
Process Control Modules

A Software Laboratory for Control Design

by

F. J. Doyle III

(fdoyle@udel.edu)

Dept. of Chemical Engineering
University of Delaware
Newark, DE 19716

with

E. P. Gatzke

R. S. Parker

ISBN 013-021107-9

to Diana and "the girls" (Sara and Brianna)

CONTENTS

PREFACE	xiii
Module Organization	xiii
Hardware and Software Requirements	xiv
Instructor's Manual	xiv
World Wide Web Page	xiv
Acknowledgments	xv
AUTHOR BIOGRAPHIES	xvii
1 INTRODUCTION TO THE PROCESS CONTROL MODULES	1
OBJECTIVE	1
INSTALLING PCM	1
START-UP MATLAB	2
WHAT IS MATLAB?	2
Example 1. Using MATLAB as a calculator.	3
Example 2. Using MATLAB for matrix calculations.	3
Example 3. Using MATLAB elementary math functions	3
SIMULINK	5
Glossary of Useful MATLAB Commands	7
INTRODUCTION TO THE PROCESS CONTROL MODULES	9
Overview of PCM	9
Summary Of Module Objectives	10
Ending Session	12
SUMMARY	13
2 STEADY STATE ANALYSIS	15
OBJECTIVE	15
	vii

Open-Ended Module Procedure	15
THINGS TO THINK ABOUT—FURNACE	15
THINGS TO THINK ABOUT—COLUMN	16
INTRODUCTION	17
PROCEDURE—FURNACE	17
Furnace Operation	18
Steady State Process Modeling—Furnace	23
Steady State System Gains	26
Manual Control of the Furnace	28
PROCEDURE—COLUMN	31
Column Operation	32
Steady State Process Modeling—Column	32
Steady State System Gains	38
Manual Control of the Column	39
SUMMARY	42
3 FIRST-ORDER DYNAMIC SYSTEM ANALYSIS	43
OBJECTIVE	43
Open-Ended Module Procedure	43
THINGS TO THINK ABOUT	43
INTRODUCTION	44
PROCEDURE	44
System Identification Problem	47
SUMMARY	50
4 SECOND-ORDER DYNAMIC SYSTEM ANALYSIS	51
OBJECTIVE	51
Open-Ended Module Procedure	51
THINGS TO THINK ABOUT	51
INTRODUCTION	52
PROCEDURE	53
System Identification Problem	55
SUMMARY	58
5 FREQUENCY DOMAIN ANALYSIS	59
OBJECTIVE	59
Open-Ended Module Procedure	59
THINGS TO THINK ABOUT	59

INTRODUCTION	60
PROCEDURE—BOTH UNITS	60
Frequency Response of a First-order System	61
General Pulse Testing Procedure	62
Frequency Response of Unknown Systems	62
PROCEDURE—FURNACE	63
PROCEDURE—COLUMN	64
SUMMARY	64
APPENDIX: plot_bode AND pcmfft COMMANDS	65
plot_bode	65
pcmfft	65
6 TRANSIENT RESPONSE ANALYSIS	67
OBJECTIVE	67
Open-Ended Module Procedure	67
THINGS TO THINK ABOUT	67
INTRODUCTION	68
PROCEDURE—FURNACE	68
Data Collection	68
Modeling the Furnace	70
Validation of Furnace Models	74
PROCEDURE—COLUMN	77
Data Collection	78
Modeling the Column	80
Validation of Column Models	82
SUMMARY	86
7 STEADY STATE FEEDBACK CONTROL	87
OBJECTIVE	87
Open-Ended Module Procedure	87
THINGS TO THINK ABOUT—BOTH UNITS	87
INTRODUCTION	88
PROCEDURE—FURNACE	89
Proportional Control	89
Proportional Control for a Disturbance	90
Proportional-Integral (PI) Control	91
Proportional-Integral (PI) Control for a Disturbance	92
PROCEDURE—COLUMN	93

Proportional Control	93
Proportional Control for a Disturbance	94
Proportional-Integral (PI) Control	95
Proportional-Integral (PI) Control for a Disturbance	97
SUMMARY	98
8 CONTROLLER TUNING	99
OBJECTIVE	99
Open-Ended Module Procedure	99
THINGS TO THINK ABOUT	99
INTRODUCTION	100
PROCEDURE—FURNACE	101
Closed-Loop Auto Relay Tuning Method	103
PROCEDURE—COLUMN	104
Closed-Loop Auto Relay Tuning Method	105
SUMMARY	106
9 FEEDFORWARD CONTROL	107
OBJECTIVE	107
Open-Ended Module Procedure	107
THINGS TO THINK ABOUT	107
INTRODUCTION	108
PROCEDURE—FURNACE	109
PROCEDURE—COLUMN	111
SUMMARY	114
10 IMC AND MULTIVARIABLE CONTROL	115
OBJECTIVE	115
Open-Ended Module Procedure	115
THINGS TO THINK ABOUT	115
INTRODUCTION	116
Internal Model Control (IMC)	116
Decoupling Control	117
PROCEDURE—FURNACE	117
Decoupling	119
PROCEDURE—COLUMN	120
Decoupling	121
SUMMARY	122

11 DISCRETE TIME SYSTEM MODELING	123
OBJECTIVE	123
Open-Ended Module Procedure	123
THINGS TO THINK ABOUT	123
INTRODUCTION	124
PROCEDURE	125
SUMMARY	128
12 DISCRETE CONTROL	129
OBJECTIVE	129
Open-Ended Module Procedure	129
THINGS TO THINK ABOUT	129
INTRODUCTION	130
PROCEDURE—FURNACE	131
Discrete Internal Model Control	133
PROCEDURE—COLUMN	135
Discrete Internal Model Control	137
SUMMARY	138
13 MODEL PREDICTIVE CONTROL	139
OBJECTIVE	139
Open-Ended Module Procedure	139
THINGS TO THINK ABOUT	139
INTRODUCTION	140
PROCEDURE - FURNACE	141
PROCEDURE—COLUMN	144
SUMMARY	146
ABOUT THE CD	147

PREFACE

The Process Control Modules (PCM) have been developed to address the key engineering educational challenge of realistic problem solving within the constraints of a typical lecture-style course in process dynamics and control.

The primary objectives in developing these modules were:

- Develop realistic computer simulation case studies that exhibited nonlinear, high order dynamic behavior.
- Develop convenient graphical interface for students that allowed them to interact in real-time with the evolving virtual experiment.
- Develop a set of challenging exercises that reinforce the conventional lecture material through active learning and problem-based methods.

Module Organization

This book is organized into 13 modules or chapters, around two distinct unit operations—a fired tube furnace and a binary distillation column. Most of the 13 modules contain separate exercises for the different units. The progression of modules follows a typical undergraduate process control textbook, starting with low-order dynamic system analysis and continuing through digital and multivariable control synthesis.

A typical module is structurally arranged with the following elements:

- **Objective** of the specific Module, including an open-ended formulation of the Module for the self-initiated users.
- **Things to Think About** exercises which are meant to be accomplished before the lab Module, and should serve as a review of the relevant material for that Module.
- **Introduction** to the specific Module.

- **Procedure** section consisting of individual steps to execute during the virtual experiment.
- **Exercises** which correspond to the steps in the **Procedure**.
- **Summary** of key learning objectives.

In a typical one semester course at the University of Delaware, we will cover all the exercises for one of the unit operations (furnace or column) in all 13 of the modules. We have used the modules as both a self-paced take-home exercise, and as the basis of a required 2 hour computational laboratory. Individual modules have also been used as the basis of short course material for industrial engineers. We anticipate that this textbook would have considerable value for the practicing process engineer looking to refresh control fundamentals.

Hardware and Software Requirements

The Process Control Modules are a set of MATLAB/SIMULINK routines which require either a full license or the Student Version of MATLAB and SIMULINK. The current version of the modules has been tested with versions 5.3 of MATLAB and 3.0 of SIMULINK, and they are compatible with older versions of MATLAB back to version 5.1, and SIMULINK back to version 2.0.

The standard release of PCM on the CD-ROM shipped with this text contains executable code for the PC (NT 4.0, Win95) and UNIX operating (Solaris) systems.

The minimum recommended system configuration is a 200 MHz Pentium Pro with 32 MB RAM (or equivalent UNIX machine). For advanced users who may wish to customize some of the source code for the unit operations (written in the C language), it will be necessary to have a MATLAB compatible compiler (such as WATCOM for the PC). More information about compilers is available on the MATLAB WWW page.

Instructor's Manual

An instructor's manual has been prepared and is available from the publisher.

World Wide Web Page

A WWW page has been setup to provide up-to-date information about the Process Control Modules, including new unit operations. The URL is:
<http://www.che.udel.edu/pcm/>

Acknowledgments

There are a number of individuals that must be acknowledged for their contributions to PCM. First, there has been a succession of “lead TAs” for my undergraduate process dynamics and control courses at Purdue and Delaware. In chronological order they include Tom Kendi, Bob Parker, Ed Gatzke, and Radhakrishnan Mahadevan. Over the course of the past seven years, a multitude of graduate and undergraduate programmers have contributed to various components of PCM—their contributions and names are numerous, and I would like to express my gratitude to all of them. It was their enthusiasm that helped sustain this effort. In a similar vein, the students that I have taught through the years in CHE 456 at Purdue and CHEG 401 at the University of Delaware have provided inspiration for this activity through their intellectual curiosity and motivation to apply their textbook understanding. I would also like to thank a number of colleagues at other universities for their valuable feedback in evaluating the modules, including Mike Henson at LSU, Yaman Arkun at Georgia Tech, and Tunde Ogunnaike at DuPont/University of Delaware. I would like to express my appreciation to my former department head, Rex Reklaitis, who encouraged the pursuit of these educational activities while I was an assistant professor at Purdue University. Finally, I would like to acknowledge partial funding for this project from the National Science Foundation via the National Young Investigator program (Grant CTS 9257059).

A brief historical note on the origins of these Modules may be of interest to some users and offers further acknowledgments to the important work of other educators. During the summer of 1982, as part of a special study with IBM, the IBM Advanced Control System (ACS) was installed at Purdue University. In 1982–83, Professor L. B. Koppel wrote a set of ACS-based instructional modules which made use of a process furnace simulation supplied by IBM (known as the Purdue Furnace). In 1987–88 Professor R.P. Andres and Professor K.P. Madhavan updated and modified the existing ACS system. This system was replaced with Hewlett-Packard workstations in 1993. Professor F.J. Doyle undertook the rewriting of the original Purdue Furnace model using the Mathworks products MATLAB and SIMULINK as the basic platform. In 1994, the binary distillation column unit was added to the Modules. In 1996, the software was ported to the Macintosh and PC platforms.

Francis J. Doyle III
April 14, 1999

AUTHOR BIOGRAPHIES



Francis J. Doyle III was born in Philadelphia, PA, in 1963. He received the B.S.E. degree from Princeton University, Princeton, NJ, in 1985, the C.P.G.S. degree from Cambridge University, Cambridge, U.K., in 1986, and the Ph.D. Degree from the California Institute of Technology, Pasadena, in 1991, all in chemical engineering.

From 1991 to 1992, he was a visiting scientist in the strategic process technology group at the DuPont Company, Wilmington, DE. From 1992-1997 he was an assistant and associate professor at the School of Chemical Engineering at Purdue University, West Lafayette, IN. Since 1997, he has been an associate professor at the Department of Chemical Engineering at the University of Delaware, Newark, DE. His research interests include nonlinear dynamics and control with applications in process and biosystems control, nonlinear model reduction, and the reverse engineering of biological control systems.

Dr. Doyle received the National Young Investigator Award from the National Science Foundation in 1992, an Office of Naval Research Young Investigator Award in 1996, an ASEE Section Outstanding Teacher Award in 1996, and a Tau Beta Pi Section Teaching Award in 1996. He was named a Fellow of the University of Delaware Institute for Transforming Undergraduate Education in 1998, and was elected as an Academic Fellow of CACHE (Computer Aids for Chemical Engineers) in 1999.



Edward P. Gatzke received his BChE from the Georgia Institute of Technology in 1995. After two years of graduate study at Purdue University, he moved to the University of Delaware Chemical Engineering Department for completion of his PhD. His research interests include process control, optimization, and artificial intelligence. Ed's industrial experience includes internships with Honeywell Technology Center, Mead Paper, and Tele-dyne Brown Engineering. His home town is Huntsville, Alabama.



Robert S. Parker was born in Rochester, NY, in 1972. He received the B.S. degree in Chemical Engineering from the University of Rochester, Rochester, NY, in 1994, and the Ph.D. degree in Chemical Engineering from the University of Delaware, Newark, DE, in 1999.

Since 2000, he has been an assistant professor in the Department of Chemical and Petroleum Engineering at the University of Pittsburgh. Dr. Parker's research interests include model identification and model-based control with a focus on biomedical and biotechnological systems applications.

Module 1

INTRODUCTION TO THE PROCESS CONTROL MODULES

OBJECTIVE

The purpose of this Module is to introduce MATLAB and SIMULINK, as well as explain the basic operation of the Process Control Modules (PCM).

INSTALLING PCM

You may run PCM from the CD-ROM. For better performance, it is suggested that you copy the *pcm* folder to a local hard disk.

To start PCM you must either currently be in the *pcm* directory or have the *pcm* directory added to your MATLAB path. From MATLAB, you can use the *pwd* to determine your current working directory. The *cd* command can be used to change directories. MATLAB comes with a path browser that allows you to view and change your path. You can also manually change the path by including an appropriate *addpath* command in your *matlabrc.m* file that specifies the full path to the *pcm* directory.

The column and furnace simulations used by PCM rely on process models. MATLAB can use interpreted models or compiled models. In our experience, compiled models run much faster. We have provided compiled models for MATLAB version 5.3 on PC and Solaris systems. You may have trouble using the provided compiled models. If you have access to a C compiler that is MATLAB compatible, you can use the *mex* command to compile the models for your system or for newer/older versions of MATLAB. The files that need compiling are: */pcm/furnace/pufcore4.c*, */pcm/column/newcolco.c*, and */pcm/column/newcol.c*. For the furnace, the result-

ing compiled file should be named *pufcore4.dll* for PC versions or *pufcore4.mexsol* for Solaris.

For PC users, you may need to copy the files *mths110.dll* and *plbs110.dll* to your system directory before starting MATLAB. Windows 95/98 users copy to the directory C:\Windows\system and NT users copy to the directory C:\WINNT\system32. The files are on the CD-ROM and may be hidden from view because they have a .dll extension. If you cannot view these files on the CD-ROM, select *View, Options, View*, and *Show all files*.

If you cannot use 'mex' to compile these files and you have trouble running the provided compiled simulations, you can use the slower interpreted version of the furnace model. Remove or rename the compiled versions of the furnace model so that MATLAB will use the file *pufcore4.m* as the model. MATLAB looks for a compiled version before using the interpreted version, so this will force the use of the interpreted version.

START-UP MATLAB

1. You should have a recent distribution of MATLAB (version 5.1 or greater) installed on your computer.
2. For UNIX systems, you should type *matlab* at a command prompt in a terminal window.
3. For DOS systems (Win 95/NT) use the mouse to select the *Start* button. Holding the left button, highlight the *Programs* menu. Another group of menus will appear. Highlight the *MATLAB* selection. This opens the MATLAB menu. Highlight the *MATLAB* command to start MATLAB.

WHAT IS MATLAB?

MATLAB is a software package which enables many of the calculations associated with control systems, including matrix computations. There are many specialized built-in functions for such things as eigenvalue computation and Bode plot analysis.

First, we will try some simple examples in MATLAB. These are meant to introduce the MATLAB interface. A full understanding of MATLAB's many utilities is far beyond the scope of this Module. MATLAB is a powerful tool which is easy to learn. You may find it useful to use many of the features of MATLAB to solve the homework problems assigned in your control class. Therefore, you are encouraged to learn as much as possible about MATLAB.

Example 1. Using MATLAB as a calculator.

At the MATLAB prompt try typing $1234 + 5678$. You should get:

```
>> 1234 + 5678 (hit enter)
ans =
    6912
```

Now try $4/5 * 10 - 3$. You should get:

```
>> 4/5*10 - 3 (hit enter)
ans =
     5
```

However, these are just scalar calculations. MATLAB was developed with the ability to easily manipulate matrices.

Example 2. Using MATLAB for matrix calculations.

Enter the example below.

```
>> a = [1 2;3 4] (hit enter)
a =
     1     2
     3     4
>> b = [5 6;7 8] (hit enter)
b =
     5     6
     7     8
>> c = a*b (hit enter)
ans =
    19    22
    43    50
```

To find the transpose of c , type c'

```
>> c' (hit enter)
ans =
    19    43
    22    50
```

Example 3. Using MATLAB elementary math functions

MATLAB also has several elementary math functions. For example:

Trigonometric functions

```
>> cos(2) (hit enter)
ans =
    -0.4161
>> sin(pi) (hit enter)
ans =
    1.2246e-16
```

Exponential functions

```
>> exp(1) (hit enter)
ans =
    2.7183
>> log10(2) (hit enter)
ans =
    0.301
>> log(2) (hit enter)
ans=
    0.6931
```

MATLAB contains many math functions, both elementary and advanced. To find more information on these functions, type *help* at the MATLAB prompt. This will give a list of help topics along with a description of the functions contained in each of the topics. Each of the topics can be further examined by typing the topic name after the help command. For example, typing *help elfun* gives more help information on MATLAB's elementary math functions.

Try the following examples in succession:

```
linspace(0, 1, 5)
logspace(0, 1, 5)
x=[1 2 3]
y=[3; 2; 1]
x + y
y + x
y1 = y'
polyfit(x, y1, 1)
```

To further explore options in MATLAB, type *demo* at the MATLAB prompt and hit enter. Select *Continue* from the menu options of the MATLAB EXPO and spend about 15 minutes investigating the demonstrations. When you are finished, select the *Close All Windows* button from the menu.

SIMULINK

SIMULINK is an interactive programming environment within MATLAB for performing dynamic simulation. Complex dynamic simulations can be performed by combining simple operations. Many of these functions are available from a precoded block library within SIMULINK. In the following exercise, a simple SIMULINK simulation will be constructed using simple functions from the block library.

1. At the MATLAB prompt, type *simulink*. The SIMULINK block library should appear.
2. At the top of the SIMULINK block library are several (six in most versions of SIMULINK) pull-down menus (labeled *File*, *Edit*, *Options*, ...). To open an empty window for creating a new simulation, click on the *File* menu and select *New*. This should produce an empty SIMULINK window.
3. In the SIMULINK block library window, double-click on the *Sources* icon. The Signal Source Library window should appear. An icon from this library can be incorporated into your simulation by placing the mouse arrow on an icon and dragging the icon from the library window into the empty SIMULINK window while holding down the left mouse button. Grab several of the icons from the Signal Source Library and put them into your empty SIMULINK window.
4. The SIMULINK window can be saved by using the *File* pull-down menu. Make sure the you save your own window (not the block library window). Click on *File* with the left mouse button, move your mouse arrow down to *Save* and click once with the left mouse button. You will be prompted to name your file which will be saved with a *.mdl* extension.
5. After saving the file, exit by using the *Exit* option on the *File* pull-down menu. Restart the simulation by typing the filename (without the *.mdl* extension) at the MATLAB prompt.
6. Within your SIMULINK window, an icon can be moved by placing your mouse arrow on the icon and dragging while the left mouse button is held down. An icon can be selected by clicking the left mouse button ONCE on the icon. When selected, an icon can be deleted by using the *backspace* key on the keyboard. Additional operations can be performed using several of the pull-down menus. Familiarize yourself with the operations in the *Edit* pull-down menu.
7. Building a SIMULINK simulation:
You will need the following icons: [icon (quantity)]
(a) Step Input (2)

- (b) Clock (1)
 - (c) Sum (1)
 - (d) Gain (2)
 - (e) To Workspace (2)
8. Construct a flow sheet such that the output of each of the *Step Input* blocks is multiplied by one of the *Gain* blocks. The output of each of the *Gain* blocks is summed using the *Sum* icon. The output of the *Sum* icon is assigned to a variable name (*out*, for instance) and sent to the MATLAB workspace using one of the *To Workspace* icons. For a dynamic simulation, the output is plotted with respect to time. Connect the *Clock* icon to the remaining *To Workspace* icon and assign a variable name of *t* or *time*. Note that if using MATLAB 5.3 or greater, you should double-click on each *To Workspace block* and change the save format to Matrix. Save your SIMULINK diagram using the *Save* option on the *File* pull-down menu.
9. Before running the simulation, check the numerical integration parameters and the simulation run time. The parameters can be viewed by clicking once with the left mouse button on the *Simulation* pull-down menu and clicking on *Parameters*. For this exercise, the default numerical integration parameters should suffice; however, change the *Stop Time* to 100. Start the simulation by clicking on the *Start* option on the *Simulation* pull-down menu. When the simulation is complete, move the mouse arrow into the MATLAB window and type *who*. This command displays the variables that are in the MATLAB workspace. This should include the variable names for the two outputs of the SIMULINK simulation (the time, and the output of the sum). In the MATLAB workspace, make a plot of the data from the SIMULINK simulation using the *plot* command. For details on the usage of the *plot* command, type *help plot* at the MATLAB prompt or check the Glossary of Useful Matlab Commands located at the end of this section. The plots should be labeled (using the *xlabel* and *ylabel* commands in MATLAB) and have a title (using the *title* command in MATLAB).
10. Experiment with different sets of parameters for the *Step Input* and *Gain* blocks. To change the parameters, double-click (using the left mouse button) on the icon you would like to change.
11. To create your own SIMULINK block library, select *New* from the *File* menu in the SIMULINK Library Window then select *Library*. This is your opportunity to create a personalized set of blocks for use throughout the course. By pressing the left mouse button, blocks from the different SIMULINK sub-menus can be highlighted. By holding the left button down, blocks can be dragged from the sub-menus into your personalized window. Double-click on the *Sources* menu and drag the blocks from Step 7 (above) into your window.

Now, from the *Linear* menu move the blocks (again from Step 7) to your library. Add the blocks needed from the Sinks menu for Step 7 to your set of blocks. In your personalized window select *File* and click on *Save*. Name this block set *simblocks.mdl*. As the course progresses, you can add more blocks to this library to assist with homework.

Glossary of Useful MATLAB Commands

1. **bode** - The bode command is used to obtain the frequency response information of a system.
Usage: $[mag, phase, w] = bode(num, den)$ calculates magnitude and phase for a given transfer function whose coefficients are given by *num* (numerator) and *den* (denominator) or alternatively, $bode(num, den)$ - which directly plots the magnitude and phase over the frequencies *w* to produce the Bode plot.
2. **eig** - finds the eigenvalues of a matrix.
Usage: $eig(A)$ produces a vector containing the eigenvalues of *A*.
3. **element-by-element** math functions can be used on vectors of the same size.
Usage: $a.*b$ produces a new vector where the *i*th element of the new vector is the product of the *i*th element in *a* and the *i*th element in *b*.
4. **gtext** - puts text onto a plot.
Usage: $gtext('text')$ brings crosshairs onto the current figure. Click on a position in the plot to place the string *text* on the current plot.
5. **inv** - calculates the inverse of a matrix.
Usage: $inv(A)$ produces the inverse of a square matrix *A*.
6. **linspace** - creates a vector of linearly spaced points.
Usage: $linspace(x1, x2, n)$ creates a vector of *n* points linearly spaced between *x1* and *x2*, inclusive.
7. **logspace** - generates a logarithmically spaced vector of points over a given range.
Usage: $m = logspace(p1, p2, n)$ generates a vector, *m*, containing *n* equally logarithmically spaced points between the values 10^{p1} and 10^{p2} .
8. **.m file** - a file consisting of a sequence of MATLAB commands that can be executed from the MATLAB prompt.
9. **.mdl file** - a file consisting of a SIMULINK model or a SIMULINK block library.

10. **plot** - command to generate X-Y plots.

Usage: *plot(x,y,'L')* generates the X-Y plot for the given data sets, *x* and *y* with line type *L*

Available line types: replace 'L' with:

solid '-'

dashed '--'

dash-dot '-.'

dotted ':'

plot(t,x1,'-',t,x2,':') plots *x1* and *x2* versus time on the same graph. The *t-x1* graph has a dashed line; while the *t-x2* plot has a dotted line. The latest version of MATLAB incorporates a number of graphics editing features directly into a toolbar at the top of the plot window.

11. **polyfit** - fits a polynomial to a set of data.

Usage: *polyfit(x,y,n)* given a set of independent data *x*, dependent data *y*, and order of polynomial *n*, a vector of coefficients (highest order to the left) is returned. This polynomial represents a least-squares fit to the data.

12. **print** - prints a plot to a printer.

Usage: *print -dpSC name.ps* produces a file named *name.ps* containing the plot in a color postscript format.

print -Swindowname prints the SIMULINK window given by *windowname*.

print -Pprntername prints a plot to printer *prntername*.

13. **roots** - solves for the roots of a polynomial function.

Usage: *roots(poly)* solves for the roots of the function *poly*. For example: given the function $10x^3 + 8x^2 + 6x + 3$, the values of *x* for which the above function equals zero can be obtained as follows:

fun = [10 8 6 3] sets the variable *fun* to the coefficients of the polynomial in descending powers of *x*

roots(fun) determines the zero solutions of the function

14. **series** - performs the multiplication of two proper functions.

Usage: *[num, den] = series(num1,den1,num2,den2)* - performs the following operation:

$$\frac{num}{den} = \frac{num1}{den1} * \frac{num2}{den2}$$

15. **simulink** - command to start up SIMULINK, an object oriented software package for simulating dynamic processes.

16. **title** - puts a title on a plot.

Usage: `title('Plot Title')` puts the title *Plot Title* on the current plot.

17. **xlabel/ylabel** - puts a title on the X or Y axis of the current plot.

Usage: `xlabel('label')` puts the title *label* on the horizontal axis of the plot.

INTRODUCTION TO THE PROCESS CONTROL MODULES

Overview of PCM

The units of PCM are organized around the following three elements:

1. Process Modeling
2. PID Controller Design and Tuning
3. Analysis and Advanced Control Design

Throughout the course, experiments involving each of these elements will be performed on the furnace or binary distillation column. Since all of these elements interact, the final design may involve several iterations. The organization of the manual is as follows.

Module 2 through Module 6 focus on steady state and dynamic modeling. In Module 2, a steady state model of the process is obtained. A steady state model is used to predict the effect that an input variable has on the final steady state value of an output variable. In the case of the furnace, one output variable of interest is the temperature of a hydrocarbon stream that is being heated to a desired value. For example, a steady state model can predict the final temperature of the hydrocarbon if the fuel gas flow into the furnace is changed by ten percent. A major shortcoming of steady state models is that they cannot predict the *transients* which occur as a result of changing an input variable. The transient behavior of a process is the time dependent trajectory of an output in response to a particular input or class of input signals. Chemical process systems typically do not reach a steady state instantaneously, thus accurate modeling of the system captures both the steady state and the transient behavior of the system. One way to represent a linear dynamic process model is in the form of a transfer function model. Modules 3 and 4 focus on two common classes of transfer function models: first-order and second-order models. The purpose of these Modules is to demonstrate the effects of the various parameters in these models on the process response, and to obtain the parameters for a set of input-output data. Frequency response models of the furnace or column are generated from pulse tests in Module 5. Transfer function models will be obtained for the furnace or column in Module 6. In Module 11, discrete-time models will be developed for subsequent use in digital controller design.

The second element in the synthesis of a control system is the control structure selection and controller tuning. There are many approaches to controller design. In Module 7 and Module 8, the following three approaches will be applied to PID controller design for the furnace or distillation column: (1) Ziegler-Nichols tuning rules; (2) Cohen-Coon tuning rules; and (3) Åström's relay-tuning method. The focus of Module 5 is the calculation of the *frequency response* of the process. The frequency response is obtained from the data generated by a pulse test using a Fourier transform to calculate two specific properties of the process response: amplitude and phase shift. Given the frequency response for a process, tuning parameters for the controllers can be obtained using the Ziegler-Nichols tuning rules. A second method of controller design is known as *model-based control*. In Module 8, controller tuning parameters are obtained using the Cohen-Coon tuning rules which recommend tuning constants based upon the parameters of the first-order transfer function model of the furnace or column obtained in Module 6. The performance of the resultant controller is dependent upon the accuracy of the model obtained previously, a clear illustration of the interaction between modeling and controller design. Finally in Module 8, a controller is designed using Åström's auto-tuning method. Students will learn that each of the techniques results in a different controller and, subsequently, different closed-loop performance.

The third element in the design of a control system is the analysis of the performance. Advanced topics in control design will also be explored, including combined feedforward/feedback (Module 9), multivariable and Internal Model Control (IMC) design (Module 10), digital design techniques (Module 12), and Model Predictive Control (MPC) (Module 13).

There are several ways of characterizing performance, which will be explored in the later Modules. The interaction of modeling, design and analysis for control system design will be demonstrated.

Summary Of Module Objectives

Process Modeling	PID Design and Tuning	Analysis and Advanced Design
Module 2	Module 7	Module 5
Module 3	Module 8	Module 6
Module 4	Module 9	Module 9
Module 5	Module 10	Module 10
Module 6	Module 12	Module 12
Module 11		Module 13

The Process Control Modules (PCM) use programs that were developed to be used with SIMULINK and MATLAB. To start the Modules, type *mainmenu*. The display in Figure 1.1 will appear on the screen. From this menu you can access

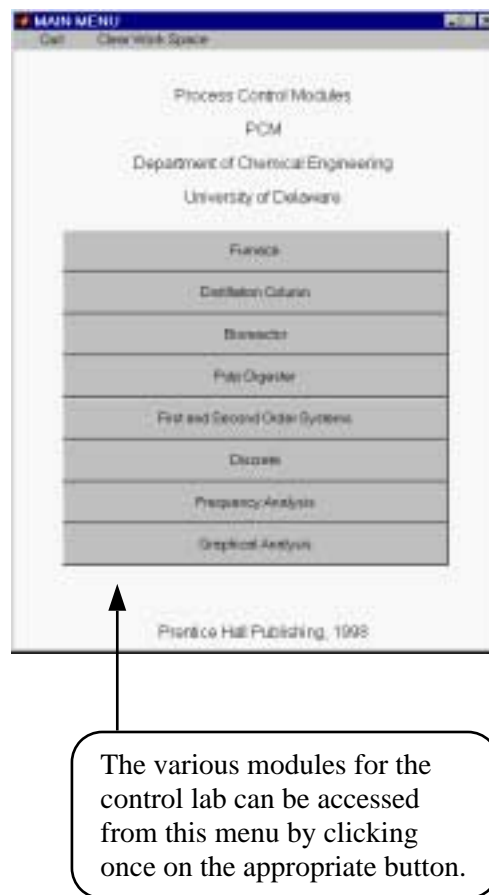


Figure 1.1. Main Menu for Process Control Modules

the various programs that will be used in the Process Control Modules. In the next two Modules, we will use the *Furnace* and the *Distillation Column* models to demonstrate basic principles in process behavior. If one selects the *Furnace* button, then the furnace menu shown in Figure 1.2 will appear on the screen. Clicking on the *Distillation Column* button will cause Figure 1.3 to appear on the screen.



Figure 1.2. Furnace Menu

The PCM units have exercises that can be used for evaluation purposes. The exercises may ask you to answer a question, print a plot, or create a table. The exercises are listed in bold. Exercises in the THINGS TO THINK ABOUT section of each Module are listed with letters (*e.g.*, **Exercise 1-A**). These should be completed before attempting to work the laboratory Module. Exercises in the procedure section are numbered (*e.g.*, **Exercise 1.1**) and should be completed while working through the laboratory Module.

Ending Session

To end the session select *Yes* under the *Quit* menu from the Main Menu. This will take you back to the MATLAB prompt. At this prompt, type *quit* to exit MATLAB.



Figure 1.3. Distillation Column Menu

SUMMARY

By the end of this Module, you should have learned how to log in at the computer. You have been introduced to MATLAB and should be able to start MATLAB and know how to run the demonstration. You should also be able to start the Process Control Modules.

Module 2

STEADY STATE ANALYSIS

OBJECTIVE

This module serves as an introduction to the simulated unit operations used in PCM, the Furnace and the Column. Adjusting input values for a system results in a change in the output levels. Various steady state values will be used to calculate the process gain for input-output relationships of the furnace or column model.

Open-Ended Module Procedure

Calculate the steady state gains for the furnace/column, filling in all the gains in Figure 2.7/2.12. Use these gains to demonstrate “manual control” in which you attempt to calculate the appropriate manipulated variable changes required to bring about an output change of a desired magnitude, as well as cancel the effect of a step change in one of the process disturbances for the furnace/column.

THINGS TO THINK ABOUT—FURNACE

Using the information in the furnace flow diagram (Figure 2.2) and the following data, perform an energy balance on the furnace (assume atmospheric pressure for the streams and use the ideal gas law for density):

heat capacity of fuel (inlet)	$29.5 \frac{J}{mol K}$
heat capacity of fuel (exit)	$49 \frac{J}{mol K}$
heat of reaction for combustion	$889 \frac{kJ}{mol CH_4}$
density of hydrocarbon	$787 \frac{kg}{m^3}$
hydrocarbon exit temperature	$605 K$
heat capacity of hydrocarbon stream	$463 \frac{J}{kg K}$
stack gas temperature	$1410 K$
volumetric flow rate of stack gas	$43.95 \frac{m^3}{min}$

The following exercises will lead you through the appropriate calculations for an overall energy balance.

Exercise 2-A What is the contribution from the combustion of fuel gas in the furnace?

Exercise 2-B What are the enthalpies of the streams (hydrocarbon, air, and fuel) entering the furnace?

Exercise 2-C What is the enthalpy of the hydrocarbon process stream leaving the furnace?

Exercise 2-D What is the enthalpy of the stack gas stream?

Exercise 2-E Using the information from the previous exercises, attempt to close the energy balance. Indicate, if appropriate, the imbalance and explain the difference.

THINGS TO THINK ABOUT—COLUMN

Exercise 2-A In order to control the composition of the distillate and the bottom streams, which variables can be used as manipulated variables?

Exercise 2-B Which variables act as load (disturbance) variables?

Exercise 2-C What will happen if the reflux ratio is increased? How will this increase affect the mole fraction of methanol in the distillate and the bottom?

Exercise 2-D A binary mixture of components with relative volatility 2 is to be separated in a distillation column. Consider a mass balance over a tray in the rectifying section (above the feed). The reflux ratio is 3.1667. The vapor enters the tray at 0.5 mol/sec with 60% purity in the more volatile component. The liquid enters the tray at 0.38 mol/sec with 80% purity in the more volatile component. Calculate the composition of the streams leaving the tray. (Assume that the vapor and liquid flows in the rectifying section do not change from tray to tray.)

INTRODUCTION

In this section, you will obtain steady state models for a process system to determine the effect of manipulated and load variables on the controlled variables of a process. This data is useful for approximating the changes in the manipulated variables necessary to keep the controlled variables at their desired setpoints.

The steady state gain of a system characterizes the effect that a change in an input variable has upon an output variable. The gain is mathematically described as follows:

$$K = \frac{\text{Change in Output}}{\text{Change in Input}}$$

It is important to note that the gain is defined on the basis of the incremental change in the respective variables. An implicit assumption in such a calculation is that changes in the controlled variables can be calculated by summing the changes in all the manipulated and load variables multiplied by constant gain coefficients. This is known as the principle of superposition and is strictly valid only for **linear** processes. The furnace and column are nonlinear systems; consequently, this approximation is valid only over a small region around the operating point where the system behaves in a nearly linear manner.

The steady state gains can be used for many purposes, some of which are listed below:

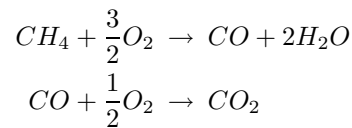
- (a) to determine the value of a manipulated variable necessary to change the setpoint of a controlled variable;
- (b) to predict the effect of a change in a load variable upon a controlled variable;
- and
- (c) to predict the change in a manipulated variable necessary to counteract a load variable change.

PROCEDURE—FURNACE

The unit operation in this module represents a furnace fueled by natural gas which is used to preheat a high molecular weight hydrocarbon feed (C_{16} - C_{26}) to a cracking unit at a petroleum refinery (see schematic in Figure 2.1). The furnace model consists of energy and component mass balances which result in coupled nonlinear differential equations. The furnace model has seven inputs and four outputs as listed below.

INPUTS	OUTPUTS
Hydrocarbon Flow Rate	Hydrocarbon Outlet Temperature
Hydrocarbon Inlet Temperature	Furnace Temperature
Air Flow Rate	Exhaust Gas Flow Rate
Air Temperature	Oxygen Exit Concentration
Fuel Gas Flow Rate	
Fuel Gas Temperature	
Fuel Gas Purity	

The combustion of the fuel is assumed to occur via the following reaction mechanism:



There are two major objectives for operation of the furnace. First, in order to minimize fuel costs, the furnace must be operated with proper oxygen composition to ensure complete combustion of the fuel (carbon monoxide is an undesired product). Second, the hydrocarbon feed stream must be delivered to the cracking unit at the desired temperature.

The furnace has the following manipulated and controlled variables:

Manipulated Variables	Controlled Variables
Air Flow Rate	Oxygen Exit Concentration
Fuel Flow Rate	Hydrocarbon Outlet Temperature

The system also has the following load (or disturbance) variables:

Load Variables
Hydrocarbon Flow Rate
Fuel Gas Purity

Furnace Operation

1. Start by selecting the furnace from the Main Menu (see Figure 1.1). This is done by clicking the left mouse button once on the *Furnace* button. This opens the menu window for the furnace modules. Click the left mouse button on the *Furnace* button. Two additional windows should open, one for the input

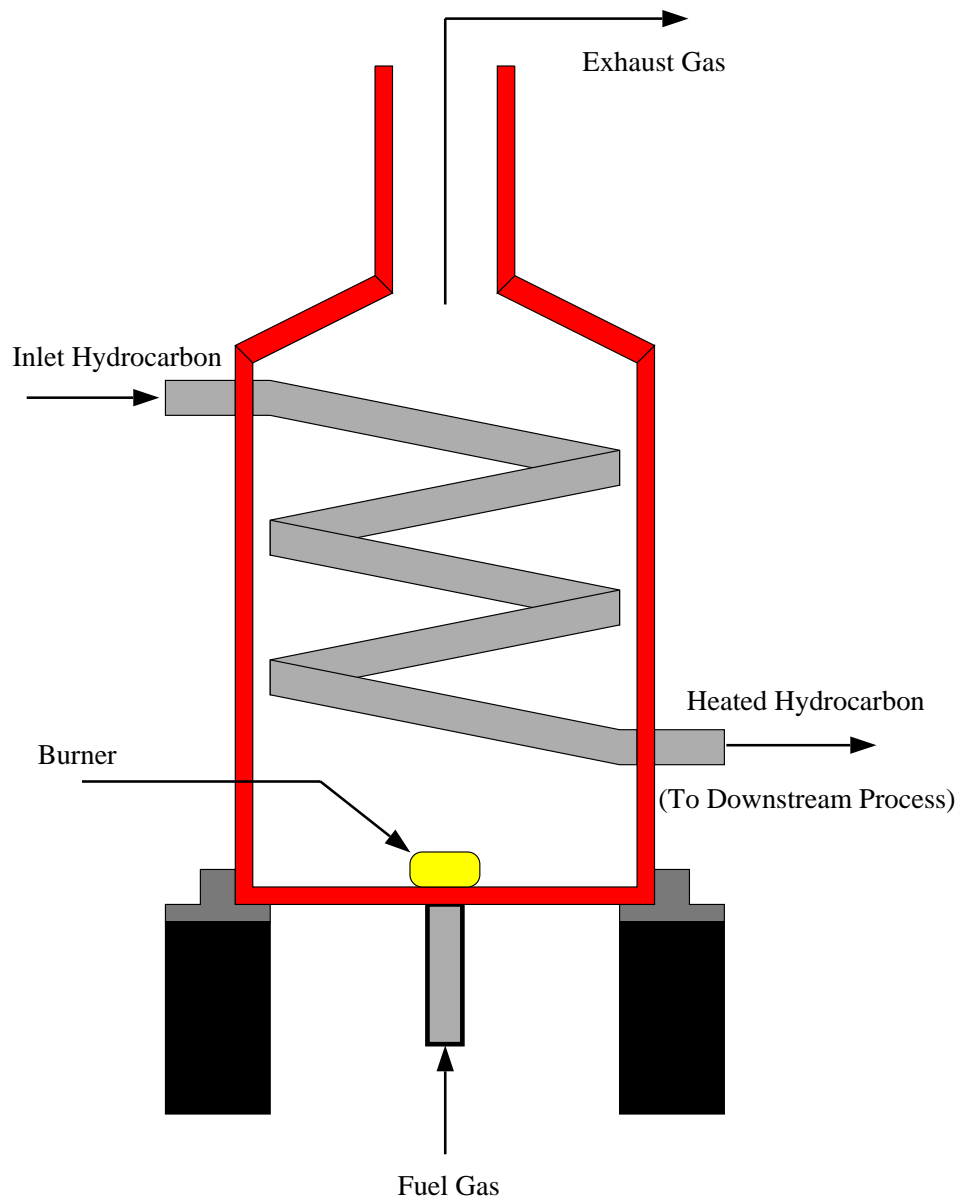


Figure 2.1. Process Schematic of the Furnace

and output graphs and one for the furnace process flowsheet. The flowsheet is shown in Figure 2.2.

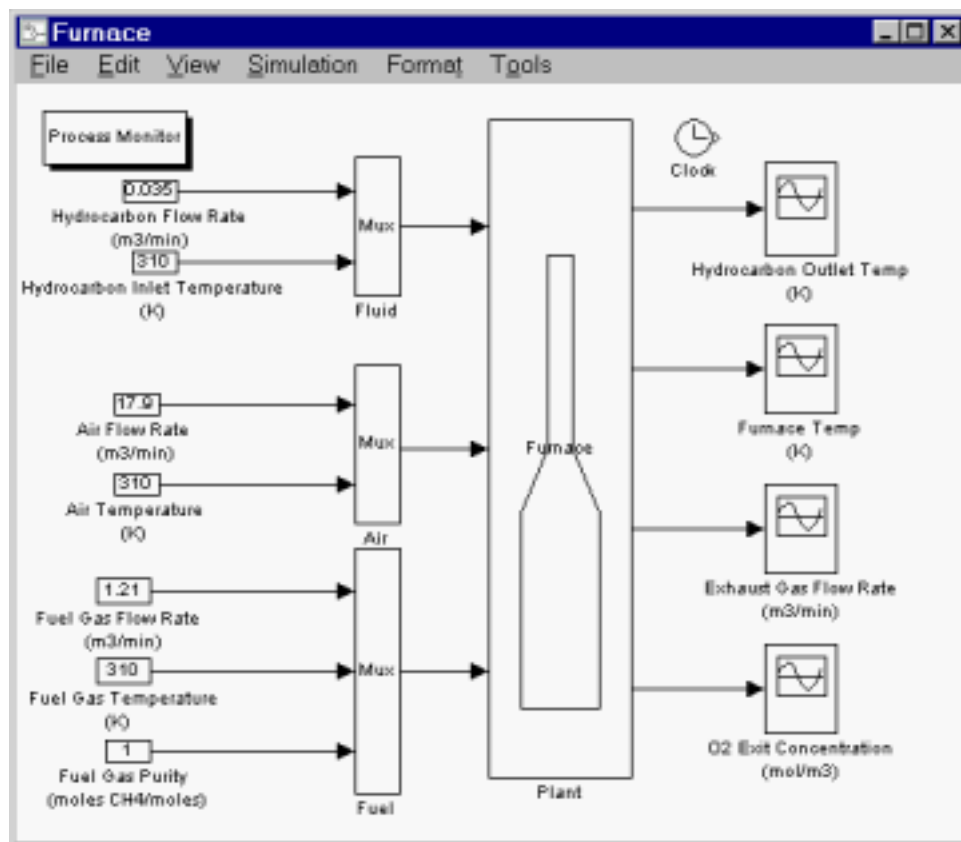
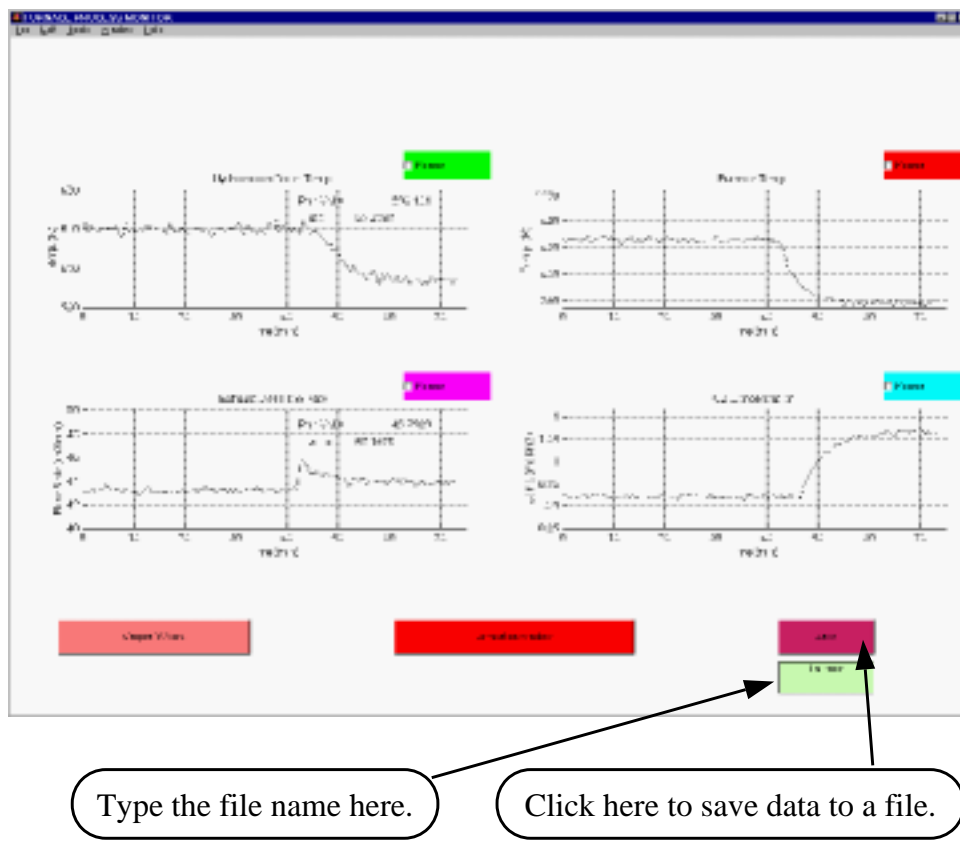


Figure 2.2. Operator Interface for the Furnace

2. Using the operating parameters given initially, we will begin by running a simulation. Under the *Simulation* menu, select *Start*. This command should be executed once during a lab session. It is the simulated equivalent to a perfect process startup. The process output graphs are located on the window labeled FURNACE PROCESS MONITOR (see Figure 2.3). Notice how the outputs remain unchanged with time.
3. Next, try decreasing the fuel gas purity. This will act as a disturbance to the system. By double-clicking on the *Fuel Gas Purity* box (see Figure 2.2) you should get a box as shown in Figure 2.4. Change the value from 1.0 to 0.95 by clicking on the value box and using the backspace key to erase the old value.

**Figure 2.3.** Furnace Process Monitor

When you have entered a new value, click on the *Close* button. Again, notice how the outputs on the process monitor are changing with time.

4. Now, change some of the operating parameters. First, turn off the disturbance in the gas purity by double-clicking on the *Fuel Gas Purity* box and return the value to 1.0. Allow the system to reach a new steady state. Now we will increase the product flow rate by 10%. To do this, double-click with the left mouse button on the *Hydrocarbon Flow Rate* box (see Figure 2.2). A box containing the product flow rate value should appear (see Figure 2.5). Change this value from 0.035 to 0.0385 and click on the *Close* button.

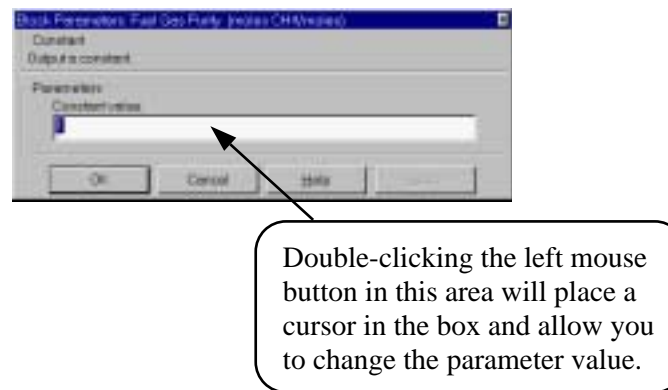


Figure 2.4. Instructions for Changing the *Fuel Gas Purity*

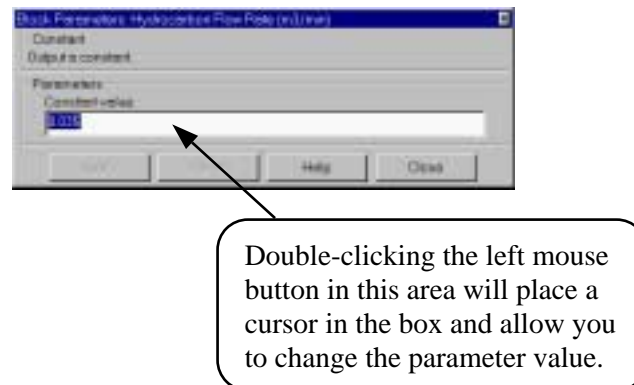


Figure 2.5. Instructions for Changing the Hydrocarbon Flow Rate

5. An important feature of the furnace module is the pointer button (see Figure 2.3). There is one pointer button for each of the controlled outputs (one for the *Hydrocarbon Outlet Temperature* and one for the *Oxygen Exit Concentration*). The pointer button allows the user to find the coordinates of a point on one of the output graphs by using the mouse. Single-clicking on one of the pointer buttons by using the left mouse button changes the mouse arrow to a crosshair sight. The crosshair sight can be moved to a spot on one of the output curves in the furnace monitor. Use the corresponding pointer button for a specified graph. The sight should be moved to the *Hydrocarbon Outlet Temperature* curve if the *Hydrocarbon Outlet Temperature* pointer is selected. Clicking the left mouse button once displays the coordinates of the selected point to the upper right corner of the output curve. Now return the *Hydrocarbon Flow Rate* to 0.035 by double-clicking on the Flow Rate box and adjusting the value, as done before.
6. Increase the air flow rate by 10% (from 17.9 to $19.69 \frac{m^3}{min}$) by using the procedure described above (see Figure 2.5). Make careful note of the time that you introduce the change. The change does not occur until you hit the *Close* or *Apply* button. Using the *Hydrocarbon Outlet Temperature* and *Oxygen Exit Concentration* pointer buttons, determine which output takes the longest to begin reacting to the change in *Air Flow Rate* (i.e., which output has the most dead time). You want to find the time on each output curve at which the output starts to change from the initial steady state value.
Exercise 2.1 What is the time when the change in *Air Flow Rate* is introduced (min)?
Exercise 2.2 Record the time when the *Hydrocarbon Outlet Temperature* begins to change (min).
Exercise 2.3 What is the time when the *Oxygen Exit Concentration* begins to change (min)?
Exercise 2.4 Which output exhibits the largest dead time?
Do not forget to return the *Air Flow Rate* to its original value ($17.9 \frac{m^3}{min}$)

Steady State Process Modeling—Furnace

7. Start the furnace.
Exercise 2.5 Record the initial steady state values for each of the inputs and outputs of the furnace:

INPUTS

Hydrocarbon Flow Rate	_____	$\frac{m^3}{min}$
Hydrocarbon Inlet Temperature	_____	K
Air Flow Rate	_____	$\frac{m^3}{min}$
Air Temperature	_____	K
Fuel Gas Flow Rate	_____	$\frac{m^3}{min}$
Fuel Gas Temperature	_____	K
Fuel Gas Purity	_____	$\frac{mol CH_4}{mol total}$

OUTPUTS

Hydrocarbon Outlet Temperature	_____	K
Furnace Temperature	_____	K
Exhaust Gas Flow Rate	_____	$\frac{m^3}{min}$
Oxygen Exit Concentration	_____	$\frac{mol O_2}{min}$

8. Make the following sequence of increases in the air flow rate by double-clicking the left mouse button on the *Air Flow Rate* box. The remaining inputs (the six other inputs) should be kept at their initial steady state values. After each change in the *Air Flow Rate*, allow the system to reach a new steady state (approximately 40 simulation minutes) and then record the values of the output variables obtained using the pointers on the output graphs.

Exercise 2.6 Record the steady state values:

Air Flow Rate	Hydrocarbon Outlet Temp.	O_2 Exit Conc.
17.9 (nominal)		
18.1		
18.3		
18.5		
18.7		

Return the *Air Flow Rate* to its initial value and allow the furnace to reach steady state.

9. Make the following sequence of increases in the *Fuel Gas Flow Rate* by double-clicking the left mouse button on the *Fuel Gas Flow Rate* box. The remaining inputs (the six other inputs) should be kept at their initial steady state values. Again, record the steady state values of the output variables at each of the new steady states.

Exercise 2.7 Record the steady state values:

Fuel Gas Flow Rate	Hydrocarbon Outlet Temp.	O ₂ Exit Conc.
1.21 (nominal)		
1.22		
1.23		
1.24		
1.25		

Return the *Fuel Gas Flow Rate* to its initial value and allow the furnace to reach steady state.

- Make the following sequence of increases in the *Hydrocarbon Flow Rate* by double-clicking the left mouse button on the *Hydrocarbon Flow Rate* box. The remaining inputs (the six other inputs) should be kept at their initial steady state values. Again, record the values of the output variables at each of the new steady states.

Exercise 2.8 Record the steady state values:

Hydrocarbon Flow Rate	Hydrocarbon Outlet Temp.	O ₂ Exit Conc.
0.0350 (nominal)		
0.0355		
0.0360		
0.0365		
0.0370		

Return the *Hydrocarbon Flow Rate* to its initial value and allow the furnace to reach steady state.

- Make the following sequence of decreases in the *Fuel Gas Purity* by double-clicking the left mouse button on the *Fuel Gas Purity* box. The remaining inputs (the six other inputs) should be kept at their initial steady state values. Again, record the values of the output variables at each of the new steady states.

Exercise 2.9 Record the steady state values:

Fuel Gas Purity	Hydrocarbon Outlet Temp.	O ₂ Exit Conc.
1.00 (nominal)		
0.99		
0.98		
0.97		
0.95		

Return the *Fuel Gas Purity* to its initial value and allow the furnace to reach steady state.

Steady State System Gains

12. Using the information from Procedures 8–11, calculate the steady state gains for each of the following input-output pairings. This can be accomplished graphically by plotting the output versus input values from the tables and calculating the best linear fit to the data. The MATLAB function polyfit can also be used to find a least-squares fit to your data.

For example, given the following input-output data,

Input A	Output B	Output C
0.45	1.69	379.7
0.55	1.72	381.5
0.65 (nominal)	1.93	393.5
0.75	1.97	401.6
0.85	2.00	410.1

the gain is obtained by plotting the outputs (B and C) versus the input and calculating the slope (see Figure 2.6).

The gain is defined as the change in the output divided by the change in input which is equivalent to the slope of the input-output curve. One feature that distinguishes the behavior of Output B from Output C, is the degree of nonlinearity in the system. The data for Output C is better fit by a linear equation than that of Output B as a result of the nonlinear behavior of Output B. Thus, the predictive ability of the gain for the Input A-Output C coupling should be much more accurate than that of the gain for the Input A-Output B coupling.

Exercise 2.10 Calculate the gains for the input-output relationships.

Inputs

Air Flow Rate
 Fuel Flow Rate
 Fuel Gas Purity
 Hydrocarbon Flow Rate

Outputs

Oxygen Exit Concentration
 Hydrocarbon Outlet Temperature

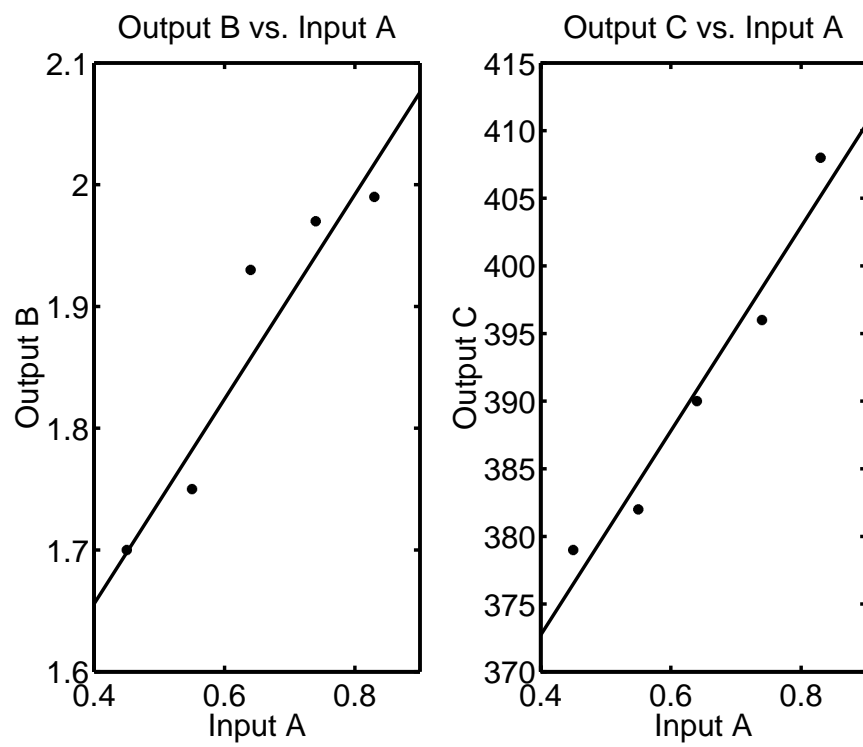


Figure 2.6. Plot of Output Data B and C versus Input Data A

$$\begin{aligned}
K_{11} &= \frac{(\text{Change in Hydrocarbon Outlet Temperature})}{(\text{Change in Fuel Gas Flow Rate})} = \frac{K}{\frac{m^3}{min}} \\
K_{12} &= \frac{(\text{Change in Hydrocarbon Outlet Temperature})}{(\text{Change in Air Flow Rate})} = \frac{K}{\frac{m^3}{min}} \\
K_{1d2} &= \frac{(\text{Change in Hydrocarbon Outlet Temperature})}{(\text{Change in Hydrocarbon Flow Rate})} = \frac{K}{\frac{m^3}{min}} \\
K_{1d1} &= \frac{(\text{Change in Hydrocarbon Outlet Temperature})}{(\text{Change in Fuel Gas Purity})} = \frac{K}{\frac{mol CH_4}{mol}} \\
K_{21} &= \frac{(\text{Change in Oxygen Exit Concentration})}{(\text{Change in Fuel Gas Flow Rate})} = \frac{\frac{mol O_2}{m^3}}{\frac{m^3}{min}} \\
K_{22} &= \frac{(\text{Change in Oxygen Exit Concentration})}{(\text{Change in Air Flow Rate})} = \frac{\frac{mol O_2}{m^3}}{\frac{m^3}{min}} \\
K_{2d2} &= \frac{(\text{Change in Oxygen Exit Concentration})}{(\text{Change in Hydrocarbon Flow Rate})} = \frac{\frac{mol O_2}{m^3}}{\frac{m^3}{min}} \\
K_{2d1} &= \frac{(\text{Change in Oxygen Exit Concentration})}{(\text{Change in Fuel Gas Purity})} = \frac{\frac{mol O_2}{m^3}}{\frac{mol CH_4}{mol}}
\end{aligned}$$

Fill in the block diagram in Figure 2.7 with the gains calculated above.

13. Increase the nominal *Air Flow Rate* by 20% and repeat Exercises 2.6 to 2.9.

Exercise 2.11 Compared with results from Exercise 2.10, is the nonlinear behavior of the furnace apparent? How is this behavior manifested?

Manual Control of the Furnace

The steady state gains can be used to determine the input-output behavior of the furnace. They can be calculated from the block diagram, the gain expressions on the previous page, and the following general equations:

$$\Delta T_h = K_{11}\Delta V_{fg} + K_{12}\Delta V_{air} + K_{1d1}\Delta d_1 + K_{1d2}\Delta d_2$$

$$\Delta O_2 = K_{21}\Delta V_{fg} + K_{22}\Delta V_{air} + K_{2d1}\Delta d_1 + K_{2d2}\Delta d_2$$

where:

ΔT_h - Change in Hydrocarbon Outlet Temperature (Controlled Variable)

ΔO_2 - Change in Oxygen Exit Concentration (Controlled Variable)

ΔV_{fg} - Change in Fuel Gas Flow Rate (Manipulated Variable)

ΔV_{air} - Change in Air Flow Rate (Manipulated Variable)

Δd_1 - Change in Fuel Gas Purity (Load Variable)

Δd_2 - Change in Hydrocarbon Flow Rate (Load Variable)

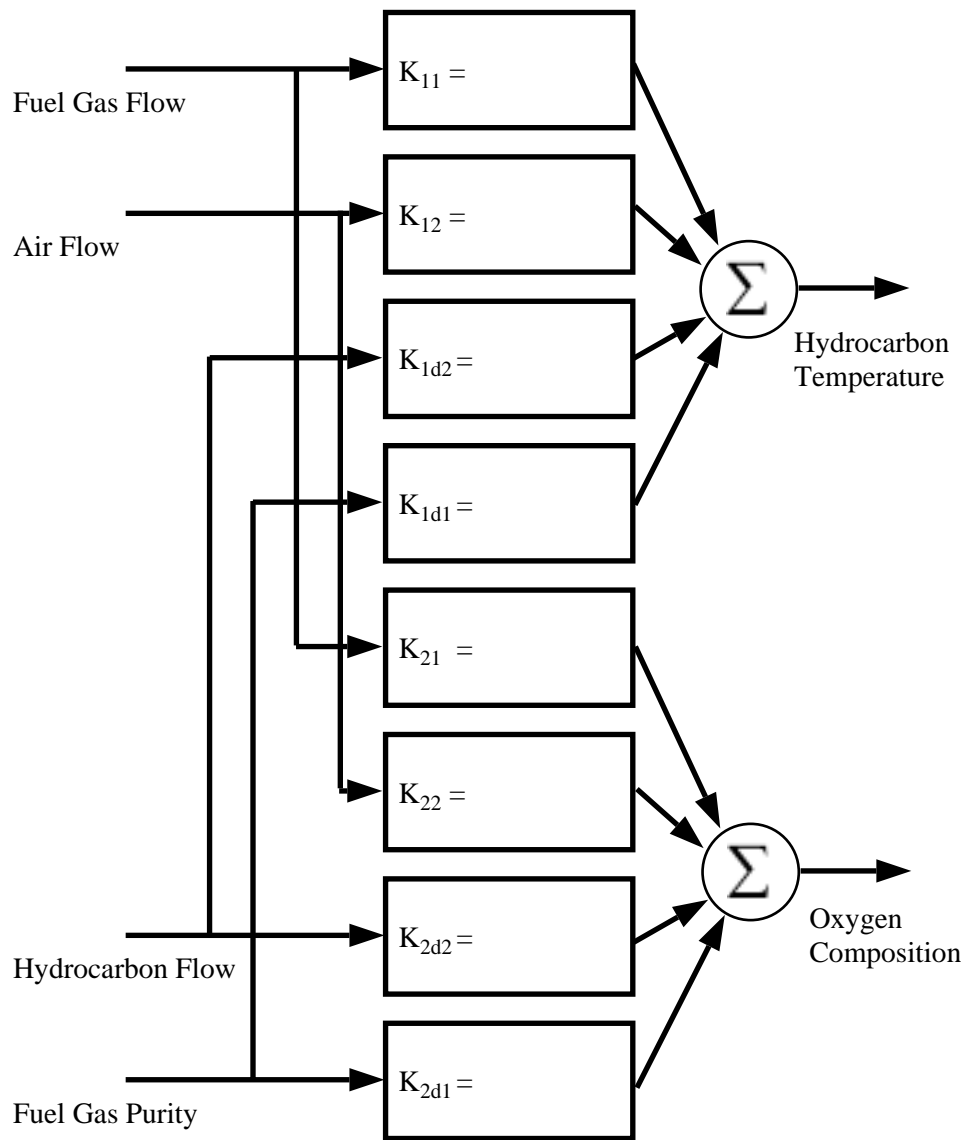


Figure 2.7. Steady State Gain Values for the Furnace

14. Using the gains obtained in Exercise 2.10, determine the values of the *Air Flow Rate* and *Fuel Gas Flow Rate* that are necessary to increase the *Hydrocarbon Outlet Temperature* by 7°C and decrease the *Oxygen Exit Concentration* by $0.05 \frac{\text{mol O}_2}{\text{m}^3}$.

Assume that the load variables remain constant:

$$\begin{aligned}\Delta T_h &= K_{11} \Delta V_{fg} + K_{12} \Delta V_{air} \\ \Delta O_2 &= K_{21} \Delta V_{fg} + K_{22} \Delta V_{air}\end{aligned}$$

Solve these equations simultaneously to find values of the manipulated variables which change the controlled variables as specified.

Exercise 2.12 Calculate the new value of the *Fuel Gas Flow Rate* ($\frac{\text{m}^3}{\text{min}}$).

Exercise 2.13 Calculate the new value of the *Air Flow Rate* ($\frac{\text{m}^3}{\text{min}}$).

15. Implement this manipulated variable change in the furnace and record the values of the output variables.

Exercise 2.14 How close are the ultimate process outputs to the desired values?

Exercise 2.15 Record the actual final (steady state) values.

Return the *Fuel Gas Flow Rate* and the *Air Flow Rate* to their initial values and allow the furnace to reach steady state.

16. Assuming that the purity of the fuel gas decreases from 1.0 to $0.96 \frac{\text{mol CH}_4}{\text{mol total}}$, use the gains obtained in Exercise 2.10 to calculate the new values for the controlled variables.

Assume that the manipulated variables and other disturbances remain constant:

$$\begin{aligned}\Delta T_h &= K_{1d1} \Delta d_1 + K_{1d2} \Delta d_2 \\ \Delta O_2 &= K_{2d1} \Delta d_1 + K_{2d2} \Delta d_2\end{aligned}$$

Solve these equations to find the new values of the controlled variables.

Exercise 2.16 Calculate the new value of the *Hydrocarbon Outlet Temperature* (K).

Exercise 2.17 Calculate the new value of the *Oxygen Exit Concentration* ($\frac{\text{mol}}{\text{m}^3}$).

17. Implement this load variable change in the furnace and record the values of the output variables.

Exercise 2.18 How close are the ultimate process outputs to the desired values?

18. Using the block diagram, calculate values for the manipulated variables which would return the controlled variables to their initial steady states for this change in load variables.

Exercise 2.19 Calculate the new value of the *Air Flow Rate* ($\frac{m^3}{min}$).

Exercise 2.20 Calculate the new value of the *Fuel Gas Flow Rate* ($\frac{m^3}{min}$).

19. Implement the changes in manipulated variables you calculated and wait for the furnace to return to a new steady state.

Exercise 2.21 Do the controlled variables return to their initial steady states?

20. To end the session, stop the simulation by selecting *Stop* under the Simulation menu, then select *Yes* under the *Quit* menu from the Main Menu window. This will return you to the MATLAB prompt. At this prompt, type *quit* to exit MATLAB.

PROCEDURE—COLUMN

The process under consideration in this unit is a binary distillation column which separates a mixture of methanol (MeOH) and ethanol. Distillation is a very important unit operation in the chemical and petroleum industries. Increasing demand for high quality products coupled with the demand for more efficient energy utilization has highlighted the role of process control for distillation columns.

The particular column studied in this unit has 27 trays, a reboiler on the bottom tray, and a total condenser on the overhead stream. A 50%-50% mixture of methanol and ethanol is fed at the fourteenth tray (counted from the bottom). This column was originally modeled by K. Weischedel and T.J. McAvoy in 1980. It represents a benchmark that has been studied by a number of researchers for the purpose of controller design. The specific control objective is to achieve an 85% methanol stream at the top and an 85% ethanol stream at the bottom of the column. This is referred to as dual-composition control. A schematic of the process can be found in Figure 2.8.

The column is modeled with component mass balances and steady state energy balances which result in coupled nonlinear differential algebraic equations. The column model has four inputs and four outputs as listed below.

INPUTS	OUTPUTS
Reflux Ratio	Overhead MeOH Composition
Vapor Flow Rate	Overhead Flow rate
Feed MeOH Composition	Bottom MeOH Composition
Feed Flow Rate	Bottom Flow Rate

The column has the following manipulated and controlled variables:

Manipulated Variables	Controlled Variables
Reflux Ratio	Overhead MeOH Composition
Vapor Flow Rate	Bottom MeOH Composition

The system also has the following load (or disturbance) variables:

Load Variables
Feed Flow Rate
Feed MeOH Composition

Column Operation

After starting MATLAB, type *mainmenu* at the MATLAB prompt. Double-click on the button *Distillation Column* (Figure 1.1). This will bring up the Distillation Column Menu (Figure 1.3). Selecting *Distillation Column* from this menu will bring up two figures: the distillation column flow diagram (Figure 2.9), and the distillation column process monitor (Figure 2.10).

1. Double-click on the box *Vapor Flow Rate*. Use the backspace key to delete the existing value and enter 0.045 m³/sec (Figure 2.11). Close the box and start the simulation. Click on the window showing the monitor to bring it to the front of your screen. It gives information about the system outputs: *Overhead Flow Rate*, *Overhead MeOH Composition*, *Bottom Flow Rate*, and *Bottom MeOH Composition*. These graphs give both the current and the nominal process values.
2. Changes in the values of *Reflux Ratio*, *Feed Flow Rate* and *Feed MeOH Composition* can be given in a similar manner. Increase the value of *Reflux Ratio* to 3.0. Observe the effect on the outputs. Once the system has reached a new steady state, increase the *Feed MeOH Composition* to 0.55. Again, observe the effect on the outputs.

When the column has reached the new steady state, return all of the input values to their nominal values and allow the system to return to its initial steady state.

Steady State Process Modeling—Column

3. Start the column.

Exercise 2.1 Record the initial steady state values for the inputs and outputs.

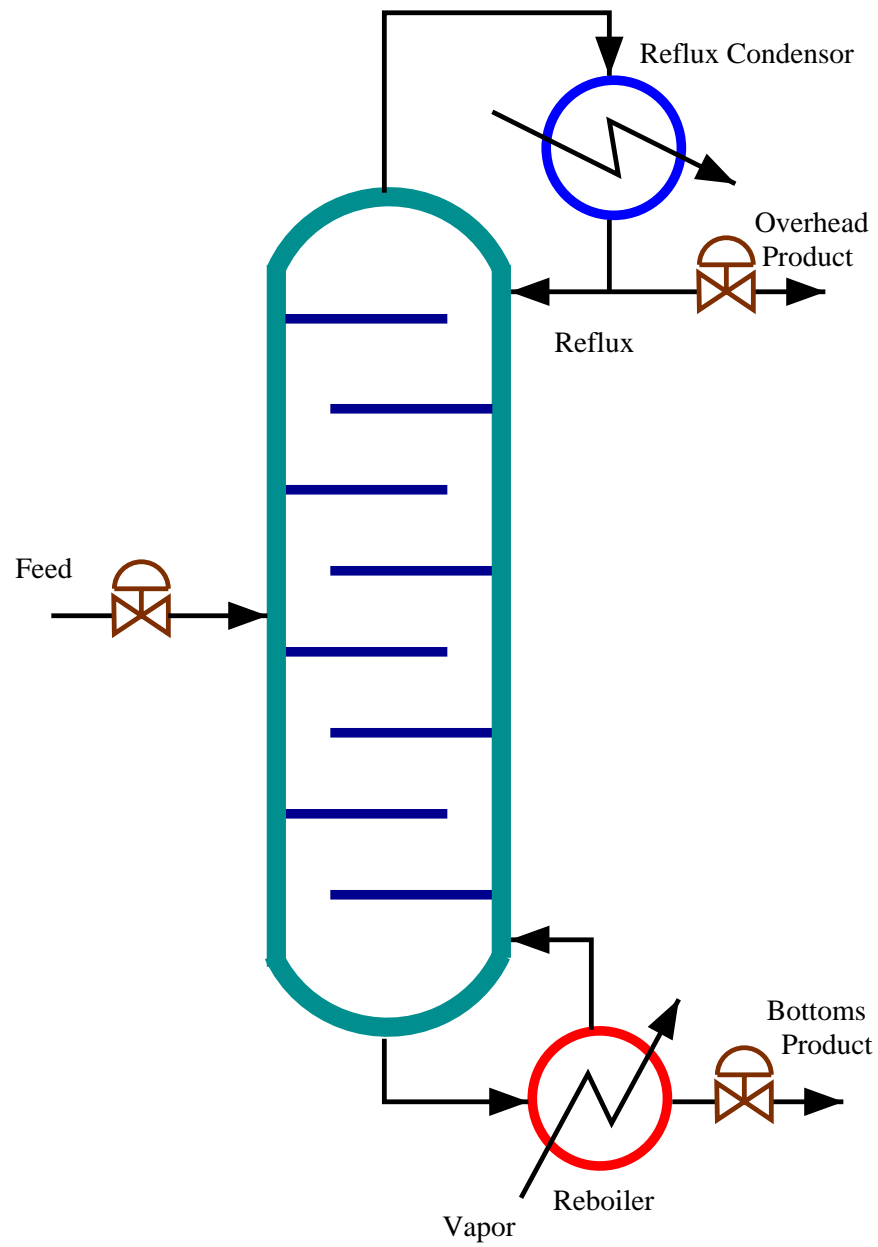


Figure 2.8. Process Schematic of the Column

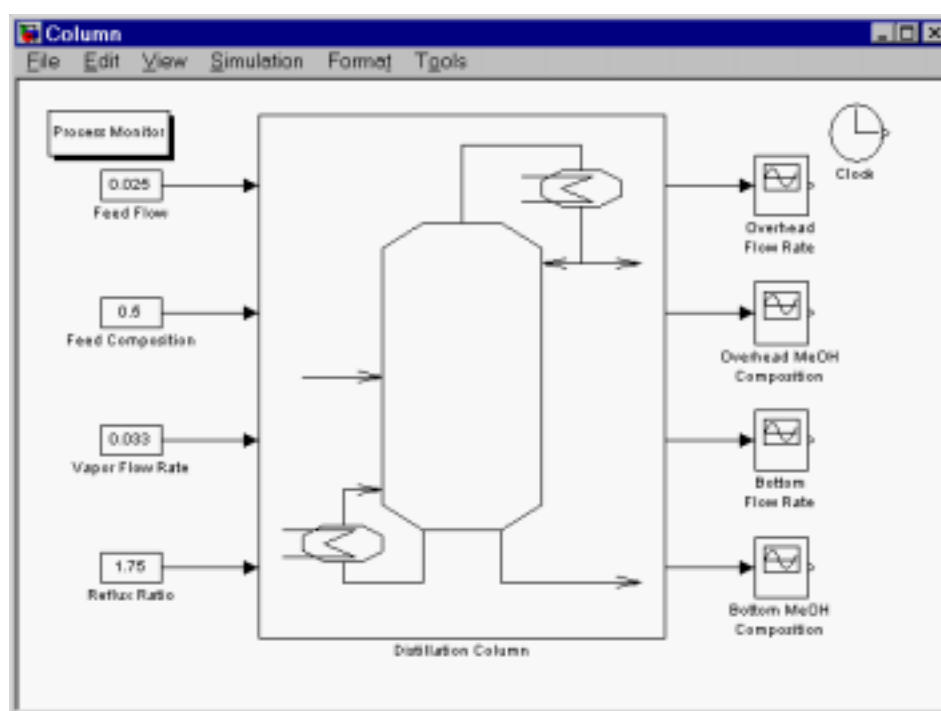
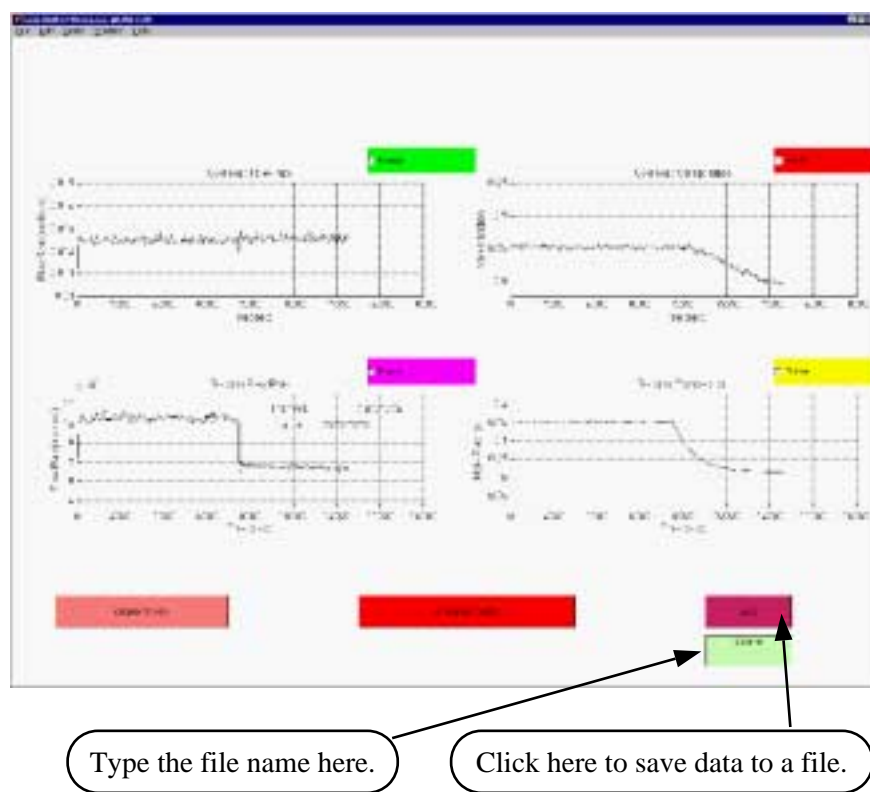


Figure 2.9. Operator Interface for the Column

**Figure 2.10.** Column Process Monitor

INPUTS

Reflux Ratio	_____	
Vapor Flow Rate	_____	$\frac{\text{mol}}{\text{sec}}$
Feed MeOH Composition	_____	$\frac{\text{mol MeOH}}{\text{mol total}}$
Feed Flow Rate	_____	$\frac{\text{mol}}{\text{sec}}$

OUTPUTS

Overhead MeOH Composition	_____	$\frac{\text{mol MeOH}}{\text{mol total}}$
Bottom MeOH Composition	_____	$\frac{\text{mol MeOH}}{\text{mol total}}$

4. Make the following sequence of moves in the *Reflux Ratio* by double-clicking the left mouse button on the *Reflux Ratio* box. Keep the remaining inputs (the three other inputs) at their initial steady state values. After each change in the *Reflux Ratio*, allow the system to reach a new steady state and then record the values of the output variables.



Double-clicking the left mouse button in this area will place a cursor in the box and allow you to change the parameter value.

Figure 2.11. Instructions for Changing the *Vapor Flow Rate*

Exercise 2.2 Record the steady state values:

Reflux Ratio	Overhead MeOH Composition	Bottom MeOH Composition
1.75 (nominal)		
1.85		
1.95		
1.65		
1.55		

Return the *Reflux Ratio* to its initial value and allow the column to reach steady state.

5. Make the following sequence of changes in the *Vapor Flow Rate* by double-clicking the left mouse button on the *Vapor Flow Rate* box. Keep the remaining inputs (the three other inputs) at their initial steady state values. Again, record the steady state values of the output variables at each of the new steady states.

Exercise 2.3 Record the steady state values:

Vapor Flow Rate	Overhead MeOH Composition	Bottom MeOH Composition
0.033 (nominal)		
0.040		
0.036		
0.034		
0.032		

Return the *Vapor Flow Rate* to its initial value and allow the column to reach steady state.

6. Make the following sequence of changes in the *Feed Flow Rate* by double-clicking the left mouse button on the *Feed Flow Rate* box. Keep the remaining inputs (the three other inputs) at their initial steady state values. Again, record the values of the output variables at each of the new steady states.

Exercise 2.4 Record the steady state values:

Feed Flow Rate	Overhead MeOH Composition	Bottom MeOH Composition
0.025 (nominal)		
0.026		
0.024		

Return the *Feed Flow Rate* to its initial value and allow the column to reach steady state.

7. Make the following sequence of changes in the *Feed Composition* by double-clicking the left mouse button on the *Feed Composition* box. Keep the remaining inputs (the three other inputs) at their initial steady state values. Again, record the values of the output variables at each of the new steady states.

Exercise 2.5 Record the steady state values:

Feed Composition	Overhead MeOH Composition	Bottom MeOH Composition
0.50 (nominal)		
0.53		
0.47		

Return the *Feed Composition* to its initial value and allow the column to reach steady state.

Steady State System Gains

8. Using the information from Procedures 4–7, fill in the equations on the following page with the calculated values of the steady state gains for each of the following input-output pairings. This can be accomplished graphically by plotting the output versus input values from the tables and calculating the best linear fit to the data.

Exercise 2.6 Calculate the gains for the input-output relationships.

Inputs

Reflux Ratio
Vapor Flow Rate
Feed MeOH Composition
Feed Flow Rate

Outputs

Overhead MeOH Composition
Bottom MeOH Composition

$$\begin{aligned}
 K_{11} &= \frac{(\text{Change in Overhead MeOH Composition})}{(\text{Change in Reflux Ratio})} = \text{---} \left(\frac{1}{1} \right) \\
 K_{12} &= \frac{(\text{Change in Overhead MeOH Composition})}{(\text{Change in Vapor Flow Rate})} = \text{---} \left(\frac{1}{\text{mol/sec}} \right) \\
 K_{1d2} &= \frac{(\text{Change in Overhead MeOH Composition})}{(\text{Change in Feed Flow Rate})} = \text{---} \left(\frac{1}{\text{mol/sec}} \right) \\
 K_{1d1} &= \frac{(\text{Change in Overhead MeOH Composition})}{(\text{Change in Feed MeOH Composition})} = \text{---} \left(\frac{1}{1} \right) \\
 K_{21} &= \frac{(\text{Change in Bottom MeOH Composition})}{(\text{Change in Reflux Ratio})} = \text{---} \left(\frac{1}{1} \right) \\
 K_{22} &= \frac{(\text{Change in Bottom MeOH Composition})}{(\text{Change in Vapor Flow Rate})} = \text{---} \left(\frac{1}{\text{mol/sec}} \right) \\
 K_{2d2} &= \frac{(\text{Change in Bottom MeOH Composition})}{(\text{Change in Feed Flow Rate})} = \text{---} \left(\frac{1}{\text{mol/sec}} \right) \\
 K_{2d1} &= \frac{(\text{Change in Bottom MeOH Composition})}{(\text{Change in Feed MeOH Composition})} = \text{---} \left(\frac{1}{1} \right)
 \end{aligned}$$

Exercise 2.7 Fill in the block diagram in Figure 2.12 with the gains calculated above.

9. The assumption implicit in the block diagram in Figure 2.8 is that changes in the controlled variables can be calculated by summing the changes in all the manipulated and load variables multiplied by constant gain coefficients. This is strictly valid only for **linear** processes. Most chemical processes, including the distillation column, are nonlinear and this linear approximation is valid only for a small region around the normal operating point. Performing tests on the column at some other operating point would yield a different set of steady state gains.

Exercise 2.8 Can you detect the process nonlinearity from the plot of *Overhead MeOH Composition* versus *Reflux Ratio*? How so?.

Manual Control of the Column

The steady state gains can be used to determine the input-output behavior of the column. They can be calculated from the block diagram, the gain expressions on the previous page, and the following general equations:

$$\Delta Y_d = K_{11}\Delta R + K_{12}\Delta V + K_{1d_1}\Delta d_1 + K_{1d_2}\Delta d_2$$

$$\Delta X_b = K_{21}\Delta R + K_{22}\Delta V + K_{2d_1}\Delta d_1 + K_{2d_2}\Delta d_2$$

where:

ΔY_d - Change in *Overhead MeOH Composition* (Controlled Variable)

ΔX_b - Change in *Bottom MeOH Composition* (Controlled Variable)

ΔR - Change in *Reflux Ratio* (Manipulated Variable)

ΔV - Change in *Vapor Flow Rate* (Manipulated Variable)

Δd_1 - Change in *Feed MeOH Composition* (Load Variable)

Δd_2 - Change in *Feed Flow Rate* (Load Variable)

10. Using the gains obtained in Procedure 8, determine the values of the *Reflux Ratio* and *Vapor Flow Rate* that are necessary to increase the *Overhead MeOH Composition* to 0.88 and the *Bottom MeOH Composition* to 0.17.

Assume that the load variables remain constant:

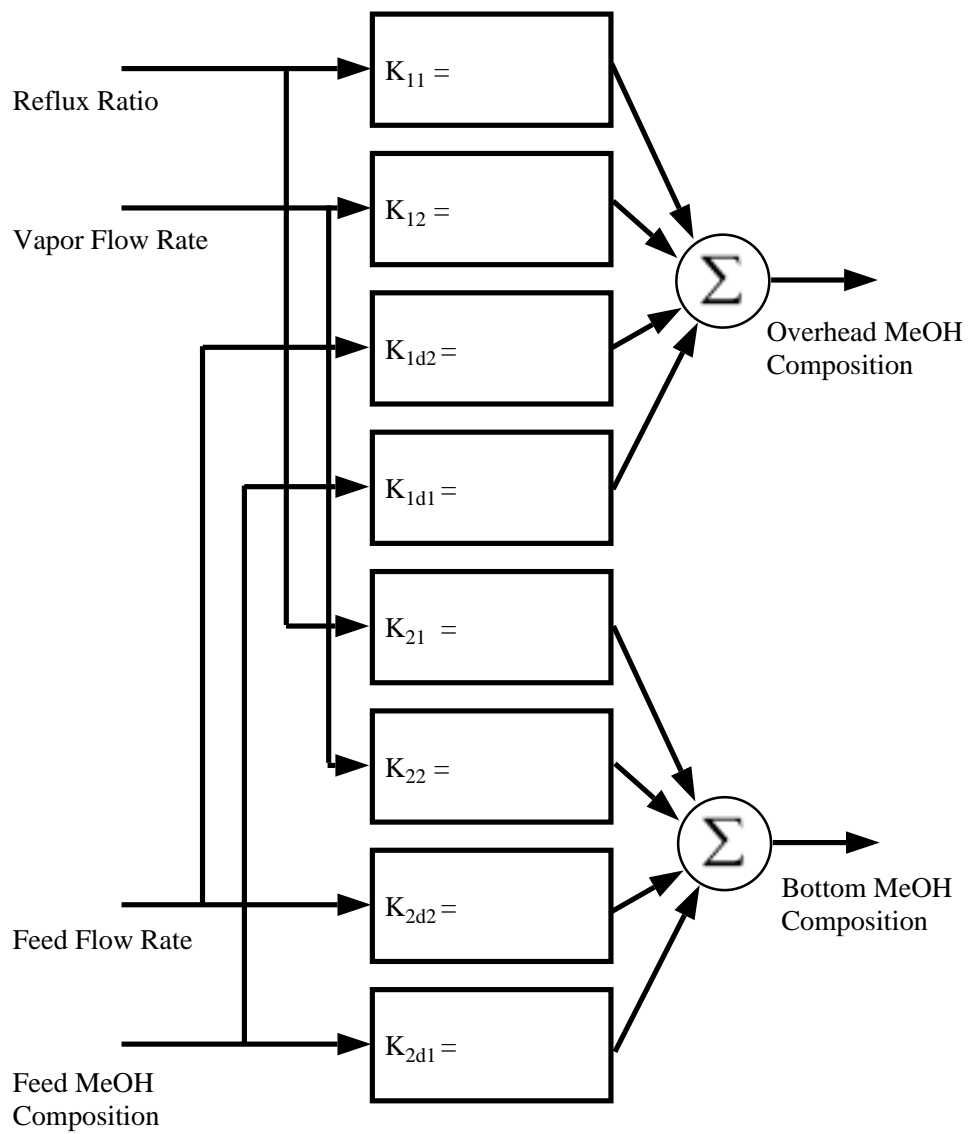


Figure 2.12. Steady-state Gain Values for the Column

$$\Delta Y_d = K_{11}\Delta R + K_{12}\Delta V$$

$$\Delta X_b = K_{21}\Delta R + K_{22}\Delta V$$

Solve these equations simultaneously to find values of the manipulated variables which change the controlled variables as specified.

Exercise 2.9 Calculate the new value of the *Reflux Ratio*.

Exercise 2.10 Calculate the new value of the *Vapor Flow Rate*.

11. Implement this manipulated variable change in the column and record the values of the output variables.

Exercise 2.11 How close do the outputs come to the desired values?

Exercise 2.12 Record the actual values.

Return the *Reflux Ratio* and *Vapor Flow Rate* to their initial values and allow the column to reach steady state.

12. Assuming that the feed flow rate increases to $0.029 \frac{\text{mol}}{\text{sec}}$, use the gains obtained in Procedure Step 8 to calculate the new values for the controlled variables. Assume that the manipulated variables remain constant:

$$\Delta Y_d = K_{1d_1}\Delta d_1 + K_{1d_2}\Delta d_2$$

$$\Delta X_b = K_{2d_1}\Delta d_1 + K_{2d_2}\Delta d_2$$

Solve these equations simultaneously to find the new values of the controlled variables.

Exercise 2.13 What is the new *Overhead MeOH Composition*?

Exercise 2.14 What is the new *Bottom MeOH Composition*?

13. Implement this load variable change in the column and record the values of the output variables.

Exercise 2.15 How close do the outputs come to the predicted values?

14. Using the block diagram, calculate values for the manipulated variables which would return the controlled variables to their initial steady states for this change in load variables.

Exercise 2.16 What is the new *Reflux Ratio*?

Exercise 2.17 What is the new *Vapor Flow Rate*?

15. Implement the changes in manipulated variables you calculated and wait for the column to return to a new steady state.

Exercise 2.18 Do the controlled variables return to their initial steady states?

16. To end the session, stop the simulation by selecting *Stop* under the Simulation menu, then select *Yes* under the Quit menu from the Main Menu. This will return you to the MATLAB prompt. At this prompt, type *quit* to exit MATLAB. To exit the system, select the *EXIT* icon located on the right bottom side of the menu bar. Select *OK* from the Logout Confirmation box.

SUMMARY

This unit covers the basic operation of system operation, including the various process variables. At the end of this unit, you should be comfortable with the notion of a process gain, as well as the use of process gains in accomplishing manual (steady state) control of a process system.

Module 3

FIRST-ORDER DYNAMIC SYSTEM ANALYSIS

OBJECTIVE

The purpose of this module is to demonstrate the properties of a first-order system for various values of the system gain and time constant. This module also illustrates the dynamic response of a first-order system to different input signals.

Open-Ended Module Procedure

Use SIMULINK to explore the different dynamic responses obtained from a first-order system as one changes the gain, and separately, the time constant. Calculate a first-order approximation for the unknown system in *System Identification Problem 1* menu.

THINGS TO THINK ABOUT

Exercise 3-A What effect does increasing the gain have on the system output?

Exercise 3-B What is meant physically by a system with a large gain?

Exercise 3-C What effect does decreasing the time constant have on the system output?

Exercise 3-D What is meant physically by a system with a small time constant?

Exercise 3-E Is it possible for a system to have a negative gain? What is the expected behavior?

Exercise 3-F Is it possible for a system to have a negative time constant? What is the expected behavior?

Exercise 3-G What is the expected response from a first-order system driven by a sinusoidal input?

INTRODUCTION

The output of a linear first-order dynamic system is described by a linear first-order differential equation. The general form for such a system is:

$$\tau_p \frac{dy}{dt} + y(t) = K_p u(t)$$

where:

- $y(t)$ is the system output;
- $u(t)$ is the input (forcing function);
- K_p is the system gain; and
- τ_p is the system time constant.

The gain and the time constant are the two parameters that determine the characteristics of the dynamic response of a system. The Laplace transform of this differential equation yields a first-order transfer function:

$$G_p(s) = \frac{y(s)}{u(s)} = \frac{K_p}{\tau_p s + 1}$$

Some typical examples of first-order systems include:

- A freely draining tank
- An isothermal CSTR (with a single reaction and first-order, irreversible kinetics)
- A mercury thermometer

PROCEDURE

To start the first-order system, click once on the *First and Second Order Systems* (Figure 3.1) button from the Main Menu (Figure 1.1) then select the *First-order System* button. Figures 3.2 and 3.3 display the two windows that are used for the first-order system; the first is the system window, and the second is the input/output window. Refer to Figures 3.4 and 3.5 for instructions on how to change the simulation parameters for the first-order system.

1. First, set the system gain K_p and the system time constant τ_p both to 10.0 (see Figure 3.4). Now set the initial value of the step function to 0.0 and the



Figure 3.1. First and Second Order Systems Menu

This window represents the first-order system. To run the simulation, select *Start* under the *Simulation* menu in this window.

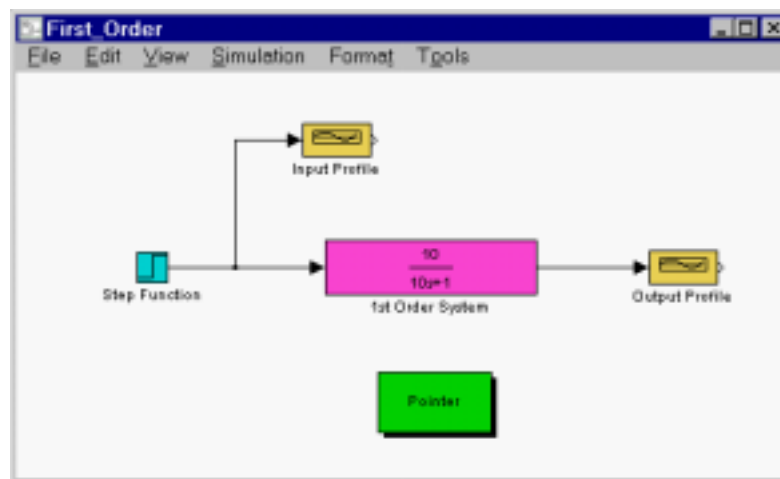


Figure 3.2. First-Order System Flowsheet Window

The input and output from the system will be plotted with respect to time in the display window.

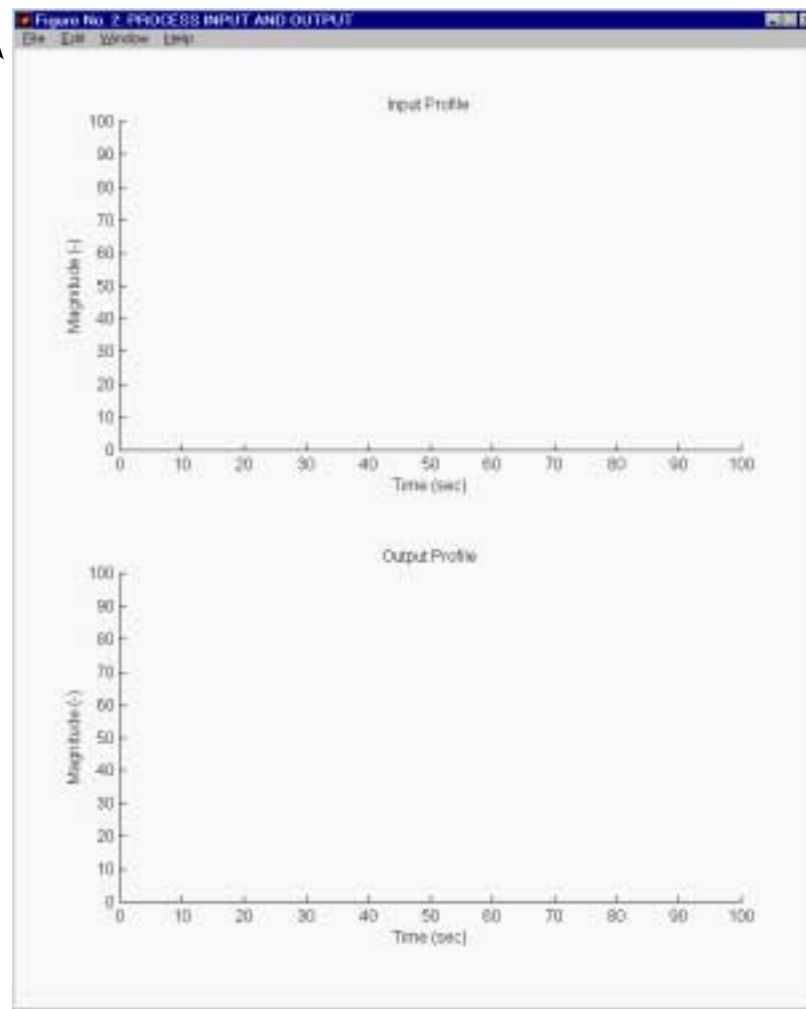


Figure 3.3. First-Order System Process Display Window

final value of the step function to 1.0 (see Figure 3.5). To start the simulation, select *Start* from the *Simulation* menu.

Exercise 3.1 Enter the new steady state value.

Exercise 3.2 Enter the length of time it takes for the output to reach the new steady state (sec).

2. Now increase the value of K_p to 40.0 and repeat the previous exercise.

Exercise 3.3 How does this response differ from the response in exercise 3.1?

3. Set K_p back to 10.0 and now try increasing the value of τ_p to 20.0. Repeat the simulation.

Exercise 3.4 How does the response differ from the response in exercise 3.1?

4. Now decrease the value of the K_p to 5.0 and decrease the value of τ_p to 5.0. Repeat the simulation.

Exercise 3.5 Enter the new steady state value.

Exercise 3.6 Enter the amount of time it takes (sec) for the output to reach the new steady value.

System Identification Problem

In practice, the transfer function for a given process operation must be determined from data generated by the system. For example, the engineer may determine a reasonable first-order approximation to the data for the purpose of controller design. With your knowledge of first-order systems and the information presented in the lectures, you will try to identify an unknown process.

5. From the Main Menu, select the *System Identification Problem 1* button. Using a step input, run the simulation to generate output data that can be used to determine the system gain (K_p) and the system time constant (τ_p). Remember to use the *Pointer* button and to take several points along the response curve in your analysis of the system output.

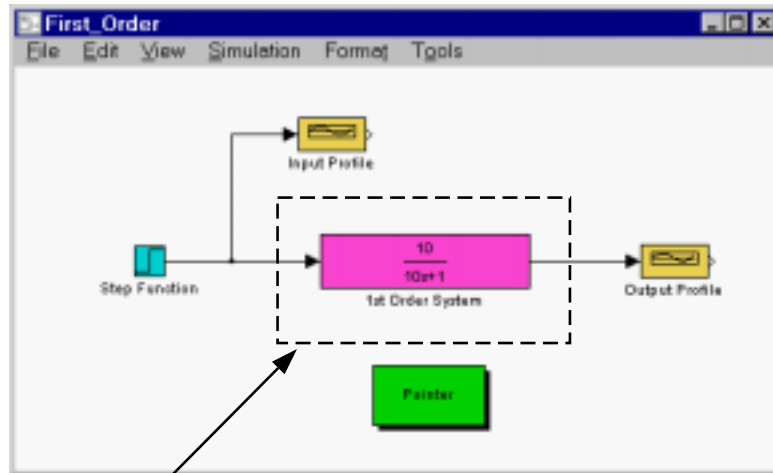
You may find it useful to calculate the following quantities:

Exercise 3.7 What is the slope of initial response?

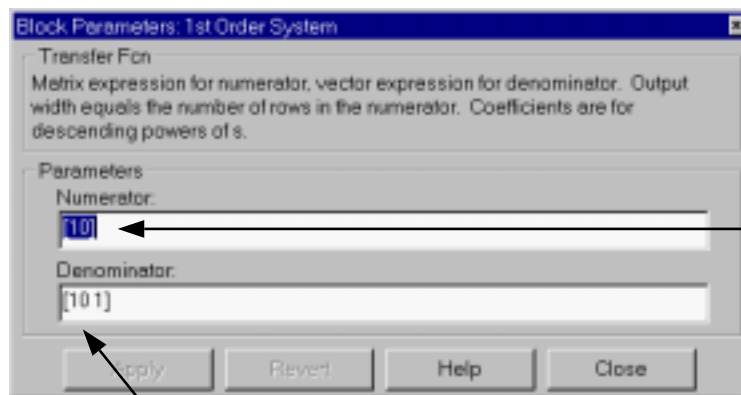
Exercise 3.8 Calculate the final output value minus the initial output value.

This information can be used to calculate the gain and time constant for this system.

Exercise 3.9 Fill in the following table with the parameter values you calculated:



By double-clicking this box, the Block Parameters window should appear. From this window you can change the system gain and system time constant.



System gain

$$G(s) = \frac{10}{10s + 1}$$

System time constant

These numbers are changed by placing the cursor inside the brackets, deleting the existing value, and typing a new value.

Figure 3.4. Instructions for First-Order System Parameters

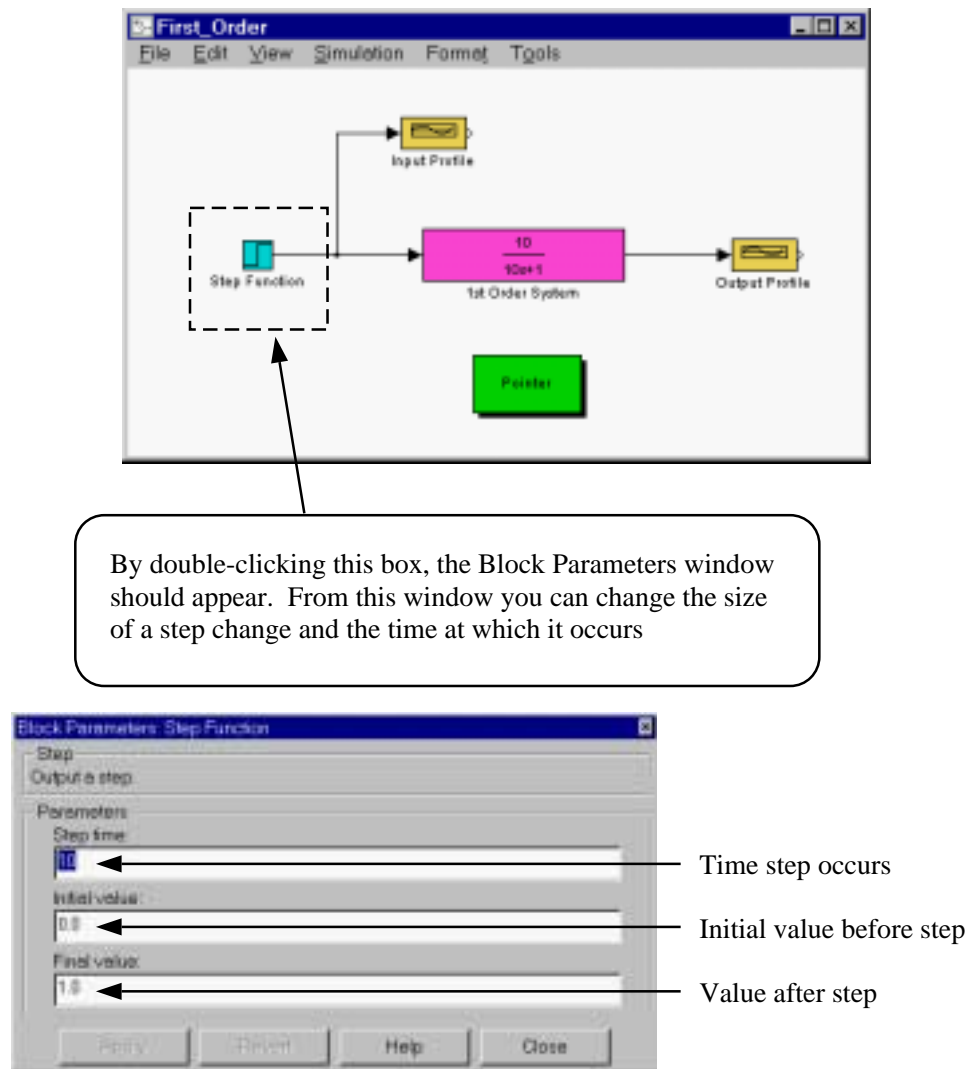


Figure 3.5. Instructions for Step Function Inputs

K_p	
τ_p	

Exercise 3.10 Give the first-order transfer function of this unknown system.

SUMMARY

This unit demonstrated the dynamic characteristics of first-order systems. One should be comfortable with the effect of changing the gain and/or the time constant. Finally, preliminary techniques for first-order system identification were covered.

Module 4

SECOND-ORDER DYNAMIC SYSTEM ANALYSIS

OBJECTIVE

The purpose of this module is to demonstrate the dynamic properties of a second-order system for various values of the system gain, natural period, and damping coefficient. This module also illustrates the dynamic response of a second-order system to different input signals.

Open-Ended Module Procedure

Use SIMULINK to explore the different dynamic responses obtained from a second-order system as one changes the gain, and separately, the natural period, and the damping coefficient. Calculate a second-order approximation for the unknown system in *System Identification Problem 2* menu.

THINGS TO THINK ABOUT

Consider the following values for the damping coefficient for a second-order dynamic system.

Region I	Region II	Region III
$\zeta < 1$	$\zeta = 1$	$\zeta > 1$

Exercise 4-A What types of response would be expected for a system with a damping coefficient in Region I? (*i.e.*, overdamped, critically damped, underdamped)

Exercise 4-B Sketch the corresponding response of the output variable to a step input.

Exercise 4-C What types of poles does this system have? (*i.e.*, real vs. imaginary; distinct vs. repeated)

Exercise 4-D What type of response would be expected for a system with a damping coefficient in Region II?

Exercise 4-E Sketch the corresponding response of the output variable to a step input.

Exercise 4-F What types of poles does this system have?

Exercise 4-G What type of response would be expected for a system with a damping coefficient in Region III?

Exercise 4-H Sketch the corresponding response of the output variable to a step input.

Exercise 4-I What types of poles does this system have?

Exercise 4-J How does a decrease in the damping coefficient affect the speed of response?

Exercise 4-K Which of the three responses would be expected to have a shorter response time?

Exercise 4-L Which of the three responses would be expected to be sluggish?

Exercise 4-M What are the trade-offs, from a control perspective, of the different responses? (*e.g.*, response times, rise times, overshoot, etc.)

INTRODUCTION

The output of a linear second-order dynamic system is described by a linear second-order differential equation (involving first and second derivatives of the output). The general form for such a system is as follows:

$$\tau_p^2 \frac{d^2 y(t)}{dt^2} + 2\zeta \tau_p \frac{dy(t)}{dt} + y(t) = K_p u(t)$$

where:

$y(t)$ is the system output;

$u(t)$ is the input (forcing function);

K_p is the system gain;

τ_p is the natural period of the system;

and ζ is the damping coefficient.

The dynamic behavior of second-order systems is determined by the gain, the time constant, and the damping coefficient. The Laplace transform of this differential equation yields a second-order transfer function:

$$G(s) = \frac{y(s)}{u(s)} = \frac{K_p}{\tau_p^2 s^2 + 2\zeta \tau_p s + 1}$$

Some typical examples of systems which exhibit second-order dynamic behavior are:

- First-order systems in series (*e.g.*, two interacting tanks)
- A stirred tank heater
- Many closed-loop systems (typically underdamped)

PROCEDURE

To start the second-order system, click once on the *First and Second Order Systems* button (Figure 3.1) from the Main Menu (Figure 1.1) then select the *Second-Order Systems* button. Figures 4.1 and 4.3 display the two windows used for the second-order system; the first is the system window, and the second is the input/output window. Refer to Figures 4.3 and 4.4 for instructions on how to change the simulation parameters for the second-order system.

This window represents the second-order system. To run the simulation, select *Start* under the *Simulation* menu in this window.

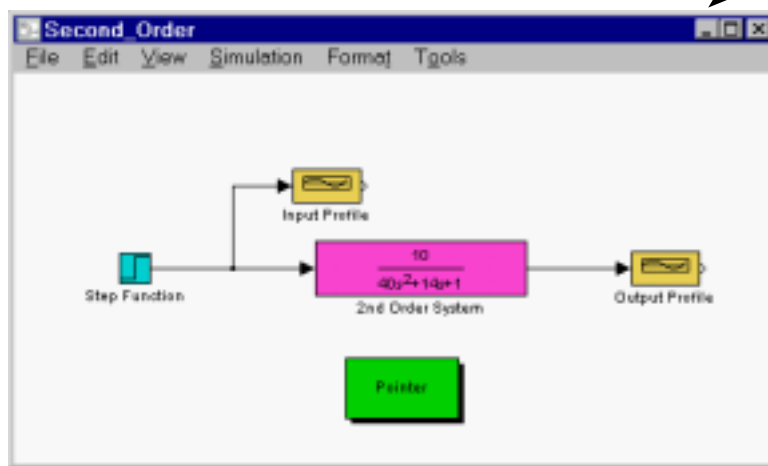


Figure 4.1. Second-Order System Flowsheet Window

The system under consideration is described by the following transfer function:

The input and output from the system will be plotted with respect to time in the display window.

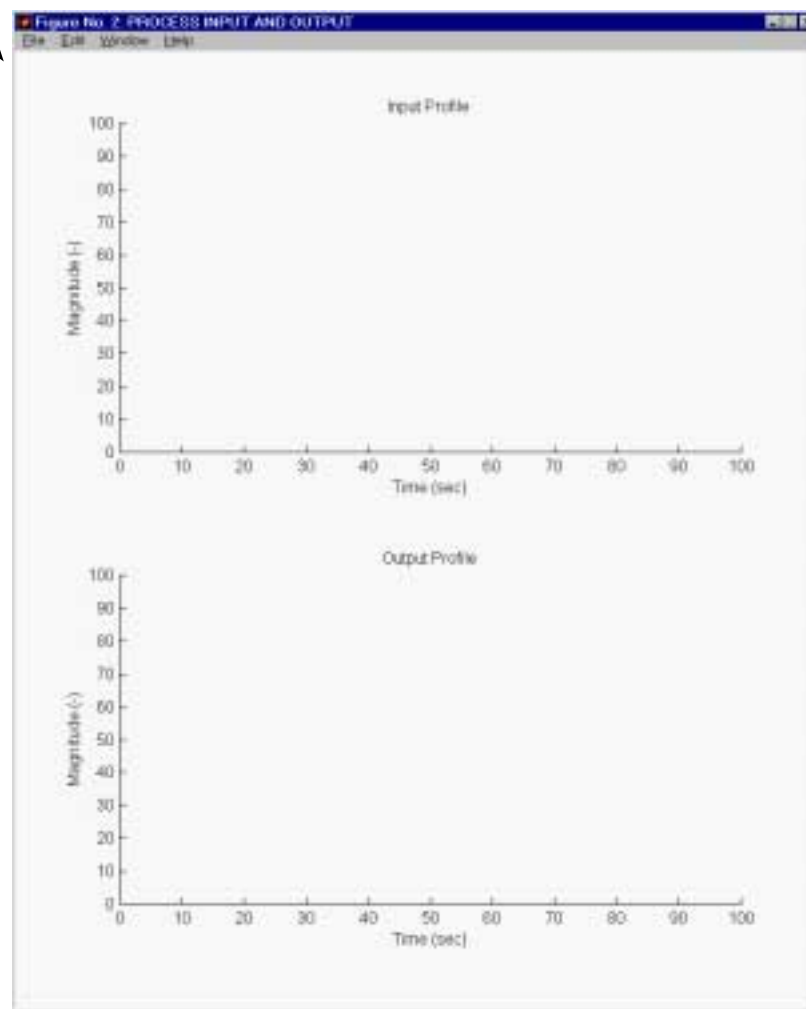


Figure 4.2. Second-Order System Display Window

$$y(s) = \frac{K_p}{As^2 + Bs + 1}$$

1. **Exercise 4.1** Calculate ζ (the damping coefficient) for the given system.
2. Set the system gain (K_p) to 10.0, the value of A to 40.0, and the value of B to 14.0 (see Figure 4.3). Now set the initial value of the Step Function to 0.0 and the final value of the Step Function to 1.0 (see Figure 4.4). To start the simulation, select *Start* from the *Simulation* menu.

Exercise 4.2 Is the system overdamped, underdamped or critically damped?

Exercise 4.3 If the system is underdamped, what is the overshoot, decay ratio, rise time, settling time and the period of oscillation?

3. Change the value of A to 18 and the value of B to 2. Repeat the simulation.

Exercise 4.4 Is the system overdamped, underdamped or critically damped?

Exercise 4.5 If the system is underdamped, what is the overshoot, decay ratio, rise time, settling time and the period of oscillation?

4. Change the value of A to 42.25 and the value of B to 13. Repeat the simulation.

Exercise 4.6 Is the system overdamped, underdamped or critically damped?

Exercise 4.7 If the system is underdamped, what is the overshoot, decay ratio, rise time, settling time and the period of oscillation?

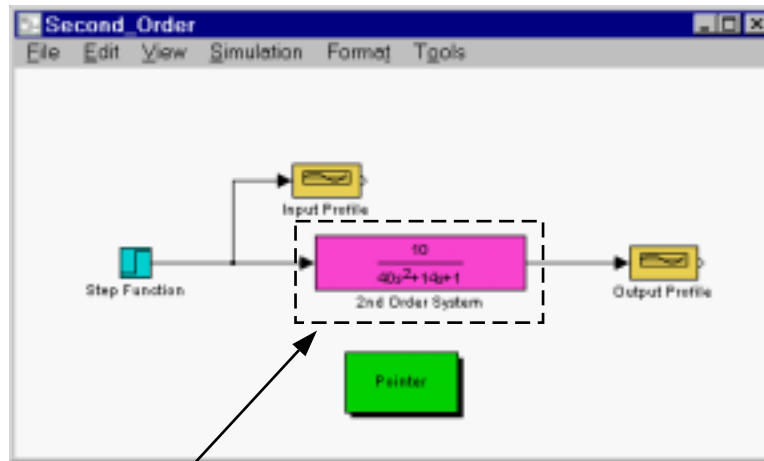
System Identification Problem

5. Close the two windows by clicking the left mouse button on the upper left-hand box of both windows and selecting *Close*. From the First and Second Order Systems Menu, select the *System Identification Problem 2* button. Using a step input, run the simulation to generate data that can be used to determine the system gain (K_p), the system time constant (τ_p), and the damping coefficient (ζ). Calculate the following quantities to characterize the system response:

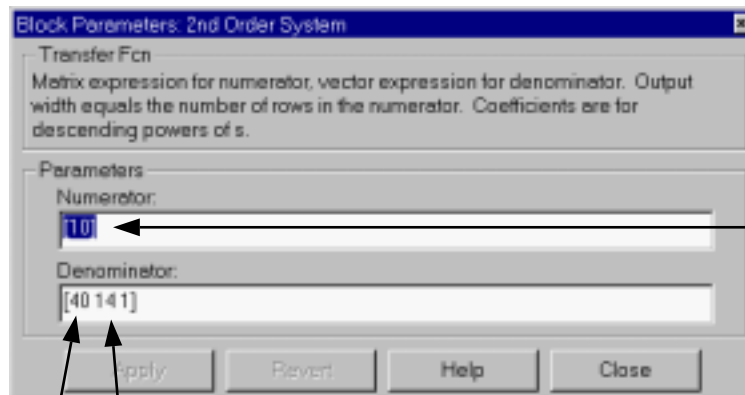
Exercise 4.8 What is the overshoot in the response?

Exercise 4.9 What is the period of the oscillatory response?

This information can be used to calculate the system gain, time constant, and damping coefficient:



By double-clicking this box, the Block Parameters window should appear. From this window you can change the system gain and system time constant.



System time constant squared

Figure 4.3. Instructions for Second-Order System Parameters

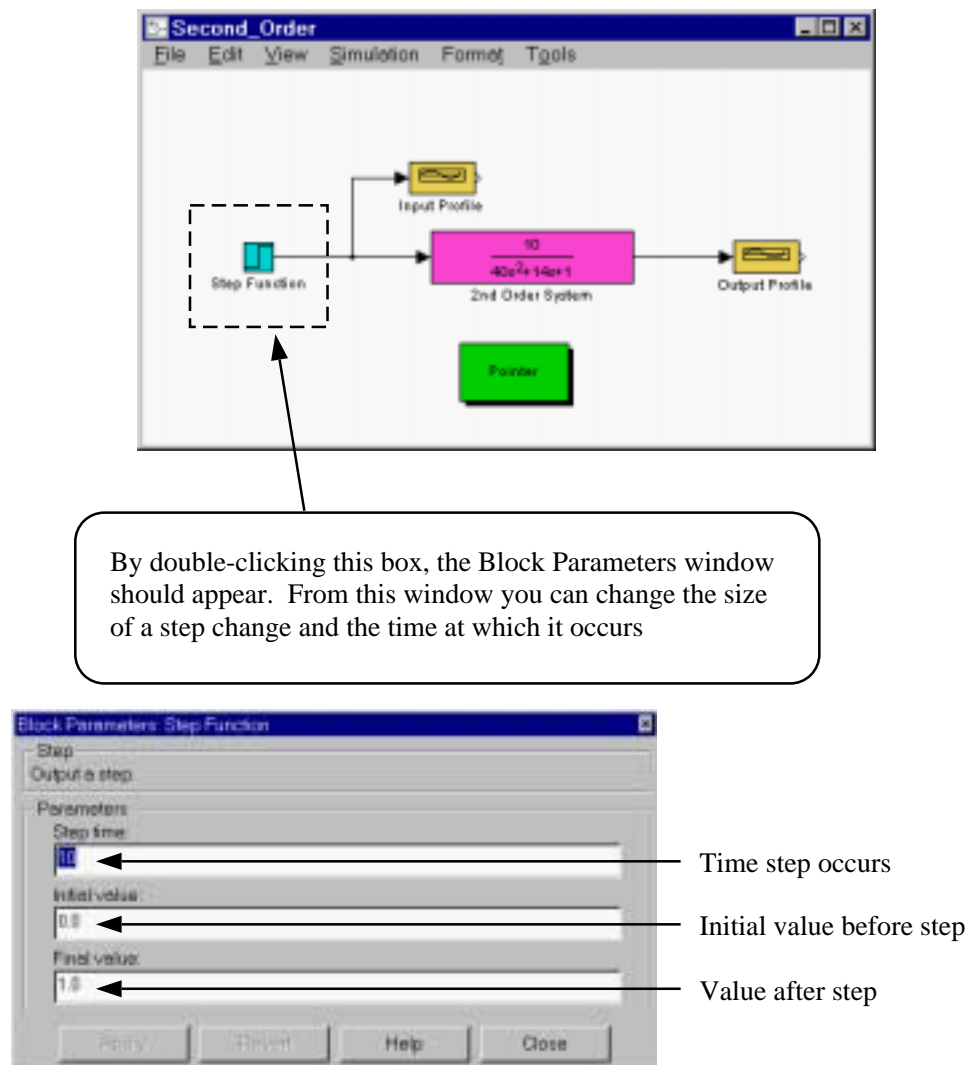


Figure 4.4. Instructions for Step Function Inputs

K_p	
τ_p	
ζ	

Hint: Use the overshoot equation and the period of oscillation equation to find ζ and τ_p :

$$\text{overshoot} = \exp\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right)$$

$$\text{period}(T) = \frac{2\pi\tau_p}{\sqrt{1-\zeta^2}}$$

Exercise 4.10 Derive the second-order transfer function for this unknown system.

SUMMARY

This unit demonstrated the dynamic characteristics of second-order systems. One should be comfortable with the effect of changing the gain, natural period, and/or damping coefficient. Finally, preliminary techniques for second-order system identification were covered.

FREQUENCY DOMAIN ANALYSIS

OBJECTIVE

In previous units, step function tests were used to develop dynamic models for the first- and second-order system identification problems. In this unit you will use pulse testing of the system identification problems and the Furnace/Column to generate frequency response data. Frequency response data, often represented in a Bode plot, are useful for describing the dynamics of a process plant and for designing stable feedback control systems.

Open-Ended Module Procedure

Construct the four open-loop Bode plots corresponding to each combination of the two manipulated inputs and the two process outputs for the Furnace/Column. For each plot, calculate the gain and phase margins for the corresponding input-output pair.

THINGS TO THINK ABOUT

Exercise 5-A What are two methods to generate frequency response data for a system?

Exercise 5-B What is the ultimate response of a stable linear system to a sinusoidal input?

Exercise 5-C What is required of the input and output to use pulse testing for identification?

Exercise 5-D What information is represented on a Bode plot?

Exercise 5-E What is the amplitude ratio and how is it determined from sinusoidal forcing data?

Exercise 5-F What is the phase shift and how is it determined from sinusoidal forcing data?

Exercise 5-G What are the gain and phase margins?

Exercise 5-H What is the significance of the crossover frequency, ω_{co} ?

Exercise 5-I Given three systems in series: $G = G_1 G_2 G_3$, where:

$$G_1(s) = \frac{0.35}{(4s + 1)} \quad G_2(s) = \frac{10}{(2s + 1)} \quad G_3(s) = \frac{5e^{-4s}}{(3s + 1)}$$

Calculate the overall transfer function, the amplitude ratio as a function of frequency, and construct the Bode plot for the overall system.

Exercise 5-J What are the gain and phase margins for the system in Exercise 5-I?

INTRODUCTION

The frequency response model of a system can be derived by the variable transformation $s = j\omega$ applied to the transfer function representation. Therefore, there is a direct relationship between the input and output time response data, and the corresponding data in the frequency domain. This relation is the *Fourier Transform*, which is given by the following expression:

$$\hat{f}(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt$$

Thus, given time domain data generated by a process, one can determine its frequency response equivalent:

$$g(j\omega) = \frac{\hat{y}(j\omega)}{\hat{u}(j\omega)} = \frac{\int_{-\infty}^{\infty} y(t)\cos\omega t dt - j \int_{-\infty}^{\infty} y(t)\sin\omega t dt}{\int_{-\infty}^{\infty} u(t)\cos\omega t dt - j \int_{-\infty}^{\infty} u(t)\sin\omega t dt}$$

Rather than doing this by hand, there are convenient functions in MATLAB to accomplish this transform. Read about *pcmf* in the Appendix to Module 5.

PROCEDURE—BOTH UNITS

The goal of this exercise is to collect frequency response data by using rectangular pulses as inputs to the first-order system, the system identification problems, and the furnace or column. With these data, Bode plots will be constructed and system information will be extracted from these plots.

Frequency Response of a First-order System

1. Select the *First-Order System* button from the *Frequency Analysis* menu. The frequency response of a first-order system is easily calculated from the transfer function using formulas found in most undergraduate texts on process dynamics and control.

Make sure that the first-order system has the following parameters:

- (a) the process gain (K_p) is set to 14.5
- (b) the system time constant (τ_p) is set to 6.5.
- (c) the magnitude of the input pulse block is set to 5.0.

The frequency response of a first-order system can be calculated analytically using the procedure available in many undergraduate texts (*e.g.*, pages 281-282 of Ogunnaike and Ray). The key steps are summarized below.

2. Generate a frequency vector which encompasses the entire range of interest for the given system.
 $\gg w = \text{logspace}(-2, 1, 100)'$
3. Substitute $s = j\omega$ into the transfer function, and then manipulate $g(j\omega)$ until you can factor it such that:

$$g(j\omega) = Re(\omega) + jIm(\omega)$$

$Re(\omega)$ is the real component of the transfer function, $g(j\omega)$, and $Im(\omega)$ represents the imaginary component.

4. In the MATLAB workspace, calculate the amplitude ratio and phase lag using the following equations:

$$AR = \sqrt{Re(\omega)^2 + Im(\omega)^2}$$

$$\phi = \text{atan} \left[\frac{Im(\omega)}{Re(\omega)} \right]$$

NOTE: *atan* is the MATLAB function for \tan^{-1} , and *atan* returns phase in radians. You will have to convert this to degrees before you can use *plot_bode*.

5. **Exercise 5.1** Produce the Bode plot for this system using the *plot_bode* command in the Appendix to Module 5.

General Pulse Testing Procedure

To eliminate the tedium of calculating frequency response data by hand one frequency at a time, and to generate bode plots of unmodeled systems, we can use pulse response tests. The purpose of the following exercise is to demonstrate a method for obtaining frequency response information using a pulse forcing of the system.

6. Select *Start* to begin the simulation.
 - (a) Observe the simulation and verify that the output profile returns to, and holds at, zero.
 - (b) Type *whos* in the MATLAB window. A list of vectors containing *input*, *output*, and *time* should appear. The vector *input* represents the pulse input to the system of interest. The vector *output* is the system response to the input pulse.
 - (c) These vectors will be used to calculate the amplitude ratio and the phase lag using *pcmfft*. The integration step size for the first order system and the system identification simulations is 0.05 sec.
 - (d) Using *A1*, *A2*, *B1*, and *B2* generated by *pcmfft*, amplitude ratio and phase lag can be calculated using the following equations:

$$AR(\omega) = \sqrt{\frac{A1^2 + A2^2}{B1^2 + B2^2}} \quad Re[g(j\omega)] = \frac{A1 * B1 + A2 * B2}{B1^2 + B2^2}$$

$$\phi(\omega) = \tan^{-1} \left(\frac{Im[g(j\omega)]}{Re[g(j\omega)]} \right) \quad Im[g(j\omega)] = \frac{A1 * B2 - A2 * B1}{B1^2 + B2^2}$$

7. Examine the output power spectrum to determine the point after which the data contains no useful information.

Exercise 5.2 Construct the Bode plot for the first-order system using the *plot_bode* command as described in the Appendix to Module 5.

Exercise 5.3 At approximately what frequency does the output power become insignificant?

Exercise 5.4 Find the corner frequency for this system.

Frequency Response of Unknown Systems

8. Select the *System Identification Problem 1* from the Frequency Response Analysis Menu and repeat the General Pulse Testing Procedure that was described above.

Exercise 5.5 Construct the Bode plot for *System Identification Problem 1*.

Exercise 5.6 Assuming that this is a first-order system, calculate the gain and the time constant.

9. Now select the *System Identification Problem 2* from the Frequency Response Analysis Menu and repeat the procedure above.

Exercise 5.7 Construct the Bode plot for *System Identification Problem 2*.

Exercise 5.8 Assuming that this is a second-order system, calculate the gain, the time constant, and the damping coefficient.

PROCEDURE—FURNACE

The goal of this procedure is to obtain the system model dependent parameters (from Bode plots) in order to tune the controllers for the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration*. Several methods exist for tuning controllers based upon the open-loop behavior of the plant (most of which will be studied in Module 8); however, the frequency-based methods tend to be more reliable than other heuristics and serve as the basis for the auto-tuning controllers available in industry today.

1. Select the *Furnace* from the Frequency Response Analysis Menu. This will bring up the furnace simulation driven by pulse inputs. Displayed on your screen are the flowsheet and the furnace strip chart on which the inputs and outputs of interest are recorded. The goal of this exercise is to use pulse inputs for both the *Air Flow Rate* and the *Fuel Gas Flow Rate*, and to collect the frequency response data from the outputs.
2. First, set the *Air Flow Switch* to 0 and the *Fuel Switch* to 1. **BE CAREFUL:** some of the outputs decrease with an increase in corresponding inputs (negative gain). Begin the simulation, and follow the General Procedure for Pulse Testing, given above.

Exercise 5.9 With the data, calculate the appropriate amplitude ratios and the phase lags and construct the Bode plot of *Oxygen Exit Concentration* for changes in the *Fuel Gas Flow Rate*.

Exercise 5.10 Construct the Bode plot for the *Hydrocarbon Outlet Temperature* response to the *Fuel Gas Flow Rate*.

3. Now change the *Fuel Switch* to 0 and the *Air Flow Switch* to 1. Start the simulation, and follow the generated pulse testing procedure above.

Exercise 5.11 Construct the Bode plot for the *Oxygen Exit Concentration* response to the *Air Flow Rate* input change.

Exercise 5.12 Construct the Bode plot for the *Hydrocarbon Outlet Temperature* response to the *Air Flow Rate* input change.

PROCEDURE—COLUMN

The goal of this procedure is to examine the frequency response behavior for high-order systems. This information is useful in controller design as well. Several methods exist for tuning controllers based upon the open-loop behavior of the plant (most of which will be studied in Module 8); however, the frequency-based methods tend to be more reliable than other heuristics and serve as the basis for the auto-tuning controllers available in industry today.

1. Select the *Column* from the Frequency Response Analysis Menu. This will bring up the column simulation driven by pulse inputs. Displayed on your screen are the flowsheet and the column strip chart on which the inputs and outputs of interest are recorded. The goal of this exercise is to use pulse inputs for both the *Vapor Flow Rate* and the *Reflux Ratio*, and to collect the frequency response data from the outputs.
2. First, set the *Reflux Ratio Switch* to 0 and the *Vapor Switch* to 1. **BE CAREFUL:** some of the outputs may decrease with an increase in corresponding inputs (negative gain). Begin the simulation, and follow the General Procedure for Pulse Testing, given above.

Exercise 5.13 With this data, calculate the appropriate amplitude ratios and the phase lags and construct the Bode plots of the *Overhead MeOH Composition* and *Bottom MeOH Composition* for the *Vapor Flow Rate* loop.

Exercise 5.14 Calculate the gain margin and phase margin from the plot for *Overhead MeOH Composition*.

3. Now change the *Vapor Switch* to 0 and the *Reflux Ratio Switch* to 1. Start the simulation, and follow the generated pulse testing procedure, above. Using this data, calculate the amplitude ratios and phase lags.

Exercise 5.15 Construct the Bode plots for the *Overhead MeOH Composition* and *Bottom MeOH Composition* for the *Reflux Ratio* loop.

Exercise 5.16 Calculate the gain margin and phase margin from the plots for *Bottom MeOH Composition*.

SUMMARY

This unit demonstrated the construction of an open-loop Bode plot, which depicts the frequency characteristics for a given input-output pair in terms of its

frequency-dependent magnitude and phase lag. One should be able to interpret the key characteristics of the curves, including the meaning of gain margin and phase margin.

APPENDIX: `plot_bode` AND `pcmfift` COMMANDS

`plot_bode`

To facilitate the process of constructing bode plots, we have defined a function in MATLAB called `plot_bode` which takes an $n \times 3$ matrix of data and plots the amplitude ratio and phase lag as functions of frequency. To use this command, you will need to construct an $n \times 3$ matrix with the first column being the frequency, the second column as the amplitude ratio, and the third column containing the phase lag. This can be accomplished by defining a variable a in the following manner (where w , amp , and phi are column vectors):

```
>> a=[w,amp,phi];
```

The first entry of each row is the frequency, the second entry of each row is the amplitude ratio, and the third entry of each row is the phase lag. At the MATLAB prompt, type `plot_bode(a)`:

```
>> plot_bode(a)
```

This will display the bode plots for the given data. In order to print this plot, select *File* and then *Print* from the figure window.

`pcmfift`

To ease the mathematical burden of taking Fourier transforms, we have defined a function in MATLAB called `pcmfift`. After generating the input-output data by running the simulations, a call of this function will complete the Fast Fourier Transform (FFT). The proper syntax is:

```
>> [A1, A2, B1, B2, w] = pcmfift(input,output,switch)
```

The variables *input* and *output* are vectors generated by the simulation and *switch* is a variable that should be set to 0 for the first-order system and system identification section, and be equal to 1 while working on the furnace. The outputs are ω , the frequency vector, and $A1$, $A2$, $B1$, and $B2$, which are each components of the FFT such that:

$$A1 = \int_0^{T_y} y(t) \cos(\omega t) dt$$

$$A2 = \int_0^{T_y} y(t) \sin(\omega t) dt$$

$$B1 = \int_0^{T_u} u(t) \cos(\omega t) dt$$

$$B1 = \int_0^{T_u} u(t) \sin(\omega t) dt$$

The integration lower bound is 0 (instead of $-\infty$) because we are interested only in time ≥ 0 . The upper bound is chosen such that the input is zero after T_u and the output is zero after T_y . This property, that both the inputs and outputs return to zero in finite time, is required to perform the FFT. Once the A and B vectors have been calculated, equations (such as those given in part e of the Generalized Pulse Testing Procedure) can be used to calculate the amplitude ratio and phase lag for the system of interest.

Module 6

TRANSIENT RESPONSE ANALYSIS

OBJECTIVE

In this module, transient response data will be collected which will be used to obtain a first-order-plus-time-delay transfer function model of the Furnace/Column. This model will later serve as the basis for several types of feedback and feedforward controller design.

Open-Ended Module Procedure

Calculate the first-order-plus-time-delay approximations for the relevant input-output pairings for the Furnace/Column, filling in all the values in Figure 6.2/6.3. Use SIMULINK to validate the resultant models against novel process data.

THINGS TO THINK ABOUT

Exercise 6-A Fill in the following table describing the relationship between the system time constant and the final value of the output for a linear first-order system (recall that the time constant is defined as the time that it takes for an output to attain approximately 63% of its final response).

Time Elapsed	τ_p	$2\tau_p$	$3\tau_p$	$4\tau_p$
$y(t)$ as a percent of its final value				

Exercise 6-B Briefly define time delay. Give an example that illustrates how time delay occurs in a process system.

Exercise 6-C Describe how one would obtain a system gain from dynamic response data generated by introducing a step change in one of the process inputs.

INTRODUCTION

Typically, the transient response of a plant (even a nonlinear process such as the furnace or column) can be approximately modeled with a first-order-plus-time-delay (FOTD) transfer function. The mathematical description of a first-order-plus-time-delay transfer function is as follows:

$$G_p(s) = \frac{K_p e^{-\theta s}}{\tau_p s + 1}$$

where: K_p is the system gain, θ is the time delay of the system, and τ_p is the system time constant.

Once a step change has been made in one of the inputs, the system gain, time constant, and the time delay may be calculated from the system response. The method for obtaining these parameters is described in most undergraduate chemical process dynamics and control textbooks.

PROCEDURE—FURNACE

This module is divided into three parts: (i) data collection, (ii) empirical modeling, and (iii) model validation. Customized MATLAB utilities have been designed to simplify each task. At the beginning of each part, a short summary of the procedure will be given. Please read these instructions carefully.

Data Collection

In this section, the data required for obtaining a first-order-plus-time-delay transfer function model of the furnace will be collected. Four runs will be conducted. Each run will consist of:

1. Changing one of the input variables
2. Recording the time at which the input was changed
3. Allowing the furnace to reach a new steady state
4. Saving the four outputs to a data file
5. Returning the input variables to their initial values
6. Allowing the furnace to return to its initial steady state

Before changing any inputs, make sure that the clock display is **open**. To open the clock display, double-click the left mouse button on the yellow clock icon located in the upper right corner of the furnace diagram, see Figure 2.2. The clock display is the small window showing the current process simulation time. **Make sure to arrange your windows such that the clock display is visible at all times.**

The first input variable that will be changed is the *Air Flow Rate* (from 17.9 to $18.26 \frac{m^3}{min}$). A detailed explanation of the data collection procedure will be given for this run.

- (a) Open the clock display. Make sure to arrange the windows such that the clock display is completely visible.
- (b) Choose a filename for the data from this run to be saved by typing a filename in the box located in the lower right corner of the process monitor, see Figure 2.3. In the MATLAB command window, use the *cd* command to change directories to a directory where you would like to store your data files. Use *pwd* to print the working directory that you are currently in. Use *ls* or *dir* to show the contents of the current directory.
- (c) Record the filename for the current data run.
- (d) Double-click on the *Air Flow Rate* box with the left mouse button.
- (e) Double-click in the white box to highlight the value to be changed.
- (f) Record the time at which the *Air Flow Rate* change is implemented. **Be careful: the *Air Flow Rate* change is not implemented until you hit *apply* or *close* with the mouse.** You should notice a slight pause in the simulation at the time in which the *Air Flow Rate* change is implemented.
- (g) Once the furnace has reached steady state, click on the *Save* button located in the lower right corner of the process monitor, see Figure 2.3, to save the data to a data file. **Be sure to act promptly, as any data that is lost when the axes rescale is not recoverable.**
- (h) Return the *Air Flow Rate* to its original value ($17.9 \frac{m^3}{min}$) and allow the furnace to return to its original steady state before making the next input change. **You must allow the furnace to return to its original steady state before making any further changes in the input variables.**

Repeat Steps (b)–(h) for subsequent runs.

CAUTION: Choose a different filename for each of the four runs. Otherwise, the data from previous runs will be lost. You may be able to save your data in a temporary directory (C:/temp or /tmp)

1. Increase the value of the *Air Flow Rate* by 2% (from 17.9 to 18.26 $\frac{m^3}{min}$). The furnace should reach a new steady state in approximately 40 simulation minutes.

Exercise 6.1

Name of data file for this run _____(example: *airflow*).

Time at which the *Air Flow Rate* change is introduced _____min (from clock display).

2. Increase the value of the *Fuel Gas Flow Rate* by 2% (from 1.21 to 1.234 $\frac{m^3}{min}$). The furnace should reach a new steady state in approximately 40 simulation minutes.

Exercise 6.2

Name of data file for this run _____(example: *fuelgas*).

Time at which the *Fuel Gas Flow Rate* change is introduced _____ min (from clock display).

3. Increase the value of the *Hydrocarbon Flow Rate* by 2% (from 0.035 to 0.0357 $\frac{m^3}{min}$). The furnace should reach a new steady state in approximately 40 simulation minutes.

Exercise 6.3

Name of data file for this run _____ (example: *hydrocarbonflow*).

Time at which the *Hydrocarbon Flow Rate* change is introduced _____ min (from clock display).

4. Decrease the value of the *Fuel Gas Purity* by 2% (from 1.0 to 0.98). The furnace should reach a new steady state in approximately 40 simulation minutes.

Exercise 6.4

Name of data file for this run _____ (example: *gaspurity*).

Time at which the *Fuel Gas Purity* change is introduced _____ min (from clock display).

Modeling the Furnace

The data files collected in the previous portion of this module will be loaded into a specially designed graphical utility, called *PCMplot*, to obtain transient plots of the output variables. To start *PCMplot*, return to main menu and click on the *Graphical Analysis* button in the lower right corner. A new menu, the Graphical Analysis Menu, will appear in the center of your screen. Click on the *PCMplot – Furnace* button to open *PCMplot*. The window depicted in Figure 6.1 will appear.

The procedure for loading a data file will be summarized below.

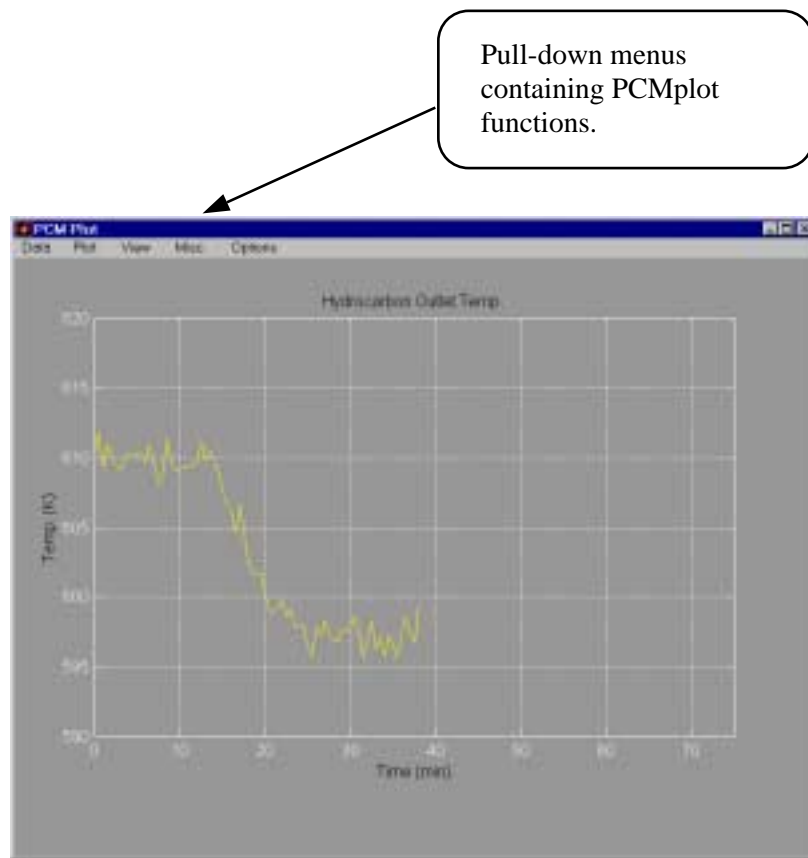


Figure 6.1. PCMplot Graphics Window

5. Click on the *Data* option on the graphics window, see Figure 6.1, and hold down the left mouse button. By moving the mouse downward, a menu with the following options will appear: *Load*, *Print*, and *Quit*. To load a file, continue holding the left mouse button and pull the mouse down to highlight the *Load* option. Release the mouse button.
 - (a) A window titled *Load Data File* should appear. Click on the data file you wish to load.
 - (b) If the file loaded successfully, the following text will be displayed in your MATLAB window:
`load selected filename`
 - (c) Once the data file has been loaded, the output variable plot is generated by clicking on the *Plot* option at the top of the graphics window with your left mouse button. Hold the left mouse button down and pull the mouse down to obtain the following options: (i) Plot Var 1, (ii) Plot Var 2, (iii) Plot Var 3, and (iv) Plot Var 4. These options are responsible for plotting the four outputs as follows:
 - i. Plot Var 1 = Plot *Hydrocarbon Outlet Temperature*
 - ii. Plot Var 2 = Plot *Furnace Temperature*
 - iii. Plot Var 3 = Plot *Exhaust Gas Flow Rate*
 - iv. Plot Var 4 = Plot *Oxygen Exit Concentration*
 - (d) To plot the change in the *Hydrocarbon Outlet Temperature* for the airflow data file, hold the left mouse button down and highlight *Plot Var 1*. Release the mouse button. A graph of the transient response of the *Hydrocarbon Outlet Temperature* should appear.
 - (e) It may be necessary to rescale the axes on the plot. This can be done using the *Trim Axes* option on the *View* menu of the graphics window (Hold your left mouse button on *View* and pull down as you have previously done to see this menu). You will be prompted to enter new values for the upper and lower limits for both the x-axis and the y-axis. **Hint: it may be helpful to set the x-axis lower limit equal to the time at which the change in the given input was introduced.**
 - (f) Once a plot of the desired output is obtained with the axes scaled as desired, a hard copy of the plot can be obtained using the *Print* option under the *File* menu (hold down the left mouse button and pull down underneath *File* on the graphics window. Highlight the *Print* option and release the left mouse button). This will open the print options window. Selecting *OK* (PC) or *Print* (UNIX) will print the current figure to the default printer.
6. Plot the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* for the *Air Flow Rate* change using *PCMplot* and the procedure described

above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.2.

Exercise 6.5 *Hydrocarbon Outlet Temperature, G_{12}*

Gain:

Time Constant:

Time Delay:

Exercise 6.6 *Oxygen Exit Concentration, G_{22}*

Gain:

Time Constant:

Time Delay:

7. Plot the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* for the *Fuel Gas Flow Rate* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.2.

Exercise 6.7 *Hydrocarbon Outlet Temperature, G_{11}*

Gain:

Time Constant:

Time Delay:

Exercise 6.8 *Oxygen Exit Concentration, G_{21}*

Gain:

Time Constant:5

Time Delay:

8. Plot the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* for the *Hydrocarbon Flow Rate* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.2.

Exercise 6.9 *Hydrocarbon Outlet Temperature, G_{1d_2}*

Gain:

Time Constant:

Time Delay:

Exercise 6.10 *Oxygen Exit Concentration, G_{2d_2}*

Gain:

Time Constant:

Time Delay:

9. Plot the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* for the *Fuel Gas Purity* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.2.

Exercise 6.11 *Hydrocarbon Outlet Temperature, G_{1d_1}*

Gain:

Time Constant:

Time Delay:

Exercise 6.12 *Oxygen Exit Concentration, G_{2d_1}*

Gain:

Time Constant:

Time Delay:

Validation of Furnace Models

The third step of this module is to validate the performance of the first-order-plus-time-delay transfer function models obtained in the previous section. The input changes from the modeling runs of the first section can be used with the linear models developed in the last section. You will compare the simulated response of the linear models to the actual plant data to determine if the modeling parameters developed in the last section are accurate. In Module 7, controllers will be designed based upon the transfer function models obtained in this module. The effectiveness of these controllers will be significantly influenced by the accuracy of the transfer function models obtained in this module. The transfer function models will be simulated using a specially modified version of SIMULINK, the MATLAB-based object oriented programming software for dynamic simulations. SIMULINK can be started from the Graphical Analysis Menu by clicking on the *Simulink – Furnace* button. Clicking on this button creates two windows, a blank SIMULINK window with a save button (where the simulation is constructed) and the SIMULINK toolbox window (where programming elements are located). The results of the simulations will be saved into data files and loaded into *PCMplot* for comparison with the original plant data.

10. Construct a SIMULINK simulation to compare the modeled response of the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* for a 2% change in the *Air Flow Rate* (from 17.9 to 18.26 $\frac{m^3}{min}$), $G_{12}(s)$, and $G_{22}(s)$, with the actual plant data. Be aware that your transfer function model relates deviation variables while the plant data is not in deviation form.

Your SIMULINK simulation should consist of the following elements:

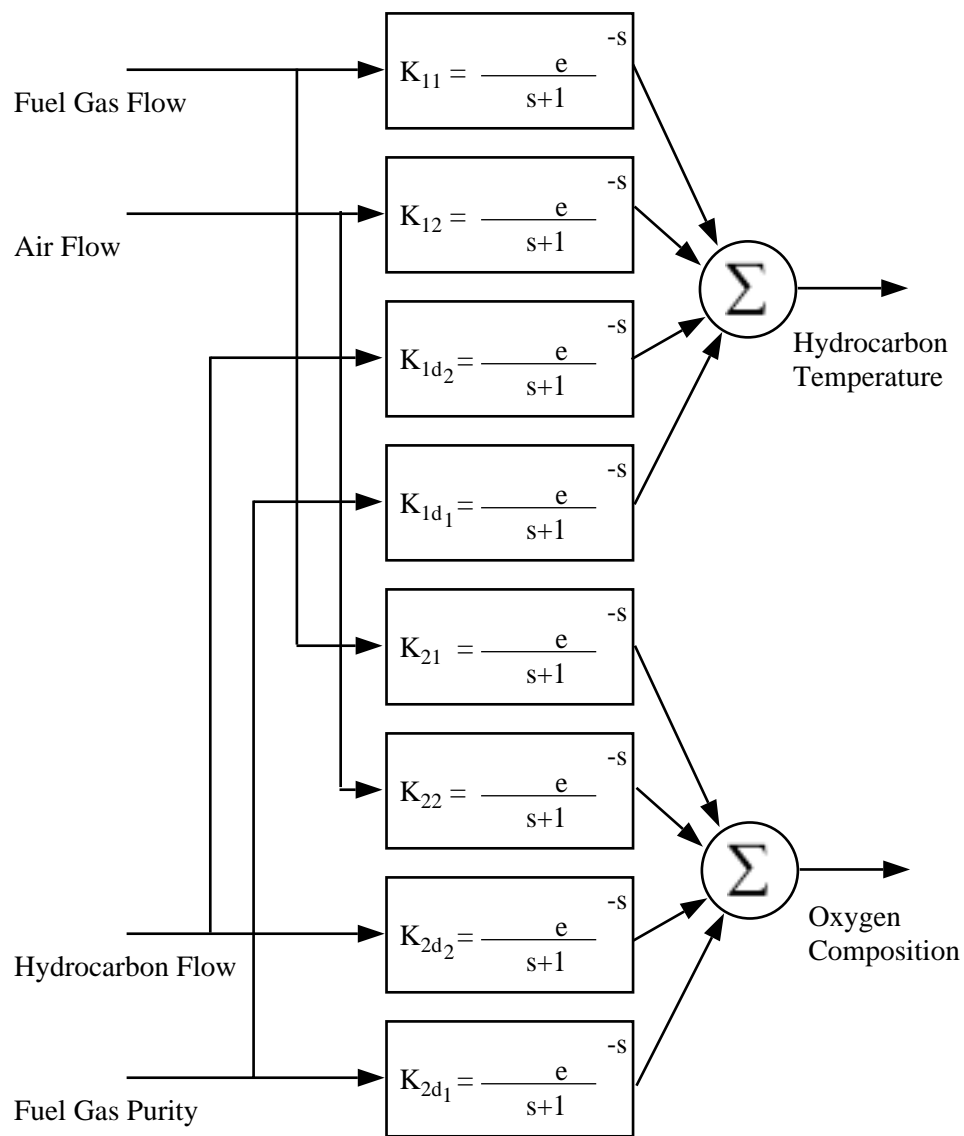


Figure 6.2. Block Diagram for Transfer Function Model of the Furnace

- (a) clock
- (b) step input
- (c) *Hydrocarbon Outlet Temperature* (to workspace)
- (d) *Oxygen Exit Concentration* (to workspace)
- (e) time (to workspace)
- (f) 2 transfer functions and 2 transport delays or 2 transfer functions (with time delay)
- (g) 2 sums
- (h) 2 constants

Run a simulation of your transfer function model and save the data into a data file. Do not repeat filenames from any previous runs.

Exercise 6.13 Filename: _____ (example: *airflow_val*).

Note: Make sure that you do not change the names of the variables in any of the *to workspace* blocks. A procedure has been written to interface the data collected from the validation experiments with *PCMplot* which is name specific.

11. Using the same SIMULINK block diagram from the previous exercise compare the modeled response of the *Hydrocarbon Outlet Temperature* and the *Oxygen Exit Concentration* to a 2% change in the *Fuel Gas Flow Rate* (from 1.21 to $1.234 \frac{m^3}{min}$); $G_{11}(s)$, and $G_{21}(s)$; with the actual plant data. Be aware that your transfer function model relates deviation variables while the plant data is not in deviation form.

Run a simulation of your transfer function model and save the results into a data file. Do not repeat filenames from any previous runs.

Exercise 6.14 Filename: _____ (example: *fuelflow_val*)

Note: Make sure that you do not change the names of the variables in any of the *to workspace* blocks. The system which interfaces the data collected from the validation experiments with *PCMplot* is name specific.

Save your SIMULINK simulation window for use later in this section.

12. To compare the data from the SIMULINK simulations with the plant data, *PCMplot* will be used in the validation mode. The validation mode loads the plant data and the data from the SIMULINK simulations sequentially and overlays the plots for comparison.

- (a) After starting *PCMplot*, turn the validate mode on by:
 - i. pulling down the *Misc* menu from the graphic display, see Figure 6.1,

- ii. highlight the *validate* option with the mouse button still depressed and
 - iii. pull the mouse to the right of *validate* and highlight the *on* option.
- (b) First, load the plant data file using the *Load* option and the prompts (remember to include the single quotes where specified).
 - (c) Using the *Plot* option, plot the *Hydrocarbon Outlet Temperature (plot var 1)*.
 - (d) Using the *Plot* option, plot the *Oxygen Exit Concentration (plot var 4)*.
 - (e) Use the *Load* option a second time to load the data from the SIMULINK simulation of the transfer function model.
 - (f) Using the *Plot* option, plot the *Hydrocarbon Outlet Temperature (plot var 1)*. The plot of the data from the SIMULINK simulation of the transfer function will automatically overlay the plant data.
 - (g) Pull down the *View* menu and select *Original Axes*.
 - (h) It may be necessary to utilize the *Trim Axes* option to view the plots.
 - (i) Print the graph using the *Print* option.
 - (j) Using the *Plot* option, plot the *Oxygen Exit Concentration (plot var 4)*.
 - (k) Select *Original Axes* under the *View* menu.
 - (l) It may be necessary to utilize the *Trim Axes* option to view the plots.
 - (m) **Exercise 6.15** Print the graph using the *Print* option.
13. How well do the models fit the dynamic plant data? Modify the parameters, if necessary, to obtain better agreement. Rerun the SIMULINK simulation and replot the data. Include these plots in your report.
14. Repeat the validation sequence for each transfer function (8 overall). Validated models will be needed for later lab units.

Exercise 6.16 Create the seven remaining transfer function validation plots.

PROCEDURE—COLUMN

This module is divided into three parts: (i) data collection, (ii) modeling, and (iii) validation. Customized MATLAB utilities have been designed to simplify each task. At the beginning of each part, a short summary of the procedure will be given. Please read these instructions carefully.

Data Collection

In this section, the data required for obtaining a first-order-plus-time-delay transfer function model of the column will be collected. Four runs will be conducted. Each run will consist of:

1. Changing one of the input variables
2. Recording the time at which the input was changed
3. Allowing the column to reach a new steady state
4. Saving the four outputs to a data file
5. Returning the input variables to their initial values
6. Allowing the column to return to its initial steady state

Before changing any inputs, make sure that the clock display is **open**. To open the clock display, double-click the left mouse button on the yellow clock icon located in the upper right corner of the column diagram, see Figure 2.9. The clock display is the small window showing the current process simulation time. **Make sure to arrange your windows such that the clock display is visible at all times.**

The first input variable that will be changed is the *Feed Flow Rate* (from 0.025 to $0.027 \frac{m^3}{sec}$). A detailed explanation of the data collection procedure will be given for this run.

- (a) Open the clock display. Make sure to arrange the windows such that the clock display is completely visible.
- (b) Choose a filename for the data from this run to be saved by typing a filename in the box located in the lower right corner of the process monitor, see Figure 2.10. Using the MATLAB command window, use the *cd* command to change directories to a directory where you would like to store your data files. Use *pwd* to print the working directory that you are currently in. Use *ls* or *dir* to show the contents of the current directory.
- (c) Record the filename for the current data run.
- (d) Double-click on the *Feed Flow Rate* box with the left mouse button.
- (e) Double-click in the white box to highlight the value to be changed.
- (f) Record the time at which the *Feed Flow Rate* change is implemented. **Be careful: the *Feed Flow Rate* change is not implemented until you hit *apply* or *close* with the mouse.** You should notice a slight pause in the simulation at the time in which the *Feed Flow Rate* change is implemented.

- (g) Once the column has reached steady state, click on the *Save* button located in the lower right corner of the process monitor, see Figure 2.10, to save the data to a data file. **Be careful not to wait too long because any data that is lost when the axes rescale is not recoverable.**
- (h) Return the *Feed Flow Rate* to its original value ($0.025 \frac{m^3}{sec}$) and allow the column to return to its original steady state before making the next input change. **You must allow the column to return to its original steady state before making any further changes in the input variables.**

Repeat Steps (b)–(h) for subsequent runs.

CAUTION: Choose a different filename for each of the four runs. Otherwise, the data from previous runs will be lost. You may be able to save your data in a temporary directory (C:/temp or /tmp)

1. Increase the value of the *Feed Flow Rate* by 2% (from 0.025 to $0.0255 \frac{m^3}{sec}$). The column should reach a new steady state in approximately 6000 simulation seconds.

Exercise 6.1

Name of data file for this run _____(example: *feedflow*).

Time at which the *Feed Flow Rate* change is introduced _____sec (from clock display).

2. Decrease the value of the *Feed MeOH Composition* by 4% (from 0.5 to $0.48 \frac{mol}{mol\ total}$). The column should reach a new steady state in approximately 6000 simulation seconds.

Exercise 6.2

Name of data file for this run _____(example: *feedconc*).

Time at which the *Feed MeOH Composition* rate change is introduced _____sec (from clock display).

3. Increase the value of the *Vapor Flow Rate* by 3% (from 0.033 to $0.03399 \frac{mol}{sec}$). The column should reach a new steady state in approximately 6000 seconds.

Exercise 6.3

Name of data file for this run _____ (example: *vaporflow*).

Time at which the *Vapor Flow Rate* change is introduced _____ sec (from clock display).

4. Increase the value of the *Reflux Ratio* by 2% (from 1.75 to 1.785). The column should reach a new steady state in approximately 6000 simulation seconds.

Exercise 6.4

Name of data file for this run _____ (example: *reflux*).

Time at which the *Reflux Ratio* change is introduced _____ sec (from clock display).

Modeling the Column

The data files collected in the previous portion of this module will be loaded into a specially designed graphical utility, called *PCMplot* to obtain transient plots of the output variables. To start *PCMplot*, return to main menu and click on the *Graphical Analysis* button in the lower right corner. A new menu, the Graphical Analysis Menu, will appear in the center of your screen. Click on the *PCMplot - Column* button to open *PCMplot*. A graph similar to the one shown in Figure 6.1 will appear.

The procedure for loading a data file will be summarized below.

5. Click on the *Data* option on the graphics window, see Figure 6.1, and hold down the left mouse button. By moving the mouse downward, a menu with the following options will appear: *Load*, *Print*, and *Quit*. To load a file, continue holding the left mouse button and pull the mouse down to highlight the *Load* option. Release the mouse button.
 - (a) A window titled *Load Data File* should appear. Click on the data file you wish to load.
 - (b) If the file loaded successfully, the following text will be displayed in your MATLAB window:
`load selected filename`
 - (c) Once the data file has been loaded, one can plot one of the output variables by clicking on the *Plot* option at the top of the graphics window with your left mouse button. Hold the left mouse button down and pull the mouse down to obtain the following options: (i) Plot Var 1, (ii) Plot Var 2, (iii) Plot Var 3, and (iv) Plot Var 4. These options are responsible for plotting the four outputs as follows:
 - i. Plot Var 1 = Plot *Overhead Flow Rate*
 - ii. Plot Var 2 = Plot *Overhead MeOH Composition*
 - iii. Plot Var 3 = Plot *Bottom Flow Rate*
 - iv. Plot Var 4 = Plot *Bottom MeOH Composition*
 - (d) To plot the change in the *Overhead MeOH Composition* for the *feedflow* data file, hold the left mouse button down and highlight *Plot Var 1*. Release the mouse button. A graph of the transient response of the *Overhead MeOH Composition* should appear.

- (e) It may be necessary to rescale the axes on the plot. This can be done using the *Trim Axes* option on the *View* menu of the graphics window (Hold your left mouse button on *View* and pull down as you have previously done to see this menu). You will be prompted to enter new values for the upper and lower limits for both the x-axis and the y-axis. **Hint: it may be helpful to set the x-axis lower limit equal to the time at which the change in the given input was introduced.**
 - (f) Once a plot of the desired output is obtained with the axes scaled as desired, a hard copy of the plot can be obtained using the *Print* option under the *File* menu (hold down the left mouse button and pull down underneath *File* on the graphics window. Highlight the *Print* option and release the left mouse button). This will open the print options window. Selecting *OK* (PC) or *Print* (UNIX) will print the current figure to the default printer.
6. Plot the *Overhead MeOH Composition* and the *Bottom MeOH Composition* for the *Vapor Flow Rate* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.3.

Exercise 6.5 *Overhead MeOH Composition, G_{12}*

Gain:

Time Constant:

Time Delay:

Exercise 6.6 *Bottom MeOH Composition, G_{22}*

Gain:

Time Constant:

Time Delay:

7. Plot the *Overhead MeOH Composition* and the *Bottom MeOH Composition* for the *Reflux Ratio* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.3.

Exercise 6.7 *Overhead MeOH Composition, G_{11}*

Gain:

Time Constant:

Time Delay:

Exercise 6.8 *Bottom MeOH Composition, G_{21}*

Gain:

Time Constant:

Time Delay:

8. Plot the *Overhead MeOH Composition* and the *Bottom MeOH Composition* for the *Feed Flow Rate* change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.3.

Exercise 6.9 *Overhead MeOH Composition, G_{1d_2}*

Gain:

Time Constant:

Time Delay:

Exercise 6.10 *Bottom MeOH Composition, G_{2d_2}*

Gain:

Time Constant:

Time Delay:

9. Plot the *Overhead MeOH Composition* and the *Bottom MeOH Composition* for the *Feed MeOH Composition* Change using *PCMplot* and the procedure described above. Fit a first-order-plus-time-delay transfer function model to each of the responses. Fill in your parameters below and on Figure 6.3.

Exercise 6.11 *Overhead MeOH Composition, G_{1d_1}*

Gain:

Time Constant:

Time Delay:

Exercise 6.12 *Bottom MeOH Composition, G_{2d_1}*

Gain:

Time Constant:

Time Delay:

Validation of Column Models

The third step of this module is to validate the performance of the first-order-plus-time-delay transfer function models obtained in the previous section. The input changes from the modeling runs of the first section can be used with the linear models developed in the last section. You will compare the simulated response of the linear models to the actual plant data to determine if the modeling parameters developed in the last section are accurate. In the next Module, controllers will be designed based upon the transfer function models obtained in this module. The effectiveness of these controllers will be significantly influenced by the accuracy of the transfer function models obtained in this module. The transfer function models will be simulated using a

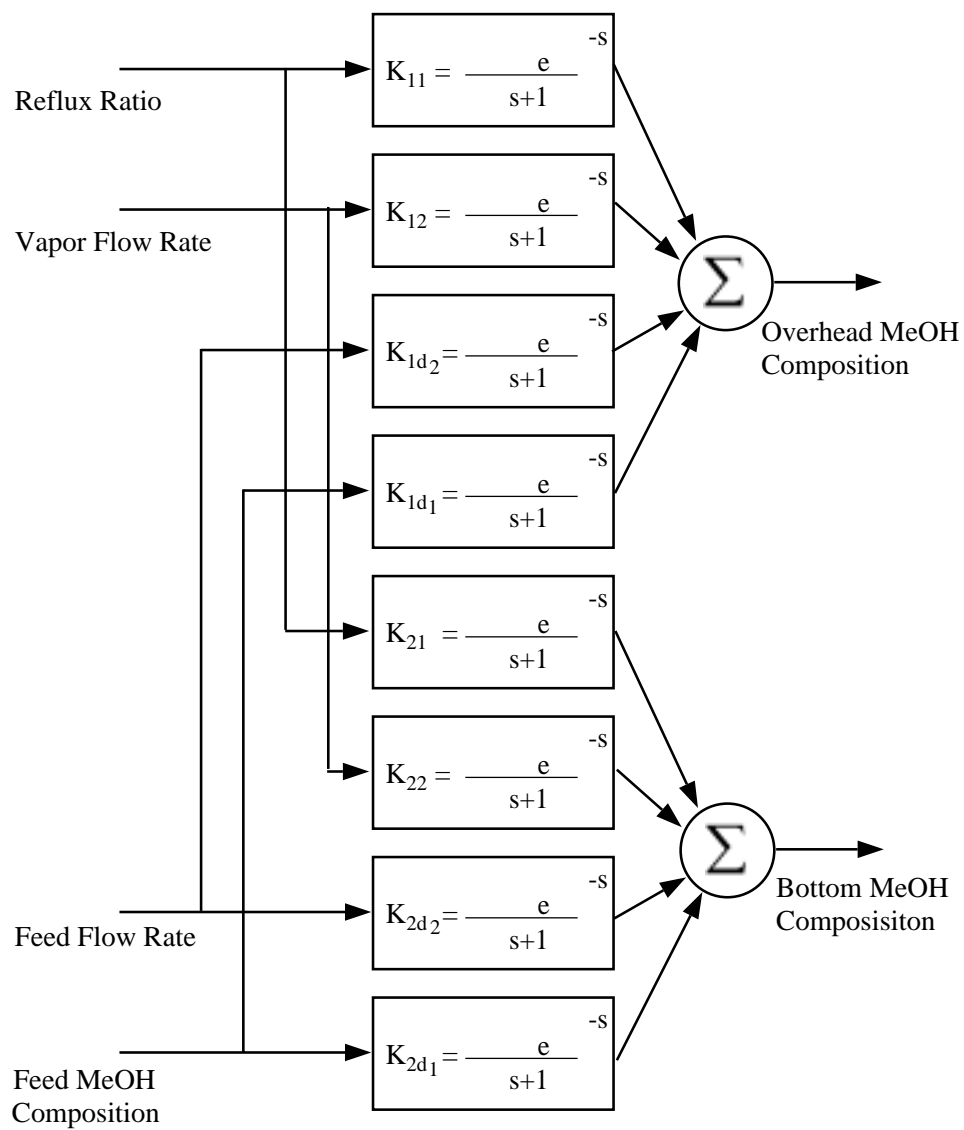


Figure 6.3. Block Diagram for Transfer Function Model of the Column

specially modified version of SIMULINK, the MATLAB-based object oriented programming software for dynamic simulations. SIMULINK can be started from the Graphical Analysis Menu by clicking on the *Simulink – Column* button. Clicking on this button creates two windows, a blank SIMULINK window with a save button (where the simulation is constructed) and the SIMULINK toolbox window (where programming elements are located). The results of the simulations will be saved into data files and loaded into *PCMplot* for comparison with the original plant data.

10. Construct a SIMULINK simulation to compare the modeled response of the *Overhead MeOH Composition* and the *Bottom MeOH Composition* for a 3% change in the *Vapor Flow Rate* (from 0.033 to $0.03399 \frac{\text{mol}}{\text{sec}}$), $G_{12}(s)$, and $G_{22}(s)$, with the actual plant data. Be aware that your transfer function model relates deviation variables while the plant data is not in deviation form.

Your SIMULINK simulation should consist of the following elements:

- (a) clock
- (b) step input
- (c) *Overhead MeOH Composition* (to workspace)
- (d) *Bottom MeOH Composition* (to workspace)
- (e) time (to workspace)
- (f) 2 Transfer Function Blocks and 2 Transport Delay Blocks or 2 Transfer Function Blocks (with dead time)
- (g) 2 sum blocks
- (h) 2 constant blocks

Run a simulation of your transfer function model and save the data into a data file. Do not repeat filenames from any previous runs.

Exercise 6.13 Filename: _____ (example: *vapor_val*).

Note: Make sure that you do not change the names of the variables in any of the *to workspace* blocks. A procedure has been written to interface the data collected from the validation experiments with *PCMplot* which is name specific.

11. Using the same SIMULINK block diagram from the previous exercise compare the modeled response of the *Overhead MeOH Composition* and the *Bottom MeOH Composition* to a 2% change in the *Reflux Ratio* (from 1.75 to 1.785); $G_{11}(s)$, and $G_{21}(s)$; with the actual plant data. Be aware that your transfer function model relates deviation variables while the plant data is not in deviation form.

Run a simulation of your transfer function model and save the data into a data file. Do not repeat filenames from any previous runs.

Exercise 6.14 Filename: _____ (example: *reflux_val*)

Note: Make sure that you do not change the names of the variables in any of the *to workspace* blocks. The system which interfaces the data collected from the validation experiments with *PCMplot* is name specific.

Save your SIMULINK simulation window for use later in this section.

12. To compare the data from the SIMULINK simulations with the plant data, *PCMplot* will be used in the validation mode. The validation mode loads the plant data and the data from the SIMULINK simulations sequentially and overlays the plots for comparison.
 - (a) After starting *PCMplot*, turn the validate mode on by:
 - i. pulling down the *Misc.* menu from the graphic display, see Figure 6.1,
 - ii. highlight the *validate* option with the mouse button still depressed and
 - iii. pull the mouse to the right of *validate* and highlight the *on* option.
 - (b) First, load the plant data file using the *Load* option and the prompts (remember to include the single quotes where specified).
 - (c) Using the *Plot* option, plot the *Overhead MeOH Composition (plot var 1)*.
 - (d) Using the *Plot* option, plot the *Bottom MeOH Composition (plot var 2)*.
 - (e) Use the *Load* option a second time to load the data from the SIMULINK simulation of the transfer function model.
 - (f) Using the *Plot* option, plot the *Overhead MeOH Composition (plot var 1)*. The plot of the data from the SIMULINK simulation of the transfer function will automatically overlay the plant data.
 - (g) Pull down the *View* menu and select *Original Axes*.
 - (h) It may be necessary to utilize the *Trim Axes* option to view the plots.
 - (i) Print the graph using the *Print* option.
 - (j) Using the *Plot* option, plot the *Bottom MeOH Composition (plot var 2)*.
 - (k) Select *Original Axes* under the *View* menu.
 - (l) It may be necessary to utilize the *Trim Axes* option to view the plots.
 - (m) **Exercise 6.15** Print the graph using the *Print* option.
13. How well do your models fit the dynamic plant data? Modify the parameters, if necessary, to obtain better agreement. Rerun the SIMULINK simulation and replot the data. Include these plots in your report.

14. Repeat the validation sequence for each transfer function (8 overall). Validated models will be needed for later modules.

Exercise 6.16 Create the seven remaining transfer function validation plots.

SUMMARY

Complex nonlinear systems can be approximated by a first-order-plus-time-delay model. This model is not globally accurate, although it is adequate for studying, designing, and tuning the controller to maintain the process at the operating point where the model was identified.

Module 7

STEADY STATE FEEDBACK CONTROL

OBJECTIVE

In this unit the characteristics of some of the various types of feedback control action and their influence on the performance of the Furnace/Column will be studied. Of particular interest is the impact of controller gain and reset time on the offset between the output and the setpoint at steady state.

Open-Ended Module Procedure

Design P and PI controllers for the furnace/column using the values suggested in the text. Examine the effect of changing the controller gain and reset time on the dynamic closed-loop response. In particular, characterize the impact of various controller tuning parameters on the offset observed at steady state.

THINGS TO THINK ABOUT—BOTH UNITS

Exercise 7-A Sketch the generalized block diagram of the furnace or column with a general controller and the following input-output pairing:

Manipulated Variable		Controlled Variable
Furnace: Fuel Gas Flow Rate	⇒	Hydrocarbon Outlet Temperature
Column: Reflux Ratio	⇒	Overhead MeOH Composition

Exercise 7-B What is the major advantage of proportional-integral (PI) control over proportional control?

Exercise 7-C The controlled variable is reduced from 100% to 90% due to a temporary bias in the sensor. The controlled variable is regulated with a proportional controller. Sketch the expected controlled response of the controlled variable to this disturbance.

Exercise 7-D The system undergoes the same disturbance as described in the preceding exercise except the controlled variable is now regulated with a PI controller. Sketch the expected controlled response of the controlled variable to this disturbance.

INTRODUCTION

The two types of controllers that will be studied in this section are proportional (P) and proportional-integral (PI) controllers. Proportional controllers consist of a constant gain which operates on the error signal generated by subtracting the current value of the controlled variable from the setpoint. The mathematical definition of proportional control is as follows:

$$u = K_c * e$$

where:

e	is $Y_{sp} - Y_{current}$;
Y_{sp}	is the setpoint of the controlled variable;
$Y_{current}$	is the current value of the controlled variable;
u	is the manipulated variable; and
K_c	is the proportional gain

Proportional-integral controllers have the same constant gain element as proportional controllers; however, PI controllers also have an integral element. Integral action forces the controller to continue changing as long as there is an error. The mathematical definition of a proportional-integral controller is as follows:

$$u = K_c * e + \frac{K_c}{\tau_I} * \int_0^t e(t') dt'$$

where:

τ_I	is the integral or reset time constant
----------	--

PROCEDURE—FURNACE

Proportional Control

1. To start the furnace with control, click once on the *Furnace with Control* button on the Furnace Menu (Figure 1.2). Double-click on the composition controller block and set the controller parameters to the following values (see Figure 7.1):

Gain of the controller (K_c)	=	$4.0 \frac{m^6}{mol \cdot min}$
Reset time or the integral time constant (τ_I)	=	1 min
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	0
Derivative Action On	=	0

Exercise 7.1 Enter the initial values of the process variables in Table 7.1.

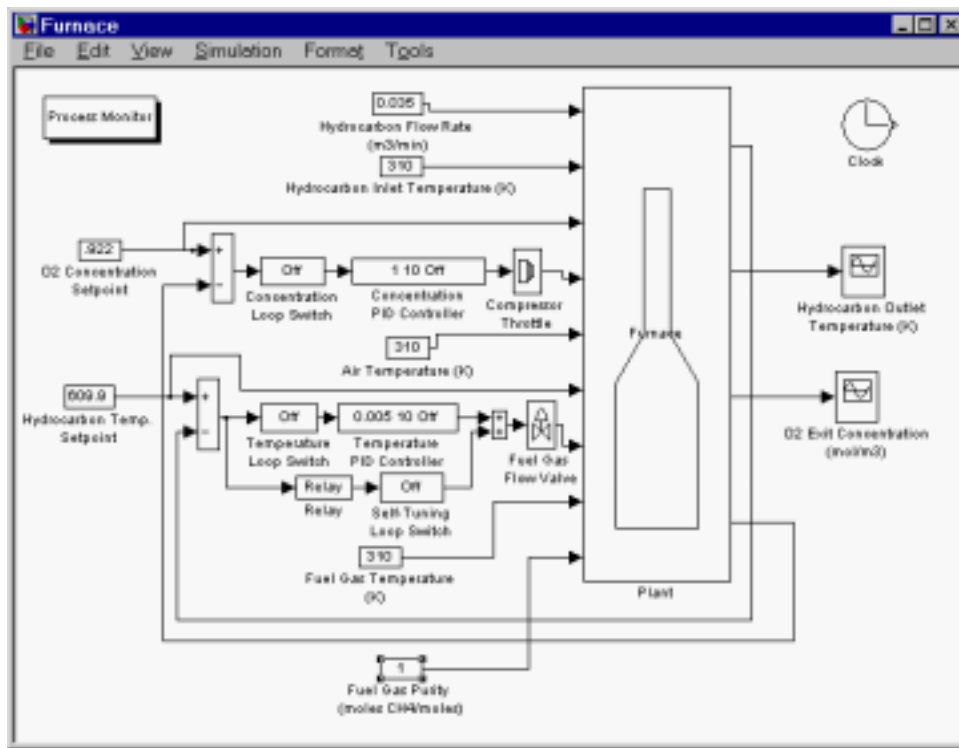


Figure 7.1. SIMULINK Flowsheet of the Furnace with Control

- Double-click on the composition loop switch and change its value to 1. Change the setpoint of *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$. When a steady state value is reached, record the final values in Table 7.1. Note the closed-loop behavior under proportional control.
- From the data recorded in the table, it can be seen that the final steady state value is not equal to the setpoint. There is a steady state error which is called an offset. This is the principle drawback of the proportional controller. From the plots on the Process Monitor, one can observe that the *Hydrocarbon Outlet Temperature* is also affected by the manipulation of the *Air Flow Rate*. This issue will be addressed later in this module with the use of PI control.

Table 7.1. Oxygen Exit Concentration—P Control Servo Response

$K_c \frac{\text{m}^6}{\text{mol min}}$	Steady State Value	Setpoint Value
0 (initial)		
4		
8.45		
13.5		

- Disconnect the controller by turning off the composition loop switch. Bring the setpoint back to $0.922 \frac{\text{mol}}{\text{m}^3}$ and the *Air Flow Rate* will return to $17.9 \frac{\text{m}^3}{\text{min}}$. Allow the system to return to its initial steady state.
- Change the controller settings to $K_c = 8.45$, turn on the composition loop, and repeat the exercise from Step 2 through Step 4. Then, try $K_c = 13.5$ and repeat again. Be sure to record the values in Table 7.1 after each run.

Exercise 7.2 Discuss the shortcomings of proportional control observed in this exercise. **Exercise 7.3** What effect did increasing the proportional gain have on the performance of the proportional controller (in terms of offset or oscillatory response)?

Proportional Control for a Disturbance

- Disconnect the controller by turning off the composition loop. Bring the furnace back to its initial steady state and change the *Oxygen Exit Concentration Setpoint* back to $0.922 \frac{\text{mol}}{\text{m}^3}$. Set the tuning parameters of the controller to the following values:

Gain of the controller (K_c)	=	$4.0 \frac{m^6}{mol \cdot min}$
Reset time or the integral time constant (τ_I)	=	1 min
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	0
Derivative Action On	=	0

7. Double-click on the composition loop switch and change its value to 1. Change the *Fuel Gas Purity* from $1.0 \frac{molCH_4}{mol}$ to $0.96 \frac{molCH_4}{mol}$.

Exercise 7.4 Record the final steady state values of *Oxygen Exit Concentration* in Table 7.2.

Table 7.2. Oxygen Exit Concentration—P Control Load Response

$K_c \frac{m^6}{mol \cdot min}$	Steady State Value	Setpoint Value
0 (Initial)		
4		
8.45		
13.5		

8. Again disconnect the controller by turning off the composition loop switch. Change the *Fuel Gas Purity* to $1.0 \frac{molCH_4}{mol}$.
9. Once the furnace has returned to steady state, set the gain $K_c = 8.45$ and turn on the composition loop switch. Repeat the exercise from Step 7. Change the gain to $K_c = 13.5$. Repeat Step 7 again.

Exercise 7.5 Be sure to enter all of the data in Table 7.2.

Proportional-Integral (PI) Control

10. Bring the system to its initial steady state condition.
11. Change the controller settings to the following values:

Gain of the controller (K_c)	=	$6.45 \frac{m^6}{mol \cdot min}$
Reset time or the integral time constant (τ_I)	=	2 min
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	1
Derivative Action On	=	0

12. Introduce a step change in the setpoint from $0.922 \frac{mol}{m^3}$ to $1.0143 \frac{mol}{m^3}$.

13. When steady state is reached note the final value of the *Oxygen Exit Concentration*. It should be equal to the setpoint indicating the absence of offset when integral action is present in the controller. Also note the overshoot (if any) and the settling time of the system.

Exercise 7.6 Record your data in Table 7.3.

Table 7.3. Oxygen Exit Concentration—PI Control Servo Response

$\tau_I(\text{min})$	Steady State Value	Setpoint Value
2 (initial)		
5		
10		
20		

14. Change the *Oxygen Exit Concentration Setpoint* back to $0.922 \frac{\text{mol}}{\text{m}^3}$ and let the system return to its initial steady state.
15. Repeat Steps 12, 13, and 14 for the following values of the integral reset time: $\tau_I = 5.0, 10.0, 20.0$ (min). Be sure to record your data in Table 7.3.

Exercise 7.7 Describe the difference between closed-loop responses obtained for the proportional (only) versus proportional-integral controllers.

Proportional-Integral (PI) Control for a Disturbance

16. Bring the system to its initial steady state.
17. Change the controller settings to the following values:

Gain of the controller (K_c)	=	$6.45 \frac{\text{m}^6}{\text{mol min}}$
Reset time or the integral time constant (τ_I)	=	2 min
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	1
Derivative Action On	=	0

18. Keep the *Oxygen Exit Concentration Setpoint* at $0.922 \frac{\text{mol}}{\text{m}^3}$. Introduce a step change in the *Fuel Gas Purity* from $1.0 \frac{\text{molCH}_4}{\text{mol}}$ to $0.96 \frac{\text{molCH}_4}{\text{mol}}$.
19. When steady state is reached note the final value of the *Oxygen Exit Concentration*. It should be equal to the setpoint indicating the absence of offset

when integral action is present in the controller. Also note the overshoot (if any) and the settling time of the system.

Exercise 7.8 Record your data in Table 7.4.

20. Change the *Fuel Gas Purity* back to $1.0 \frac{\text{mol CH}_4}{\text{mol}}$ and let the system return to its initial steady state.

Table 7.4. Oxygen Exit Concentration - PI Control Load Response

$\tau_I(\text{min})$	Steady State Value	Setpoint Value
2 (initial)		
5		
10		
20		

21. Repeat Steps 18, 19, and 20 for the following values of the integral reset time: $\tau_I = 5.0, 10.0, 20.0$ (min). Be sure to record your data in Table 7.4.

Exercise 7.9 Describe the difference between closed-loop responses obtained for the proportional (only) versus proportional-integral controllers.

PROCEDURE—COLUMN

Proportional Control

1. To start the column with control, click once on the *Column with Control* button on the Distillation Column Menu (Figure 1.3). Double-click on the *Overhead Composition Controller* block and set the controller parameters to the following values (see Figure 7.2):

Gain of the controller (K_c)	=	55 $\frac{\text{mol total}}{\text{mol MeOH}}$
Reset time or the integral time constant (τ_I)	=	300 sec
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	0
Derivative Action On	=	0

Exercise 7.1 Enter the initial values of the process variables in Table 7.5.

2. Double-click on the overhead loop to change it to the value *On* and switch the bottom loop switch to *Off*. Change the setpoint of the *Overhead MeOH Composition* to $0.83 \frac{\text{mol MeOH}}{\text{mol total}}$. When a steady state is reached, record the

final values in Table 7.5. Note the closed-loop behavior under proportional control.

Table 7.5. Overhead MeOH Composition - P Control Servo Response

$K_c \frac{\text{mol total}}{\text{mol MeOH}}$	Steady State Value	Setpoint Value
0 (Initial)		
55		
65		
75		

- From the data recorded in the table, it can be seen that the final steady state value is not equal to the setpoint. There is a steady state error called an offset. This is the principle drawback of the proportional controller. From the plots on the Process Monitor, one can observe that the *Bottom MeOH Composition* is also affected by the manipulation of the *Reflux Ratio*. The solution to this problem will be postponed until PI control is introduced.
- Disconnect the controller by turning off the *Composition Loop Switch*. Bring the setpoint back to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$ and the *Reflux Ratio* will return to its nominal steady state value. Allow the system to return to its initial steady state.
- Change the controller settings to $K_c = 65 \frac{\text{mol MeOH}}{\text{mol total}}$, turn on the overhead loop, and repeat the exercise from Step 2 through Step 4. Then, try $K_c = 75 \frac{\text{mol MeOH}}{\text{mol total}}$ and repeat again. Be sure to record the values in Table 7.5 after each run.

Exercise 7.2 Discuss the shortcomings of proportional control observed in this exercise. **Exercise 7.3** What effect did increasing the proportional gain have on the performance of the proportional controller (in terms of offset or oscillatory response)?

Proportional Control for a Disturbance

- Disconnect the controller by turning off the *Composition Loop Switch*. Bring the column back to its initial steady state and change the *Overhead MeOH Composition Setpoint* back to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Set the tuning parameters of the controller to

Gain of the controller (K_c)	=	55 $\frac{\text{mol total}}{\text{mol MeOH}}$
Reset time or the integral time constant (τ_I)	=	300 sec
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	0
Derivative Action On	=	0

7. Double-click on the *Overhead Loop Switch* and change its value to *On*. Change the *Feed MeOH Composition* from $0.5 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.4 \frac{\text{mol MeOH}}{\text{mol total}}$. Record the final steady state values of the *Overhead MeOH Composition* in Table 7.6.

Table 7.6. Overhead MeOH Composition - P Control Load Response

$K_c \frac{\text{mol total}}{\text{mol MeOH}}$	Steady State Value	Setpoint Value
0 (Initial)		
55		
65		
75		

8. Again disconnect the controller by turning off the overhead loop switch. Change the *Feed MeOH Composition* back to $0.5 \frac{\text{mol MeOH}}{\text{mol total}}$.
9. Once the column has returned to steady state, set the gain $K_c = 65 \frac{\text{mol MeOH}}{\text{mol total}}$ and turn on the *Composition Loop Switch*. Repeat the exercise from Step 2. Change the gain to $K_c = 75 \frac{\text{mol MeOH}}{\text{mol total}}$. Repeat again.

Exercise 7.4 Be sure to enter all of the data in Table 7.6.

Proportional-Integral (PI) Control

10. Bring the system to its initial steady state.
11. Change the controller settings to :

Gain of the controller (K_c)	=	55 $\frac{\text{mol total}}{\text{mol MeOH}}$
Reset time or the integral time constant (τ_I)	=	300 sec
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	1
Derivative Action On	=	0

12. Introduce a step change in the setpoint from $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.83 \frac{\text{mol MeOH}}{\text{mol total}}$.

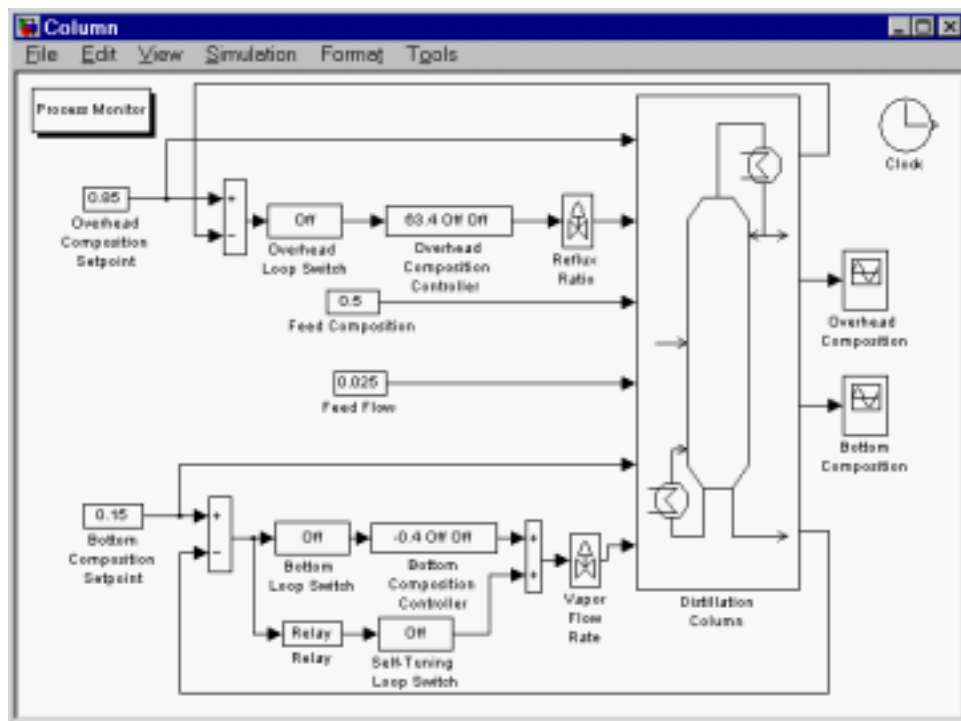


Figure 7.2. SIMULINK Flowsheet of the Column with Control

13. When steady state is reached note the final value of the *Overhead MeOH Composition*. It should be equal to the setpoint indicating the absence of offset when integral action is present in the controller. Also note the overshoot (if any) and the settling time of the system. Record your data in Table 7.7.

Table 7.7. Overhead MeOH Composition - PI Control Servo Response

$\tau_I(sec)$	Steady State Value	Setpoint Value
300 (Initial)		
350		
400		
450		

14. Change the setpoint back to $0.85 \frac{mol MeOH}{mol total}$ and let the system return to its initial steady state.
15. Repeat Steps 12, 13, and 14 for the following values of the integral reset time: $\tau_I = 350, 400, 450$ (sec). Be sure to record your data in Table 7.7.

Exercise 7.5 Describe the difference between closed-loop responses obtained for the proportional (only) versus proportional-integral controllers.

Proportional-Integral (PI) Control for a Disturbance

16. Bring the system to its initial steady state.
17. Change the controller settings to

Gain of the controller (K_c)	=	$55 \frac{mol total}{mol MeOH}$
Reset time or the integral time constant (τ_I)	=	$300 sec$
Rate time or the derivative time constant (τ_D)	=	0
Integral Action On	=	1
Derivative Action On	=	0

18. Keep the *Overhead MeOH Composition Setpoint* at $0.85 \frac{mol MeOH}{mol total}$. Introduce a step change in the *Feed MeOH Composition* from $0.50 \frac{mol MeOH}{mol total}$ to $0.40 \frac{mol MeOH}{mol total}$.
19. When steady state is reached note the final value of the *Overhead MeOH Composition*. It should be equal to the setpoint indicating the absence of offset when integral action is present in the controller. Also note the overshoot (if any) and the settling time of the system. Record your data in Table 7.8.

20. Change the *Feed MeOH Composition* back to $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$ and let system return to its initial steady state.

Table 7.8. Overhead MeOH Composition - PI Control Servo Response

$\tau_I(\text{sec})$	Steady State Value	Setpoint Value
300 (Initial)		
350		
400		
450		

21. Repeat Steps 18, 19, and 20 for the following values of the integral reset time: $\tau_I = 350, 400, 450$ (sec). Be sure to record your data in Table 7.8.

Exercise 7.6 Describe the difference between closed-loop responses obtained for the proportional (only) versus proportional-integral controllers.

SUMMARY

In this unit you have developed proportional and proportional-integral feedback controllers which automatically adjust the output variables by making changes in the manipulated variables. You have also seen that the feedback control system exhibits oscillatory behavior for certain control settings. As the controller gain is increased to eliminate steady state offset, such oscillatory behavior can become pronounced and for certain controller settings the closed-loop system can become unstable. In order to adjust or tune the controller settings, one must have information not only about the steady state behavior of the process, but also about its dynamic characteristics.

Module 8

CONTROLLER TUNING

OBJECTIVE

The performance of a feedback control system depends on the values of the controller tuning constants. The Proportional-Integral-Derivative (PID) controller will be used in this module for the Furnace/Column. A trial and error selection process for PID controller tuning constants requires a lengthy iterative procedure. In this module, you will be introduced to standard tuning algorithms which produce good initial estimates of K_c , τ_I , and τ_D .

Open-Ended Module Procedure

Design PID controllers for the *Fuel Gas Flow Rate—Hydrocarbon Outlet Temperature* (furnace) or *Vapor Flow Rate—Bottom MeOH Composition* (distillation column) loop using: (i) Ziegler Nichols, (ii) Cohen-Coon, and (iii) Relay Tuning rules. Test the performance of your controllers in closed-loop for both setpoint and disturbance changes.

THINGS TO THINK ABOUT

Exercise 8-A Calculate the controller parameters using the Cohen-Coon tuning method for the first-order-plus-time-delay transfer functions that you obtained in the Transient Response Module for the following variable pairings:

Furnace: *Fuel Gas Flow Rate* \Rightarrow *Hydrocarbon Outlet Temperature*
Column: *Reflux Ratio* \Rightarrow *Overhead MeOH Composition*

Proportional Control	$K_c =$		
Proportional-Integral Control	$K_c =$	$\tau_I =$	
Proportional-Integral-Derivative Control	$K_c =$	$\tau_I =$	$\tau_D =$

Furnace: *Air Flow Rate* \Rightarrow *Oxygen Exit Concentration*
 Column: *Vapor Flow Rate* \Rightarrow *Bottom MeOH Composition*

Proportional Control	$K_c =$		
Proportional-Integral Control	$K_c =$	$\tau_I =$	
Proportional-Integral-Derivative Control	$K_c =$	$\tau_I =$	$\tau_D =$

Exercise 8-B Given the following data, use the Ziegler-Nichols tuning rules to calculate the controller parameters for the control loop.

Furnace: *Fuel Gas Flow Rate* \Rightarrow *Hydrocarbon Outlet Temperature* loop:

$$P_u = 14.3(\text{min})$$

$$K_u = 0.0102$$

Column: *Vapor Flow Rate* \Rightarrow *Bottom MeOH Composition* loop:

$$P_u = 63.5(\text{min})$$

$$K_u = -4.6$$

Proportional Control	$K_c =$		
Proportional-Integral Control	$K_c =$	$\tau_I =$	
Proportional-Integral-Derivative Control	$K_c =$	$\tau_I =$	$\tau_D =$

Exercise 8-C Compare the controller parameters obtained with the Ziegler-Nichols tuning method with those obtained by the Cohen-Coon tuning rules. Comment on any significant differences.

INTRODUCTION

Two classical algorithms for PID controller tuning are the Cohen-Coon method and the Ziegler-Nichols method. These two methods yield controllers with at most a 0.25 value of the decay ratio (ratio of second peak to first peak in an oscillatory response). While this leads, in principle, to an attenuation of oscillations, these techniques typically lead to controllers with an aggressive response.

The Cohen-Coon tuning method requires an open-loop first-order-plus-time-delay transfer function model of the process. This can be obtained from a process reaction curve, generated from a step input. From the identified effective gain, time constant and dead time (K_p, τ_p, θ), one can tune the controller using the rules which are summarized below:

Controller Type	Controller Gain	Reset Time	Derivative Time Constant
P	$K_c = \left(\frac{1}{K_p}\right) \left(\frac{\tau_p}{\theta}\right) \left(1 + \frac{\theta}{3\tau_p}\right)$		
PI	$K_c = \left(\frac{1}{K_p}\right) \left(\frac{\tau_p}{\theta}\right) \left(\frac{9}{10} + \frac{\theta}{12\tau_p}\right)$	$\tau_I = \theta \left(\frac{30+3\theta/\tau_p}{9+20\theta/\tau_p}\right)$	
PID	$K_c = \left(\frac{1}{K_p}\right) \left(\frac{\tau_p}{\theta}\right) \left(\frac{4}{3} + \frac{\theta}{4\tau_p}\right)$	$\tau_I = \theta \left(\frac{32+6\theta/\tau_p}{13+8\theta/\tau_p}\right)$	$\tau_D = \theta \frac{4}{11+2\theta/\tau_p}$

The Ziegler-Nichols tuning method requires two key process parameters: the ultimate period (P_u), and the ultimate gain (K_u). These quantities can be obtained from a simple closed-loop test with a proportional controller. The proportional gain of the controller is increased until the closed-loop system exhibits sustained oscillations of constant amplitude. The period of these oscillations is defined as the ultimate period (P_u); the controller gain corresponding to this behavior is referred to as the ultimate gain (K_u). The Ziegler-Nichols method also seeks to achieve a quarter decay ratio performance objective and the rules for controller tuning are summarized below.

Controller Type	Controller Gain	Reset Time	Derivative Time Constant
P	$K_c = 0.5K_u$		
PI	$K_c = 0.45K_u$	$\tau_I = \frac{P_u}{1.2}$	
PID	$K_c = 0.6K_u$	$\tau_I = \frac{P_u}{2}$	$\tau_D = \frac{P_u}{8}$

Before you start the procedure, perform the following sequence of steps:

1. From the Furnace Menu click on the *Furnace with Control* or from the Distillation Column Menu click on the *Column with Control* button.
2. Make sure that both of the feedback loops are in manual by setting the loop switch blocks to the *off* position.
3. Start the simulation.
4. Make sure that the controller is properly tuned.
5. Allow the simulation to reach steady state before changing either of the feedback loops to automatic.

PROCEDURE—FURNACE

1. Tune the *Hydrocarbon Outlet Temperature* controller with the proportional-integral controller tuning constants that you calculated from the Cohen-Coon tuning rules. (Make sure that the integral action is *On* and the derivative action is *Off*.) Introduce a new setpoint on the *Hydrocarbon Outlet Temperature* of 616 K.

Exercise 8.1 Sketch the response.

Exercise 8.2 Calculate the decay ratio. Note the amount of time it takes for the *Hydrocarbon Outlet Temperature* to reach the new steady state. (**Hint:** it may prove useful to use the MATLAB *zoom* command to enlarge portions of the process monitor plots).

2. Reset the *Hydrocarbon Outlet Temperature Setpoint* to its initial value, 609.9 K. Once the system has returned to steady state, change the controller parameters to the PID settings obtained with the Cohen-Coon tuning rules for the *Hydrocarbon Outlet Temperature* loop. Make sure that integral action and derivative action are both *On*. Introduce a new setpoint on the *Hydrocarbon Outlet Temperature* of 616 K.

Exercise 8.3 Sketch the response.

Exercise 8.4 Calculate the decay ratio. Note the time it takes for the system to reach steady state. (**Hint: it may prove useful to use the MATLAB *zoom* command to enlarge portions of the process monitor plots**).

Exercise 8.5 How does the PID-controlled response differ from the PI-controlled response?

3. Return the system to steady state by resetting the *Hydrocarbon Outlet Temperature Setpoint* to 609.9 K. When the system has equilibrated, set the controller parameters using the proportional gain obtained from the Ziegler-Nichols tuning rules. Make sure that the integral action and derivative action are both *Off*. Disturb the system by changing the *Hydrocarbon Flow Rate* from $0.035 \frac{m^3}{min}$ to $0.037 \frac{m^3}{min}$.

Exercise 8.6 Sketch the system response.

Exercise 8.7 Calculate the maximum deviation from the setpoint. (**Hint: it may prove useful to use the MATLAB *zoom* command to enlarge portions of the process monitor plots**).

Exercise 8.8 Describe, qualitatively, the response under this controller.

4. Return the system to steady state by resetting the *Hydrocarbon Flow Rate* to $0.035 \frac{m^3}{min}$. Once the system has equilibrated, set the controller using the PID parameters obtained from the Ziegler-Nichols tuning rules. Make sure that integral action and derivative action are both *On*. Perturb the system by changing the *Hydrocarbon Flow Rate* from $0.035 \frac{m^3}{min}$ to $0.037 \frac{m^3}{min}$.

Exercise 8.9 Sketch the system response.

Exercise 8.10 Calculate the maximum deviation from the setpoint. (**Hint: it may prove useful to use the MATLAB *zoom* command to enlarge portions of the process monitor plots**).

Exercise 8.11 Compare the PID-controlled system response to the proportional-controlled system response in terms of the dynamic characteristics (in terms of speed, oscillations, *etc.*).

Closed-Loop Auto Relay Tuning Method

5. Tuning controllers with the closed-loop Ziegler-Nichols rules requires extensive trials to obtain the ultimate gain and ultimate period. A more efficient method to determine these quantities for controller tuning was developed by Åstrom and co-workers (Åstrom, K.J. and Hagglund, T., *Automatica*, **20**, pp. 645 (1984)). In order to use this method, the PID controller for a system is replaced by a relay element. In order to use the relay controller, turn off both of the feedback loops using the switch which follows the controller. Also, set the switch which follows the relay controller to the *On* position.

The relay controller forces the system to oscillate at a sustained amplitude without using trial and error as was the case for obtaining the Ziegler-Nichols rule-based controller parameters. The user must specify the amplitude of the input oscillation, d . From the observed magnitude (a) and period (P_u) of the oscillations in the output, one can determine the ultimate gain:

$$K_u = \frac{4 * d}{\pi * a}$$

where:

a = amplitude of output oscillations

d = amplitude of input oscillations

Given a specified phase margin (ϕ_m), a PID controller can be calculated as follows:

$$\begin{aligned} K_c &= K_u \cos(\phi_m) \\ \tau_I &= \frac{\alpha \tau_D}{\tan \phi_m \sqrt{\frac{4}{\alpha} + \tan \phi_m}} \\ \tau_D &= \frac{2 * \omega_c}{\tan \phi_m \sqrt{\frac{4}{\alpha} + \tan \phi_m}} \\ \omega_c &= \frac{2 * \pi}{P_u} \end{aligned}$$

where α is a design parameter.

These ideas are summarized in Figure 8.1.

Use the relay controller to obtain the ultimate gain. Assume that $\phi_m = 45$ degrees and $\alpha = 4$. The input oscillation amplitude for the furnace case study is $d = 0.5$.

Exercise 8.12 What is the ultimate gain?

Exercise 8.13 Calculate the PID controller parameters.

Exercise 8.14 How do the PID parameters compare with the Ziegler-Nichols tuning parameters obtained previously?

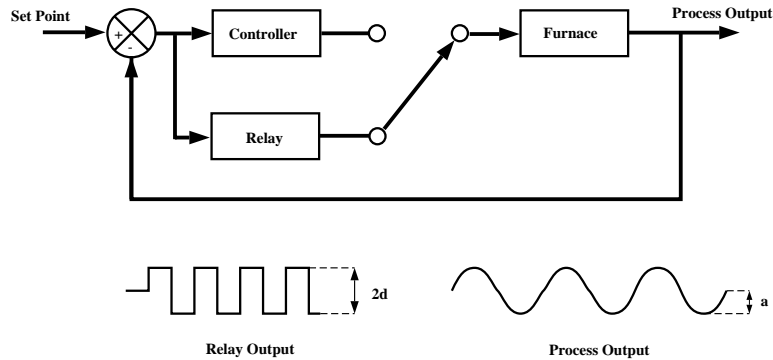


Figure 8.1. Inputs and Outputs for the Relay Tuning Method

6. Start with the system at steady state and set the controller by using the PID parameters obtained from the relay tuning rules in Procedure 5. Make sure that integral action and derivative action are both *On*. Introduce a new setpoint in the *Hydrocarbon Outlet Temperature* loop of 616 K.

Exercise 8.15 Observe the response and calculate the decay ratio.

Exercise 8.16 Compare the results with those obtained in part 2 for the Cohen-Coon tuning parameters.

PROCEDURE—COLUMN

1. Tune the *Bottom MeOH Composition* controller with the proportional-integral (PI) controller tuning constants that you calculated from the Cohen-Coon tuning rules. Make sure that the integral action is *On* and the derivative action is *Off*. Introduce a new setpoint on the *Bottom MeOH Composition* of $0.13 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 8.1 Sketch the response.

Exercise 8.2 Calculate the decay ratio (if any). Note the amount of time that it takes for the *Bottom MeOH Composition* to reach 95% the new steady state.

2. Reset the *Bottom MeOH Composition* to its initial value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$. Once the system has returned to steady state, change the controller parameters to the PID settings obtained with the Cohen-Coon tuning rules for the *Bottom MeOH Composition*. Make sure that integral action and derivative action are both *On*. Introduce a new setpoint on the *Bottom MeOH Composition* of $0.13 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 8.3 Sketch the response.

Exercise 8.4 Calculate the decay ratio (if any). Note the time that it takes for the system to reach 95% of the new steady state. **Exercise 8.5** How does the PID-controlled response differ from the PI-controlled response?

- Return the system to steady state by resetting the *Bottom MeOH Composition Setpoint* to $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$. When the system has equilibrated, set the controller parameters using the proportional gain obtained from the Ziegler-Nichols tuning rules. Make sure that the integral action and derivative action are both *Off*. Disturb the system by changing the *Feed MeOH Composition* from $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.40 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 8.6 Sketch the system response.

Exercise 8.7 Calculate the maximum deviation from the setpoint. **Exercise 8.8** Comment, quantitatively, on the controller performance.

- Return the system to steady state by resetting the *Feed MeOH Composition* to $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$. Once the system has equilibrated, set the controller using the PID parameters obtained from the Ziegler-Nichols tuning rules. (Make sure that integral action and derivative action are both *On*). Perturb the system by changing the feed concentration from $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.40 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 8.9 Sketch the system response.

Exercise 8.10 Calculate the maximum deviation from the setpoint. **Exercise 8.11** Compare the PID-controlled system response to the proportional-controlled system response.

Closed-Loop Auto Relay Tuning Method

- Refer to the discussion of the relay tuning method on Page 103. Use the relay controller to obtain the ultimate gain. Assume that $\phi_m = 45$ degrees and $\alpha = 4$. The input oscillation amplitude for the distillation column case study is $d = 0.005$.

Exercise 8.12 What is the ultimate gain? **Exercise 8.13** Calculate the PID controller parameters. **Exercise 8.14** How do they compare with the Ziegler-Nichols tuning parameters obtained previously?

- Start with the system at steady state and set the controller by using the PID parameters obtained from the relay tuning rules in Procedure 5. (Make sure that integral action and derivative action are both *On*.) Introduce a new setpoint in the bottom concentration $0.13 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 8.15 Observe the response and calculate the decay ratio. **Exercise 8.16** Compare the results with those obtained in Procedure 2 for the Cohen-Coon tuning parameters.

SUMMARY

Closed-loop performance of control systems is greatly affected by the selection of controller tuning parameters. Trial and error methods to obtain controller tuning constants require many iterations. A number of methods were presented in this unit which provide good initial values for controller tuning.

FEEDFORWARD CONTROL

OBJECTIVE

In the previous modules, process outputs were controlled using a feedback control strategy. The disadvantage of a feedback strategy is that control action is not initiated at the onset of the disturbance to a process, but rather action is only taken after the controlled variable starts deviating from its setpoint. This drawback becomes significant in cases where a process is dominated by slow dynamics and the disturbance occurs at a fast rate. A feedforward control strategy can be used to provide corrective action soon after the onset of a disturbance, thereby limiting deviation of the controlled variable from its setpoint. In this unit, you will implement a feedforward strategy on the Furnace/Column to reduce the effect of disturbances on process outputs.

Open-Ended Module Procedure

Derive feedforward controllers for the *Fuel Gas Purity* (furnace) or *Feed MeOH Composition* (column) disturbance based upon the FOTD models developed in Module 6. Implement the feedforward controller both alone, and in combination with a feedback controller. Consider both setpoint and disturbance changes.

THINGS TO THINK ABOUT

Exercise 9-A What is the fundamental difference between feedforward and feedback control?

Exercise 9-B Which is more detrimental to a feedforward controller, a slow (large time delay) plant or a slow disturbance?

Exercise 9-C Given the two cases in the previous question, how will the block diagram structure with feedforward control differ from that of feedback control?

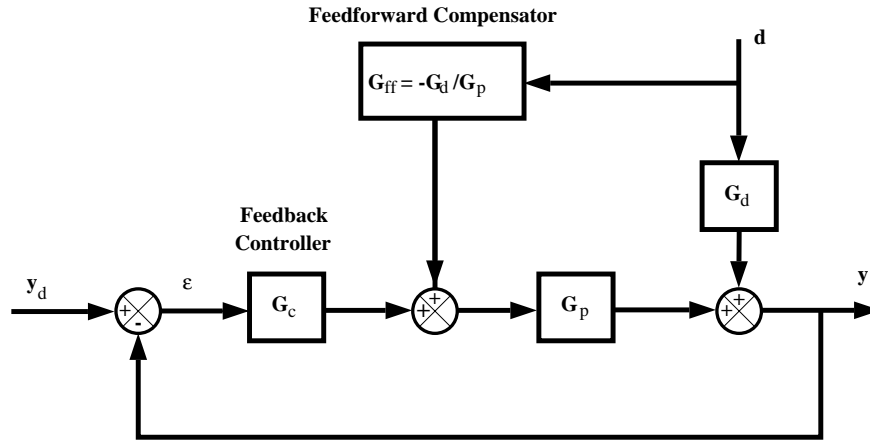


Figure 9.1. Feedforward Control Scheme

Exercise 9-D Why is a process model required for feedforward controller design?

Exercise 9-E Is feedforward control possible on an unmeasured disturbance?

INTRODUCTION

The control configuration that will be used in this unit is shown in Figure 9.1. It consists of feedforward compensation for both setpoint changes and load changes. As discussed in previous modules, the manipulated variables for the furnace are *Air Flow Rate* and *Fuel Gas Flow Rate* and the manipulated variables for the column are *Vapor Flow Rate* and *Reflux Ratio*. The load variables for the furnace are the *Fuel Gas Purity* and *Hydrocarbon Flow Rate* and the load variables for the column are the *Feed Flow Rate* and *Feed MeOH Composition*.

The feedforward controller receives information about the status of load variables. From a knowledge of the process and how disturbances affect the system, the feedforward controller calculates the manipulated variable change required and uses some form of dynamic compensation to account for the process dynamics.

The models can be obtained in different ways. Here, we are using the first-order-plus-time-delay system models developed in the Transient Response Analysis Module (Module 6). Given the following process model and disturbance model:

$$G_p(s) = \frac{K_p}{\tau_p s + 1} e^{-\theta_p s} \quad G_d(s) = \frac{K_d}{\tau_d s + 1} e^{-\theta_d s}$$

A perfect feedforward controller is given by:

$$G_f = -\frac{G_d}{G_p} = -\frac{K_d}{K_p} \times \frac{\tau_p s + 1}{\tau_d s + 1} e^{(\theta_p - \theta_d)s}$$

The controller is physically realizable only if $\theta_p < \theta_d$. If this condition is violated, one can use an approximate dynamic compensation by using a lead-lag element as shown below:

$$G_f = -\frac{K_d}{K_p} \times \frac{\tau_p s + 1}{\tau_d s + 1}$$

There are several strict requirements for feedforward control: the disturbance must be measurable, a very accurate model is needed of all the disturbances affecting the system, and an invertible model for the plant is needed. In real life all of these requirements may be difficult to obtain. Therefore, a feedforward controller can be used in conjunction with a feedback control loop for improved results.

PROCEDURE—FURNACE

1. Consider the loop between the *Oxygen Exit Concentration* and *Air Flow Rate* with the *Fuel Gas Purity* as the disturbance. Note: be sure to set the composition loop switch and temperature loop switch to *Off*. Use the models obtained in Module 6 to get the K_p and τ_p values for the disturbance and plant models. Insert the values into the controller and start the simulation. After 10 simulation minutes, change the *Fuel Gas Purity* to $0.96 \frac{\text{molCH}_4}{\text{mol}}$.

Exercise 9.1 Sketch the response, and qualitatively describe the behavior.

Exercise 9.2 How effective is the controller in maintaining the *Oxygen Exit Concentration* at its setpoint?

Exercise 9.3 What is the maximum deviation from setpoint?

Exercise 9.4 How much does the steady state *Hydrocarbon Outlet Temperature* change?

2. Change the *Fuel Gas Purity* back to $1.0 \frac{\text{molCH}_4}{\text{mol}}$. Let the system return to its initial steady state.
3. Now consider the loop between the *Hydrocarbon Outlet Temperature* and the *Fuel Gas Flow Rate* with the *Hydrocarbon Flow Rate* as the disturbance. Enable the appropriate feedforward controller (disable the one you used previously). Change the *Hydrocarbon Flow Rate* to $0.037 \frac{\text{m}^3}{\text{min}}$.

Exercise 9.5 How effective is the controller in rejecting the disturbance?

Exercise 9.6 Sketch the response, and qualitatively describe the behavior.

Exercise 9.7 What is the maximum deviation from setpoint for the *Hydrocarbon Outlet Temperature*?

Exercise 9.8 How much does the *Oxygen Exit Concentration* change (steady state offset)?

4. Change the setpoint of the *Hydrocarbon Flow Rate* back to $0.035 \frac{m^3}{min}$. Allow the system to return to its initial steady state.
5. Now change the *Hydrocarbon Outlet Temperature* setpoint from 609.9 K to 616 K.

Exercise 9.9 Sketch the response, and qualitatively describe the behavior.

Exercise 9.10 How much does *Hydrocarbon Outlet Temperature* deviate from setpoint?

Exercise 9.11 What is the maximum deviation from setpoint for the *Hydrocarbon Outlet Temperature*?

Exercise 9.12 How much does the *Oxygen Exit Concentration* change?

6. Return the *Hydrocarbon Outlet Temperature* to its setpoint of 609.9 K. Let the system return to its initial steady state.
7. Step the *Oxygen Exit Concentration* from $0.922 \frac{mol}{m^3}$ to $0.876 \frac{mol}{m^3}$.

Exercise 9.13 Sketch the response, and qualitatively describe the behavior.

Exercise 9.14 How well is the controller able to keep the *Oxygen Exit Concentration* at its setpoint? What is the maximum deviation from setpoint?

Exercise 9.15 Is there steady state offset? If so, how much?

Exercise 9.16 How much does the *Hydrocarbon Outlet Temperature* change at steady state?

8. Return the system to its initial steady state, *Oxygen Exit Concentration* at $0.922 \frac{mol}{m^3}$.
9. If a disturbance of large magnitude is introduced, then the feedforward controller may not perform very well for a complex (nonlinear) process. In this case, a feedback controller can be used in conjunction with the feedforward to get improved disturbance rejection. Enable the feedforward controller for *Fuel Gas Purity* (disable the one you used previously). Change the *Fuel Gas Purity* to $0.70 \frac{molCH_4}{mol}$.

Exercise 9.17 How effective is the controller for this disturbance?

Exercise 9.18 Sketch the response, and qualitatively describe the behavior.

Exercise 9.19 What is the maximum deviation from setpoint for the two controlled variables?

Exercise 9.20 Is there offset for the controlled variables? If so, how much?

10. Return *Fuel Gas Purity* to the normal steady state value, $1.0 \frac{\text{molCH}_4}{\text{mol}}$.

Exercise 9.21 What would you expect to see if the feedback controller was used in conjunction with the feedforward controller?

11. Implement the feedforward-feedback controller by entering the Cohen-Coon Controller tuning constants calculated in Module 8 for Proportional-Integral control of the *Oxygen Exit Concentration* loop. Make sure the switch value is set to *On*. Again, change the *Fuel Gas Purity* to $0.7 \frac{\text{molCH}_4}{\text{mol}}$.

Exercise 9.22 Sketch the output responses, and qualitatively describe the behavior.

Exercise 9.23 Describe the improvement (if any) in performance on each of the outputs.

Exercise 9.24 Describe, quantitatively and qualitatively (offsets, rise time, etc.), the improvements gained by the addition of feedback control in compensating for this disturbance.

12. Return *Fuel Gas Purity* to its steady state value of $1.0 \frac{\text{molCH}_4}{\text{mol}}$, and let the system return to steady state.

13. With the feedback controller still operating on the *Oxygen Exit Concentration* loop, enter the Cohen-Coon tuning constants calculated in Module 8 for PI control of the *Hydrocarbon Outlet Temperature* loop. Make sure the switch value is set to *On*. Again, change the *Fuel Gas Purity* to $0.7 \frac{\text{molCH}_4}{\text{mol}}$.

Exercise 9.25 Sketch the output responses for the controlled variables.

Exercise 9.26 Does the control of both outputs improve?

Exercise 9.27 Quantitatively and qualitatively describe the improvements gained by the addition of feedback control in compensating for this disturbance.

PROCEDURE—COLUMN

- Consider the loop between the *Overhead MeOH Composition* and *Reflux Ratio* with the *Feed MeOH Composition* as the disturbance. Note: be sure to set the *Overhead* and *Bottom Loop Switches* to *Off*. Use the models obtained in the Transient Response Analysis Module to get the K_p and τ_p values for the disturbances and the plant models. Insert the values into the controller and start the simulation. After 500 simulation seconds, change the feed concentration to $0.47 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.1 Sketch the response.

Exercise 9.2 How effective is the controller in maintaining the *Overhead MeOH Composition* at its setpoint?

Exercise 9.3 What is the maximum deviation from setpoint?

Exercise 9.4 How much does the *Bottom MeOH Composition* change?

2. Change the *Feed MeOH Composition* back to $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$. Let the system return to its initial steady state.
3. Now consider the loop between the *Bottom MeOH Composition* and the *Vapor Flow Rate* with the *Feed Flow Rate* as the disturbance. Enable the feedforward controller for *Feed Flow Rate* (disable the one you used previously). Change the *Feed Flow Rate* to $0.027 \frac{\text{m}^3}{\text{sec}}$.

Exercise 9.5 How well does the controller reject the disturbance?

Exercise 9.6 Sketch the response.

Exercise 9.7 What is the maximum deviation from setpoint for the *Bottom MeOH Composition*?

Exercise 9.8 How much does the *Overhead MeOH Composition* change (steady state offset)?

4. Change the value of the *Feed Flow Rate* back to $0.025 \frac{\text{m}^3}{\text{sec}}$. Allow the system to return to its initial steady state.
5. Now change the *Overhead MeOH Composition Setpoint* from $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.86 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.9 Sketch the response, and qualitatively describe the behavior.

Exercise 9.10 How much does *Overhead MeOH Composition* deviate from setpoint?

Exercise 9.11 What is the maximum deviation from setpoint for the *Overhead MeOH Composition*?

Exercise 9.12 How much does the *Bottom MeOH Composition* deviate from its setpoint?

6. Return the *Overhead MeOH Composition Setpoint* to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Let the system return to its initial steady state.
7. Step the *Bottom MeOH Composition Setpoint* from $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.14 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.13 Sketch the response and qualitatively describe the behavior.

Exercise 9.14 How well is the controller able to keep the *Bottom MeOH Composition* at its setpoint? What is the maximum deviation from setpoint?

Exercise 9.15 Is there an observable steady state offset? If so, how much?

Exercise 9.16 How much does the *Overhead MeOH Composition* change at steady state?

8. Return the system to its initial steady state setpoint, *Bottom MeOH Composition* of $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.
9. If a disturbance of large magnitude is introduced, then the feedforward controller does not perform very well. In this case, a feedback controller can

be used in conjunction with the feedforward to get improved disturbance rejection. Enable the feedforward controller for *Feed MeOH Composition* (disable the one you used previously). Change the *Feed MeOH Composition* to $0.35 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.17 How well does the controller perform for this disturbance?

Exercise 9.18 Sketch the response.

Exercise 9.19 What is the maximum deviation from setpoint for the two controlled variables?

Exercise 9.20 Is there offset for the controlled variables? If so, how much?

10. Return the *Feed MeOH Composition* to the normal steady state value.

Exercise 9.21 What would you expect to see if the feedback controller was used in conjunction with the feedforward controller in Procedure 9?

11. Implement the feedforward-feedback controller by entering the Cohen-Coon Controller tuning constants calculated in the previous module for Proportional-Integral control of the *Overhead MeOH Composition* loop. Make sure the switch value is set to *On*. Again, change the *Feed MeOH Composition* to $0.35 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.22 Sketch the output responses.

Exercise 9.23 Does control of both outputs improve? **Exercise 9.24** Describe, quantitatively and qualitatively (offsets, rise time, *etc.*), the improvements gained by the addition of feedback control in compensating for this disturbance.

12. Return *Feed MeOH Composition* to its steady state value of $0.50 \frac{\text{mol MeOH}}{\text{mol total}}$, and let the system return to steady state.
13. With the feedback controller still operating on the *Overhead MeOH Composition* loop, enter the Cohen-Coon tuning constants calculated in the previous unit for PI control of the *Bottom MeOH Composition* loop. Make sure the switch value is set to *On*. Again, change the *Feed MeOH Composition* to $0.35 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 9.25 Sketch the output responses for the controlled variables.

Exercise 9.26 Describe the improvement (if any) in performance for each of the outputs. **Exercise 9.27** Describe, quantitatively and qualitatively the improvements gained by the addition of feedback control in compensating for this disturbance.

SUMMARY

In this module, feedforward controller design has been introduced, based on first-order-plus-time-delay process models. One should have sufficient knowledge to design and implement a feedforward controller for disturbance rejection. For the operating conditions considered here, a first-order-plus-time-delay model approximates the chemical process reasonably well. For this reason one can achieve good performance with the feedforward controller.

Module 10

IMC AND MULTIVARIABLE CONTROL

OBJECTIVE

In Modules 7 and 8, classical controllers were applied to a process system. In this module we will address model-based control and multivariable control. The first-order-plus-time-delay models you validated in Module 6 will be used to design single-loop Internal Model Control (IMC) controllers as well as decouplers to reduce interaction between the two control loops.

Open-Ended Module Procedure

Using the RGA, determine the best pairing on inputs and outputs for SISO control design. Design an IMC controller for both loops, and evaluate its performance for individual as well as combined setpoint changes. Calculate a dynamic decoupler from the FOTD process models, and evaluate its performance.

THINGS TO THINK ABOUT

Exercise 10-A Calculate the Relative Gain Array (RGA) between the outputs and manipulated variables. For the furnace, the outputs are *Hydrocarbon Outlet Temperature* and *Oxygen Exit Concentration* and the manipulated inputs are *Air Flow Rate* and *Fuel Gas Flow Rate*. For the column, the outputs are *Overhead MeOH Composition* and *Bottom MeOH Composition* and the manipulated inputs are *Reflux Ratio* and *Vapor Flow Rate*.

Exercise 10-B Calculate the Single-Input-Single-Output (SISO) IMC controllers for the inputs and outputs in the previous question. Consider all permutations, which will yield 4 controllers in total.

Exercise 10-C How does changing λ , the filter time constant, affect the performance of the controller (or output response)?

Exercise 10-D Given a multivariable system, how do you determine stability of the transfer function matrix?

Exercise 10-E Is the following transfer function matrix stable? Does this plant have multivariable zeros?

$$G(s) = \begin{pmatrix} \frac{4}{s+1} & \frac{2}{s+3} \\ \frac{3s+1}{s+1} & \frac{5}{s+3} \end{pmatrix}$$

INTRODUCTION

Internal Model Control (IMC)

Designing an inverse-based controller from a validated process model is relatively straightforward. First, the transfer function relating the input to its corresponding output must be factored into an invertible portion, denoted $G_-(s)$, and a noninvertible portion, denoted $G_+(s)$, as seen below:

$$G(s) = G_-(s) \times G_+(s)$$

Time delays and Right-Half-Plane (RHP) zeros are included in $G_+(s)$. The balance of the model is factored into $G_-(s)$. The controller is then calculated as $[G_-(s)]^{-1}$. Typically, the power of s in the numerator is greater than that of the denominator. Consequently, the model inverse is improper (not realizable). This can be fixed by the addition of a simple filter, which also introduces an adjustable tuning parameter into the design procedure:

$$Q(s) = F(s) \times [G_-(s)]^{-1}$$

$$F(s) = \frac{1}{(\lambda s + 1)^n}$$

Here $F(s)$ is a filter applied to make the controller physically realizable. The filter time constant, λ , can be adjusted to change the aggressiveness of the controller. The variable n is chosen such that the highest power of s in the denominator of the controller $Q(s)$ is at least equal to that of the numerator of $Q(s)$. Once the controller has been designed, it can be applied to the system by running the plant model in parallel with the actual plant. The deviation in the output signals of these two components acts as the input to the inverse-based controller.

Decoupling Control

Controlling a multivariable process using multiple single-input, single-output controllers can improve overall performance, but it can also lead to controller interactions. In this case, the action implemented by one SISO controller can perturb the other output from its setpoint, and the SISO controllers interact with one another. To alleviate this problem, decouplers can be designed as depicted in Figure 10.1. In

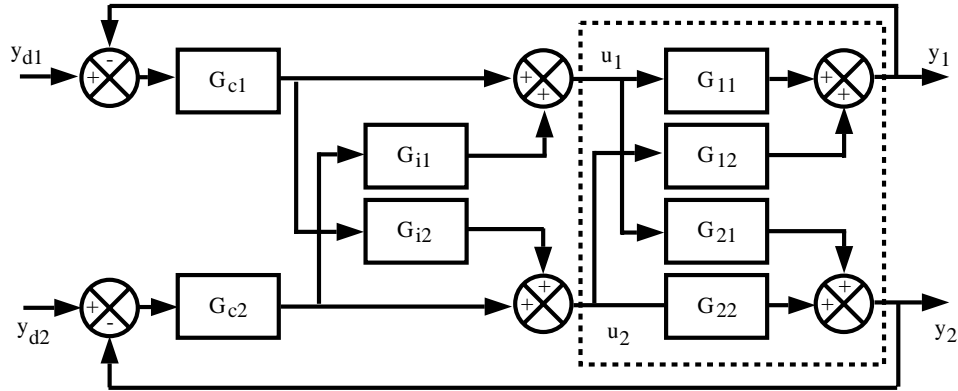


Figure 10.1. Decoupling Block Diagram

this module, G_{c1} and G_{c2} are the IMC controllers designed in the Things to Think About section. The decoupling blocks G_{i1} and G_{i2} vary depending on the type of decoupler desired. For steady state decouplers, they are given as:

$$G_{i1} = -\frac{K_{12}}{K_{11}}$$

$$G_{i2} = -\frac{K_{21}}{K_{22}}$$

If one has additional dynamic information about a process (*e.g.*, time constants and/or time delays), then it is possible to design more advanced decouplers by substituting the full transfer function for each of the gains above (*i.e.*, K_{12} becomes G_{12}).

PROCEDURE—FURNACE

1. From the Furnace Menu select the *Multivariable Control - Furnace*. Based on your RGA calculations (Things to Think About section), select an appropriate input-output pairing for SISO control. Use the switch blocks to turn on or off

specific models or controllers in the simulation. Change the switches for the models you will be using to *On*. Enter your model values into the *Hydrocarbon Outlet Temperature* model and *Oxygen Exit Concentration* model blocks.

Also from the Things to Think About section, insert the values that you have calculated into the IMC temperature controller and the IMC composition controller. If your controller requires a filter, set the filter time constant to 0.5 times the dominant model time constant.

2. Make sure the *Switch* for controlling the *Hydrocarbon Temperature* loop in the *IMC Controller* is set to *On* and the *Switch* for controlling the *Oxygen Composition* in the *IMC Controller* is set to *Off*. Start the simulation.

Exercise 10.1 What is your filter time constant?

Exercise 10.2 Does the *Hydrocarbon Outlet Temperature* track the 609.9 K setpoint?

3. Change the *Hydrocarbon Outlet Temperature Setpoint* to 622 K.

Exercise 10.3 What are the values of overshoot, rise time, and settling time for *Hydrocarbon Outlet Temperature* response in this setpoint tracking exercise?

Exercise 10.4 Comment critically on the performance of the controller regarding both of the outputs (offset, speed of response, *etc.*).

4. Return the system to its steady state (*Hydrocarbon Outlet Temperature Setpoint* 609.9 K). Multiply the filter time constant by 4, adjust the *IMC Controller* block transfer functions, and implement the same change as in the last exercise (*Hydrocarbon Outlet Temperature Setpoint* to 622 K.)

Exercise 10.5 What are the changes in the overshoot and settling times for the controlled output?

Exercise 10.6 Comment on the overall effect of this change.

5. Stop the simulation. Return the temperature loop filter time constant to its original value and reset the *Hydrocarbon Outlet Temperature Setpoint* to 609.9 K. Set the *Switch* for controlling the *Hydrocarbon Temperature* loop in the *IMC Controller* to *Off* and set the *Switch* for controlling the *Oxygen Composition* in the *IMC Controller* to *On*. Restart the Simulation.

Does the *Oxygen Exit Concentration* track the $0.922 \frac{\text{mol}}{\text{m}^3}$ setpoint?

6. Set the *Switch* for controlling the *Hydrocarbon Temperature* loop in the *IMC Controller* to *On* (both controllers on) and perform the setpoint change in Procedure 3.

Exercise 10.7 What are the values of overshoot, rise time, and settling time for the *Hydrocarbon Outlet Temperature* response in this setpoint tracking exercise?

Exercise 10.8 What are the values of overshoot, rise time, and settling time for the *Oxygen Exit Concentration* response in this setpoint tracking exercise?

Exercise 10.9 What trends do you notice?

Exercise 10.10 Why would increased oscillations result from having both controllers operational?

Decoupling

7. Stop the simulation. Go to the Main Menu and select *Clear Work Space*. Then click on *Decoupling Control* from the Furnace Menu. Double-click on *Controllers and Models* in the SIMULINK flowsheet window. Add the blocks necessary to design a PID-based decoupling control architecture. Implement your Cohen-Coon PI controller tuning constants from Module 8 for the two control loops. Then design steady state decouplers and incorporate them into your current flowsheet.

8. Start the simulation. Change the *Hydrocarbon Outlet Temperature* setpoint to 622 K.

Exercise 10.11 How does the response differ from that of Procedure 6?

Exercise 10.12 What are the values of overshoot, rise time, and settling time for the *Hydrocarbon Outlet Temperature* response in this setpoint tracking exercise?

Exercise 10.13 What are the values of overshoot, rise time, and settling time for the *Oxygen Exit Concentration* response in this setpoint tracking exercise?

9. Return the system to steady state. Replace the steady state decouplers with dynamic decouplers.

10. Once again, implement the setpoint change from Procedure 8. Change the *Hydrocarbon Outlet Temperature Setpoint* to 622 K.

Exercise 10.14 Compare the responses of the dynamic decouplers to that of the steady state decouplers. How do they differ?

Exercise 10.15 What are the values of overshoot, rise time, and settling time for the *Hydrocarbon Outlet Temperature* response in this setpoint tracking exercise?

Exercise 10.16 What are the values of overshoot, rise time, and settling time for the *Oxygen Exit Concentration* response in this setpoint tracking exercise?

- Examine your results for multivariable control of the furnace.

Exercise 10.17 Based on your results, choose a control method for the furnace *Hydrocarbon Outlet Temperature* and *Oxygen Exit Concentration*. Justify your answer.

PROCEDURE—COLUMN

- From the Distillation Column Menu select the *Multivariable Control - Column*. Based on your RGA calculations (Things to Think About section), determine the appropriate input-output pairing for SISO control. Use the switch blocks to turn on or off specific models or controllers. In the *Column Model* block, change the switches for the models you will be using to *On*. Enter your model values into the model blocks.

Also from the Things to Think About, insert the controllers that you have calculated into the *IMC Controller*. If your controller requires a filter, set the filter time constant to 0.5 times the dominant model time constant.

- Make sure the *Switch* for controlling the *Overhead MeOH Composition* loop in the *IMC Controller* is set to *On* and the *Switch* for controlling the *Bottom MeOH Composition* in the *IMC Controller* is set to *Off*. Start the simulation.

Exercise 10.1 What is your filter time constant?

Exercise 10.2 Does the *Overhead MeOH Composition* loop track the $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$ setpoint?

- Change the *Overhead MeOH Composition Setpoint* to $0.87 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 10.3 What are the values of overshoot, rise time, and settling time for the *Overhead MeOH Composition* response in this setpoint tracking exercise? **Exercise 10.4** Describe, quantitatively, the performance of the controller with respect to the outputs (offset, speed of response, etc.).

- Return the system to its steady state (*Overhead MeOH Composition Setpoint* $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$). Multiply the filter time constant by 4, implement the new controller in the *IMC Controller* block, and implement the same change as in the last exercise (*Overhead MeOH Composition Setpoint* to $0.87 \frac{\text{mol MeOH}}{\text{mol total}}$).

Exercise 10.5 What are the changes in the overshoot and settling times for the controlled output? **Exercise 10.6** What effect does this change have on control?

- Stop the simulation. Return the *Overhead MeOH Composition* loop filter time constant to its original value and reset the *Overhead MeOH Composition* setpoint to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Set the *Switch* for controlling the *Overhead*

MeOH Composition loop in the *IMC Controller* to *Off* and set the *Switch* for controlling the *Bottom MeOH Composition* in the *IMC Controller* to *On*. Restart the Simulation.

Does the *Bottom MeOH Composition* track the $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$ setpoint?

- Set the *Switch* for controlling the *Overhead MeOH Composition* in the *IMC Controller* to *On* (both controllers on) and perform the setpoint change in Procedure 3.

Exercise 10.7 What are the values of overshoot, rise time, and settling time for the *Overhead MeOH Composition* response in this setpoint tracking exercise?

Exercise 10.8 What are the values of overshoot, rise time, and settling time for the *Bottom MeOH Composition* response in this setpoint tracking exercise?

Exercise 10.9 What trends do you notice? **Exercise 10.10** Why could more oscillation result from having both controllers operational?

Decoupling

- Stop the simulation. From the Distillation Column Menu select *Clear Work Space*. Then click on *Decoupling Control*. Double-click on *Controllers and Models* in the SIMULINK flowsheet window. Add the blocks necessary to design a PID-based decoupling control architecture. Implement your Cohen-Coon PI controller tuning constants from the controller tuning unit for the two control loops. Design steady state decouplers and incorporate them into your current flowsheet.

- Start the simulation. Change the *Overhead MeOH Composition* setpoint to $0.87 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 10.11 How does the response differ from that of Procedure 6?

Exercise 10.12 What are the values of overshoot, rise time, and settling time for the *Overhead MeOH Composition* response in this setpoint tracking exercise?

Exercise 10.13 What are the values of overshoot, rise time, and settling time for the *Bottom MeOH Composition* response in this setpoint tracking exercise?

- Return the system to steady state. Replace the steady state decouplers with Dynamic Decouplers.

- Implement the setpoint change in Procedure 8.

Exercise 10.14 Compare the responses of the Dynamic Decouplers to that of the steady state decouplers. How do they differ? **Exercise 10.15** What are the values of overshoot, rise time, and settling time for the *Overhead MeOH*

Composition response in this setpoint tracking exercise? **Exercise 10.16**
What are the values of overshoot, rise time, and settling time for the *Bottom MeOH Composition* response in this setpoint tracking exercise?

11. Examine your results for multivariable control of the column.

Exercise 10.17 Based on your results, choose a control method for the column *Overhead* and *Bottom MeOH Composition*. Justify your answer.

SUMMARY

In this unit, you have designed model-based controllers, given an accurate process model. In addition, you have explored controller interaction in multivariable systems, and the impact on closed-loop performance. Given an accurate model of a system, you should be able to design a dynamic decoupler to negate, or at least reduce, controller interaction.

DISCRETE TIME SYSTEM MODELING

OBJECTIVE

This module is designed to introduce the issues involved in modeling dynamic systems in the discrete-time domain. Relationships between continuous-time and discrete-time systems are also described in the context of low-order process models.

Open-Ended Module Procedure

Using SIMULINK, construct a simple example to demonstrate the aliasing effect of digital sampling. Identify the process system contained in the *Discrete Model ID* menu item. Perform model validation of the resultant model.

THINGS TO THINK ABOUT

Exercise 11-A What are the physical and/or theoretical limiting factors which place upper or lower bounds on the sampling time for a discrete-time system?

Exercise 11-B Calculate the Z -transform of the continuous time system:

$$\frac{y(s)}{u(s)} = \frac{5e^{-4s}}{6s + 1}$$

for a sampling time (Δt_s) of 1 sec. Recalculate the discrete transfer function for the case of $\Delta t_s = 1.5$ sec.

Exercise 11-C Describe one method by which a discrete-time signal can be converted to a continuous-time signal.

Exercise 11-D What assumption(s) are made in reconstructing a continuous signal using the above method?

INTRODUCTION

Most chemical engineering processes are continuous systems, and can be mathematically described by a set of continuous differential equations. Recall that a first-order system can be represented as:

$$\tau_p \frac{dy(t)}{dt} + y(t) = K_p u(t)$$

where:

- $y(t)$ is the system output;
- $u(t)$ is the input (forcing function);
- K_p is the system gain; and
- τ_p is the system time constant.

To represent this model in discrete-time, we need to define the *sampling time* for a discrete process. In terms of measurements, this can be interpreted of as the sampling rate of a measurement. To approximate the continuous derivative in discrete time, the *forward difference* notation is used, which yields:

$$\frac{dy(t)}{dt} \cong \frac{y(k) - y(k-1)}{\Delta t_s}$$

where:

- Δt_s is the sampling time; and
- $y(k)$ is the value of $y(t)$ at $t = k\Delta t_s$

The equation for a continuous system can now be recast in the following discrete-time form:

$$y(k) = \left(1 - \frac{\Delta t_s}{\tau_p}\right) y(k-1) + \left(\frac{K_p \Delta t_s}{\tau_p}\right) u(k-1)$$

Consequently, the output of a discrete-time process can be viewed as a weighted sum of past outputs, $y(k-1)$, and past inputs, $u(k-1)$. By introducing the shift operator, z^{-1} , we can rewrite the previous equation as:

$$y(k) = \left(1 - \frac{\Delta t_s}{\tau_p}\right) z^{-1} y(k) + \frac{K_p \Delta t_s}{\tau_p} z^{-1} u(k)$$

Solving for the output, $y(k)$, one obtains the discrete-time (pulse) transfer function relationship:

$$\frac{y(z)}{u(z)} = \frac{\frac{K_p \Delta t_s}{\tau_p} z^{-1}}{1 - \left(1 - \frac{\Delta t_s}{\tau_p}\right) z^{-1}}$$

Model discretization can also be performed directly on the Laplace transform. By introducing the variable transformation $z = e^{s\Delta t_s}$, the Z -transform of a process is defined by:

$$F(z) = \sum_{n=0}^{\infty} f(n\Delta t_s) z^{-n}$$

One important consideration in using the Z-transform is the fact that the controller output signal (digital) must be converted to a continuous signal for ultimate implementation with an actuator (*e.g.*, a process pump). Typically, a so-called hold function is employed, the most elementary of which is the Zero-Order-Hold (ZOH). The ZOH transfer function (in continuous time) is:

$$\mathcal{H}(s) = \frac{(1 - e^{-s\Delta t_s})}{s}$$

In other words, the discrete output is held constant between samples. The convolution (product) of the simple first-order system with the zero-order hold can then be transformed to discrete-time by use of the Z-transform, resulting in the model:

$$\frac{y(z)}{u(z)} = \frac{K_p \left(1 - e^{-\frac{\Delta t_s}{\tau_p}}\right) z^{-1}}{1 - e^{-\frac{\Delta t_s}{\tau_p}} z^{-1}}$$

Note that unlike the backward difference approximation above, the Z-transform is an *exact* discretization. A time delay ($e^{-\theta s}$), if present in the s-domain system, manifests itself in the discrete domain as z^{-n} , where n is given by $\frac{\theta}{\Delta t_s}$. The characteristics of discrete-time systems and utility of the preceding discussion will be examined in the balance of this module.

PROCEDURE

1. To start the discrete-time systems modeling unit, click once on the Discrete Menu button from the Main Menu (Figure 1.1) and select *Aliasing*. This will bring up the flowsheet shown in Figure 11.1.

One concern in discrete-time modeling is a problem known as aliasing. A poorly chosen sample time will cause the digital reconstruction of a signal to display dynamics different than that of the real system.

2. The effects of sample time selection on process discretization can be summarized by filling out the following table. The Frequency should be entered into the *Sine Wave* block under frequency (remember that π in MATLAB can be entered using the two-letter symbol *pi* rather than 3.14159....). The sample time is placed in the *Sampling* block. The output will draw straight lines between each collected data point.

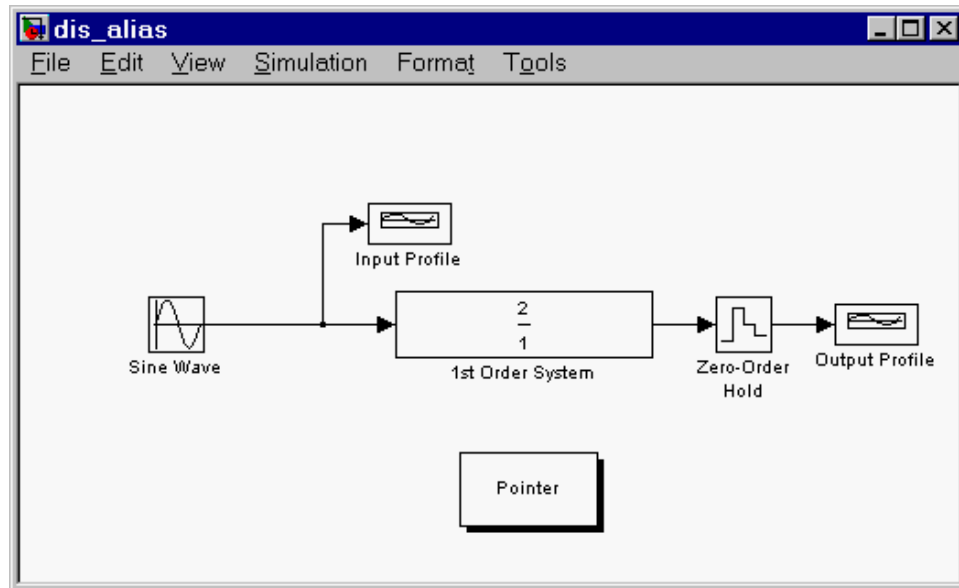


Figure 11.1. Flowsheet for Discrete-Time Aliasing Simulations

Exercise 11.1 Fill in the table below:

Input Frequency (rad/sec)	Sample Time (sec)	Output Amplitude (-)	Output Frequency (rad/sec)
$\pi/5$	5		
$\pi/5$	2.5		
$\pi/5$	0.2		
$\pi/10$	5		
$\pi/10$	2.5		
$\pi/10$	0.2		

Exercise 11.2 Although it provides the most accurate representation of the process data, what are the possible drawbacks of the sampling time of 0.2 sec?

- Clearly the selection of sample times is an important issue in discrete-time process modeling. Although many heuristics exist, a good rule of thumb is to select the sample time as approximately 0.2τ , where τ is the dominant time constant for the process.

From the Discrete Menu, now select *Discrete Model ID*. The flowsheet is shown in Figure 11.2. One has an unknown plant, and the objective is to develop a discrete-time dynamic model. By utilizing a step input, the process gain

and time constant can be calculated using the process reaction curve method (identical to that used in the Dynamic Model Identification modules). To get a step input make sure the *Step Switch* is set to *On*, and the *Sine Switch* is set to *Off*. Also make sure the *CT Switch* and *DT Switch* are set to *Off*.

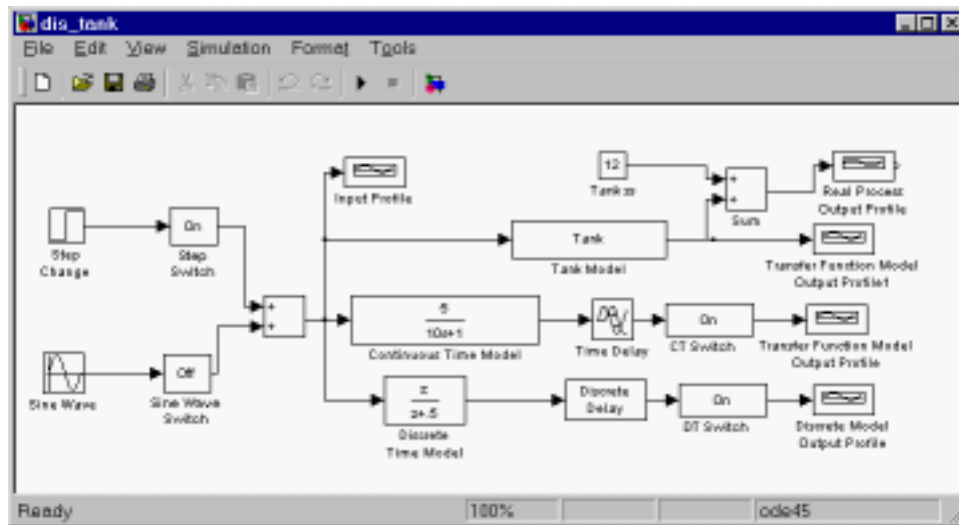


Figure 11.2. Flowsheet for Discrete Model Identification

Exercise 11.3 Introduce a step input with magnitude of +1 to the unknown plant, and from the process reaction curve identify a first-order-plus-time-delay transfer function model of the plant. **Exercise 11.4** What sampling time do you recommend for this system?

Exercise 11.5 If it were necessary to have the time delay be an integer multiple of sample times, while still keeping $\Delta t_s \approx 0.2 \tau$, what sample time would you choose? (Remembering that a time delay of θ in the Laplace domain is represented in the discrete domain by z^{-n} , with n given by $\frac{\theta}{\Delta t_s}$). What would happen if the time delay was not an integer multiple of the sampling time? **Exercise 11.6** Having developed a continuous-time model of the system, use the sample time from the previous exercise and the material from the introduction to calculate the discrete-time dynamic model of this system (in the z -domain).

4. An important aspect of developing models for systems is the validation of those models. In the *Continuous Time Model* block, enter your continuous-time transfer function model (generated from the process reaction curve), placing the time delay in the appropriate block. Likewise, for the discrete-time model, enter it in the *Discrete Time Model* block (remembering that the delay will be given by z^{-n}). Do not forget to set the sample time in all

discrete blocks. Both switches must be set to *On* in order to compare all three outputs.

Exercise 11.7 Using the step input value of $+1$, simulate all three systems and comment on differences between the real process (blue), the continuous model (red), and the discrete model (green). How accurate are your continuous and discrete models with respect to gain, time constant, *etc.*?

Exercise 11.8 Change the step input to -1 and run the simulation. Are your models still accurate? If not, describe the differences between your models and the process. How do you account for the discrepancies?

SUMMARY

The concept of sampling time, in the context of a discrete-time model structure, was introduced in this unit. Aliasing is seen to be an important factor in discrete-time modeling, as the sampling rate affects the signal properties. Selection of time constants and the handling of process time delay were also addressed.

Module 12

DISCRETE CONTROL

OBJECTIVE

This module addresses the synthesis of discrete-time feedback controllers for the Furnace/Column. Discrete PID controllers will be developed, and performance will be analyzed through tuning constant selection. The effects of choosing the proper sample time will also be explored.

Open-Ended Module Procedure

Design discrete PID controllers for the Furnace/Column to control hydrocarbon temperature or overhead composition using Ziegler-Nichols and Cohen-Coon tuning rules. In addition, derive and implement a discrete-time IMC controller. Examine the setpoint tracking response of each controller.

THINGS TO THINK ABOUT

Exercise 12-A How does the zero-order hold reconstruct a continuous-time signal from a discrete-time signal?

Exercise 12-B In addition to the PID controller coefficients, what other user-selected (or measurement device determined) parameters will affect closed-loop controller performance?

Exercise 12-C Determine the Z -transform of the following first-order filter:

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

Exercise 12-D Given that the previous transfer function is the filter used in Internal Model Control (IMC), design an IMC controller for your system using your first-order-plus-time-delay models from the Transient Response Analysis Module. For the furnace, design the controller for the *Fuel Gas Flow Rate* to

Hydrocarbon Outlet Temperature loop. For the column, design the controller for the *Reflux Ratio to Overhead MeOH Composition* loop. (**Hint: recall that continuous time IMC design is $F(s) * [G(s)]^{-1}$ and that one must incorporate the hold operator (zero-order hold, $H(s)$) in continuous time and then perform the Z-transform of $H(s) * F(s) * [G(s)]^{-1} = HFG^{-1}(z)$ which is not equivalent to $H(z) * F(z) * [G(z)]^{-1}$.**)

Exercise 12-E Will a system (with time constant τ) under closed-loop control subjected to a step change in the setpoint settle faster with a sample time of 0.2τ or 2τ ? Why?

INTRODUCTION

Analog controllers are becoming less common in industry due to the implementation of distributed control systems for process regulation. Using these computer systems, control loops are implemented in a discrete-time framework. This motivates engineers to work with discrete-time controllers and controller tuning.

Recall Module 8, where Ziegler-Nichols and Cohen-Coon rules were used to select controller tuning parameters. This unit will start by analyzing controller performance under these tuning rules converted to discrete-time PID coefficients.

The continuous PID controller obeys the following equation in the time domain (in deviation form):

$$u(t) = K_c \left[\epsilon(t) + \frac{1}{\tau_I} \int_0^t \epsilon(t') dt' + \tau_D \frac{d\epsilon}{dt} \right]$$

If one uses the rectangular rule for numerical approximation of the integral term,

$$\int_0^t \epsilon(t') dt' \approx \sum_{i=1}^k \epsilon(i) \Delta t$$

and the backward difference to approximate the derivative term,

$$\frac{d\epsilon}{dt} \approx \frac{\epsilon(k) - \epsilon(k-1)}{\Delta t}$$

then the discrete PID controller equation can be written as:

$$u(k) = K_c \left[\epsilon(k) + \frac{\Delta t}{\tau_I} \sum_{i=1}^k \epsilon(i) + \frac{\tau_D}{\Delta t} (\epsilon(k) - \epsilon(k-1)) \right] \quad (12.1)$$

This is known as the *position* form. Implementation of discrete-time controllers is usually in *velocity* form. We can get to this by shifting the indices of Equation (12.1) by one, and subtracting the result from (12.1). This yields:

$$\Delta u(z) = K_c \left[\left(1 + \frac{\Delta t}{\tau_I} + \frac{\tau_D}{\Delta t} \right) - \left(\frac{2\tau_D}{\Delta t} + 1 \right) z^{-1} + \left(\frac{\tau_D}{\Delta t} \right) z^{-2} \right] \epsilon(z) \quad (12.2)$$

In this way the parameters developed in the controller tuning unit can be used in the discrete-time domain.

Later in this module, the emphasis will be on the design of the Internal Model Controller (IMC) in discrete time. Recall from the Multivariable Control Module, that an IMC controller is designed by first partitioning the process model into an invertible ($G_-(s)$) and noninvertible (G_+s) portion. Then G_- is inverted to yield the IMC controller. If this inversion is not physically realizable, then a filter must be applied prior to controller implementation. In the Z -domain, a first-order filter is given by:

$$F(z) = \frac{1 - \lambda}{z - \lambda}$$

where the filter constant λ is $0 < \lambda < 1$. Here, the filter time constant λ can be used to tune the aggressiveness of the controller. Smaller values (closer to zero) yield faster response. Note that in the limit as $t \rightarrow \infty$, the filter gain approaches unity (recall: $t \rightarrow \infty \Leftrightarrow z \rightarrow 1$).

PROCEDURE—FURNACE

Before you start the exercises, Click on the *Discrete* button on the Main Menu, then select *PID Control - Furnace* from the Discrete Menu. This will display Figure 12.1.

1. A good initial estimate of the sample time for discrete control is $\frac{1}{5}$ of the dominant open-loop time constant, rounded such that any delay is an integer number of sample times.

Exercise 12.1 What is your first-order-plus-time-delay model for the *Fuel Gas Flow Rate* to *Hydrocarbon Outlet Temperature* loop?

Exercise 12.2 What sample time do you recommend using to control this loop?

Exercise 12.3 Using Equation 12.2 above, and your K_c , τ_I , and τ_D coefficients from Module 8, fill in the following table with the discrete-time tuning constants using Cohen-Coon rules for the *Fuel Gas Flow Rate* to *Hydrocarbon Outlet Temperature* loop.

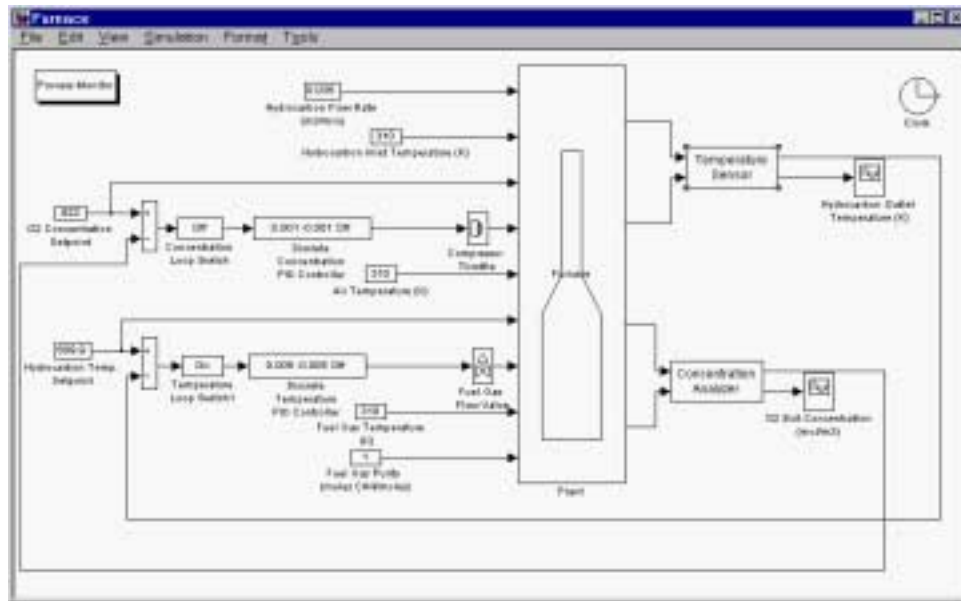


Figure 12.1. Flowsheet for the Furnace using Discrete PID Control

Controller Type	$\epsilon(k)$ Coefficient	$\epsilon(k-1)$ Coefficient	$\epsilon(k-2)$ Coefficient
P			
PI			
PID			

2. **Exercise 12.4** Use **Ziegler-Nichols** rules for the *Fuel Gas Flow Rate* to *Hydrocarbon Outlet Temperature* loop. Fill in the following table based on your Module 8 continuous-time tuning constants:

Controller Type	$\epsilon(k)$ Coefficient	$\epsilon(k-1)$ Coefficient	$\epsilon(k-2)$ Coefficient
P			
PI			
PID			

3. From the Discrete Menu, select *PID Control*. Tune the *Discrete Temperature PID Controller* using the Cohen-Coon discrete tuning constants for PI control. Make sure the integral action is *On* and the derivative action is *Off*. Remember to include the sample time in the controller block and the *Temperature Sensor*. Introduce a new *Hydrocarbon Outlet Temperature Setpoint* of 620 K.

Exercise 12.5 Sketch the response.

Exercise 12.6 Note the settling time (time required for the *Hydrocarbon Outlet Temperature* to remain within 5% of the new steady state value).

- Return the system to steady state by setting the *Hydrocarbon Outlet Temperature Setpoint* to 609.9 K. Enter the derivative time constant in the *Discrete Temperature PID Controller*, and set the derivative action to *On*. Make the same setpoint change as above.

Exercise 12.7 Sketch the response.

Exercise 12.8 How does the use of derivative action affect the settling time?

- Return the setpoint to its initial value, 609.9 K. Once the system has reached steady state, double the sample time, and make the necessary changes in the *Discrete Temperature PID Controller* and *Temperature Sensor*. Also update the controller parameters (**REMEMBER: PID tuning coefficients are sample time dependent**). Again set the *Hydrocarbon Outlet Temperature Setpoint* to 620 K.

Exercise 12.9 How does the settling time compare to that achieved before?

- Again return the system to steady state by setting the *Hydrocarbon Outlet Temperature Setpoint* to 609.9 K. Return the *Temperature Sensor* and *Discrete Temperature PID Controller* sample times to their initial values. Replace the Cohen-Coon parameters with those calculated using the Ziegler-Nichols rules. Make the setpoint change described above.

Exercise 12.10 Compare the Ziegler-Nichols PID results to the PID results generated under Cohen-Coon tuning. Which is more aggressive?

Exercise 12.11 What is the settling time for this controller, and how does that compare with those found above?

Discrete Internal Model Control

- Stop the simulation. Find the Discrete Menu, and click on *Discrete IMC - Furnace*. This will bring up the flowsheet in Figure 12.2. Discretize your first-order-plus-time-delay model, and place it in the *Discrete Model* block. Take the discrete IMC controller you developed in the Things to Think About section and enter the values in the correct blocks in the *IMC Controller* block. Do not forget to enter the sample time in the *Temperature Sensor*, *Discrete Model*, and controller blocks. Make a step change in the *Hydrocarbon Outlet Temperature Setpoint* from 609.9 K to 620 K.

Exercise 12.12 Sketch the response.

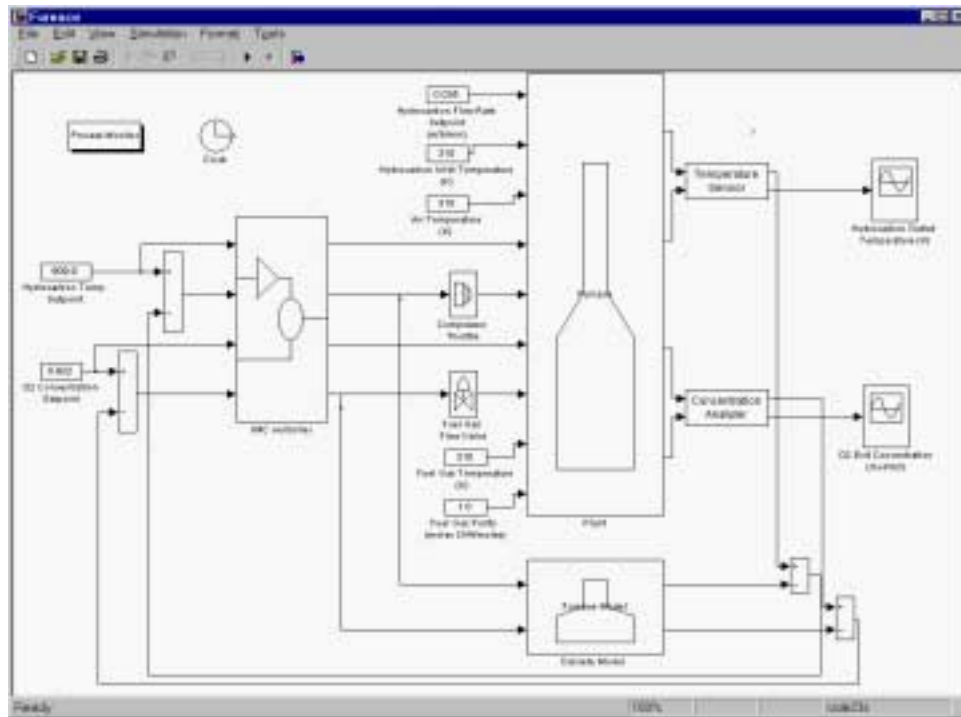


Figure 12.2. Flowsheet for the Furnace under Discrete IMC Control

Exercise 12.13 How does this response compare to those which were generated in the PID controller section?

8. Return the system to steady state. Examine the performance of the closed-loop in response to the same setpoint change for the various IMC filter values in the table below.

Exercise 12.14 Fill in the following table:

λ Filter Value	Overshoot	Rise time (min)	Settling time (min)
0.9			
0.5			
0.1			

PROCEDURE—COLUMN

Before you start the exercises, Click on the *Discrete* button on the Main Menu, then select *PID Control - Column* from the Discrete Menu. This will display the window in Figure 12.3

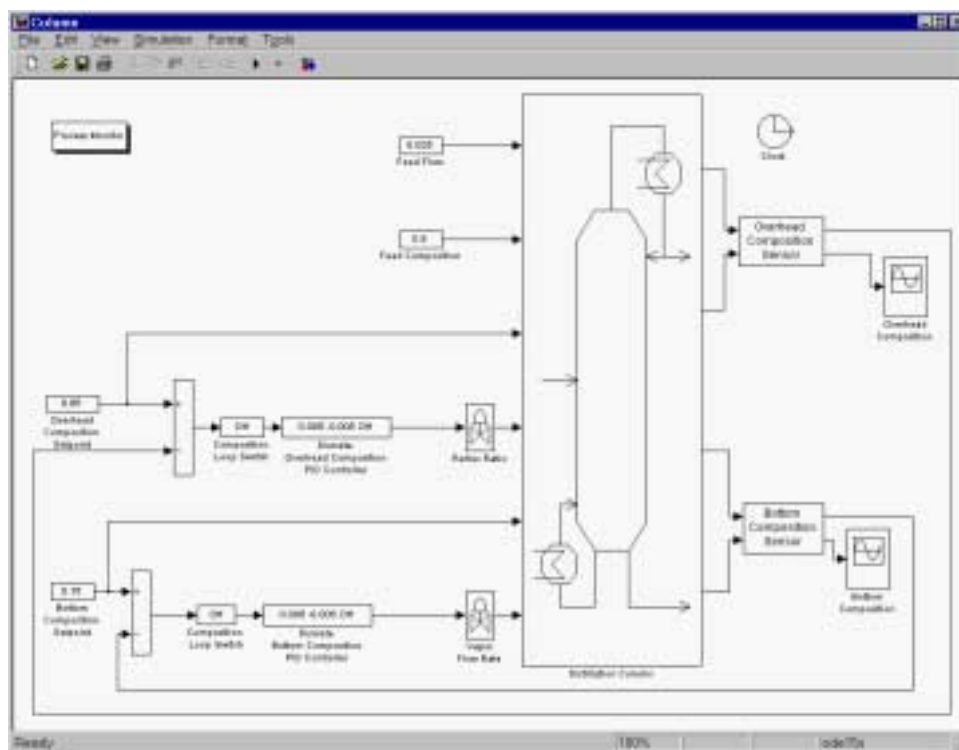


Figure 12.3. Flowsheet for the Column under Discrete PID Control

1. A good initial estimate of the sample time for discrete control is approximately $\frac{1}{5}$ of the dominant open-loop time constant, rounded such that any delay is an integer number of sample times.

Exercise 12.1 What is your first-order-plus-time-delay model for the *Reflux Ratio* to *Overhead MeOH Composition* loop? **Exercise 12.2** What sample time do you recommend using to control this loop? **Exercise 12.3** Using Equation 12.2 and your K_c , τ_I , and τ_D coefficients from Module 8, fill in the following table with the discrete-time tuning constants using Cohen-Coon rules for *Reflux Ratio* to *Overhead MeOH Composition* loop.

Controller Type	$\epsilon(k)$ Coefficient	$\epsilon(k-1)$ Coefficient	$\epsilon(k-2)$ Coefficient
P			
PI			
PID			

2. **Exercise 12.4** Use Ziegler-Nichols rules for the *Reflux Ratio to Overhead MeOH Composition* loop. Fill in the following table based on the continuous-time tuning constants from Module 8:

Controller Type	$\epsilon(k)$ Coefficient	$\epsilon(k-1)$ Coefficient	$\epsilon(k-2)$ Coefficient
P			
PI			
PID			

3. Tune the *Overhead MeOH Controller* using the Cohen-Coon discrete tuning constants for PI control. Remember to include the sample time in the controller block and in the *Overhead Composition Sensor*. Introduce a new composition setpoint of $0.89 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 12.5 Sketch the response. **Exercise 12.6** Note the settling time (time required for the *Overhead MeOH Composition* to remain within 5% of the new steady-state value).

4. Return the system to steady state by setting the *Overhead MeOH Composition Setpoint* to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Enter the Cohen-Coon derived controller tuning constants for PID control in the *Overhead Composition Controller*, and set the derivative action to *On*. Make the same setpoint change as above.

Exercise 12.7 Sketch the response. **Exercise 12.8** How does the use of derivative action affect the settling time?

5. Return the setpoint to its initial value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Once the system has reached steady state, double the sample time, and make the necessary changes in the *Overhead Composition Controller* and *Overhead Composition Sensor*. Also update the controller parameters (**REMEMBER: discrete-time PID tuning coefficients are sample time dependent**). Again set the *Overhead MeOH Composition Setpoint* to $0.89 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 12.9 How does the settling time compare to that achieved before?

6. Again return the system to steady state by setting the *Overhead MeOH Composition Setpoint* to $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Return the *Overhead Composition Sensor* and *Overhead Composition Controller* sample times to their initial values.

Replace the Cohen-Coon parameters with those calculated with the Ziegler-Nichols rules. Make the setpoint change described above.

Exercise 12.10 Compare the Ziegler-Nichols PID results to the PID results generated under Cohen-Coon tuning. Which is more aggressive?

Exercise 12.11 What is the settling time for this controller, and how does that compare with those found above?

Discrete Internal Model Control

- Stop the simulation. From the Discrete Menu, click on *Discrete IMC - Column*. This will bring up the flowsheet in Figure 12.4. Discretize your first-order-plus-time-delay model, and place it in the *Discrete Model* block. Take the discrete IMC controller you developed in the Things to Think About section and enter the values in the *Overhead Composition Controller* block. Do not forget to enter the sample time in the *Overhead Composition Sensor*, *Discrete Model*, and controller blocks. Make a step change in the *Overhead MeOH Composition Setpoint* from $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$ to $0.89 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 12.12 Sketch the response. **Exercise 12.13** How does this response compare to those that were generated in the PID controller section?

- Return the system to steady state. Examine the performance of the closed-loop in response to the same setpoint change for the various IMC filter values in the table below.

Exercise 12.14 Fill in the following table:

λ Filter Value	Overshoot	Rise time (sec)	Settling time (sec)
0.99			
0.9			
0.5			
0.1			

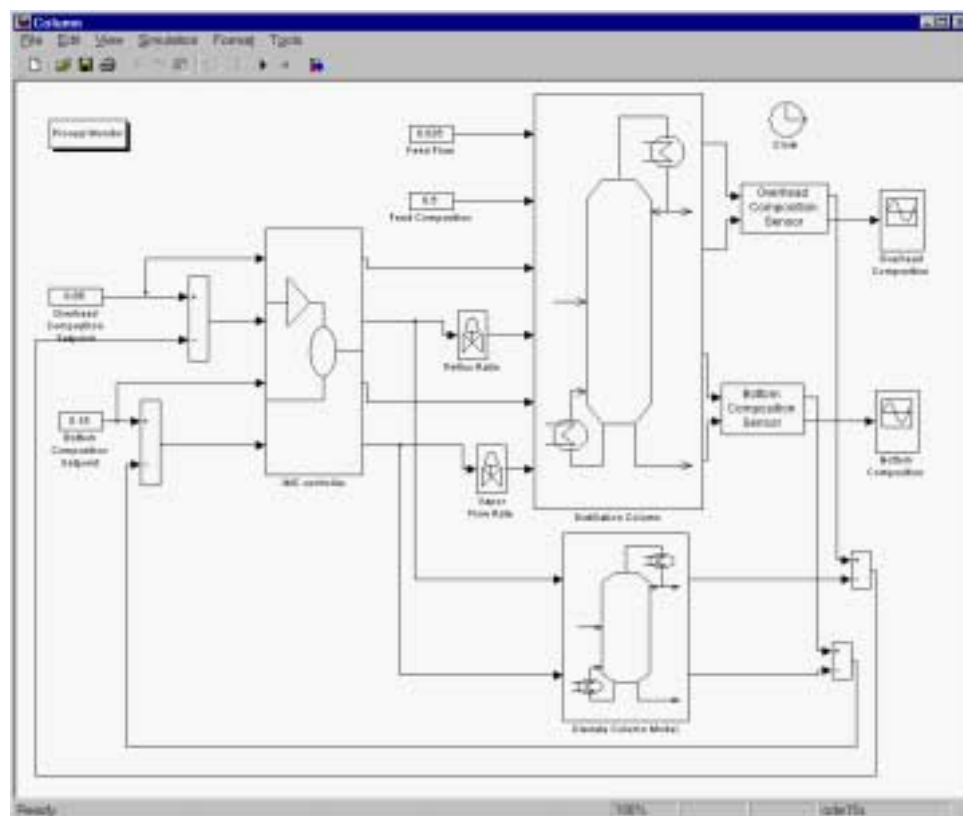


Figure 12.4. Flowsheet for the Column under Discrete IMC Control

SUMMARY

Discrete-time control is affected by controller tuning parameters as well as the sampling-time chosen for the system. Continuous-time tuning rules can be used for discrete-time control, and the ease of discrete-time IMC tuning is demonstrated as compared to trial-and-error procedures for PID controllers.

MODEL PREDICTIVE CONTROL

OBJECTIVE

The objective of this unit is to introduce the basic concepts involved in designing a Model Predictive Controller (MPC). Discrete-time models of the Furnace/Column will be created from existing continuous-time models. Various different controller parameters will be explored. These parameters include the output weights, input weights, model prediction horizon, and move horizon. An accurate linear model of the process will be provided. Comparisons between first-order-plus-time-delay models and the more accurate model will be made.

Open-Ended Module Procedure

Design a 2×2 MPC controller for the Furnace/Column using the previously developed FOTD models, and evaluate the setpoint response. Examine the sensitivity of the response to the 4 key tuning parameters in the MPC algorithm.

THINGS TO THINK ABOUT

Exercise 13-A What is a typical time constant for each of the input-output pairings in the furnace or column? To construct a discrete-time model, you should sample at a rate equal to approximately $\frac{1}{5}$ of the time constant of a process. What is a reasonable sampling rate for your system?

Exercise 13-B If one developed a discrete-time model of your process system by sampling every 0.5 sec (for the furnace) or every 60 sec (for the column), how many model coefficients would be needed to capture the complete response of the system to a step change in the input?

Exercise 13-C Two of the controller tuning parameters in MPC are the input move suppression weight and the output error weight. Explain how one could adjust these weights to make the controller more aggressive.

INTRODUCTION

In many chemical processes, there are complex interactions between manipulated and controlled variables. Simple SISO control is often suboptimal and MIMO decoupling can become very difficult to implement for large scale systems. One of the more advanced control algorithms being employed in industry for such problems is called Model Predictive Control (MPC) and is also known under the trade name of Dynamic Matrix Control (DMC).

As the name suggests, model predictive control makes use of an existing model of the process, typically a discrete-time step-response model. To develop a discrete-time step-response model, the sampling time and model memory must be determined. The issues involved in selecting the sampling rate were discussed in the last two modules. The model memory is the number of coefficients used in the model when discretized at the sampling times. Figure 13.1 shows the continuous and discrete step response for a second-order system with a delay. The continuous signal was sampled every 1 time unit. Typically, discrete step-response models are developed such that sampling time is less than $\frac{1}{5}$ the dominant time constant of the system. The model memory is chosen large enough so that the model reaches 98%

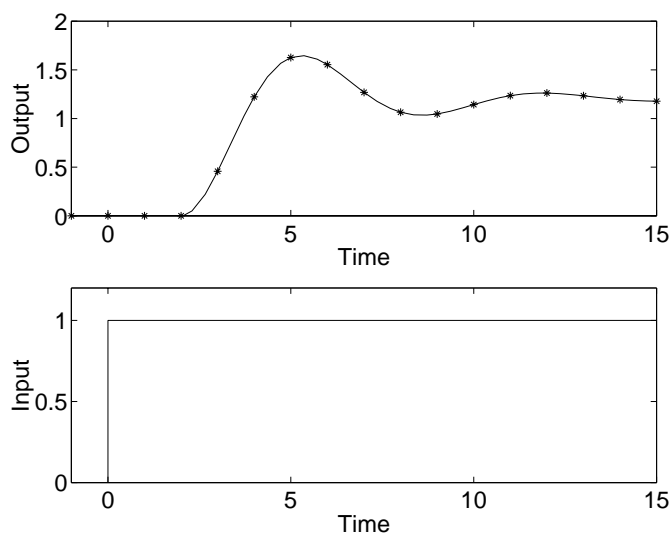


Figure 13.1. Discrete Sampling of a Delayed Second-Order-Transfer Function

of the steady state value (typically between 20 and 40). A smaller sampling time and more model coefficients lead to more accurate models, but the mathematical manipulations become more difficult.

Using the model of the system, predictions of future output values can be made. MPC solves an optimization problem by searching for an input move which minimizes a scalar cost function. The difference between the predicted output variables and their reference values is part of the cost function, so that the controlled variables optimally track the setpoint. The difference between control moves from one time to the next can also be incorporated in the cost function to minimize excessive use of system actuators and introduces an extra degree of freedom for controller tuning. The optimization problem which is solved at every time step can be written as:

$$\min_{k=1\dots p} \|y_r(k) - \hat{y}(k)\| + \|u(k) - u(k-1)\|$$

where:

- $y_r(k)$ is the reference trajectory value of the controlled variables at time k ;
- $\hat{y}(k)$ is the model estimate of the controlled variables at time k ;
- $u(k)$ is the predicted input move at time k ;
- p is the model prediction horizon; and
- $\|x\|$ is a norm of the vector x (typically 2-norm (energy))

PROCEDURE - FURNACE

1. From the *Main Menu*, click on *Furnace* and *Horizon-based Control - Furnace*. Locate your first-order-plus-time-delay models of the furnace which were developed in the Transient Response Analysis Module. Use the *pcm_c2d* command at the MATLAB prompt to convert your continuous-time models to discrete-time models. This function makes a discrete step-response model for the 2×2 system and places the step-response coefficients in variables *model1* and *model2* in the MATLAB workspace. Make sure that the discrete models capture the steady state and dynamic behavior of the furnace. If they do not, change the model memory or sampling time.

Exercise 13.1 What is the sampling time that you have determined for your system?

Exercise 13.2 What is the model memory you have selected? Plot the model and actual data for validation purposes.

2. Double-click on the *MPC Controller* block. Set the output error weight Γ_y to $\begin{bmatrix} 1 & 1 \end{bmatrix}$. The move horizon should be 1 and the prediction horizon should be 20. Change the setpoint for the *Hydrocarbon Outlet Temperature* to 600 K.

Exercise 13.3 Comment on the performance of your controller, noting, in particular, the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, 609.9 K . Change the setpoint for the *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.4 Comment on the performance of your controller, noting, in particular, the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, $0.922 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.5 Does the controller decouple the outputs in the system response?

3. MPC controllers are used on multivariable systems because they do not require the design of explicit decouplers to compensate for variable interactions. If the two controlled outputs, *Hydrocarbon Outlet Temperature* and *Oxygen Exit Concentration*, did not **both** track their respective setpoints, then the models generated in the Transient Response Analysis Module may be inaccurate. Replace your models with the following first-order-plus-time-delay models:

$$\begin{array}{ll}
 \text{Fuel Gas Flow Rate to Hydrocarbon Outlet Temperature} & \frac{220e^{-2s}}{6.5s + 1} \\
 \text{Air Flow Rate to Hydrocarbon Outlet Temperature} & \frac{-13e^{-2s}}{6.2s + 1} \\
 \text{Fuel Gas Flow Rate to Oxygen Exit Concentration} & \frac{2.0e^{-4s}}{3.8s + 1} \\
 \text{Air Flow Rate to Oxygen Exit Concentration} & \frac{0.14e^{-4s}}{4.2s + 1}
 \end{array}$$

From these models, use the `pcm_c2d` command to generate step response models. Incorporate these models into the controller, and implement the same setpoint change as in the last exercise.

Exercise 13.6 Does controller performance improve (in terms of steady state tracking, rise time, and settling time)?

4. **Use these new models for the remainder of this module.** Assume that the *Oxygen Exit Concentration* has been designated a higher priority for control. To account for this, the output error weight can be increased such that Γ_y is $\begin{bmatrix} 0.1 & 1 \end{bmatrix}$. Keep the move and prediction horizons the same. Change the setpoint for the *Hydrocarbon Outlet Temperature* to 600 K .

Exercise 13.7 Comment on the performance of your controller, noting, in particular, the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, 609.9 K . Change the setpoint for the *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.8 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, $0.922 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.9 Does the controller decouple the outputs in the system response?

Exercise 13.10 Can you detect a difference in the aggressiveness of the controller, with respect to the *Oxygen Exit Concentration*?

5. Now, consider a change in the input move weights to detune the controller and (potentially) reduce the amount of “chatter” in the system (excessive, high frequency actuator response). Set the input move weight Γ_u to $\begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$. Keep the output error weight Γ_y set to $\begin{bmatrix} 0.1 & 1 \end{bmatrix}$. The move and prediction horizons again remain the same. Change the setpoint for the *Hydrocarbon Outlet Temperature* to 600 K .

Exercise 13.11 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, 609.9 K . Change the setpoint for the *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.12 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, $0.922 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.13 Does the controller decouple the system?

Exercise 13.14 Can you detect a difference in the aggressiveness of the controller, with respect to the movement of the *Air Flow Rate* valve?

6. Now, change the move horizon to 2. Keep the input move weight Γ_u set at $\begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$. The output error weight Γ_y should also remain unchanged at $\begin{bmatrix} 0.1 & 1 \end{bmatrix}$. The prediction horizon should remain the same (20).

Change the setpoint for the *Hydrocarbon Outlet Temperature* to 600 K .

Exercise 13.15 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, 609.9 K . Change the setpoint for the *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.16 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, $0.922 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.17 Can you detect a difference in the aggressiveness of the controller, with respect to manipulated variable movement? With respect to the output tracking?

7. Keep the move horizon set to 2. If the air flow valve were more susceptible to damage from “chatter.” Adjust the input move weight Γ_u to $\begin{bmatrix} 1 & 0.1 \end{bmatrix}$. Keep the output error weight Γ_y at $\begin{bmatrix} 0.1 & 1 \end{bmatrix}$. The prediction horizon should remain the same. Change the setpoint for the *Hydrocarbon Outlet Temperature* to 600 K .

Exercise 13.18 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, 609.9 K. Change the setpoint for the *Oxygen Exit Concentration* to $1.0143 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.19 Record the rise and settling times for the outputs.

Return the setpoint to the normal steady state value, $0.922 \frac{\text{mol}}{\text{m}^3}$.

Exercise 13.20 Can you detect a difference in the behavior of the controller with the new tuning?

PROCEDURE—COLUMN

1. From the *Main Menu*, click on *Column* and *Horizon-based Control - Column*. Locate your first-order-plus-time-delay models of the column which were developed in the Transient Response Analysis Module. Use the *pcm_c2d* command at the MATLAB prompt to convert your continuous-time models to discrete-time models. This function makes a discrete step-response model for the 2×2 system and places the step-response coefficients in variables *model1* and *model2* in the MATLAB workspace. Make sure that the discrete models capture the steady state and dynamic behavior of the column. If they do not, change the model memory or sampling time.

Exercise 13.1 What is the sampling time that you have determined for your system? **Exercise 13.2** What is the model memory you have selected? Plot the model and actual data for validation purposes.

2. Double-click on the *MPC Controller* block. Set the output error weight Γ_y to $\begin{bmatrix} 1 & 1 \end{bmatrix}$. The move horizon should be 1 and the prediction horizon should be 20. Change the setpoint for the *Overhead MeOH Composition* to $0.90 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.3 Comment on the performance of your system, recording the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Change the setpoint for the *Bottom MeOH Composition* to $0.1 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.4 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.5 Does the controller decouple the outputs in the system response?

3. MPC controllers are used on multivariable systems because they do not require the design of explicit decouplers to compensate for variable interactions. If the two controlled outputs, *Overhead MeOH Composition* and *Bottom MeOH Composition*, did not **both** track their respective setpoints, then the models generated in the Transient Response Analysis Module may be inaccurate.

Replace your models with the following first-order-plus-time-delay models:

$$\begin{aligned}
 \text{Reflux Ratio to Overhead MeOH Composition} & \quad \frac{0.126e^{-138s}}{762s + 1} \\
 \text{Vapor Flow Rate to Overhead MeOH Composition} & \quad \frac{-1.1953e^{-548s}}{1252s + 1} \\
 \text{Reflux Ratio to Bottom MeOH Composition} & \quad \frac{0.124e^{-166s}}{840s + 1} \\
 \text{Vapor Flow Rate to Bottom MeOH Composition} & \quad \frac{-22.07e^{-26s}}{911s + 1}
 \end{aligned}$$

From these models, use the *pcm.c2d* command to generate step-response models. Incorporate these models into the controller, and implement the same setpoint change as in the last exercise.

Exercise 13.6 Does controller performance improve (in terms of steady state tracking, rise time, and settling time)?

4. **Use these new models for the remainder of this module.** Assume that the *Overhead MeOH Composition* has been designated a higher priority for control. To account for this, the output error weight can be increased such that Γ_y is $\begin{bmatrix} 1 & 0.1 \end{bmatrix}$. Keep the move and prediction horizons the same. Change the setpoint for the *Overhead MeOH Composition* to $0.90 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.7 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Change the setpoint for the *Bottom MeOH Composition* to $0.1 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.8 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.9 Does the controller decouple the outputs in the system response? **Exercise 13.10** Can you detect a difference in the aggressiveness of the controller, with respect to the *Bottom MeOH Composition*?

5. Now, consider a change in the input move weights to detune the controller and (potentially) reduce the amount of “chatter” in the system (excessive, high frequency actuator response). Set the input move weight Γ_u to $\begin{bmatrix} 1 & 10 \end{bmatrix}$. Keep the output error weight Γ_y set to $\begin{bmatrix} 1 & 0.1 \end{bmatrix}$. The move and prediction horizons again remain the same. Change the setpoint for the *Overhead MeOH Composition* to $0.90 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.11 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Change the setpoint for the *Bottom MeOH Composition* to $0.1 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.12 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.13 Does the controller decouple the outputs in the system response? **Exercise 13.14** Can you detect a difference in the aggressiveness of the controller, with respect to the movement of the *Reflux Ratio* and *Vapor Flow Rate* valves?

6. Now, change the move horizon to 2. Keep the input move weight Γ_u set at $\begin{bmatrix} 1 & 10 \end{bmatrix}$. The output error weight Γ_y should also remain unchanged at $\begin{bmatrix} 1 & 0.1 \end{bmatrix}$. The prediction horizon should remain the same.

Change the setpoint for the *Overhead MeOH Composition* to $0.90 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.15 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Change the setpoint for the *Bottom MeOH Composition* to $0.1 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.16 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.17 Can you detect a difference in the aggressiveness of the controller, with respect to manipulated variable movement? With respect to the output tracking?

7. Keep the move horizon set to 2. Adjust the input move weight Γ_u to $\begin{bmatrix} 3 & 10 \end{bmatrix}$. Keep the output error weight Γ_y at $\begin{bmatrix} 1 & 0.1 \end{bmatrix}$. The prediction horizon should remain the same. Change the setpoint for the *Overhead MeOH Composition* to $0.90 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.18 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.85 \frac{\text{mol MeOH}}{\text{mol total}}$. Change the setpoint for the *Bottom MeOH Composition* to $0.1 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.19 Record the rise and settling times for the outputs. Return the setpoint to the normal steady state value, $0.15 \frac{\text{mol MeOH}}{\text{mol total}}$.

Exercise 13.20 Can you detect a difference in the behavior of the controller with the new tuning?

SUMMARY

In this unit you have developed a model predictive controller. Different input move and output error weighting values were used for controller tuning. The effect of changing the move horizon was also explored. The use of an improved system model was shown to improve closed-loop performance.

ABOUT THE CD

This CD that accompanies the book contains the files needed to run PCM using MATLAB version 5.1 or higher and SIMULINK version 2.0 or higher.

The files are located on the CD in the *pcm* directory. PCM can be run directly from the CD, but it is recommended that you install PCM on your computer's hard drive. To install PCM, copy the *pcm* directory to your local hard drive.

To start PCM, type *mainmenu* at the MATLAB prompt. Note that you must either currently be in the *pcm* directory or have the *pcm* directory added to your MATLAB path. You can use the *pwd* command to determine your current working directory. The *cd* command can be used to change directories. MATLAB comes with a path browser that allows you to view and change your path. You can also manually change the path by including an appropriate *addpath* command in your *matlabrc.m* file.

For PC users, you may need to copy the files *mths110.dll* and *plbs110.dll* to your system directory before starting MATLAB. System directories may be *C:\Windows\system* or *C:\WINNT\system32*. These files are on the CD-ROM and may be hidden from view because they have a *.dll* extension. If so, select *View, Options, View*, and *Show all files*.

The column and furnace simulations rely on process models. MATLAB can use interpreted models or compiled models. In our experience, compiled models run much faster. We have provided compiled models for MATLAB version 5.3 on PC and Solaris systems. If you have access to a C compiler that is MATLAB compatible, you can use the command *mex* to compile the models for your system or for newer/older versions of MATLAB. The files that need compiling are: */pcm/furnace/pufcore4.c*, */pcm/column/newcolco.c*, and */pcm/column/newcol.c*. For the furnace, the resulting compiled file should be named *pufcore4.dll* for PC versions or *pufcore4.mexsol* for Solaris.

If you cannot use *mex* to compile these files or if you have trouble running the simulations, you can use the slower interpreted version of the furnace model. Remove or rename the compiled versions of the furnace model so that MATLAB will use the file *pufcore4.m* as the model. MATLAB looks for a compiled version before using the interpreted version.

For information and updates, see the WWW page for the book at:
<http://che.udel.edu/pcm>