Improved Polymer Process Operability and Control through Steady-State and Dynamic Simulation Models

Y.A. Liu and Niket Sharma

Department of Chemical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, U.S.A

Abstract

This chapter introduces a novel approach to dynamic polymer process modeling using Aspen Plus and Aspen Plus Dynamics (AD). By transitioning from steady-state to dynamic simulation, we present a powerful tool for analyzing and optimizing polymer production processes, offering deeper insights into process operability and control.

We outline a comprehensive workflow for converting steady-state models into dynamic simulations, enabling users to simulate time-dependent behaviors in polymer processes. The chapter explores various aspects of dynamic simulation, including simulation modes, real-time data visualization, and advanced process control strategies. We also integrate key tasks such as PID controller configuration, snapshot management, and discrete event sequencing to enhance the management of complex polymer production scenarios, particularly grade transitions.

Through practical workshops and case studies, we demonstrate the application of these dynamic simulation techniques to industrial polymer processes, focusing on high-density polyethylene (HDPE) and linear low-density polyethylene (LLDPE) production. The methodologies and insights presented aim to equip engineers and researchers with effective tools for implementing dynamic simulations, ultimately improving process efficiency and control in polymer manufacturing.

This is a preprint version of a chapter from our book - *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing [22,31]. Please cite the original work if referenced*

7.1 WS 7.1 Workflow for Dynamic Process Modeling Using Aspen Plus and Aspen Plus Dynamics

Section 7.1 demonstrates a four-step workflow for dynamic process modeling using Aspen Plus and Aspen Plus Dynamics. For convenience, *we use the term Aspen Dynamics, or abbreviation AD, for Aspen Plus Dynamics throughout this chapter.* Section 7.2 introduces how to run dynamic simulation in AD. We cover types of dynamic simulations (flow-driven and pressure-driven), graphical interface of AD, simulation run modes and run control, viewing simulation results using predefined tables and plots, specification status and analysis, creating new tables and plots, and variable finding in AD. Section 7.3 discusses process control in AD, including adding and configuring a PID controller. Section 7.4 explains snapshot management and taking a snapshot of simulation inputs and results. Section 7.5 introduces tasks for defining a sequence of discrete events, which are useful to simulating grade changes in polyolefin production. Section 7.6 presents a workshop, dynamic simulation and grade change of a slurry high-density polyethylene (HDPE) process and demonstrates the use of tasks in simulating grade

changes. Section 7.7 presents another workshop to demonstrate how to implement various control schemes in a commercial slurry HDPE process. Section 7.8 demonstrates the controller design for a gas-phase fluidized-bed LLDPE process under a condensed mode operation previously presented in Section 5.8 and introduces the concept of a split-range controller. Section 7.9 presents the inferential controller of a slurry HDPE process. Section 7.10 is the bibliography.

There are several published studies of dynamic simulation and control of polyolefin processes using Aspen Plus and Aspen Plus Dynamics [1 to 5]. However, none of the published studies has presented sufficient details to enable the readers to apply software tools for this application. An objective of this chapter is to present the workflow and details of dynamic polymer process simulation with illustrative workshops to help our readers.

We begin with an illustrative example to demonstrate the following steps in our workflow:

1. Create a steady-state simulation flowsheet in Aspen Plus (which includes Aspen Polymers), resulting in **a steady-state simulation file, *.bkp**);
2. Enter the data required to calculate the dynamics of the process: particularly the vessel type and geometry required for vessel volume; vessel initial fill percentage required for starting vessel holdup; process heat-transfer option, equipment heat-transfer option, and environmental heat-transfer method for heat-transfer calculations. Run the steady-state simulation and save the results.
3. Export the simulation to Aspen Plus Dynamics, resulting in **a dynamic simulation file, *.dynf**, and **an Aspen property data file, *.appdf**,
4. Open the dynamic simulation file in AD, apply process disturbances, change the default level, pressure, and temperature controllers, add new controllers, and add tables or figures for displaying simulation results before running the dynamic simulation.

For step 1, we open a steady-state simulation file, ***WS7.1.bkp***. Figure 7.1 shows the flowsheet, and Table 7.1 specifies the inputs.

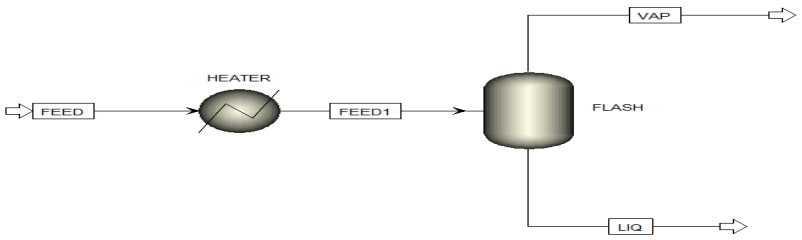


Figure 7.1 Flowsheet for WS7.1

Table 7.1 Specifications of the steady-state simulation, WS7.1

Components	N2, NC6 (n-hexane) and water
Property method	NRTL thermodynamic method with Henry component, N2
Stream	Feed: 20°C, 1.1 bar, mole flow (kmol/hr): N2 - 2, NC6 - 20, and water - 70
Blocks	Heater: 0 bar (no pressure drop), output vapor fraction = 0.4 Flash: 1 bar, 0 MMkcal/hr (adiabatic)

For step 2, we click the Dynamics tab on the ribbon, and click on the Dynamic Mode button to activate the dynamic input form. See Figure 7.2 for inputs for the heater, which include a *dynamic* heater type, instead of an instantaneous type and heater volume specifications.

For heat-transfer option, we choose the default, *constant duty*. Other options include: (1) *constant medium temperature* with heat duty depending on the temperature difference between process fluid and the heating/cooling medium; and (2) *LMTD (log-mean temperature difference)* between the process fluid and the heating/cooling medium.

We do not consider *equipment heat transfer* in this example. When there are large changes in the equipment temperature (including startup, shutdown, or pressure relief), equipment heat transfer becomes important, and we need to enter *the mass and heat capacity* of the equipment, *the overall heat-transfer coefficient*, and *the ambient temperature* in order to model equipment heat capacity and heat transfer with the environment.

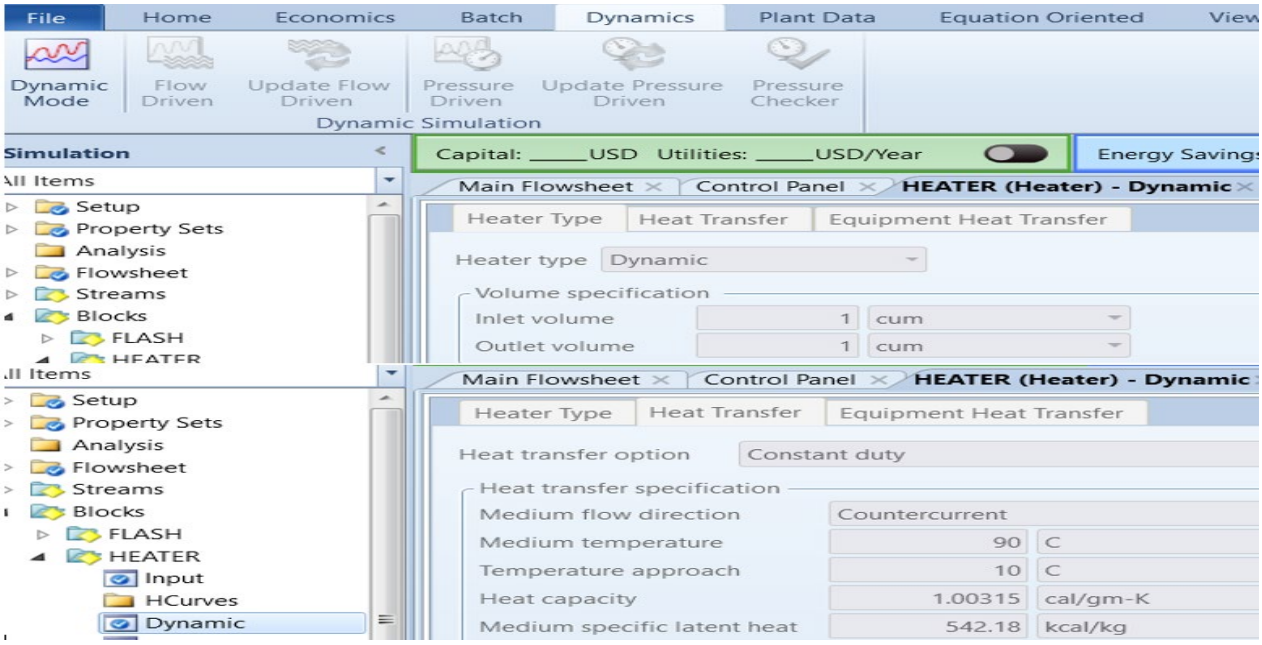


Figure 7.2 Dynamic data inputs for the heater.

The flash unit is a vertical vessel with an elliptical head, a length of 4.5 m and a diameter of 1.5 m. We assume *a constant duty option* for heat transfer and specify *a liquid volume fraction of 0.3* for the initial conditions. We do not consider equipment heat transfer and vessel wall heat transfer. A search of the AD online help, "Vessel Geometry", gives pictures of different vessel orientations and head types, and show how volume and height are calculated and how liquid level is determined.

In converting a steady-state simulation model to a dynamic simulation model, AD automatically adds default *controllers* according to the following rules: (1) If there is a liquid in the vessel, add a level controller (LC); (2) If there is a vapor in the vessel, add a pressure controller (PC); and (3) If the vessel is a reactor that uses kinetics, add a temperature controller (TC). Following rules (1) and (2), we add default

controllers for level and pressure. We then run the steady-state simulation, and save the resulting file as ***WS7.1.bkp*** within our AD working folder.

For step 3, we click the Dynamic tab on the ribbon, and click the Flow Driven button to export the simulation from Aspen Plus to AD, and save the dynamic simulation in the AD working folder as ***WS7.1.dynf***. This will automatically open the simulation flowsheet in AD with the default level and pressure controllers on top of the Aspen Plus simulation window. See Figure 7.3.

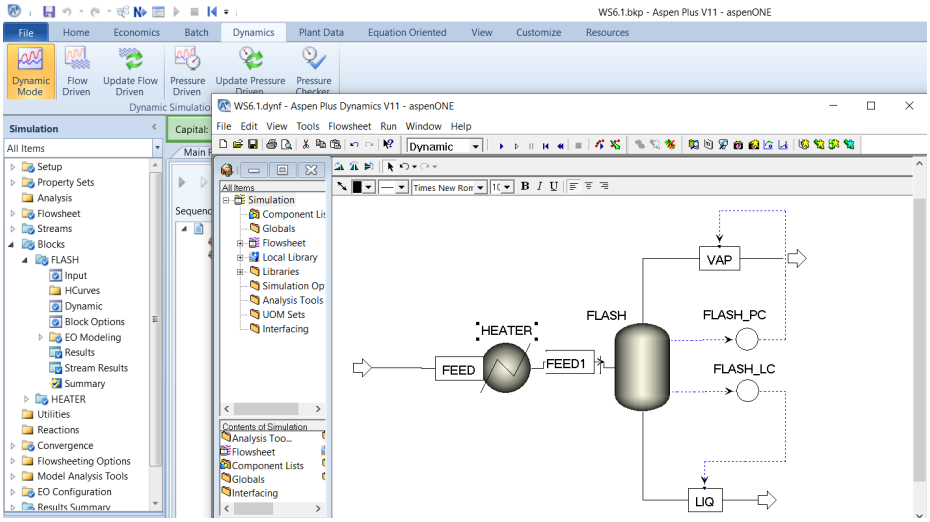


Figure 7.3 Clicking on the Flow Driven button within the Dynamic Tab within Aspen Plus to export the flowsheet to display within Aspen Plus Dynamics.

When we check the AD working folder, we see three simulation files for our example: (1) steady-state simulation Aspen Plus backup file, ***WS7.1.bkp***; (2) dynamic simulation, Aspen Dynamics language file, ***WS7.1.dynf***; and (3) Aspen Plus property data, or problem definition file, ***WS7.1dyn.appdf***. To run the dynamic simulation file property, *we must place the dynamic simulation and property data files within the same working folder.*

For step 4, we open the dynamic simulation, ***WS7.1.dynf***, within AD. We introduce the operation of AD to run a simulation in the next section.

7.2 Running Simulation in Aspen Plus Dynamics

7.2.1 Types of Dynamic Simulations: Flow-Driven and Pressure-Driven

In Figure 7.3, we choose Flow-Driven simulation type, instead of Pressure-Driven. Table 7.2 compares the two types of dynamic simulations. For most liquid-phase polymer processes, we choose flow-driven simulations.

Table 7.2 A comparison of flow-driven and pressure-driven dynamic simulations

Item	Flow-driven simulation	Pressure-driven simulation
1. Specified	Feed flow rate and pressure specified	All feed and product pressures specified; feed flow rate not specified

2. Stream flow rate and pressure difference	Flow rate is not controlled by pressure difference	Flow rate results from pressure difference
3. Outlet flow rate from a block	Specified or determined from mass balance, and not affected by downstream pressure	Determined by pressure and by pressure/flow relationship between upstream and downstream pressures
4.Flow and pressure control	Perfect control assumed for liquid processes, as the pressure/flow dynamics for liquids are very fast	Proper control required
5.Simplicity or complexity of application	Easy to set up; useful for an initial approach to studying the dynamics of liquid processes	More rigorous, and complex to set up (need to balance the pressures within Aspen Plus with valves, pumps, etc.)

7.2.2 Graphical Interface of Aspen Plus Dynamics

Figure 7.4 illustrates the graphical interface of AD. The interface displays several areas: (a) a large *flowsheet window*, in which we see the flowsheet; (b) *simulation explorer*, including areas labeled by (2), (3), (4) and (6); (c) *model libraries for adding streams and blocks to the flowsheet*, area labeled by (5); and (d) *message window*, area labeled by (7).

To display the message window, we click on “View” on the top left row, followed by “Massages”. To display the simulation explorer, we click on the graphical button labeled by (8), which is one of the buttons on the top horizontal tool bar, labeled by (1); we may also click on “Tools” on the top left row, followed by “Explorer”. We then see the area labeled by (2), or “Exploring - Simulation”. Within the simulation explorer, we see the “Flowsheet” folder, labeled by (3), which includes “Blocks” (FLASH, FLASH_LC, FLASH_PC, and HEATER) and “Streams” (FEED, FEED1, IS0, IS1, IS2, IS3, LIQ, and VAP).
We note that IS represents a control signal, labeled by (6), as one of the six types of streams within the model libraries, labeled by (5). Within the flowsheet window, we see the dashed control signal lines for the level controller (LC) and pressure controller (PC).

Figure 7.5 shows the contents of the simulation folder within the simulation explorer.

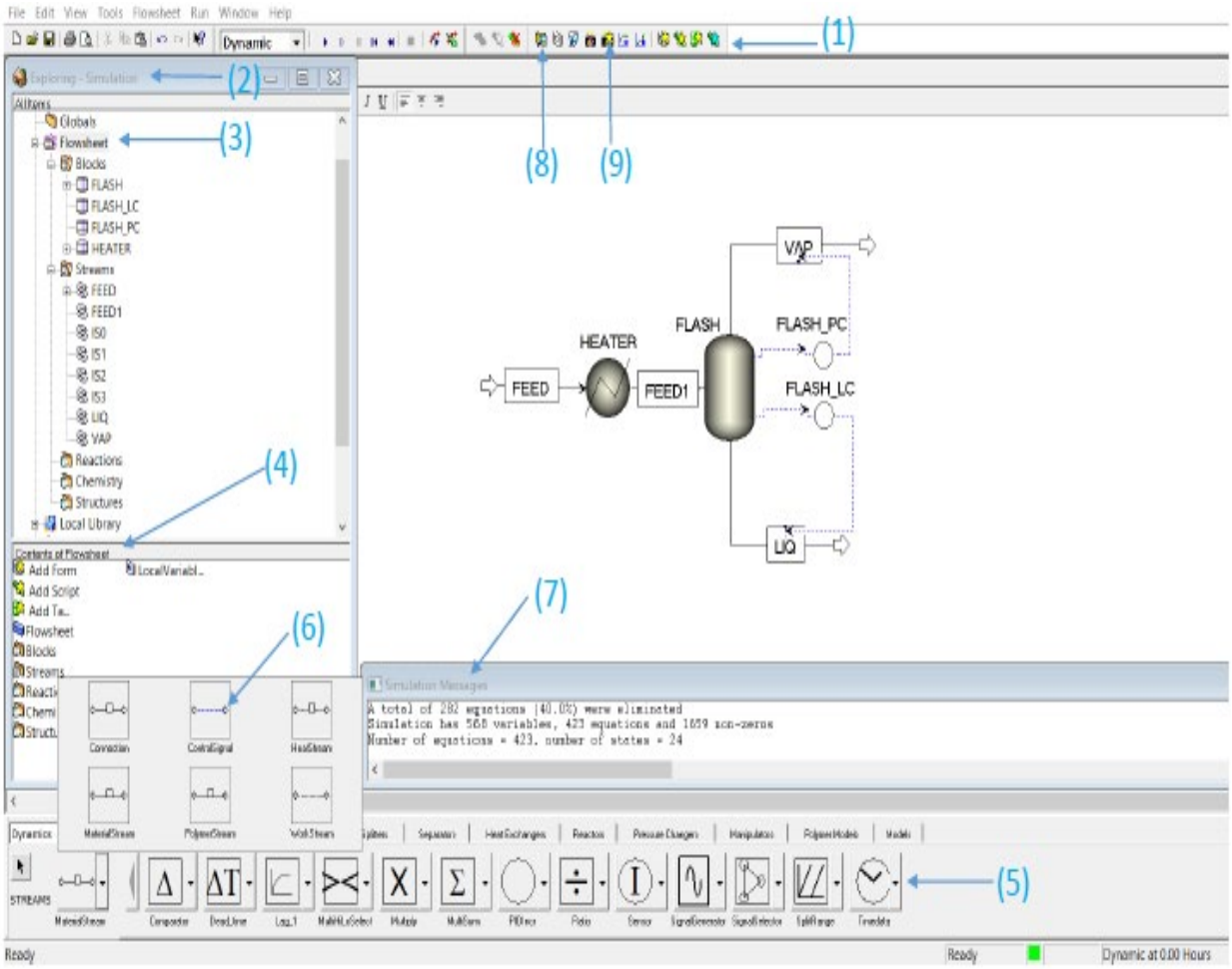


Figure 7.4 An illustration of the graphical interface of Aspen Plus Dynamics

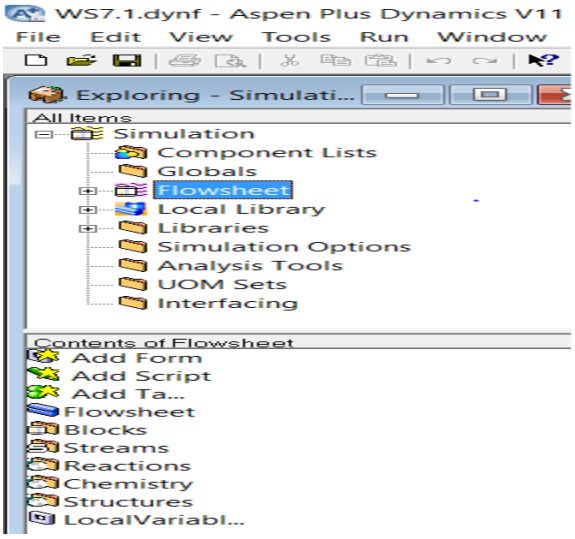


Figure 7.5 Contents of simulation folder and flowsheet folder.

The simulation folder displayed in Figure 7.5 includes the following: (1) *component list* - components and physical properties; (2) *globals* - details of variables global to the simulation, such as ambient temperature, ideal gas law constant R and numerical constant Pi (π = 3.1416) (more later); (3) *flowsheet* - streams and blocks in the flowsheet; (4) *local library* – customized models; (5) *libraries* – library of models; and (6) *diagnostics* – information on simulation resolution.

7.2.3 Simulation Run Modes and Run Control

AD includes the initialization and dynamic run modes. To do a simulation, we begin with an *initialization* run, which solves the equations representing the process at time zero to find the values of the free or unspecified variables. To do a *dynamic* run, we first make an initialization run at time zero, and then integrate the system of equations representing the process step-by-step. AD reports the simulation results at each specified *communication interval.*

Before we make a dynamic simulation, we ***must*** save our original AD dynamic simulation file for a given problem, such as ***WS7.1.dynf***, in a new name, such as ***WS7.1-1.dynf***. We then proceed to use the newly saved file for our dynamic simulation. This step is important, as a time-dependent, dynamic simulation in AD does not automatically save the specifications of the starting file for us, and we must save the original starting file for future use.

To verify that the dynamic simulation file exported from Aspen Plus has correctly specified the required inputs and is ready for simulation runs, we see a "Ready" word together with a green status indictor in the lower right corner of Figure 7.6. Table 7.3 lists the types of variable specifications, and Table 7.4 summarizes the types of run modes in AD.

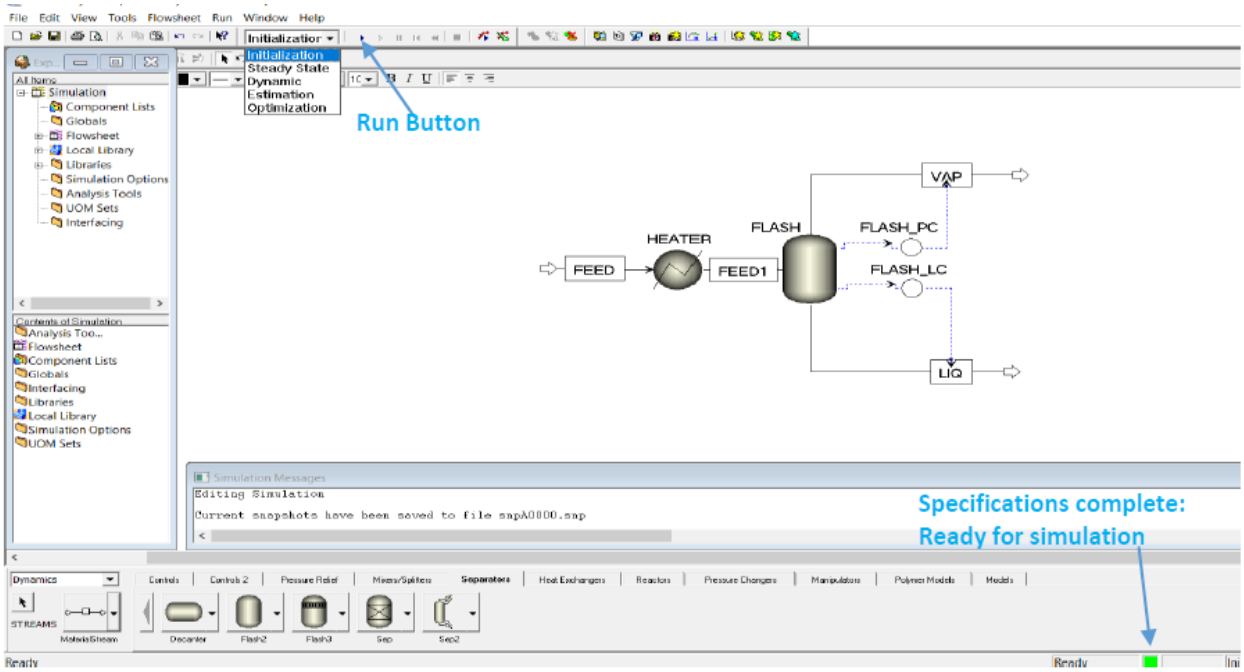


Figure 7.6 An illustration of the initialization run mode, the run button, and the green ready status indicator of specifications complete.

Table 7.3 Types of variable specifications in Aspen Plus Dynamics (AD)

Specification Type	Description
1. Fixed	Variable value specified by the user, and not solved in the simulation
2. Free	Variable value solved by the simulation
3. Initial	Variable value known at time zero for an initialization or a dynamic run
4.RateInitial	Variable whose time derivative is known at time zero for an initialization or a dynamic run

Table 7.4 Run modes in Aspen Plus Dynamics (AD)

Run type	Description
1. Initialization run	Specifying the initial conditions for a subsequent dynamic run. May initialize process variables, their time derivatives, or a combination of both.
2. Steady-state run	Running a simulation where the time derivatives of a dynamic simulation are equal to zero.
3. Dynamic run	Running a simulation where the variables change over time.
4. Estimation run	Fitting model parameters to experimental or process data (parameter estimation); or comparing the steady-state process performance with that predicted by a model (data reconciliation)
5. Optimization run	Optimizing steady-state or dynamic solutions using an objective function and supplied constraints.

7.2.4 Viewing Simulation Results Using Predefined Tables and Plots

We first run an initialization. Next, we set up some predefined tables and plots for our dynamic run. Specifically, we right-click the Feed name to open Forms, followed by setting up the predefined Manipulate table and the TPF (temperature-pressure-molar flowrate) plot. See Figure 7.7.

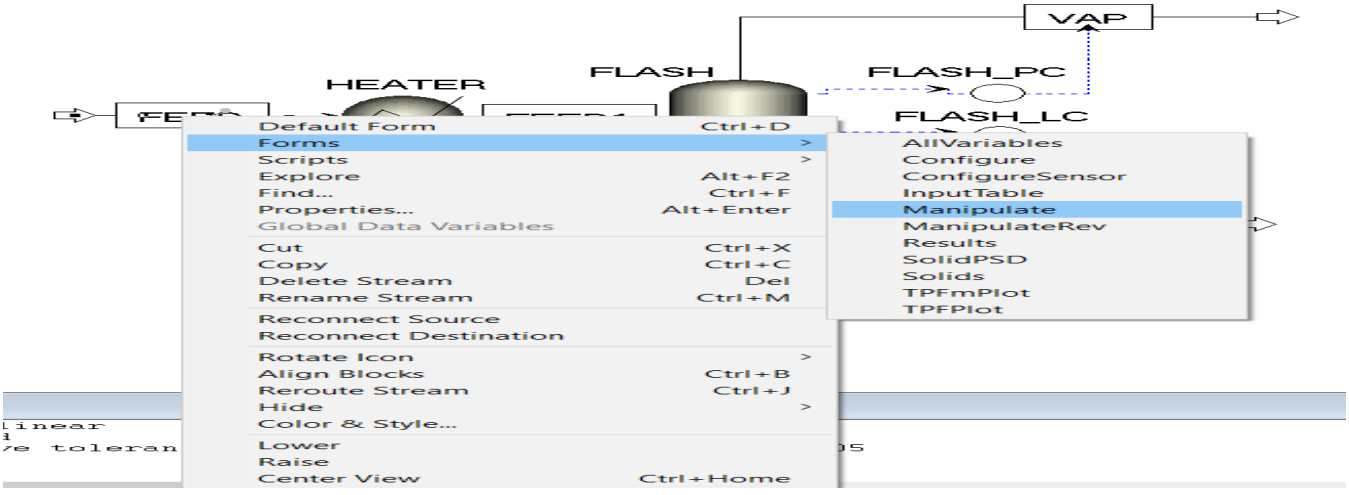


Figure 7.7 Right-click on Feed name to open up Forms, followed by setting up the predefined Manipulate table and TPF (temperature-pressure-molar flow rate) plot.

In the same way, we set up the predefined TPF plots for Feed1 stream from the Heater and both VAP and LIQ streams from the Flash unit. We arrange the resulting table and plots properly within the flowsheet window, as illustrated in Figure 7.8. We note that in the Feed.Manipulate Table, both temperature and pressure are fixed, but the total molar flow is free.

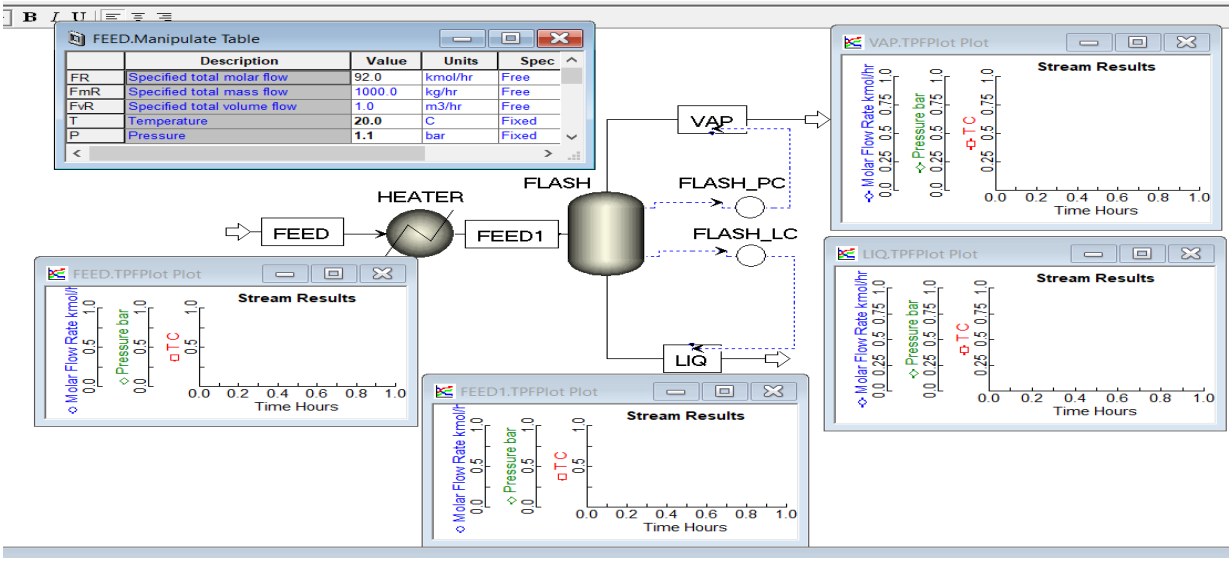


Figure 7.8 Setting up the predefined tables and plots

Next, we change the run type from Initialization to Dynamic, and select Run Options from the Run menu. We enter the required inputs according to Figure 7.9. Note that we pause the dynamic simulation at 1 hr.

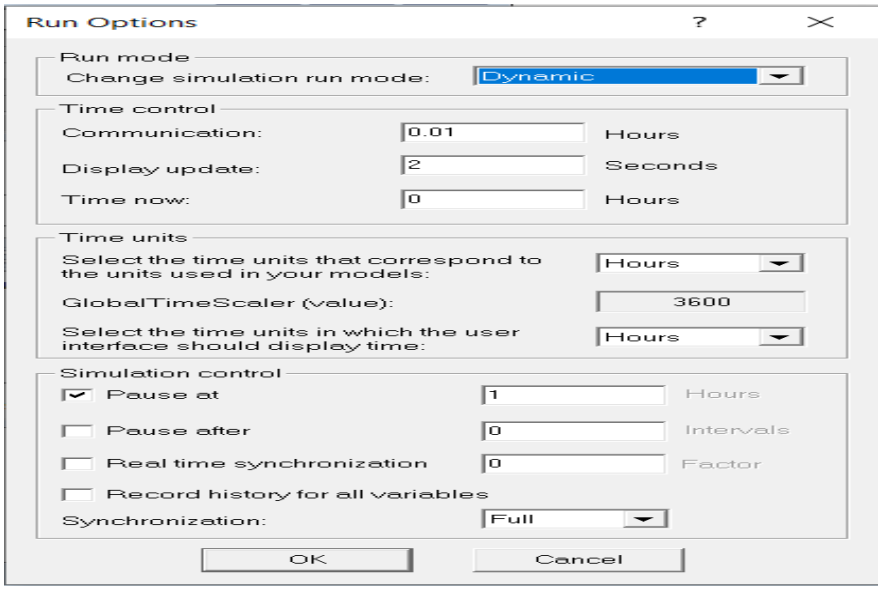


Figure 7.9 Run options showing a pause at 1 hr, with other inputs at default values.

We run the dynamic simulation until it pauses at 1 hr. At that time, we change the Feed mole flow (FR) in the Feed.Manipulate table from 92 to 115 kmol/hr, and change its variable type from Free to Fixed.

We then see the Specification Status button (previously illustrated in Figure 7.6) changes from green to red, indicating that the system is over-specified. See Figure 7.10.

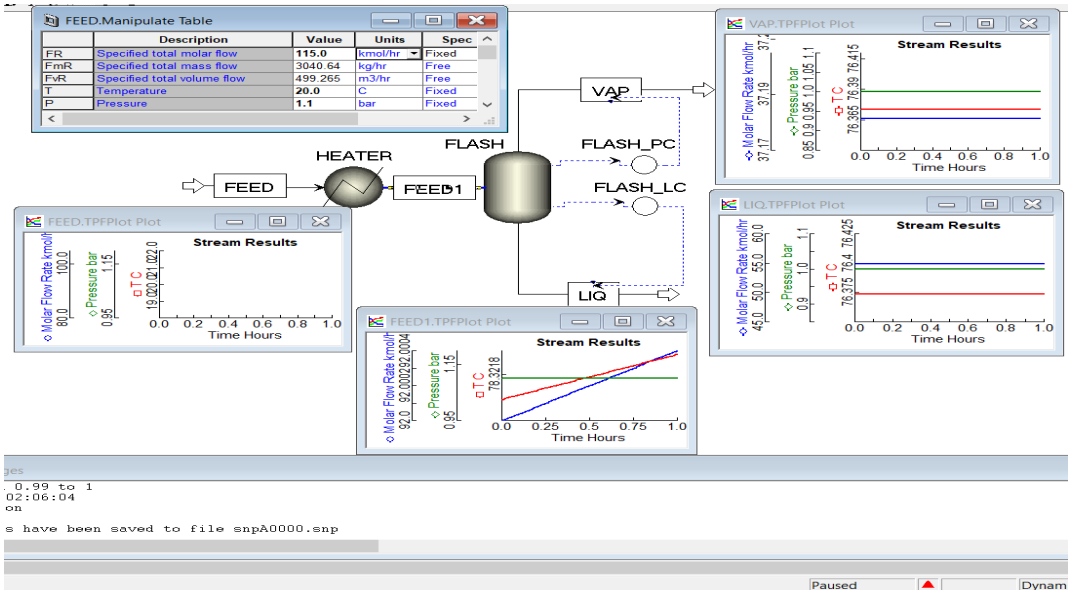


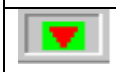
Figure 7.10 Changing Feed stream from 92 to 115 kmol/hr and from a free variable to a fixed variable at 1 hr leads to a change of specification status button from green (complete) to red (over-specified).

7.2.5 Specification Status and Analysis

Table 7.5 summarizes the several types of specification status that we frequently encounter while running AD. Basically, to solve a simulation, the number of equations must equal the number of variables to be calculated (or unknowns). In other words, the total number of degrees of freedom must be satisfied. If necessary, we can adjust the specification by unfixing and then fixing an equal number of variables. For more complex specification status not listed in the table, we recommend the reader to search for "overview of specification status" with AD online help.

Table 7.5 Specification status buttons within AD

Specification status button	**Meaning**	**Explanation**	**Suggested Solution**
	Underspecified	The number of equations is less than the number of variables to be calculated.	Fix more variables or add more equations.
	Overspecified	The number of equations is greater than the number of variables to be calculated.	Fix less variables or remove redundant equations.
	Complete	The number of equations equals the number of variables to be calculated.	Ready to run simulation.
	Undetermined	Cannot determine the status of the flowsheet.	Check the simulation message for possible causes. Could be an empty

			simulation, or inconsistency in the types used, or in the flowsheet connectivity.
	Underspecified of initial variable	The number of initial variables is less than the number of process variables.	Initialize more process variables or initialize more time-differential variables.
	Overspecified of initial variables	The number of initial variables is greater than the number of process variables.	Un-initialize some process variables, or un-initialize some time-differential variables.

According to Table 7.5 and Figure 7.11, our simulation example is over-specified when changing Feed stream from 92 to 115 kmol/hr and from a free variable to a fixed variable. To know more details of this overspecifcation, we double-click the red status button to open the *Specification Analysis* tool of AD. We then click the Analyze button to show the "Recommended changes to specifications", that is, to change the mole flow of nitrogen in the Feed from a fixed variable to a free variable. See Figure 7.11. In other words, after changing Feed steam from 92 to 115 kmol/hr, we let the simulation calculate the new mole flow of nitrogen in the Feed (the nitrogen mole flow in the Feed changes from 2 kmol/hr specified in Table 7.1 to 25 kmol/hr).

We should always think over the recommendation changes by the Specification Analysis to see if they are reasonable, and do not blindly accept all the recommended changes. In the current example, we choose to accept the recommended change, and click on the Accept button. We see the status button change from red to green again.

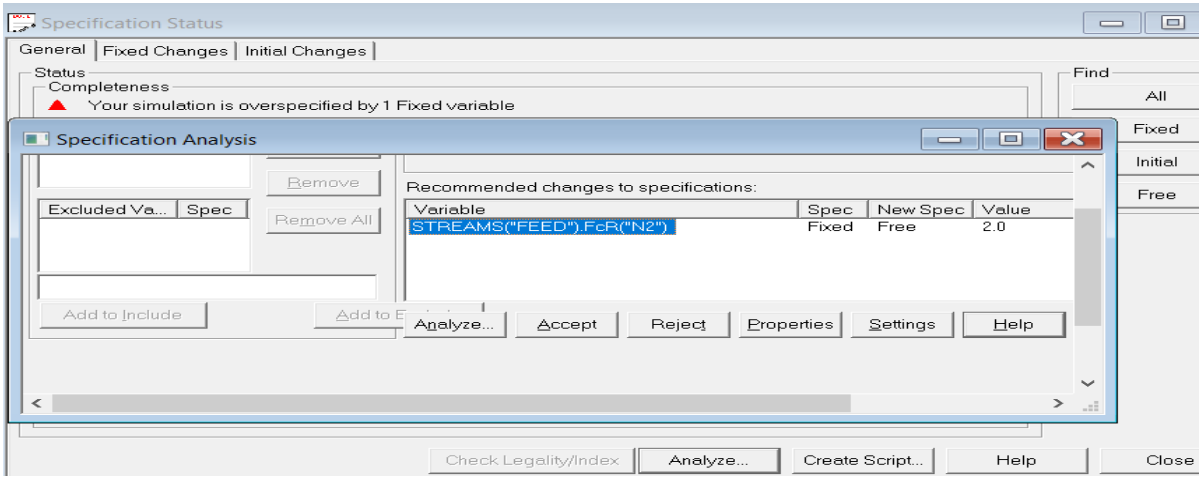


Figure 7.11 Using the Specification Analysis tool with AD to identify recommended changes to make variable specifications complete.

We then run the dynamic simulation from 1 hr to 5 hr by following Figure 7.9 and ask AD to pause the simulation at 5 hr. We illustrate the results in Figure 7.12. In displaying the graphical results, we need to double-click the x-axis of each figure ("Time Hours") to open up the "Plot Axis Scale Setting" and change its "Axis Range" from 0 to 5 hr. We see that mole flows, temperature, and pressure of VAP and LIQ approach new steady-state values at 5 hr.

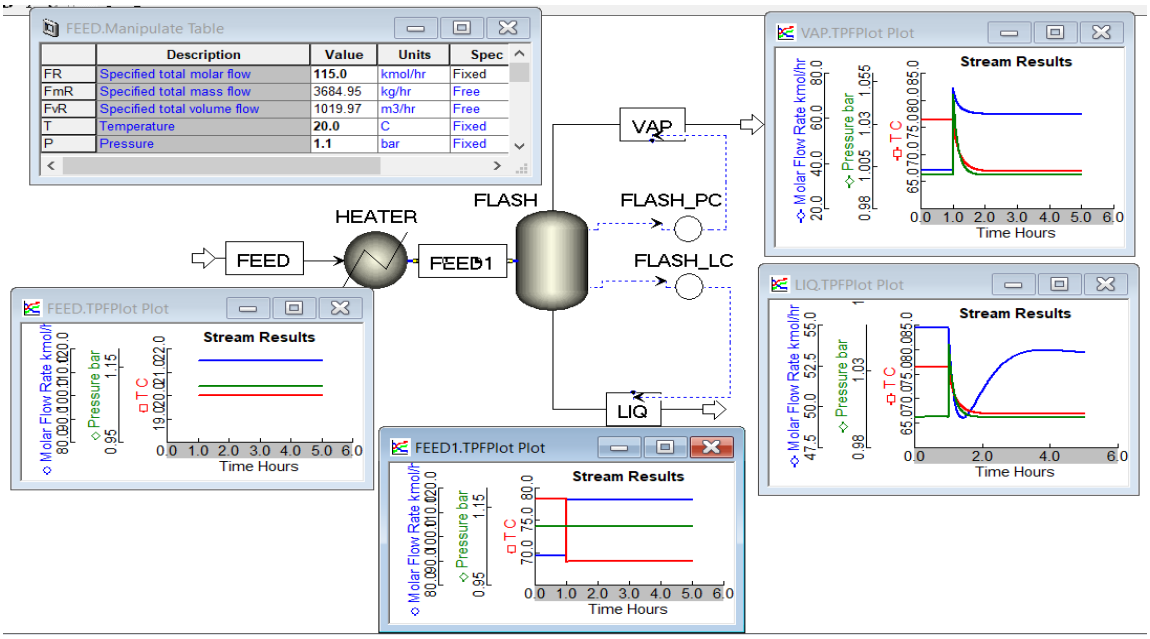


Figure 7.12 Simulation results from 1 hr to 5 hr after increasing Feed mole flow from 92 to 115 kmol/hr at 1 hr.

7.2.6 Creating New Tables and Plots

Instead of using predefined tables and plots (Figure 7.7), we demonstrate how to create our own tables and plots in AD. First, we open our original simulation file, ***WS7.1.dynf***, and save it as ***WS7.1-2.dynf***. Following the run options in Figure 7.9, we run dynamic simulation for 1 hr and pause. We then follow Figure 7.10 to change Feed stream from 92 to 138 kmol/hr and from a free variable to a fixed variable at 1 hr. We also change the mole flow of nitrogen in the Feed from a fixed variable to a free variable according to Figure 7.11. We then run the dynamic simulation to pause at 10 hr.

To create a plot or a table, we click the New Form button in the top horizontal tool bar that is circled in red in Figure 7.13 to open a New Flowsheet Form. For a plot, we select the type, Plot, and enter name, Flash, and click OK to generate an initially empty "Flash Plot". For a table, we select the type, Table, and enter name, FlashResult, and click OK to generate an initially empty "FlashResult Table".

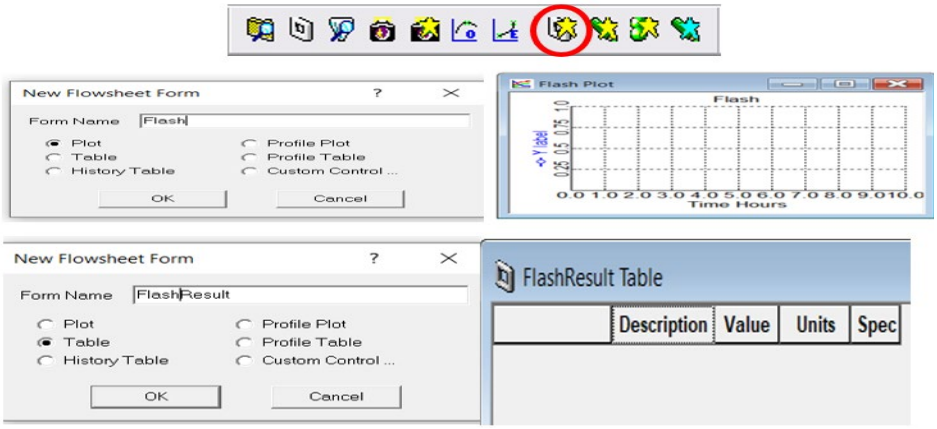


Figure 7.13 Creating a plot or a table through the New Form button circled in red.

Refer to Figure 7.14. We double-click the Flash block to open the large FLASH.Results Table. We select the variable T (temperature), and drag and drop it to the small new table, FlashResult Table. Specifically, we click the selected variable once, click and hold until a circle-with-slash icon appears, and then drag and drop. Repeat the same operation for variables P (pressure) and level (liquid level). This result in a FlashResult Table with T at 109.411 °C, P at 16.3486 bar, and level at 1.625 m at 10 hr. Next, to show the time-dependent change of these three variables from 0 hr to 10 hr in the Flash Plot, we drag and drop the same three variables from the FLASH.Results Table to the Flash Plot.

We note that the names of the created plot, Flash, and the created table, FlashResult, appear in the Contents of Flowsheet, as illustrated previously in area labeled (4) in Figure 7.4.

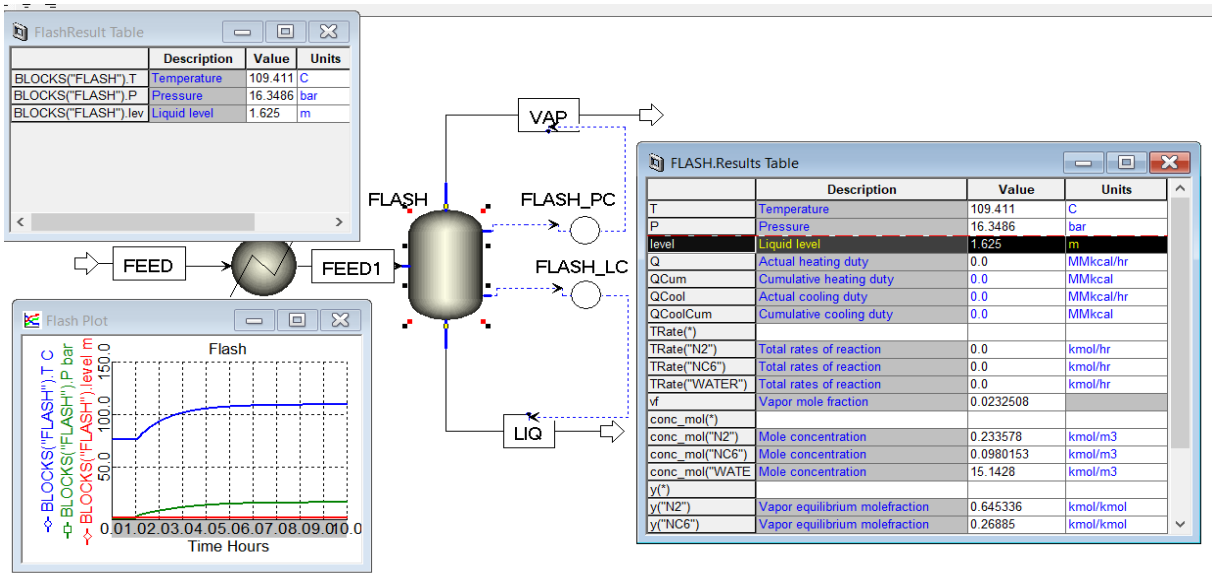


Figure 7.14 Dragging and dropping selected variables from FLASH.Resuts Table to the created FlashResult Table and Flash Plot.

7.2.7 Variable Finding in Aspen Plus Dynamics (AD)

We can use Variable Find to create a list of variables that we have specified. From this list, we can create a table or a plot, or change the properties of a variable. To use “Variable Find”, we click on “Tools” on the top left row, followed by “Variable Find”. Alternatively, we click the “Variable Find” button in the top horizontal tool bar that is circled in red in Figure 7.14 to open a Variable Find Form.

As in Section 7.2.6, we first open our original simulation file, ***WS7.1.dynf***, and save it as ***WS7.1-3.dynf***. Following the run options in Figure 7.9, we run dynamic simulation for 1 hr and pause. We then follow Figure 7.10 to change Feed stream from 92 to 138 kmol/hr and from a free variable to a fixed variable at 1 hr. We also change the mole flow of nitrogen in the Feed from a fixed variable to a free variable according to Figure 7.11. Before we run the dynamic simulation to pause at 10 hr, we demonstrate how to use Variable Find to set up a plot of the free nitrogen mole flow in the Feed stream. We complete this by following the steps illustrated in Figures 7.15 and 7.16.

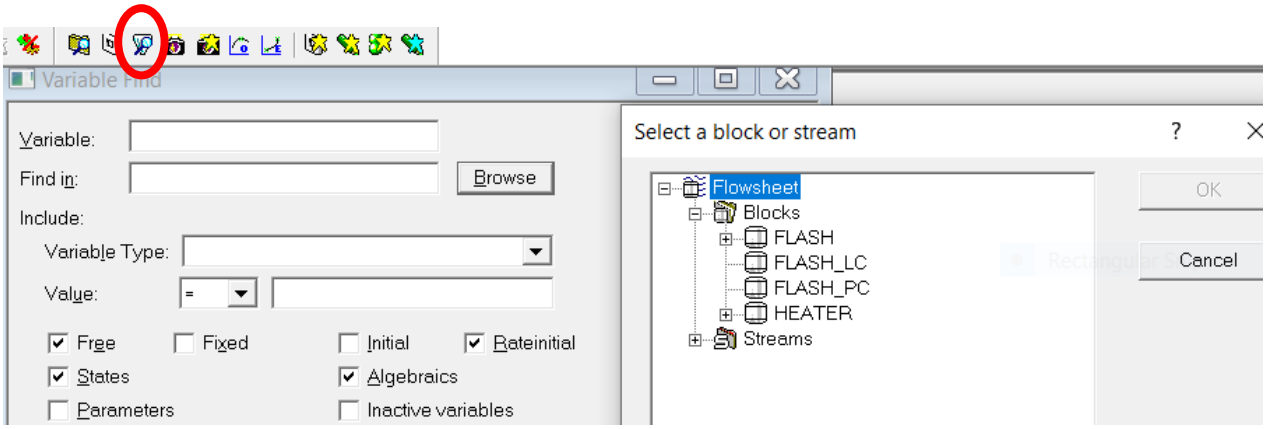


Figure 7.15 Clicking on Variable Find button (circled in red) to open the Variable Find Form; choosing free, algebraic and state variables; and clicking on Browse to find free variables in the FLASH block.

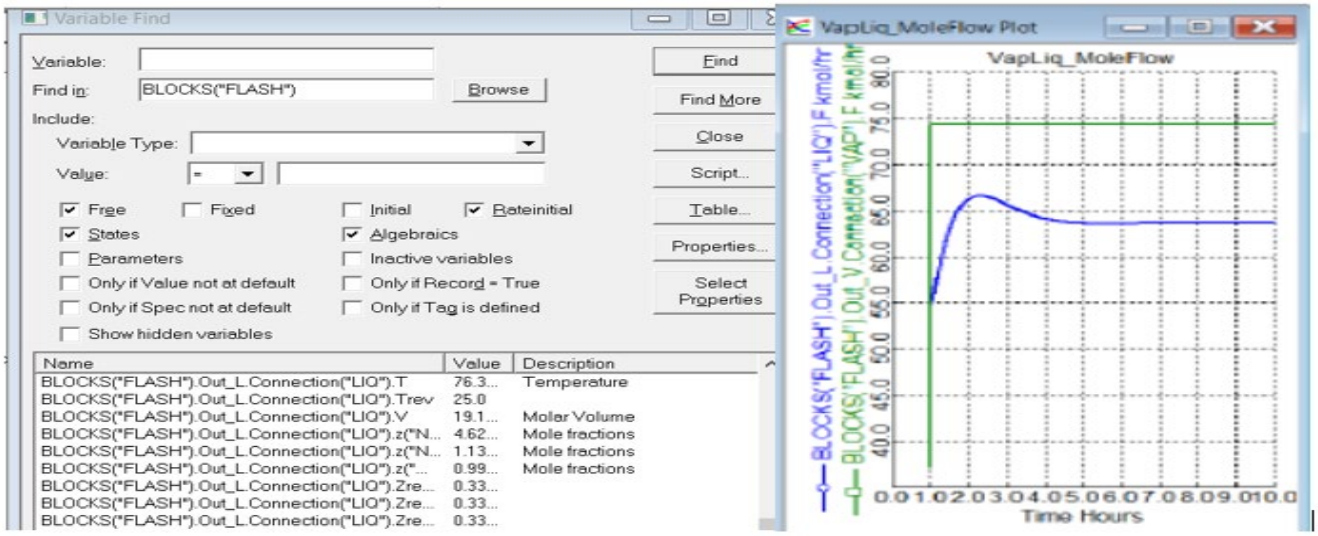


Figure 7.16 Clicking on "Find" button in the Variable Find to identify free variables in the FLASH block; setting up a plot named VapLiq_MoleFlow following Figure 7.14; dragging the free FLASH Vap and Liq mole flow variables and dropping them to the plot.

We then choose Run Options from the Run menu, and ask the dynamic simulation to pause at 10 hr. Figure 7.16 displays the resulting Vap and Liq mole flows from the FLASH block.

7.3 Process Control in Aspen Plus Dynamics

7.3.1 Adding a PID Controller

We begin by opening the Aspen Plus simulation file, ***WS7.2.bkp.*** We click the Dynamics tab on the ribbon and click the Dynamic Mode button to activate the dynamic input form. We complete the relevant dynamic data according to Table 7.6. After running the steady-state simulation, we export the simulation from Aspen Plus to AD, and save the resulting dynamic simulation file, ***WS7.2.dynf***, and the Aspen Property problem definition file, ***WS7.2.appdf***. Figure 7.17 shows the resulting flowsheet with the added level and pressure controllers.

Table 7.6 Specifications of the steady-state simulation, WS7.2.bkp

Components	N2, NC6 (n-hexane) and water
Property Method	NRTL thermodynamic method with Henry component, N2
Stream	Feed: 20°C, 1.1 bar, mole flow (kmol/hr): N2 - 2, NC6 – 20 and water - 70
Heater block	0 bar (no pressure drop), output vapor fraction = 0.4; dynamic heater, inlet volume =1 cum, outlet volume = 1 cum; medium flow direction – countercurrent; medium temperature = 60°C; temperature approach = 10°C
Flash block	75°C, 1 bar; vertical vessel; elliptical type; length = 4.5 m; diameter = 1.5 m; initial condition – liquid volume fraction = 0.4; default controllers - pressure and level

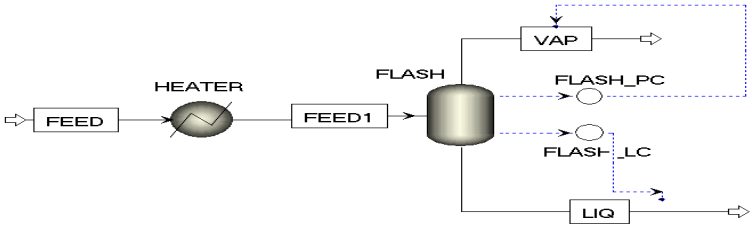


Figure 7.17 The starting dynamic simulation flowsheet for ***WS7.2.dynf***

After opening the exported dynamic simulation in AD, we make an Initialization run first, and then change the run mode to Dynamic. In the following, we demonstrate how to add a PID temperature controller on stream FEED1 (feed to the FLASH block) and manipulate the heater heat duty from the HEATER block.

Step 1: See Figure 7.18. Go to View menu to display the Model Library. Click the **PID Incr** icon (circled in red in the figure) on the Model Library. Click on the flowsheet to place the new controller B1. Right-click the controller block to see “rename block”. Change controller name from B1 to TC.

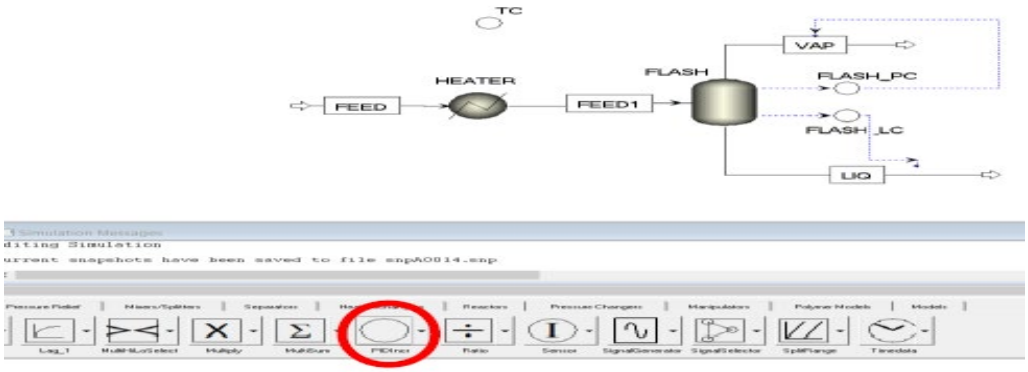


Figure 7.18 Adding a PID temperature controller to the flowsheet.

Step 2. (1) See Figure 7.19a. Click the **Control Signal** stream type in the Model Library. On the TC controller block, we see an input signal arrow (for process variable or controlled variable, FEED 1 temperature) on the left and an output signal arrow (for manipulated variable or controller output, HEATER heat duty) on the right.

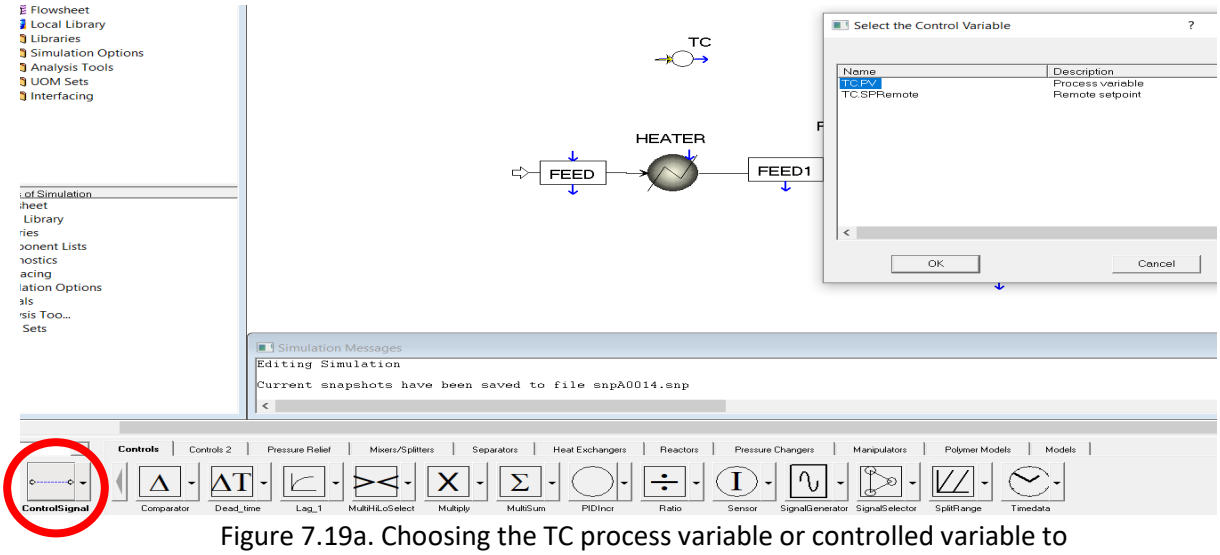


Figure 7.19a. Choosing the TC process variable or controlled variable to connect to the left input signal of TC.

Step 2. (2) See Figure 7.20b. Connect the left control signal to the output signal arrow of FEED1 and choose FEED1 temperature as the process variable or controlled variable.

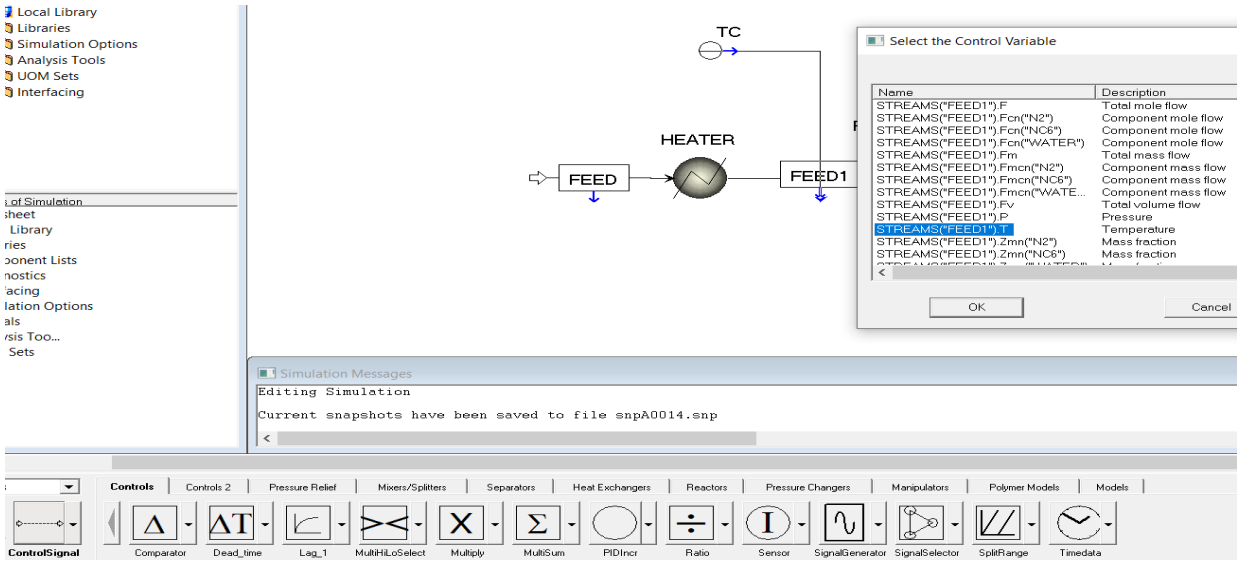


Figure 7.19b. Choosing FEED1 temperature as TC process variable or controlled variable to connect the right output signal of TC.

Figure 7.19c shows the resulting left controller input signal connection to the process variable or controlled variable, FEED 1 temperature.

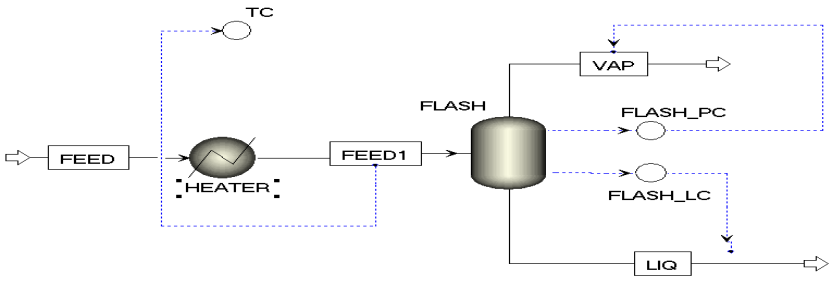


Figure 7.19c. Connection between the left controller input signal to the process variable or controlled variable, FEED1 temperature.

Step 3. (1) See Figure 7.19d. Click the Control Signal stream type in the Model Library. On the TC controller block, we see an output signal arrow (for manipulated variable or controller output, HEATER heat duty) on the right.

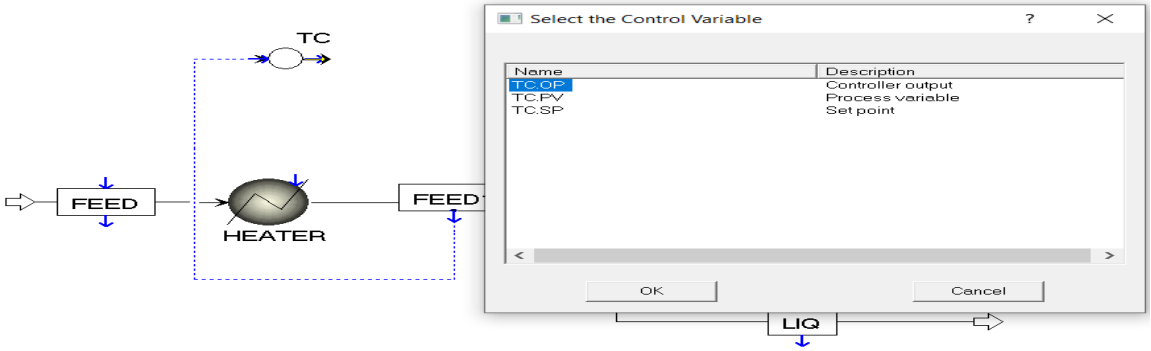


Figure 7.19d. Choosing the TC manipulated variable or controller output to connect to the right output signal of TC.

Step 3. (2) See Figure 7.19e. Connect the right control signal of TC to the input signal arrow of HEATER and choose heat duty as the manipulated variable or controller output.

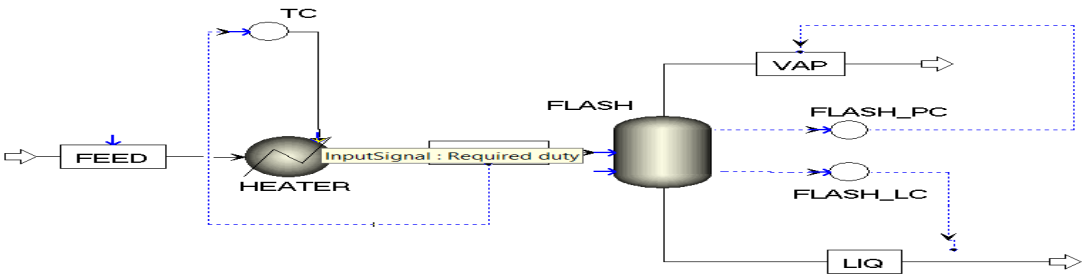


Figure 7.19e. Choosing HEATER heat duty as TC manipulated variable or controller output to connect the right output signal of TC.

Figure 7.19f shows the resulting right controller output signal connection to the manipulated variable or controller output, HEATER heat duty.

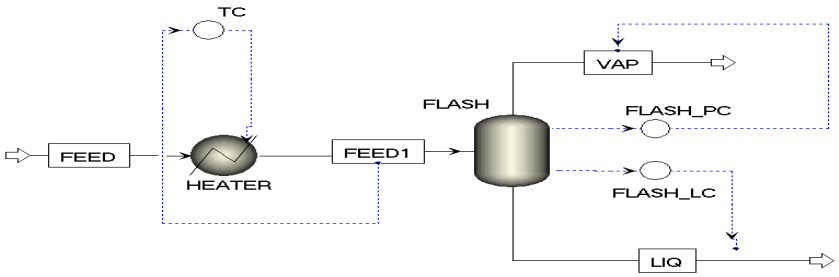


Figure 7.19f. Connection between the right controller output signal to the manipulated variable or controller output, HEATER heat duty.

7.3.2 Configuring a PID Controller

After we have added the controller, we save the current dynamic simulation file under a new name, ***WS7.2-1.dynf*** and proceed with the first case study of the controller. We save the current dynamic simulation file, ***WS7.2.dynf***, as a backup starting file for additional case studies.

We double-click the controller TC to display the controller faceplate. See Figure 7.20.

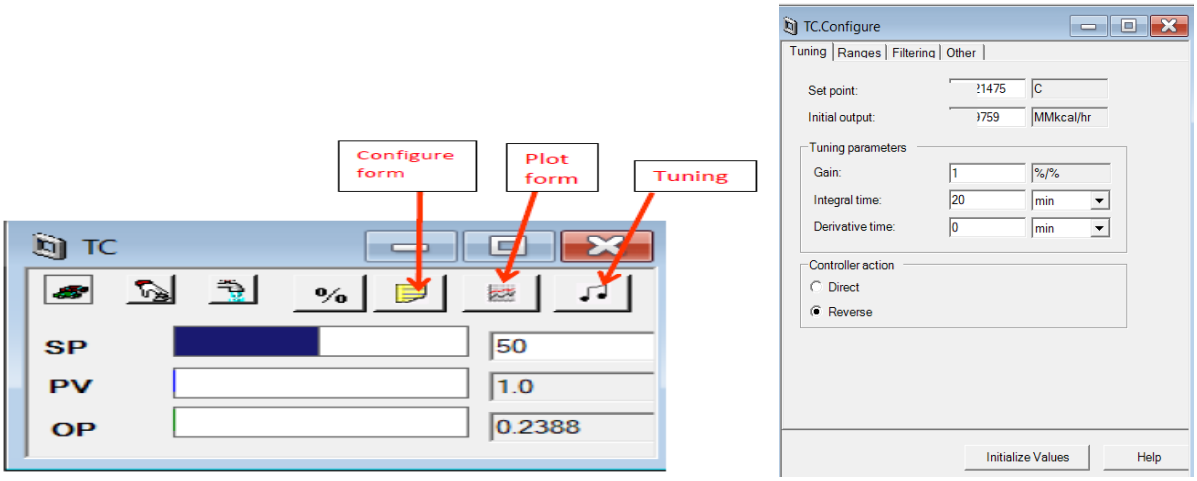


Figure 7.20 Controller faceplate and configure form.

In the faceplate, the first three buttons from left to right at the top level enable us to switch among *auto, manual and cascade modes*, respectfully. The fourth button (%) allows us to switch between viewing values in *process units or percentages of range*. To see the process units, we can hold the mouse over the label SP (setpoint), PV (process variable) or OP (controller output). The fifth to seventh buttons are, *Configure form, Plot form and Tuning form*, respectively, which we demonstrate further below.

First, we click the Configure form button, and then the **Initialize Values** button. This configures the controller with default values: (1) SP is the current controlled variable (FEED1 stream temperature at 78.3214°C); (2) OP is set to the current controller output or manipulated variable (HEATER het duty at 0.259759 MMkcal/hr). The range of PV and OP are specified to 0 and twice the current variable value.

Clicking the "Other" tab within the Configure form of Figure 7.21 shows the types of PID controller algorithms available with AD.

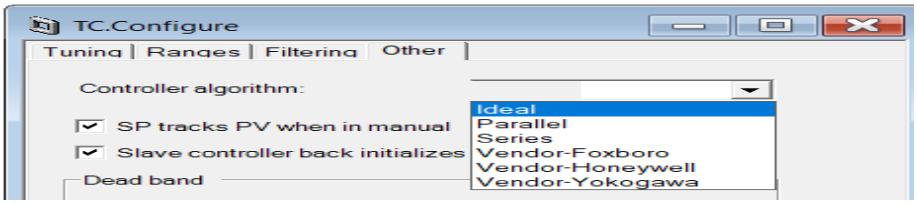


Figure 7.21 PID controller algorithms available displayed within the Other folder in Configure form.

Table 7.7 describes the ideal, series and parallel algorithms. See AD online help wth vendor algorithms. We choose the Ideal algorithm for the current example.

Table 7.7 Typical PID controller algorithms

PID algorithm	**Model equation**
1. Ideal	OP =Bias + K_c x {E_p + $(1/T_I)\int E_I\,dt$ +$T_D\,dE_D/dt$}
2. Series (interacting form)	OP =Bias + K_c x {E_p + $(1/T_I)\int E_I\,dt$} x { 1 + $T_D\,dE_D/dt$}
3. Parallel (standard , ISA, or non-interacting form)	OP =Bias + K_c x E_p + $(1/T_I)\int E_I\,dt$ + $T_D\,dE_D/dt$
OP = controller output; Bias =the value of the manipulated variable when the process is at steady state; Gain = proportional gain K_c; E = error = setpoint – process variable; E_p = proportional mode error; E_I = integral mode error; E_D = Derivative mode error; T_I = integral time; T_D = derivative error	

We give some observations on the PID controller tuning parameters below:

(1) The proportional gain, Kc, is typically expressed as 100/(proportional band, PB). The PB is the error (expressed in percentage of the range of the measured variable) required to move the control valve (the final control element) from fully closed to fully open. Typically, PB has a value between 30 and 300.

(2) The larger the proportional gain, K_c, the smaller the difference between the setpoint value and the process variable value at steady state (called steady-state error or offset). P(proportional) control corrects the present error between the setpoint and the process variable.

(3) The integral time or the reset time, T_I, is the time it takes the controller to give an output that is twice the output from a proportional controller, following a step change in error. It other words, the integral time is the time it takes for the controller to "repeat" the proportional controller action.

(4) The integral controller acts to eliminate the steady-state error (offset) that is present in applying the proportional controller action alone. A large integral or reset time essentially minimizes the integral controller action. PI(proportional-integral) control corrects the past and present errors between the setpoint and process variable.

(5) The derivative action increases the stability of the controlled system, and permits either a higher controller gain K_c, or a lower integral or reset time, T_I, to be used. The latter speeds up the dynamic response of the controlled system. If the derivative time T_D is large enough, the controlled system would be theoretically stable for all the controller gains. PID(proportional-integral-derivative) control corrects the past, present and future errors between setpoint and process variable, considering the positive or negative rate of change of the error in the output variable in affecting the future error.

Table 7.8 gives the heuristic PID controller tuning parameters for different types of controllers [1,2].

Table 7.8 Initial PID controller tuning parameters

Controller type	Proportional gain, K_c	Integral time, T_I, min	Derivative time, T_D, min
Flow	0.1	0.2	0
Level	2	10	0
Pressure	2	2	0
Temperature	1	20	0
Composition	0.1	0.2	0

We follow these initial values for our temperature controller, as illustrated in Figure 7.20. We also choose **Reverse** controller action for this temperature controller because we need to decrease the HEATER heat duty when the temperature increases. Table 7.9 summarizes the other options for controller action.

Table 7.9 PID controller actions

When the action is	And the measured variable (controlled variable or process variable)	Then the manipulated variable
Direct	Increases	Increases
Direct	Decreases	Decreases
Reverse	Increases	Decreases
Reverse	Decreases	Increases

Lastly, we mention that AD automatically generates pressure, level and temperature according to the guidelines of Table 7.10.

Table 7.10 Guidelines for AD default controllers

Controller added	When	Measured variable (controlled variable or process variable)	Manipulated variable	PID tuning parameters
Pressure	Vapor holdup is modeled	Pressure in vessel	Vapor outlet flow rate	Kc = 0.2, T_I = 12 min
Level	Liquid holdup is modeled	Liquid level	Liquid outlet flow rate	Kc = 0.2
Temperature	Stirred reactor block	Vessel temperature	Heat duty	Kc = 0.2, T_I = 12 min

We now select Run Options from the Run menu, accept the default parameters, and ask the simulation to pause at 5 hr. On the Controller faceplate, we click the Plot form to display both the controlled variable or process variable (FEED1 temperature) and the controller output or manipulated variable (HEATER heat duty). See Figure 7.22. We then run the simulation until it pauses at 5 hr. Next, we follow Figure 7.12 to change Feed stream from 92 to 138 kmol/hr and from a free variable to a fixed variable at 1 hr. We also change the mole flow of nitrogen in the Feed from a fixed variable to a free variable according to Figure 7.12. We then run the dynamic simulation to pause at 15 hr. We see the controller response in Figure 7.22.

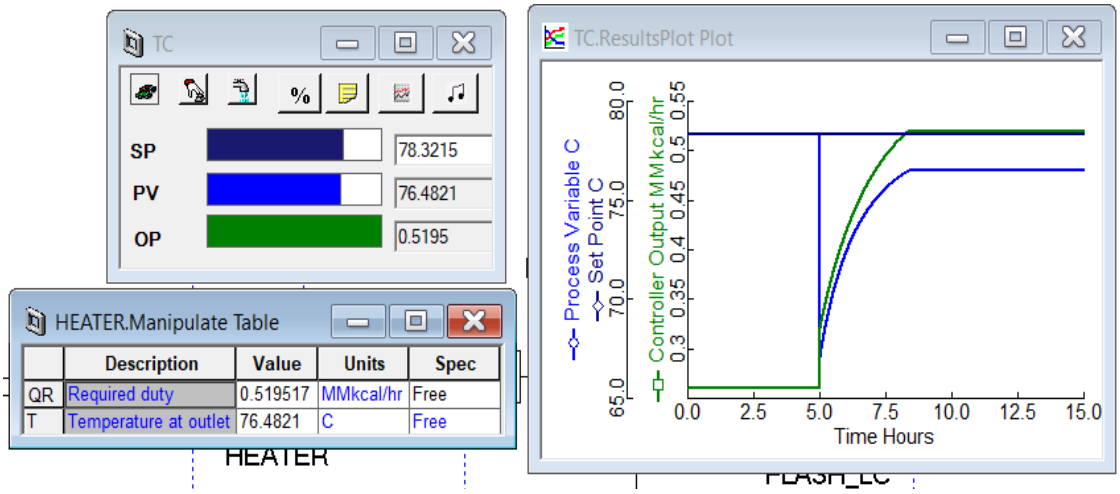


Figure 7.22 Controller response after increasing the FEED mole flow from 92 to 138 kmol/hr: FEED1 temperature (PV) changes from 78.3215 to 76.4821°C, while HEATER duty increases from 0.2598 to 0.5195 MMkcal/hr.

We show in Table 7.11 the available control-related models within AD. The reader may refer to AD online help for the description and example of using a specific model. In Section 7.7, we illustrate the use of the split-range controller in a HDPE workshop.

Table 7.11 Control-related models available within AD

Model	Description
Comparator	Calculates the difference between two input signals
Dead_time	Delays a signal by a specified time
DMCplus	Interface to DMCplus online control
Discretize	Discretizes a signal
HiLoSelect	Selects the higher or lower of two input signals
IAE	Calculates the integral of the absolute value
ISE	Calculates the integral of the squared error
Lag_1	Models a first-order lag between the input and output
Lead_lag	Models a lead-lag element
Multiply	Calculates the product of two input signals
PIDIncr	A three-mode proportional-integral-derivative controller
PRBS	Generates a pseudo-random binary signal
Ratio	Calculates the ratio of two input signals
Scale	Scales an input signal
SplitRange	Models a split-range controller
Sum	Calculates the sum of two input signals
Transform	Performs a loge, square, square root, or power transform
Valve_dyn	Models the dynamics of a valve actuator

7.4 Snapshots

A snapshot is a saved set of values and specifications for all variables in a simulation. It is useful for moving back to a point when simulation was converged. The snapshot is created automatically after every run type, except for dynamic. We can also take manual snapshots at desired time.

Refer to Figure 7.23. We see the Snapshot Management and Take a Snapshot buttons in the top horizontal tool bar. Click Snapshot Management and select the Creat tab. We see that we can request AD to take snapshots automatically under the specified conditions.

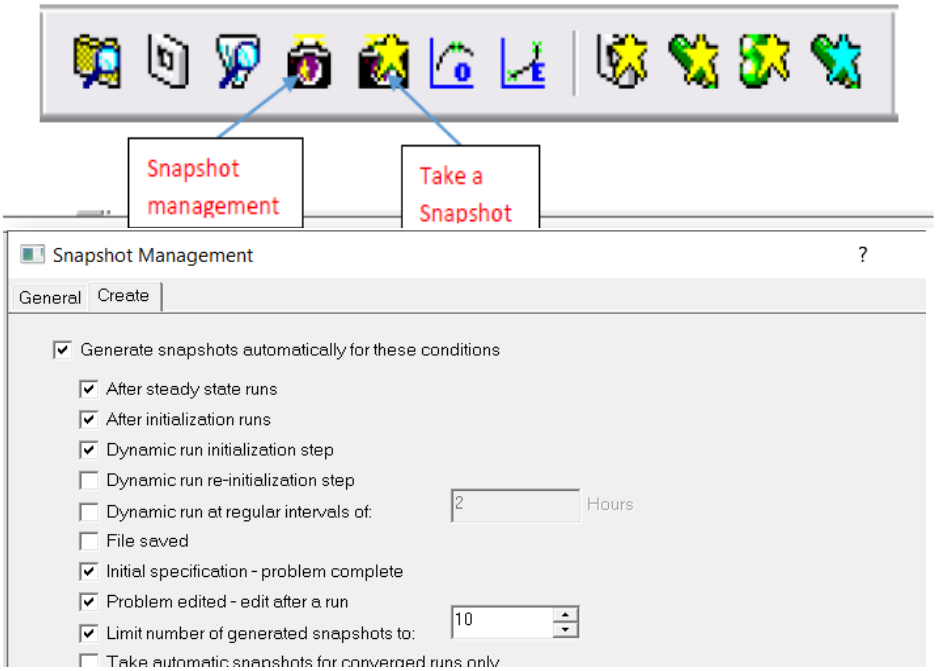


Figure 7.23 Snapshot Management and Take a Snapshot buttons, and choosing when to take an automatic snapshot under Create within Snapshot Management.

Next, refer to Figure 7.24. We can also ask AD to take a snapshot at a desired simulation time. We click the Take a Snapshot button, enter the description “End_of_5_hr_run” and click OK. We then see a snapshot under this name as a new listing under the General folder of Snapshot Management.

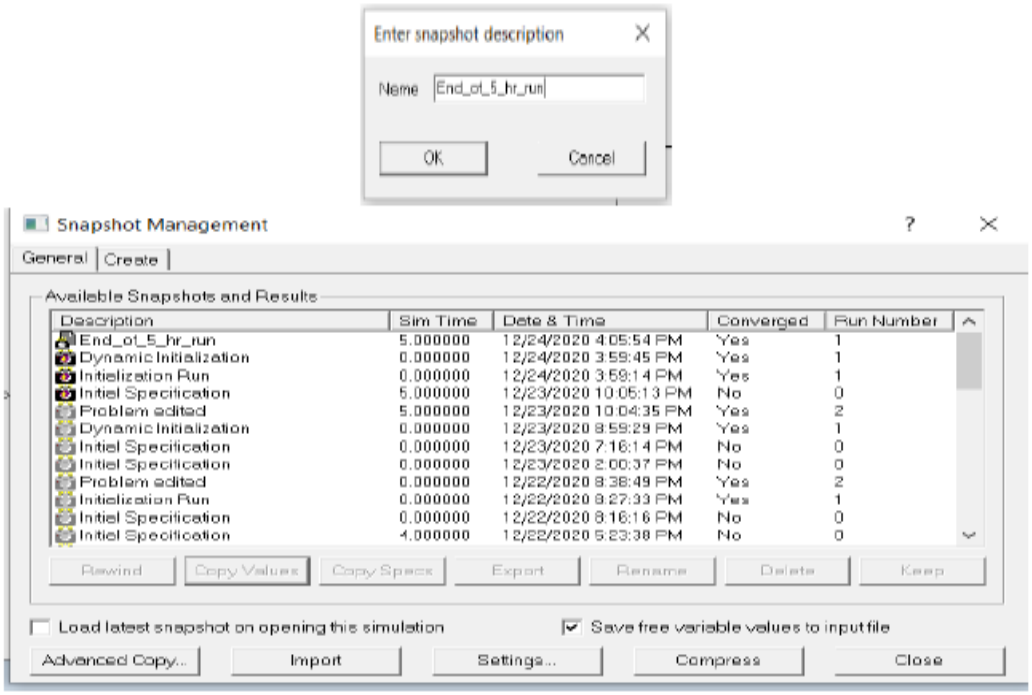


Figure 7.24 Taking a Snapshot and Listing Under Snapshot Management.

We can copy values from any snapshot or result to the current simulation. This copies values when the name of the variable in the snapshot or result matches the name of a variable in the current simulation.

We can rewind the simulation back to a previous time point if snapshots are available. If there are no snapshots, rewinding will take the simulation back to time zero.

7.5 Tasks

A task is a set of instructions which defines a sequence of discrete actions, such as disturbances in feed condition, or changes to controller set points. Tasks can trigger an event or action when a predefined condition becomes true. An example is to close or open a valve when the fluid level falls/rises above/ below a value. All task statements are executed in sequence.

There are two types of tasks within AD: (1) *Event-driven tasks* – it requires a conditional expression to determine when the task begins. These events can be explicit or events which always happen and are usually time-specific, or implicit or events may or may not happen depending on other events or conditions. (2) *Callable tasks* - they are called by other tasks instead of being triggered by an event. You can optionally define callable tasks with parameter call lists, with the calling task passing the parameter values. You can make calls to tasks within models, to tasks in a tasks folder, or other tasks in your flowsheet.

Figure 7.25 shows the structure of a task manager. An event-driven task must be *activated* to be considered by the task manager during the dynamic simulation.

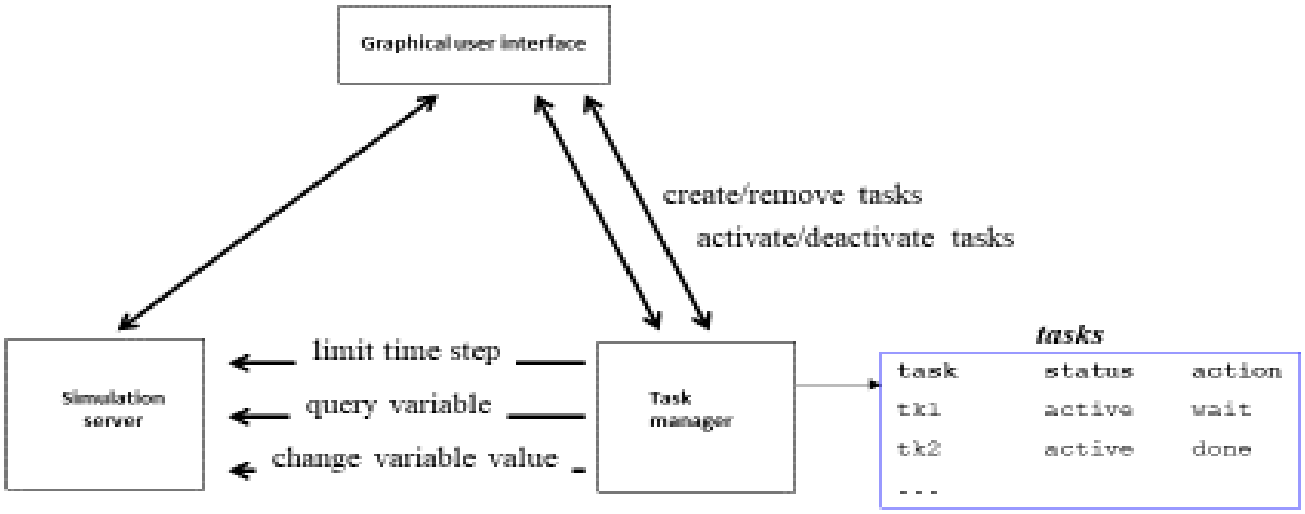


Figure 7.25 The structure of a task manager.

We illustrate the task by opening a steady-state simulation file, ***WS7.3.bkp***. Figure 7.26 shows the flowsheet, and Table 7.12 specifies the input.

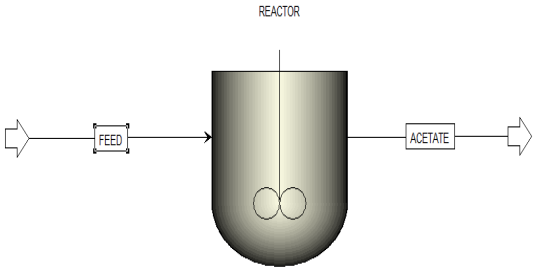


Figure 7.26 Flowsheet for WS7.3

Table 7.12 Specifications of the steady-state simulation, ***WS7.3.bkp***

Components	Acetic acid (HOAc), ethanol(EtOH), ethyl acetate (EtOAc), water
Thermodynamic Method	NRTL
Stream	Feed: 70°C, 1 atm, Mass flow (kg/hr) – 2300 (HOAc), 1600 (EtOH), 100 (water)
Block	Reactor: 1 atm, 70°C, liquid-only, reactor volume = 2.5 cum; Dynamic data: vertical vessel, elliptical, length = 2 m, heat-transfer option- constant temperature, medium temperature = 20°C, initial condition – liquid volume fraction = 0.5
Reactions	HOAc + EtOH ->EtOAc + H2O; k = 1.9E8, E = 5.95 J/kmol

Running the steady-state simulation and then exporting the simulation to AD gives a dynamic simulation file, ***WS7.3.dynf***, and an Aspen property data file, ***WS7.3dyn.appdf***. On the dynamic simulation flowsheet of Figure 7.27, we delete the automatically generated level controller and associated control signal connections. We change the run mode to Initialization and perform an initialization run.

In Figure 7.27, we also show the “Add task” button within the Flowsheet folder. We demonstrate below the following tasks:

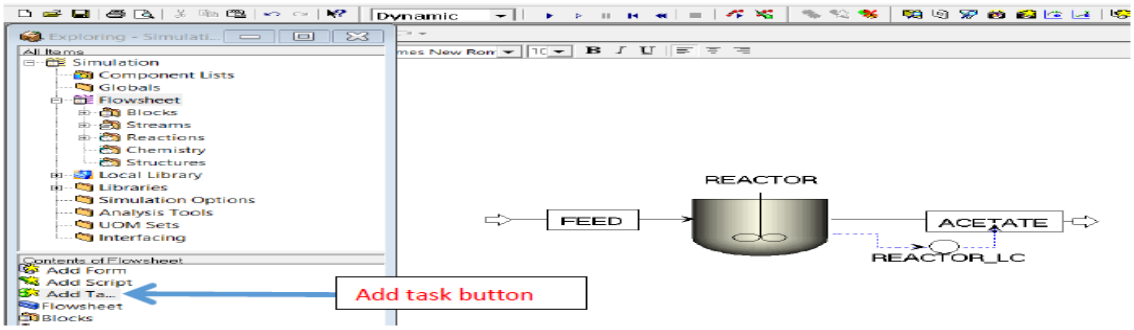


Figure 7.27 The exported dynamic simulation flowsheet with an automatically generated level controller and the “Add task” button within Flowsheet folder

Task A: Start the sequence at time 0.01 hr
Task B: Shut off the feed
Task C: Drain the vessel by emptying the product at a rate of 60 kg/hr until the reactor liquid level falls below 0.1 m.
Task D: Shut off the ethyl acetate product liquor
Task E: Charge the reactor with feed at a mass flow rate of 4000 kg/hr for a period of 0.7 hr. Start by ramping the feed flow to 4000 kg/hr over a period of 0.1 hr.
Task F: Shut off the feed.
Task G: Wait for the concentration of ethyl acetate in the reactor to become greater than 7 kmol/m^3.
Task H: Set up a task to call all the preceding subtasks.

(1) Task A: Run the dynamic simulation and ask to AD to pause at 0.01 hr.
(2) Task B: Following Figure 7.27, click “Add Task” button within Contents of Flowsheet to create the task “ShutoffFeed” from Flowsheet window. As wee see in Figure 7.28b, we have AD print to the message window that the task has started; set feed molar flow rate (FmR), that is, (STREAMS("FEED").FmR), to zero; and print out to message that the task has ended. At the end of defining the task “ShutoffFeed” or any task within AD, we compile the task script to detect any errors. Referring to Figure 7.28b, we do this by right-clicking the mouse inside the task windown to bring up the “compile” tab. Clicking on the tab to compile the tash script. Note in Figure 7.28b, line 1 gives the task name, and lines 2 to 9 give the standard instructions for tasks within AD. To save figure space, we will not display lines 1 to 9 in Figures 7.29 to 7.33 defining tasks C to H.

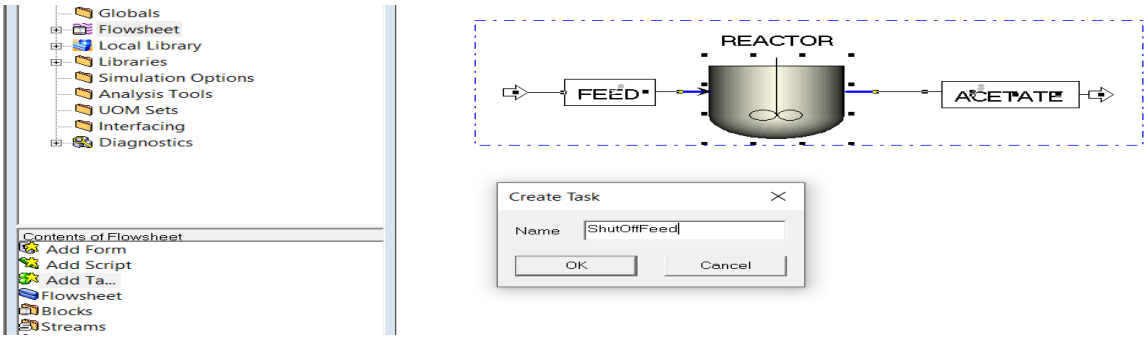


Figure 7.28a Click “Add Task” button within Contents of Flowsheet to create the task “ShutOffFeed”.

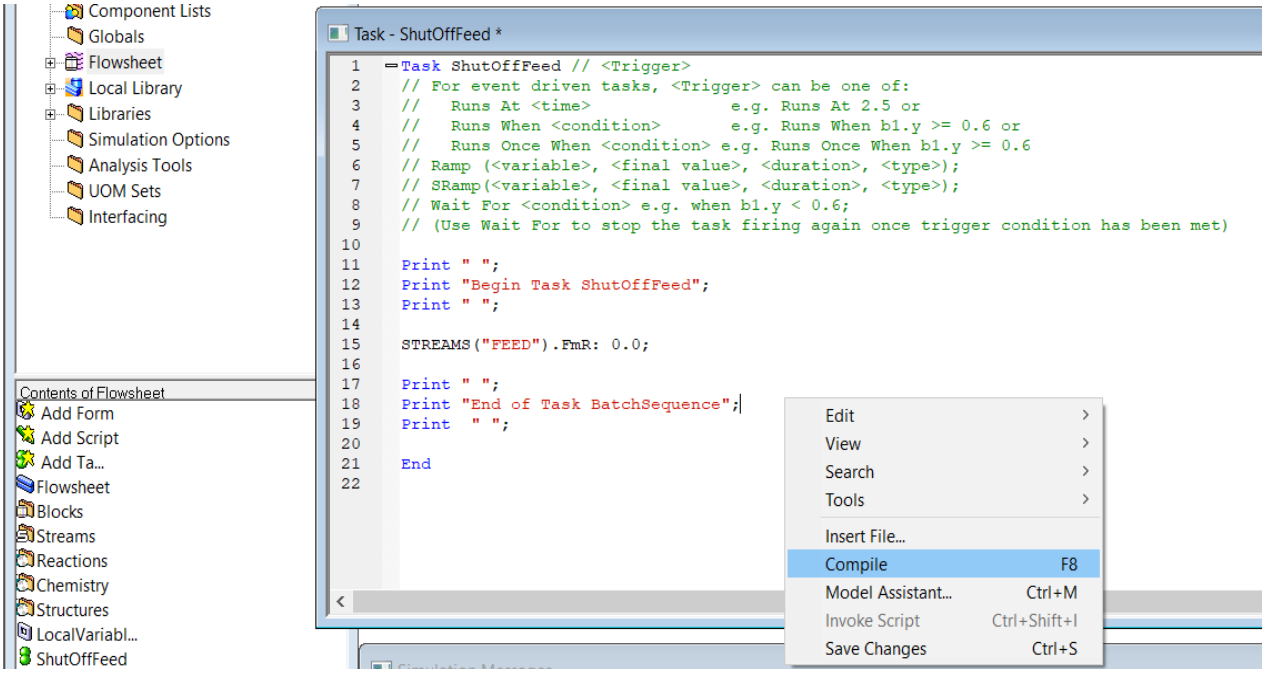


Figure 7.28b Content of task ShutOffFeed.

(3) Task C: Create the task “DrainVessel” from Flowsheet window; print out to messages when the task starts; set the Acetate mass flow rate (FR), that is, (Streams("Acetate").FR), to 60 kg/hr; wait for reactor level (Blocks(“Reactor”).Level) to reach less than or equal to 0.1 m; and print out to messages when the task ends. See Figure 7.29.

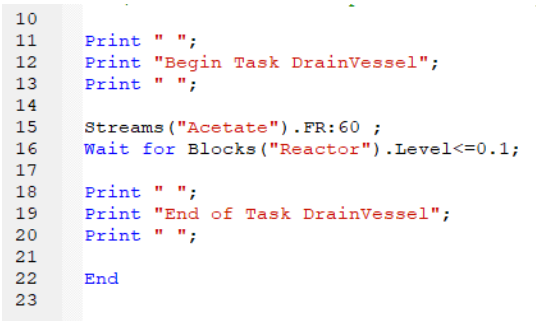


Figure 7.29 Content of task “DrainVessel”.

(4) Task D: Create the task “ShutOffProduct” from the Flowsheet window; print out to messages when the task starts; RAMP changes Acetate mass flow rate (FR), that is, (Streams(“Acetate”).FR), to 0 linearly over a period of 0.1 time units; print out to messages when the task ends. See Figure 7.30.

```
10
11    Print " ";
12    Print "Begin Task ShutOffProduct";
13    Print " ";
14
15    RAMP(Streams("Acetate").FR, 0.0, 0.1);
16
17    Print " ";
18    Print "End of Task ShutOffProduct";
19    Print " ";
20    End
21
```

Figure 7.30 Content of task “ShutOffProduct”.

RAMP is a simple linear ramp function. Its function syntax, RAMP (Variable, Target, Period, Ramp type), says the following:

- Variable: The name of the variable whose value is to be ramped. The variable must have a specification of fixed.
- Target: The value to which the variable’s value is to be changed.
- Period: The period over which the ramp takes place.
- Ramp type: Can be either DISCRETE or CONTINUOUS. CONTINUOUS ramps are re-calculated whenever the integrator moves its time. DISCRETE ramps are re-calculated at communication points only. The ramp type may be omitted; in which case the default ramp type of CONTINUOUS is used.

(5) Tasks E and F: Create the task “Charge Reactor” from the Flowsheet window; print out to messages when the task starts; define “Time_next” as a real parameter (Time_next as RealParameter); set Time_next to TIME + 0.7 (Time_next: TIME + 0.7); SRAMP changes feed stream mass flow rate (Streams(“Feed”).FmR) with the shape of a sinusoidal curve to 4,000 kg/hr over an interval of 0.1 time units; wait for Time to become greater than or equal to Time_next; “Shut off the Feed Stream Flow” (task F); and print out to messages when the task ends. See Figure 7.31.

```
10
11    Print " ";
12    Print "Begin Task ChargeReactor";
13    Print " ";
14
15    Time_next as RealParameter;
16    Time_next: TIME + 0.7;
17    SRAMP(Streams("Feed").FmR,4000,0.1);
18    Wait for Time >= Time_next;
19    Streams("Feed").FmR:0.0;
20
21    Print " ";
22    Print "End of ChargeReactor";
23    Print " ";
24
25    End
26
```

Figure 7.31 Content of tasks “ChargeReactor” and “Shut off Feed”.

SRAMP is a sinusoidal ramp, which gives a smoother S-shaped curve for the ramped variable. With SRAMP, if the ramp is decreasing the value of a variable, then the ramp follows the shape of a sinusoidal curve between 0 and π (=3.1416). If the target is an increase, then the curve follows the shape of a sinusoidal curve between π and 2π. SRAMP’s function syntax follows that of RAMP described above.

(6) Task G: Create the task “AgeReactor” in the Flowsheet window; print out to messages when the task starts; wait for the reactor molar concentration of acetate, Blocks(“Reactor”). conc_mol(“acetate”), to reach 7.0 kmol/hr; print out to messages when the task ends. See Figure 7.32.

```
10
11    Print " ";
12    Print "Begin Task AgeReactor";
13    Print " ";
14
15    Wait For Blocks("Reactor").conc_mol("ETOAc") >= 7.0;
16
17    Print " ";
18    Print "End of Task AgeReactor";
19    Print " ";
20    End
21
```

Figure 7.32 Content of task “Age Reactor”.

(7) Task H: Create the task “BatchSequenceRuns” in the Flowsheet window in order to call the subtasks that we have just created. See Figure 7.33.

```
10
11    Runs At 0.01;
12    Call ShutOffFeed;
13    Call DrainVessel;
14    Call ShutOffProduct;
15    Call ChargeReactor;
16    Call ShutOffFeed;
17    Call AgeReactor;
18    END
19
```

Figure 7.33 Content of task “BatchSequenceRuns”.

Next, we create three plots to observe: (1) reactor liquid volume, liquid level, and temperature; (2) reactants and products concentrations in the reactor – use one axis for the concentrations; and (3) feed and product stream mass flows - use one axis for both variables.

Open AllVariables table for Acetate stream. Change mole flow rate FR to fixed and mass flow rate FmR to free. Run initialization, Steady-state, and then dynamic simulations to pause at 10 hr. Before running the dynamic simulation, we must *activate* the task, “BatchSequenceRuns”, that controls all the defined sequencing tasks by right-clicking on the task name within Contents of Flowsheet. Figures 7.34 illustrates the results. Note that to see the distict drop inlevel from the drain step and then the rise in leve from the charge step, we must re-scale the x-coordinate to show in smaller time steps.

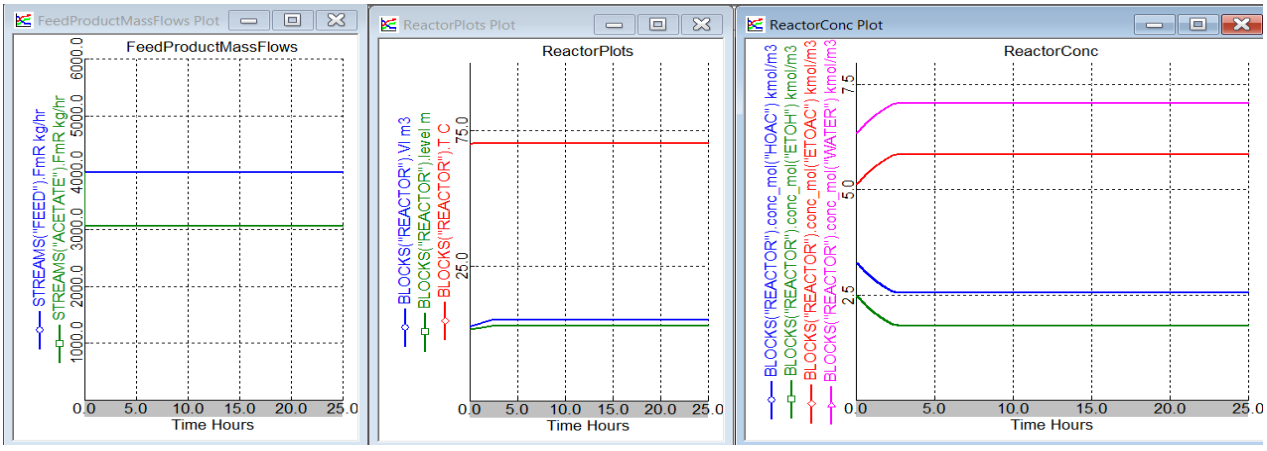


Figure 7.34 An illustration of results from task workshop, WS7.3

7.6 WS7.4 Dynamic Simulation and Grade Change of a Slurry HPDE Process

7.6.1 Objectives

We wish to demonstrate how to convert a steady-state simulation model to a dynamic simulation model, illustrate the initial adjustments of the dynamic model and the tests of the validity of the default control schemes. We then show how to use a task procedure to implement grade-change operations applied to HDPE productions.

7.6.2 Stepwise Procedure to Develop Aspen Plus Dynamics (AD) Simulation Model

We begin with a steady-state simulation model of a commercial slurry HDPE process developed by Aspen Plus (which include Aspen Polymers) in Chapter 5 and present a step-by-step dynamic simulation of adding various controllers and making grade changes in Aspen Plus Dynamic (AD). Reference [3] includes some brief details of this application.

Step 1. Making the steady-state simulation model ready for conversion to a dynamic simulation model.

We open a steady-state simulation model, ***WS7.4.bkp***, representing the reactors and separators in a serial-flow slurry HDPE process using multi-site Ziegler-Natta kinetics [3]. Figure 7.35 shows a steady-state simulation flowsheet, in which M1 and M2 are mixers, R1 and R2 are reactors, F1 and F2 are flash units, C1 and C2 are compressors, E1 and E2 are exchangers, and S1 and S2 are splitters. The reader may see the inputs for steady-state simulation from the Aspen Plus simulation file.

For the continuous stirred tank reactors R1 and R2, we can use the RCSTR model with one vapor product stream and one liquid product stream (as in RCSTR2 in Figure 3.1). For readers with process simulation models developed with early versions of Aspen Plus, the RCSTR model could have only one exiting product stream. In that situation, we can use an alternative representation of our vapor and liquid product streams from the reactor. Specifically, we use flash units F1 and F2 following reactors R1 and R2 to generate the vapor products (V1 and V2) and liquid products (MXD and PROD), with both the CSTR and flash units being specified at the same temperature and pressure.

In this text, the reader will see again this alternative representation of a RCSTR model with both vapor product and liquid product streams by a combination of a RCSTR model with a single product stream followed by a two-phase flash model.

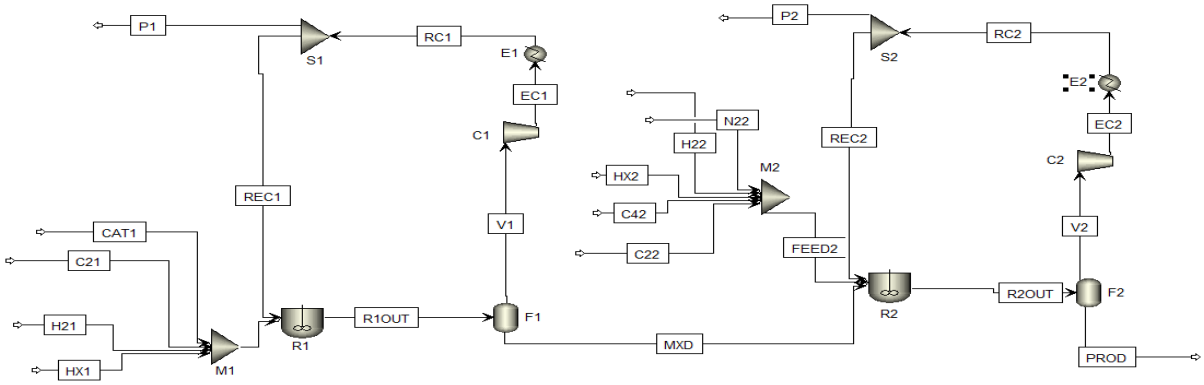


Figure 7.35 A steady-state simulation flowsheet for ***WS7.4***

Our first task in preparing an Aspen Plus model for conversion into a dynamic model is to minimize the number of unnecessary computations. Unnecessary computations appear in three places: (1) unnecessary components; (2) extraneous phase calculations; and (3) extraneous unit-operation models [1].

Unnecessary components are those that appear in trace quantities and do not affect the major results of the simulation. When AD perturbs these variables, small amounts of the components with zero flow rate enter the system. If the system is sensitive to their presence, this can cause problems. By checking the component mass flow and mass fraction in the stream summary of the steady-state simulation results, we do not find such components in the steady-state model.

We examine the resulting vapor fraction (VFRAC) for each feed stream in the steady-state model. If a feed stream has a single phase only, we specify that stream as single phase in the flash options. This avoids singularities that result from flashing single-phase streams. We also check the outlet streams for each unit operation. If a stream is a single phase, we alter the flash options for the block so that there is no vapor-liquid check.

Extraneous unit-operation models are those that we can easily combine into other unit operations. In the current steady-state simulation, we do not have such models.

Our next task is to complete the dynamic simulation specificationsf or the reactors and other unit-operation blocks. We enter the data in Table 7.13.

Table 7.13 Dynamic specifications for ***WS7.4***

Block	Specifications
F1 and F2	Instantaneous flash with no dynamic effects
C1 and C2	Instantaneous compression
R1 and R2	Vertical vessel, elliptical, length =6.678 m, heat-transfer option - constant duty, initial condition - liquid volume fraction = 0.5
E1 and E2	Instantaneous heat transfer, heat-transfer option-constant duty
M1 and M2	Instantaneous mixing

Step 2. Converting to a dynamic model

We click the Dynamic tab on the ribbon, click the Flow Driven button to export the simulation from Aspen Plus to AD, and save the dynamic simulation as ***WS7.4.dynf***, and the corresponding property file as ***WS7.4dyn.appdf*** in our AD working folder. Figure 7.36 shows the resulting dynamic simulation flowsheet in AD.

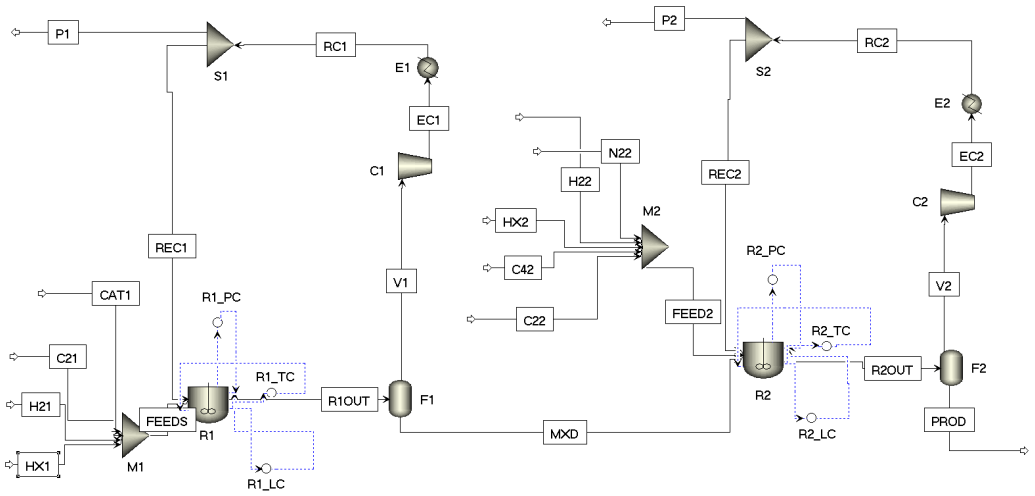


Figure 7.36 The dynamic simulation flowsheet of ***WS7.4***

Step 3. Initial Adjustments to the AD Model

Our next step is to make initial adjustments to the AD model, focusing on: (1) rigorous property option; (2) polymer attributes for streams and blocks; (3) heat duties and temperatures; (4) calculation of derived polymer attributes; and (5) revising the control scheme setup.

(1) Rigorous Property Option

Using rigorous properties is generally slower, but much more robust than local properties, especially when working with polymer systems. Follow the following path to change the property option: Explorer -> Global -> Dynamic Options -> (1) Global Property Mode: Rigorous; and (2) Global Flash Basis: Equation. See Figure 7.37.

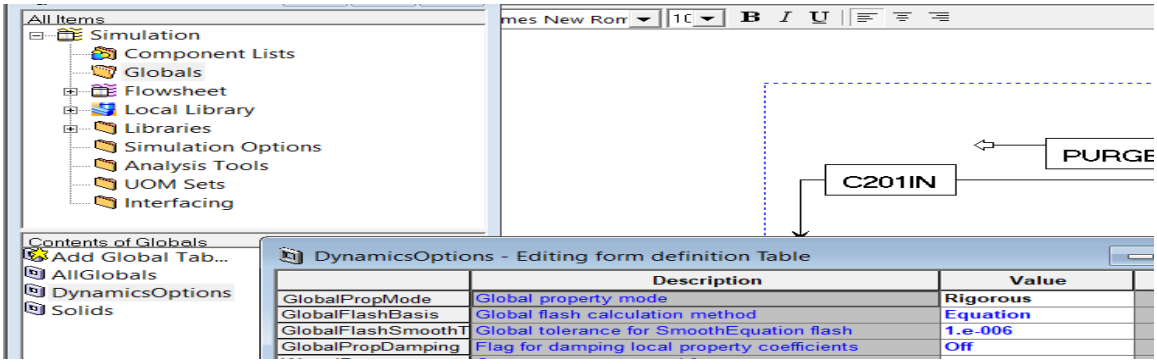


Figure 7.37. Specifying global property mode and flash basis.

(2) Polymer Attributes for Streams and Blocks

If there are trace amounts of catalyst or polymer in the overhead streams, AD will make the stream a polymer stream. This means that the equations involving component attributes are included for the stream. This is a problem when a stream contains a trace amount of polymer or catalyst, which is a numerical artifact. Figure 7.38 shows how we can change the stream type to eliminate this problem (Flash unit F1-> overhead stream V1 -> Right-click Forms -> PolymerInputs table -> Change the Value of

Catalyst_in_Stream from Yes to No.). We note that compressor C1, to which the overhead vapor V1 enters, does not support polymers. During converting from a steady-state to dynamic simulation file, AD has automatically dropped catalyst attributes. We repeat the same step with overhead stream V2 leaving flash unit F2.

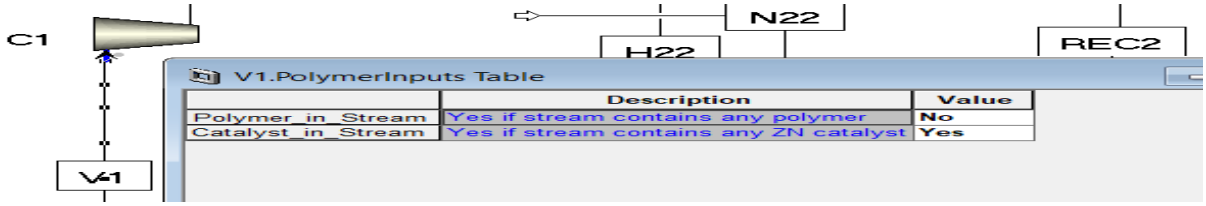


Figure 7.38 Changing the Value of Catalyst_in_Stream in stream V1 from Yes to No.

Once we make these adjustments to all the streams not containing catalyst or polymer, the model should be 'square', meaning that the number of fixed and free variables corresponds appropriately to the number of equations in the model. We run in Initialization mode and then run in Steady State mode. We save each one as a snapshot for future use. It is important to make steady-state runs and save snapshots after making each change to the dynamic model. If we make many changes at once, the system may not converge.

(3) Heat Duties and Temperatures

We must review each unit operation to see if AD has fixed the heat duty (Qr), and to make adjustments if necessary. We should not fix the heat duties for heat exchangers because we do not know what they are. Instead, we should fix the temperatures and leave heat duty as a fee variable.

(4) Calculation of Derived Polymer Attributes

For the polymer product stream exiting reactor R2, we must ask AD to calculate the relevant polymer attributes, such as MWW, PDI, etc. To do this, right-click the stream name, R2OUT, choose Forms, and then PolymerResults. Change "Calculate derived polymer attributes" from No to Yes.

(5) Revising Control Scheme Setup

We save the current dynamic simulation file as ***WS7.4Original.dynf*** as a backup before we continue and modify the control schemes in the simulation file ***WS7.4.dynf***. This flowsheet includes three default controllers for pressure, level and temperature specified previously in Table 7.9. These default controllers are not necessarily valid. We must add and/or delete controllers to match the control scheme in the actual process. Table 7.14 gives the initial specifications of the default controllers.

Table 7.14 Initial Specifications of Default Controllers for ***WS7.4.dynf***

Controller	***Measured Variable (Process variable PV or controlled variable)***	***Manipulated variable (Controller output OP)***
R1_PC and R2_PC	Reactor pressure: R1 at 7.55 bar; R2 at 3.5 bar	Specified vapor flow rate: R1 at 19.1579 kmol/hr (same flow rate as V1 in Figure 7.37); R2 at 628.3473 kmol/hr (as flow rate as V2)

R1_TC and R2_TC	Reactor temperature: R1 at 85°C; R2 at 80°C	Specified heat duty: R1 at -19.8786 GJ/hr; R2 at -16.8558 GJ/hr
R1_LC and R2_LC	Reactor liquid level: R1 at 4.8287 m; R2 at 4.8287 m	Specified liquid mass flow rate: R1 at 20914.4 kg/hr (same flow rate as MXD in Figure 7.37); R2 at 26222.7 kg/hr (same flow rate as PROD)

These tabulated values match the values in the initial controller faceplates in Figure 7.39.

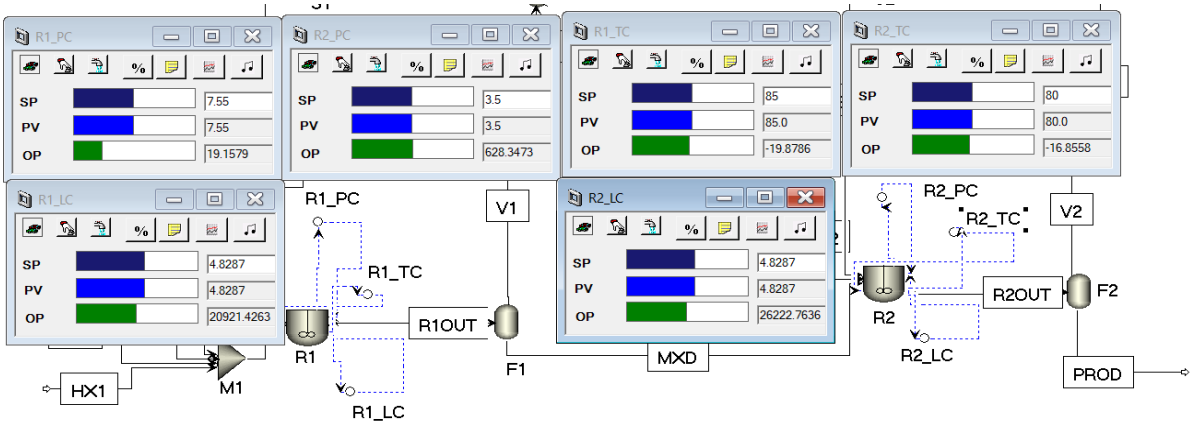


Figure 7.39 Initial controller faceplates

The default pressure controllers on the reactors R1 and R2 are *not* valid. Each controller maintains the reactor pressure by increasing the vapor flow rate out of the reactor (that is, vapor flow rate of streams V1 and V2 in Figure 7.36). This only increases the flow rate of material recycled to the reactor, resulting in an increase in reactor pressure. In addition, the actual reactor in the plant does not contain a pressure controller. Therefore, we delete both pressure controllers and their control signals.

We follow Figure 7.21 to initialize values of the controllers, and to revise the proportional gain and integral time for the temperature and level controllers according to Table 7.8. We save the resulting flowsheet with controllers as ***WS7.4a.dynf***.

Next, we wish to test the resulting temperature and level controller in response to an increase in ethylene mass flow rate. Specifically, we re-save ***WS7.4a.dynf*** as ***WS7.4b.dynf***, and use the latter file for our controller testing. We make initialization, steady-state and dynamic runs of ***WS7.4b.dynf*** to pause at 10 hr and then increase the ethylene mass flow to both reactors R1 and R2: Stream C21->Right-click Forms->Manipulate -> Change FmR (specified total mass flow) from 5700 to 7000 kg/hr; do the same for Stream C22 and change the mass flow from 5400 to 7000 kg/hr. Run the dynamic simulation again until 30 hr. Follow Figure 7.21 to display the resulting plots for the temperature and level controller.

Figure 7.40 shows that after increasing the ethylene feed mass flow for both reactors at 10 hr, the temperature and level controllers for both reactors respond quickly to return to the setpoints.

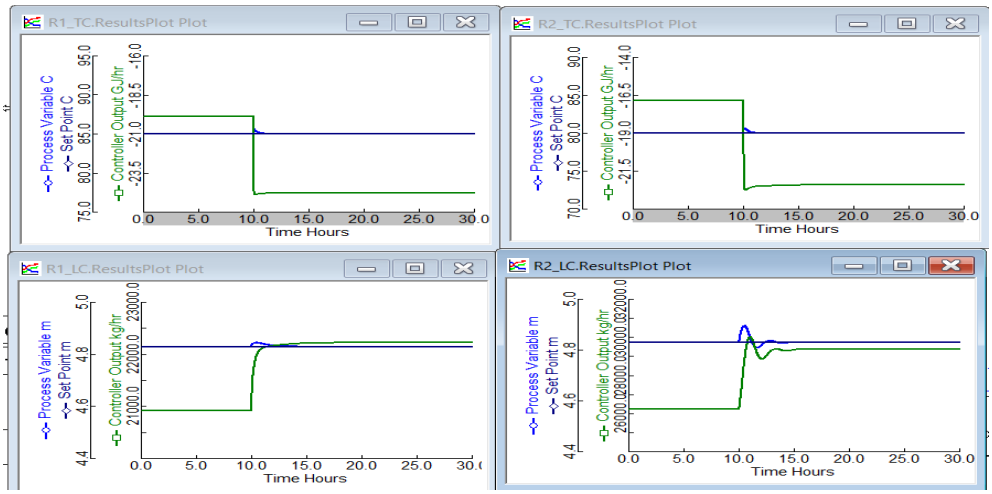


Figure 7.40 Temperature and level controllers for both reactors R1 and R2 respond quickly to return to the setpoints after an increase in ethylene mass flow at 10 hr.

We can fine-tune the controller parameters by following the following heuristics:

(1) When we make a small step change in the setpoint for process variable for a particular controller, if a dynamic run approaches the setpoint, but does not flatten out at the setpoint value, we increase the integral time for the controller.
(2) If the dynamic run meets the setpoint, but takes too long to do so, we increase the proportional gain for the controller. If the controller displays unstable behavior (not settling to a steady state), we decrease the gain.
(3) Iterate between these two adjustments until a sufficiently tight control of the process variable occurs.

(4) Note that bad performance by one controller can affect the performance of other controllers. We should simultaneously monitor the control plots for all controllers when adjusting parameters for each one individually.

7.6.3 Simulation of Grade-Change Operations

Table 7.15 summarizes the key process and quality variables for ***WS7.4***.

Table 7.15 Process and quality variables for ***WS7.4***

Process variable	**Description**
C21	Ethylene monomer flow in the feed to the first reactor (kg/hr)
H21	Hydrogen flow in the feed to the first reactor (kg/hr)
CAT	Catalyst flow to the first reactor (kg/hr)
HX1	Solvent (n-hexane) flow to the first reactor (kg/hr)
C42	1-Butene co-monomer flow to the second reactor (kg/hr)
C22	Ethylene monomer flow to the second reactor (kg/hr)
H22	Hydrogen flow to the second reactor (kg/hr)
HX2	Solvent (n-hexane) flow to the second reactor (kg/hr)

Quality variable	Description
MWW	Weight-average molecular weight of polymer in outlet stream
SFRAC	Co-monomer fraction in outlet polymer
Rate_pol	Polymer flow rate in outlet stream

Table 7.16 summarizes the operational details of four HDPE grades that we wish to produce.

Table 7.16 Values of process variables for grades 1 to 4

	Current	**Grade 1**	**Grade 2**	**Grade 3**	**Grade 4**
Tasks G1 to G4 run at	@0 hr	G1 @5 hr	G2 @40 hr	G3 @80 hr	G4 @120 hr
H21, kg/hr	8	4	4	10	10
H22, kg/hr	1	0.5	0.5	0.75	0.75
C42, kg/hr	1000	1000	750	750	900

For the purpose of demonstrating grade change for this case, we only vary the Hydrogen flow (H21, H22) and the Comonomer flow(C42) here, keeping rest of the variables same. Following the illustration of Figure 7.27 to Figure 7.34, we define tasks G1 to G4 under Flowsheet -> Contents of Flowsheet -> Add Task. Figure 7.41 shows the details of task G1. Here, we use the SRAMP function explained previously with Figure 7.31. Basically, SRAMP(Streams("H21"). FmR,4,4) changes the mass flow rate of stream H21 according to the shape of a sinusoidal curve to 4 kg/hr over an interval of 4 time units.

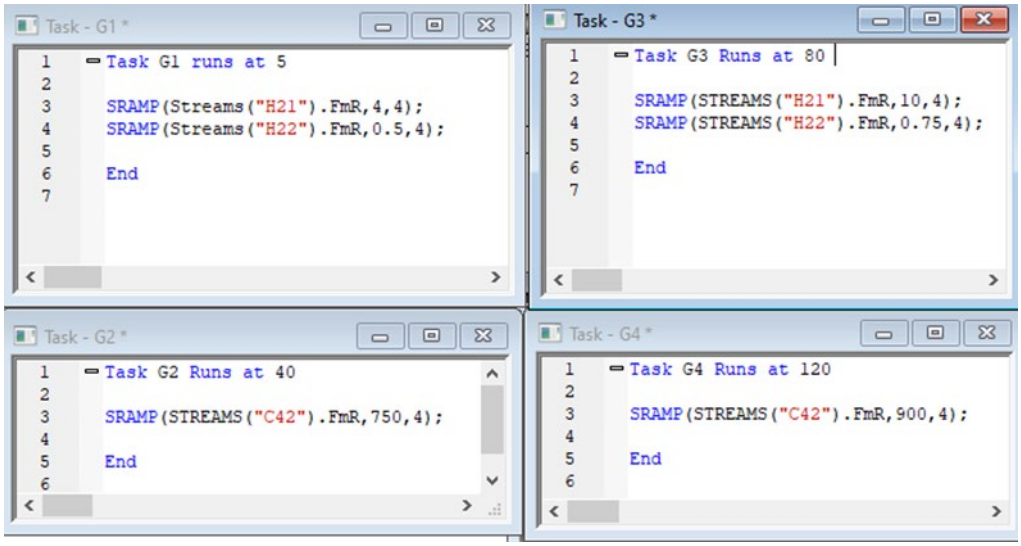


Figure 7.41 Specification of task G1 to run at 24 hr.

Following Figure 7.41 and Table 7.16, we complete the specifications of tasks G2 to G4 in the same way. We compile all four tasks and find no errors.

We create the plots for calculated process quality variables, such as melt index and copolymer density, based on empirical correlations suggested by Sinclair [8] that we used in Eq. (5.16) and (5.17). Specifically, we correlate plant data for melt index (MI) as a function of the weight-average molecular weight (MWW) of the HDPE product according to the equation:

$$MI = A\_MI\left(\frac{MWW}{C\_MI}\right)^{B\_MI} \quad (7.2)$$

For illustrative purposes, we assume A_MI = 901, B_MI= -5, C_MI = 1e5. When plant data for MI are available, the reader can regress new values of A_MI and B_MI. Additionally, we correlate the copolymer density (DENSITY) in kg/m^3 by the equation:

$$\text{DENSITY}/1000 = 0.996 - \text{A\_DN}(\text{SFRAC\_Comonomer})^{\text{B\_DN}} \quad (7.3)$$

where SFRAC_Comonomer is the mole fraction of segments of the comonomer, butene (C_4H_8), and the assumed correlation parameters A_DN = 0.02386 and B_DN = 0.514.

To implement these correlations in the simulation, we right-click Flowsheet within Contents of Flowsheet and choose Edit to open the Constraints-Flowsheet window, and enter the correlations following Figure 7.42. Be sure to compile the equations entered following the example of Figure 7.28b to ensure that there is no coding error.

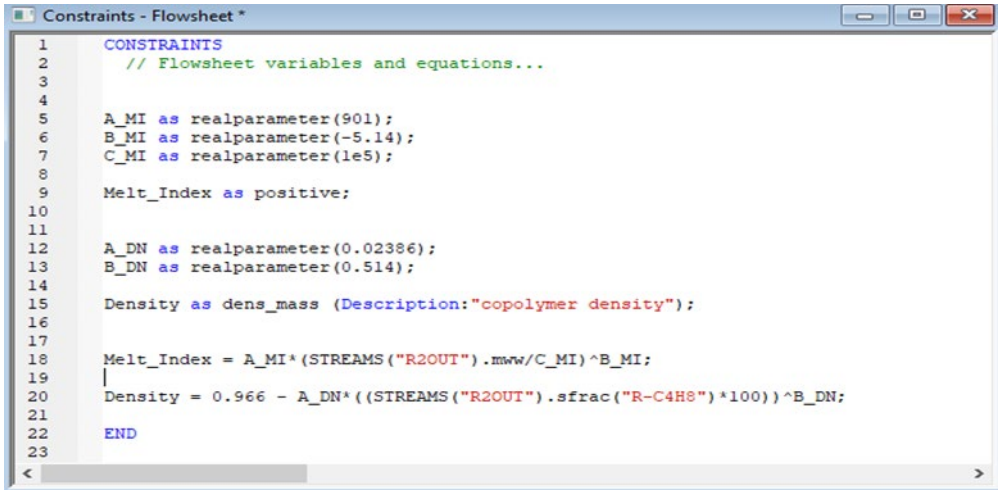


Figure 7.42 Specification of the flowsheet constraints, MI and copolymer density correlations.

After compiling the flowsheet constraint equations, we right-click Local Variables within Contents of Flowsheet to see the initially calculated values of Density and Melt_Index in the LocalVariables Table. We follow the example of Figure 7.14 to set up a plot named MI_Density. See Figure 7.43.

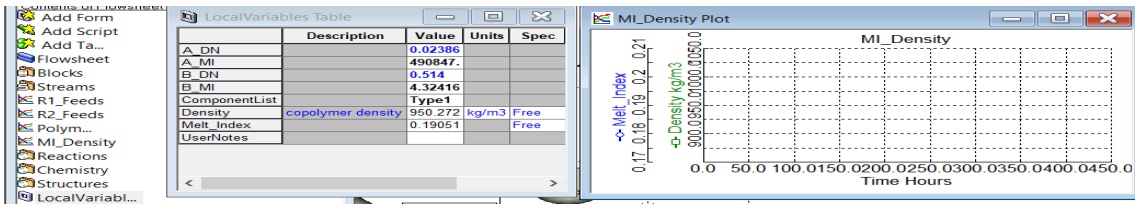


Figure 7.43 Setting up a figure of calculated melt index and copolymer density.

Next, we create three plots to display: (1) R1_Feeds: mass flow rates (FmR), kg/hr, of H21 feed to reactor R1; (2) R2_Feeds: mass flow rates (FmR), kg/hr, of C42, H22 feeds to reactor R2; and (3) Polymer: polymer production rate (Rate_pol), kg/hr; weight-average molecular weight, MWW, kg/kmol; and. To display each y-axis variable on its own y-axis scale, we put the mouse in the middle of the plot, right-click to show "properties", and choose AxisMap, followed by "One for Each" indicating one y-axis scale for each variable. See Figures 7.44.

We make initialization and steady-state runs, and then make a dynamic simulation to pause at 150 hr. Figure 7.44 shows the resulting plots of R1_Feeds, R2_Feeds, and Polymer. Before running the dynamic simulation, it is important to *activate* all tasks G1 to G4 by right-clicking on the task name within Contents of Flowsheet. At the end of 150 hr, if you do not see the result plot spanned over the time range of 0 to 150 hr, then right-click on the plot name (e.g., R1_Feeds) within Contents of Flowsheet, and choose "Show as Plot". You will then see the complete results as plotted in Figure 7.44. You can also see the evolution of the computed melt index and copolymer density in Figure 7.45.

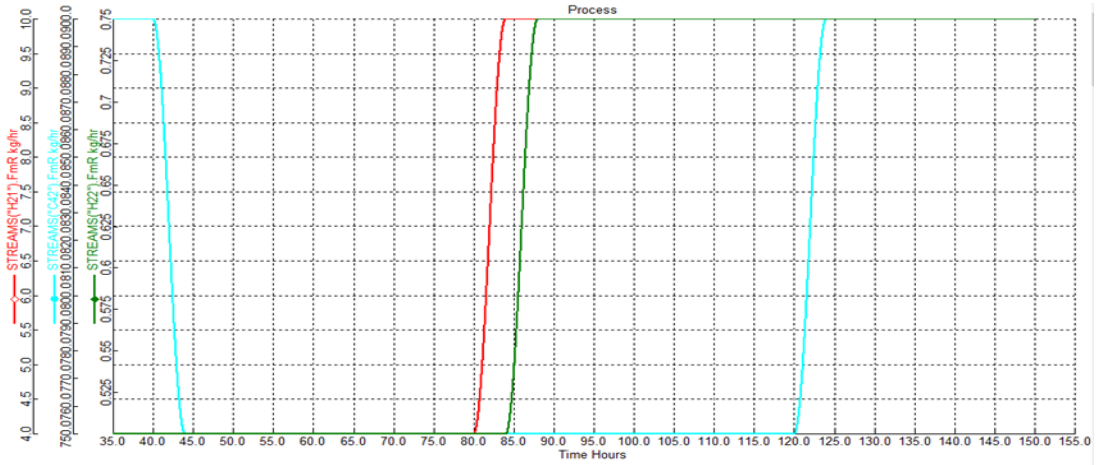


Figure 7.44 Evolution of feed mass flow rates for producing grades G1 to G4.

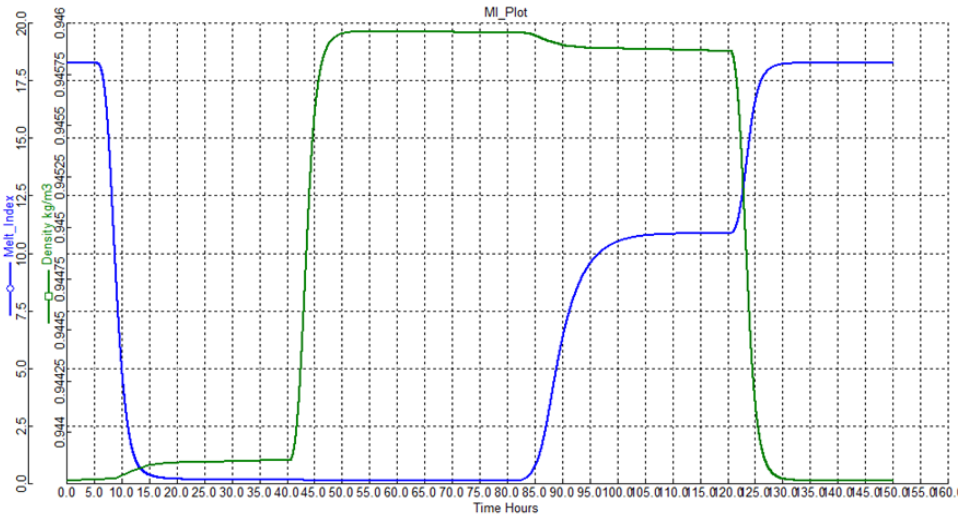


Figure 7.45 Evolution of the computed melt index and copolymer density.

7.7 WS7.5 Dynamic Simulation and Control of a Commercial Slurry HDPE Process

7.7.1 Objectives

We wish to demonstrate how to implement various control schemes in a commercial slurry HDPE process. We introduce the use of flowsheet variables and constraints, and ratio block in implementing the actual controller configuration in a commercial process.

7.7.2 Converting a Steady-State Simulation Model to a Dynamic Simulation Model

We open a steady-state simulation model, ***WS7.5.bkp***, representing a set of reactor and separators in a parallel-flow slurry HDPE process [23]. For simplicity, we use single-site kinetics in developing the steady-state simulation file in this example. This assumption does not affect our procedure for dynamic simulation and control. Figure 7.47 shows a steady-state simulation flowsheet, in which M1 is a mixer, D201 is a reactor, 201F and D205 are flash units, C201 is a compressor, E201 is an exchanger, P202 is a pump, and S1 and S2 are flow splitters. The reader may see the inputs for steady-state simulation from the Aspen Plus simulation file.

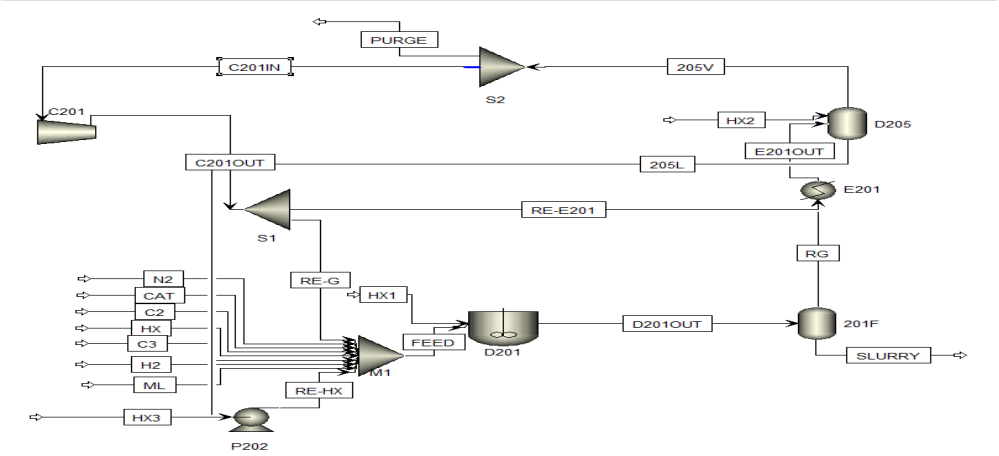


Figure 7.47 A steady-state simulation flowsheet for ***WS7.5***

Table 7.17 shows the dynamic data for ***WS7.5.***

Table 7.17 Dynamic specifications for ***WS7.5***

Block	Specifications
201F	Instantaneous flash with no dynamic effects
D205	Vertical vessel, elliptical, length =4.85 m, diameter = 2.5 m, heat-transfer option - constant duty, initial condition - liquid volume fraction = 0.5
C201	Instantaneous compression
D201	Vertical vessel, elliptical, length =6.678 m, heat-transfer option - constant duty, initial condition - liquid volume fraction = 0.5

E201 and E2	Instantaneous heat transfer, heat-transfer option-constant duty
M1	Instantaneous mixing

We complete the dynamic data according to Table 7.17. After running the steady-state simulation, we export the simulation from Aspen Plus to AD, and save the resulting dynamic simulation file, ***WS7.5.dynf***, and the Aspen Property problem definition file, ***WS7.5dyn.appdf***. Figure 7.47 shows the resulting flowsheet with the added default controllers.

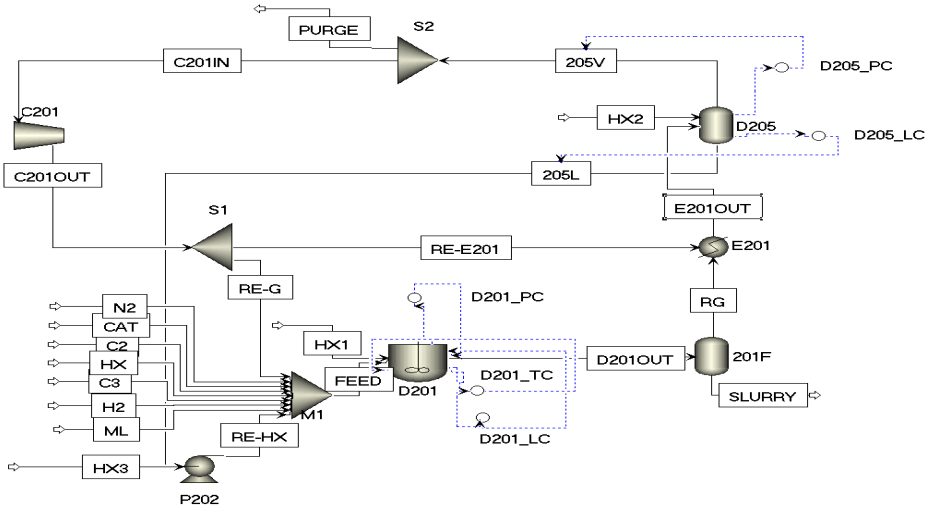


Figure 7.47 The starting dynamic simulation flowsheet for ***WS7.5.dynf***

7.7.3 Initial Adjustments of the AD Model

(1) Polymer Attributes for Streams and Blocks

We follow the path: Flash unit 201F -> Overhead stream RG -> Right-click Forms -> We do not see "PolymerInputs Table", indicating that stream RG does not contain polymer components.

(2) Implementation of Reactor Level Control Using Mechanical Weir

The reactor D201 has no level controller, but it has a weir. We delete the default controller D201_LC and two associated control signals. We enter the equations for weir in the flowsheet form (Exploring-Simulation -> Flowsheet -> Contents of Flowsheet -> Flowsheet -> Right-click Edit ->Constraints ->Flowsheet variables and equations), as illustrated in Figure 7.48:

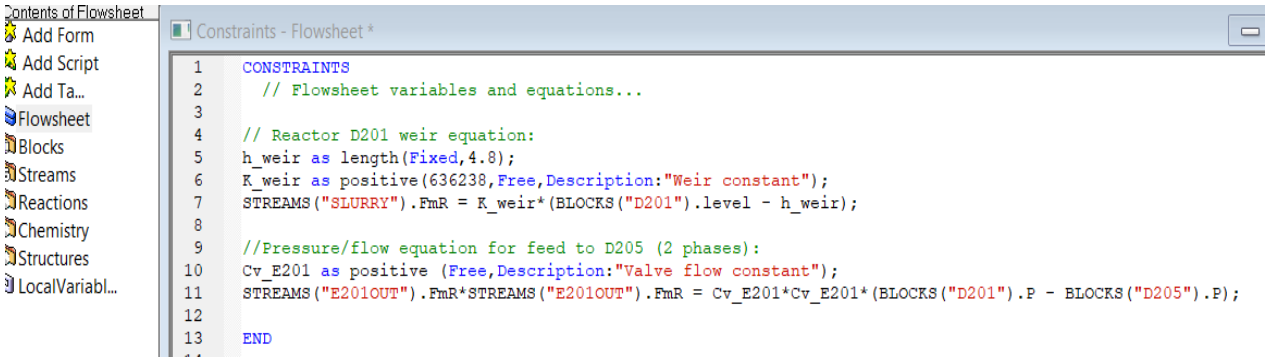


Figure 7.48 Using flowsheet constraints to define the weir variables and equations

We note that the weir constant, K_weir, results from: (1) the current slurry mass flow rate, STREAMS("SLURRY").FmR of 18254.9 kg/hr; (2) D201 liquid level, BLOCKS("D201").level of 4.82869 m; and (3) weir height, h_weir of 4.8 m; and K_weir = 18254.9 kg/hr /(4.82869 – 4.8) m =636,238 kg/hr-m. We will explain the pressure/flow equation with a valve flow constant, Cv_E201 in Figure 7.48 shortly below.

When we compile the flowsheet form after adding the weir equations, we find that the model is over-specified by one variable. We free the weir constant (K_weir) and run the simulation. The computed value for K_weir appears in the Local Variables table within the Contents of Flowsheet. See Figure 7.49.

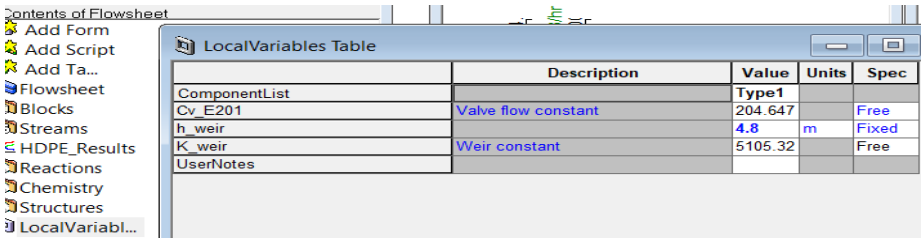


Figure 7.49 Access to Local Variables Table for values of weir constants in flowsheet constraint variables and equations

(3) Improvement of the Reactor Temperature Controller

The reactor has a temperature controller that adjusts the amount of overhead gas exiting the compressor, stream C201OUT, that recycles to the reactor inlet, stream RE-G. The remainder of this stream, stream RE-E201, enters cooler E201 with the vapor outlet of the reactor, stream RG. We reconnect the temperature controller D201_TC to stream splitter S1 and choose the fraction leaving in stream RE-G as the manipulated variable. We double-click the temperature controller, click the Configure button and enter the temperature controller parameters according to Table 7.8. We make the controller Direct because an increase in reactor temperature should result in an increase in the amount of vapor sent back to the reactor via stream RE-G. We initialize the controller, run in steady state, and save the snapshot.

The temperature controller does not operate correctly until we add a Flowsheet Constraint to Figure 7.48 to compute the pressure for the outlet of heat exchanger E201. The equation computes the pressure drop across the heat exchanger using the following relation:

$$\text{Flow}_{\text{E201OUT}} = C_v\sqrt{\Delta P} \tag{7.3}$$

where C_v is a valve flow constant and ΔP is the pressure drop across the heat exchanger. See Figure 7.50. When we compile the flowsheet form, the model is over-specified by one variable. We free the valve constant and make a steady state run. This computes the valve constant. We then fix the valve constant and free the mass flow in stream RG. We save a snapshot of the steady-state run.

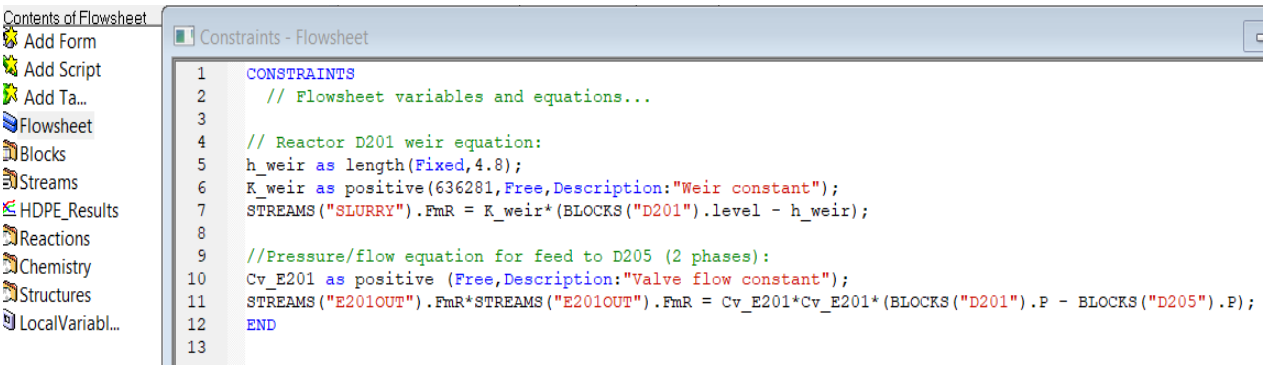


Figure 7.50 Adding the flowsheet constraint to compute pressure drop across exchanger E201

(4) Deletion of Pressure Controllers

We delete pressure controllers, D201_PC andD205_PC and the associated control signals, as the actual plant does not these controllers.

(5) Adding a Hydrogen/Ethylene Ratio Controller to the Recycle Gas

There is a controller that manipulates the feed rate of hydrogen and the purge rate from vessel D205 to maintain the hydrogen/ethylene ratio in the recycle gas. The process variable for this controller is the hydrogen-ethylene ratio in the recycle-gas stream. We must introduce three blocks to the dynamic model when implementing this controller. See Figure 7.51.

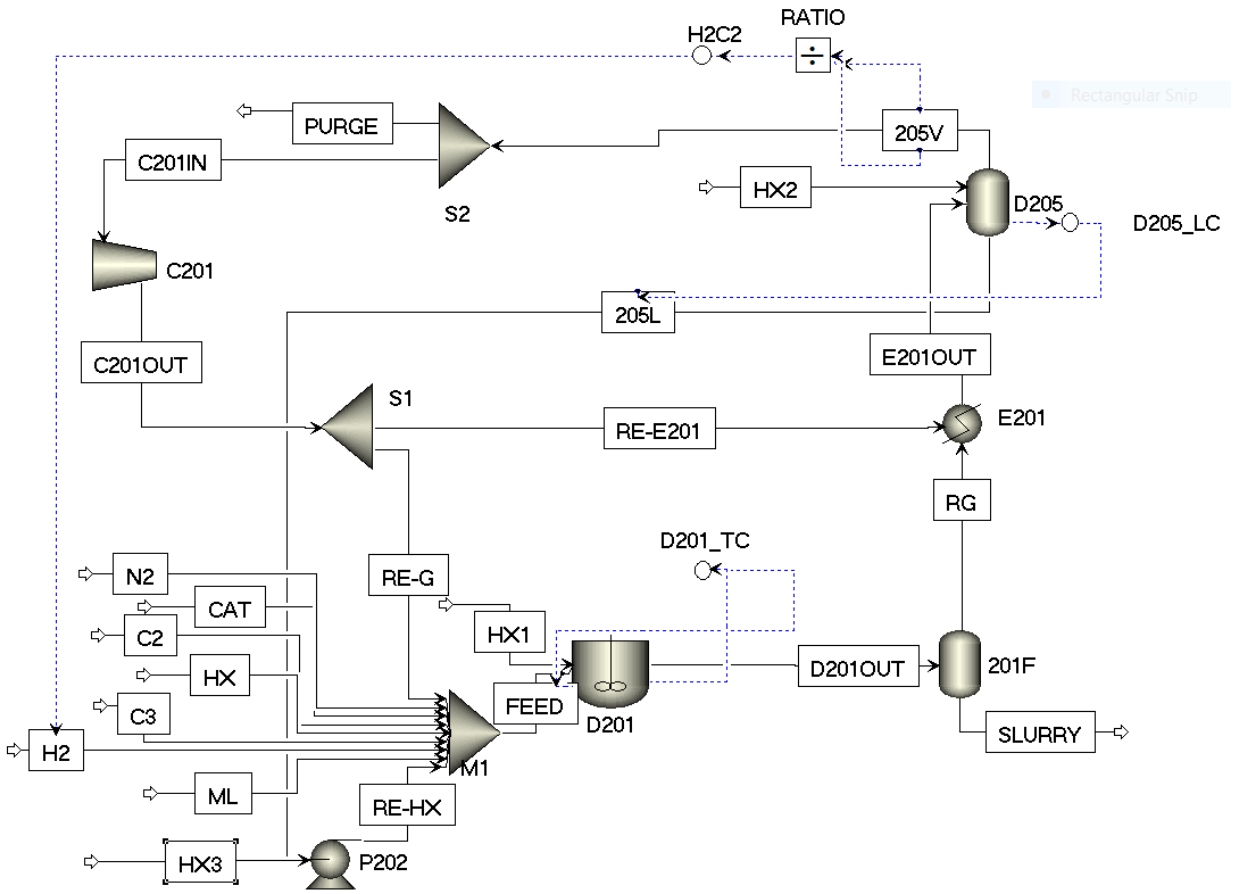


Figure 7.51 Adding a hydrogen/ethylene ratio controller to the recycle gas

The first block is a Ratio block, which computes the ratio of the mole flow of hydrogen, STREAMS("205V"). Fcn("H2") as input 1 to that of ethylene, STREAMS("205V").Fcn("C2H4") as input 2 in the recycle gas, stream 205V. The controller output is the computed mole ratio of hydrogen to ethylene.

The second block is a PID controller that accepts the hydrogen/ethylene ratio from the ratio block. We name the controller as H2C2 and specify its controller action as reverse. We set the proportional gain and integral time according to those for composition controllers in Table 7.8. The PID output signal specifies the mass split fraction of the purge stream exiting the flow splitter S2, BLOCKS("S2").sf("PURGE").

We open the faceplates and plots for controllers H2C2, D205_LC and D201_TC, make initialization and steady-state runs, and then make a dynamic run to pause at 10 hr. Figure 7.52 shows that the current controller appears to work properly.

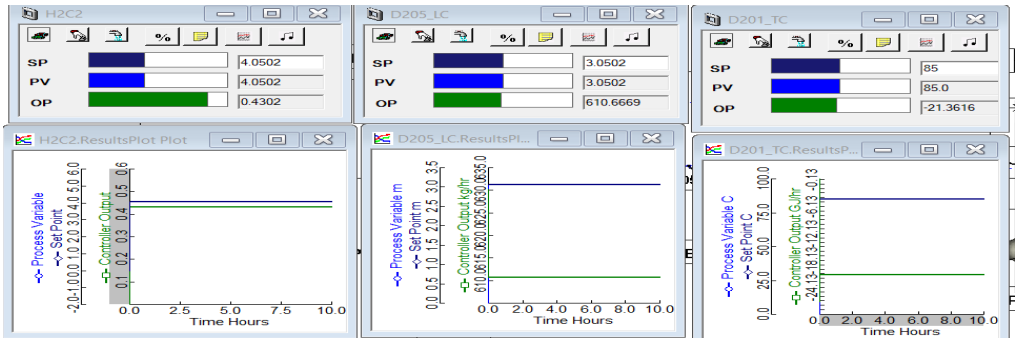


Figure 7.52 Performance of the current ratio, level and temperature controllers

At 10 hr, we increase: (1) the specified ethylene mass flow rate from 5950 kg/hr to 7500 kg/hr (STREAM C2 -> Right-click Forms ->AllVariables ->FmR, specified total mass flow), We then run dynamic simulation to pause at 20 hr. Figure 7.53 shows the performance plots of the three controllers, indicating an acceptable performance.

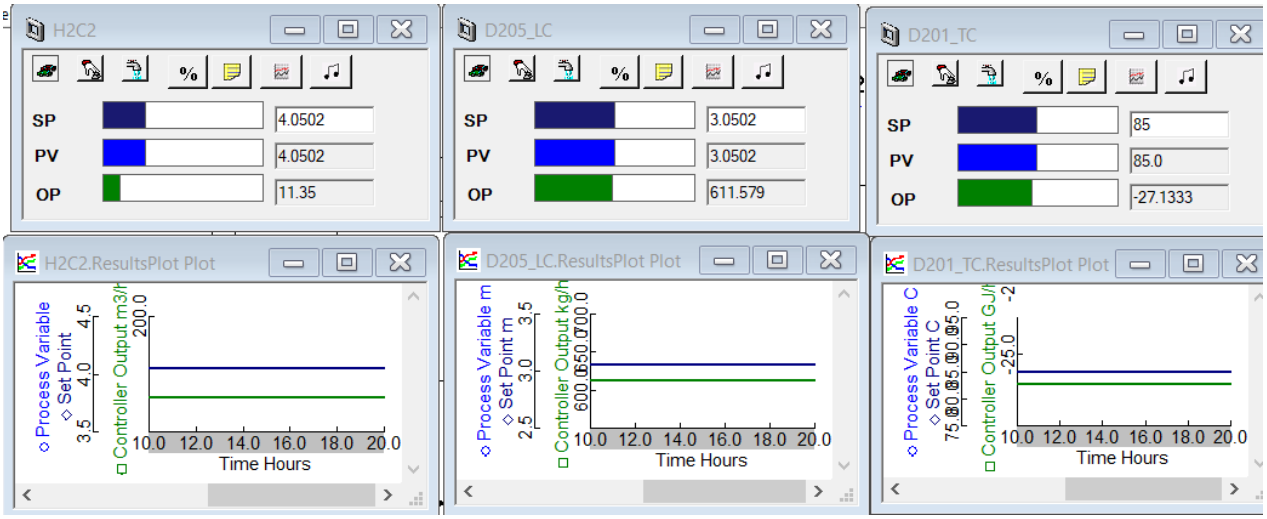


Figure 7.54 Performance of controllers after increasing ethylene mass flow rate to 7500 kg/hr

Now we specifically test the critical PID H2/C2 controller against a change in input ethylene flow rate (STREAMS("C2").FmR. We increase: (1) the flow rate of ethylene from 7500 kg/hr to 9000 kg/hr; (2) the specified catalyst total mass flow rate from 49.206 kg/hr to 60 kg/hr (STREAM CAT-> Right-click Forms ->AllVariables ->FmR, specified total mass flow), and (3) the specified propylene total mass flow rate from 85 kg/hr to 100 kg/hr (STREAM C3-> Right-click Forms ->AllVariables ->FmR, specified total mass flow). We run the controllers to pause at 40 hr. Figure 7.54 shows the performance plots. The process output curve matches the set point closely.

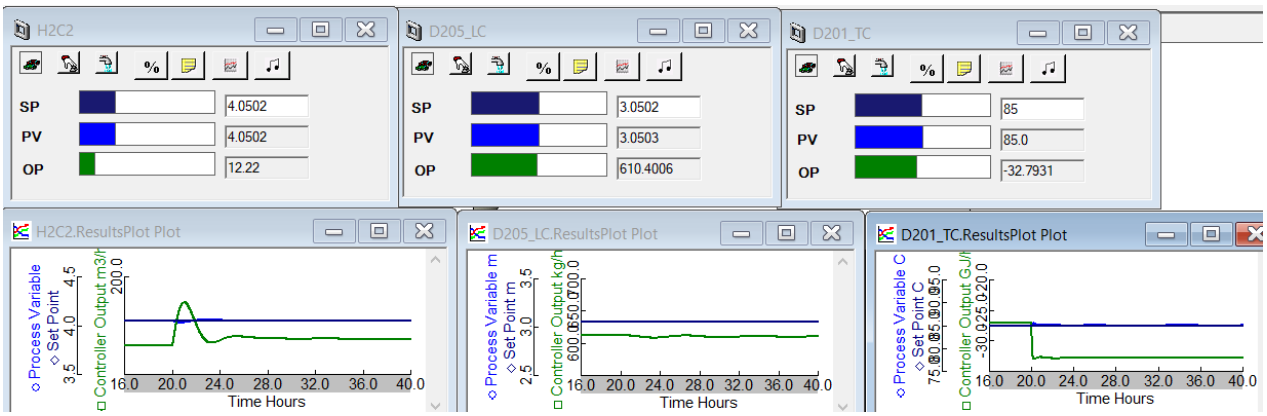


Figure 7.54 Performance of controllers after increasing ethylene mass flow rate to 9000 kg/hr, catalyst mass flow rate to 60 kg/hr and C3 mass flow rate to 100 kg/hr.

For the H2C2 controller, we find the current tuning parameters are: Gain= 35.1131 %/%, and integral time = 77.76604 min (see Figure 7.55). In the following, we want to learn how to use the tuning tool within the PID controller and check if these tuning parameters are optimum. We first save the simulation file as ***WS7.5-Final-2.dynf.***

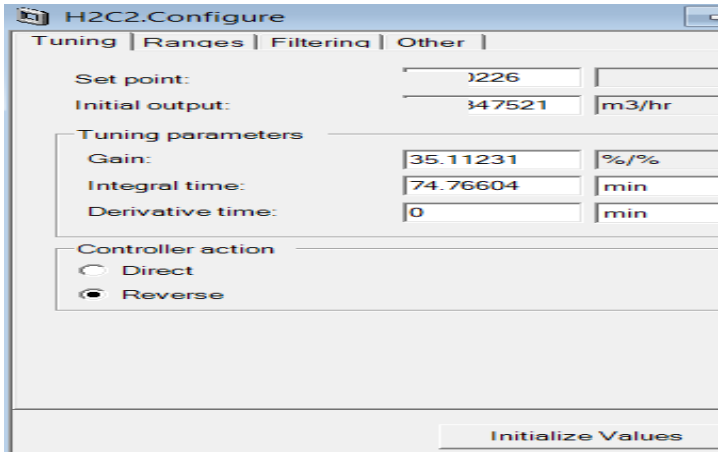


Figure 7.55 Gain and integral time for the H2C2 controller.

Next, we follow the heuristic gain and integral time for a composition controller given in Table 7.8, and change the gain displayed in Figure 7.55 to 0.1, and the integral time to 0.2 min, and initialize values. We then follow the following procedure to tune the controller parameters.

1. **Run** the simulation in **Dynamic** run mode for a few steps.
2. **Pause** the simulation.
3. Open the controller **Faceplate**.
4. Display the result plot. 
5. Click the tune button.  See the resulting Figure 7.56.
6. Click the <Start Test> button, as seen in Figure 7.56.

7. Run the simulation and you should observe that the OP of the controller is stepped up, and the ratio on the stage PV increases., **Pause** the simulation when it reaches steady state. Click **Finish Test** on the controller tuning sheet.
8. Click the **Tuning parameters** tab.

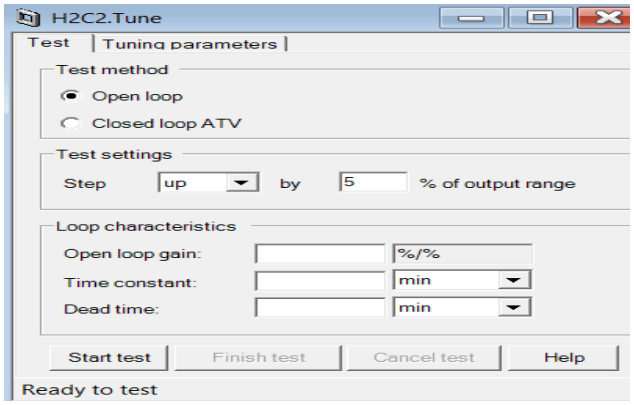


Figure 7.56 Controller "Tune" interface

9. Select the tuning method (PI controller and Cohen-Coon tuning rule). Click the <Calculate> button. See Figure 7.57. The reader may search Aspen Dynamics online help for "Cohen-Coon" about explanations of different tuning rules.

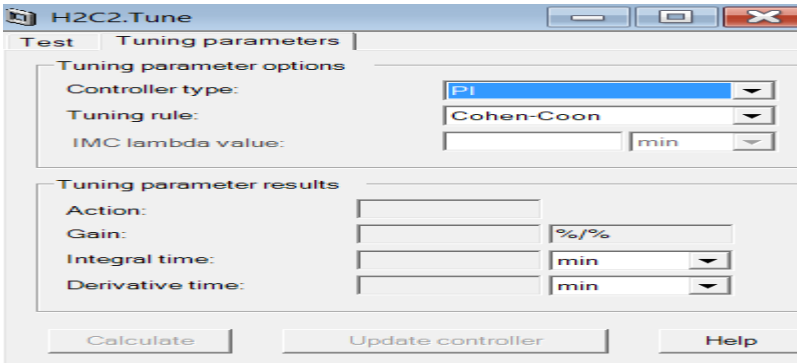


Figure 7.57 Choosing tuning rule.

10. You click **<Update Controller>** the new settings are applied. **Restart** the simulation. Click **Update controller** button. We see the updated tuning parameters in Figure 7.58, and the continually decaying controller response in Figure 7.59. This concludes the current workshop. We save the simulation file as ***WS5.3-Final-3.dynf.***

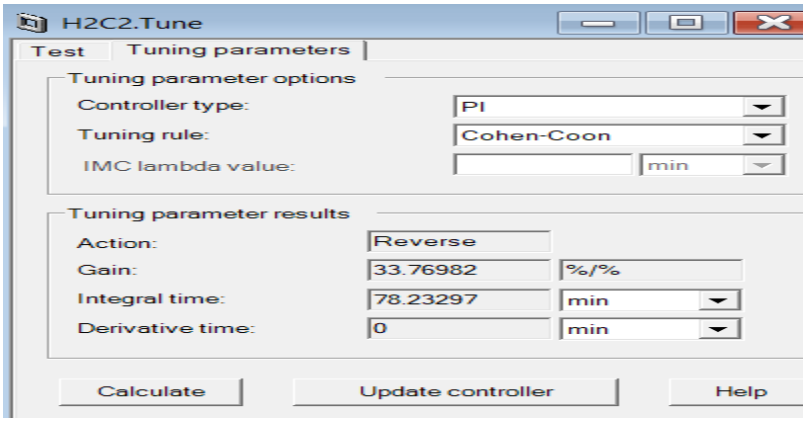


Figure 7.58 Updated tuning parameters.

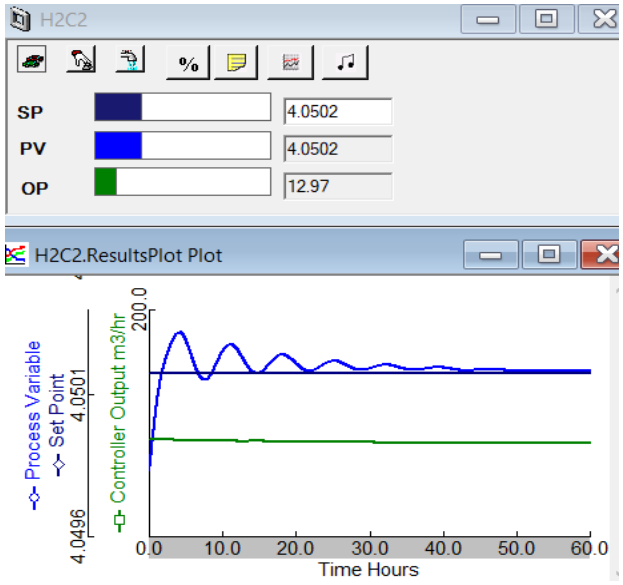


Figure 7.59 Continually decaying response resulting from the updated tuning parameters

7.8 WS7.6 Dynamic Simulation and Control of a Gas-Phase Fluidized-Bed Process for Producing LLDPE in Condensed Mode Operation

7.8.1 Objectives:

We want to show how to implement various control schemes in a gas-phase fluidized-bed process for producing LLDPE in a condensed mode operation that we covered in Section 5.8. We demonstrate the reactor pressure control using a split-range controller.

7.8.2 Converting a Steady-State Simulation Model to a Dynamic Simulation Model

Referring to Figure 5.71, we open simulation file **WS7.6_LLDPE.bkp.** Figure 7.60 shows the resulting flowsheet.

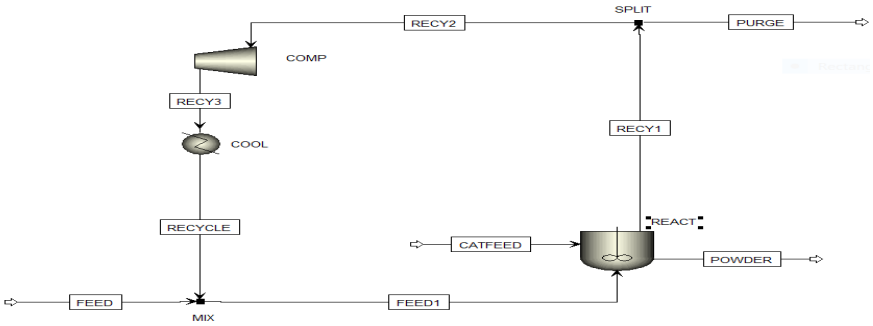


Figure 7.60 Process flowsheet for LLDPE for WS7.6

Table 7.18 gives the dynamic specifications for the current workshop.

Table 7.18 Dynamic specifications for WS7.6

Block	Specifications
COMP	Instantaneous compression
REACT	Vertical vessel, elliptical head, length =3.5 m, heat-transfer option - constant duty
COOL	Instantaneous heat transfer, heat-transfer option-constant duty
MIX1, MIX2	Instantaneous mixing

We complete the dynamic data according to Table 7.18. After running the steady-state simulation, we export the simulation from Aspen Plus to AD, and save the resulting dynamic simulation file, ***WS7.6.dynf***, and the Aspen Property problem definition file, ***WS7.6dyn.appdf***. Figure 7.61 shows the resulting flowsheet with the added default controllers.

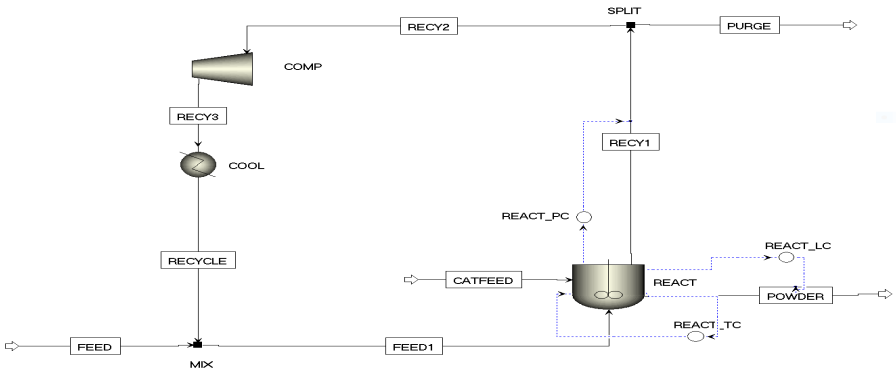


Figure 7.61 Dynamic simulation flowsheet with default controllers

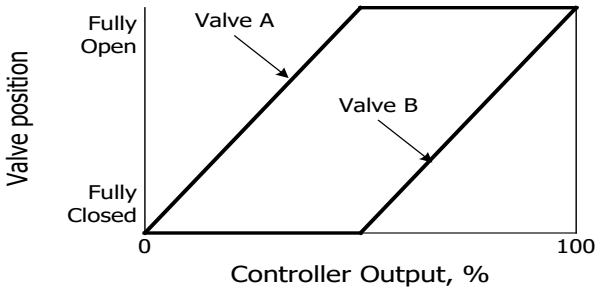
Table 7.19 Default controller specifications

Controller	Manipulated variable	Controlled variable
REACT_LC	POWDER liquid mass flow rate	REACT liquid level
REACT_TC	REACT heat duty	REACT temperature
REACT_PC	RECY1 vapor molar flow rate	REACT pressure

As the RECY1 vapor recycle molar flow rate tends to increase, leading to an increase in the reactor pressure, we need a better control scheme for the reactor pressure. We want to introduce the use of a split-range (SR) controller for our reactor pressure control. Referring to Figure 7.62, we see that valve A

opens when the controller output goes from 0 to 50%, and valve B opens when the controller output goes from 51 to 100%.

Figure 7.63 shows a modified flowsheet where we have added a split-range controller, where the PC controller output, BLOCKS("REACT_PC"), becomes the input to the SR controller. When this PC controller output is between 0 to 50%, the manipulated variable for the PC controller is the mass flow rate of N2 in the FEED stream, STREAMS("FEED").FmcR("N2"), whose value is bounded between 0 to 100 kg/hr. When this PC controller output is between 51 to 100%, the manipulated variable for the PC controller is the split fraction for the PURGE stream, BLOCKS("SPLIT").sf("PURGE"), whose value is bounded between 1E-6 to 0.1. We follow the path: Controller SplitRange -> Right-click "Forms" -> Configure: See Figure 7.64.



Figue7.62 An illustration of a split-range (SR) controller

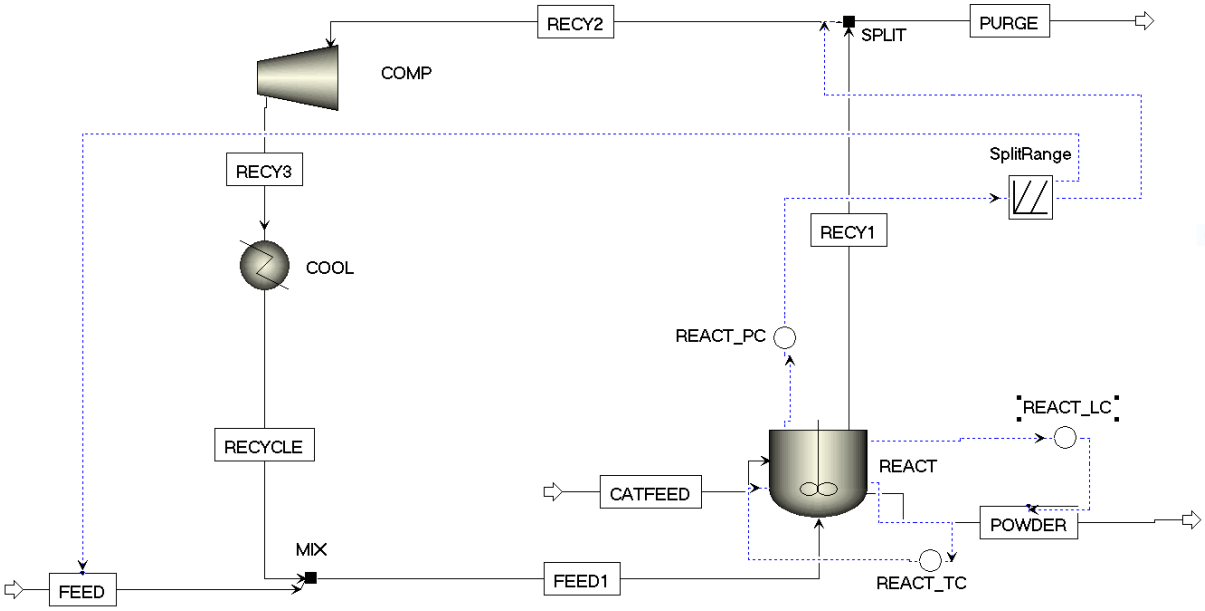
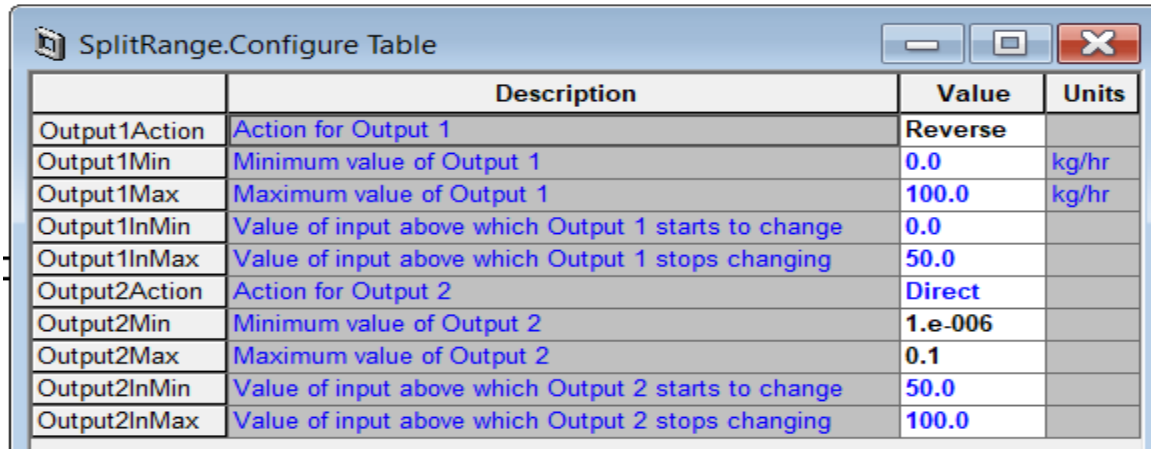


Figure 7.63 A modified dynamic process simulation flowsheet with a split-range controller



	Description	Value	Units
Output1Action	Action for Output 1	Reverse	
Output1Min	Minimum value of Output 1	0.0	kg/hr
Output1Max	Maximum value of Output 1	100.0	kg/hr
Output1InMin	Value of input above which Output 1 starts to change	0.0	
Output1InMax	Value of input above which Output 1 stops changing	50.0	
Output2Action	Action for Output 2	Direct	
Output2Min	Minimum value of Output 2	1.e-006	
Output2Max	Maximum value of Output 2	0.1	
Output2InMin	Value of input above which Output 2 starts to change	50.0	
Output2InMax	Value of input above which Output 2 stops changing	100.0	

Figure 7.64 The configuration specifications of the split-range controller.

We run the dynamic simulation to pause at 5 hr. We find that the controllers perform correctly. We save the converged simulation file as ***WS7.6-1_LLDPE.dynf*** for possible future use. We then re-save the simulation file as ***WS7.6-2_LLDPE.dynf*** before we change the FEED pressure from 30 bar to 25 bar. changes. We then run the dynamic simulation to pause at 15 hr. Figure 7.65 displays that in response to the FEED pressure decrease from 30 to 25 bar, the pressure controller performs correctly. As the controller output is at 12.75%, the SR controller activates increase of the N2 mass flow rate from 25 to 46.4267 kg/hr, while keeping the PURGE split fraction unchanged and the reactor pressure at the set point of 21.6975 bar.

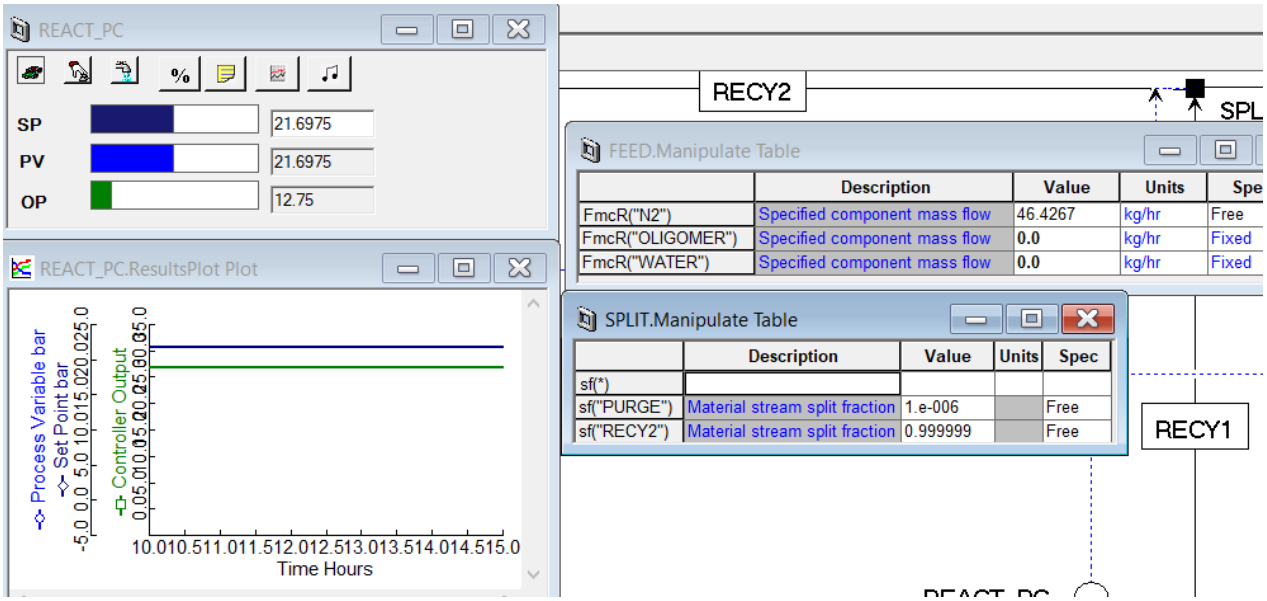


Figure 7.65 Keeping the reactor pressure at 21.6975 bar by increasing the mass flow rate of N2 from 25 to 46.4267 kg/hr as determined through a split-range controller.

The concludes the current workshop. We save the simulation file as ***WS7.6-2_LLDPE.dynf*.**

7.9 WS7.7 Dynamic Simulation and Control of a Slurry HDPE Process Using an Inferential Controller

7.9.1 Objective

The objective of this workshop is to develop an inferential control for melt index (MI) measurement in a HDPE production process. The controller controls the MI to target and minimize the difference between the target and measured MI values during grade transitions. We perform grade transition of the polymer for different MI values using the inferential controller, and we also try to improve the controller to minimize the off-spec product.

7.9.2 Inferential Control Theory and Recent Applications

In many industrial processes, it is difficult to measure certain product quality targets. In such cases, we measure some secondary process outputs to correlate the product quality with primary outputs for quality control. The additional measurement of the secondary output gives an inference on the key unmeasured variables, which is often called inferential control [11].

Inferential controllers use simple models to predict the controlled variables using plant measurements such as raw material feed rates and reactor temperature. Once a measurement is available for the controlled variable, we compare the measurement with the model prediction to adjust the model. The model can then recommend a new control action.

In one of the first applications of inferential controllers, Joseph and Brosilow [12] apply the inferential control to adjust the column temperature in a petroleum process. They build steady-state and dynamic inferential control systems [13]. Parrish and Brosilow [14] showcase how inferential controller performs better than a cascade PID controller when applied to heat exchanger and industrial reactor control.

In a recent application, Wang et al. [15] use the inferential control for temperature and simultaneous composition control of a divided wall column. Behrooz [16] use the inferential control for controlling the product quality of crude oil distillation using stochastic optimization. Choi et. al. [17] apply the inferential control for controlling the grade transition in pulping process. Durr et. al. [18] use the inferential control for controlling the quality of produced granules in a fluidized-bed process. Nikhil et. al. [19] apply the inferential control to control a fermentation process by maintaining product ethanol concentrations.

In polymer processes, the product quality measurements, such as melt index, are sparse and not very accurate. Therefore, control becomes important for reaching the quality target. MI is dependent on the hydrogen flow rate, which becomes an important manipulated variable. However, using a traditional feedback controller is inefficient, because the plant will only measure the MI of the product once every six hours, and there is a long time delay between where the hydrogen is fed to where the MI is measured. That is why we need an inferential controller for MI. Ogawa et. al. [9] use the inferential control for quality control of a HDPE process in one of the earliest applications to polymer processes. Oshima and Tanigaki [11] apply the inferential control for optimizing grade change.

7.9.3. HDPE Process Description and Steady-State Model Empirical Correlation

We consider a single CSTR for modeling the HDPE process. With a HDPE production rate of 5 metric ton (MT) per hour and with product grades of MI values of 1 to 20, we wish to optimize the grade transition while minimizing the amount of off-spec product. The HDPE process also consists of a centrifuge and

extruder, which we do not model. This follows because we can approximate the MI of product in the downstream process by the MI value at the reactor outlet by considering the time lag.

We first make a steady-state HDPE model with a single CSTR., Figure 7.66 shows a simplified HDPE process flowsheet with a single reactor and a flash drum. We and save the simulation file as ***WS7.7.bkp***. MI is dependent on the hydrogen feed flow, which represents an inferential variable. We use the model to determine an empirical correlation of MI with hydrogen feed flow rate by simulating data with different hydrogen flow rates, keeping other variables constant and calculating the product MI value.

We approximate the steady-state MI data given shown in ***WS7.7.xlsx*** by the following empirical equation applicable to the current problem:

$$\ln(\mathrm{MI_i}) = 3.266\ln(H_2) - 3.215 \quad (7.4)$$

We refer to this steady-state MI_i as the instantaneous MI, which a function of the hydrogen feed flow rate (H_2). Based on the steady-state model, we can calculate the hydrogen feed flow value for a particular MI grade.

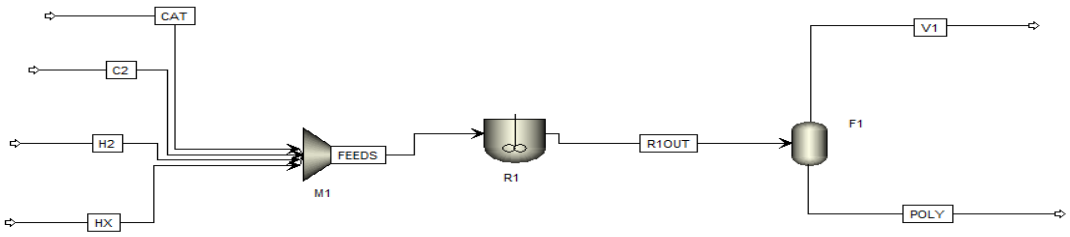


Figure 7.66. A simplified steady-state flowsheet for a slurry HDPE process.

We convert the steady-state model to a dynamic model as shown in Figure 7.67. We save the dynamic simulation file as ***WS7.7.dynf***, and the property file as ***WS7.7.dyn.appdf***. We leave the default controller settings for the temperature (R1_TC) and level control (R1_LC), but delete the pressure controller since it is not critical for the liquid-phase reactor.

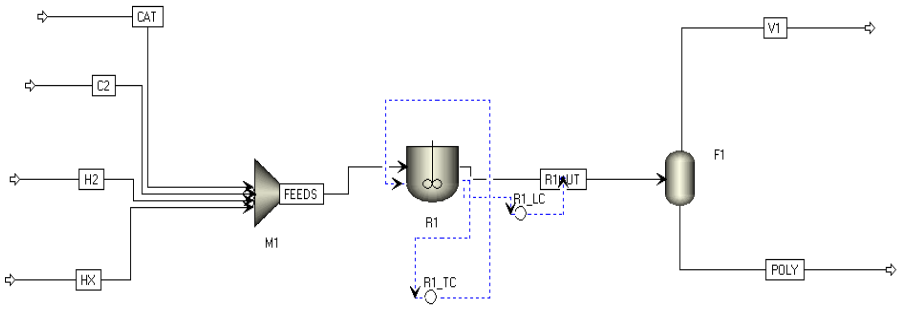


Figure 7.67. Dynamic HDPE process flowsheet

We consider the industrial correlation [21] of the melt index (MI) as function of the weight-average molecular weight (MWW) to calculate the reactor outlet MI and assume that as the plant data.

$$Melt\_Index = (11152.5/MWW)^{3.472} \quad (7.5)$$

Following WS7.4 and Figure 7.41, we enter the MI correlation as a flowsheet constraint within AD as follows:

```
# Plant model (industrial MI correlation)

A_MI as realparameter(11152.5);

B_MI as realparameter(3.472);

Plant_MI as positive;

Melt_Index = (A_MI/(STREAMS("R1OUT").MWW))^B_MI;
```

where A_MI, B_MI are the parameters and STREAMS("R1OUT"), and MWW is the weight-average molecular weight of the polymer at the reactor outlet.

7.9.4. Grade Change Transition Using Basic H2-Based Controller

The basic traditional methodology of grade change is to change the grades by changing the hydrogen feed flow rate. From the steady-state model, we calculate the H2 feed flow value to produce a particular polymer MI. To change the grade from MI of value 10 to 20, we increase the hydrogen flow rate from 5.41 kg/hr to 6.68 kg/hr, with the steady-state values obtained from the model to reach the respective MI value. Following our discussion of AD task in Section 7.5, and examples of tasks in Figures 7.28 to 7.33 and in Figure 7.41, we define the following task to increase H2 flow (STREAMS("H2").FmR) in minimum time (0.1 hr).

```
#Task for grade change using basic control using H2 feed flow rate

Task M1 runs at 5

SRamp (STREAMS("H2").FmR,6.68,0.1);

End
```

This basic control grade change process will lead to grade change, but with a constant H2-setpoint-based control, the transition will be slower leading to much larger amount of off-spec product. For the mentioned grade change, the amount off-spec material is approximately 75 metric tons (MT) in 15 hours as shown in Figure 7.68. Therefore, introducing an inferential control would become useful in reducing the off-spec material, as it constantly updates the hydrogen setpoint based on the controller error difference between the cumulative MI and the target MI.

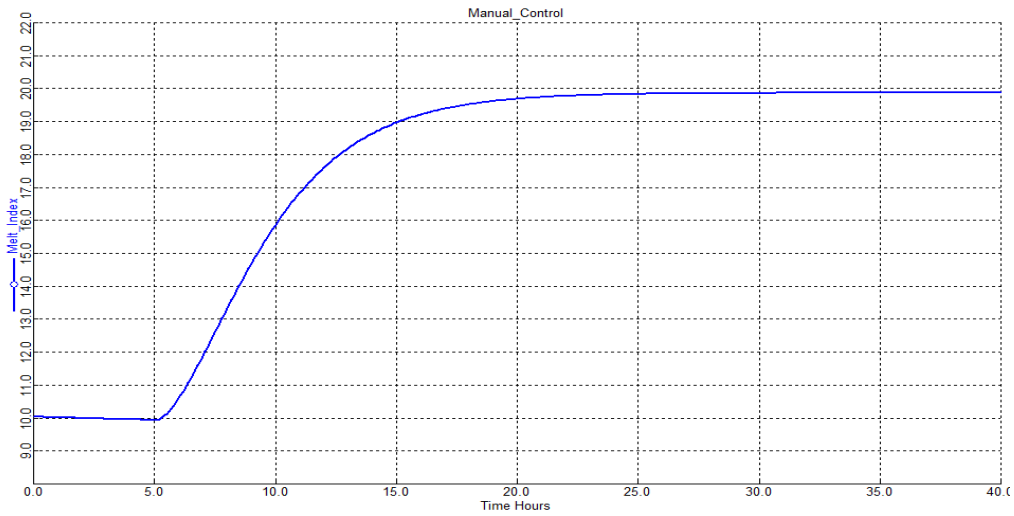


Figure 7.68. Grade change using H2-setpoint-based controller

7.9.5. Open-loop Inferential Controller Using Dynamic Model

To find a dynamic inferential control relation, we use the methodology showcased by Ogawa et. al. [9] to calculate the cumulative MI exiting the reactor. They derive the ordinary differential equation (ODE) using mass balance, quantifying the relation between the instantaneous MI and the cumulative MI formed at the exit of the reactor

$$\frac{dln(MI_c(t))}{dt} = \frac{1}{\tau_1}\log\big(MI_i(t)\big) - \frac{1}{\tau_1}\log\big(MI_c(t)\big) \quad (7.6)$$

where the MI_i is the instantaneous MI and MI_c is the cumulative MI at the exit of the reactor.

We substitute Eq. (7.4) into Eq. (7.6) to calculate the cumulative MI and then use the dynamic model to tune the time constant (τ_1) and build an inferential controller.

We model the inferential controller by simulating the dynamic differential equation (7.6) and try to iterate for an appropriate value of time constant so that the inferential model matches the plant MI correlation (7.5) with error less than 5% described by the industrial correlation. We find that a value of time constant of 4.077 hours gives a good match between plant and model values as shown in Figure 7.69. By substituting Eq. (7.4) into Eq. (7.5) and considering value of time constant, we find the inferential model ODE as Eq. (7.7):

$$\frac{d(\ln(MI_c))}{dt} = \frac{1}{4.077}[3.266ln(H_2) - 3.215 - \ln(MI_c))] \quad (7.7)$$

MI_c gives the cumulative MI at the exit of the reactor and its logarithm is defined as log_mi in the AD tasks. The hydrogen feed flow rate (H_2) is represented by STREAMS("H2").FmR in the AD model. It is defined this way since AD does not allow the logarithm of a differential.

The flowsheet constraint commands for simulating the differential equation model are given below:

```
#Open Loop Inferential Model ODE

log_mi as realvariable;
```

$log_mi = (1/4.0775)*((3.266*LOGe(STREAMS("H2").FmR)) - 3.2157 - log_mi);

predicted_mi as realvariable;

predicted_mi = EXP(log_mi);

In order to compare the MI values and iterate value for the time constant for an open-loop controller, we perform grade transition by changing the values of H2 feed flow for grades of MI 1, 10, 20 by creating task as explained previously. We change the grades by using tasks to change H2 flow rate (Streams("H2").FmR) at 2 hrs to 5.46 kg/hr and at 25 hrs to 2.67 kg/hr, respectively, as explained previously. We set the H2 flow rate as a fixed variable prior to implementing the task.

Figure 7.69 compares MI values from the open-loop inferential control with the actual correlation-based MI values at a time constant of 4.07 hr, resulting in a lowest error of 3%.

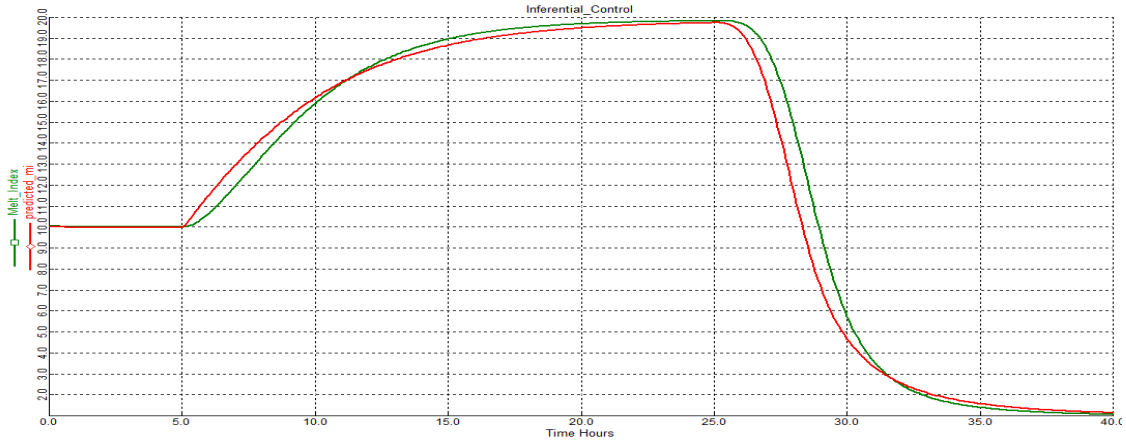


Figure 7.69. Comparison of MI values from the open-loop inferential control MI (predicted_MI) with the actual correlation-based MI (Melt_Index) (time constant = 4.07 hours)

7.9.6. Closed-Loop Inferential Controller

To automate the inferential controller, we need a closed-loop controller using the open-loop model ODE. This allows us to input a target MI, so that the inferential model can calculate the control action accordingly. To form the closed-loop inferential model, we need to discretize the inferential control ODE using the Euler method and simplify to get the following equation:

$$(\ln(MI_{target}) - \ln(MI_c))/\Delta t = (1/4.07) * (3.266ln(H2_{setpoint}) - 3.215 - \ln(MI_c)) \quad (7.8)$$

where MI_{target} is the fixed target MI that the controller needs to achieve and MI_c is the cumulative MI at the exit of the reactor. We use the same open-loop time constant 4.07hr for the closed-loop discretized ODE. Thus, Eq. (7.8) closes the loop since by inputting a given target MI, it continuously back-calculates the hydrogen setpoint which can then be used to equate to the hydrogen flow rate resulting in the inferential control action. The time interval Δt is a fixed parameter that can be used to tune the controller and reduce overshoot.

Following WS7.4 and Figure 7.42, we enter the closed-loop inferential control as a flowsheet constraint within AD. #Closed Loop Inferential Control

target_mi as realvariable(fixed,1);

delta_t as realvariable(fixed, 1);

h2_setpoint as realvariable(free);

(LOGe(target_mi)-log_mi)/delta_t = (1/4.0775)*((3.2667*LOGe(h2_setpoint)) - 3.2157 - log_mi);

h2_setpoint= STREAMS("H2").FmR;

The variable as defined in the constraint are MI_{target} as target_mi, Log(MI_C) as log_mi ,Δt as delta_t. We create a new variable called 'target_mi' and set it as a fixed variable and use log_mi as the initial variable condition for the discretized ODE. Also make Streams("H2").FmR as a free variable since it is equal to the H2 setpoint.

Figure 7.70 shows all the three flowsheet constraints for the current workshop.

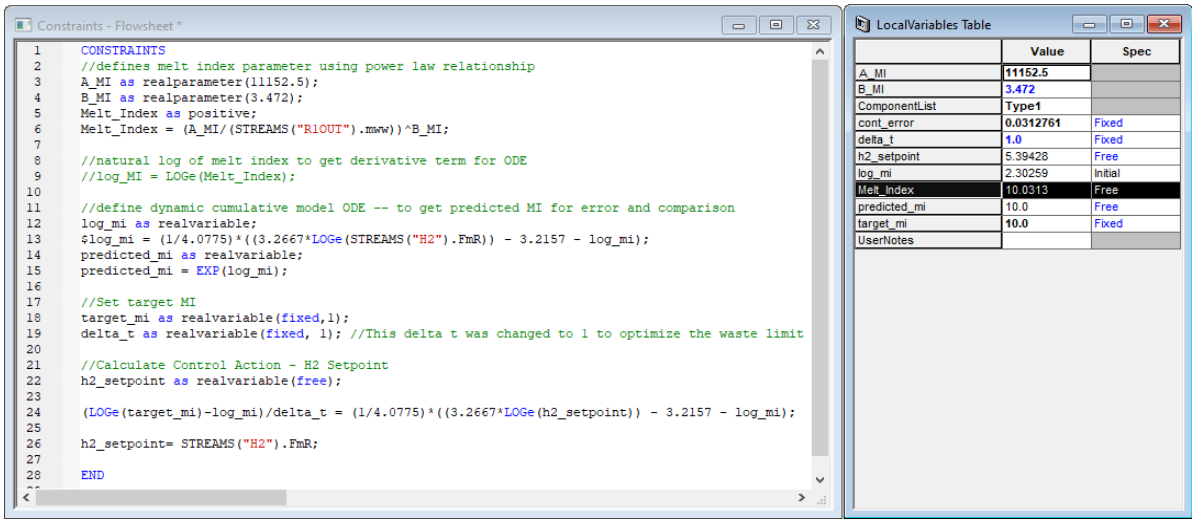


Figure 7.70. Snapshot of the overall constraints and the local variable specification

We simulate a grade change from 10 to 20 using AD tasks. The grade change for the MI happens in 5 hrs with only 25 metric ton (MT) of off-spec material. Figure 7.71 illustrates the inferential model prediction for grade change. The y-coordinate, predicted_mi, represents predicted MI value by the inferential controller.

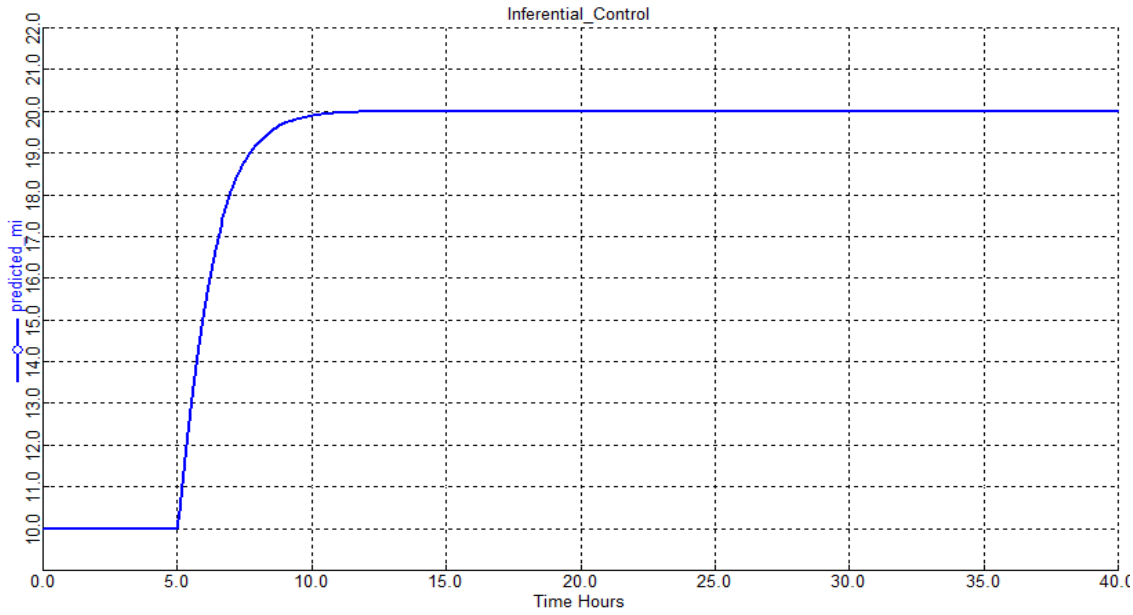


Figure 7.71. MI grade change from 10 to 20 using Inferential Control

Thus, we can compare the grade change due to the inferential control Melt Index with the basic constant hydrogen flow based control resulting in faster grade transition in inferential control (5hrs) compared to basic control (15 hrs). Figure 7.72 illustrates that the inferential control reduces off-spec material by 50 metric ton (MT) and by 10 hours to reach the new MI target. Hence, we have showcased the utility of inferential control for polymer grade change. This concludes the current workshop.

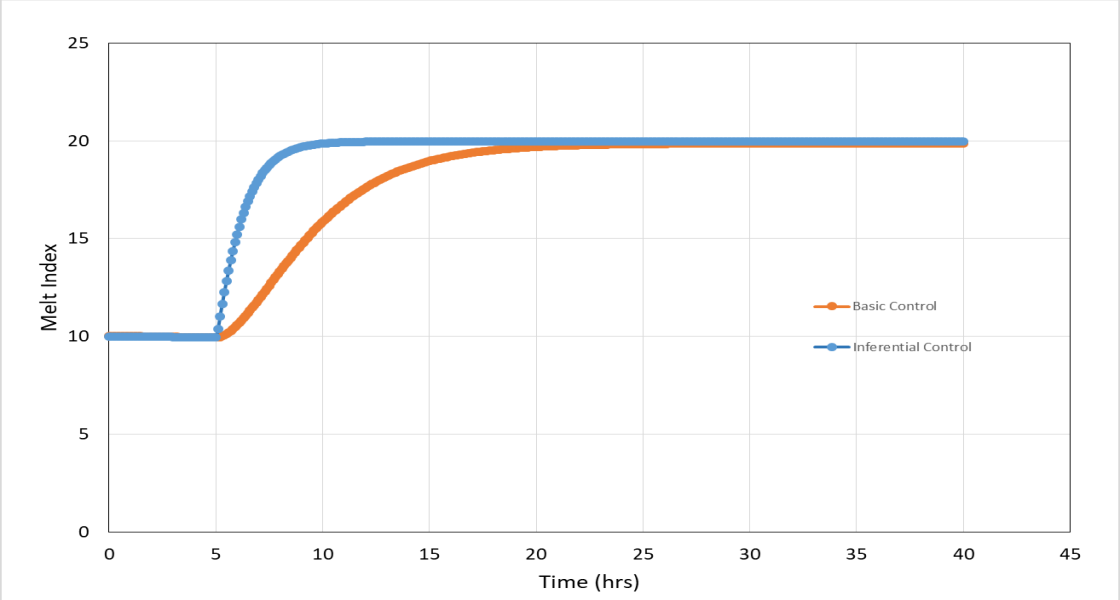


Figure 7.72. Comparison of grade change process using the inferential control with the basic H2-based control

This concludes the current workshop on inferential control applied to grade change operationsof a slurry HDPE process.

7.10 Conclusion

The transition from steady-state to dynamic simulation in polymer process modeling, facilitated by Aspen Plus and Aspen Plus Dynamics, represents a significant advancement in process engineering. This chapter has demonstrated that dynamic simulation offers valuable insights into the time-dependent behavior of polymer production processes, particularly for HDPE and LLDPE manufacturing. By providing a comprehensive workflow and practical examples, we have shown how engineers can effectively implement these advanced modeling techniques to improve process control, optimize grade transitions, and enhance overall production efficiency. The integration of real-time visualization, PID controller configuration, and discrete event sequencing further empowers users to manage complex scenarios in polymer manufacturing. As a future direction, integrating machine learning with first-principle dynamic simulations could be utilized for more efficient process control [23]. Ultimately, the adoption of dynamic simulation techniques as presented in this chapter has the potential to drive significant improvements in the process industry, leading to more efficient, flexible, and responsive production processes.

This chapter is published with Wiley publication in the book *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing by Liu & Sharma.[22,25-38]*

7.11 Bibliography

1. Khare, N.P.; Seavey, K. C.; Liu, Y.A.; Ramanathan, S.; Lingard, S.; Chen, C. C. (2002) Steady-state and dynamic modeling of commercial slurry high-density polyethylene (HDPE) processes. *Ind. Eng. Chem. Res*. **41**, 5601.

2. Khare, N. P.; Lucas, B.; Seavey, K.C.; Liu, Y. A.; Sirohi, A.; Ramanathan, S.; Lingard, S.; Song, Y.; Chen, C.C. (2004). Steady-State and Dynamic Modeling of Commercial Gas-Phase Polypropylene Processes Using Stirred-Bed Reactors. *Ind. Eng. Chem. Research*, **43**, 884.

3.Zheng, Z. W.; Shi, D. P.; Su, P. L.; Luo, Z. H.; Li, X. J. (2011). Steady-State and Dynamic Modeling of the Basell Multireactor Olefin Polymerization Process. *Ind. Eng. Chem. Research*, **50**, 322.

4.Luo, Z. H., Su, P. L.; Shi, D. P.; Zhang, Z. W. (2009). Steady-State and Dynamic Modeling of Commercial Bulk Polypropylene Process of Hypol Technology. *Chem. Eng. J*., **149**, 370.

5.You, C. (2006). Modeling and Analysis of Polypropylene Process in Steady and Dynamic States, M. S. thesis, College of Chemical Engineering, Tianjin University.

6. Aspen Technology, Inc. (2020). *Aspen Plus Dynamics Online Help*, V11. "Troubleshooting Aspen Plus Dynamics".

7. Smith, Carlos A.; Corripio, Armando, B. (1997). *Principles and Practice of Automatic Process Control*, 2nd edition, Wiley, New York.

8. Sinclair, K. B. (1983). "Characteristics of Linear LPPE and Description of UCC Gas Phase Process," Process Economics Report, SRI International, Menlo Park, CA (1983).

9. Ogawa, M.; M. Ohshima; K. Morinaga; Watanabe, F. (1999). Quality Inferential Control of an Industrial High Density Polyethylene Process. *Journal of Process Control*, **9**, 51.

10. Lou, H. C.; Su, H. Y.; Xie, L.; Gu, Y.; Rong, G. (2012). Inferential Model for Industrial Polypropylene Melt Index Prediction with Embedded Priori Knowledge and Delay Estimation. *Ind. Eng. Chem. Res*. **51**, 8510.

11. Marlin, T. A. (2004). Inferential Control. In *Process Control: Designing Processes and Control Systems for Dynamic Performance*; McGraw-Hill Inc. pp 555-580.
12. Joseph, B.; Brosilow, C. B. (1978). Inferential Control of Processes: Part I. Steady State Analysis and Design. *AIChE Journal,* **24**, 485.
13. Joseph, B.; Brosilow, C. (1978). Inferential Control of Processes: Part III. Construction of Optimal and Suboptimal Dynamic Estimators. *AIChE Journal***, 24**, 500.
14. Parrish, J.; Brosilow, C. (1985). Inferential Control Applications. *Automatica,* **21**, 527.
15. Wang, J.; Yu, N.; Chen, M.; Cong, L.; Sun, L. (2018). Composition Control and Temperature Inferential Control of Dividing Wall Column Based on Model Predictive Control and PI Strategies. *Chinese Journal of Chemical Engineering***, 26**, 1087.
16. Behrooz, H. A. (2019). Robust Set-Point Optimization of Inferential Control System of Crude Oil Distillation Units. *ISA Transactions*, **95**, 93.
17. Choi, H.-K.; Son, S. H.; Sang-Il Kwon, J. (2021). Inferential Model Predictive Control of Continuous Pulping under Grade Transition. *Industrial and Engineering Chemistry Research***, 60**, 3699.
18. Dürr, R.; Neugebauer, C.; Palis, S.; Bück, A.; Kienle, A. (2020). Inferential Control of Product Properties for Fluidized Bed Spray Granulation Layering. *IFAC-Papers OnLine,* **53**, 11410.
19. Pachauri, N.; Singh, V.; Rani, A. (2017). Two Degree-of-Freedom PID-Based Inferential Control of Continuous Bioreactor for Ethanol Production. *ISA transactions***, 68**, 235.
20. Ohshima, M.; Tanigaki, M. (2000). Quality Control of Polymer Production Processes. *Journal of Process Control***, 10,** 135.
21. Mattos Neto, A.G., Freitas, M.F., Nele, M. and Pinto, J.C. (2005). Modeling Ethylene/1-butene Copolymerizations in Industrial Slurry Reactors. *Industrial and Engineering Chemistry Research,* **44***,* 2697.

22. Liu, Y. A., & Sharma, N. (2023). *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing*. Wiley-VCH GmbH (https://doi.org/10.1002/9783527843831)

23. Sharma, N., & Liu, Y. A. (2019). 110th anniversary: an effective methodology for kinetic parameter estimation for modeling commercial polyolefin processes from plant data using efficient simulation software tools. *Industrial & Engineering Chemistry Research*, *58*(31), 14209-14226. https://doi.org/10.1021/acs.iecr.9b02277

24. Sharma, N., & Liu, Y. A. (2022). A hybrid science-guided machine learning approach for modeling chemical processes: A review. *AIChE Journal*, *68*(5), e17609. https://doi.org/10.1002/aic.17609

25. Liu, Y. A., & Sharma, N. (2023). Introduction to Integrated Process Modeling, Advanced Control, and Data Analytics in Optimizing Polyolefin Manufacturing. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing* (Chapter 1, pp. 1-40). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch1

26. Liu, Y. A., & Sharma, N. (2023). Selection of Property Methods and Estimation of Physical Properties for Polymer Process Modeling. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing* (Chapter 2, pp. 41-86). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch2

27. Liu, Y. A., & Sharma, N. (2023). Reactor Modeling, Convergence Tips, and Data-Fit Tool. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing* (Chapter 3, pp. 87-114). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch3

28. Liu, Y. A., & Sharma, N. (2023). Free Radical Polymerizations: LDPE and EVA. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing* (Chapter 4, pp. 115-162). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch4

29. Liu, Y. A., & Sharma, N. (2023). Ziegler–Natta Polymerization: HDPE , PP , LLDPE, and EPDM. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 5, pp. 163-265). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch5

30. Liu, Y. A., & Sharma, N. (2023). Free Radical and Ionic Polymerizations: PS and SBS Rubber. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 6, pp. 267-319). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch6

31. Liu, Y. A., & Sharma, N. (2023). Improved Polymer Process Operability and Control Through Steady-State and Dynamic Simulation Models. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 7, pp. 321-379). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch7

32. Liu, Y. A., & Sharma, N. (2023). Model-Predictive Control of Polyolefin Processes. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 8, pp. 381-476). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch8

33. Liu, Y. A., & Sharma, N. .2023. Application of Multivariate Statistics to Optimizing Polyolefin Manufacturing. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing* (Chapter 9, pp. 477-531). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch9

34. Liu, Y. A., & Sharma, N. (2023). Applications of Machine Learning to Optimizing Polyolefin Manufacturing. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 10, pp. 553-650). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch10.

35. Liu, Y. A., & Sharma, N. (2023). A Hybrid Science-Guided Machine Learning Approach for Modeling Chemical and Polymer Processes. In *Integrated Process Modeling, Advanced Control and Data Analytics for Optimizing Polyolefin Manufacturing.* (Chapter 11, pp. 651-698). Wiley-VCH GmbH. https://doi.org/10.1002/9783527843831.ch11

36. Sharma, N. and Liu, Y., 2019, November. Polyolefin Process Modeling and Monitoring. In *2019 AIChE Annual Meeting*. AIChE.

37. Sharma, N. and Liu, Y., 2020, November. Polyolefin Process Improvement Using Machine Learning. In *2020 Virtual AIChE Annual Meeting*. AIChE

38. Sharma, N., 2022, November. Polyolefin Property Estimation using Process Modeling and Machine Learning in Industry. In *2022 AIChE Annual Meeting*. AIChE.