

UCSB CHEMICAL ENGINEERING

Process Control Modules

Tutorials

Chemical Engineering

05/26/2010

Contents

Introduction	3
Module organization.....	4
Hardware and software requirements	5
Installation	5
Running the software.....	5
Furnace	7
Introduction	7
Furnace modules.....	9
Operator Interface	9
Proportional-Integral-Derivative (PID) Control.....	11
Feedforward Control.....	12
Multivariable Control	13
Model Predictive Control (MPC)	14
Furnace Tutorial.....	16
Open-loop identification	16
Closed-loop control - PID	20
Distillation Column.....	25
Introduction	25
Column modules	27
Operator Interface	27
Proportional-Integral-Derivative (PID) Control.....	29
Feedforward Control.....	30
Multivariable Control	31
Decoupling	32
Model Predictive Control (MPC)	33
Distillation Column Tutorial	35
Open-loop identification	35
Closed-loop control - PID	39
References	43

Introduction

The Process Control Modules (PCM), originally developed at Purdue University by Prof. Doyle and colleagues have been designed 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 (Doyle III, Gatzke et al. 1998; Doyle III, Gatzke et al. 2001). These modules have been updated and adapted, by Dr. Eyal Dassau at the University of California Santa Barbara, to be used in conjunction the 3rd edition of *Process Dynamics and Control*. The primary objectives in developing these MATLAB® modules were to:

- Develop realistic computer simulation case studies based on physical properties that exhibited nonlinear, high-order dynamic behavior in a rapid simulation environment
- 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

Eight distinct unit operations and control examples, which range from single input single output (SISO) to 2-by-2 control loops, are formulated with a modular approach. The progression of the modules follows a typical undergraduate process-control textbook, starting with low-order dynamic system analysis and continuing through multivariable control synthesis.

Module	Modes						
Furnace	Operator Interface	PID	Feedforward	Multivariable	MPC		
Distillation Column	Operator Interface	PID	Feedforward	Multivariable	Decoupling	MPC	
Bioreactor	Operator Interface	PID	Feedforward	Multivariable	Decoupling		
Four Tanks	Operator Interface	PID	Feedforward	Multivariable	MPC		
Fermentor	Operator Interface	PID					
Diabetes	Operator Interface	Bergman	PID	Feedforward	MPC		
First and Second Order Systems	First Order System	Second Order System	System Identification #1	System Identification #2			
Discrete	Aliasing	Model ID	PID- Furnace	PID- Column	PID- Four Tanks	IMC- Furnace	IMC- Column

Hardware and software requirements

The Process Control Modules are a set of MATLAB/SIMULINK routines that require either a full license or the Student Version of MATLAB and SIMULINK. The current version of the modules has been tested with version 2007a of MATLAB and SIMULINK. The minimum recommended system configuration is a Windows (XP or Vista) with 1 GB RAM.

Installation

Download the Process Control Modules (PCM) software from www.wiley.com/college/seborg onto your own computer. Then double-click on the PCM file and follow the instructions on the installer to install the software. Note that you should have MATLAB on your computer in order to use these modules. During the installation, you will be asked if you would like to create a shortcut icon to the software on your desktop (recommended).

Running the software

There are two ways to execute the software; the first is to double-click the PCM button on the desktop, which will launch MATLAB and the PCM interface (Figure H.1). The other way is to manually open MATLAB and to call the PCM software by pointing to the PCM installation folder and typing *PCM* followed by <enter>.

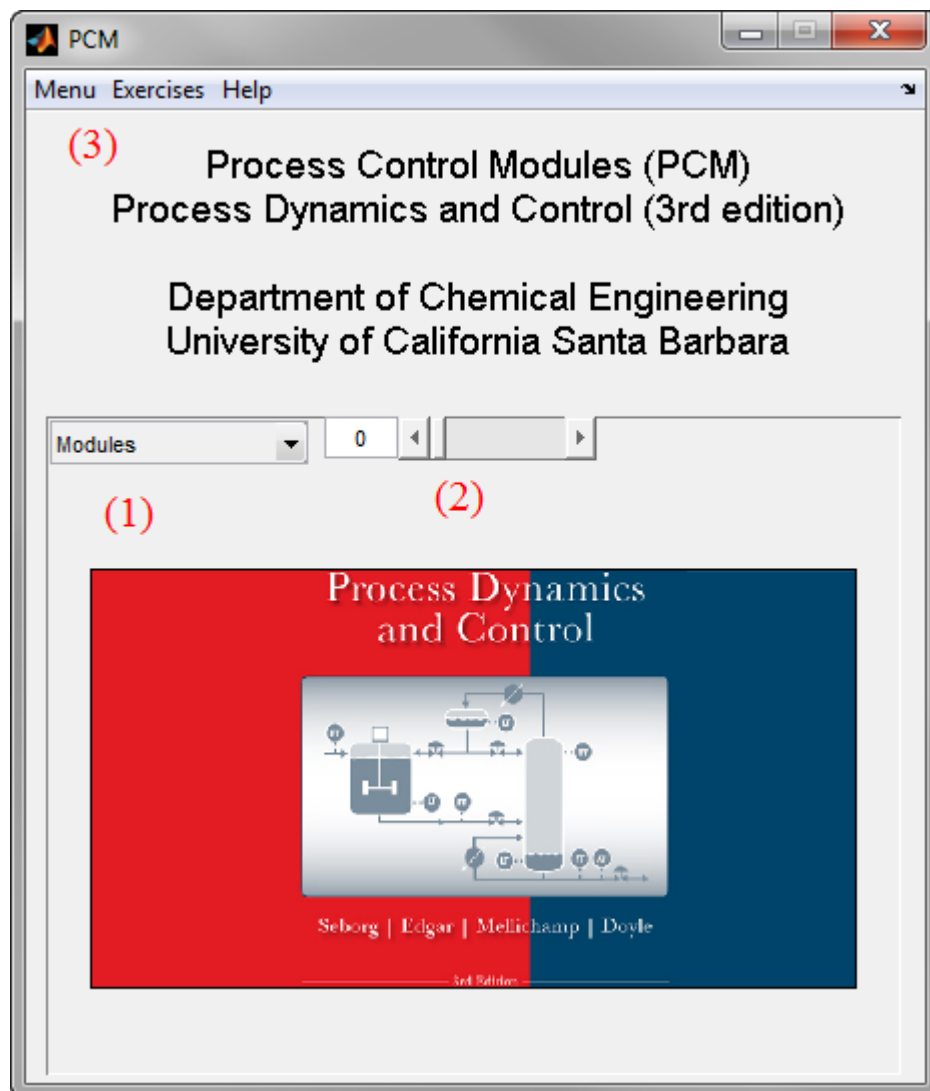


Figure 1 - PCM main interface

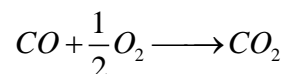
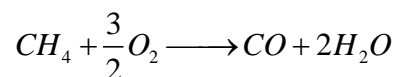
Furnace

Introduction

The Purdue Furnace is a model of a hydrocarbon heating furnace. A hydrocarbon stream is passed through a shell tube furnace. The air flow rate and fuel gas flow rate are considered manipulated variables. Hydrocarbon flow rate and fuel gas purity are two of the disturbances. The hydrocarbon outlet temperature, furnace temperature, exhaust gas flow rate, and O₂ exit concentration are all measured outputs.

The unit operation in this module represents a furnace fueled by natural gas which is used to preheat a high molecular weight hydrocarbon feed (C₁₆ – C₂₆) to a cracking unit at a petroleum refinery. The furnace model consists of energy and component mass balances that result in coupled nonlinear differential equations.

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, to minimize fuel costs, the furnace must be operated with an Oxygen composition that ensures 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.

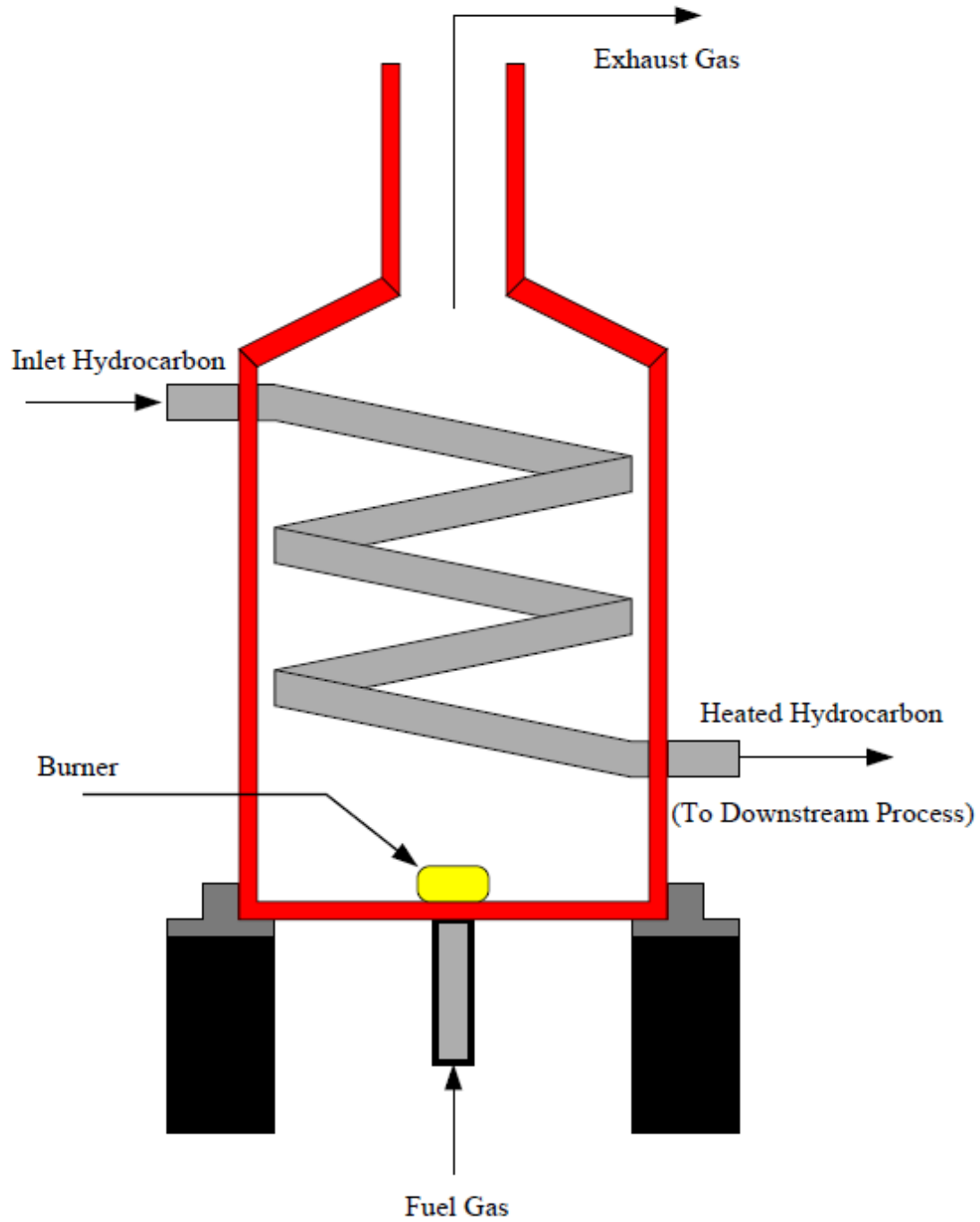


Figure 2 – Process schematic of the furnace

The furnace simulation can be executed by selecting *Furnace* from the “Modules” menu. Five different control simulations can be performed, each by clicking on one of the push buttons as appears in Figure 3.

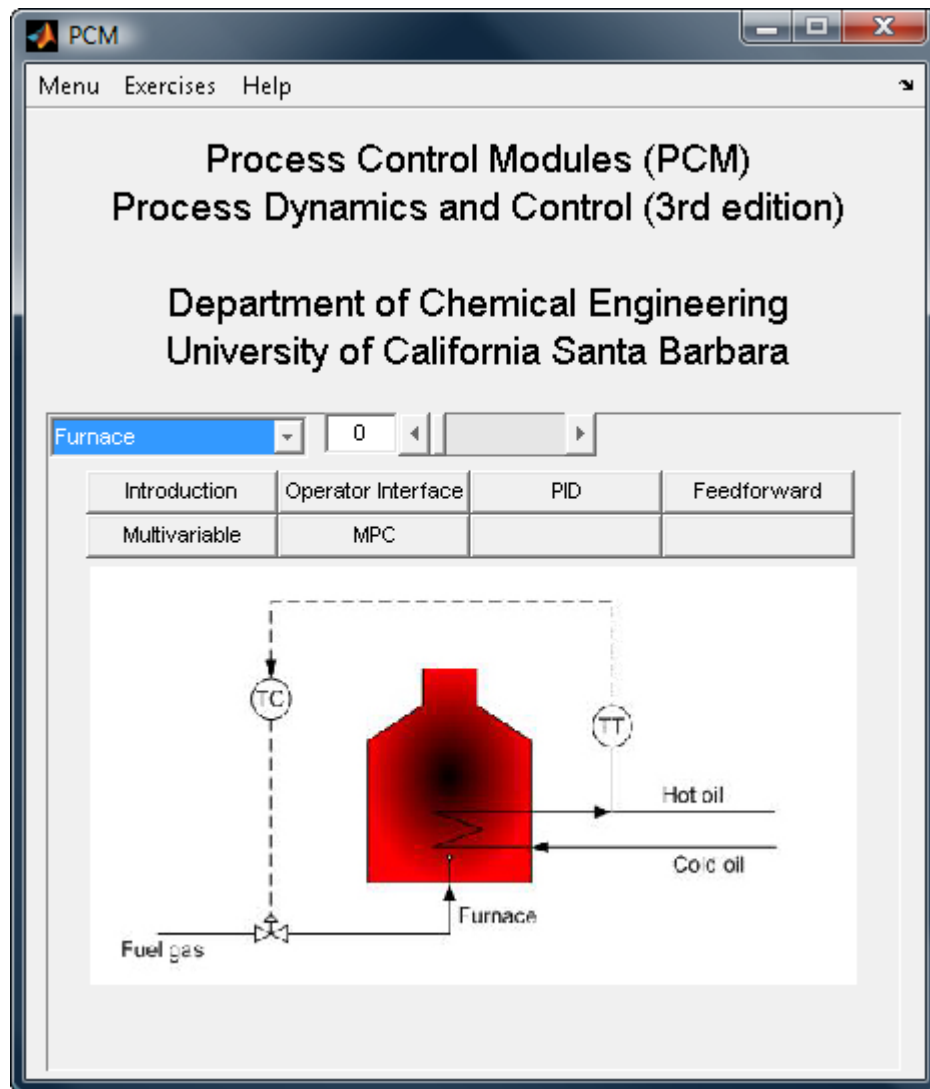


Figure 3 – Furnace interface

Furnace modules

Operator Interface

The Operator interface can be used for manual operation and control of the furnace, as well as process modeling. You can manually adjust the inputs and monitor the outputs using the monitor (Figure 4). The module allows you to introduce different Simulink blocks, such as “step,” to assist in model identification.

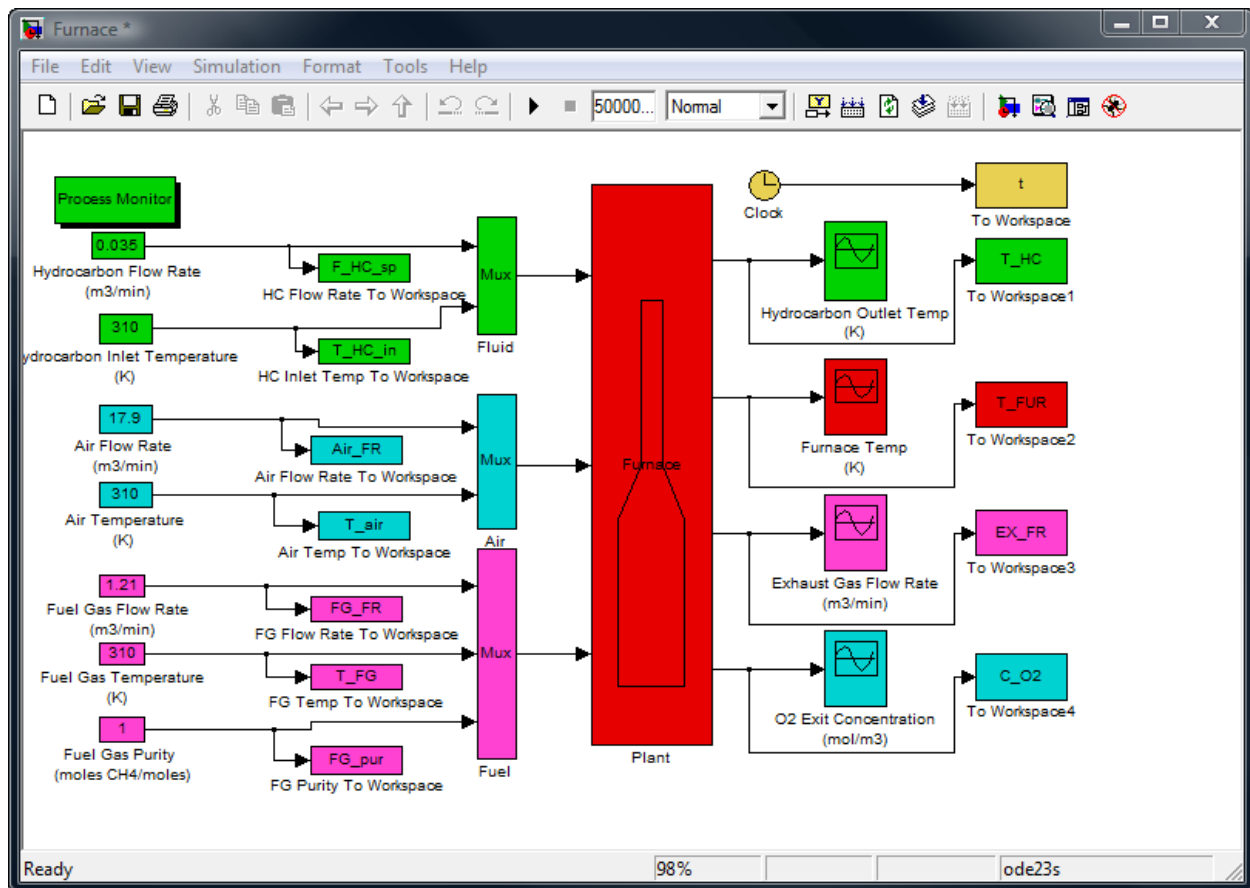


Figure 4 – Furnace operator interface

As can be seen from Figure 5, a step of 20% in air flow rate was introduced at ~ 30 min by pausing the simulation and manually adjusting the air flow rate. This information is available to you by clicking on the save button in the process monitor (Figure 5) and selecting a name for the MATLAB .mat file.

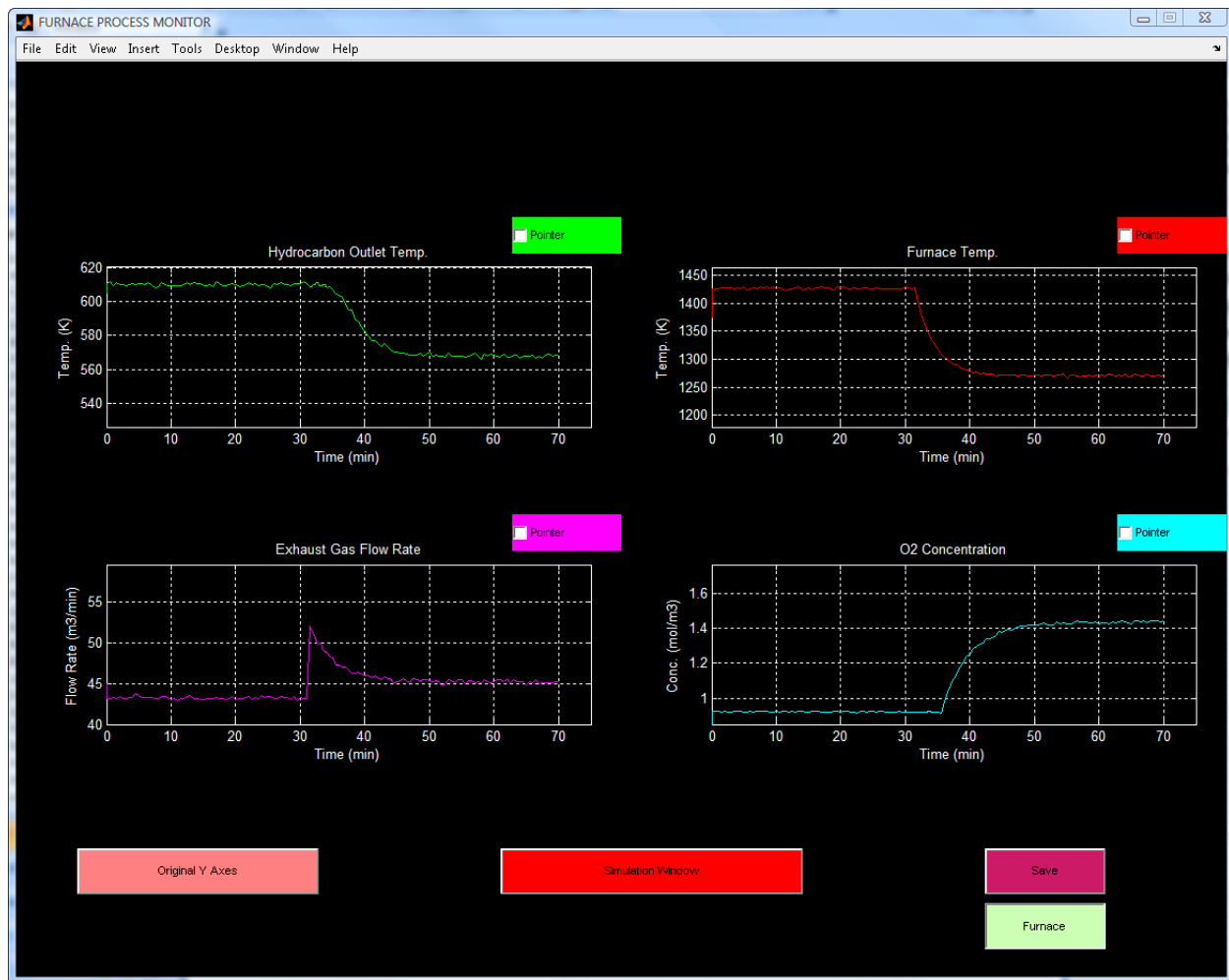


Figure 5 – Furnace monitor

Proportional-Integral-Derivative (PID) Control

In this module the characteristics of some of the various types of feedback control action and their influence on the performance of the Furnace will be studied. Of particular interest are the impacts of controller gain and reset time on the offset between the output and the setpoint at steady state. The Proportional-Integral-Derivative (PID) controller will be used in this module. A trial-and-error selection process for PID controller tuning constants requires a lengthy iterative procedure. In this module, you can experience different tuning algorithms that produce good initial estimates of controller gain (K_c), Integral reset time (τ_i), and derivative reset time (τ_D).

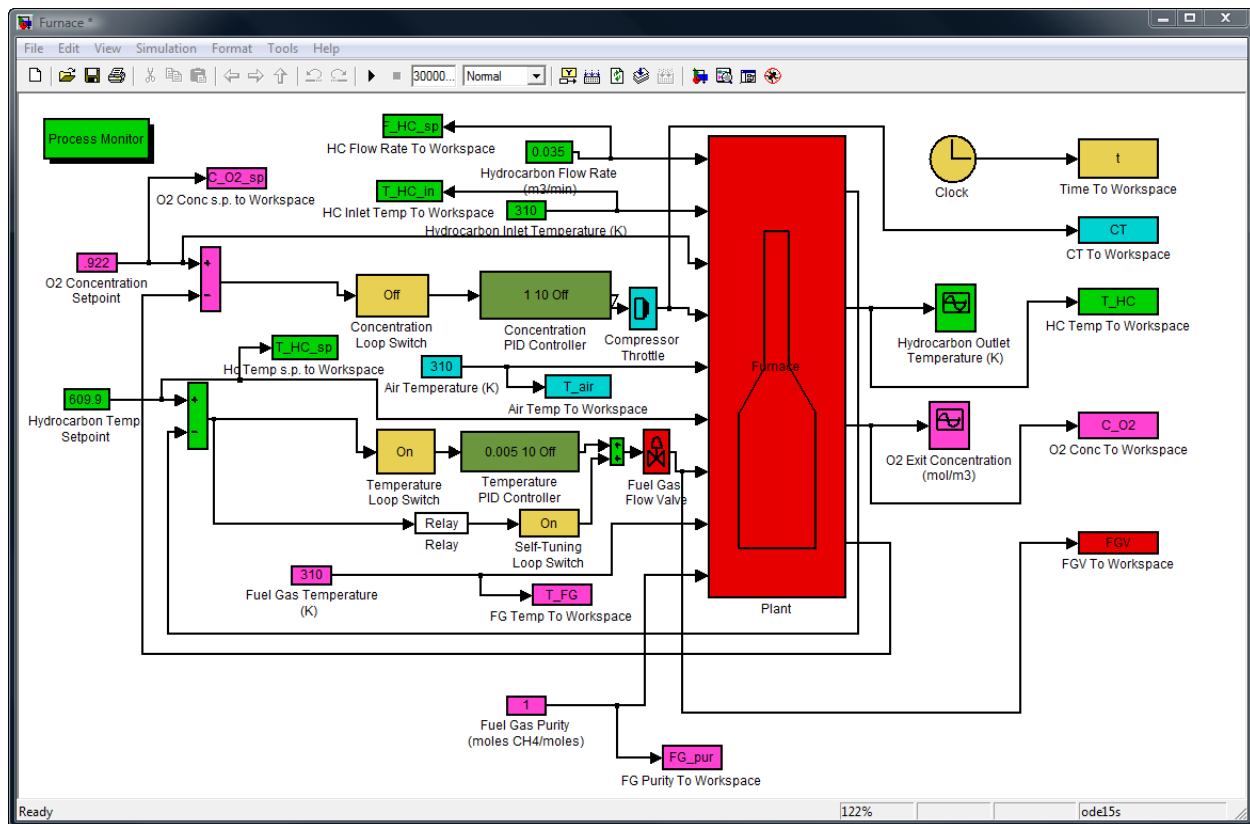


Figure 6 – Furnace PID interface

Feedforward Control

In the PID module, 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 taken only 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 can implement a feedforward strategy on the Furnace to reduce the effect of disturbances on process outputs.

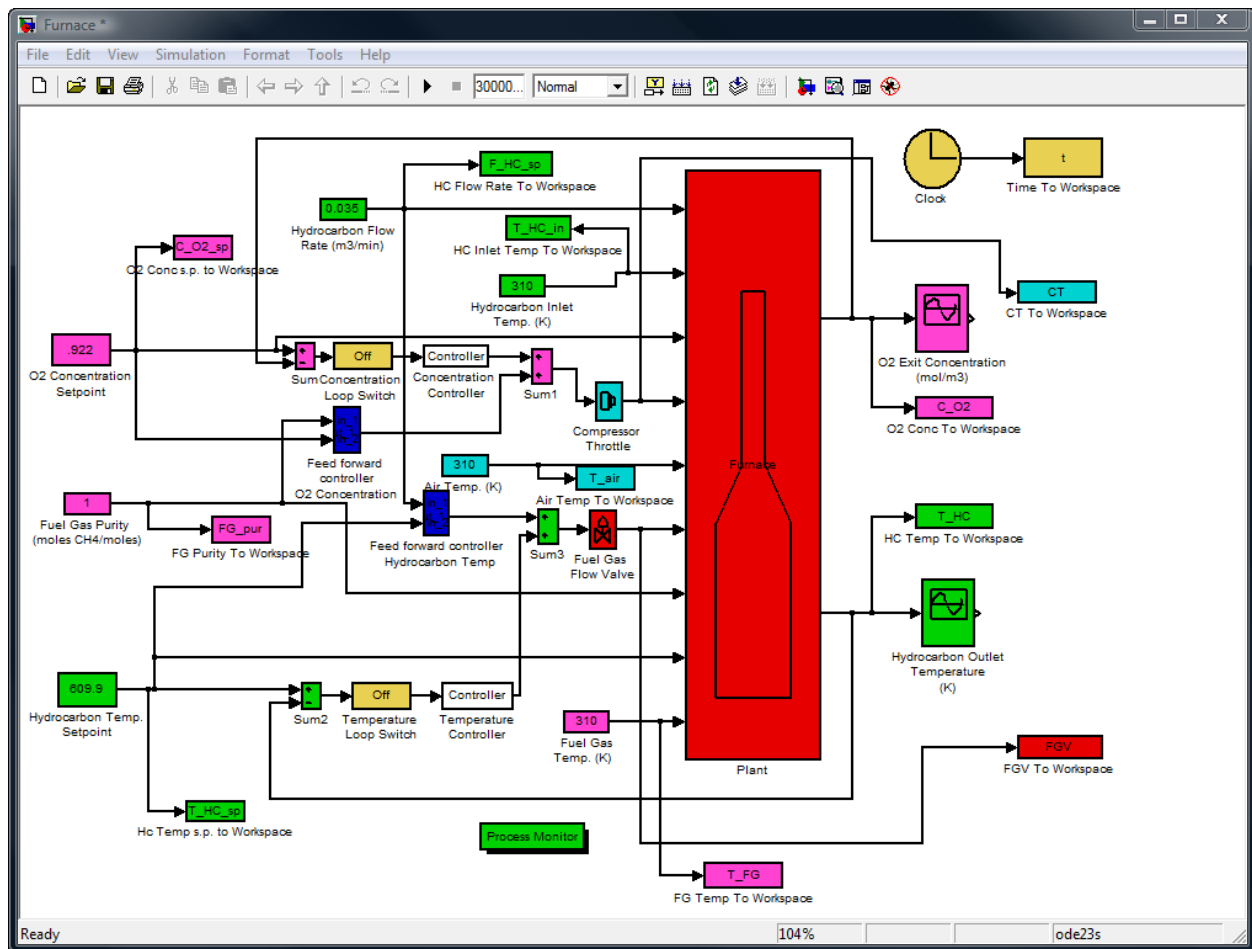


Figure 7 – Furnace Feedforward interface

Multivariable Control

In this module we will address model-based control and multivariable control. The first-order-plus-time-delay models you validated previously will be used to design single-loop Internal Model Control (IMC) controllers as well as decouplers that reduce interaction between the two control loops.

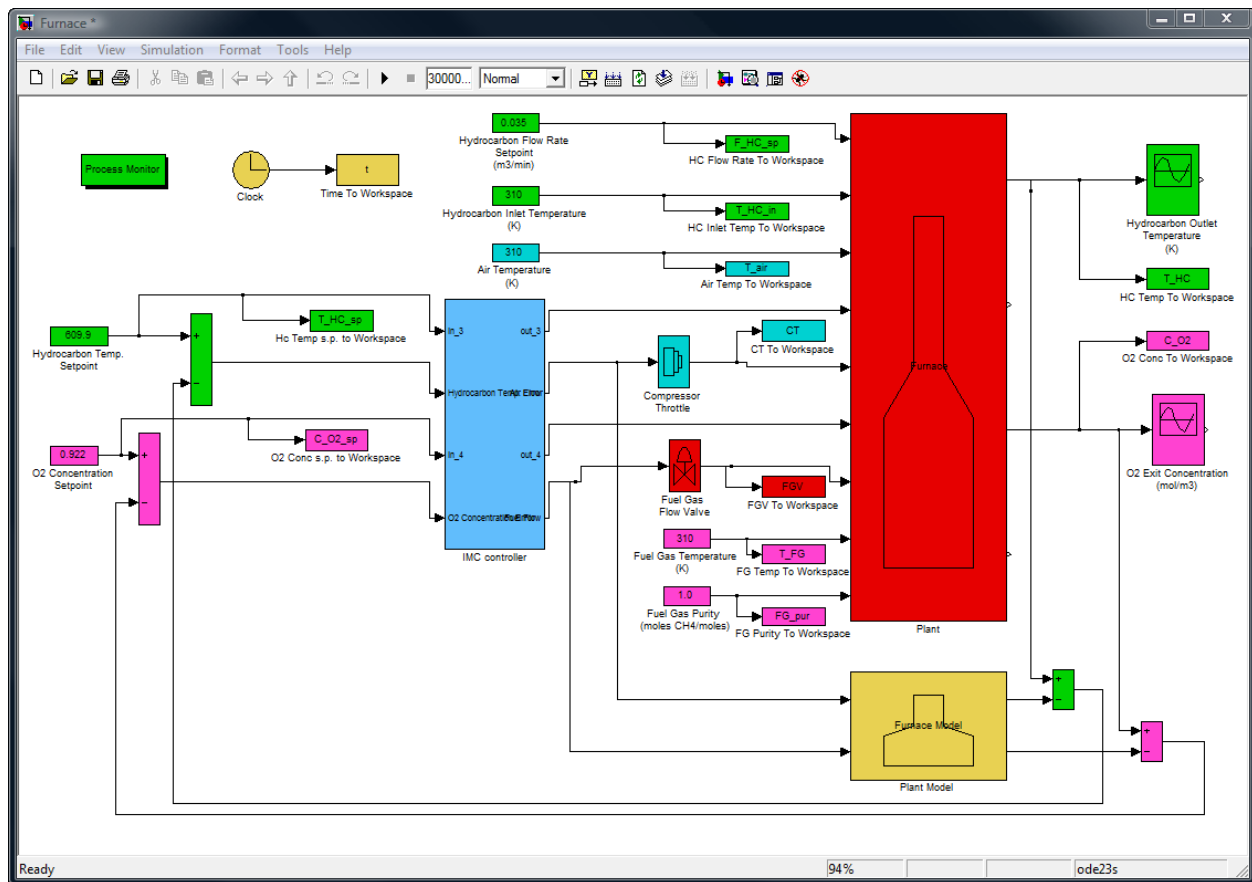


Figure 8 – Furnace multivariable interface

Model Predictive Control (MPC)

This module allows you to experience the basic concepts involved in designing a Model Predictive Controller (MPC). Discrete-time models of the Furnace 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.

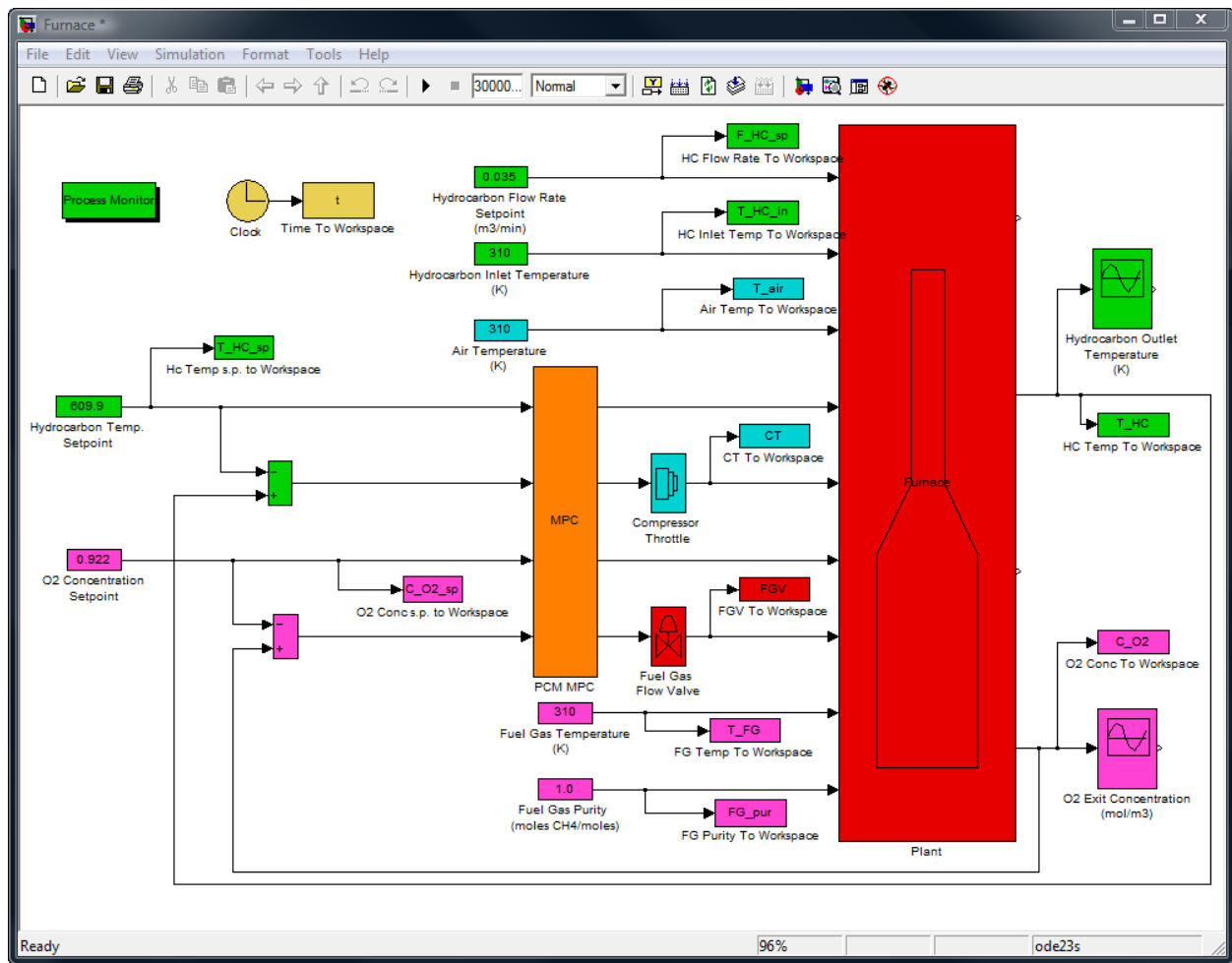


Figure 9 – Furnace MPC interface

Furnace Tutorial

Open-loop identification

1. Open and initialize the Furnace simulation

- Type *0.1* into the text box that initially shows a value of *0*. The new value reduces the speed at which the real-time plots are plotted.
- Next, select *Furnace* from the drop-down “Modules” menu.
- Click the “Introduction” button and read the information about the Purdue Furnace.

2. Organize MATLAB windows

- Click the “Operator Interface” button. The “Furnace” and “FURNACE PROCESS MONITOR” windows will open, although one window may initially be hidden behind the other.
- In the “FURNACE PROCESS MONITOR” window, click the drop down menu for the Desktop and select “DOCK FURNACE PROCESS MONITOR”.
- Then resize the MATLAB and Furnace windows so that you can see both windows clearly at the same time.

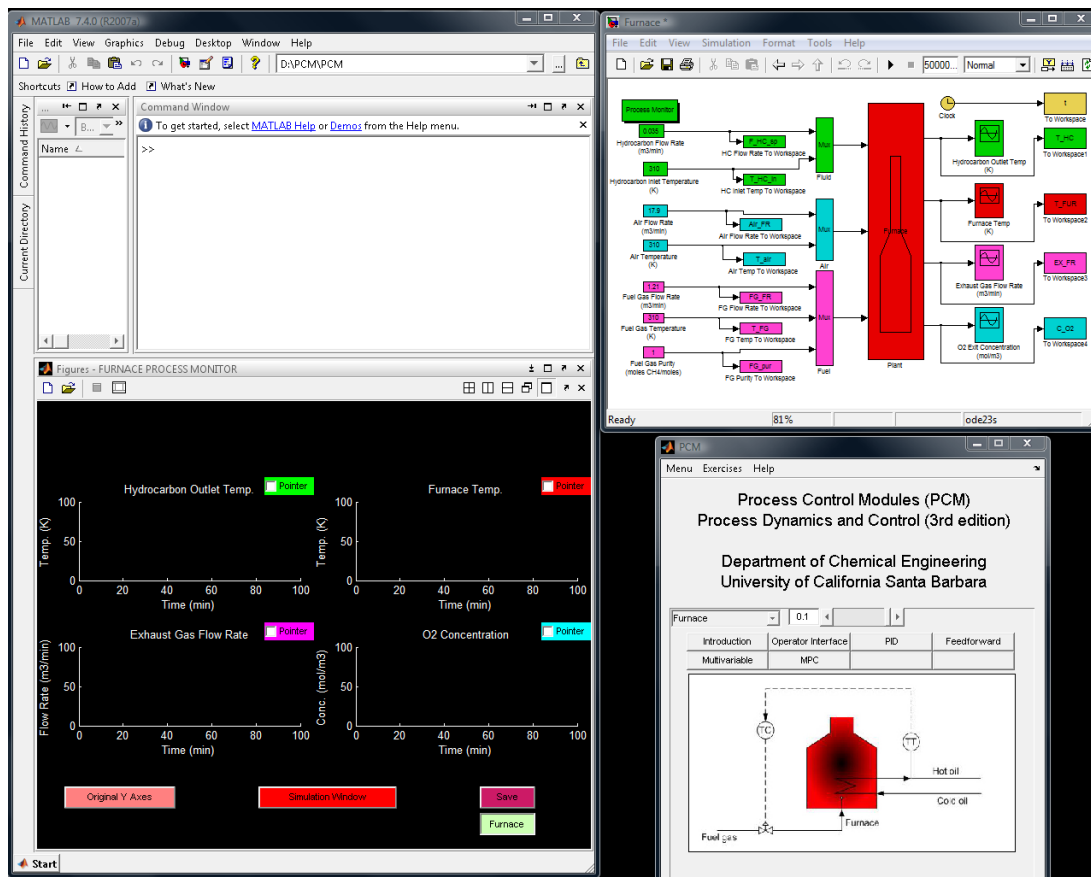


Figure 10 – Suggested Matlab window organization

3. Simulate the Furnace response to three step changes in the Air Flow Rate: +1.8 m³/min, then -1.8 m³/min, and back to the nominal value

- There are multiple ways to introduce a series of step changes in Simulink:
 - i. Manually change the value of interest by pausing the simulation at each step point. It is useful to keep track of the simulation time; note when this step was done for future analysis.
 - ii. Introduce a series of step blocks instead of the constant value

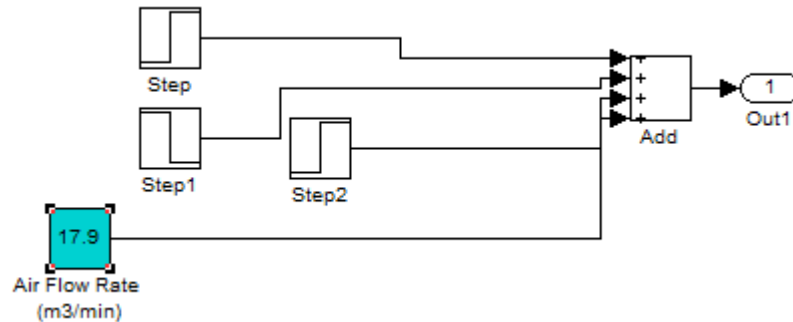


Figure 11 – A series of step changes implemented using several step blocks and a nominal constant

- iii. Create a 2-D variable in the command window using time and value columns then replace the constant input by a workspace input block

```
>> Air_Flow_Rate_Step=[0 17.9;64 17.9;64 19.7;166 19.7;166 16.1;274 16.1;274 17.9]
```

```
Air_Flow_Rate_Step =
```

```

      0    17.9000
    64.0000    17.9000
    64.0000    19.7000
   166.0000    19.7000
   166.0000    16.1000
   274.0000    16.1000
   274.0000    17.9000

```

```
>>
```

Figure 12 – A series of step changes implemented as a 2-D variable in the command window

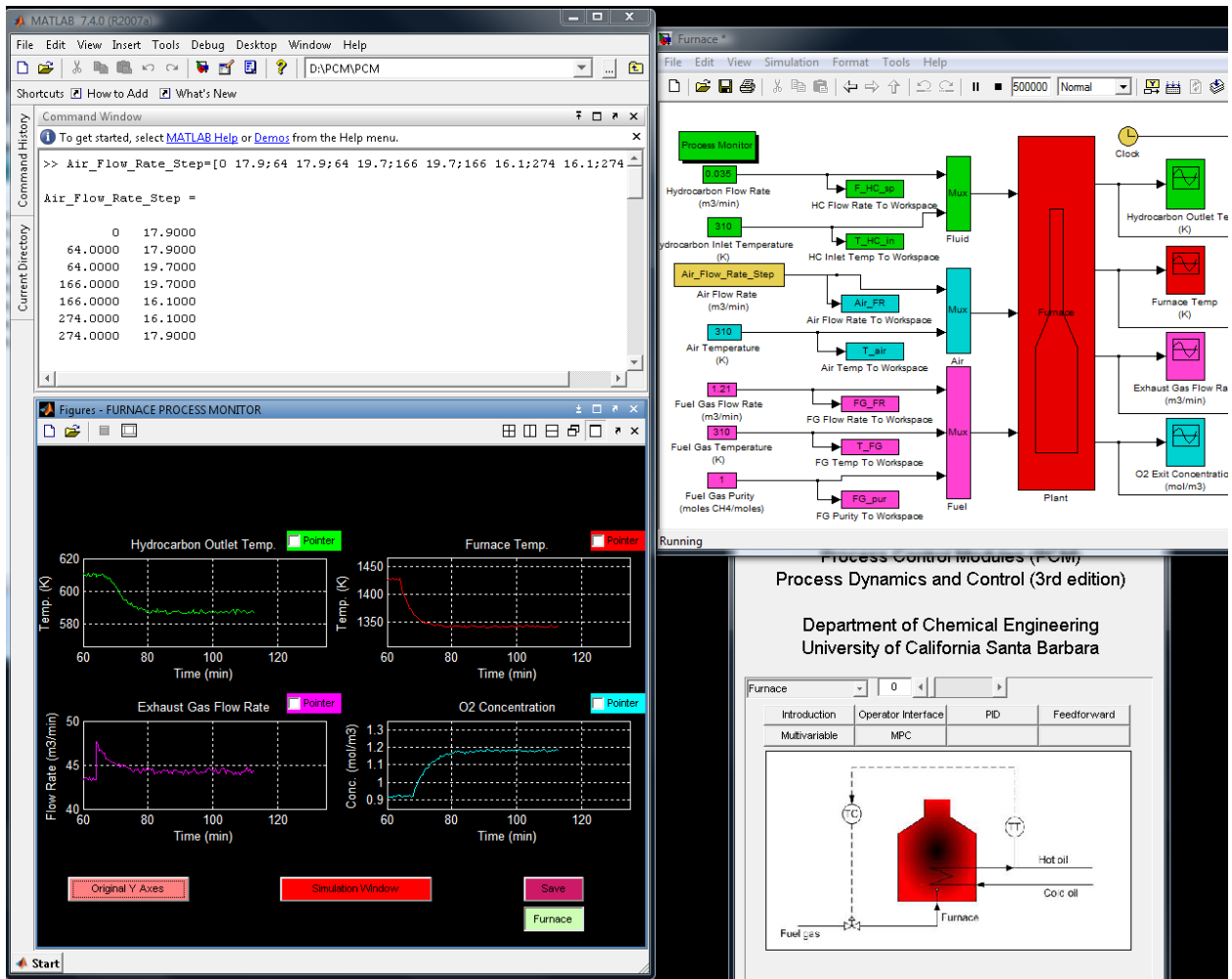


Figure 13 – A screenshot of the simulation results and the definition of the 2-D Air_Flow_Rate_Step

4. Save the results using the save button. You may change the name of the saved values. Alternatively, you can use the values that were exported to your workspace, such as the Air_FR and T_HC.

- Plot both the step change and the output variable, and calculate a first-order-plus-time-delay function

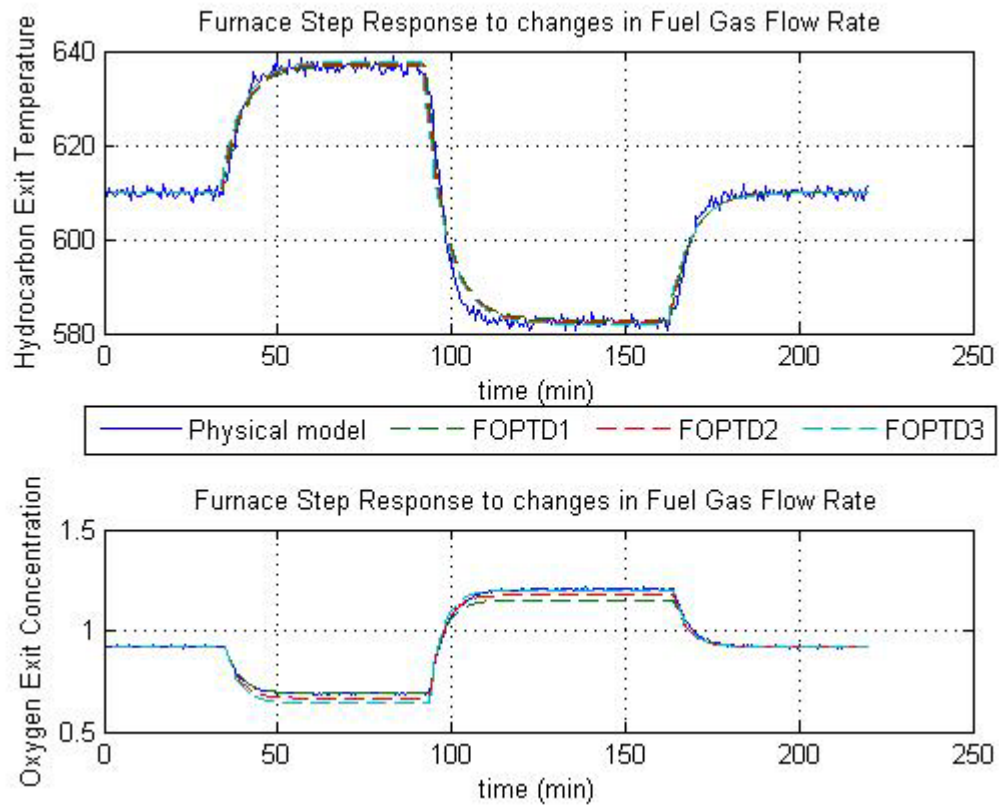


Figure 14 – Simulation results for a step change in the Air_Flow_Rate

- Repeat this for the other manipulated variables.

Closed-loop control - PID

1. Open and initialize the Furnace simulation

- Type *0.1* into the text box that initially has a value of *0*. The new value reduces the speed at which the real-time plots are plotted. Note, to slowdown the real-time plots used a higher number, a value of *.1* should be sufficient for ost systems.
- Next, select “Furnace” in the drop down “Module” menu.

2. Organize MATLAB windows

- Click on the “PID” button. The “Furnace” and “FURNACE WITH CONTROL PROCESS MONITOR” windows will open, although one window may initially be hidden behind the other.
- In the “FURNACE WITH CONTROL PROCESS MONITOR” window, click the drop down menu for the Desktop and select “DOCK FURNACE WITH CONTROL PROCESS MONITOR”.
- Then, resize the MATLAB and Furnace windows so that you can see both windows clearly at the same time.

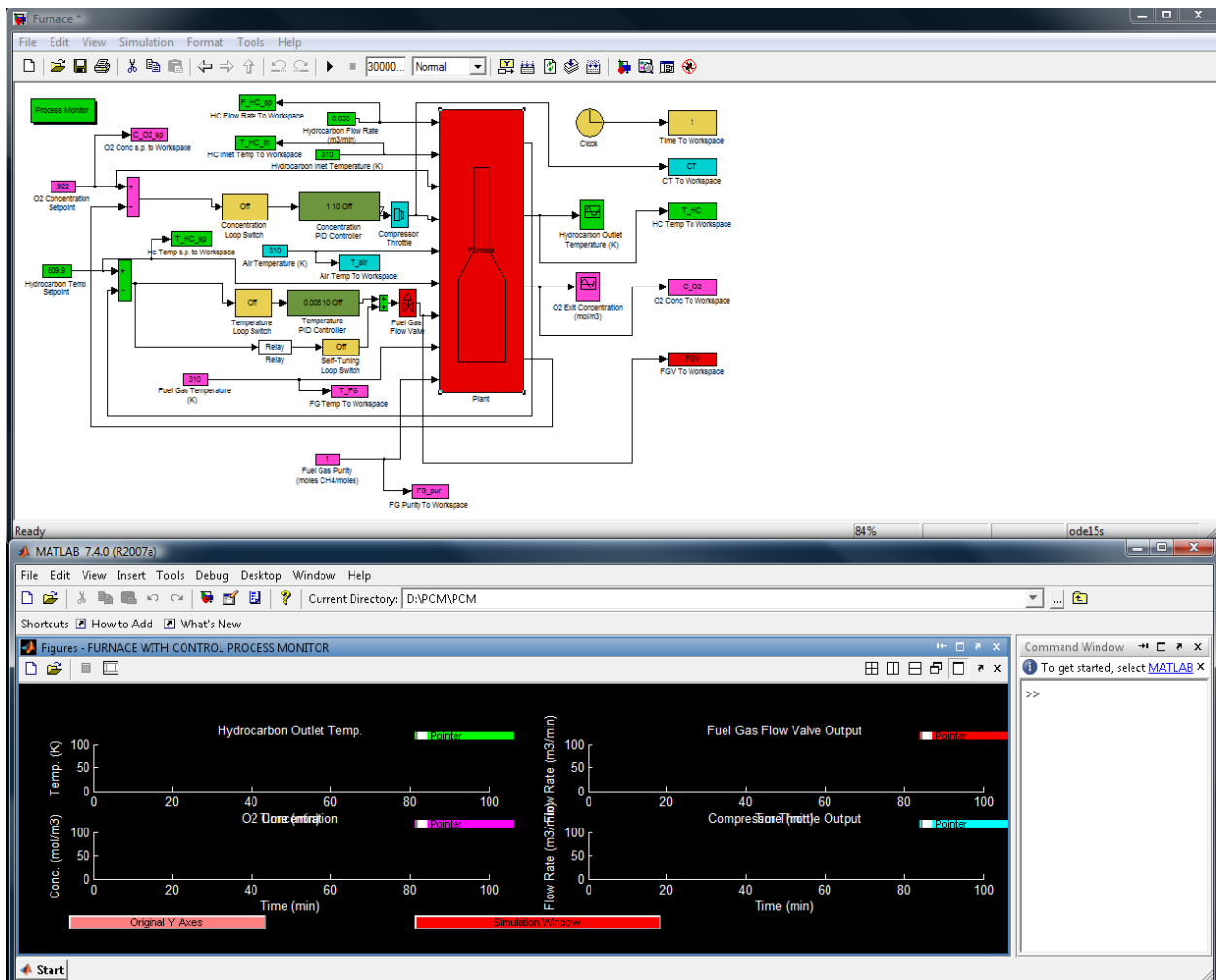


Figure 15 – Screenshot of the closed-loop control using PID

3. Controller settings

- Calculate the PID control settings, based on IMC tuning rules, for both control loops.
- Enter these settings into the PID Controllers (Green boxes) and change the Derivative Action to 1.

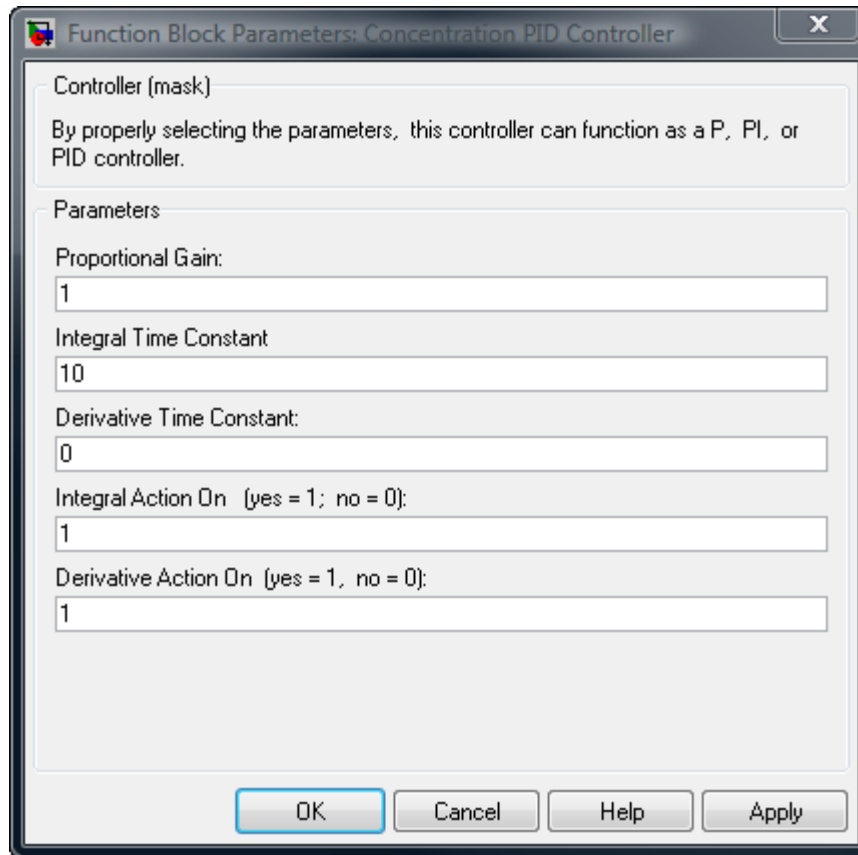


Figure 16 – PID tuning interface

4. Simulate the Furnace response to a step changes O_2 concentration

- Switch the Oxygen loop to *On* by clicking the toggle switch.

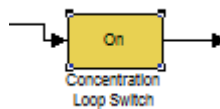


Figure 17 – Oxygen loop switch

- Introduce a step change in the Oxygen (O_2) concentration: 0.0923 at 200min and back to the nominal value after 400 min.

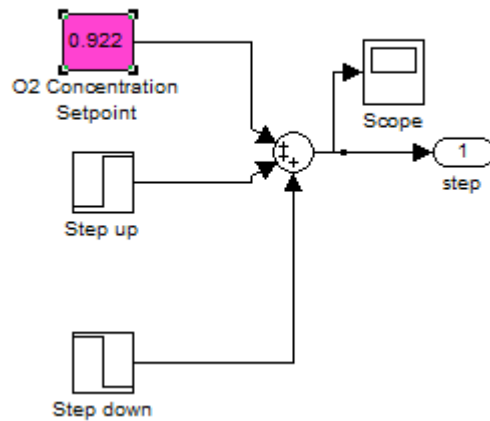


Figure 18 – Step change on O₂ concentration setpoint

- Simulate the response and plot the results of this step on both hydrocarbon exit temperature and O₂ concentration.

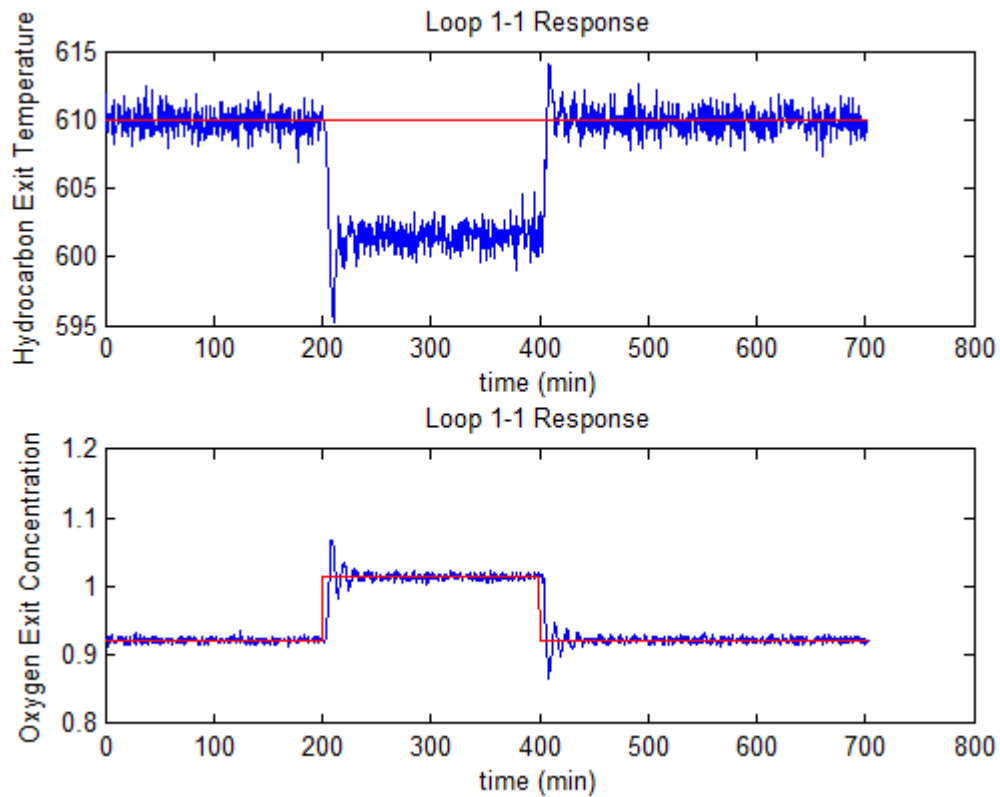


Figure 19 – Simulation results of a step change in O₂ concentration

- You can see that the oxygen loop is tracking the setpoint change.
5. **Simulate the Furnace response to a step change in Hydrocarbon temperature**
- Switch the oxygen loop to *Off*.

- Switch the temperature loop to *On*.
- Introduce a step change in the Hydrocarbon temperature setpoint: 9.9 at 20 min and back to the nominal after 200 min.
- Simulate the response and plot the results of this step on both Hydrocarbon exit temperature and Oxygen concentration.

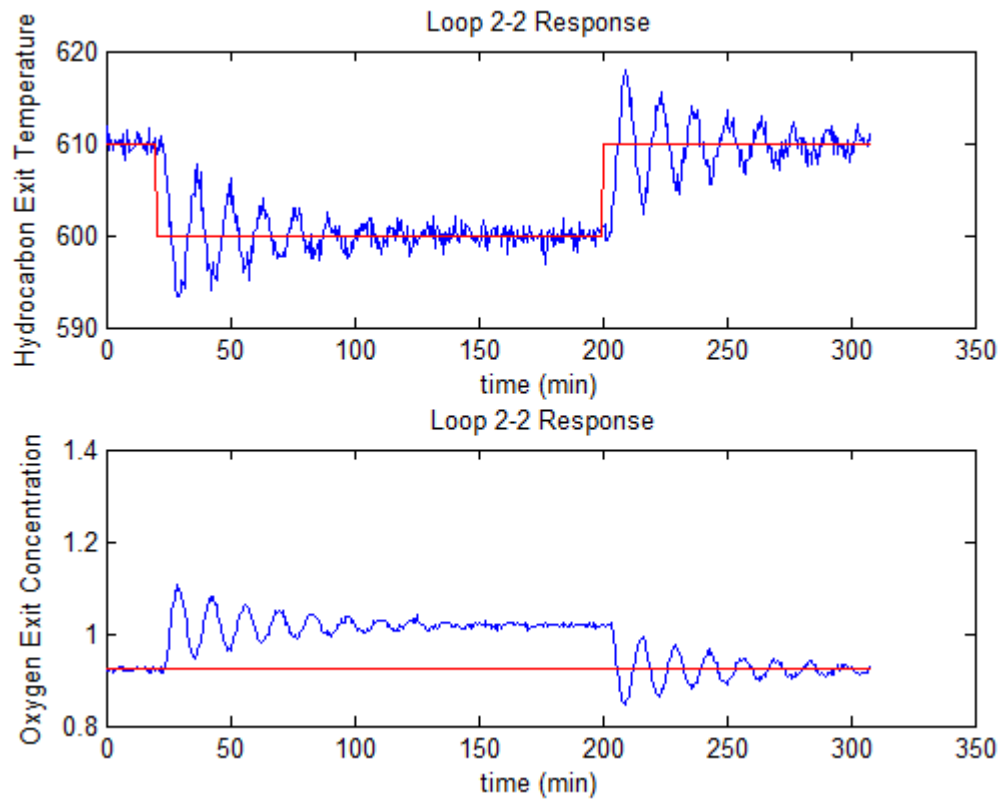


Figure 20 – Simulation results of a step change in Hydrocarbon temperature

- You can see that the Temperature loop is tracking the setpoint change
6. **Now turn on the Oxygen loop and simulate the Furnace response to a step change in Hydrocarbon temperature**
- Did you succeed in controlling both loops?
 - Detune the controllers and repeat the simulation.

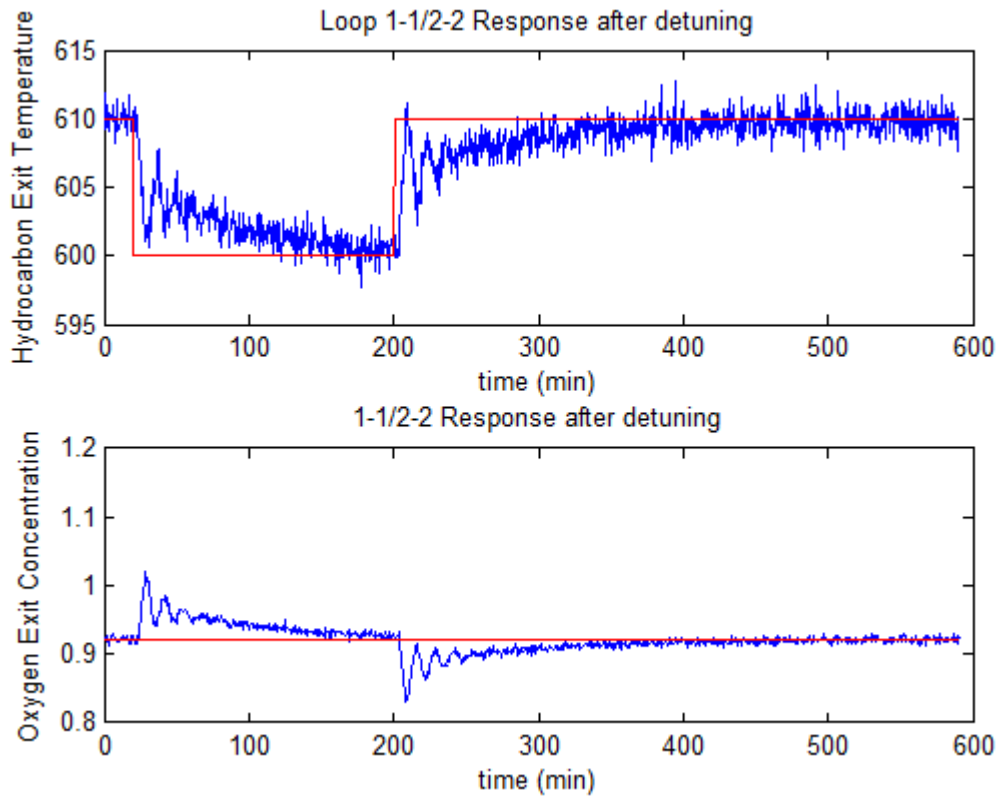


Figure 21 – Simulation results of a step change in Hydrocarbon temperature with both loops under control

- You can see that now both loop are tracking the setpoint change

Distillation Column

Introduction

The process under consideration in this module is a binary distillation column that 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 module 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 (Weischedel and McAvoy 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 22. 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

Reflux Ratio
Vapor Flow Rate
Feed MeOH Composition
Feed Flow Rate

Outputs

Overhead MeOH Composition
Overhead Flow rate
Bottom MeOH Composition
Bottom Flow Rate

The column has the following manipulated and controlled variables:

Manipulated Variables

Reflux Ratio
Vapor Flow

Controlled Variables

Overhead MeOH Composition
Bottom MeOH Composition

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

Load Variables

Feed Flow Rate
Feed MeOH Composition

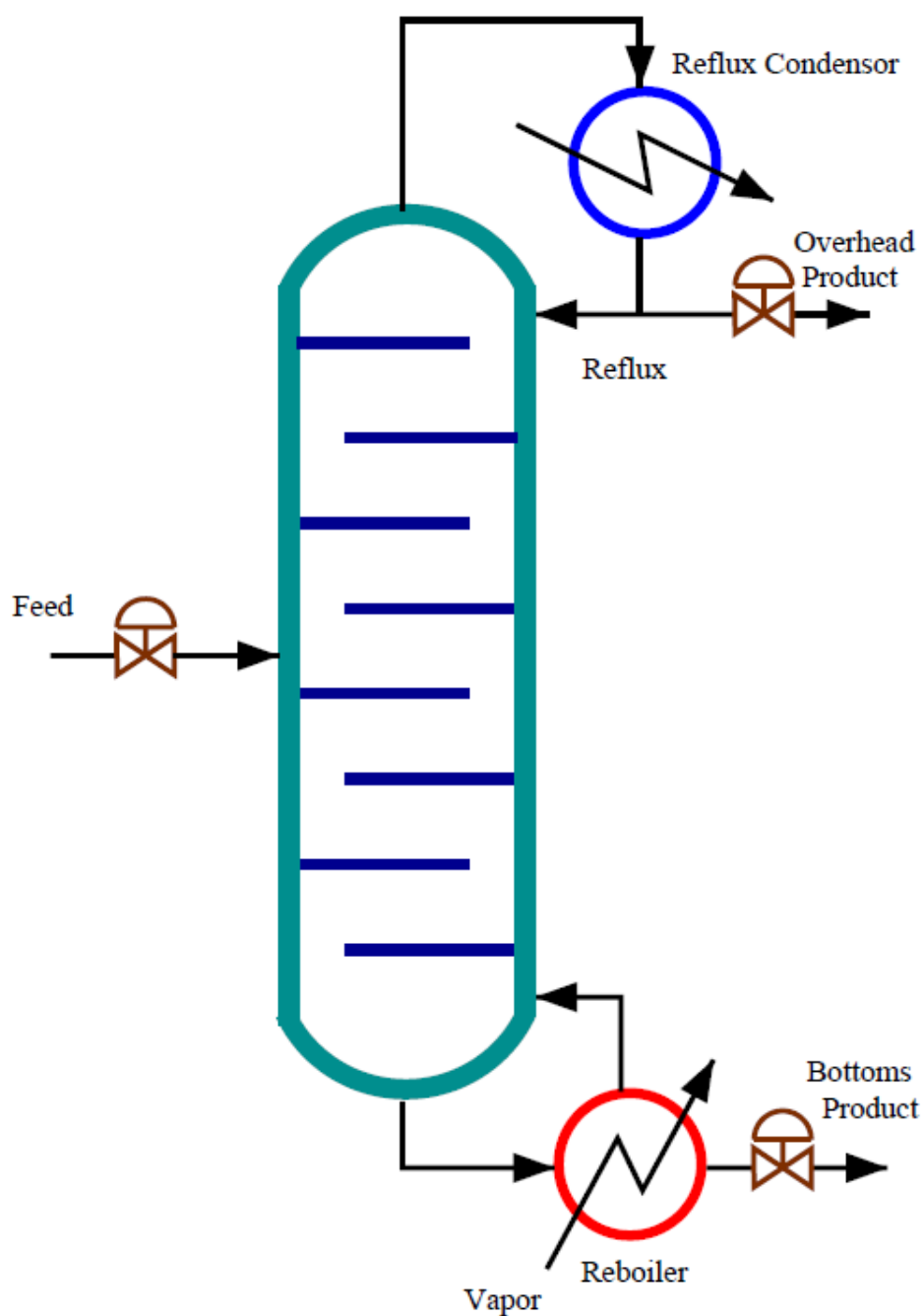


Figure 22 – Process schematic of the column

The column simulation can be executed by selecting *Distillation Column* from the “Modules” menu. Six different control simulations can be performed, each by clicking on one of the push buttons as appears in Figure 32.

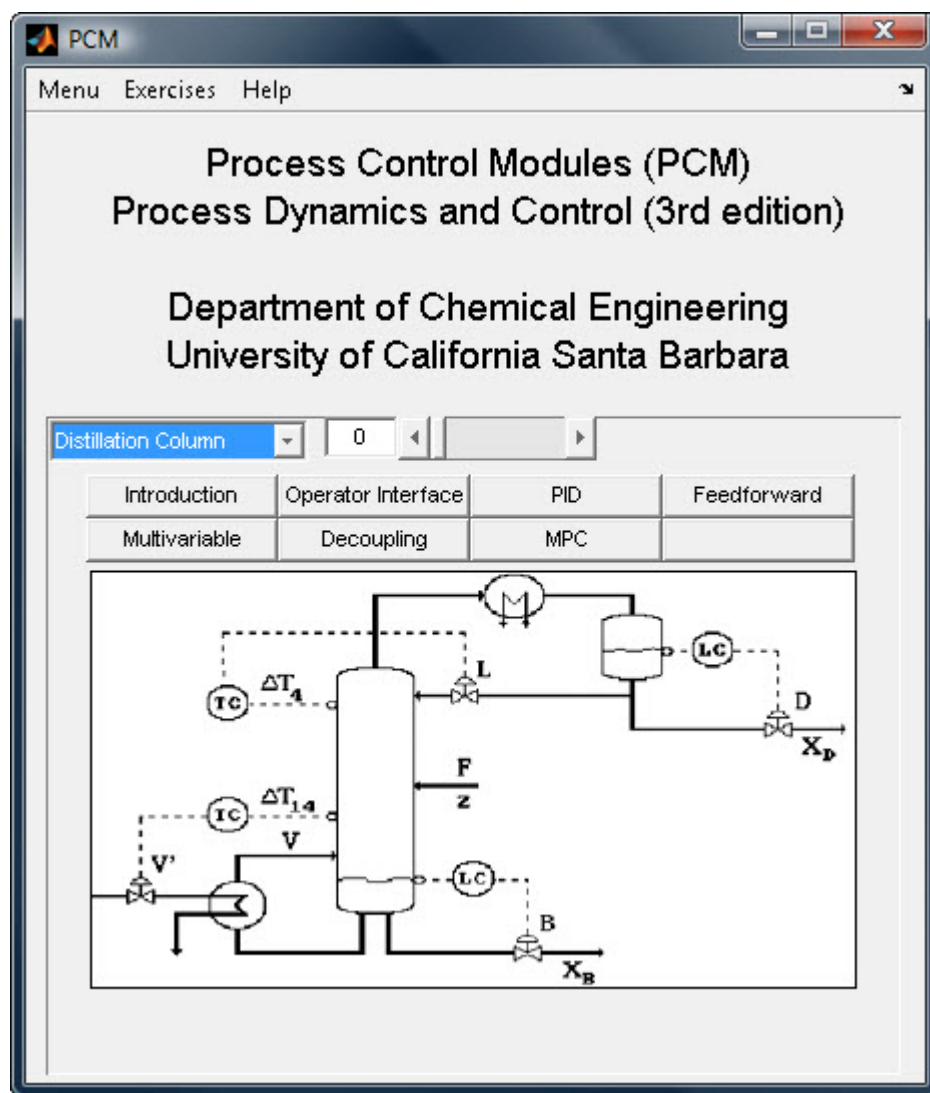


Figure 23 – Column interface

Column modules

Operator Interface

The Operator interface can be used for manual operation and control of the column, as well as process modeling. You can manually adjust the inputs and monitor the outputs using the monitor (Figure 24). The module allows you to introduce different Simulink blocks, such as "step," to assist in model identification.

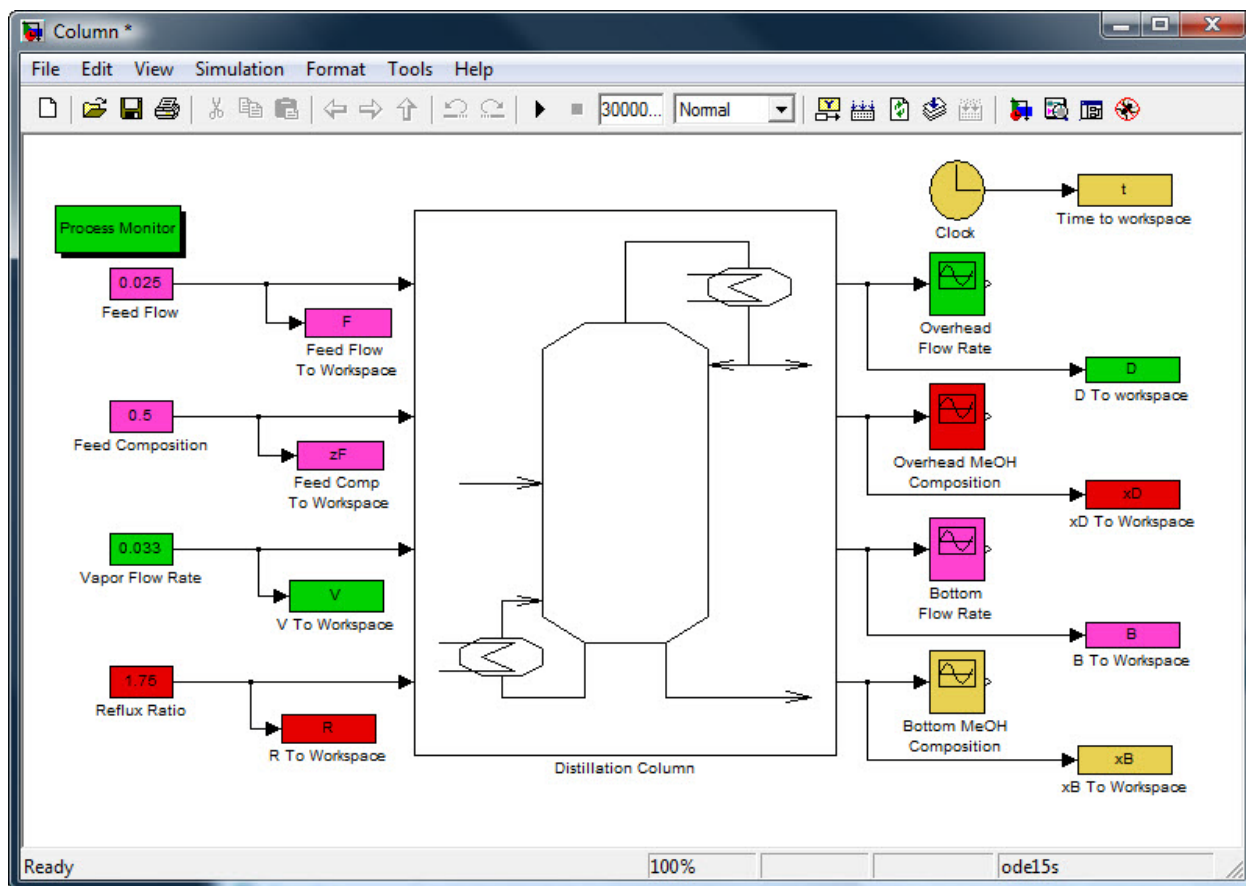


Figure 24 – Column operator interface

As can be seen from Figure 25, a step of 10% in Reflux Ratio was introduced at ~ 800 sec by pausing the simulation and manually adjusting the Reflux Ratio. This information is available to you by clicking on the *save* button in the process monitor (Figure 25) and selecting a name for the MATLAB .mat file.

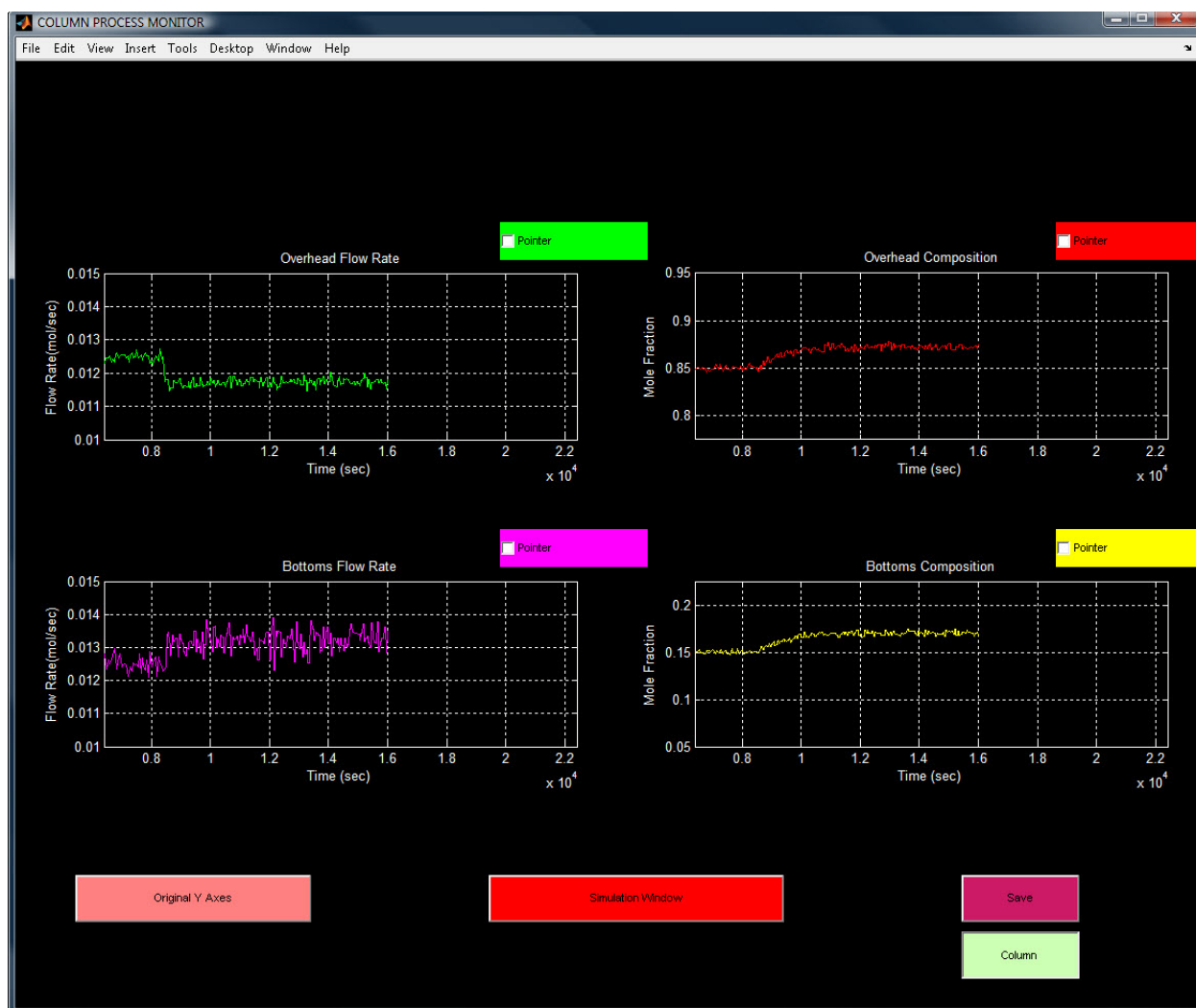


Figure 25 – Column monitor

Proportional-Integral-Derivative (PID) Control

In this module the characteristics of some of the various types of feedback control action and their influence on the performance of the Distillation Column will be studied. Of particular interest are the impacts of controller gain and reset time on the offset between the output and the setpoint at steady state. The Proportional-Integral-Derivative (PID) controller will be used in this module. A trial-and-error selection process for PID controller tuning constants requires a lengthy iterative procedure. In this module, you can experience different tuning algorithms that produce good initial estimates of controller gain (K_c), Integral reset time (τ_i), and derivative reset time (τ_D).

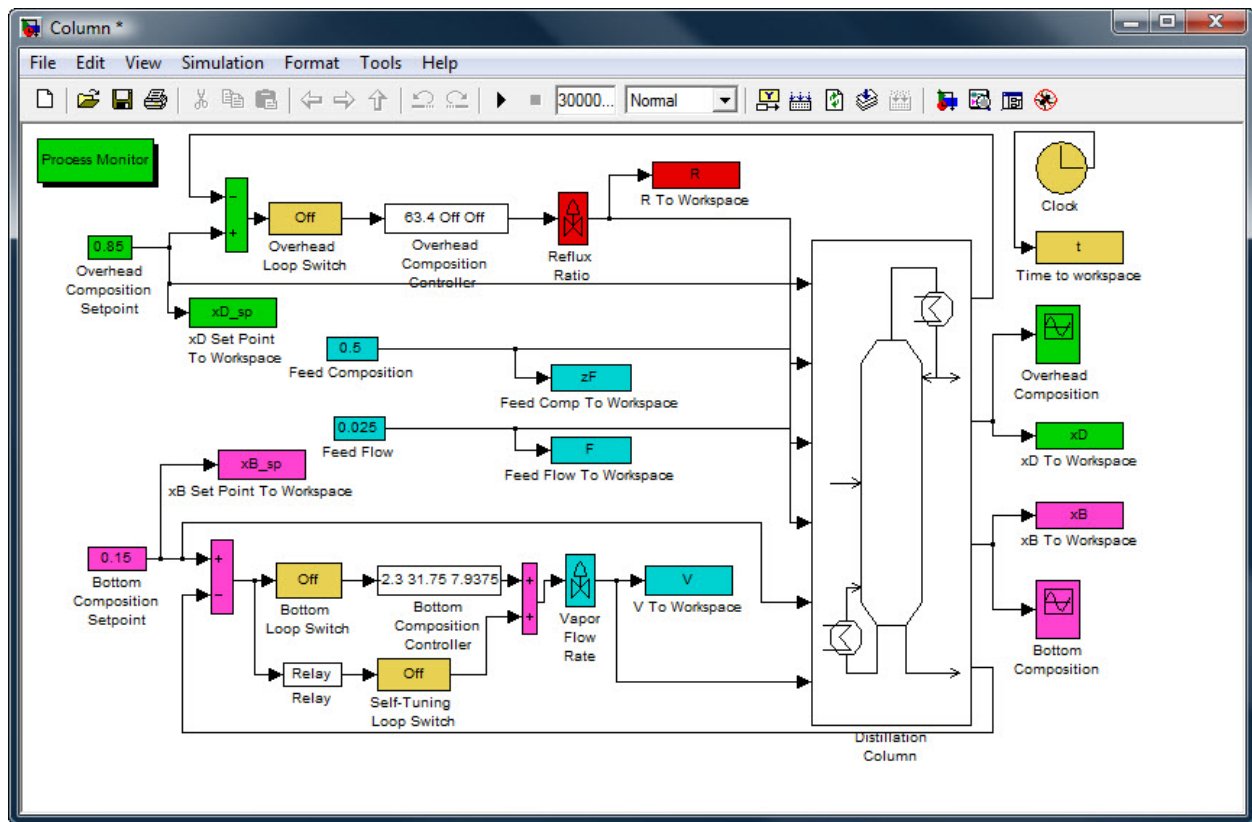


Figure 26 – Column PID interface

Feedforward Control

In the PID module, 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 taken only 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 can implement a feedforward strategy on the Distillation Column to reduce the effect of disturbances on process outputs.

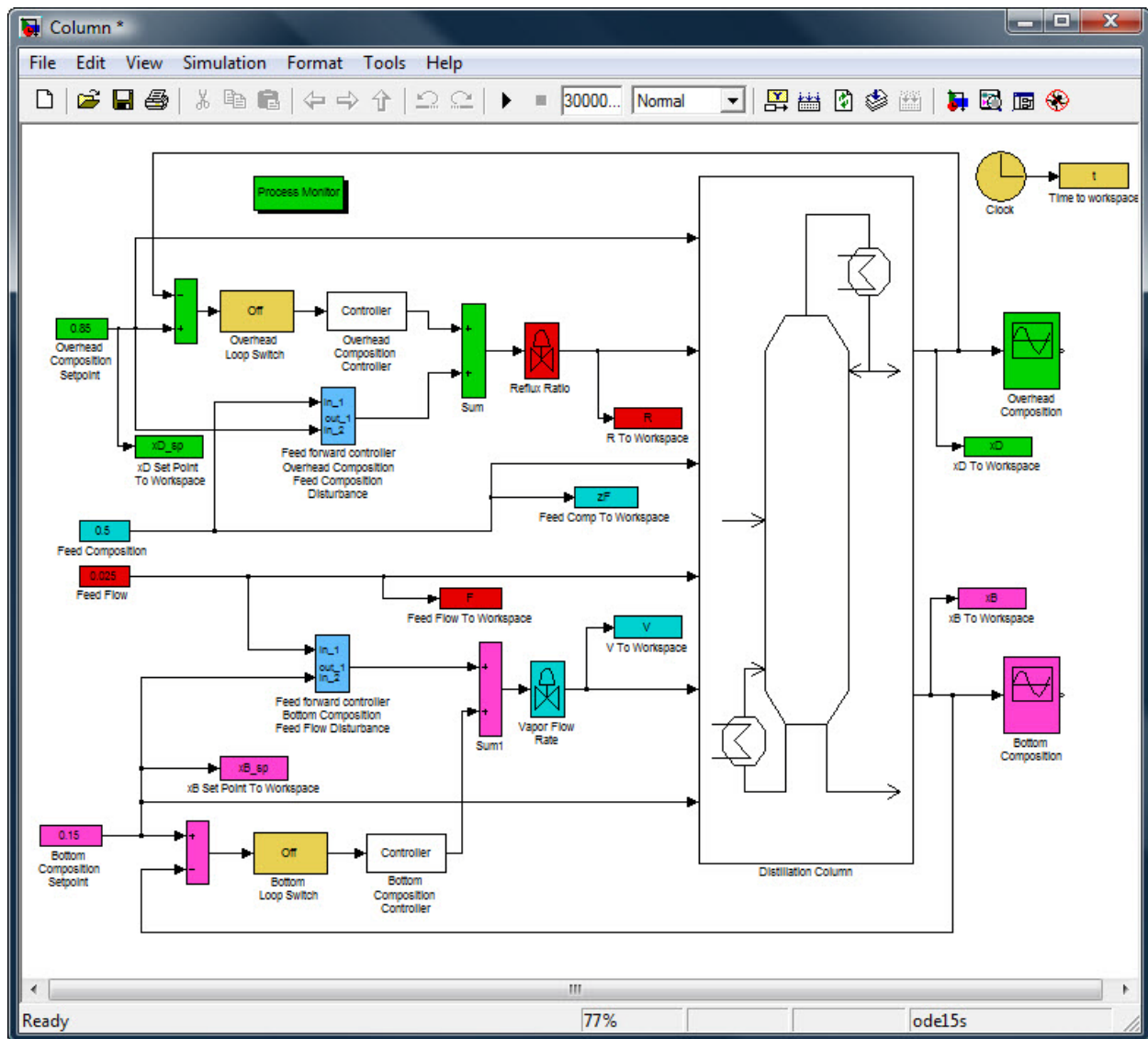


Figure 27 – Column feedforward interface

Multivariable Control

In this module we will address model-based control and multivariable control. The first-order-plus-time-delay models you validated previously will be used to design single-loop Internal Model Control (IMC) controllers.

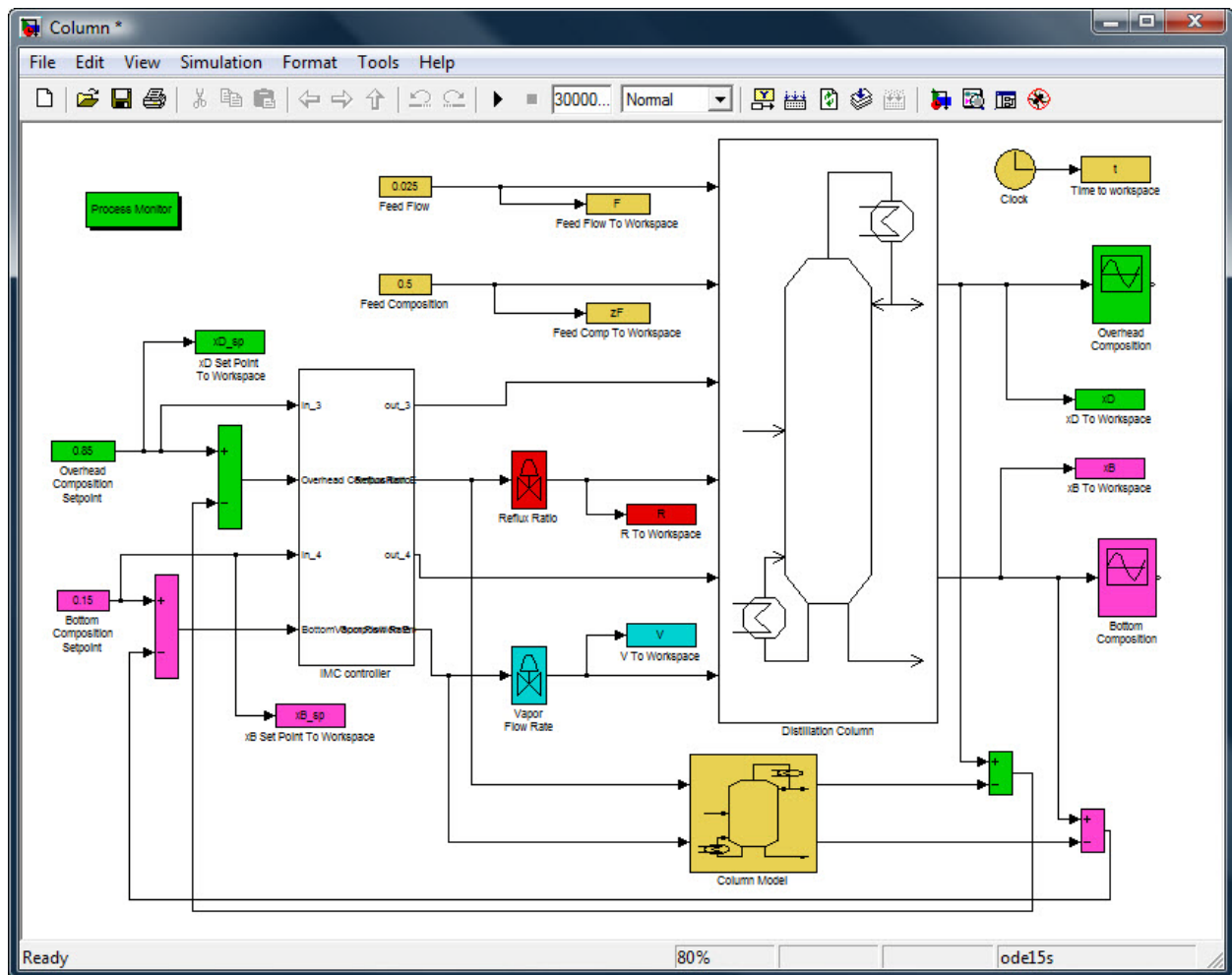


Figure 28 – Column multivariable interface

Decoupling

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. In this module we will use IMC controllers with decouplers that reduce interaction between the two control loops

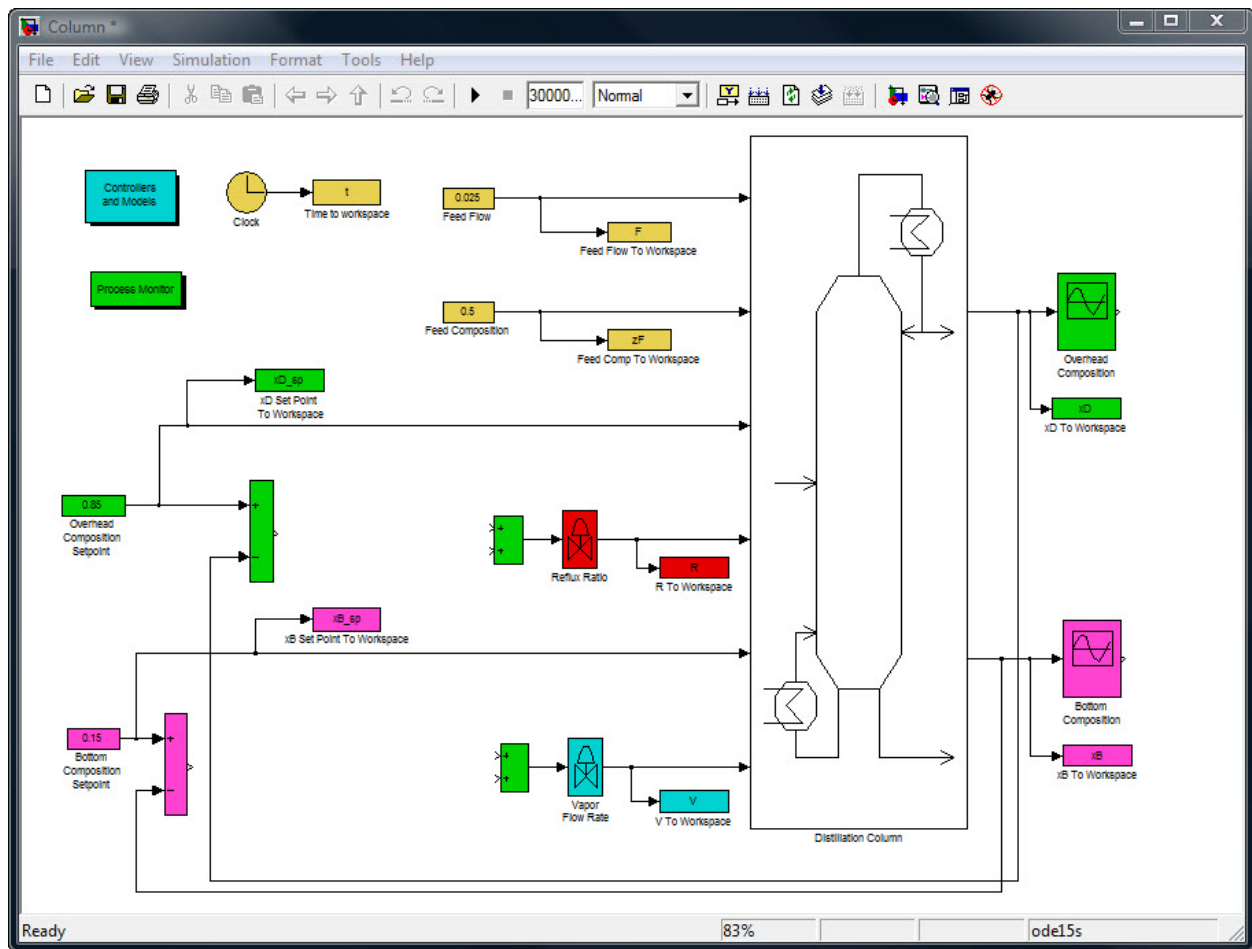


Figure 29 – Column decoupling interface

Model Predictive Control (MPC)

This module allows you to experience the basic concepts involved in designing a Model Predictive Controller (MPC). Discrete-time models of the Distillation 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.

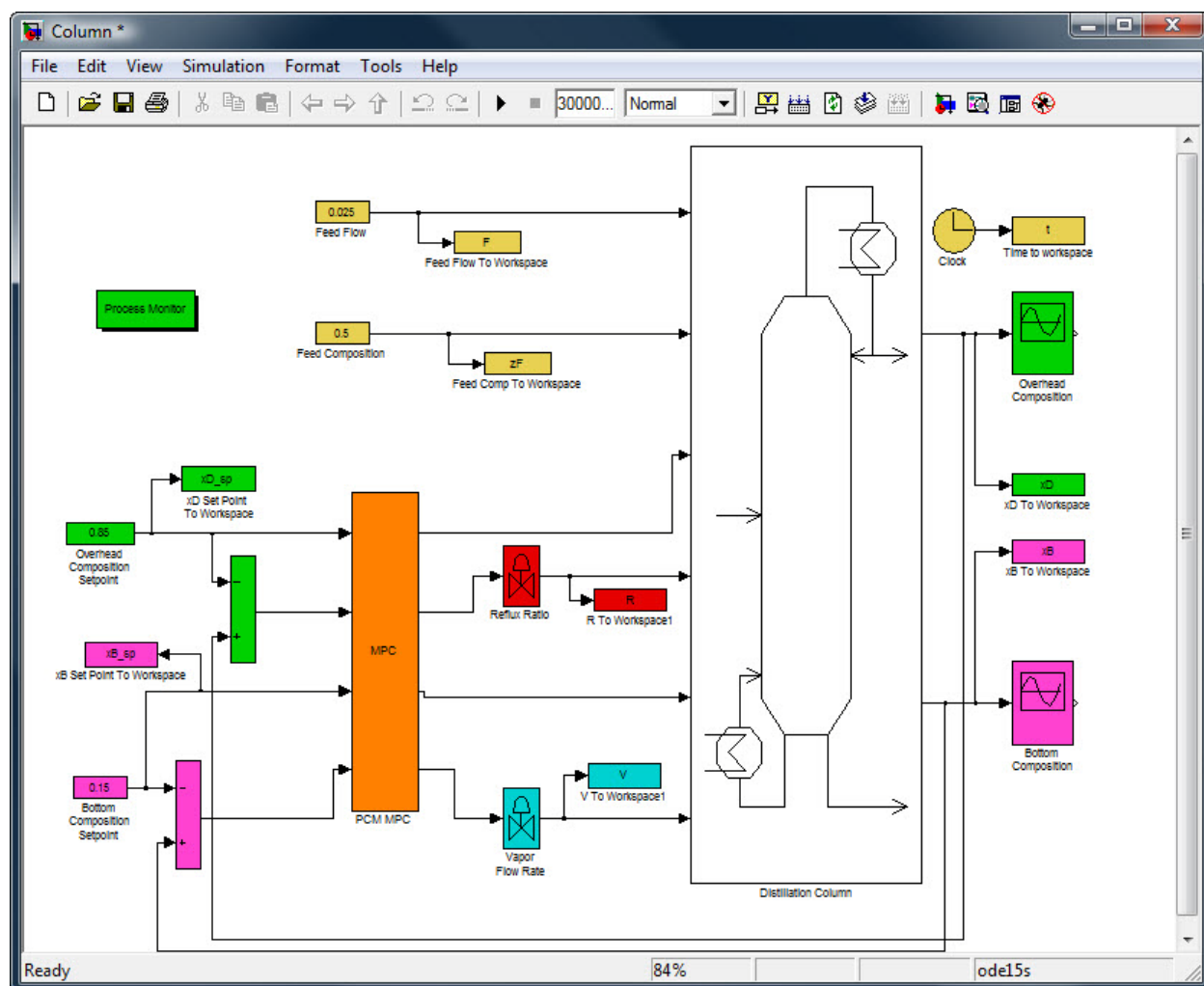


Figure 30 – Column MPC interface

Distillation Column Tutorial

Open-loop identification

1. Open and initialize the Distillation Column simulation

- Type 0.1 into the text box that initially has a value of 0 . The new value reduces the speed at which the real-time plots are plotted.
- Next, select “Distillation Column” in the drop down “Module” menu.
- Click the “Introduction” button and read the information about the “Binary Distillation Column” file.

2. Organize MATLAB windows

- Click on the “Operator Interface” button. The “column” and “COLUMN PROCESS MONITOR” windows will open, although one window may initially be hidden behind the other.
- In the “COLUMN PROCESS MONITOR” window, click the drop-down menu for the desktop and select “DOCK COLUMN PROCESS MONITOR”.
- Then resize the MATLAB and Column windows so that you can see both windows clearly at the same time.

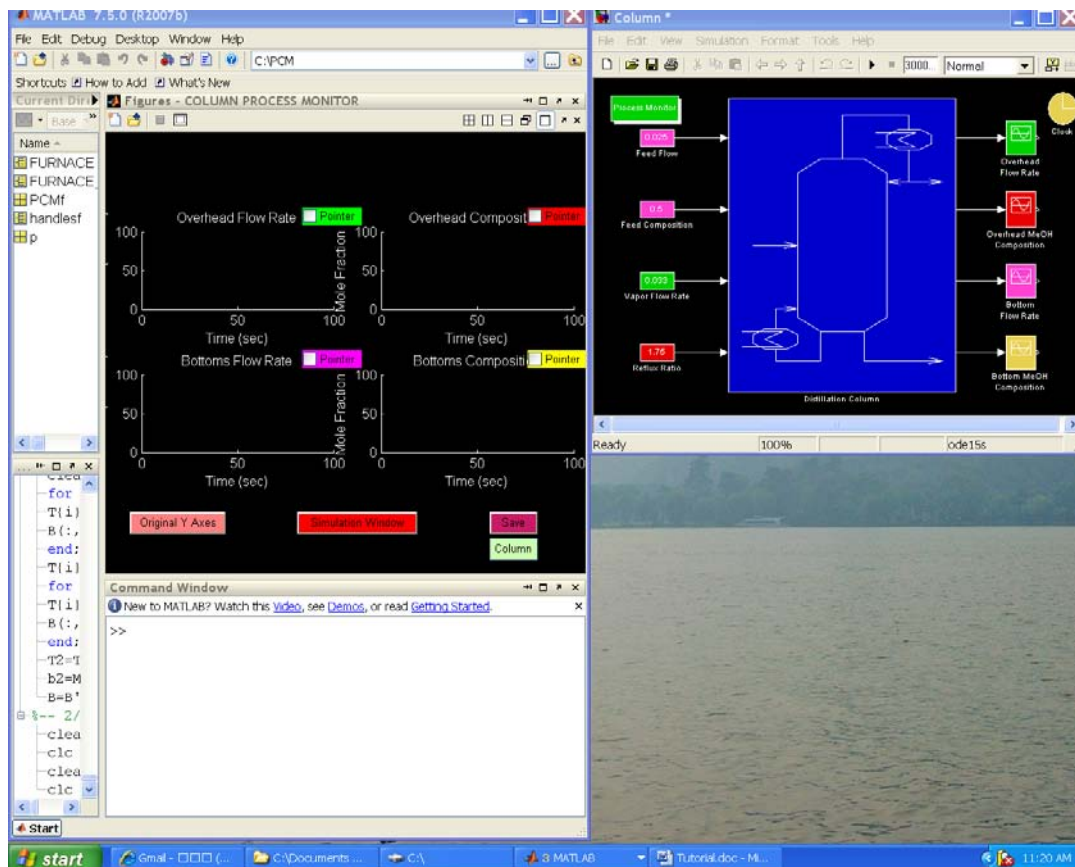


Figure 31 – Screenshot of MATLAB and Column windows after resizing

3. Simulate the column response to three step changes in the overhead composition.

- Press the “Play” button (i.e., the black triangular icon in the top row) in the “Column” window to start the simulation. Let the simulation proceed for approximately 3000 seconds. In the simulations, seconds are not real-time units but are simulated units clocked in the “COLUMN PROCESS MONITOR” window. Click the “Pause” button (marked by a symbol with two vertical lines.). Now you can read the exact time from the “Column” window (as shown in the following figure).

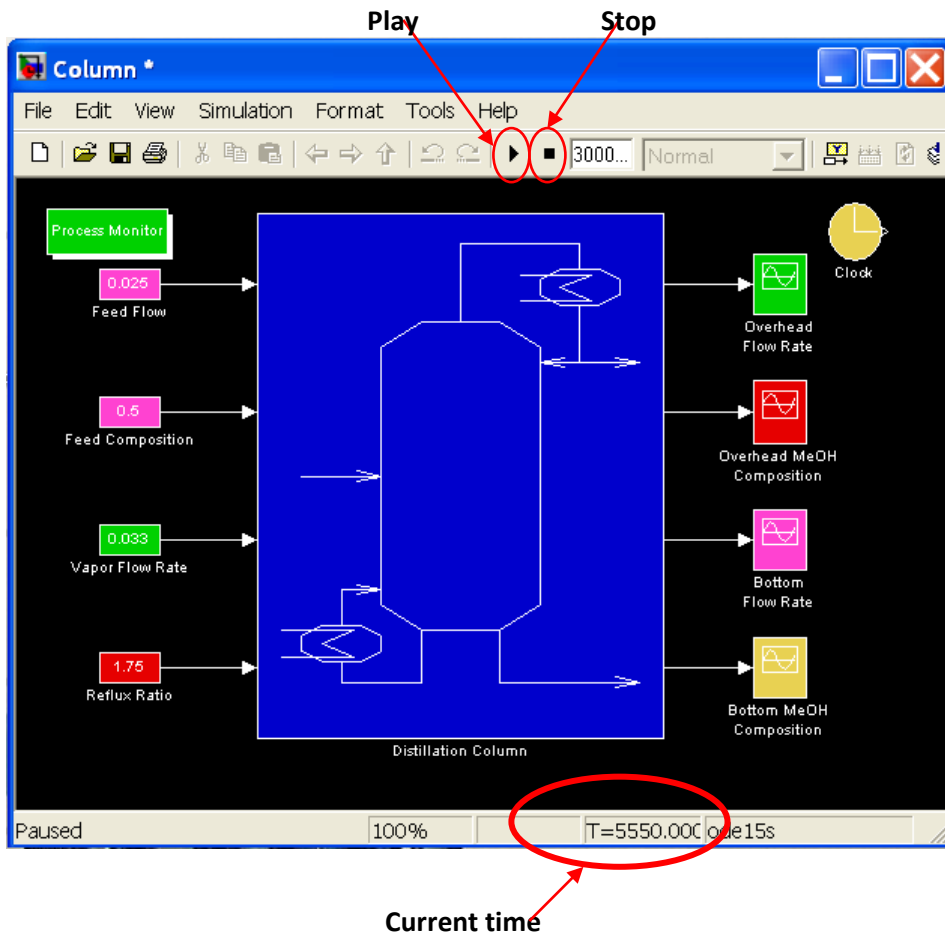


Figure 32 – Column window interface

- Next, double-click on the “reflux ratio” block in the “Column” window, and multiply the current entry in the “Constant Value” tab by a factor of 1.1.
- Begin the simulation again by pressing “Play”. Let the simulation run for approximately 3000 additional seconds, and then pause the simulation. You will be monitoring the bottom composition (output).
- Reset the reflux ratio to its original value. Run the simulation for approximately 3000 additional seconds, and then press the “Pause” button and change the reflux ratio to

another value (e.g, 1.75×1.2 , 1.75×1.3). Then run the simulation for 3000 seconds, as clocked in the “COLUMN PROCESS MONITOR” window.

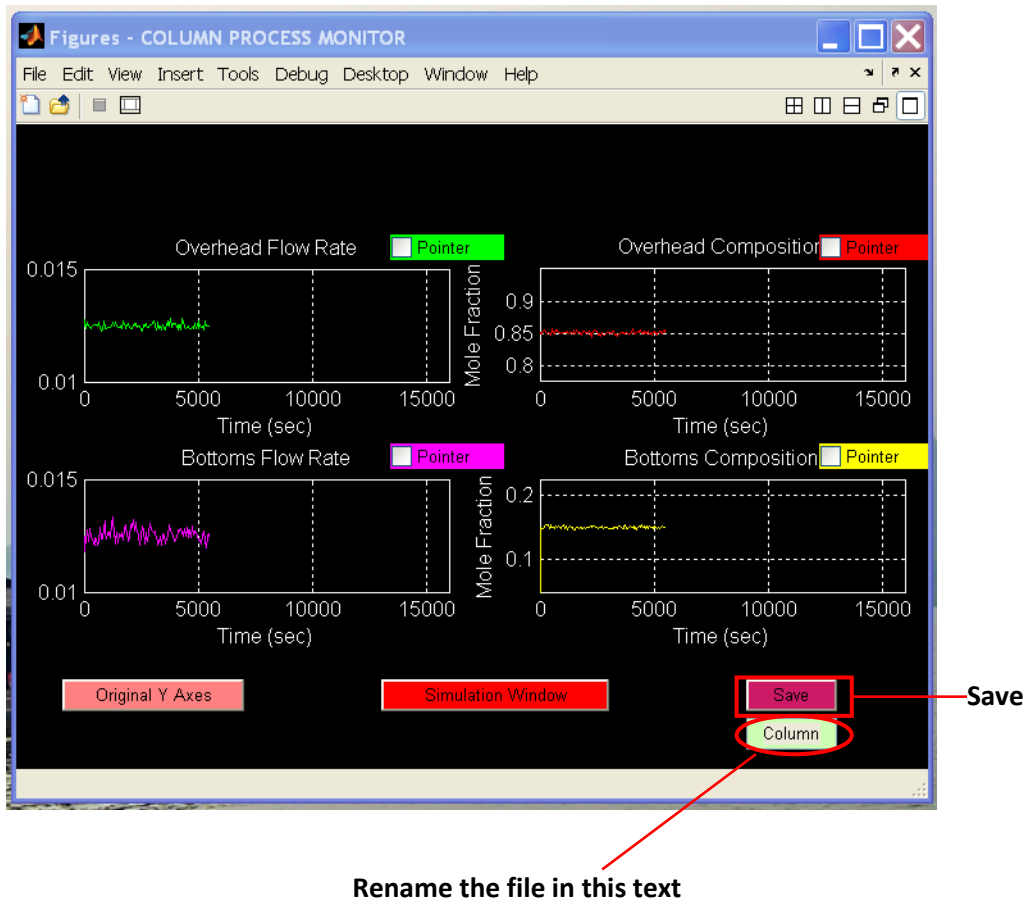


Figure 33 – Screenshot of the Column process monitor with simulation results

- After you have finished three step response for the reflux ratio, press “Stop” in the “Column” window (as shown in Figure 32).
- Set the name of data file (i.e., reflux_ratio) in the “COLUMN PROCESS MONITOR” window (as shown in Figure 33).
- Press “Save”. A data file named “reflux_ratio.mat” is generated in current dictionary.
- Then do the similar work to get three step responses for the vapor flow rate input and bottom composition (output). Multiply the original value by 1.05, 1.1 and 1.15 in the three step responses.

Useful tips:

- Please do not forget to read the time when you press “Pause”; you may need a value for your simulation.
- A run time of 3000 seconds is sufficient for this simulation. If you run any step of the simulation too long, you may lose data when you save it to the local file.
- It is much better to run a new simulation for the step change of vapor flow rate, in order to make sure that all the data is successfully saved. Before you run a new simulation, first press “Stop” in the “Column” window and close both windows, and do not save the changes. Then type “clear all” in the command window in MATLAB, restart a new simulation, and save the useful data.

4. Stop the simulation to acquire data in Excel.

- Now you have your own data file (i.e., reflux_ratio.mat) in the current directory. Double-click on the file name in the “Current dictionary” window and all the data will be released to the “workspace” window.
- Open the “workspace” window and select the useful data.
- The time variables ‘d1t’, ‘d2t’, ‘d3t’, and ‘d4t’ are the same and represent the time in the simulation with unit “second”.

Output variables name	Output variables description	Units
d1y	Overhead flow rate	(mol/sec)
d2y	Overhead composition	(fraction)
d3y	Bottoms flow rate	(mol/sec)
d4y	Bottoms composition	(mol/sec)

You can copy these data to Excel and then analyze them.

If these data are row vectors, which are not so convenient to copy, you can type the following script into the command window of MATLAB to change them into column vectors.

```
d1t=d1t' ;
```

```
d1y=d1y' ;
```

```
d2y=d2y' ;
```

```
d3y=d3y' ;
```

```
d4y=d4y' ;
```

Closed-loop control - PID

1. Open and initialize the Distillation Column simulation

- Type *0.1* into the text box that initially has a value of *0*. The new value reduces the speed at which the real-time plots are plotted. Note, to slowdown the real-time plots used a higher number, a value of *.1* should be sufficient for most systems.
- Next, select “Distillation Column” in the drop down “Module” menu.

2. Organize MATLAB windows

- Click on the “PID” button. The “Column” and “DISTILLATION COLUMN WITH CONTROL PROCESS MONITOR” windows will open, although one window may initially be hidden behind the other.
- In the “DISTILLATION COLUMN WITH CONTROL PROCESS MONITOR” window, click the drop down menu for the Desktop and select “DOCK DISTILLATION COLUMN WITH CONTROL PROCESS MONITOR”.
- Then, resize the MATLAB and Furnace windows so that you can see both windows clearly at the same time.

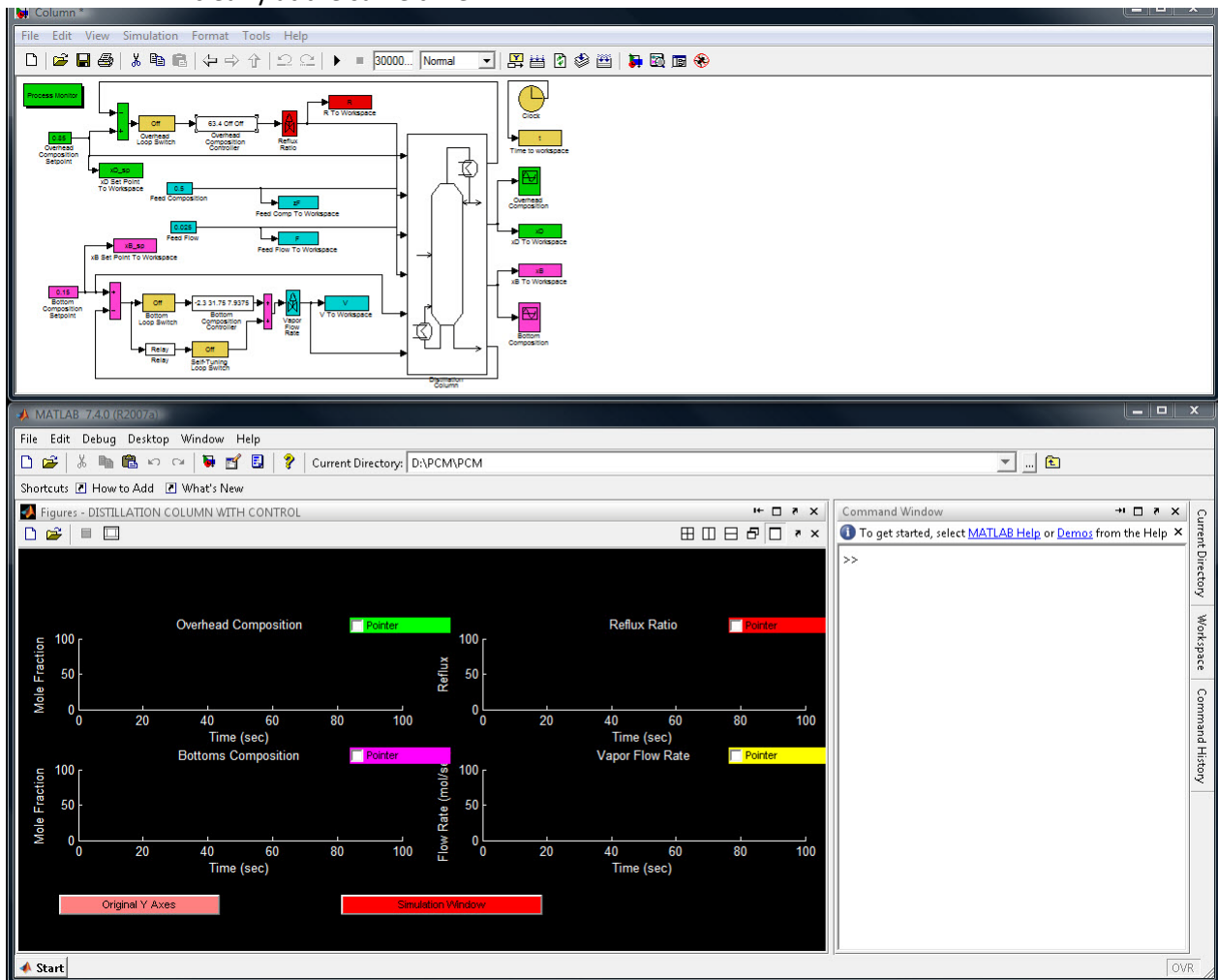


Figure 34 – Screenshot of the closed-loop control using PID

3. Controller settings

- Calculate the PID control settings, based on IMC tuning rules, for both control loops.
- Enter these settings into the PID Controllers and change the Derivative Action to 1.
- Note, you can define your controller settings as variables. This will allow you to modify them from the Matlab workspace.

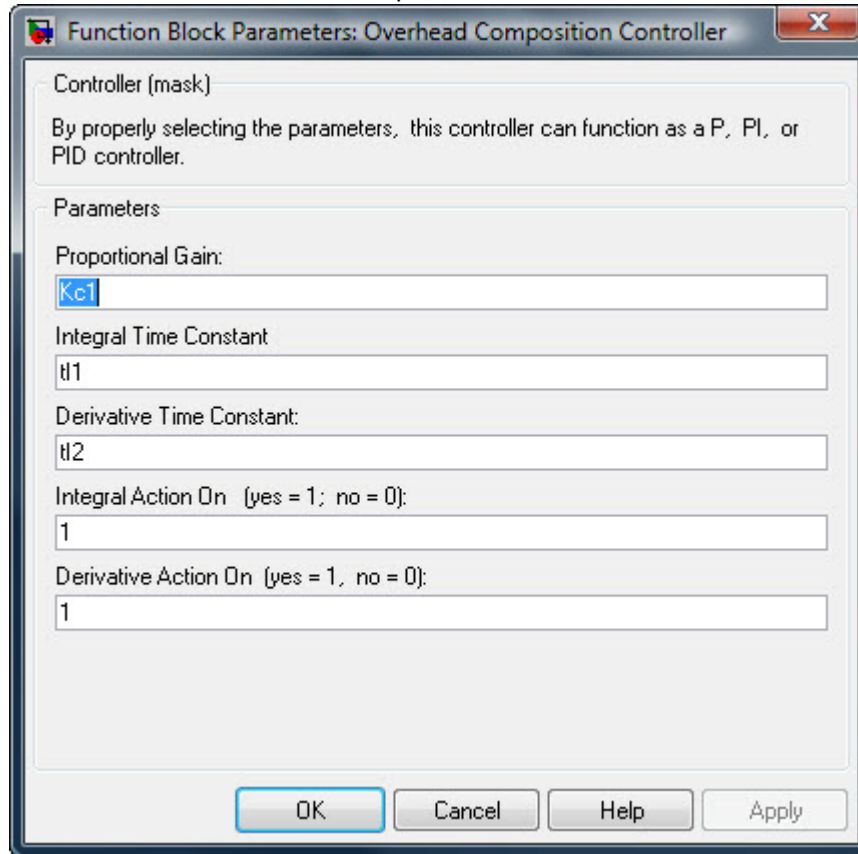


Figure 35 – PID tuning interface

4. Simulate the Column response to a step changes Overhead composition concentration

- Switch the Overhead composition loop to *On* by clicking the toggle switch.

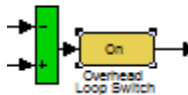


Figure 36 – Overhead loop switch

- Replace both compositions setpoint constants with workspace input variable step1 and step2 respectively

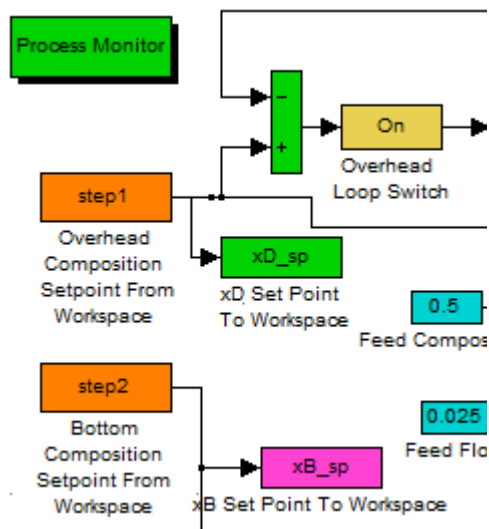


Figure 37 – Setpoint of Overhead and Bottom composition defined from workspace

- Define in Matlab command window the following step change to the Overhead and the Bottom composition setpoint respectively: `step1=[0 .85; 500 .85; 500 .8; 5000 .8; 5000 .9; 10000 .9; 10000 .85;];` `step2=0.15;`
- Simulate the response and plot the results of this step.

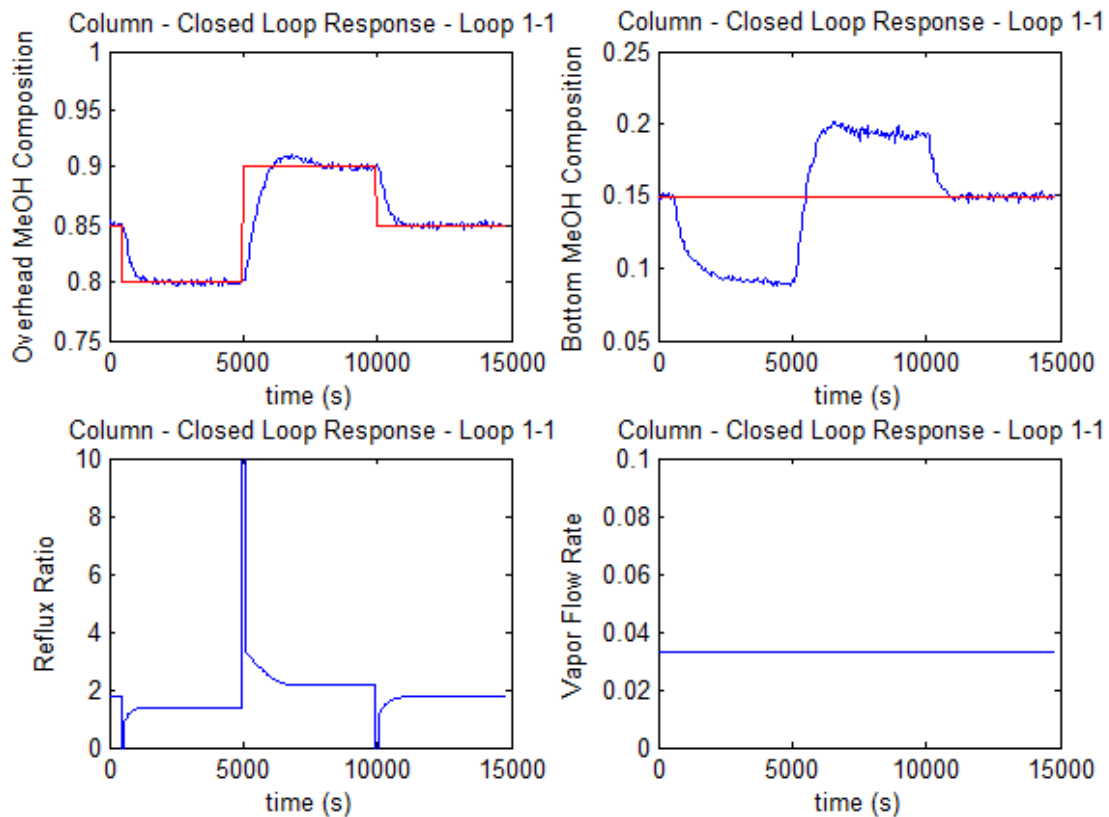


Figure 38 – Simulation results of a step change to Overhead composition setpoint

- You can see that the Overhead composition loop is tracking the setpoint change.
- 5. Simulate the Column response to a step change in Bottom composition concentration**
- Switch the Overhead loop to *Off*.
 - Switch the Bottom loop to *On*.
 - Define in Matlab command window the following step change to the Bottom and the Overhead composition setpoint respectively: `step2=[0 .15; 500 .15; 500 .1; 5000 .1; 5000 .2; 10000 .2; 10000 .15;];` `step1=0.85;`
 - Simulate the response and plot the results of this step.

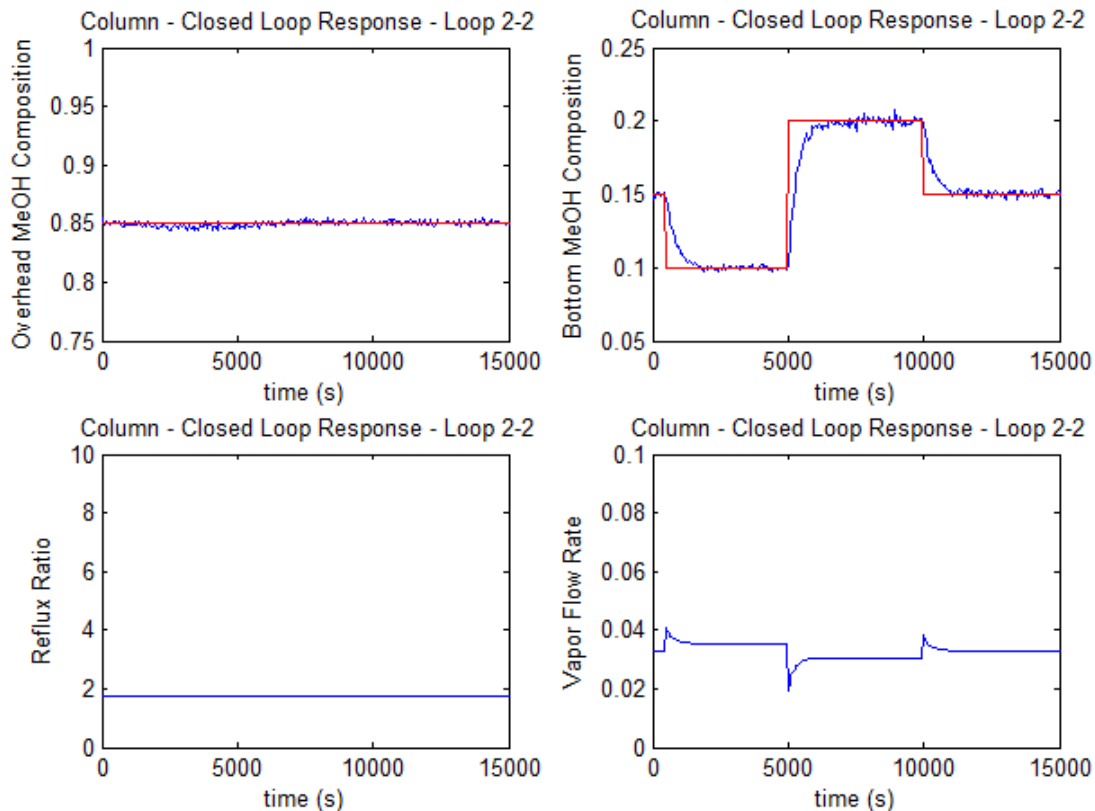


Figure 39 – Simulation results of a step change to Bottom composition setpoint

- You can see that the Bottom loop is tracking the setpoint change
- 6. Now turn on the Overhead loop and simulate the Column response to a step change in both loops**
- Switch the Overhead loop to *On*.
 - Switch the Bottom loop to *On*.
 - Define in Matlab command window the following step change to the Overhead and the Bottom composition setpoint respectively:
 - `step1=[0 .85; 500 .85; 500 .8; 5000 .8; 5000 .9; 10000 .9; 10000 .85;];`
 - `step2=[0 .15; 500 .15; 500 .1; 5000 .1; 5000 .2; 10000 .2; 10000 .15;];`
 - Simulate the response and plot the results of this step.

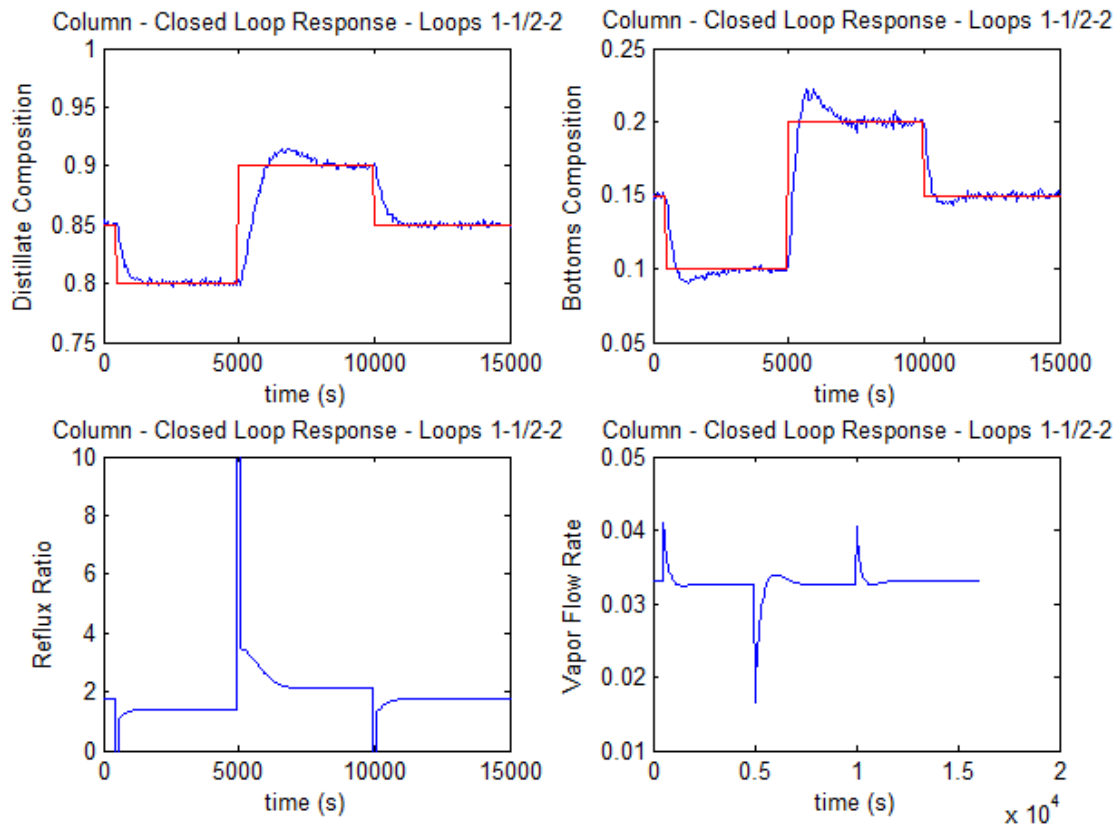


Figure 40 – Simulation results of a step change in both Overhead and Bottom composition setpoint

You can see that now both loops are tracking the setpoint change

References

- Doyle III, F. J., E. P. Gatzke and R. S. Parker (1998). "Practical case studies for undergraduate process dynamics and control using process control modules." *Computer Applications in Engineering Education* 6(3): 181-191.
- Doyle III, F. J., E. P. Gatzke and R. S. Parker (2001). *Process Control Modules: A Software Laboratory for Control Design*. New Jersey, Prentice Hall PTR.
- Weischedel, K. and T. J. McAvoy (1980). "Feasibility of Decoupling in Conventionally Controlled Distillation Columns." *Industrial & Engineering Chemistry Fundamentals* 19(4): 379-384.