% Example 7-2-1 SO2 solubility model  
%  
clear; close all;  
clc  
%  
% Experimental data  
%  
pSO2 = [0.3 0.8 2.2 3.8 5.7 10.0 19.3 28 44];  
cw = [0.02 0.05 0.1 0.15 0.2 0.3 0.5 0.7 1.0];  
%  
MSO2 = 64; % SO2 molecular weight  
MH2O = 18; %H2O molecular weight  
%  
p = pSO2/760; % conversion of pressure units  
x = (cw/MSO2)./(cw/MSO2+100/MH2O); % mole fraction of SO2  
%
```

MATLAB program design:

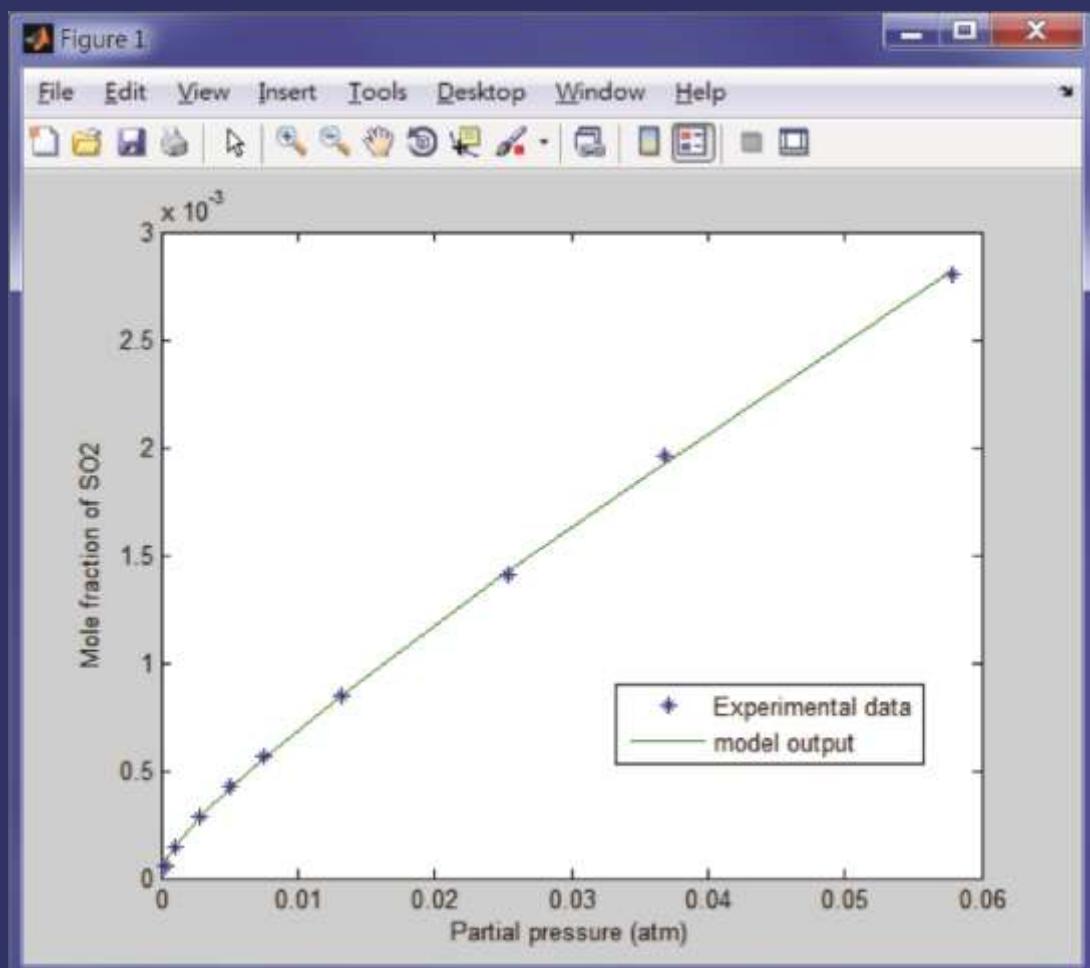
— ex7_2_1.m —

```
A = [p' sqrt(p')];  
B = x';  
y = A\B;  
a = y(1);  
b = y(2);  
%  
% Results printing  
%  
fprintf('\n The estimated parameter values are a=%5f, b=% 5f.\n', a, b)  
%  
% Plotting a figure to show the model fitness  
%  
fit = a*p+b*sqrt (p);  
plot(p, x, '*', p, fit)  
xlabel('Partial pressure (atm)')  
ylabel('Mole fraction of SO2')  
legend('Experimental data', 'model output')
```

Execution results:

```
>> ex7_2_1
```

The estimated parameter values are $a = 0.03439$, $b = 0.00344$



Example 7-2-2

The rate equation of a catalyzed reaction system

$$r_0 = \frac{k K_H^3 K_B P_H^3 P_B}{(1 + K_H P_H + K_B P_B)^4}$$

Estimate the reaction rate constants, k , K_H and K_B , in the above rate equation based on the following experimental data measured at 160°C:

P_H (atm)	0.7494	0.6721	0.5776	0.5075	0.9256
-------------	--------	--------	--------	--------	--------

P_B (atm)	0.2670	0.3424	0.4342	0.5043	0.1020
-------------	--------	--------	--------	--------	--------

r_o (g-mole/g·hr)	0.2182	0.2208	0.2235	0.1892	0.1176
---------------------	--------	--------	--------	--------	--------

P_H (atm)	0.9266	0.8766	0.7564	0.5617	0.5241
-------------	--------	--------	--------	--------	--------

P_B (atm)	0.0997	0.1471	0.2607	0.4501	0.4877
-------------	--------	--------	--------	--------	--------

r_o (g-mole/g·hr)	0.1151	0.1472	0.2178	0.2122	0.2024
---------------------	--------	--------	--------	--------	--------

Problem formulation and analysis:

Step 1: Divide $P_H^3 P_B$ on both sides of the reaction rate equation

$$\frac{r_0}{P_H^3 P_B} = \frac{k K_H^3 K_B}{(1 + K_H P_H + K_B P_B)^4}$$

Step 2: Take the reciprocal of the above equation and then the one-fourth power on both sides

$$R = a + bP_H + cP_B$$

$$R = \left(\frac{P_H^3 P_B}{r_0} \right)^{\frac{1}{4}}$$

$$a = \frac{1}{(kK_H^3 K_B)^{\frac{1}{4}}}$$

$$b = \frac{K_H}{(kK_H^3 K_B)^{\frac{1}{4}}}$$

and

$$b = \frac{K_H}{(kK_H^3 K_B)^{\frac{1}{4}}}$$



$$\underbrace{\begin{bmatrix} 1 & P_{H,1} & P_{B,1} \\ 1 & P_{H,2} & P_{B,2} \\ \vdots & \vdots & \vdots \\ 1 & P_{H,n} & P_{B,n} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \underbrace{\begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{bmatrix}}_{\mathbf{B}}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{B} = \mathbf{A} \setminus \mathbf{B}$$

$$K_H = \frac{b}{a}, \quad K_B = \frac{c}{a}, \quad k = a^{-4} K_H^{-3} K_B^{-1}.$$

MATLAB program design:

— ex7_2_2.m —

```
%  
% Example 7-2-2 Reaction rate equation for a catalyzed reaction system  
%  
clear  
clc  
%  
% Experimental data  
%  
PH = [0.7494 0.6721 0.5776 0.5075 0.9256 0.9266 0.8766...  
      0.7564 0.5617 0.5241];  
PB = [0.2670 0.3424 0.4342 0.5043 0.1020 0.0997 0.1471...  
      0.2607 0.4501 0.4877];  
r0 = [0.2182 0.2208 0.2235 0.1892 0.1176 0.1151 0.1472...  
      0.2178 0.2122 0.2024];  
%  
n = length(r0);
```

MATLAB program design:

— ex7_2_2.m —

```
A = [ones(n,1) PH' PB'];  
R = (PH.^3.*PB./r0).^(1/4);  
B = R';  
%  
% calculate the unknown parameter values  
%  
y = A\B;  
a = y(1); b = y(2); c = y(3);  
%  
% Model parameters  
%  
KH=b/a;  
KB=c/a;  
k=a^(-4)*KH^(-3)/KB;  
%  
% Results printing  
%
```

MATLAB program design:

— ex7_2_2.m —

```
fprintf('\n The estimated model parameter values are:')
fprintf('\n KH=% .4f, KB=% .4f, k=% .4f.\n', KH, KB, k)
%
% Model error analysis
%
r0_model=k*KH^3*KB*PH.^3.*PB./(1+KH*PH+KB*PB).^4;
error=r0-r0_model;
%
disp(' ')
disp('data no.   r0 (measured)   r0 (model)      error')
for i=1:n
    fprintf('\n %2i %15.4f %15.4f %15.4f', i, r0(i), r0_model(i), error(i))
end
```

Execution results:

```
>> ex7_2_2
```

The estimated model parameter values are:

$$KH = 8.3798, KB = 4.9252, k = 3.6238.$$

data no.	r0 (measured)	r0 (model)	error
1	0.2182	0.2163	0.0019
2	0.2208	0.2280	-0.0072
3	0.2235	0.2168	0.0067
4	0.1892	0.1932	-0.0040
5	0.1176	0.1156	0.0020
6	0.1151	0.1135	0.0016
7	0.1472	0.1538	-0.0066
8	0.2178	0.2144	0.0034
9	0.2122	0.2125	-0.0003
10	0.2024	0.1998	0.0026

Example 7-2-3

Isothermal adsorption model for activated carbon

$$Q = \frac{bC}{1 + aC^\beta}$$

Estimate the model parameters and the 95% confidence interval of the parameter estimates based on the following experimental data:

C (mg/L)	1.60	4.52	6.80	8.16	11.5	12.7	18.2	29.0
----------	------	------	------	------	------	------	------	------

Q (mg/g-activated carbon)	170.7	228.1	258.0	283.7	321.3	335.4	378.6	401.3
---------------------------	-------	-------	-------	-------	-------	-------	-------	-------

C (mg/L)	38.9	48.9	57.3	65.0	75.0	85.0	90.0	100.0
----------	------	------	------	------	------	------	------	-------

Q (mg/g-activated carbon)	422.3	425.3	429.0	430.9	431.6	431.7	431.6	432.1
---------------------------	-------	-------	-------	-------	-------	-------	-------	-------

Note that the parameter values are roughly in the range of $0 \leq a \leq 1$, $100 \leq b \leq 200$, and $0 \leq \beta \leq 1$.

MATLAB program design:

— ex7_2_3.m —

```
function ex7_2_3
%
% Example 7-2-3 Isothermal adsorption model for activated carbon
%
clear; close all;
clc
%
% Experimental data
%
C1 = [1.60 4.52 6.80 8.16 11.5 12.7 18.2 29.0];
C2 = [38.90 48.9 57.3 65.0 75.0 85.0 90.0 100.0];
Q1 = [170.7 228.1 258.0 283.7 321.3 335.4 378.6 401.3];
Q2 = [422.3 425.3 429.0 430.9 431.6 431.7 431.6 432.1];
C = [C1 C2];
Q = [Q1 Q2];
%
% Upper and lower bounds of the model parameters
```

MATLAB program design:

— ex7_2_3.m —

```
%  
pL = [0 100 0];  
pU = [1 200 1];  
%  
% Parameter estimation and 95% confidence interval  
%  
p0 = (pL+pU)/2; % Initial values  
pl = lsqcurvefit(@fun7_2_3, p0, C, Q, pL, pU); % Initial estimation  
[p, res, jacob] = nlinfit(C, Q, @fun7_2_3, pl); % Refine the estimation  
ci = nlparci(p, res, jacob);  
a = p(1); b = p(2); beta = p(3);  
%  
% Results printing  
%  
fprintf('\n The estimated model parameters are:')  
fprintf('\n a=% .4f, b=% .4f, beta = % .4f\n', a, b, beta)  
fprintf('\n The 95% confidence interval for each model parameter is:')
```

MATLAB program design:

— ex7_2_3.m —

```
fprintf('\n %.4f <= a <= %.4f', ci(1,1), ci(1,2))
fprintf('\n %.4f <= a <= %.4f', ci(2,1), ci(2,2))
fprintf('\n %.4f <= a <= %.4f', ci(3,1), ci(3,2))
%
% Fitness of the model: analysis by plotting
%
Qm = fun7_2_3(p, C);
plot(C, Q, '*', C, Qm, 'r')
xlabel('Concentration of the solute (mg/L)')
ylabel('Adsorption amount (mg/g-activated carbon)')
legend('Experimental data', 'Model output')
%
% Adsorption model
%
function Q = fun7_2_3(p, C)
a = p(1); b = p(2); beta = p(3);
Q = b*C./(1 +a*C.^beta);
```

Execution results:

```
>> ex7_2_3
```

The estimated model parameters are:

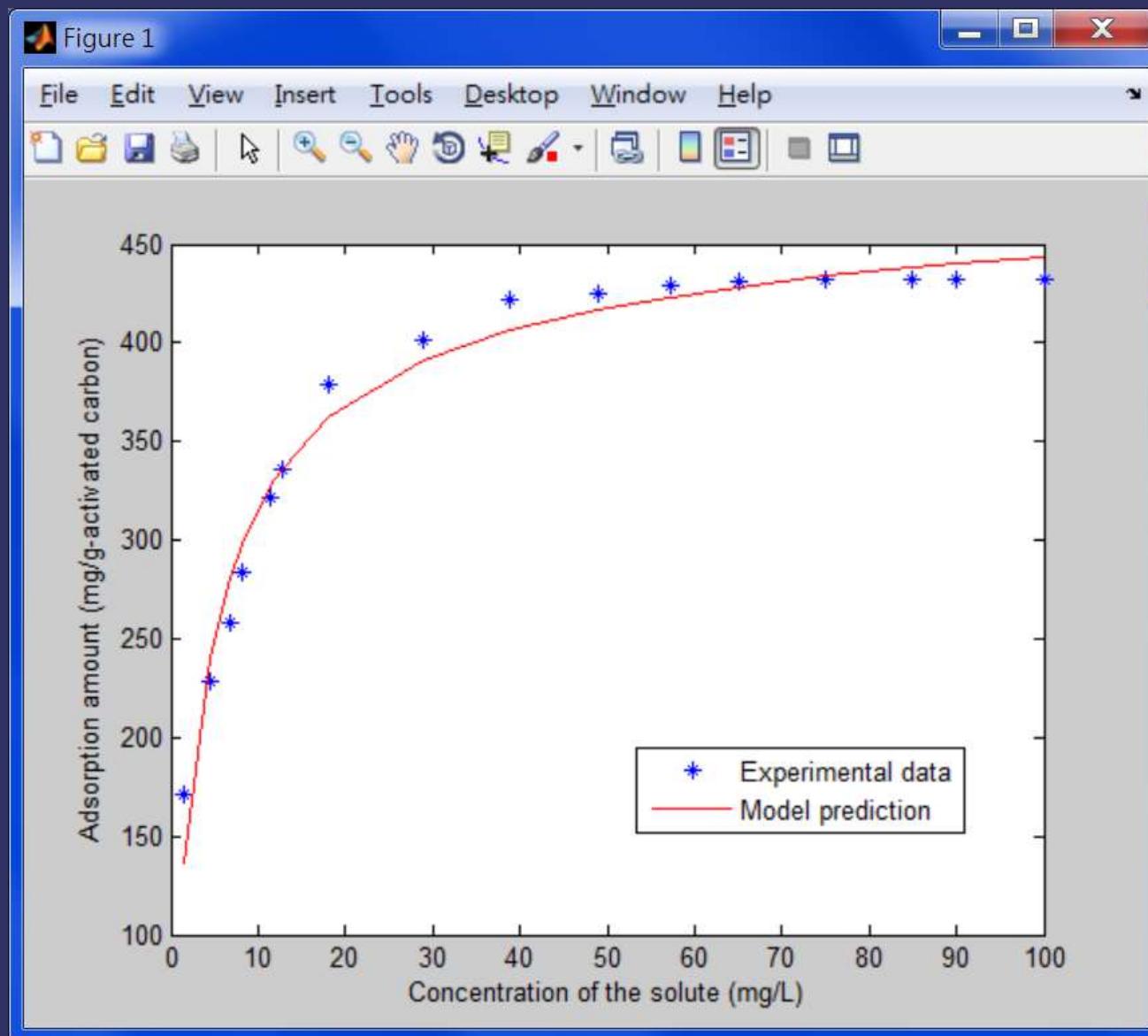
$$a=0.3478, b=131.3340, \beta = 0.9575$$

The 95% confidence interval for each model parameter is:

$$0.1111 \leq a \leq 0.5845$$

$$78.4555 \leq b \leq 184.2126$$

$$0.8909 \leq \beta \leq 1.0241$$



Example 7-2-4

Transfer function of a heating process

$$G(s) = \frac{T(s)}{Q(s)} = \frac{K_p e^{-t_d s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

time t (min)	0	1	2	3	4	5	6	7
T (°C)	0	0.6	2.1	5.7	12.3	19.6	26.4	32.2
time t (min)	8	9	10	20	30	40	70	100
T (°C)	36.8	41.3	44.6	58.9	61.2	61.2	61.2	61.2

Based on these step response data, estimate the SOPDT model parameters and calculate the 95% confidence interval of each parameter.

Problem formulation and analysis:

$$Q(s) = \frac{\Delta Q}{s}$$

$$T(s) = \frac{K_p e^{-t_d s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \frac{\Delta Q}{s}$$

$$T(t) = \begin{cases} 0, & t \leq t_d \\ K \left(1 - \frac{\tau_1}{\tau_1 - \tau_2} \exp\left(-\frac{t - t_d}{\tau_1}\right) + \frac{\tau_2}{\tau_1 - \tau_2} \exp\left(-\frac{t - t_d}{\tau_2}\right) \right), & t > t_d \end{cases}$$

where $K = K_p \Delta Q$.

MATLAB program design:

— ex7_2_4.m —

```
function ex7_2_4
%
% Example 7-2-4 Transfer function of a heating process
%
clear; close all;
clc
%
% Experimental data
%
dQ = 0.7; % Staircase changes in the flow at the inlet
t = [0:10 20 30 40 70 100]; % time (min)
T = [0 0.6 2.1 5.7 12.3 19.6 26.4 32.2 36.8 41.3 44.6...
      58.9 61.2 61.2 61.2 61.2]; % Changes in temperature (oC):
%
% Initial guess values
%
p0 = [50 3 2 1];
```

MATLAB program design:

— ex7_2_4.m —

```
%  
% Parameter estimation  
%  
[p, res, jacob] = nlinfit (t, T, @fun7_2_4, p0);  
ci = nlparci(p, res, jacob); % 95% confidence interval  
Kp = p(1)/dQ; T1 = p(2); T2 = p(3); td = p(4);  
%  
% Results printing  
%  
fprintf('\n The estimated model parameters are:')  
fprintf('\n Kp=%3f, tau_1=%.3f, tau_2=%.3f, td=%.3f.', Kp, T1, T2, td)  
fprintf('\n\n The 95% confidence interval for each model parameter is:')  
fprintf('\n %.4f <= Kp <= %.4f', ci(1,1), ci(1,2))  
fprintf('\n %.4f <= tau_1 <= %.4f', ci(2,1), ci(2,2))  
fprintf('\n %.4f <= tau_2 <= %.4f', ci(3,1), ci(3,2))  
fprintf('\n %.4f <= td <= %.4f\n', ci(4,1), ci(4,2))
```

MATLAB program design:

— ex7_2_4.m —

```
%  
% Fitness of the model  
%  
t_m = linspace(0, 100);  
T_m = fun7_2_4(p, t_m);  
plot(t, T, '*', t_m, T_m, 'r')  
xlabel('Time (min)')  
ylabel('System response, T (oC)')  
legend('Experimental data', 'Model output')  
%  
% Process model  
%  
function T = fun7_2_4(p, t)  
K = p(1); T1=p(2); T2=p(3); td=p(4);  
tt = t-td;  
n = length(t);  
for i =1:n
```

MATLAB program design:

— ex7_2_4.m —

```
if tt(i)>0
    T(i)=K*(1-T1/(T1-T2)*exp(-tt(i)/T1)+T2/(T1-T2)*exp(-tt(i)/T2));
else
    T(i) = 0;
end
end
```

Execution results:

```
>> ex7_2_4
```

The estimated model parameters are:

$$K_p = 87.467393, \tau_1 = 4.814, \tau_2 = 1.920, t_d = 1.347.$$

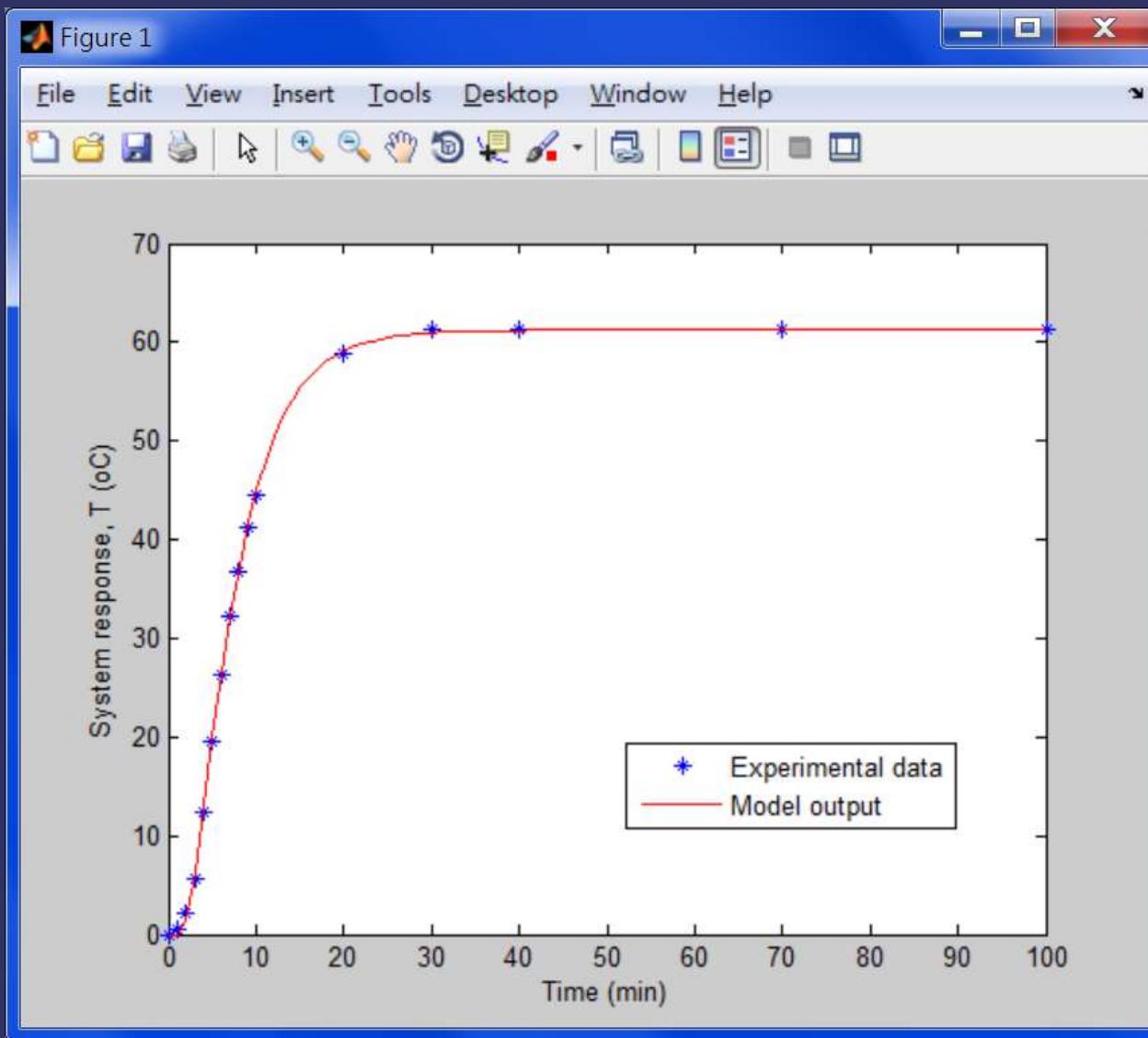
The 95% confidence interval for each model parameter is:

$$60.788859 \leq K_p \leq 61.6655$$

$$4.1371 \leq \tau_1 \leq 5.4911$$

$$1.2614 \leq \tau_2 \leq 2.5784$$

$$1.1303 \leq t_d \leq 1.5637$$



Example 7-2-5

Estimation of the reaction rate constants for a fermentation

$$\frac{dy_1}{dt} = k_1 y_1 \left(1 - \frac{y_1}{k_2} \right)$$

$$\frac{dy_2}{dt} = k_3 y_1 - k_4 y_2$$

Estimate the reaction rate constants, k_1 , k_2 , k_3 , and k_4 , in the dynamic model based on the following experimental data:

t (hr)	0	10	22	34	46	58	70	82	94
y1 (percent dry weight)	0.18	0.12	0.48	1.46	1.56	1.73	1.99	2.62	2.88
<hr/>									
y2 (units/mL)	0	0	0.0089	0.0062	0.2266	0.4373	0.6943	1.2459	1.4315
t (hr)	106	118	130	142	154	166	178	190	210
y1 (percent dry weight)	3.43	3.37	3.92	3.96	3.58	3.58	3.34	3.47	3.45
y2 (units/ mL)	2.0402	1.9278	2.1848	2.4204	2.4615	2.2830	2.7078	2.6542	2.6831

MATLAB program design:

— ex7_2_5.m —

```
function ex7_2_5
%
% Example 7-2-5 Estimation of reaction rate constants in a fermentation process
%
clear; close all;
clc
%
global k y0
%
% Experimental data
%
tout=[0 10 22 34 46 58 70 82 94 106 118 130 142 154 166 ...
       178 190 210]; % hr
y1=[0.18 0.12 0.48 1.46 1.56 1.73 1.99 2.62 2.88 3.43 3.37 ...
       3.92 3.96 3.58 3.58 3.34 3.47 3.45];
y2=[0 0 0.0089 0.0062 0.2266 0.4373 0.6943 1.2459 1.4315 2.0402 ...
       1.9278 2.1848 2.4204 2.4615 2.283 2.7078 2.6542 2.6831];
```

MATLAB program design:

— ex7_2_5.m —

```
y=[y1' y2'];  
y0=[0.18 0];  
%  
% Initial guess values of the parameters  
%  
p0=[0.01 3 0.01 0.02];  
%  
% Parameter bounds  
%  
pL=[0 0 0 0];  
pU=[1 5 1 1];  
%  
% Parameter estimation using lsqcurvefit  
%  
[p,resnorm,res,EXITFLAG,OUTPUT,LAMBDA,jacob]=lsqcurvefit(@fun7_2_5, p0, ...  
tout, y, pL, pU);  
ci=nlparci(p,res,jacob); % 95% confidence interval
```

MATLAB program design:

— ex7_2_5.m —

```
k1=p(1);k2=p(2);k3=p(3);k4=p(4);
k=[k1 k2 k3 k4];
%
% Result printing
%
fprintf('\n\n The estimated parameter values are:')
fprintf('\n k1=% .3f, k2=% .3f, k3=% .3f, k4=% .3f', k1, k2, k3, k4)
fprintf('\n\n The 95%% confidence interval of each parameter is:')
fprintf('\n %.3f <= k1 <= %.3f, ci(1,1), ci(1,2)')
fprintf('\n %.3f <= k2 <= %.3f, ci(2,1), ci(2,2)')
fprintf('\n %.3f <= k3 <= %.3f, ci(3,1), ci(3,2)')
fprintf('\n %.3f <= k4 <= %.3f\n', ci(4,1), ci(4,2))
%
% Model fitness analysis
%
y_m=fun7_2_5(k, tout);
figure(1)
```

MATLAB program design:

— ex7_2_5.m —

```
plot(tout, y(:,1), '*', tout, y_m(:,1), 'r')
xlabel('Time (hr)')
ylabel('Cell conc. (percent dry wt.)')
legend('Experimental data', 'Model output')
%
figure(2)
plot(tout,y(:,2),'*',tout,y_m(:,2),'r')
xlabel('Time (hr)')
ylabel('Conc. of penicillin (units/mL)')
legend('Experimental data', 'Model output')
%
% Process model outputs
%
function y_m=fun7_2_5(p1,tout)
global k y0
k=p1;
tout;
```

MATLAB program design:

— ex7_2_5.m —

```
[t, y_m]=ode15s(@model7_2_5, tout, y0);  
%  
% Kinetics model  
%  
function dydt=model7_2_5(t, y)  
global k y0  
k1=k(1);k2=k(2);k3=k(3);k4=k(4);  
y1=y(1);y2=y(2);  
dydt=[k1*y1*(1-y1/k2)  
k3*y1-k4*y2];
```

Execution results:

```
>> ex7_2_5
```

The estimated parameter values are:

$$k1 = 0.050, k2 = 3.609, k3 = 0.020, k4 = 0.026$$

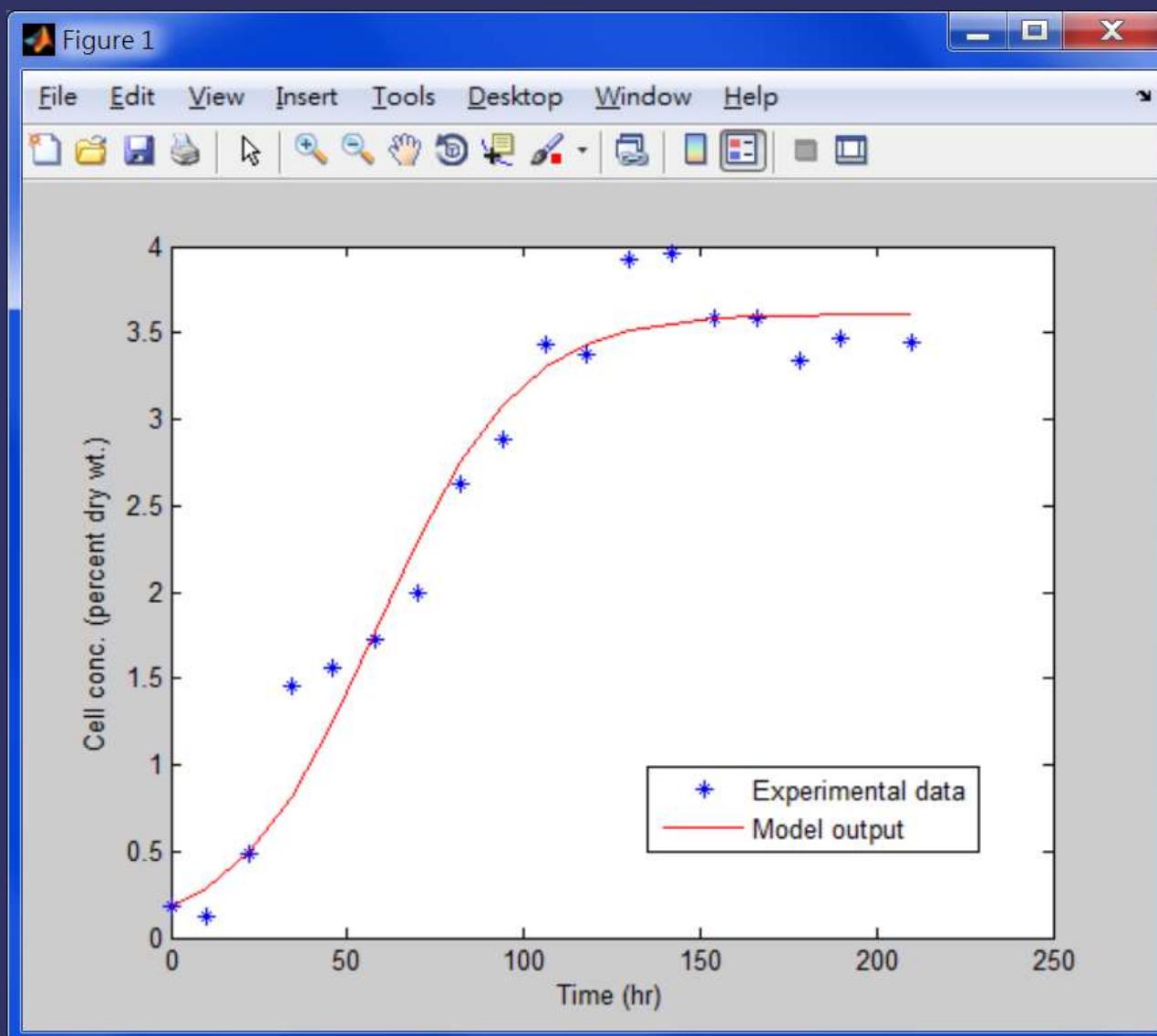
The 95% confidence interval of each parameter is:

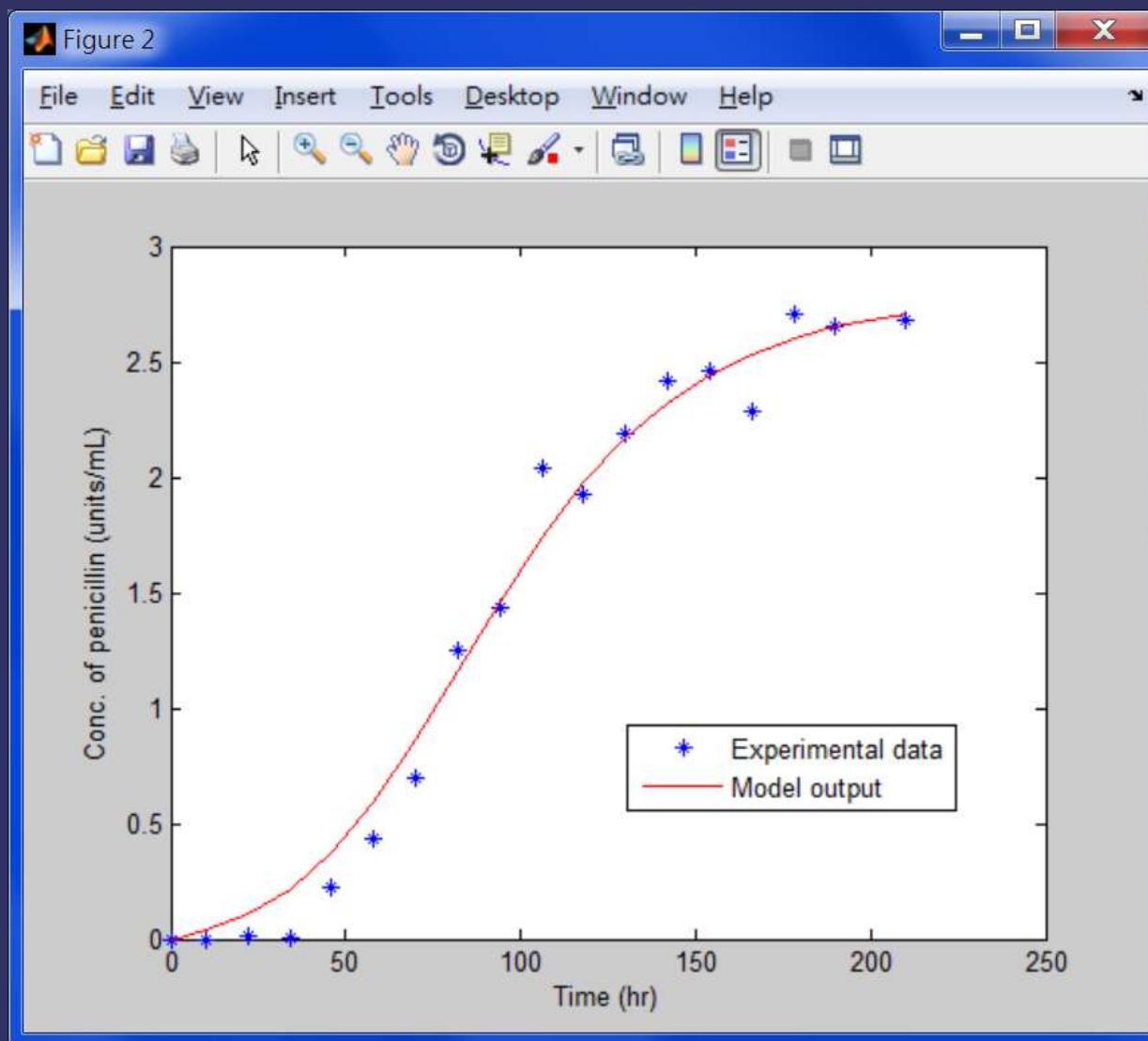
$$0.046 \leq k1 \leq 0.054$$

$$3.447 \leq k2 \leq 3.772$$

$$0.012 \leq k3 \leq 0.028$$

$$0.013 \leq k4 \leq 0.040$$





Example 7-2-6

Parameter estimation for a packed bed reactor described by a partial differential equation model

$$\frac{\partial T}{\partial z} = k_1 \frac{\partial^2 T}{\partial r^2} + k_3 (1 - C) \exp\left(k_5 \left(1 - \frac{1}{T}\right)\right)$$

$$\frac{\partial C}{\partial z} = k_2 \frac{\partial^2 C}{\partial r^2} + k_4 (1 - C) \exp\left(k_5 \left(1 - \frac{1}{T}\right)\right)$$

z	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
\bar{T}	1.00	1.02	1.06	1.10	1.17	1.32	1.85	2.18	2.09	2.01	1.96
\bar{C}	0	0.02	0.05	0.09	0.15	0.26	0.65	0.99	1.00	1.00	1.00

Estimate these model parameter values based on the experimental data.

$0.8 \leq k_1, k_2 \leq 1.4, 0.1 \leq k_3, k_4 \leq 0.5, 5.8 \leq k_5 \leq 12, k_1 \cong 1,$
 $k_2 \cong 1, k_3 \cong 0.3, k_4 \cong 0.3,$ and $k_5 \cong 10.$

$$\frac{\partial T}{\partial r} \Big|_{r=0} = \frac{\partial C}{\partial r} \Big|_{r=0} = 0$$

$$-\frac{\partial T}{\partial r} \Big|_{r=1} = \beta(T \Big|_{r=1} - T_w)$$

$$-\frac{\partial C}{\partial r} \Big|_{r=1} = 0$$

$$T \Big|_{z=0} = T_0$$

$$C \Big|_{z=0} = C_0$$

Problem formulation and analysis:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}_* * \frac{\partial}{\partial z} \begin{bmatrix} T \\ C \end{bmatrix} = \frac{\partial}{\partial r} \begin{bmatrix} k_1 \frac{\partial T}{\partial r} \\ k_2 \frac{\partial C}{\partial r} \end{bmatrix} + \begin{bmatrix} k_3(1-C) \exp\left(k_5\left(1-\frac{1}{T}\right)\right) \\ k_4(1-C) \exp\left(k_5\left(1-\frac{1}{T}\right)\right) \end{bmatrix}$$

where

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$f = \begin{bmatrix} k_1 \frac{\partial T}{\partial r} \\ k_2 \frac{\partial C}{\partial r} \end{bmatrix}$$

Problem formulation and analysis:

$$s = \begin{bmatrix} k_3(1-C) \exp\left(k_5\left(1-\frac{1}{T}\right)\right) \\ k_4(1-C) \exp\left(k_5\left(1-\frac{1}{T}\right)\right) \end{bmatrix}$$

The initial conditions at the position $z = 0$

$$T|_{z=0} = T_0$$

$$C|_{z=0} = C_0$$

The left boundary conditions at $r = 0$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} \frac{\partial T}{\partial r} \\ \frac{\partial C}{\partial r} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Problem formulation and analysis:

That is,

$$pl = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$ql = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

the right boundary conditions at $r = 1$

$$\begin{bmatrix} \beta(T|_{r=1} - T_w) \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}_* * \begin{bmatrix} \frac{\partial T}{\partial r} \\ \frac{\partial C}{\partial r} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Problem formulation and analysis:

$$pr = \begin{bmatrix} \beta(T|_{r=1} - T_w) \\ 0 \end{bmatrix}$$

$$qr = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

MATLAB program design:

— ex7_2_6.m —

```
function ex7_2_6
%
% Example 7-2-6 Parameter estimation of a PDE system
%
clear; close all;
clc
%
global k m r
global k
global z y
global T0 c0 Tw
%
% PDE system information
%
m=0; % cylinder
r=linspace(0,1,11);
z=linspace(0,1,11);
```

MATLAB program design:

— ex7_2_6.m —

```
T0=1; % initial temperature
c0=0; % initial concentration
Tw=0.9; % wall temperature
%
% Experimental data
%
y1=[1.00 1.02 1.06 1.10 1.17 1.32 1.85 2.18...
     2.09 2.01 1.96]; % average temperature
y2=[0 0.02 0.05 0.09 0.15 0.26 0.65 0.99...
     1.00 1.00 1.00]; % average concentration
%
y=[y1' y2'];
%
% initial guess values and the parameter bounds
%
p0=[1 1 0.3 0.3 10];
pL=[0.8 0.8 0.1 0.1 8];
pU=[1.4 1.4 0.5 0.5 12];
```

MATLAB program design:

— ex7_2_6.m —

```
%  
% Parameter estimation  
%  
options=optimset('TolFun', 1.e-6, 'TolX', 1.e-6);  
p=lsqnonlin(@fun7_2_6, p0, pL, pU, options);  
%  
k1=p(1); k2=p(2); k3=p(3); k4=p(4); k5=p(5);  
k=[k1 k2 k3 k4 k5];  
%  
% Results printing  
%  
fprintf('\n\n The estimated parameter values are:')  
fprintf('\n k1=%0.3f, k2=%0.3f, k3=%0.3f, k4=%0.3f, and k5=%0.3f', ...  
k1, k2, k3, k4, k5)  
%  
% System simulation based on the estimated model parameters  
%
```

MATLAB program design:

— ex7_2_6.m —

```
options=odeset('RelTol', 1.e-6, 'AbsTol', 1.e-6);
sol=pdepe(m,@ex7_2_6pdefun, @ex7_2_6ic, @ex7_2_6bc, r, z, options);
u1=sol(:,:,1); % temperature
u2=sol(:,:,2); % concentration
ym1=mean(u1'); % average temperature
ym2=mean(u2'); % average concentration
ym=[ym1' ym2'];
%
% Model fitness analysis via plotting
%
figure(1)
plot(z, y1, '*', z, ym1)
title('Fitness of the Model')
xlabel('z')
ylabel('Dimensionless average temperature')
legend('Experimental data', 'Model output')
axis([0 1 0 2.5])
```

MATLAB program design:

— ex7_2_6.m —

```
%  
figure(2)  
plot(z, y2, '*', z, ym2)  
title('Fitness of the Model')  
xlabel('z')  
ylabel('Dimensionless average concentration')  
legend('Experimental data','Model output')  
axis([0 1 -0.5 1.5])  
%  
figure(3)  
surf(r,z,u1)  
xlabel('r')  
ylabel('z')  
zlabel('Dimensionless temperature, T')  
%  
figure(4)  
surf(r,z,u2)
```

MATLAB program design:

— ex7_2_6.m —

```
xlabel('r')
ylabel('z')
zlabel('Dimensionless concentration, C')
%
% Error function
%
function J=fun7_2_6(p1)
global k m r
global z y
k=p1;
%
% PDE solution with pdepe
%
options=odeset('RelTol', 1.e-6, 'AbsTol', 1.e-6);
sol=pdepe(m, @ex7_2_6pdefun, @ex7_2_6ic, @ex7_2_6bc, r, z, options);
u1=sol(:,:,1); % temperature
u2=sol(:,:,2); % concentration
```

MATLAB program design:

— ex7_2_6.m —

```
ym1=mean(u1'); % average temperature  
ym2=mean(u2'); % average concentration  
ym=[ym1' ym2'];  
J=y-ym;  
%  
% PDE model  
%  
function [c, f, s]=ex7_2_6pdefun(r, z, u, DuDx)  
global k  
T=u(1); C=u(2);  
c=[1 1]';  
f=[k(1) k(2)]'.*DuDx;  
F=(1-C)*exp(k(5)*(1-1/T));  
s=[k(3)*F k(4)*F]';  
%  
% Initial conditions  
%
```

MATLAB program design:

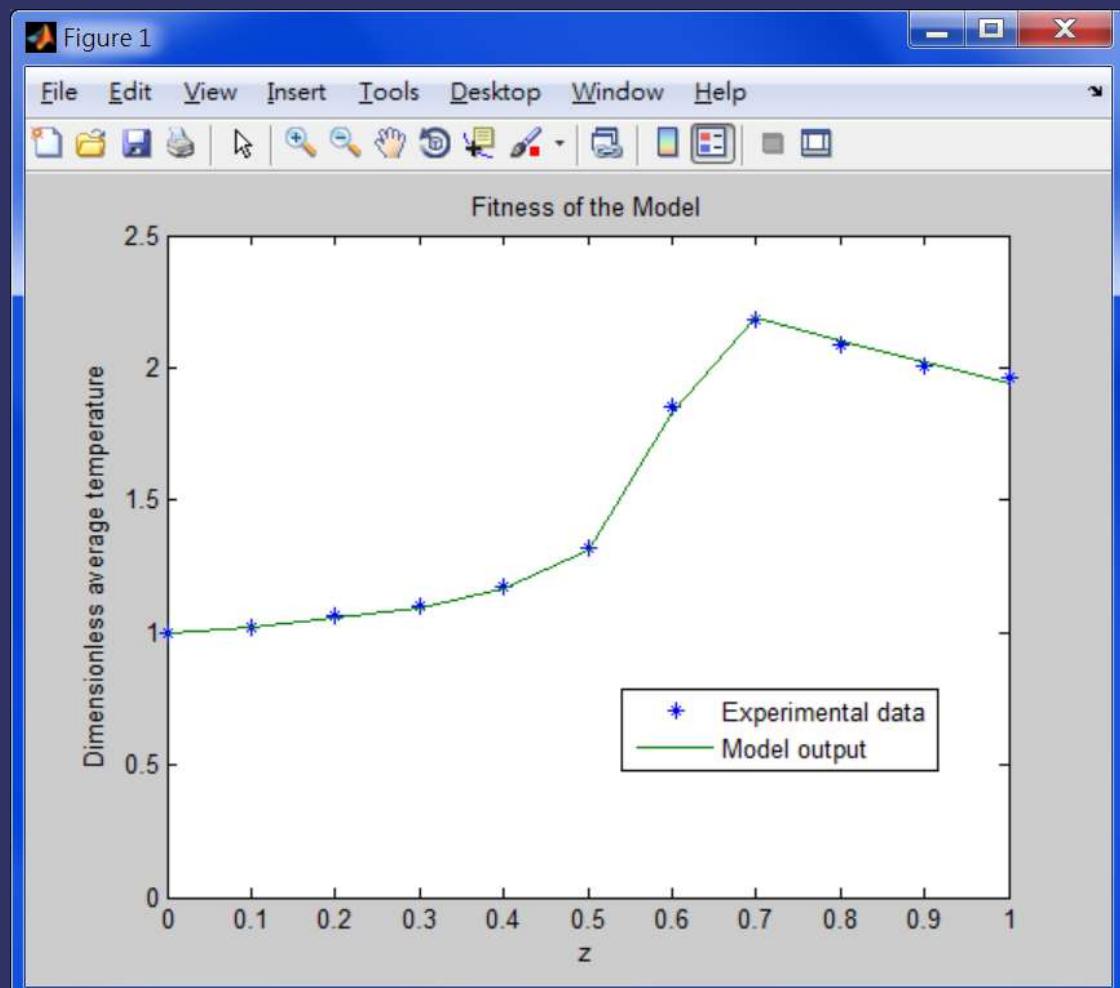
— ex7_2_6.m —

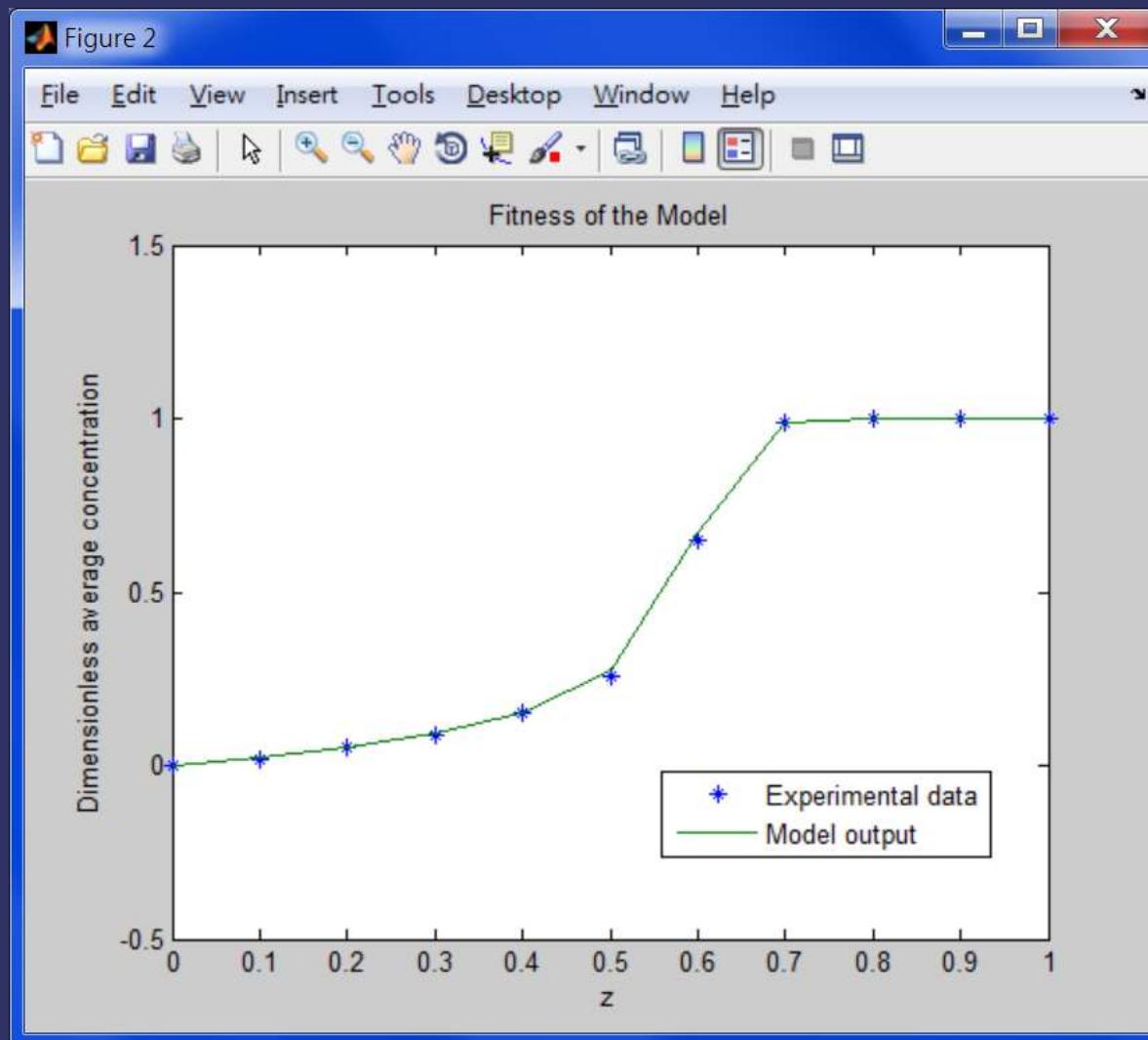
```
function u0=ex7_2_6ic(r)
global T0 c0 Tw
u0=[T0 c0]';
%
% Boundary conditions
%
function [pl, ql, pr, qr]=ex7_2_6bc(xl, ul, xr, ur, z)
global T0 c0 Tw
beta=1;
pl=[0 0]';
ql=[1 1]';
pr=[beta*(ur(1)-Tw) 0]';
qr=[1 1]';
```

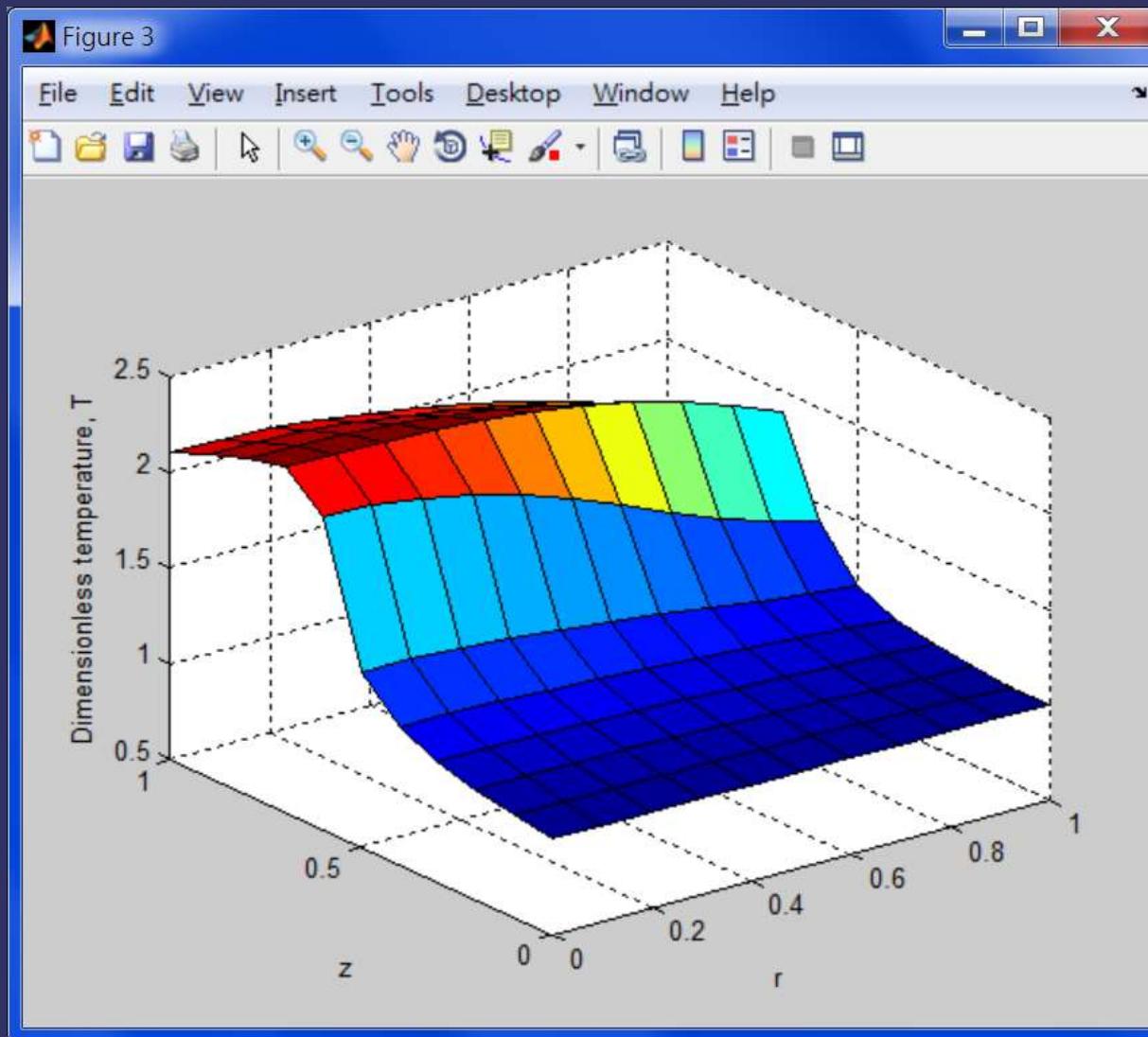
Execution results:

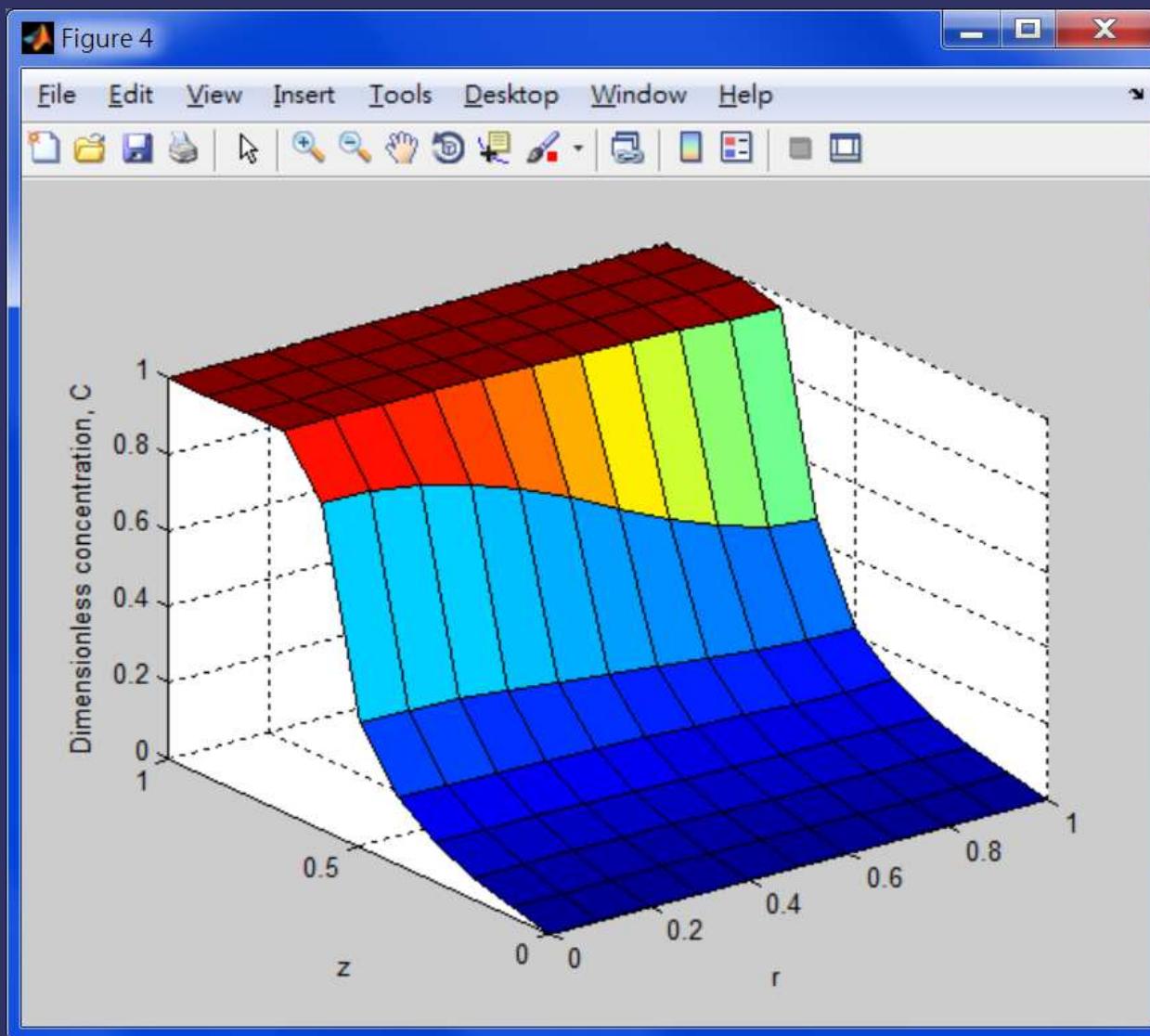
The estimated parameter values are:

$k_1=0.846$, $k_2=0.800$,
 $k_3=0.290$, $k_4=0.208$, and
 $k_5=10.300$.









Example 7-2-7

Parameter estimation using D-optimal experimental design method

Hougen-Watson equation:

$$y = \frac{\alpha_1 x_2 - x_3 / \alpha_5}{1 + \alpha_2 x_1 + \alpha_3 x_2 + \alpha_4 x_3}$$

Perform the D-optimal experimental design method (*Atkinson and Donev*, 1992) to gather response data and based on which estimate the parameters of the Hougen-Watson model. The admissible operating ranges of the partial pressures are as follows:

$$100 \leq x_1 \leq 470, \quad 80 \leq x_2 \leq 300, \text{ and } 10 \leq x_3 \leq 120.$$

NOTE:

The true parameter values are

$$\alpha_1^* = 1.250,$$

$$\alpha_2^* = 0.064,$$

$$\alpha_3^* = 0.0378,$$

$$\alpha_4^* = 0.1326,$$

$$\alpha_5^* = 1.5183,$$

and, for generating the experimental data, the measured output contains random noises with zero mean and standard deviation of 0.02.

MATLAB program design:

simulator of the reaction system

$$y^* = \frac{\alpha_1^* x_2 - x_3 / \alpha_5^*}{1 + \alpha_2^* x_1 + \alpha_3^* x_2 + \alpha_4^* x_3}$$

[settings, x] = cordexch (n, m, model)

Output argument	Description
settings	m×n collocation matrix
x	Design matrix

Input argument	Description
n	The number of independent variables
m	Number of experiments
model	The regression model used, which has four options “linear” or “l” “interaction” or “i” “quadratic” or “q” “pure quadratic” or “p”
	linear model with constants and linear terms (defaulted) constant, linear, and interaction terms the above interaction model plus squared terms model containing constants, linear, and squared terms

Level	Partial pressure of hydrogen, x_1	Partial pressure of pentane, x_2	Partial pressure of isopentane, x_3
1 (high level)	470	300	120
0 (medium level)	285	190	65
-1 (low level)	100	80	10

 ex7_2_7.m

```
function ex7_2_7
%
% Example 7-2-7 Parameter estimation using D-optimal design of experiments
%
clear
clc
%
% Levels of each variable
%
x=[470 300 120 % level 1;]upper;
    285 190 65 % level 2 (medium)
    100 80 10 ]; % level 3 (lower)
%
% D-optimal design of experiments
%
n=13; % no. of experiments
set=cordexch(3, n, 'q'); % produce the factor settings matrix
dx=(x(1,:)-x(3,:))/2;
bx=x(2,:);
```

ex7_2_7.m

```
exp=set.*dx(ones(n,1),:)+bx(ones(n,1),:); % experimental conditions
%
% Produce the experimental data based on the simulator
%
x1=exp(:,1);
x2=exp(:,2);
x3=exp(:,3);
%
y=zeros(n,1);
for k=1:n
    y(k)=(1.25*x2(k)-x3(k)/1.5183)./(1+0.064*x1(k)+0.0378*x2(k)+0.1326*x3(k));
    y(k)=y(k)*normrnd(1, 0.02); % add measurement noises to the data
end
%
% Parameter estimation using nlinfit
%
alpha0=[1.2 0.1 0.1 0.1 1.5]'; % initial guess values
%
[alpha, residue, jacob]=nlinfit(exp, y, @fun, alpha0);
```

ex7_2_7.m

```
%  
% Calculate 95% confidence interval  
%  
ci=nlparci(alpha, residue, jacob);  
%  
% Results printing  
%  
fprintf('\n The estimated model parameters are:')  
for i=1:5  
    fprintf('\n alpha_%i = %7.3f +- %.3f', i, alpha(i), ci(i,2)-alpha(i))  
end  
%  
% Hougen-Watson model equation  
%  
function yhat=fun(alpha, x)  
a1=alpha(1);  
a2=alpha(2);  
a3=alpha(3);  
a4=alpha(4);
```

ex7_2_7.m

```
a5=alpha(5);  
x1=x(:,1);  
x2=x(:,2);  
x3=x(:,3);  
yhat=(a1*x2-x3/a5)./(1+a2*x1+a3*x2+a4*x3);  


---


```

Execution results:

```
>> ex7_2_7
```

The estimated model parameters are:

alpha_1 = 1.034 +- 0.746

alpha_2 = 0.051 +- 0.038

alpha_3 = 0.032 +- 0.025

alpha_4 = 0.104 +- 0.078

alpha_5 = 1.788 +- 1.303

7.4 Summary of the MATLAB commands related to this chapter

Command	Description
A\B	Calculate the linear least squared-error estimates, i.e., $(A^T A)^{-1} A^T B$
lsqnonlin	Solve nonlinear least-squares (nonlinear curve fitting) problems. This command requires the global declarations to import the experimental data into the user-specified subroutine for the calculation of model errors.
lsqcurvefit	Solve nonlinear least-squares (nonlinear curve fitting) problems. This command is more convenient to use because the experimental data can be transferred into the model file directly through the input arguments; no global declarations of the experimental data are required.
nlinfit	Solve the nonlinear regression (curve fitting) problems, which is able to provide the Jacobian matrix for the calculation of 95% confidence interval.
nlparci	Calculate the 95% confidence interval of the estimated parameter value.