



# D-RM Builder User Manual

Version 2014.10.0

October 31, 2014



This Material was produced under the DOE Carbon Capture Simulation Initiative (CCSI), and copyright is held by the software owners: ORISE, LANS, LLNS, LBL, PNNL, CMU, WVU, et al. The software owners and/or the U.S. Government retain ownership of all rights in the CCSI software and the copyright and patents subsisting therein. Any distribution or dissemination is governed under the terms and conditions of the CCSI Test and Evaluation License, CCSI Master Non-Disclosure Agreement, and the CCSI Intellectual Property Management Plan. No rights are granted except as expressly recited in one of the aforementioned agreements.

## Revision Log

Version Number	Release Date	Description
Version 2013.10.0	10/31/2013	Initial Release
Version 2014.10.0	10/31/2014	2014 October IAB Release – This release includes more parameters and plots to describe and show the accuracy of the generated D-RMs. The uncertainty quantification feature based on unscented Kalman filter is also included.

## Table of Contents

<b>1. Introduction .....</b>	<b>1</b>
1.1. Motivating Example.....	1
1.2. Features List .....	1
<b>2. Tutorial .....</b>	<b>3</b>
2.1. Installation Instructions .....	3
2.2. Building Data-Driven D-RM using DABNet Framework and BP-Based ANN Training ..	6
2.3. Building Data-Driven D-RM using DABNet Framework and IPOPT-Based ANN Training .....	24
2.4. Building Data-Driven D-RM using DABNet Framework with Pole Values Optimized...	26
2.5. Building Data-Driven D-RM using DABNet Framework with Ramp Changes to Replace Step Changes for Hard-to-Converge Stiff Models.....	37
2.6. Building Data-Driven D-RM using NARMA Framework .....	43
2.7. Testing Generated D-RM for Dynamic Simulation in MATLAB .....	47
<b>3. USAGE Information .....</b>	<b>49</b>
3.1. Environment/Prerequisites .....	49
3.2. Support.....	49
3.3. Restrictions .....	49
<b>4. Advanced Features .....</b>	<b>50</b>
<b>5. Debugging.....</b>	<b>50</b>
5.1. How to Debug .....	50
5.2. Known Issues.....	51
5.3. Reporting Issues .....	51
<b>6. References .....</b>	<b>51</b>

## List of Figures

Figure 1: Welcome to the InstallShield Wizard for D-RM Builder Window .....	3
Figure 2: Destination Folder Window .....	4
Figure 3: Ready to Install the Program Window .....	4
Figure 4: InstallShield Wizard Completed Window.....	5
Figure 5: Simulink Flowsheet with Embedded ACM Model .....	7
Figure 6: Simulink Windows for Setting Up Input and Output Variables.....	7
Figure 7: Main Window of the D-RM Builder .....	8
Figure 8: Input Variable Dialog Window .....	10
Figure 9: Output Variable Dialog Window.....	11
Figure 10: Training Sequence Dialog Window .....	12
Figure 11: Validation Sequence Dialog Window .....	13

Figure 12: MATLAB Window Showing the Input Step Changes and Output Responses .....	14
Figure 13: DABNet Submenu for Selecting a D-RM Model Type .....	15
Figure 14: DABNet DRM Parameter Dialog Window .....	15
Figure 15: Result Plotting Dialog Window.....	17
Figure 16: Plots of Input and Output Data as Functions of Time for Training Data .....	18
Figure 17: Plots of Input and Output Data as Functions of Time for Validation Data .....	19
Figure 18: Process and Measurement Uncertainty Dialog Window .....	20
Figure 19: Result Plotting Dialog Window.....	21
Figure 20: Plots Showing the Results of UQ Analysis for the Validation Sequence.....	22
Figure 21: Simulink Flowsheet with Embedded ACM Model .....	27
Figure 22: Simulink Dialog Window for Setting Up Input and Output Variables .....	27
Figure 23: Input Variable Dialog Window .....	28
Figure 24: Result Plotting Dialog Window.....	30
Figure 25: Predicted Response of Training Sequence using Default Pole Values.....	31
Figure 26: Predicted Response of Validation Sequence using Default Pole Values .....	32
Figure 27: Predicted Response of Training Sequence using Optimized Pole Values.....	33
Figure 28: Predicted Response of Validation Sequence using Optimized Pole Values.....	34
Figure 29: Process and Measurement Uncertainty Dialog Window .....	35
Figure 30: Result Plotting Dialog Window.....	35
Figure 31: Plots of UQ Analysis Results on the D-RM of the pH Neutralization Reactor.....	36
Figure 32: Simulink Dialog Window for Setting Up Input and Output Variables .....	38
Figure 33: Input Variable Dialog Window .....	39
Figure 34: Result Plotting Dialog Window.....	41
Figure 35: Predicted Response of CO <sub>2</sub> Removal Fraction for the Training Sequence .....	42
Figure 36: NARMA DRM Parameter Dialog Window .....	43
Figure 37: Predictions for Validation Sequence using High-Fidelity Model History data.....	45
Figure 38: Predictions for Validation Sequence using D-RM Predicted History Data.....	46
Figure 39: Predicted Result by the D-RM Object in MATLAB .....	48

To obtain support for this package, please send an email to  
[ccsi-support@acceleratecarboncapture.org](mailto:ccsi-support@acceleratecarboncapture.org).

## 1. INTRODUCTION

The Dynamic Reduced Model (D-RM) Builder is a software tool used to generate data-driven dynamic reduced models from rigorous dynamic process simulations of high-fidelity dynamic models consisting of differential and algebraic equations (DAEs). DAE-based models are usually computationally intensive to solve, especially when stiff DAEs are involved. For instance, the sorbent-based bubbling fluidized bed CO<sub>2</sub> adsorber-reactor model contains over 20,000 DAEs and very small time steps (<0.001 second) have to be used to solve the stiff equations in the rigorous model. The D-RMs generated by the D-RM Builder enable for faster computation of the system responses (up to several orders of magnitude faster), which enable the development of an advanced process control framework and the integration of the dynamic models within a large-scale dynamic simulation.

### 1.1. Motivating Example

The solid-sorbent-based bubbling fluidized bed CO<sub>2</sub> adsorber-reactor for the post-combustion carbon capture is a good example to which the D-RM Builder tool can be applied. The dynamics of the adsorber-reactor was modeled by a CCSI team using Aspen Custom Modeler (ACM). The high-fidelity dynamic model of the twin-bed adsorber-reactor contains over 20,000 equations. The feed streams include CO<sub>2</sub>-containing flue gas, solid sorbent, and multiple cooling water streams. Since some equations are very stiff, a minimum integration time step of 0.001 second has to be used. As a result, the computer CPU time required to calculate the response of the system over a period of operating time is much longer than the real operating time, especially when there is a change in model inputs (step or ramp change). The data-driven D-RM can be generated by fitting the system outputs in response to the changes of system inputs through certain plant identification models. The response of the system to the input changes can be simulated by the ACM model. The D-RM Builder is provided for a user to configure the input and output variables of interest, prepare a sequence of step changes of input variables, launch ACM simulations, generate a D-RM, and export the D-RM in a form of a MATLAB function file (.m file), which can be called by MATLAB to calculate the system response given the model inputs with a CPU execution time a few orders of magnitude shorter than that required by the ACM model. The speedup in CPU time enables the implementation of advanced process control (APC) systems.

### 1.2. Features List

The D-RM Builder is a Windows<sup>®</sup> application with a graphic user interface (GUI). Data-driven D-RMs can be generated by the D-RM Builder based on a set of high-fidelity model outputs in response to a sequence of input changes within a range of operating conditions. The D-RM Builder can parse the case file of a rigorous ACM model, identify the input and output variables and their steady-state values, and recommend some default parameters used for building the D-RMs. Through the GUI, a user can configure the input variables, specify their lower and upper limits, and prepare a sequence of step changes or ramp changes based on the Latin Hypercube Sampling (LHS) method with desired durations of the step changes to excite the system in a range of frequencies. The simulations of high-fidelity models, currently implemented for ACM only, can be launched directly in the D-RM Builder through MATLAB/Simulink's embedded custom block. The simulation results stored in the MATLAB

workspace can be transferred back to the memory of the D-RM Builder by the MATLAB engine and can be used to generate the D-RMs. A separate set of step-change sequence can also be generated for model validation purpose and the step-change sequence's response calculated by the ACM model through the MATLAB engine. The generated D-RMs and the user inputs for case setup and configuration can be saved in a D-RM Builder case file with the extension “.drm”. Once a case file is saved, the user can copy the file to a different directory or to a different computer where the D-RM Builder is installed and open the file later to continue the D-RM building process or visualize the properties of the generated D-RMs. The case file also contains the equations and the data of the original high-fidelity model, which can be exported from the D-RM Builder, if needed.

Two types of D-RMs are supported in the current version including the Decoupled A-B Net (DABNet) model, a state-space based model, and the Nonlinear Auto-regressive Moving Average (NARMA) model. Different building options are provided for the user to select from including a training method for model parameter optimization. Both D-RM model types require the training of artificial neural networks (ANNs). Two ANN training methods, back propagation (BP) and interior point optimization (IPOPT), are provided. The generated D-RM can be validated by performing another set of high-fidelity model simulations with a different input change sequence and comparing the response to that predicted by the generated D-RMs. The response calculated by the high-fidelity model and that by the D-RM can be displayed through the D-RM Builder's GUI. The accuracy of the generated D-RM can be visualized by comparing the D-RM predictions to the corresponding ACM predictions, including the relative errors and the coefficient of determination  $R^2$ . For a state-space based data-driven model, Uncertainty Quantification (UQ) analysis can be performed on the validation data using Unscented Kalman Filter (UKF), providing the covariance matrices of state-space and output variables. The messages of the model building process including command sequence, outputs during model training, and optimized model parameters are displayed in the main text window of the D-RM Builder, which can be saved to a log file for future reference. The generated D-RM can be exported to a MATLAB script file, which can be used to calculate the model response, as well as the sensitivity matrix given the model inputs. The results of the high-fidelity model simulations and D-RM predictions for both the training and the validation input sequences can also be exported to text files in comma-separated value (CSV) format, which can be opened by Microsoft® Excel®.

Along with the D-RM Builder application, several MATLAB files are also provided as part of the software tool. These files define the MATLAB classes that can be used to create the D-RM objects in the MATLAB workspace given the D-RM files exported from the D-RM Builder. The functions in the D-RM objects can be called to perform dynamic simulations.

## 2. TUTORIAL

### 2.1. Installation Instructions

The installer file of the D-RM Builder is “Setup.exe”. The file is a standard Microsoft Windows installer. The tool, currently a 32-bit version, can be installed on any Windows computer with an operating system version of XP or higher. To install the application, browse the file system to the folder where the installer “Setup.exe” file is located and double-click the file name or icon to launch the installer. An InstallShield Wizard window displays, a message of “Preparing to install...” displays, and then a “Welcome to the InstallShield Wizard for DRM Builder” window as shown in Figure 1 displays.

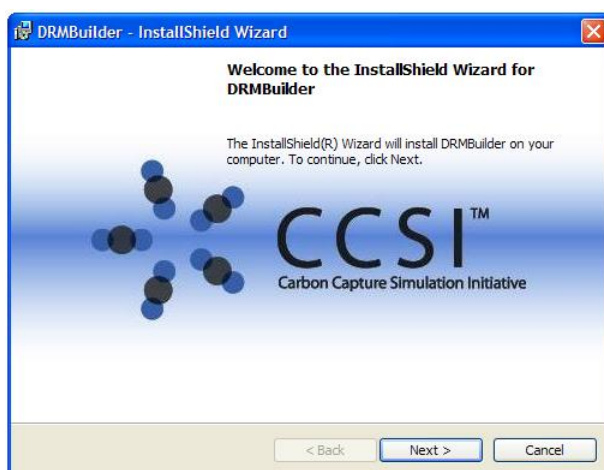
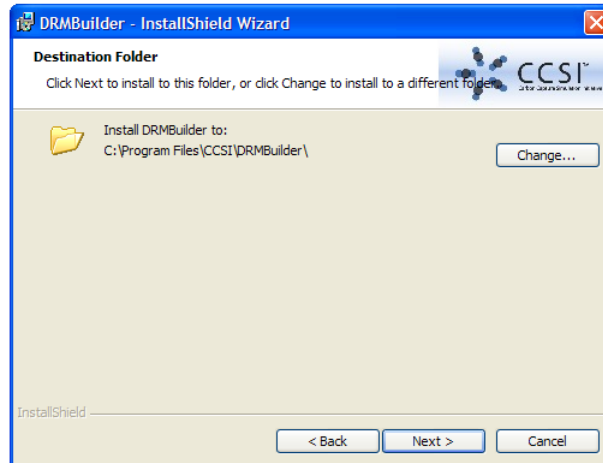


Figure 1: Welcome to the InstallShield Wizard for D-RM Builder Window

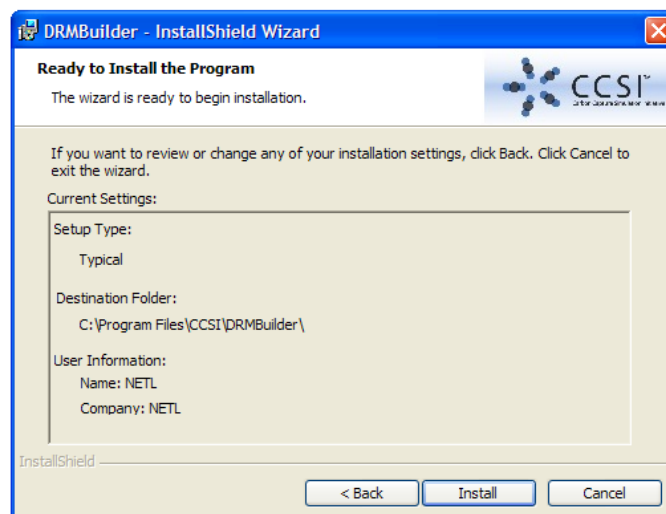


The license agreement displays if the user clicks “Next” on the “Welcome” window. If the user agrees with the CCSI license terms after reading the license agreement, select the “I accept...” option and then click “Next” to continue. The Destination Folder window as shown in Figure 2 displays.



**Figure 2: Destination Folder Window**

The default installation folder is “C:\Program Files\CCSI\DRMBuilder”. If the operating system is a 64-bit system, the default folder is “C:\Program Files(x86)\CCSI\DRMBuilder”. Select a different folder by clicking “Change” to browse through the local file system and select an existing folder to install the software. The “C:\Program Files\CCSI\DRMBuilder” folder is used as the root path of the installed directory in the tutorials of this User Manual. If the user selects a different installation folder, use the path of the selected installation folder in place of the default installation folder. Click “Next” to display the “Ready to Install the Program” window as shown in Figure 3.



**Figure 3: Ready to Install the Program Window**

Click “Install” on the “Ready to Install the Program” window to start the installation process. Since the D-RM Builder requires ACM from AspenTech and MATLAB/Simulink (32-bit version) from MathWorks, the installer checks if those two software packages are installed on the machine. If either one of the software packages are not installed, a warning message window displays, reminding the user to install those packages first and then the installer exits. The installer also checks if the folder for the MATLAB engine is included in the Windows environment variable “PATH”. If the folder is not included, a pop-up message displays indicating that the folder is appended to the “PATH” variable. The installer also registers the MATLAB engine if it has not been registered. After the installation is completed, an InstallShield Wizard Completed window as shown in Figure 4 displays.



**Figure 4: InstallShield Wizard Completed Window**

Click “Finish” to finish the installation. The installer creates a shortcut of the D-RM Builder application on the desktop. The application is also added to the program list of the Windows “Start” button. To confirm the correct installation, browse to the installation folder to verify that four subfolders have been created by the installer, a “bin” folder for executables and DLLs, a “docs” folder for user manual and license related documents, an “examples” folder for examples to be used in the tutorials, and a “matlab\_files” folder for the MATLAB class, function, and script files to be used by the D-RM Builder and the user to create the D-RM object in MATLAB for dynamic simulations.

## 2.2. Building Data-Driven D-RM using DABNet Framework and BP-Based ANN Training

### Description

This feature uses a state-space nonlinear process identification method, Decoupled A-B Net (DABNet), to build a D-RM that is composed of a decoupled linear dynamic system followed by a nonlinear static mapping from state-space variables to output variables. The linear dynamic system is initially spanned by a set of discrete Laguerre systems and then cascaded with a single hidden layer perception based on a feedforward artificial neural network (ANN). This methodology has been based on a journal article by Sentoni et al., 1998 [1]. A model reduction technique (linear balancing) helps reduce the dimensionality of the perception input although the user is given the option of also using the Laguerre states directly (without balancing) as the neural network input. In this section of the tutorial, the system identification using DABNet framework along with back propagation (BP) based ANN training feature of the D-RM Builder is demonstrated. As the first tutorial, detailed descriptions of the individual commands and dialog windows are provided. It is recommended that the user go through this tutorial first.

### Example

To demonstrate the feature, an example named Van-de-Vusse reactor is provided in a subfolder “C:\Program Files\CCSI\DRMBuilder\Examples\VdV” under the D-RM Builder’s installation directory. The nonlinear reactor model is a benchmark process popular in control literature. The following are the steps to build a D-RM using DABNet framework and BP-based ANN training:

1. Copy the entire subfolder of the Van-de-Vusse reactor example “C:\Program Files\CCSI\DRMBuilder\examples\VdV” to a desired location. Confirm the user of the D-RM Builder has write-permissions to this folder. Browse to the “VdV” subfolder.
2. Open the “VdV\_Reactor.acmf” file by double-clicking the file. The Van-de-Vusse Reactor model opens within ACM.
3. Make a steady-state run of the model. For the D-RM Builder to work, the initial states should be at steady-state. If the user has trouble getting convergence for a steady-state run, carefully look within the model or at other criteria for troubleshooting. From the ACM menu, select **Run → Mode → Steady-State** (or in the Run Control toolbox select “Steady State” from the RunMode drop-down list). Make a steady-state run by clicking “Run” on the toolbar or pressing F5. Once the “Run Complete” dialog box displays, click “OK”. Navigate to **Tools → Snapshot**. In the “Snapshot Management” window, select the most recent steady-state run (at the top of the list) and then click “Export”. Save the steady-state snapshot file as an ASCII file (.asnp) in the current working folder. Change the RunMode back to “Dynamic” by selecting **Run → Mode → Dynamic** (this step is crucial) and then save the ACM file. Close the ACM window. Note: The snapshot file can be found in the installed “examples\VdV” folder.

- Open MATLAB and set the current folder to the “VdV” folder in the toolbar. Within the list of files on the left, double-click the “VdV\_Reactor.mdl” Simulink model file. A Simulink window as shown in Figure 5 displays. In the Simulink window, double-click the “AMSimulation” block.

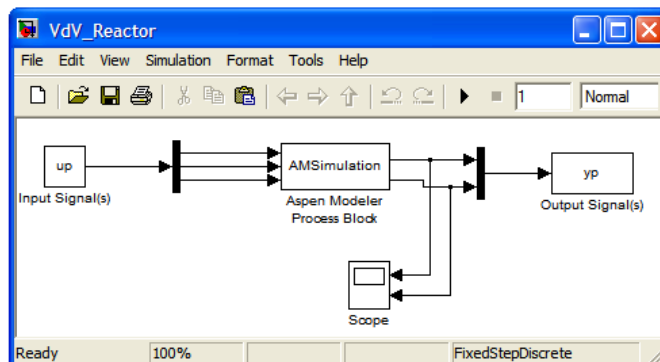


Figure 5: Simulink Flowsheet with Embedded ACM Model

In the “Configure AMSimulation Block” window as shown in Figure 6, browse to the “VdV\_Reactor.acmf” file, the same file that was used in Step 2. The ACM window also displays. Note: The path tied to the corresponding ACM file is absolute. The full path name of the ACM file in the example is “C:\Program File\CCSI\DRMBuilders\Examples\VdV\VdV\_Reactor.acmf”. If the D-RM Builder installation folder is “C:\Program File\CCSI\DRMBuilders”, the ACM window displays when the user opens the “VdV\_Reactor.mdl” file. Otherwise, browse and select the ACM file. This step must be repeated every time the ACM file location changes. Click the “Inputs” and “Outputs” tab and then confirm the correct variables are selected as shown in Figure 6.

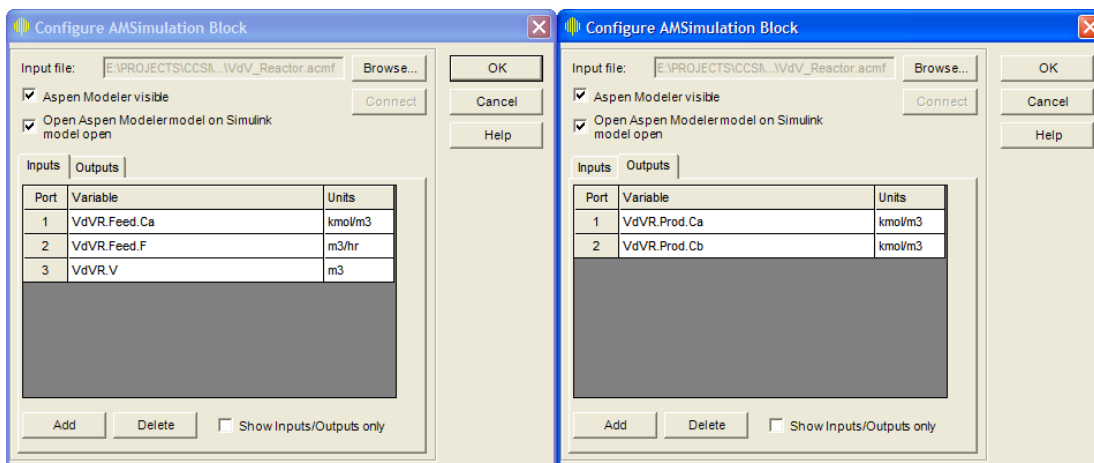


Figure 6: Simulink Windows for Setting Up Input and Output Variables

While the user should use the default selections of the input and output variables for the “AMSimulation” block in this tutorial, the user can try to add a new input or output variable through the user interfaces shown in Figure 6. Any fixed input variable in the ACM model can be selected as an input variable of the embedded custom block. Any free variable can be selected as an output variable of the block. If the number of inputs is changed and “OK” clicked, the user needs to double-click the “Demux” block before the “AMSimulation” block to set the “Number of outputs” of the “Demux” equal to the number of inputs of the “AMSimulation”. Likewise, if the number of outputs of the “AMSimulation” block is changed, the user needs to double-click the “Mux” block after the “AMSimulation” block to set the “Number of inputs” of the “Mux” equal to the number of outputs of the “AMSimulation” block. The user also needs to link the ports of the “Demux” and “Mux” blocks with the ports of the “AMSimulation” block by dragging-and-dropping the ports.

In this example, keep the default selections of the input and output variables. If the user has made any changes of the default selection, exit the MATLAB window without saving any files and then reopen the Simulink file or set the selections back to the default. Click “OK” to save the settings for the “AMSimulation” block and exit the “Configuration AMSimulation Block” window. Save and close the Simulink model. The model automatically closes the ACM window. Close the MATLAB window as well.

**Warning:** Do **not** close the ACM window manually; this may cause the Simulink–ACM linkage to fail and subsequent simulations may also fail. If the user has closed this window accidentally, repeat Step 4. If the user is unable to complete Step 4, refer to Section 5.2.

5. After the MATLAB/Simulink file is prepared, start the D-RM Builder by double-clicking the shortcut icon of the D-RM Builder on the desktop or from the “Start” menu. The main window of the D-RM Builder displays as shown in Figure 7.

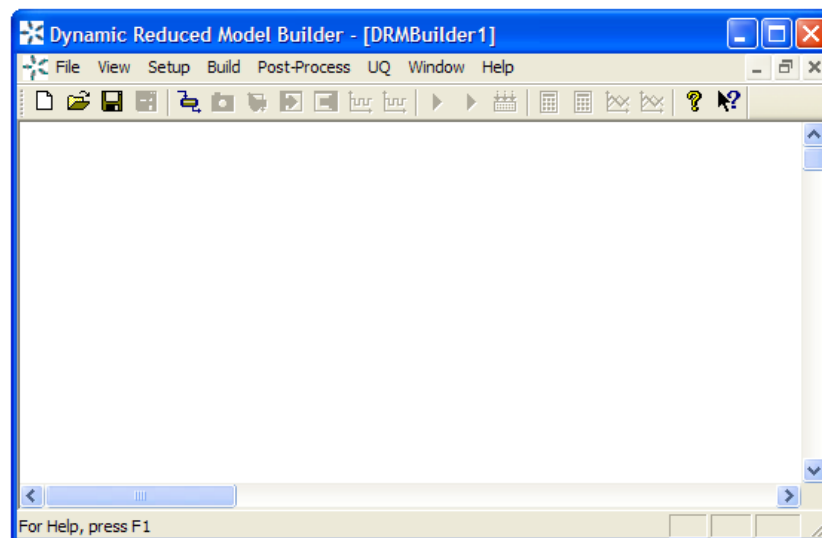










Figure 7: Main Window of the D-RM Builder

The default case name is “DRMBuilder1” as shown in the title bar of the main window. The GUI of the D-RM Builder is similar to a classical Microsoft Windows program with menus and a toolbar in the top part of the window, a white background client area in the middle, and a status bar at the bottom. The client area is used to display the messages. Browse the individual menus to see the submenus or commands. Some of the submenus and commands are shaded/unavailable. They are enabled later when the status of the D-RM building process is updated. The D-RM Builder is a multiple document application; multiple cases can exist in one application and are under one main window. If multiple cases are opened, the user can switch from one case to another through the case list under the “Window” menu. Meanwhile, multiple applications can be opened on the desktop, each of which could contain one or more cases. Each case should have a unique case name. Add a new case to the main window by issuing the **File → New** command or clicking  on the toolbar. Close the current case by issuing the **File → Close** command. If the data for the case has been modified, a pop-up message displays, asking the user to either save the case or discard the modifications. Open an existing case by issuing the **File → Open** command or clicking  on the toolbar. The user can also open an existing case by double-clicking the file name or icon in Windows Explorer. Save a case by issuing the **File → Save** command or clicking  on the toolbar. Save a case to a different name by issuing the **File → Save As...** command. The typical workflow for building and testing a D-RM is from the “Setup” menu, to the “Build” menu, and then finally to the “Post-Process” menu (from left to right). Within each menu, the general workflow is from the top command (submenu) to the bottom command. The “File” menu handles typical file opening, saving, and exporting commands, which could be issued at any time. Depending on the status of the D-RM building process, the user may or may not export certain files under the **File → Export** command. The toolbar icons are arranged from left to right following the typical D-RM building workflow. Note: The context-sensitive help command (  on the toolbar) is not fully implemented.

6. To start the D-RM building process for a blank case, select a high-fidelity model first. Select **Setup → Choose High-Fidelity Model → Aspen Custom Modeler File...** or click  on the toolbar to issue the command. An “Open” window displays for file browsing and selection. Browse and select the “VdV\_Reactor.acmf” file (the same file that was used in Steps 2–4). This loads the ACM-based high-fidelity model into the D-RM Builder. A text message with a time stamp confirming the file selection and loading displays in the client area of the D-RM Builder window.
7. Navigate to **Setup → Choose High-Fidelity Model → ACM Snap Shot File...** and then select the steady-state snapshot file created in Step 3. Alternatively, click  on the toolbar. This enables the D-RM Builder to find the steady-state values of the input and output variables exported from the ACM case in Step 3. A confirmation message displays in the client area of the window.

8. From the main menu select the **Setup → Choose High-Fidelity Model → MATLAB/Simulink File...** command and then select the Simulink model file “VdV\_Reactor.mdl” (which was saved in Step 4). Alternatively, click  on the toolbar to open the “Open” window. This file provides the interface for the D-RM Builder to run the MATLAB scripts and to communicate with ACM while performing high-fidelity model simulations. Note: The folder where the MATLAB/Simulink file is located is used by the D-RM Builder as the working directory, where temporary files are created during the D-RM building process. Confirm the user of the D-RM Builder has write-permissions in the directory.
9. Configure the input variables. Navigate to the **Setup → Configure Input Variables...** command or click  on the toolbar. The “Input Variable Dialog” window displays as shown in Figure 8.

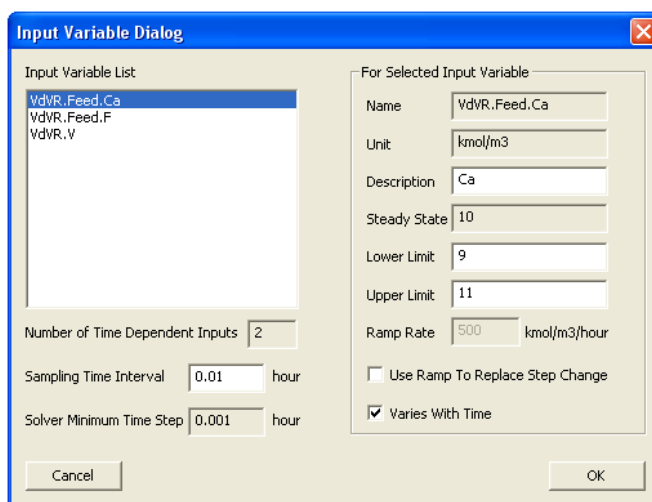



Figure 8: Input Variable Dialog Window

The D-RM Builder automatically imports all of the variables specified in the Simulink file along with their units and steady-state values (from the snapshot file). The input variables are listed in the “Input Variable List” list box. Click each of the variables to see the description, unit, and default ranges ( $\pm 10\%$ ) used for conducting training/validation. The “Name” and “Unit” text boxes are for read-only. If the input variable is dimensionless, a space is displayed in the “Unit” text box. The default description assigned by the D-RM Builder is the substring of the variable name string after the last dot. The user can edit the description as desired. This description is used for the labeling of future plots. For each input, define whether or not it is included in the D-RM. If excluded, the input variable is assumed to be constant and does not change with time. This can be useful for cases where the user wants to use an input as an unmodeled/unknown disturbance or where an input is a fixed parameter. In this example, the third input ( $VdV.V$ ) represents the volume of the reactor and is never going to change in the course of the dynamic runs. Therefore, specify the input as a time-invariant input by clearing the “Varies With Time” check box (make sure the other two inputs are selected). The total number of variables with their “Varies With Time” check box selected is displayed in the read-only “Number of Time Dependent Inputs” text box.

Another specification, the “Use Ramp To Replace Step Change” option, is not used in this example (keep this check box cleared for all three inputs). This option is used and discussed in the BFB example. Also notice that when the “Varies With Time” check box is cleared, the “Lower Limit”, “Upper Limit”, and “Ramp Rate” text boxes and the “Use Ramp To Replace Step Change” check box are disabled. When the “Varies With Time” check box is selected, the user can change the lower and upper limits of the variable to desired values. However, the lower limit cannot be larger than the steady-state value displayed at the read-only “Steady State” text box. Likewise, the upper limit cannot be smaller than the steady-state value. If this rule is broken, a warning message dialog box displays when the user clicks “OK” and the user cannot exit the dialog box unless the user clicks “Cancel”, which cancels all of the values and options that the user has entered on the Input Variable Dialog window. For this example, leave the default values of the lower and upper limits for the first two variables unchanged.

Depending on the time scale of the system response, the user may need to revise the sampling time interval, which is the time interval that is used in the discrete-time D-RM. Confirm the 0.01 hour is specified in the “Sampling Time Interval” text box. The default value shown corresponds to the fixed time interval specified by the user and saved in the Simulink file. The sampling time interval should be comparable with the time constants of the output variables in response to the change of input variables. This is discussed in other tutorials in this manual. The read-only “Solver Minimum Time Step” text box contains the minimum integration time step used by the ACM solver. The sampling time interval should be higher than the solver minimum time step. Click “OK” to accept the new configured values and close the “Input Variable Dialog” window. The messages in the client area confirm that the input variables are properly specified for the D-RM.

10. Configure the output variables by issuing the **Setup → Configure Output Variables...** command or clicking  on the toolbar. The “Output Variable Dialog” window as shown in Figure 9 displays.

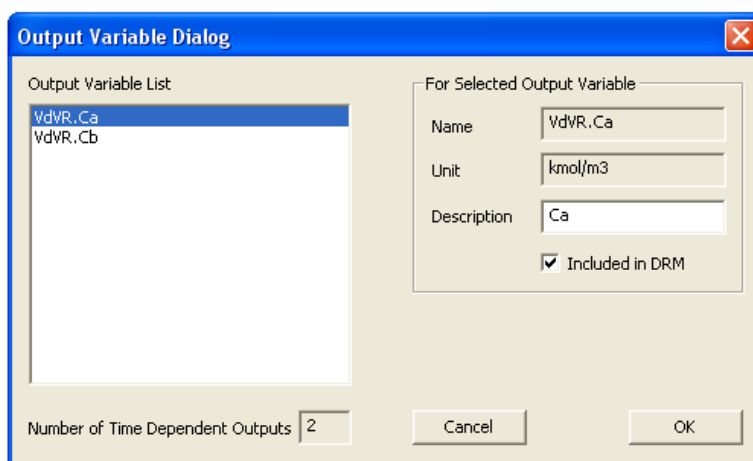



Figure 9: Output Variable Dialog Window



There are two output variables for the reactor. Select each output in the “Output Variable List” and then confirm the “Included in DRM” check box is selected. This means both output variables are predicted by the D-RM. Typically, include all of the output variables that change with time when any of the input variables changes. Remember that the user can select the variables of interest in Step 4. This window provides the user with another chance to eliminate any output variables that are not important. Click “OK” to accept the default values. A confirmation message displays in the client area of the main window.

11. Once input and output variables are configured, the user is ready to prepare a training sequence, a series of step changes of input variables with defined holding durations that are simulated by the high-fidelity model. To prepare the training sequence, select the **Setup → Prepare Training Sequence...** command or click  on the toolbar. The “Training Sequence Dialog” window displays as shown in Figure 10.

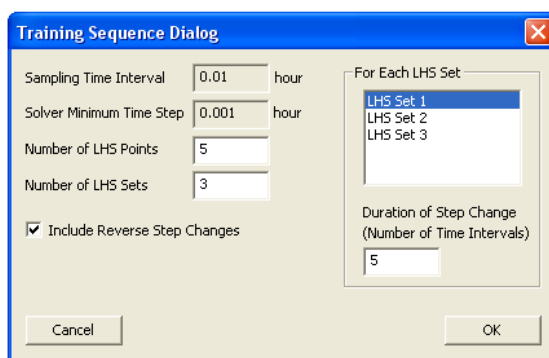

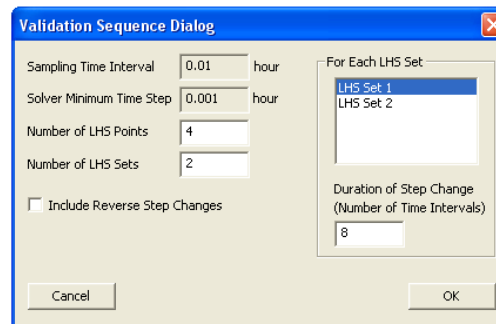


Figure 10: Training Sequence Dialog Window

The sampling time interval and solver minimum time step are displayed in the first two read-only text boxes for reference. The D-RM Builder uses the Latin Hypercube Sampling (LHS) method to fill the steady-state input space for input variables. A sequence of step changes of the input variables, one at a time, is obtained by moving from one steady-state point to the next steady-state point in a set of LHS. For each LHS point set, specify the number of steady-state points to be sampled. For this example, enter 5 in the “Number of LHS Points” text box. Within each LHS point set, the holding time between the step changes or the duration of the step change is the same. The durations for different LHS point sets should be varied such that different frequencies in a range can be excited. Increasing the number of LHS point sets makes the step change sequence longer and the number of training data points larger, which generally leads to a more accurate D-RM. It takes a longer CPU time to perform the ACM dynamic model simulation. For this example, enter 3 in the “Number of LHS Sets” text box and then press “Enter”. The number of items in the list box in the “For Each LHS Set” section is updated. Three items are listed for this example. Confirm that the “Include Reverse Step Changes” check box is selected. This means, the order of the step changes are reversed after the forward path from the first steady-state point to the last is completed, which makes the length of the step change sequence doubled. For each LHS set, a holding duration is assigned. The duration is measured as the number of sampling time intervals. Click the “LHS Set 1” item in the “For Each LHS Set” list box and then enter 5 in the “Duration of Step Change” text box. Since the time interval is 0.01 hour in this case, the


holding duration of 5 intervals corresponds to 0.05 hour of holding time. Select the “LHS Set 2” item in the list box and then enter 10 in the “Duration of Step Change” text box. Select the “LHS Set 3” item in the list box and then enter 15 in the “Duration of Step Change” text box. It is recommended to enter different durations for different LHS sets such that it covers a range of frequencies of interest. Click “OK” to accept the inputs. A confirmation message displays in the client area of the main window.

12. Prepare an input change sequence for validation by issuing the **Setup → Prepare Validation Sequence...** command or clicking  on the toolbar. The “Validation Sequence Dialog” window displays as shown in Figure 11.




**Figure 11: Validation Sequence Dialog Window**

The validation sequence is used as inputs for the high-fidelity model to simulate another set of response to validate the D-RM to be generated. This command can be issued before or after the D-RM is generated. In this example, create the step change sequence before the D-RM model is generated. Enter 4 in the “Number of LHS Points” text box. Note: The number of LHS points is not necessarily the same as the value used for training. Enter 2 in the “Number of LHS Sets” text box. Clear the “Include Reverse Step Changes” check box. Usually the sequence for validation is shorter than that for training to reduce the CPU time required to perform the high-fidelity model simulation. Therefore, use fewer LHS sets and exclude the reverse step change. Select the “LHS Set 1” item in the list box and then enter 8 for the duration of the first LHS set. Then select the “LHS Set 2” item in the list box and then enter 12 for the duration of the second LHS set. Note: The duration value for validation should be in the range of the training durations specified in the previous step. Click “OK” to accept the specifications. A confirmation text message displays in the client area of the main window.

13. Save the file by issuing the **File → Save** or **File → Save As...** command or clicking  on the toolbar. The “Save As” dialog window displays. Enter “VdV” in the “File Name” text box and then click “OK”. A binary D-RM Builder case file, “VdV.drm”, is written to the current working directory. This name also shows on the title bar of the main window, replacing the previous default name of “DRMBuilder1.drm”. The binary case file contains the contents of the ACM model file, ACM snapshot file, and MATLAB/Simulink file. The file also contains the inputs that have been entered. Copy the case file to any folder on the computer, open it via the D-RM Builder, and continue the D-RM building work there. If the user saves a case file after the user has selected the ACM and Simulink files, the user no longer needs to select them again since the case file contains the files of the high-fidelity model. If the user saves the case after completing

the high-fidelity model simulations, the results of the simulations are also saved in the case file. Likewise, if the user saves the case after a D-RM is generated, the data for the D-RM is also saved in the case file. The D-RM Builder also keeps track of the status of the building process and knows what stage the user is in. When the user opens an existing case file, the D-RM Builder brings the user to the stage that they left off at when the case was saved.

14. Launch the ACM high-fidelity model simulation for training by issuing the **Build → Perform Training Simulation** command or clicking  on the toolbar. This creates temporary files and a folder in the current working directory and brings up a MATLAB command window followed by an ACM window. The ACM simulation starts immediately. If the user receives a pop-up message stating “Initializing MATLAB engine failed!”, refer to Section 5.2 to fix the problem. The temporary files created in the current working directory include a MATLAB/Simulink file and an ACM file, both of which have time stamps appended to the file names. There are also three temporary MATLAB script files that are copied to the directory and used by the MATLAB engine during the high-fidelity model simulation. The ACM iteration message can be viewed inside the ACM window if the ACM’s “Simulation Message” window is turned on. Wait without closing the MATLAB command window or ACM window. After the high-fidelity model simulation is completed, a MATLAB plot window as shown in Figure 12 displays with the 2-D plots of input steps and output responses.

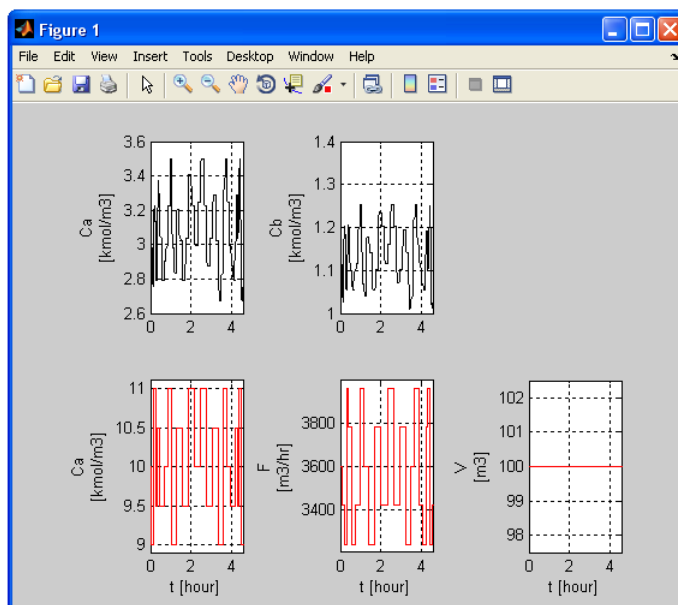



Figure 12: MATLAB Window Showing the Input Step Changes and Output Responses

A message box window displays confirming the successfully completion of the high-fidelity model simulation. After reviewing the plots, close the plot window and the MATLAB command window by clicking “OK” in the message box window. The temporary files and folder are then deleted.

15. Launch the ACM high-fidelity model simulation for validation by issuing the **Build → Perform Validation Simulation** command or clicking  on the toolbar. Note: The toolbar buttons for training are blue and those for validation are magenta. The Perform Validation Simulation command displays the MATLAB and ACM windows as described in Step 14. The message box confirming the successful completion of the simulation displays later. Click “OK” in the message box window to close the MATLAB windows and the ACM window.
16. Select DABNet as the D-RM model type by issuing the **Build → D-RM Model Type → DABNet** command and then confirming the “DABNet” pop-up submenu is checked as shown in Figure 13.

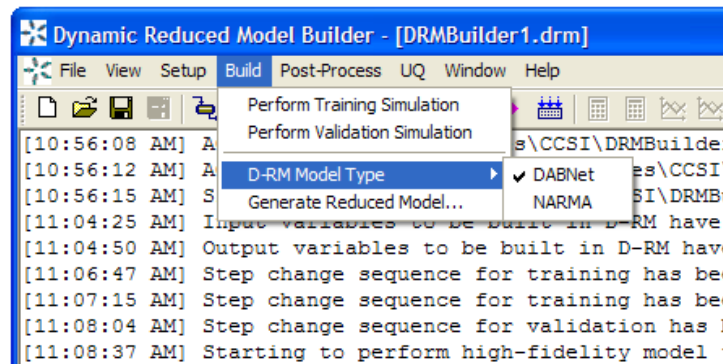



Figure 13: DABNet Submenu for Selecting a D-RM Model Type

Since the Decoupled A-B Net (DABNet) model is generated as the model type for this example, confirm the DABNet option is selected, which is the default option.

17. Build the DABNet model by issuing the **Build → Generate Reduced Model...** command or clicking  on the toolbar. The “DABNet DRM Parameter Dialog” window displays as shown in Figure 14.

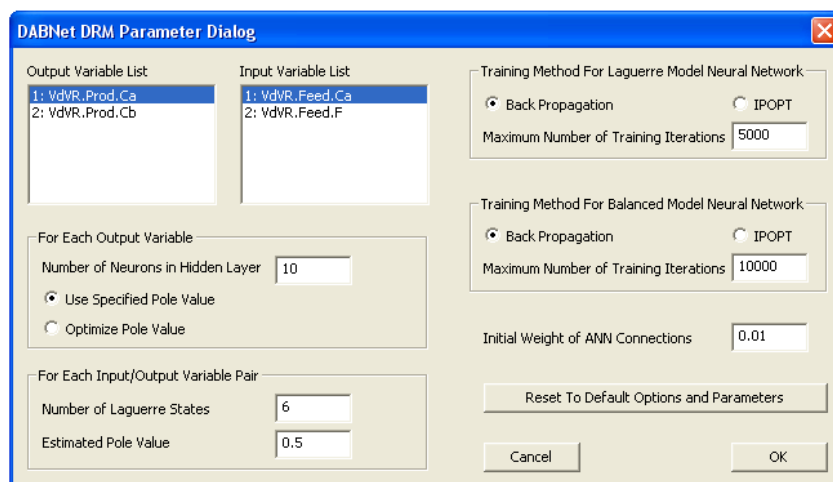





Figure 14: DABNet DRM Parameter Dialog Window

There is an “Output Variable List” list box and an “Input Variable List” list box in the upper left corner of the window. Each output variable has its own DABNet model. An array of the DABNet model is generated if multiple output variables exist. In this case, there are two output variables and, therefore, two DABNet models are generated. The model parameters for each DABNet model include number of Laguerre states and estimated pole value for each input, and the number of neurons in the hidden layer of the feed-forward neural network. Only one hidden layer is permitted in the current version of the D-RM Builder. Select an output variable in the “Output Variable List” list box and select the number of hidden neurons. Then for each input variable corresponding to a selected output or input/output pair, specify the number of Laguerre states and the pole value. The D-RM Builder enables the user to define a fixed pole value for each input/output pair or to ask the software to optimize the pole values for the individual inputs. Regarding the neural network training, there are two options available. The first option, the default option, is the back propagation (BP) method. The second option is Interior Point Optimization (IPOPT). The D-RM building process involves the generation of discrete-time process matrices A and B based Laguerre series and mapping the state space to the model outputs through a neural network, followed by a balanced realization of A and B matrices and mapping of balanced state space to the outputs through another neural network. Therefore, two neural network trainings are required for each output variable. Specify the training option and maximum number of training iterations for both Laguerre and balanced neural networks. The initial weights of ANN connections can also slightly affect the results of balanced realization. For this tutorial, use the default values for the model parameters and options by clicking “OK” to accept the defaults. The D-RM Builder starts the model generation process with multiple text messages displaying in the client area of the main window. Read the messages paying attention to the relative training errors reported, especially for the balanced model. A training error of less than  $10^{-4}$  (0.01%) indicates a good selection of DABNet model parameters. In this case the relative error of the DABNet model for the first output ( $C_a$ ) is approximately  $1.18 \times 10^{-5}$  (based on the balanced model) and that for the second output ( $C_b$ ) is approximately  $10^{-5}$ . Note: The user may get different numbers depending on the version of ACM that is being used and the operating system the D-RM Builder is installed on.

18. Save the case by issuing the **File → Save** command or clicking  on the toolbar. Since the dynamic reduced model has been generated, it is time to save the case setup, high-fidelity model simulation results, and the data of the generated D-RM to the case file.
19. To find how good the D-RM model prediction is compared to the high-fidelity model prediction, use the commands under the “Post-Process” menu. First confirm the “Use Balanced Model for Prediction” option under the “Post-Process” menu is selected. Usually this option is selected to examine the accuracy of the model. If this option is cleared, the unbalanced Laguerre models are used for D-RM predictions. Usually the unbalanced model predictions are not as accurate as the balanced model predictions, especially for the training sequence.

20. Examine the response for the training data. To predict the training response using the generated D-RM, issue the **Post-Process → Predict Training Response** command or click  on the toolbar. The D-RM Builder calculates the response to the training sequence of the input changes based on the generated D-RM. This command should be completed very quickly since the calculations for the D-RM based responses of the output variables are very fast. A text message displays in the main window indicating the completion of the calculation.
21. With both ACM and D-RM predictions for the response of the training sequence available, issue the **Post-Process → Plot/Compare Training Responses...** command or click  on the toolbar to visualize the results. The “Result Plotting Dialog” window as shown in Figure 15 displays.

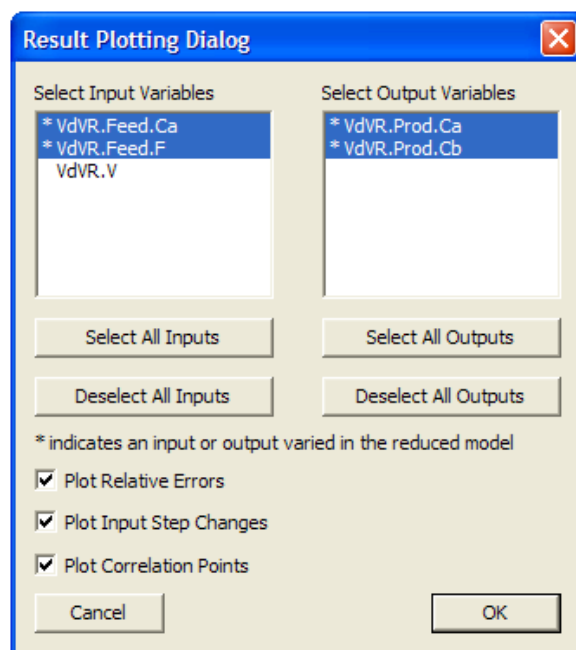


Figure 15: Result Plotting Dialog Window

The Result Plotting Dialog window lists the input and output variables for the user to plot by MATLAB through the MATLAB engine. Note: The variable names listed with an asterisk (“\*”) indicate the input or output variables that are varied (not fixed) in the reduced model. Click the items in the list boxes to select and deselect the variables. The four buttons below the two list boxes enable the user to select or deselect all of the input or output variables. Here the user can select the first two input variables and the first two output variables as shown in Figure 15. The D-RM Builder always plots the responses predicted by the generated D-RM and those by the ACM model in the first row of the figures. The user can also plot the relative errors, input step changes, and correlation points by selecting the corresponding check boxes in the lower part of the window. Accept the defaults with all of the check boxes selected and then click “OK”. A MATLAB command window displays, immediately followed by a MATLAB figure window as shown in Figure 16, and then by a message box window confirming the successful completion of the plotting command.



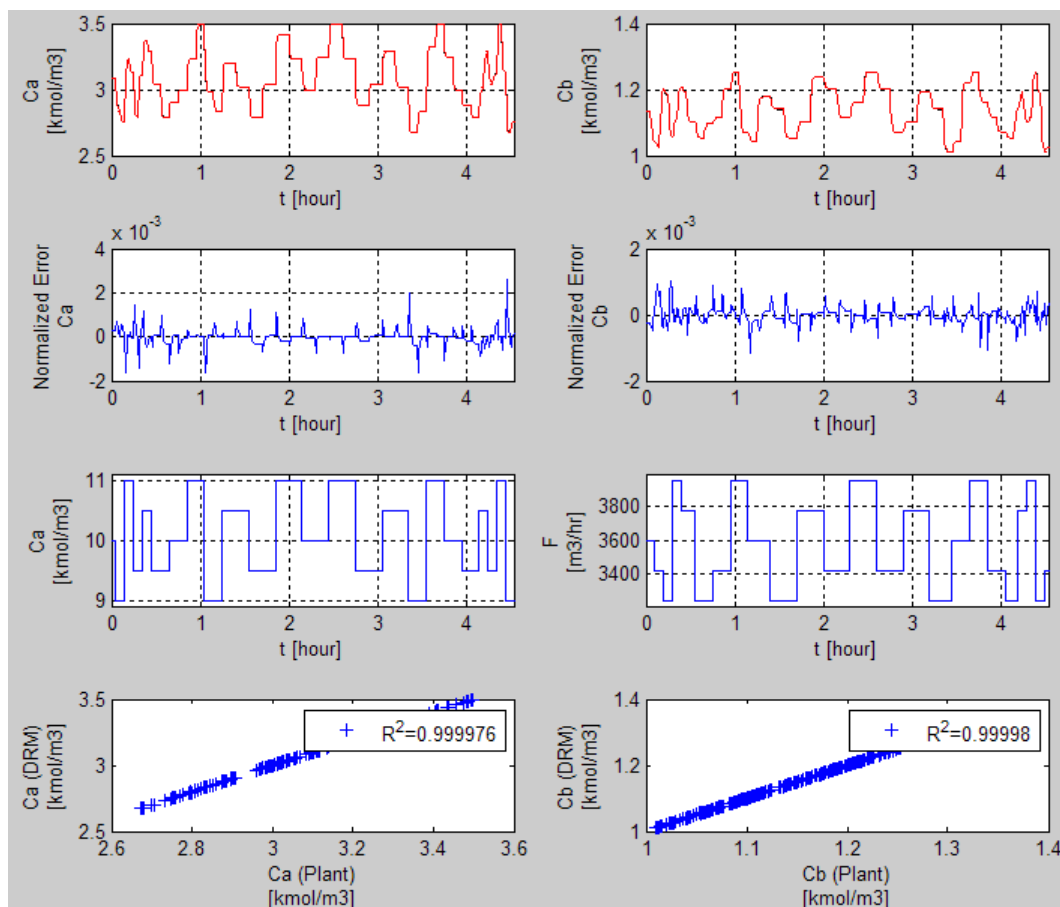




Figure 16: Plots of Input and Output Data as Functions of Time for Training Data

Maximize the plot window to see the plots of the input and output variables clearly. The plots in the top row are the curves of the output variables versus time. If the user looks carefully, there are two curves plotted for each output variable. The one in black is the high-fidelity model (ACM model) simulation result and the other in red is the D-RM prediction. The red curve seems to sit on the top of the black one unless the user zooms in to a selected region. The second row contains the normalized error plots showing the relative errors of the D-RM predictions compared to the ACM simulation results. Notice that the relative errors are very small in this case, indicating that the generated D-RM fits the ACM model results very well. The third row contains the plots of step changes of individual input variables, one input variable on each plot. If the user looks carefully, the step changes of two inputs do not happen at the same time. Actually the two inputs take turns to make step changes, one at a time. The bottom row plots the points of the D-RM model predictions versus the ACM model predictions (plant data). If a D-RM model prediction is exactly the same as the ACM model prediction, the point sits exactly on the 45 degree line if the scales of the two axes are identical. The legends of the bottom row figures also show the  $R^2$  values of the correlation. As seen in Figure 16, the  $R^2$  (coefficient of determination) values for the two outputs are very close to 1, indicating that the D-RM is quite accurate. Note:  $R^2$  is always less than 1. Generally, the higher the  $R^2$  value, the more accurate the generated D-RM. Close the plot and MATLAB

command windows after examining the results for the training sequence by clicking “OK” in the message box window. If needed, issue the **Post-Process → Plot/Compare Training Responses...** command again with only one output variable selected at a time to visualize the plot in a single column. Likewise, clear some of the check boxes in the Result Plotting Dialog window shown in Figure 15 to remove some of the rows of figures so the user can see the larger figures of interest.

22. Examine the response for the validation data. Usually the generated D-RM can predict the response of the training sequence very close to that from the high-fidelity model simulation since the reduced model is trained using the training data. A good D-RM should also be able to predict the response of the validation data quite well. To predict the validation response using the generated D-RM, issue the **Post-Process → Predict Validation Response** command or click  on the toolbar.
23. Issue the **Post-Process → Plot/Compare Validation Responses** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 15 displays. Select the first two input variables and the first two output variables and click “OK”. A MATLAB command window displays immediately followed by the plot window as shown in Figure 17 and then a confirmation message window.

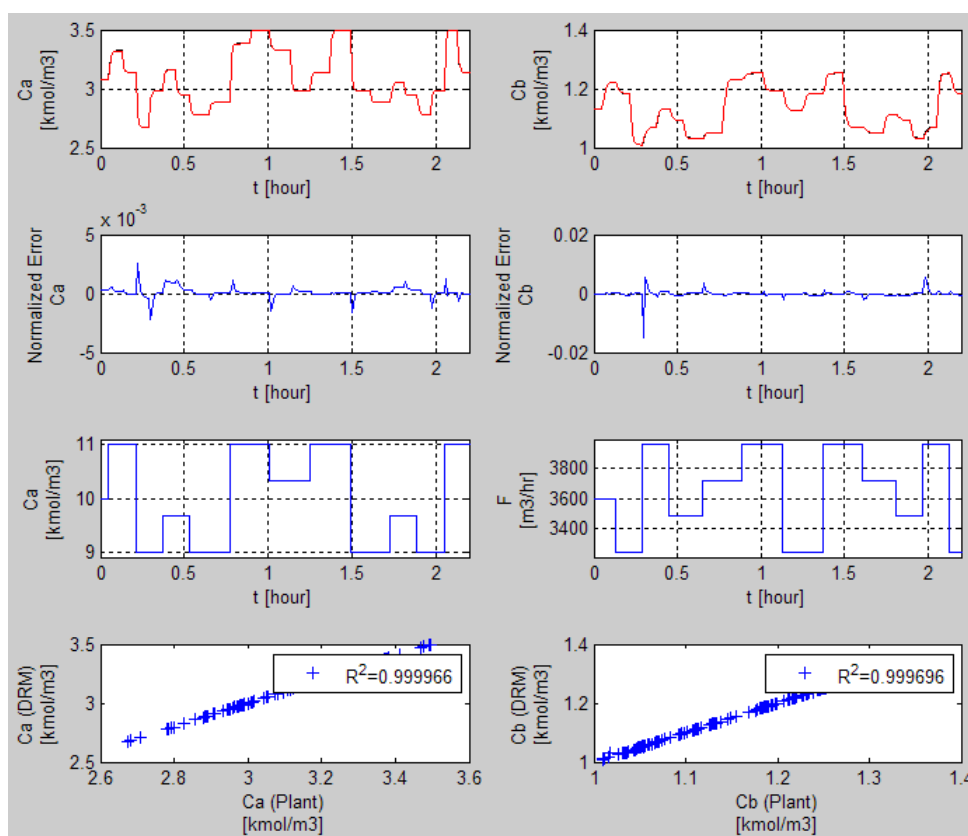



Figure 17: Plots of Input and Output Data as Functions of Time for Validation Data

Notice that the red curves are still on the top of the black curves for both the  $C_a$  and  $C_b$  outputs, indicating a quite accurate D-RM generated by the D-RM Builder.



24. Save the file by clicking  on the toolbar. The case file contains the data predicted by the D-RM generated for both the training and validation sequences.
25. Perform the Uncertainty Quantification (UQ) analysis on the generated D-RM, using the ACM predictions as the plant data. Note: The UQ commands are available only when the DABNet model option is selected. For the UQ analysis, it is assumed that the process has certain noise in terms of standard deviation of state-space variables and the measurements of output variables also have noise in terms of standard deviations. During the UQ analysis, the measured output values are obtained from the ACM prediction with the noise term added based on the standard deviation provided by the user. To setup the noise levels of the state and output variables, issue the **UQ → Set Noise...** command. The “Process and Measurement Uncertainty Dialog” window as shown in Figure 18 displays.

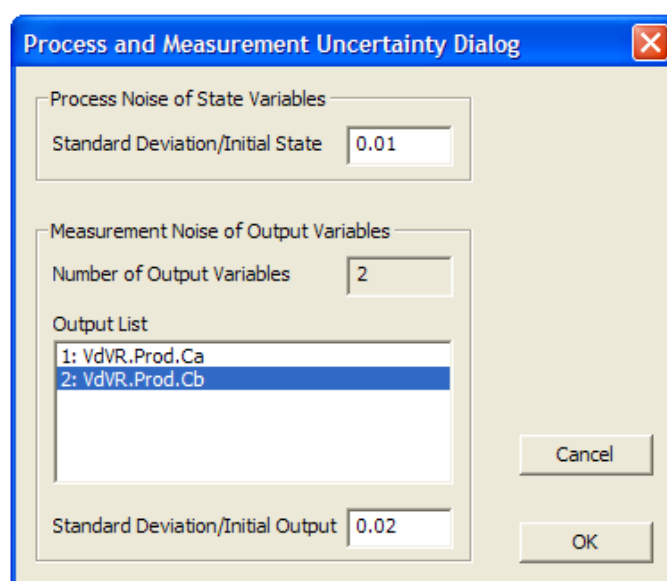


Figure 18: Process and Measurement Uncertainty Dialog Window

Specify the ratios of standard deviations to the initial values of the state and output variables. Note: The initial conditions are provided by the snapshot file loaded to the D-RM Builder in Step 7. In the current version, the ratio of the standard deviation to the initial value of a state variable is assumed to be identical for any state variables, which is not necessarily true for a real-world process. The identification of process noise has not been built in to the D-RM Builder. The user can specify a single number for the noise level. For this tutorial, use the default value of 0.01 in the “Standard Deviation/Initial State” text box in the “Process Noise of State Variables” section. This means that the standard deviation of every state-space variable is 1% of its initial steady-state value. The measurement noises can be specified separately, one for each output variable. These noises are related to the sensitivity of the sensors and the electronics used to measure the output variables. For this tutorial, select the first output variable “1: VdVR.Prod.Ca” in the “Output List” and then enter 0.02 in the “Standard Deviation/Initial Output” text box. Then, select the second variable “2: VdVR.Prod.Cb” and then enter 0.02 as shown in Figure 18. Click “OK” to accept the noise specifications.

26. Indicate to the D-RM Builder to run the UQ analysis on the generated DABNet model following the validation step-change sequence. The algorithm of Unscented Kalman Filter (UKF) is used to predict the state variables step by step using the D-RM and update the state and output variables using the measurement data, which is calculated by adding the random noise to the ACM predictions. The mean values of the state and output variables are called “filtered” values, which are the weighted averages by considering both the D-RM predictions and measurements. The Unscented Transform (UT) calculations are performed to calculate the means and covariances for the non-linear functions involved. Multiple samples known as the sigma points around the means are used to evaluate the non-linear functions in the UT calculation. The UT approach is proved in the literature to be much faster than Monte Carlo sampling with reasonable accuracy. The covariance matrices are also updated step by step. The elements in the covariance matrices tend to reach their asymptotic values after a certain period of time along the step-change sequence. To run the UQ analysis, issue the **UQ → Calculate...** command. The “Result Plotting Dialog” window as shown in Figure 19 displays, asking the user to specify the options for plotting the results of the UQ analysis.

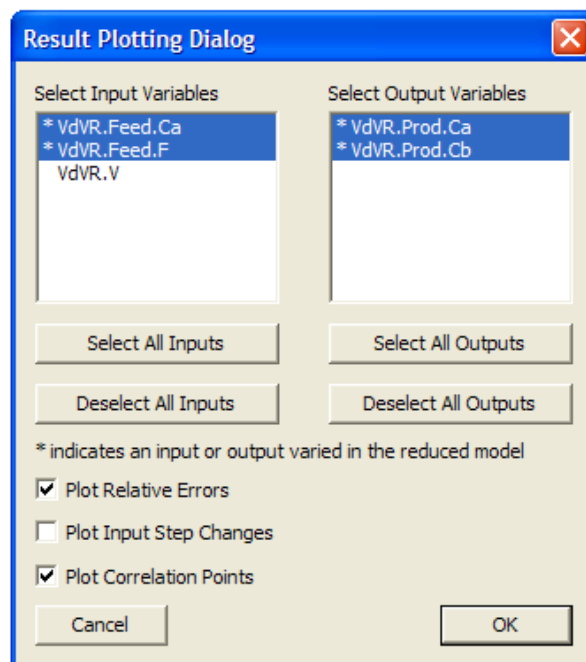


Figure 19: Result Plotting Dialog Window

Since the volume of the reactor is constant, the user can select only the first two inputs in the “Select Input Variables” section on the window. Select both output variables in the “Select Output Variables” section. This time clear the “Plot Input Step Changes” check box if the user is not interested in the input versus time plots. Click “OK”, the window closes, and the UQ calculation begins. The calculation usually takes a while to complete since UKF algorithm is quite time consuming. After the calculation is completed, the results are plotted in a pop-up window as show in Figure 20. A separate confirmation message window displays indicating the successful completion of the UQ calculation.

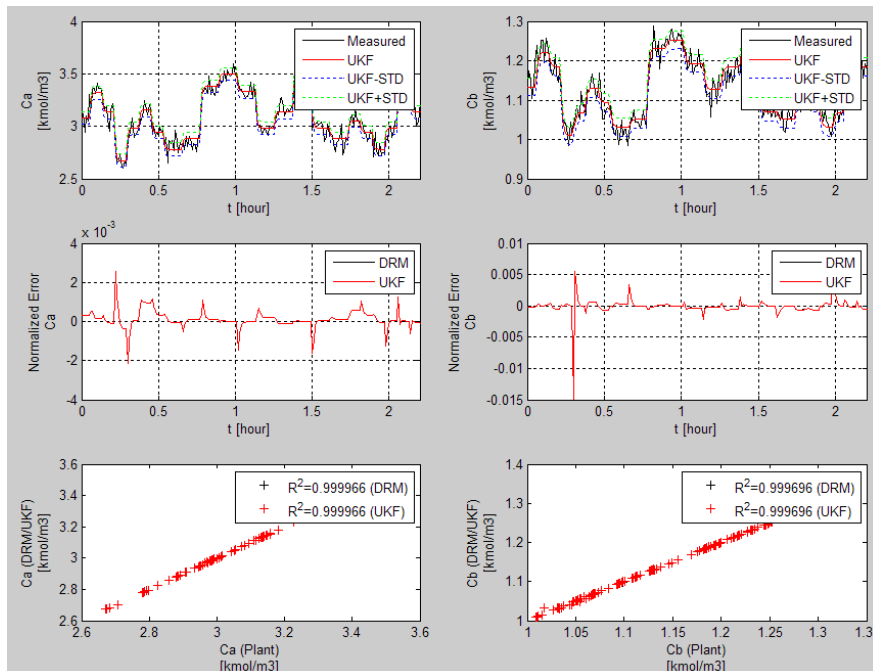




Figure 20: Plots Showing the Results of UQ Analysis for the Validation Sequence

Adjust the size of the plot window before clicking “OK” on the message window, which closes both the plot and the message windows. The first row of the plot window contains the two output variables (concentrations of Species A and B) as functions of time. The black curves show the measured values which are calculated based on the ACM predictions and the randomly imposed measurement noise. The red curves are the filtered mean output values. The blue and green dotted curves show the error ranges ( $\pm$  the standard deviation from the calculated covariance matrix). The relative errors and correlation plots are similar to those shown in Figure 17 except that both the D-RM and UKF results are plotted. Since the generated D-RM is very accurate, the standard deviations of the output variables are dominated by the noise of the measurements, which is 2% of the initial values.

27. Save the case by clicking  on the toolbar. Export the covariance matrices at the end of the validation sequence by issuing the **File → Export → Covariance Matrices...** command. The “Save As” dialog window displays. Click “Save” to accept the default name for the file. The exported matrices are written in simple text format.

28. Export the generated D-RM to a MATLAB script file by issuing the **File → Export → D-RM As Matlab Script File...** command or clicking  on the toolbar. The “Save As” dialog window displays. Use the default file name “VdV\_DRMParameters.m” and then click “OK” to save the D-RM to the “VdV\_DRMParameters.m” file. This file is the main output file of the D-RM Builder and can be used by a MATLAB programmer for dynamic simulations. The usage of this file is given in another tutorial in this manual.
29. Save the log messages that have been displayed in the client area of the main window by selecting the **File → Export → Log File...** command. The “Save As” dialog window displays. Use the default file name “VdV.log” to save the log file. The log file contains the information regarding the training errors and the optimized parameters during the reduced model building process, if applicable. The information can be examined later when comparing to a new case with new user inputs.
30. Save the actual input and output sequence data to a comma separated value (.csv) file by issuing the **File → Export → Training Data...** command and the **File → Export → Validation Data...** command for training data and validation data, respectively. The data contains the step changes of inputs versus time and the responses predicted by the high-fidelity model. If the D-RM predictions are available, they are also included in the exported file.
31. If the user has changed the minimum integration time step in the original ACM model, which is not the case in this tutorial, save the modified ACM model to a file by issuing the **File → Export → ACM File...** command. Likewise, save the ACM’s snapshot file by issuing the **File → Export → ACM Snapshot File...** command.
32. If the user has changed the sampling time interval in the Simulink file, save the current .mdl file by issuing the **File → Export → Simulink File...** command. Since the Simulink file contains the path of the ACM model, a pop-up window displays enabling the user to browse the local drives to select a valid ACM file.
33. Close the main window of the D-RM Builder to complete the model generation process.


## 2.3. Building Data-Driven D-RM using DABNet Framework and IPOPT-Based ANN Training

### Description

In the previous example, the default back propagation method was used to train the neural networks inside the DABNet model. The D-RM Builder provides an alternative method, the interior point optimization (IPOPT) method, to train the neural networks. IPOPT is a general optimization algorithm to minimize an objective function as long as the code to evaluate the objective function, Jacobian matrix, and Hessian matrix is provided. IPOPT is implemented in the D-RM Builder as an alternative to train neural networks in which the objective function is the sum of the errors squared at all discrete time points between the high-fidelity model prediction and the D-RM prediction. Usually, the IPOPT method takes more CPU time than the default back propagation method. Sometimes the IPOPT training method could minimize the training error (the objective function) to a lower value or generate a balanced model with fewer state-space variables than the back propagation method. This tutorial demonstrates the usage of this option.

### Example

The Van-de-Vusse reactor model created in the previous tutorial is used in this tutorial. To open the previous case file “VdV.drm”, double-click the file name or icon. This displays the main window of the D-RM Builder. The following steps are to generate the DABNet model using the IPOPT training algorithm.

1. Save the case of the previous tutorial with a different name by selecting the **File → Save As...** command and then saving the case as “VdV\_IPOPT”. Save the file to the same working directory or to a different working directory. Note: If the file is saved to a different directory, the user does not need to copy the ACM and MATLAB/Simulink files to the directory since the D-RM Builder case file with the extension “.drm” contains the contents of the ACM and MATLAB/Simulink files if those files have been loaded to the D-RM Builder before the case file is saved.
2. Select the **Build → Generate Reduced Model...** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 14 displays. This time select the “IPOPT” option in both the “Training Method For Laguerre Model Neural Network” section and the “Training Method For Balanced Model Neural Network” section. The IPOPT method can also be used to train the Laguerre model only and the BP method used to train the balanced model. Notice that the values in the text boxes of “Maximum Number of Training Iterations” of both sections have changed. These are the default values for the IPOPT training. Since the IPOPT method is much slower than the back propagation method, the default maximum iteration numbers set by the D-RM Builder are smaller. The user can increase the maximum iteration numbers. For this tutorial, keep the default numbers. Click “OK” in the window to start the model generation process. Text messages display in the client area of the main window. The mouse icon is switched from an arrow icon to a “busy” icon. Be patient since the process to training the neural networks is slow. Upon completion, a text message displays indicating that the D-RM has been generated and the mouse icon is switched back from the “busy” icon to

the standard ready icon (arrow). Take a look at the text message related to the state-space size for the balanced model. Notice that the training errors with the IPOPT option are slightly higher than those with the default back propagation option, probably because the maximum number of iterations in the IPOPT case are smaller. However, this is not always the case. For some non-linear cases, the IPOPT method could generate a better D-RM with smaller state-space size and higher accuracy.

3. Use the commands under the “Post-Process” menu to predict the training and validation responses using the new D-RM and visualize the results. If the response of the training sequence is plotted, the relative errors are slightly larger than those with the default back propagation method. However, if the response of the validation sequence is plotted, the maximum relative error for the second output variable is actually smaller in the IPOPT case.
4. Perform UQ analysis by following the same procedure in the first tutorial to set the process and measurement noises and run the UQ analysis using the commands in the “UQ” menu.
5. Export the new D-RM to a file with an “.m” extension. Save the case file and then close the D-RM Builder application.

## 2.4. Building Data-Driven D-RM using DABNet Framework with Pole Values Optimized

### Description

Ideally, the pole value for each input/output pair in the DABNet model should match the time constant of the system for the output with respect to the input. It is recommended by Sentoni et al., 1998 [1] that the pole value  $a$  should be set as:

$$a = 1 - \frac{T_s}{\tau}$$

where  $T_s$  is the sampling time interval set through the **Setup → Configure Input Variables...** command and  $\tau$  is the time constant of the output with respect to the input. Note: The time constants are generally different for an output with respect to different inputs. Therefore, different pole values should be used. The exact value of the time constant for an input/output pair is hard to determine although a user can estimate that by running the high-fidelity model simulations. Fortunately, the DABNet model is generally robust with respect to the user's selection of pole values. A default value of 0.5 for each input/output pair could result in a D-RM with reasonable accuracy. However, for some strongly non-linear system, the accuracy of the generated D-RM is quite sensitive to the pole values especially when the selected sampling time interval does not match the time constant (the pole value  $a$  calculated from the equation listed above is much higher or much lower than 0.5). The D-RM Builder can optimize the pole values by minimizing the training error of the unbalanced Laguerre model. Note: The optimization routine does not guarantee the global optimum and the user could provide the initial guess through the GUI of the D-RM Builder and the optimization is performed around the initial guess. If the user has no idea about the reasonable initial guess, the default value of 0.5 is recommended. This tutorial demonstrates the usage of the pole value optimization option.

### Example

To demonstrate the feature, an example named “pH\_Neut” is provided in a subfolder named “C:\Program Files\CCSI\DRMBuilder\Examples\pH” under the D-RM Builder's installation directory. This example models the two-tank pH neutralization reactor described in Reference 1. The default options are first be used to build a D-RM. Then the user selects the pole value optimization option to see the improvement of the D-RM over the default option. The DABNet model is still used as the D-RM model type. The following are the steps to build data-driven D-RM using DABNet framework with pole values optimized:

1. Copy the entire example subfolder of the pH-Neutralization reactor “C:\Program Files\CCSI\DRMBuilder\Examples\pH” to a desired location. Confirm the user of the D-RM Builder has write-permissions to this folder. Browse to the “pH” subfolder.
2. Open the “pH\_Neut.acmf” file by double-clicking the file name or icon. The pH Neutralization Reactor model opens by ACM.
3. Make a steady-state run of the model. For the D-RM Builder to work, the initial states should be at steady-state. From the ACM menu, select **Run → Mode → Steady-State** (or in the Run Control toolbox select “Steady State” from the RunMode drop-down list).

Make a steady-state run by clicking “Run” on the toolbar or pressing F5. Once the “Run Complete” dialog window displays, click “OK”. Navigate to **Tools → Snapshot**. In the “Snapshot Management” window, select the most recent steady-state run (at the top of the list) and then click “Export”. Save the steady-state snapshot file as an ASCII file (pH\_Neut.asnp) in the current working folder. Change the RunMode back to “Dynamic” by selecting **Run → Mode → Dynamic** and then save the ACM file. Close the ACM window.

Open MATLAB by double-clicking the “pH\_Neut.mdl” file. A Simulink window as shown in Figure 21 displays.

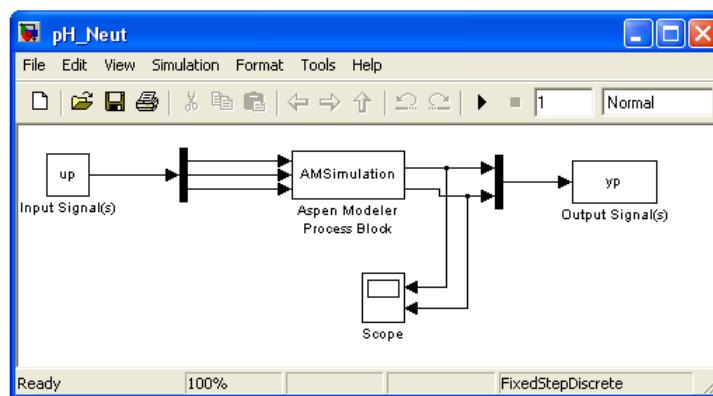


Figure 21: Simulink Flowsheet with Embedded ACM Model

Note: If the D-RM Builder installation folder is located in the “C:\Program Files\CCSI\DRMBuilder” folder, an ACM window also displays. In the Simulink window, double-click the “AMSimulation” block. A window displays for the user to select an ACM file. Browse to the current working directory and then select the pH\_Neut.acmf file. An ACM window displays if it has not been opened yet and the “Configure AMSimulation Block” window displays. Click the “Inputs” and “Outputs” tabs of the latter and then confirm the correct variables are selected as shown in Figure 22.

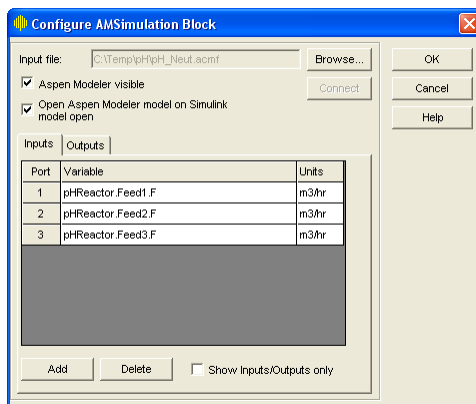




Figure 22: Simulink Dialog Window for Setting Up Input and Output Variables



In this example, three input and two output variables are selected. The three input variables are the volumetric flow rates of three feed streams to the first reactor. The two output variables are the product stream pH value and the concentration of Species A.

Click “OK” to save the settings for the “AMSimulation” block and exit the Configuration window. Save and close the Simulink model. This automatically closes the ACM window. Close the MATLAB window as well.

4. After the MATLAB/Simulink file is prepared, start the D-RM Builder by double-clicking the shortcut icon of the D-RM Builder on the desktop or by the “Start” menu. A blank D-RM Builder window displays.
5. From the main menu, select **Setup → Choose High-Fidelity Model → Aspen Custom Modeler File...** or click  on the toolbar to issue the command. An “Open” window displays for file browsing and selection. Browse to the directory in which the three files are located and then select the “pH\_Neut.acmf” file. This loads the ACM-based high-fidelity model into the D-RM Builder.
6. Navigate to **Setup → Choose High-Fidelity Model → ACM Snap Shot File...** and then select the steady-state snapshot file “pH\_Neut.asnp”.
7. From the main menu select **Setup → Choose High-Fidelity Model → MATLAB/Simulink File...** and then select the Simulink model file “pH\_Neut.mdl”. Note: The file directory where the MATLAB/Simulink file is located is used by the D-RM Builder as the working directory, where temporary files are created during the D-RM building process.
8. Configure the input variables. Navigate to **Setup → Configure Input Variables...** or click  on the toolbar. The “Input Variable Dialog” window displays as shown in Figure 23.

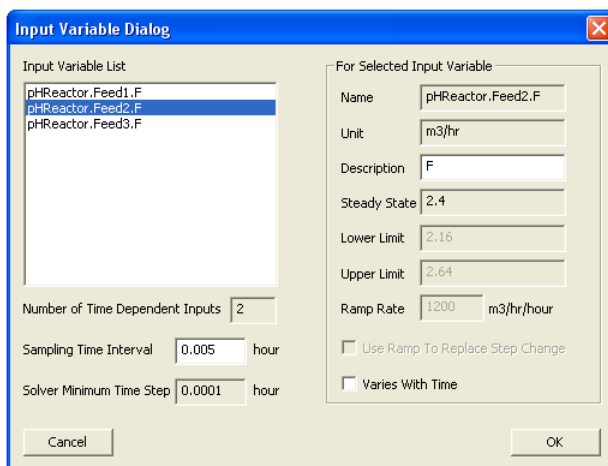












Figure 23: Input Variable Dialog Window

For this example, leave the default values of the lower and upper limits for the first and third variables unchanged. Click the “pHReactor.Feed2.F” item in the “Input Variable List” list box and then clear the “Varies With Time” check box. This makes the second

- input variable fixed (not changing with time). Click “OK” to accept the modifications and exit the window.
9. Configure the output variables by issuing a command of **Setup → Configure Output Variables...** or clicking  on the toolbar. Accept the default options by clicking “OK”.
  10. To prepare the training sequence, select **Setup → Prepare Training Sequence...** command or click  on the toolbar. The “Training Sequence Dialog” window displays. For this example, enter 5 in the “Number of LHS Points” text box and then enter 3 in the “Number of LHS Sets” text box. Three rows show in the “For Each LHS Set” list box. Select the first row “LHS Set 1” item in the list box and then enter 5 in the “Duration of Step Change” text box. Select the second row “LHS Set 2” item in the list box and then enter 10 in the “Duration of Step Change” text box. Select the third row “LHS Set 3” item in the list box and then enter 15 in the “Duration of Step Change” text box. Confirm that the “Include Reverse Step Changes” check box is selected. Click “OK” to accept the user inputs.
  11. Prepare an input change sequence for validation by issuing the **Setup → Prepare Validation Sequence...** command or clicking  on the toolbar. The “Validation Sequence Dialog” window displays. Enter 4 in the “Number of LHS Points” text box. Leave all of the other entries in the window unchanged. Click “OK” to accept the change and then close the window.
  12. Save the file by issuing the **File → Save** or **File → Save As...** command or clicking  on the toolbar. The “Save As” dialog window displays. Enter “pH\_Neut” in the “File Name” text box and then click “OK”. A binary D-RM Builder case file named “pH\_Neut.drm” is written to the current working directory. This name is also show on the title bar of the main window, replacing the previous default name of “DRMBuilder1.drm”.
  13. Launch the ACM high-fidelity model simulation for training by issuing the **Build → Perform Training Simulation** command or by clicking  on the toolbar. This creates temporary files and a folder in the current working directory and displays a MATLAB command window followed by an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACM’s “Simulation Message” window is turned on. Wait without closing the MATLAB command window or ACM window. After the high-fidelity model simulation is completed, a MATLAB plot window displays with the 2-D plots of input steps and output responses. A message box window confirming the successfully completion of the high-fidelity model simulation displays. After reviewing the plots, close the plot window and the MATLAB command window by clicking “OK” in the message box window. The temporary files and folder are deleted. The simulation results are automatically saved to the memory of the D-RM Builder.
  14. Launch the ACM high-fidelity model simulation for validation by issuing the **Build → Perform Validation Simulation** command or by clicking  on the toolbar b. The MATLAB and ACM windows display. A message box window displays after the simulation is completed. Click “OK” in the message box window to close the MATLAB windows and the ACM window.
  15. Select DABNet as the D-RM model type by issuing the **Build → D-RM Model Type → DABNet** command and then confirm the “DABNet” pop-up submenu is selected.

16. Build the DABNet model by issuing the **Build → Generate Reduced Model...** command or clicking  on the toolbar. The “DABNet DRM Parameter Dialog” window displays. Click “OK” in the window to accept all of the default options. The default pole value is 0.5 for every input/output pair and the default neural network training method is back propagation. Examine the text messages displayed in the main window, to find that the relative error for training the balanced model of the first output variable is approximately 0.00049 while the error for the second variable is approximately 0.0145. These errors are quite high since the default pole values of 0.5 are not good for the dynamic process.
17. Save the case by issuing the **File → Save** command or clicking  on the toolbar. Since the dynamic reduced model has been generated, it is time to save the case setup and model data to the case file.
18. To find how good the D-RM model prediction is compared to the high-fidelity model prediction, use the commands under the “Post-Process” menu. Confirm the “Use Balanced Model for Prediction” option under the “Post-Process” menu is selected.
19. Examine the response for the training data. To predict the training response using the generated D-RM, issue the **Post-Process → Predict Training Response** command or click  on the toolbar.
20. Issue the **Post-Process → Plot/Compare Training Responses...** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 24 displays.

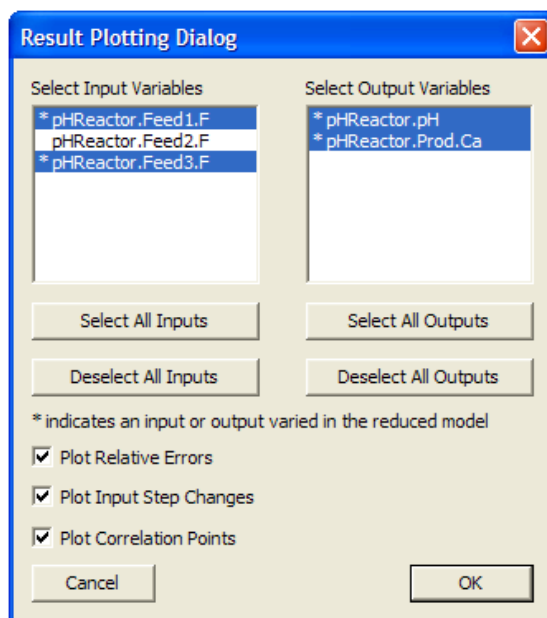


Figure 24: Result Plotting Dialog Window

Select the first and the third input variables (the variables that are marked with asterisks), both of the output variables, and then click “OK”. A MATLAB command window displays, immediately followed by the plot window as shown in Figure 25, and then by a message box window confirming the successful completion of the plotting command.

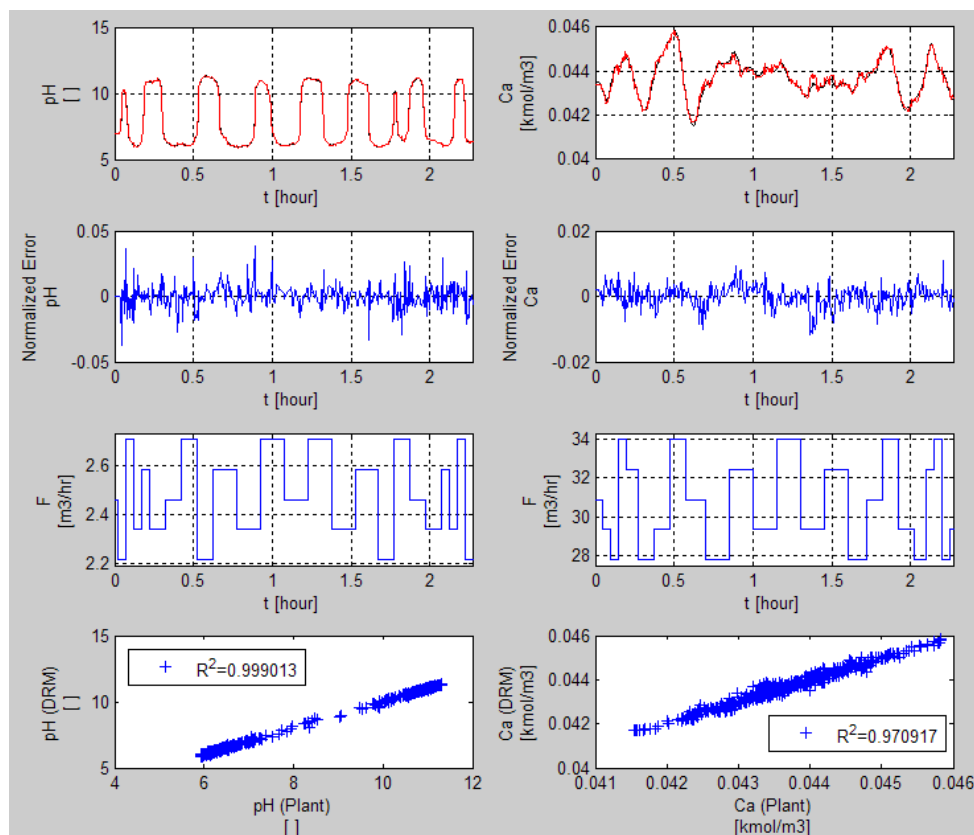




Figure 25: Predicted Response of Training Sequence using Default Pole Values

Maximize the plot window to see the plots of the output variables clearly. If the user looks carefully, there are two curves plotted for each output variable. The one in black is the high-fidelity model (ACM model) simulation result and the other in red is the D-RM prediction. The user can find the mismatch between the two curves especially for the second variable ( $C_a$ ) with a low  $R^2$  value around 0.97. The normalized error plots also show relative errors over 2% in some regions. Close the plot and the MATLAB command windows after examining the results for the training sequence by clicking “OK” in the message box window. If needed, issue the **Post-Process → Plot/Compare Training Responses...** command again with only one output variable selected at a time to visualize the plot in a higher resolution.

21. Examine the response for the validation sequence. To predict the validation response using the generated D-RM, issue the **Post-Process → Predict Validation Response** command or click  on the toolbar.

22. Issue the **Post-Process → Plot/Compare Validation Responses...** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 24 displays. Select the first and the third input variables and all of the output variables and then click “OK”. A MATLAB command window displays immediately followed by the plot window as shown in Figure 26 and then a confirmation message window.

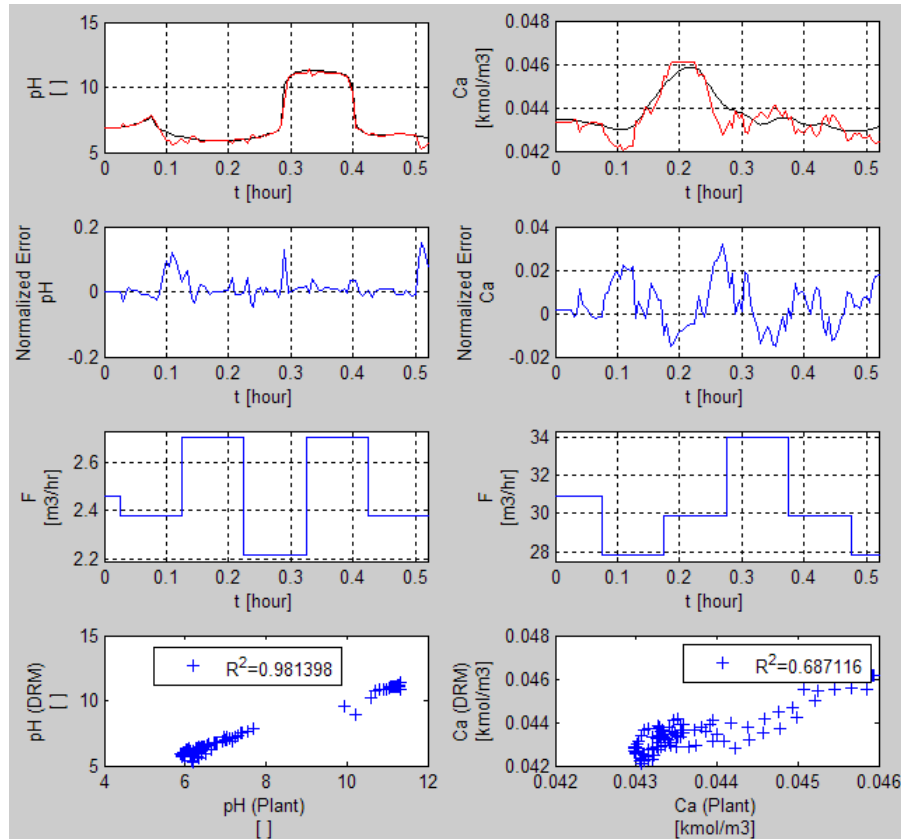



Figure 26: Predicted Response of Validation Sequence using Default Pole Values

Notice that the relative errors of the D-RM predictions are quite high and the  $R^2$  values are quite low, indicating an inaccurate D-RM generated by the D-RM Builder. Click “OK” in the message box window to close the MATLAB windows.

23. Try to improve the D-RM accuracy by optimizing the pole values. Build the D-RM again by issuing the **Build → Generate Reduced Model...** command or clicking  on the toolbar. The “DABNet DRM Parameter Dialog” window displays. This time, turn on the pole value optimization option. In the “Output Variable List” list box, click the first output variable and then click the “Optimize Pole Value” option in the “For Each Output Variable” section. Repeat this for the second output variable. Click “OK” in the window to accept the option changes. The D-RM Builder starts the D-RM generation process for both output variables, one at a time. During the generation process, different pole values are tried to minimize the neural network training errors for the Laguerre models. The text messages are displayed in the main window. Export these messages to a log file by issuing the **File → Export → Log File...** command. Examine the text messages, and find that the optimized pole values for the first output variable are 0.554102 and

0.642578 for the first and second input variables, respectively and the optimized pole values for the second output variables are 0.539063 and 0.84375. The relative error for training the balanced model of the first output variable is approximately 0.00015 while the error for the second variable is approximately  $3.5 \times 10^{-5}$ . These errors are much smaller than those when the default pole values are used.

24. Repeat Steps 19–22 to examine the predictions by the new D-RM. Notice that the predicted responses by the new D-RM are in much better agreement with the responses predicted by the ACM model. Figure 27 shows the comparison of the training data.

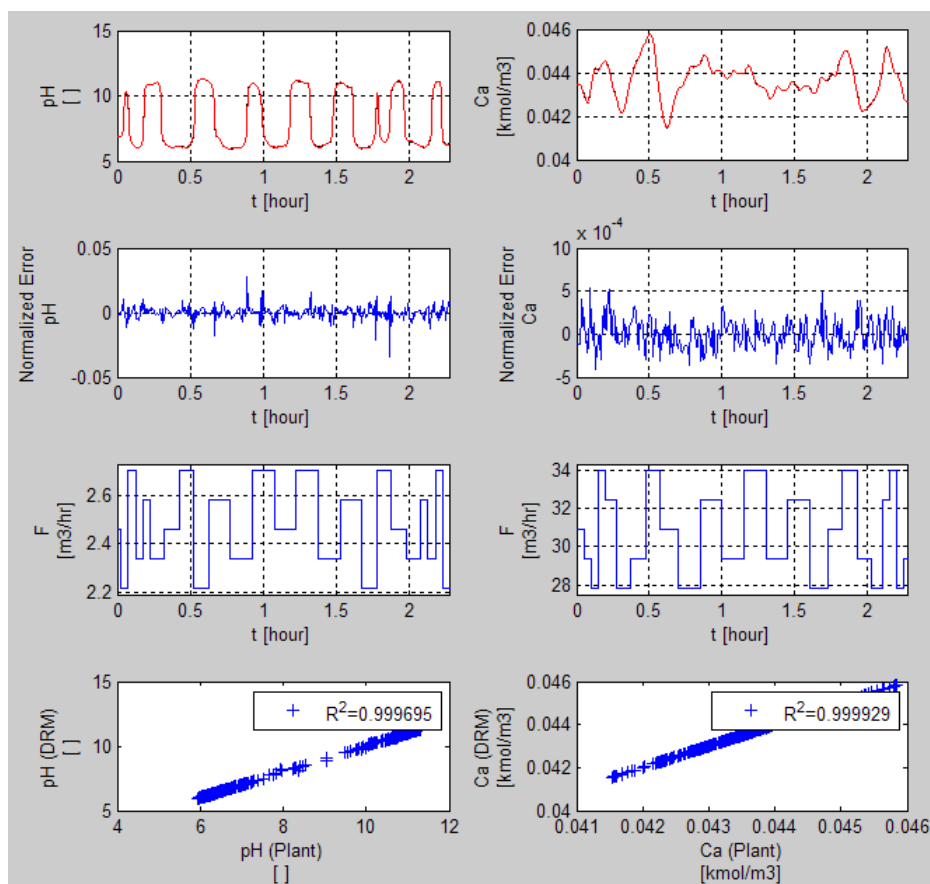
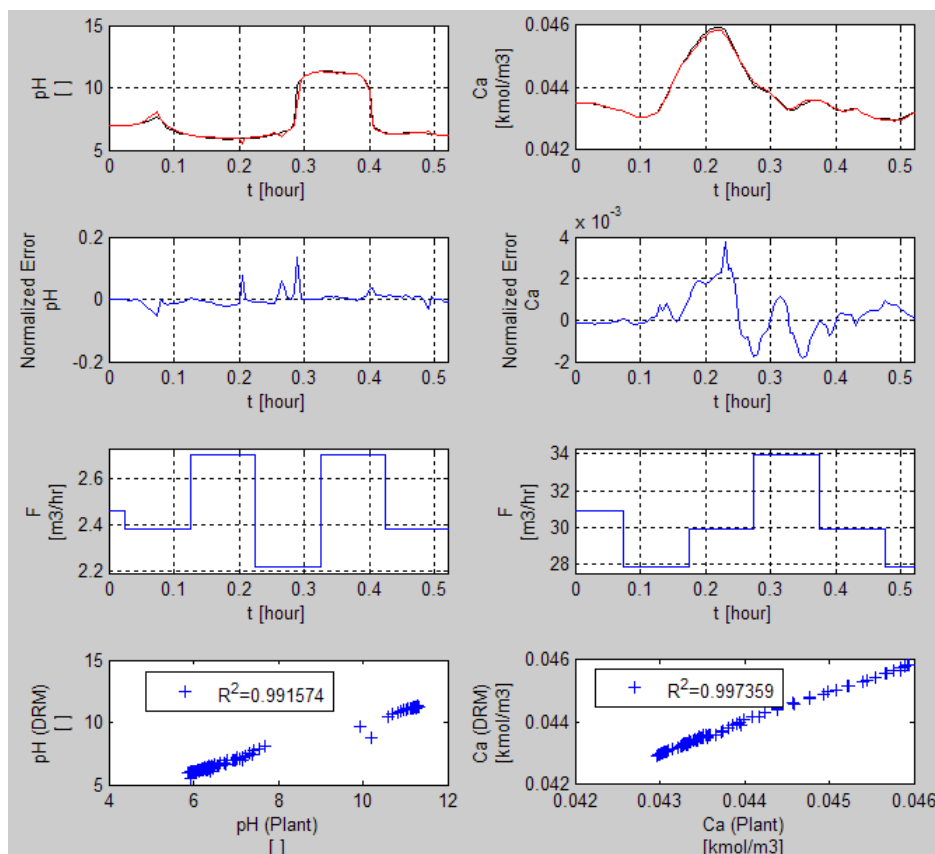


Figure 27: Predicted Response of Training Sequence using Optimized Pole Values

Notice that the  $R^2$  values are very close to 1, indicating a good fit of the NABNet model to the ACM model. Figure 28 shows the comparison of the validation data.



**Figure 28: Predicted Response of Validation Sequence using Optimized Pole Values**

Again the  $R^2$  values for the validation data are much larger than those shown in Figure 26. Therefore the accuracy of the D-RM has been improved substantially after the pole value optimization.

25. Perform the UQ analysis, by specifying the process and measurement noise levels by issuing the **UQ → Set Noise...** command. The “Process and measurement Uncertainty Dialog” window as shown in Figure 29 displays.

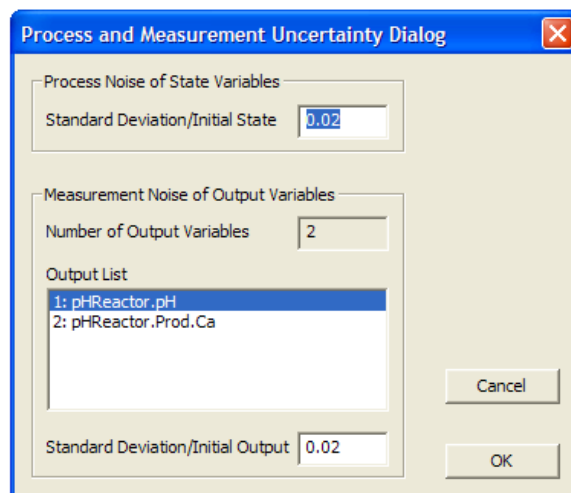


Figure 29: Process and Measurement Uncertainty Dialog Window

Set the ratio of the standard deviation to the initial state value to 0.02. Set the ratio for both of the output measurements to 0.02. Click “OK” to accept the specified inputs. Then launch the UQ calculation by issuing the **UQ → Calculate...** command. The “Result Plotting Dialog” window as shown in Figure 30 displays.

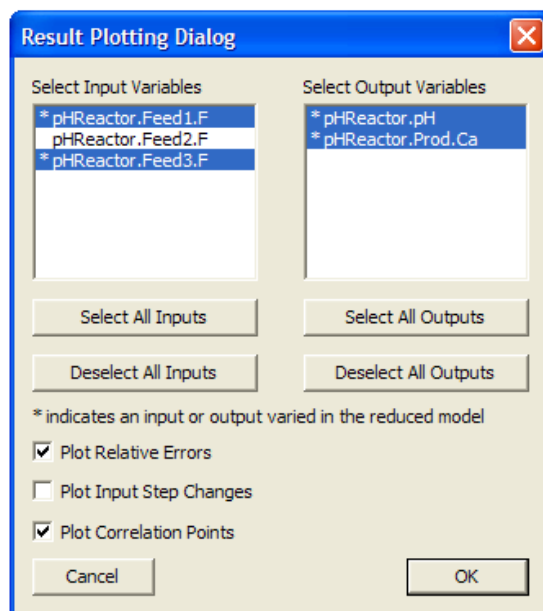


Figure 30: Result Plotting Dialog Window



Select the two input variables and two output variables as shown in Figure 30. This time clear the “Plot Input Step Changes” check box and then click “OK” to start the UQ calculation. After the UQ calculation is completed, the results are plotted in a plot window as shown in Figure 31.

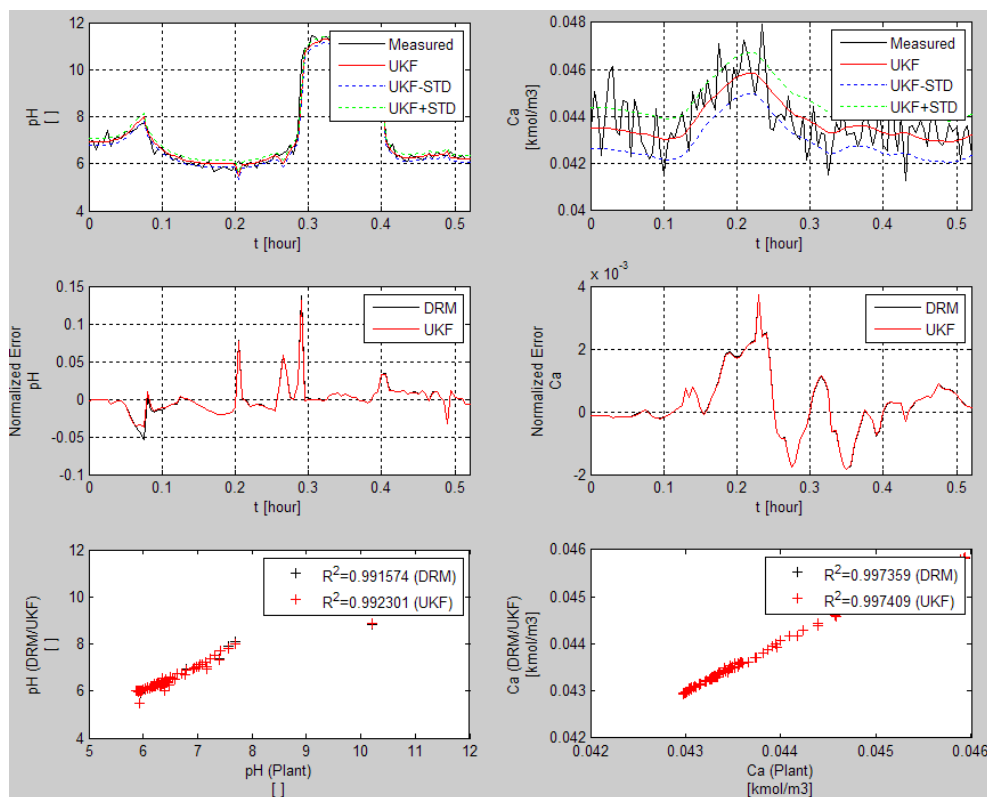


Figure 31: Plots of UQ Analysis Results on the D-RM of the pH Neutralization Reactor

26. Save the D-RM case, export the covariance matrices and the D-RM model, and then close the D-RM Builder window by issuing the corresponding commands under the “File” menu.

## 2.5. Building Data-Driven D-RM using DABNet Framework with Ramp Changes to Replace Step Changes for Hard-to-Converge Stiff Models

### Description

Depending on the dynamic system to be modeled, a high-fidelity model containing tens of thousands of DAEs could be hard to be solved especially when those equations are very stiff. For some complicated models, the change rates of certain input variables need to be kept low or the DAE solver fails to converge. Typically, the D-RM Builder prepares a sequence of step changes of the input variables and performs the ACM simulation to calculate the output variables of the system responding to the step changes of the input variables. A step change could be a big jump from one input condition to another, causing the DAE solver to fail to find a converged solution unless the internal time step for integration is extremely small. To deal with this numerical problem, the big step change is approximated by a series of small steps at a constant ramp rate to reach the new input condition. Therefore, the big step change is now replaced by a ramp change, which is implemented as multiple small step changes. Note: A discrete-time D-RM has a fixed sampling time interval. Those small step changes have to be completed to reach the desired input value within a fraction of the sampling time interval. The maximum fraction permitted by the D-RM Builder is 0.5. This option can be enabled through the GUI and the ramp rate specified. This tutorial demonstrates the usage of this option.

### Example

To demonstrate the feature, an example named “BFB” is provided in a subfolder named “C:\Program Files\CCSI\DRMBuilder\Examples\BFB” in the D-RM Builder’s installation directory. This example models the sorbent-based two-bed bubbling fluidized bed (BFB) CO<sub>2</sub> adsorber-reactor system developed by a CCSI team. This version of the BFB adsorber-reactor model contains over 20,000 DAEs. The input variables include the flow rate, temperature, and composition of a feed flue gas stream. The minimum integration time step is 0.001 second for the ACM solver while the sampling time interval for the D-RM is 0.1 second, which can be modified later. Due to the complexity of the model, a step change of as low as 1 K in inlet flue gas temperature could cause the ACM model to fail to converge. Likewise, a large step change in CO<sub>2</sub> mole fraction in the inlet flue gas could also cause the DAE solver to fail. Therefore, use the ramp change option to approximate the step change. The DABNet model is used as the D-RM type. The pole value optimization option is used to generate the D-RM. Note: A few hours are required to run the ACM simulations in this tutorial since the ACM model is complicated. The following are the steps to build data-driven D-RM using DABNet framework with ramp changes to replace step changes for hard-to-converge stiff models:

1. Copy the entire example subfolder of the pH-Neutralization reactor “C:\Program Files\CCSI\DRMBuilder\Examples\BFB” to a desired location. Confirm the user of the D-RM Builder has write-permissions to this folder. Browse to the “BFB” subfolder.
2. Open the “BFB.acmf” file by double-clicking the file. The BFB adsorber-reactor model is opened by ACM.

3. Make a steady-state run of the model. For the D-RM Builder to work, the initial states should be at steady-state. From the ACM menu, select **Run → Mode → Steady-State** (or in the Run Control toolbox select “Steady State” from the RunMode drop-down list). Make a steady-state run by clicking “Run” on the toolbar or pressing F5. Once the “Run Complete” dialog window displays, click “OK”. Navigate to **Tools → Snapshot**. In the “Snapshot Management” window, select the most recent steady-state run (at the top of the list) and then click “Export”. Save the steady-state snapshot file as an ASCII file (BFB.asnp) in the current working folder. Change the RunMode back to “Dynamic” by selecting **Run → Mode → Dynamic** and then save the ACM file. Close the ACM window.
4. Open MATLAB by double-clicking the “BFB.mdl” file. A Simulink window similar to Figure 5 displays. Note: If the D-RM Builder installation folder is “C:\Program Files\CCSI\DRMBuilders”, an ACM window also displays. In the Simulink window, double-click the “AMSimulation” block. A window displays for the user to select an ACM file. Browse to the current working directory and then select the “BFB.acmf” file. An ACM window and the “Configure AMSimulation Block” window displays. Click the “Inputs” and “Outputs” tabs of the latter and then confirm the correct variables are selected as shown in Figure 32.

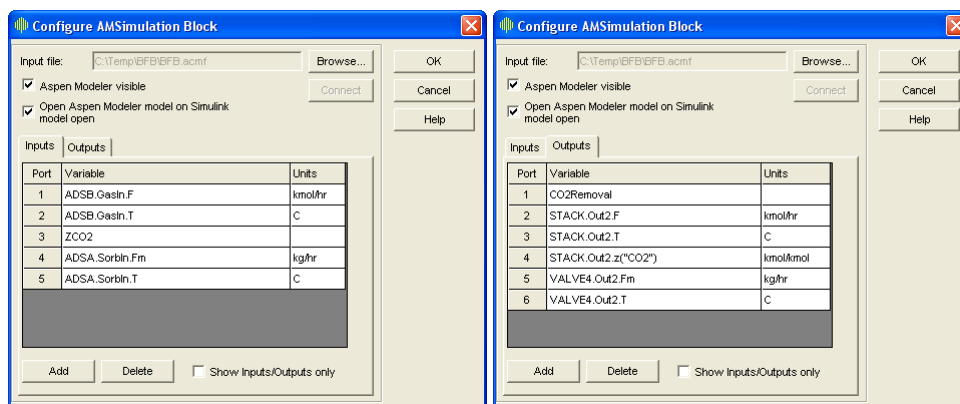


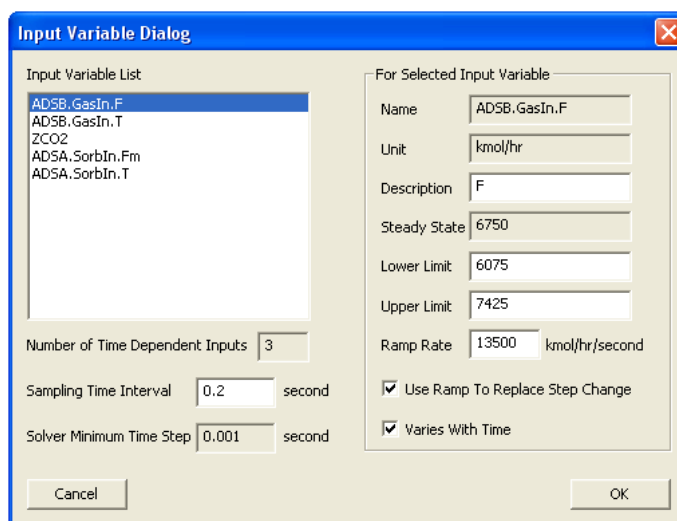


Figure 32: Simulink Dialog Window for Setting Up Input and Output Variables

Click “OK” to save the settings for the “AMSimulation” block and exit the Configuration window. Save and close the Simulink model. This automatically closes the ACM window. Close the MATLAB window as well. The “AMSimulation.m” file can be deleted.







5. After the MATLAB/Simulink file is prepared, start the D-RM Builder by double-clicking the shortcut icon of the D-RM Builder on the desktop or by the “Start” menu. A blank D-RM Builder window displays.
6. From the main menu, select **Setup → Choose High-Fidelity Model → Aspen Custom Modeler File...** or click  on the toolbar to issue the command. The “Open” window displays for file browsing and selection. Browse to the directory in which the three files are located and then select the “BFB.acmf” file. This loads the ACM-based high-fidelity model into the D-RM Builder.





7. Navigate to **Setup → Choose High-Fidelity Model → ACM Snap Shot File** and then select the steady-state snapshot file “BFB.asnp”.
8. From the main menu select **Setup → Choose High-Fidelity Model → MATLAB/Simulink File...** and then select the Simulink model file “BFB.mdl”. Note: The file directory where the MATLAB/Simulink file is located is used by the D-RM Builder as the working directory, where temporary files are created during the D-RM building process.
9. Configure the input variables. Navigate to **Setup → Configure Input Variables...** or click  on the toolbar. The “Input Variable Dialog” window as shown in Figure 33 displays.



**Figure 33: Input Variable Dialog Window**

For this example, turn on the “Use Ramp To Replace Step Change” option for the first three input variables in the “Input Variable List” list box. Change the “Sampling Time Interval” from the 0.1 second to 0.2 second. Note: The initial value of 0.1 second is the value set in the “BFB.mdl” file. Double the time interval such that smaller values (slopes) can be used for the ramp rate. To use the ramp option, click the first item “ADSB.GASIn.F” in the list box and then select the “Use Ramp To Replace Step Change” check box. The “Ramp Rate” text box is enabled. Enter 13500 in the “Ramp Rate” text box. These are the user inputs for the flow rate of the feed flue gas stream. Click the second item “ADSB.GasIn.T” in the list box and then select the “Use Ramp To Replace Step Change” check box. Enter 80 in the “Ramp Rate” text box. These are the user inputs for the temperature of the feed flue gas stream. Click the third item “ZCO2” in the list box and then select the “Use Ramp To Replace Step Change” check box. Enter 0.264 in the “Ramp Rate” text box. These are the user inputs for the mole fraction of CO<sub>2</sub> in the feed flue gas stream. Note: The ramp rate specified here for each input variable corresponds to the rate of change from the lower limit to the upper limit in half of the sampling time interval. For the fourth and fifth items, clear the “Varies With Time” check box to fix the composition and temperature of the feed sorbent stream. Click “OK” to accept the modifications and exit the window.

10. Configure the output variables by issuing the **Setup → Configure Output Variables...** command or clicking  on the toolbar. For this example, build the D-RM for the first four output variables only. Select the fifth item in the “Output Variable List” list box and then clear the “Included in DRM” check box. Select the sixth item and clear the check box. Accept the modifications by clicking “OK”.
11. Prepare the training sequence, by selecting the **Setup → Prepare Training Sequence...** command or clicking  on the toolbar. The “Training Sequence Dialog” window displays. For this example, enter 5 in the “Number of LHS Points” text box and then enter 4 in the “Number of LHS Sets” text box. Four rows show in the “For Each LHS Set” list box. Select the first row “LHS Set 1” item in the list box and then enter 8 in the “Duration of Step Change” text box. Select the second row “LHS Set 2” item in the list box and then enter 10 in the “Duration of Step Change” text box. Select the third row “LHS Set 3” item in the list box and then enter 12 in the “Duration of Step Change” text box. Select the fourth row “LHS Set 4” item in the list box and then enter 14 in the “Duration of Step Change” text box. Confirm the “Include Reverse Step Changes” check box is selected. Click “OK” to accept the user inputs.
12. Prepare an input change sequence for validation by issuing the **Setup → Prepare Validation Sequence...** command or clicking  on the toolbar. The “Validation Sequence Dialog” window displays. Enter 5 in the “Number of LHS Points” text box. Leave all of the other entries in the window unchanged. Click “OK” to accept the change and then close the window.
13. Save the file by issuing the **File → Save** or **File → Save As...** command or by clicking  on the toolbar. The “Save As” dialog window displays. Enter “BFB” in the “File Name” text box and then click “OK”. A binary D-RM Builder case file named “BFB.drm” is written to the current working directory. This name is also show on the title bar of the main window, replacing the previous default name of “DRMBuilder1.drm”.
14. Launch the ACM high-fidelity model simulation for training by issuing the **Build → Perform Training Simulation** command or by clicking  on the toolbar. This creates temporary files and a folder in the current working directory and displays a MATLAB command window followed by an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACM’s “Simulation Message” window is turned on. Wait for a few hours without closing the MATLAB command window or ACM window since it takes a while to finish the ACM simulation for a complicated model like the BFB adsorber-reactor model. After the high-fidelity model simulation is completed, a MATLAB plot window displays with the 2-D plots of the input steps and output responses. A message box window confirming the successfully completion of the high-fidelity model simulation displays. After reviewing the plots, close the plot window and the MATLAB command window by clicking “OK” in the message box window. The temporary files and folder are deleted.
15. Launch the ACM high-fidelity model simulation for validation by issuing the **Build → Perform Validation Simulation** command or by clicking  on the toolbar. The MATLAB and ACM windows display. A message box window displays after the simulation is completed. Click “OK” in the message box window to close the MATLAB windows and the ACM window.

16. Select DABNet as the D-RM model type by issuing the **Build → D-RM Model Type → DABNet** command and then confirming the “DABNet” pop-up submenu is selected.
17. Build the DABNet model by issuing the **Build → Generate Reduced Model...** command or by clicking  on the toolbar. The “DABNet DRM Parameter Dialog” window displays. Turn on the pole value optimization option for each output variable. Select an output variable in the “Output Variable List” box and then select the “Optimize Pole Value” option in the “For Each Output Variable” section. Repeat this for all four of the output variables. Click “OK” in the window to accept all of the modifications. The D-RM Builder starts to train the DABNet models with the pole values optimized. It could take quite a while to complete the training and optimization process since the D-RM contains three input variables and four output variables. The mouse icon changes to the “busy” icon until the model generation process is completed.
18. Save the case by issuing the **File → Save** command or by clicking  on the toolbar. Since the dynamic reduced model has been generated, save the case setup and model data to the case file.
19. To find how good the D-RM model prediction is compared to the high-fidelity model prediction, use the commands under the “Post-Process” menu. Confirm the “Use Balanced Model for Prediction” option under the “Post-Process” menu is selected.
20. Examine the response for the training data. To predict the training response using the generated D-RM, issue the **Post-Process → Predict Training Response** command or click  on the toolbar.
21. Issue the **Post-Process → Plot/Compare Training Responses...** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 34 displays.

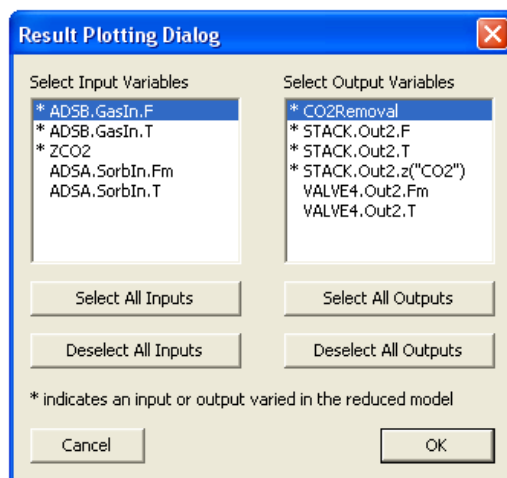


Figure 34: Result Plotting Dialog Window

Select the first input variable “ADSB.GasIn.F” in the “Select Input Variables” list, select the first output variable “CO2Removal” in the “Select Output Variables” list, and then click “OK”. A MATLAB command window displays, immediately followed by the plot window as shown in Figure 35, and then by a message box window confirming the successful completion of the plotting command.

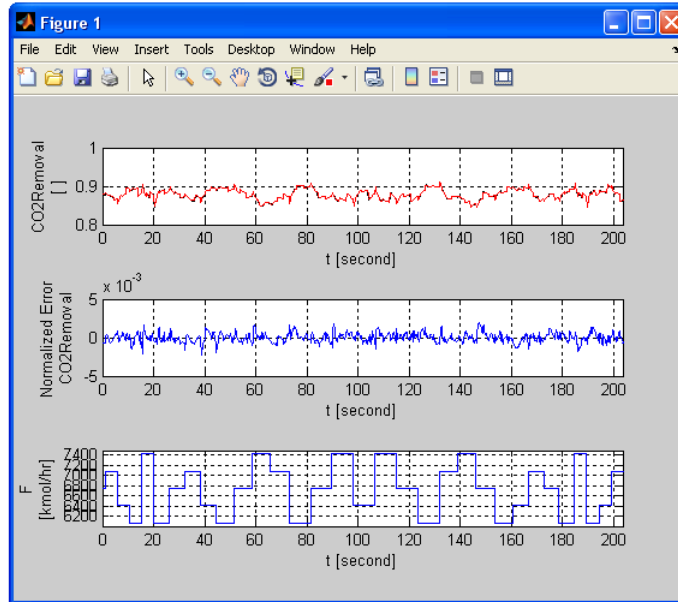




Figure 35: Predicted Response of CO<sub>2</sub> Removal Fraction for the Training Sequence

Maximize the plot window to see the plots of the output variables clearly. If the user looks carefully, there are two curves plotted for each output variable. The one in black is the high-fidelity model (ACM model) simulation result and the other in red is the D-RM prediction. The normalized error plots also show relative errors less than  $5 \times 10^{-3}$ . Close the plot and MATLAB command windows after examining the results for the training sequence by clicking “OK” in the message box window. If needed, issue the **Post-Process → Plot/Compare Training Responses...** command again to view the plot for other output variables.

22. Examine the response for the validation data. To predict the validation response using the generated D-RM, issue the **Post-Process → Predict Validation Response** command or click  on the toolbar.
23. Issue the **Post-Process → Plot/Compare Validation Responses...** command or click  on the toolbar. The “Result Plotting Dialog” window as shown in Figure 34 displays. Select one or a few input variables and one or a few output variables and then click “OK”. A MATLAB command window displays immediately followed by the plot window and then a confirmation message box window. Click “OK” in the message box window to close the MATLAB windows.
24. Save the D-RM case, export the D-RM model, and close the D-RM Builder window by issuing the corresponding commands under the “File” menu.




## 2.6. Building Data-Driven D-RM using NARMA Framework

### Description

The Nonlinear Autoregressive Moving Average (NARMA) model is another type of D-RM implemented in the D-RM Builder. This is also a discrete-time plant identification model (refer to Reference 2). This model is simpler than the DABNet model in terms of model formulation. In this model, the predicted value of an output variable at the next future time step is a function of the current and past values of the input and output variables. The functional relationship is modeled by an artificial feed-forward neural network with a single hidden layer. No internal state-space variables are included in this model. Therefore, this model is simply an input/output mapping. The user inputs for the model generation include only the number of neurons in the hidden layer of the neural network and the number of discrete historical time steps.

### Example

The Van-de-Vusse reactor model created in the first tutorial of this manual is used as the high-fidelity model for this tutorial. If the user has saved the first tutorial case to a case file as “VdV.drm”, double-click the file name or icon. This opens the main window of the D-RM Builder. Since the high-fidelity model simulations have been performed in the first tutorial, those results can be used to build the NARMA model. The following are the steps to generate the NARMA-based D-RM.

1. Save the case file with a different name by selecting the **File → Save As...** command and then giving the case file a new name “VdV\_NARMA”. Save the file to the same working directory or to a different working directory. Note: If the user saved the file to a different directory, the user does not need to copy the ACM and MATLAB/Simulink files to the directory since the D-RM Builder case file contains the contents of the ACM and MATLAB/Simulink files if those files have been loaded before the case file is saved.
2. Select the **Build → D-RM Model Type → NARMA** command. This switches the D-RM model type from DABNet to NARMA.
3. Select the **Build → Generate Reduced Model...** command or click  on the toolbar. The “NARMA DRM Parameter Dialog” window as shown in Figure 36 displays.

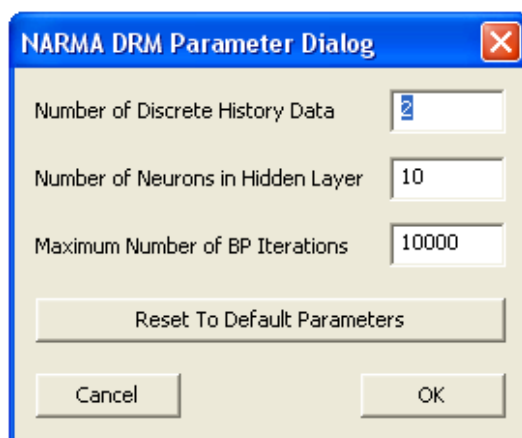




Figure 36: NARMA DRM Parameter Dialog Window



Notice that the window is different from the window for the DABNet model. The value in the “Number of Discrete History Data” text box indicates the number of historical data points (for both the input and output types of variables) to be used in the NARMA model. If this number is 1, only the data at the current time is used to predict the outputs of the next time step. Typically, this number should be either 2 or 3. Enter the number of neurons in the hidden layer in the second text box. This value is generally related to the total number of the input and output variables. A large number should be used if the numbers of input and output variables are large. Too large a number of hidden neurons could possibly over-fit the training data which could cause the accuracy for the validation data to decrease. The user can manually modify this D-RM parameter to obtain the best fit for the validation data. In the current version of the D-RM Builder, no procedure has been implemented to optimize the number of neurons in the hidden layer. Enter the maximum number of back propagation iterations in the third text box. Note: For the NARMA model, back propagation algorithm for the neural network training should work quite well and the IPOPT algorithm is not provided as an option. For this example, accept the default values and then click “OK”. The training process starts immediately. The mouse icon is switched to a “busy” icon until the training process is completed.

4. Before predicting the responses for the training and validation sequences using the generated D-RM, select where to get the historical output data. Since the NARMA model requires the historical output data as inputs to the neural network, the user can either use the “true” data, which are the high-fidelity model simulation results in this case, or the output data predicted by the NARMA model itself from previous steps. The D-RM Build enables the user to use one of the two options. Try both options in this tutorial. First navigate to **Post-Process → Use High-Fidelity Model History For Prediction** and then confirm this option is selected. This option makes the D-RM Builder use the high-fidelity model simulation results in the current and previous time steps to predict the response of a future time step.
5. Use the commands under the “Post-Process” menu to predict the training and validation responses using the NARMA D-RM that has just been generated and then visualize the results. For example, click  followed by  on the toolbar to plot the predictions for the validation sequences. Figure 37 shows the plots of the validation data.

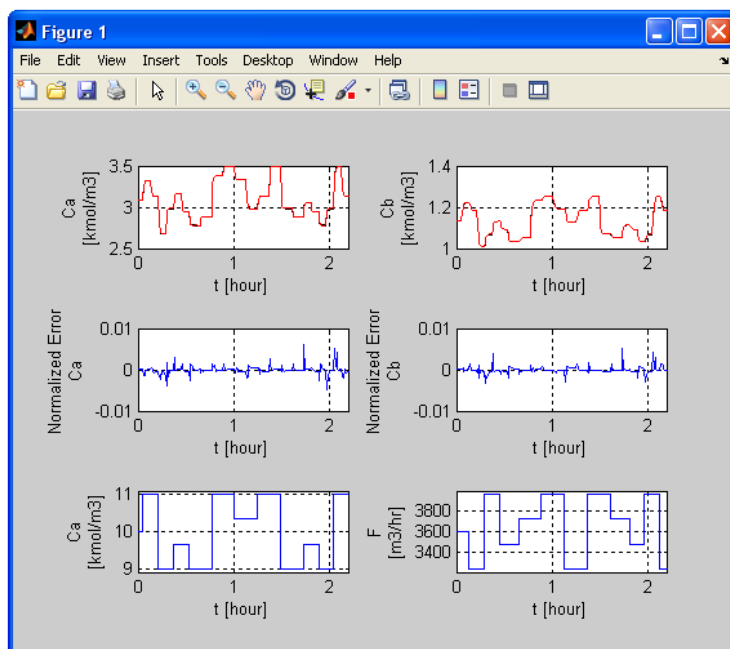




Figure 37: Predictions for Validation Sequence using High-Fidelity Model History data

Notice that when the “ture” output values of the current and previous time steps are used, the predicted responses by the NARMA model are in good agreement with the high-fidelity model predictions. This means the NARMA model is capable of predicting short-term future responses given the accurate values of the output variables at the moment and in the near history.

6. To try the other prediction option, navigate to **Post-Process → Use High-Fidelity Model History For Prediction** and then confirm it is cleared. This makes the D-RM Builder use the current and previous output values predicted by D-RM itself to predict the output values for the next time step.
7. Repeat the commands in Step 5 by clicking  followed by  on the toolbar. Figure 38 shows the plots of the validation sequence.

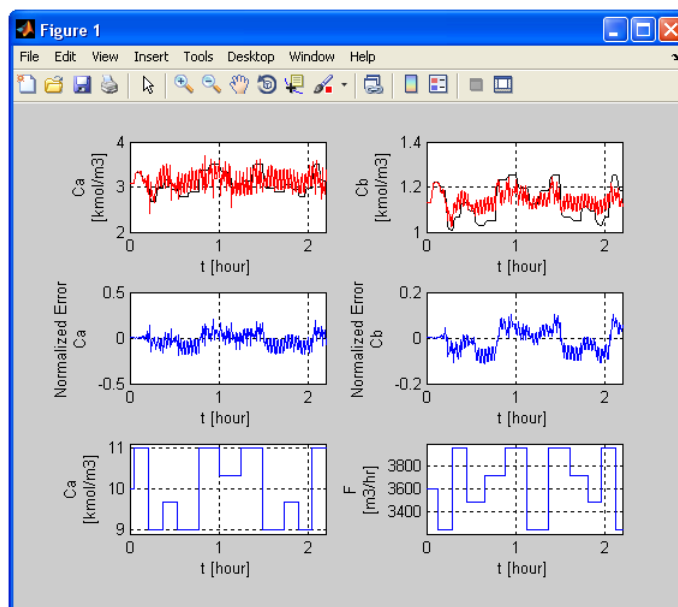


Figure 38: Predictions for Validation Sequence using D-RM Predicted History Data

It can be seen that the D-RM predictions are very different from the high-fidelity model predictions except in the first few time steps. Oscillations are found from the D-RM predictions. This means the NARMA model is not good for long-term prediction without using the measured plant output data. The DABNet models, on the other hand, are capable of predicting quite accurately long-term responses without the need for the measured output data. Therefore, DABNet is more suitable for model predictive control in which the maneuver through control variables is optimized on a quite long time horizon.

8. Export the NARMA-based D-RM to a file with an “.m” extension. Note: When **Build → D-RM Model Type → NARMA** is selected, the exported D-RM is in the NARMA format. Since the DABNet D-RM was built in the first tutorial, the DABNet D-RM is still in the memory of the D-RM Builder and can be exported after **Build → D-RM Model Type → DABNet** is selected. Save the case file and then close the D-RM Builder application.

## 2.7. Testing Generated D-RM for Dynamic Simulation in MATLAB


### Description

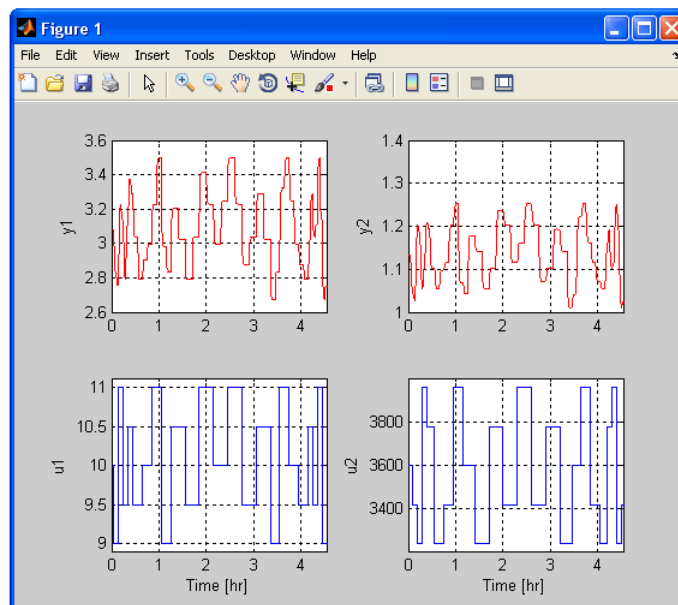
After a D-RM is generated by the D-RM Builder application, the D-RM can be exported to a file in MATLAB script format. In the exported MATLAB file, the internal parameters of the D-RM as MATLAB variables and arrays including cell arrays are assigned with values. To perform dynamic simulations using the exported D-RM, three MATLAB classes have been developed to work with the D-RM Builder. These classes are packed inside the “Setup.exe” file, the installer of the D-RM Builder. The files for the three MATLAB classes, “DRM\_ANN.m”, “DRM\_DABNet.m”, and “DRM\_NARMA.m” are installed in the “drm\_models” subfolder under the “matlab\_files” folder in the D-RM Builder installation directory. Also installed in the “matlab\_files” folder is a MATLAB driver code named “Test\_DRM.m”, which contains MATLAB scripts to read the exported D-RM file to construct the corresponding MATLAB objects and then perform a dynamic simulation using the input step change sequence created by the D-RM Builder. This tutorial demonstrates the usage of the three MATLAB classes and the driver script.

### Example

The Van-de-Vusse reactor model created in the first tutorial of this manual is used as the model for this tutorial. The DABNet-based D-RM is demonstrated. The following are the steps to test the generated D-RM for the dynamic simulation in MATLAB:

1. If the user has not exported the DABNet-based D-RM, the user can do so by opening the case file “VdV.drm” and then issuing the **File → Export → D-RM As MATLAB Script File** command. The “Save As” dialog window displays. Accept the default file name as “VdV\_DRMParameters.m” and then click “Save”.
2. If the user has not exported the training data to a “.csv” file, the user can do so by issuing the **File → Export → Training Data...** command. In the “Save As” window, enter the file name as “VdV\_training.csv” and then click “Save”.
3. Find the MATLAB files in the D-RM Builder installation directory. Browse to the “C:\Program Files\CCSI\DRMBuilder\matlab\_files\drm\_models” folder and then copy the three MATLAB files, “DRM\_ANN.m”, “DRM\_DABNet.m”, and “DRM\_NARMA.m”. Paste these three files to a desired working directory. Browse to the “C:\Program Files\CCSI\DRMBuilder\matlab\_files” folder and then copy and paste the “Test\_DRM.m” file to the same working directory. View the “Test\_DRM.m” file by opening the file in MATLAB or any text editor. If the user is familiar with the MATLAB script language, the user can see that the script enables the user to open an “.m” file that is exported from the D-RM Builder and contains the D-RM parameters. Run the “.m” file to assign the variables to the MATLAB workspace, create either a “DRM\_DABNet” object or a “DRM\_NARMA” object, and then pass the variables in the workspace as parameters of the class constructor. Select a training or a validation sequence in a “.csv” file and then initialize the D-RM object (setting the initial conditions) based on the first point in the training or validation sequence. The member function of “evalNextStep()” is called to perform the dynamic simulation. The predicted results by the D-RM can be saved to a “.csv” file.

4. To run the “Test\_DRM.m” file on a computer with MATLAB installed, double-click the file name or icon, a MATLAB command window displays along with the MATLAB editor window. In the editor window, click  on the toolbar. A “Choose D-RM Model Parameter File” window displays.
5. Browse to the directory where the “VdV\_DRMParameters.m” file is located and select the file. The “Choose Training or Validation Data File” dialog window displays.
6. Browse to the directory where the “VdV\_training.csv” file is located and then select the file. The dynamic simulation by the D-RM object is completed immediately. The “Save D-RM Predicted Results As” dialog window displays. Browse to the same directory, enter a file name as “VdV\_training\_D-RM”, and then click “Save”. A file named “VdV\_training\_D-RM.csv” is created in the directory. Meanwhile, a MATLAB plot window as shown in Figure 39 displays.



**Figure 39: Predicted Result by the D-RM Object in MATLAB**

Compare the plots with those in Figure 16; the curves of the output variable are essentially the same, indicating the MATLAB object for the D-RM predicts the same responses as the C++ object does in the D-RM Builder.

7. Close the MATLAB command window to finish the tutorial. Note: The MATLAB script file “Test\_DRM.m” can also be used to test the NARMA-based D-RMs. The NARMA model requires the user to provide the initial condition of both of the input and output variables while the DABNet model requires only the steady-state input variables as the initial condition. For testing the NARMA model using the “Test\_DRM.m” file, the historical output data used for the neural network inputs are based on the predictions of the D-RM itself.

## 3. USAGE INFORMATION

### 3.1. Environment/Prerequisites

The D-RM Builder requires the commercial licenses of ACM and MATLAB with Simulink toolbox. ACM is required since the differential and algebraic equations in the high-fidelity model are specified in the ACM script language and solved by the ACM engine. MATLAB/Simulink is used to embed the ACM model to a custom block of the Simulink flowsheet. Simulink can handle multiple simulation calls to ACM in a flexible and seamless manner. MATLAB also provides functions to plot input/output and relative error with respect to time. The current version of the D-RM Builder has been developed and tested on ACM Versions 7.3 and 8.0, MATLAB Releases 2011b and 2014a, and Simulink Versions 7.8 and 8.3. The 32-bit version of MATLAB including Simulink toolbox has to be used since the ACM library and executable are 32-bit only. It is recommended to use these versions for running the examples and testing purpose. In addition, if the user has multiple versions of ACM and MATLAB installed, confirm that double-clicking the ACM files (.acmf) and Simulink model files (.mdl) invoke the above-mentioned versions of the software.

### 3.2. Support

To obtain support for this package, send an email to [ccsi-support@acceleratecarboncapture.org](mailto:ccsi-support@acceleratecarboncapture.org), and/or fill out the “Submit Feedback/Request Support” form available on the product distribution page.

### 3.3. Restrictions

The D-RM Builder requires a robust and stable ACM model in terms of model convergence. Solver failure sometimes could happen to a complicated ACM model if a large step change of an input variable is imposed. If it can be avoided by replacing the step change by a ramp change, the D-RM Builder provides an option to use a ramp to mimic a step change.

## 4. ADVANCED FEATURES

A user of the D-RM Builder needs to be familiar with the high-fidelity ACM model including the typical response of the output variables with respect to the change in the input variables. To build a D-RM model with reasonable accuracy, the user needs to be aware of the typical time constant of an output variable with respect to an input variable. The generated D-RM is a discrete-time model and the sampling time interval is an important parameter. This parameter can be modified in the dialog window of the “Configure Input Variables” command under the “Setup” menu. The user could try to modify the sampling time interval to obtain a better D-RM.

Another feature to reduce the D-RM error is to optimize the pole values if the model type is DABNet. The D-RM Builder has a build-in derivative-free optimizer to optimize the pole values with respect to each input/output pair. The optimizer does not guarantee to reach global optimum. However, the user can provide the initial guess for the pole values.

The default method of back propagation is a good training method to use with the training of the artificial neural network. The IPOPT method provides an alternative, which sometimes could lead to a better model. However, the CPU time required by IPOPT might be five times longer than that of the back propagation method. IPOPT is not recommended if the pole optimization option is selected.

## 5. DEBUGGING

The D-RM Builder displays messages to the main window of the application. If some critical error occurs, the program displays pop-up windows with warning messages and sometimes recommendations. A good indication of a successful generation of a D-RM is the small errors in the D-RM prediction of a validation sequence compared to the high-fidelity model simulation result.

### 5.1. How to Debug

Mistakes made by a user during the setup stage can usually be caught by the D-RM Builder with warning messages. These mistakes include incorrect files selected, incorrect specification of input variable ranges, and model parameters. These mistakes could be corrected by following the instructions included in the warning messages.

If the generated D-RM gives a poor prediction of response of the system compared to the high-fidelity model simulation, the user can check the message log displayed in the main window, especially the relative error reported during the model generation process. If the neural network training error is larger than 0.01, the D-RM does not have a good prediction for the training and validation data. The user should try to find the typical time constants of the system by manually simulating the ACM model with step changes. Then the user should try to adjust the time interval and to optimize the pole values in case of the DABNet model.

If the simulation of the high-fidelity model fails, error messages are displayed in MATLAB or ACM windows. If this happens, the user needs to correct the high-fidelity model.

## 5.2. Known Issues

The software module for the link between MATLAB and ACM was developed by AspenTech and is included as a part of the Aspen installation. However, a DLL named “stdole.dll” might not be correctly registered by Aspen’s installer, which could cause the configuration between Simulink and ACM described in Step 4 of the tutorial in Section 2.2 to fail. Sometimes, installing a version of Microsoft Office suite could fix the problem. If the system already has Microsoft Office or Microsoft Visual Studio® installed, the DLL must have been registered correctly.

The D-RM Builder relies on the MATLAB engine to communicate between MATLAB and itself. The D-RM Builder’s installer registers the server of the MATLAB engine at the end of the installation if it has not yet been registered. However, a user may have accidentally unregistered the server after the installation of the D-RM Builder by a command at a DOS prompt such as “matlab /unregserver”. Then the user receives a pop-up message stating “Initializing MATLAB engine failed!” when the user launches the high-fidelity model simulation or plot/compare the training or validation responses. If this happens, the user needs to manually register the server again by running the “matlab /regserver” command at a DOS prompt.

If the D-RM Builder is used to call the MATLAB engine for the first time since the computer reboots, it might take quite a long time to bring up the MATLAB command. If the D-RM Builder is idle for a long period of time (several hours), it could fail to open the MATLAB engine when a command for plotting or running an ACM simulation is issued and sometimes could cause the application window to close without warning. This problem is caused by bugs in the MATLAB engine and has been confirmed by MATLAB support. Until the MATLAB bug is fixed, the user should save the case first after a long period of idle time.

## 5.3. Reporting Issues

To report an issue, send an email to [ccsi-support@acceleratecarboncapture.org](mailto:ccsi-support@acceleratecarboncapture.org).

## 6. REFERENCES

- [1] Sentoni, G.B., Biegler, L.T., Guiver, J.B., and Zhao, H., “State-Space Nonlinear Process Modeling: Identification and Universality,” *AIChE Journal*, 1998, 44 (10).
- [2] Narendra, K.S., “Adaptive Control Using Neural Networks and Approximate Models,” *IEEE Transaction on Neural Networks*, 1997, 8 (3), p. 475–485.