D-RM Builder
User Manual

Version 2.0.0

February 2018

Copyright (c) 2012 - 2018

Copyright Notice

DRM Builder was produced under the DOE Carbon Capture Simulation Initiative (CCSI), and is copyright (c) 2012 - 2018 by the software owners: Oak Ridge Institute for Science and Education (ORISE), Los Alamos National Security, LLC., Lawrence Livermore National Security, LLC., The Regents of the University of California, through Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest Division through Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, The University of Texas at Austin, URS Energy & Construction, Inc., et al.. All rights reserved.

NOTICE. This Software was developed under funding from the U.S. Department of Energy and the U.S. Government consequently retains certain rights. As such, the U.S. Government has been granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable, worldwide license in the Software to reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit other to do so.

License Agreement

DRM Builder Copyright (c) 2012 - 2018, by the software owners: Oak Ridge Institute for Science and Education (ORISE), Los Alamos National Security, LLC., Lawrence Livermore National Security, LLC., The Regents of the University of California, through Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest Division through Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, The University of Texas at Austin, URS Energy & Construction, Inc., et al. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the Carbon Capture Simulation Initiative, U.S. Dept. of Energy, the National Energy Technology Laboratory, Oak Ridge Institute for Science and Education (ORISE), Los Alamos National Security, LLC., Lawrence Livermore National Security, LLC., the University of California, Lawrence Berkeley National Laboratory, Battelle Memorial Institute, Pacific Northwest National Laboratory, Carnegie Mellon University, West Virginia University, Boston University, the Trustees of Princeton University, the University of Texas at Austin, URS Energy & Construction, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code ("Enhancements") to anyone; however, if you choose to make your Enhancements available either publicly, or directly to Lawrence Berkeley National Laboratory, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative works thereof, in binary and source code form. This material was produced under the DOE Carbon Capture Simulation Initiative

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 – Included 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.
Version 2016.02.0	2/29/2016	Released as a part of FOQUS. Added features to handle processes with both fast and slow responses.
Version 2.0.0	3/31/2018	Open Source release. Included a feature to build D-RMs based on given custom dynamic response data from either high-fidelity dynamic models or experiments.

Table of Contents

List of Figures

1. Introduction

While a steady-state flowsheet-type process system engineering software is used to model and optimize a system in steady-state operating conditions, the dynamics of individual components and the entire system also needs to be modeled. The best design of a coal-fired power generation and CO_2 capture system should not only achieve the lowest cost per kilowatt-hour of electricity generated in the optimal steady-state operating conditions, but also ensure easy startup and shutdown, quick and smooth transition for load change, and process noise rejection. Sometimes an advanced process control (APC) method needs to be implemented, which requires the development of dynamic models of the system and its individual components. Typically the dynamics of a process can by modeled by a set of differential and algebraic equations (DAEs) based on the first principles. A DAE-based high-fidelity dynamic model is simulated by a DAE solver such as Aspen Custom Modeler (ACM). Due to the complexity of the process involved, the CPU time required to solve the dynamics of the process is too long to implement the APC method. Therefore, a reduced dynamic model (D-RM) is desired with its execution time reduced by several orders of magnitude.

The Dynamic Reduced Model (D-RM) Builder is a software tool used to generate data-driven dynamic reduced models from high-fidelity dynamic models consisting of differential and algebraic equations (DAEs). With dynamic variables configured by SinterConfigGUI, a software tool with graphic user interface developed by the CCSI team to configure the high-fidelity dynamic model simulations, the D-RM Builder allows a user to import a dynamic ACM model, sample the input space to form a sequence of step changes, launch the high-fidelity ACM simulations, generate a D-RM, and finally visualize and validate the D-RM. For instance, the sorbent-based bubbling fluidized bed (BFB) CO_2 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-RM Builder tool is used to perform a set of dynamic simulations using the rigorous model with main control and disturbance variables undergoing multiple random step changes within the typical operating ranges and then use the responses of the step changes to train D-RMs. The D-RMs generated by the D-RM Builder enable much 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.

D-RM Builder can also be used to generate D-RMs from measured dynamic data of an existing plant. The generated nonlinear data-driven D-RMs can be used for advanced process control.

1.1. Motivating Example

The solid-sorbent-based bubbling fluidized bed CO_2 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_2-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 an application on Windows platform 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 read the input and output variables pre-configured through SinterConfigGUI application, which is also a part of the CCSI software package. Through the D-RM Builder GUI, a user can select a set of the pre-configured 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 ConsoleSinter, which is also a part of the CCSI package. The simulation results can be used to generate the D-RMs. A separate set of step-change sequence can also be generated and its response be simulated by the ACM model and predicted by the D-RM for validation purpose. The generated D-RMs and the user inputs for case setup and configuration can be saved in a text file in JSON format with extension “drmb”. 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.

Two types of D-RMs are supported in the current version including the Decoupled A-B Net (DABNet) model [1], and the Nonlinear Auto-regressive Moving Average (NARMA) model [2].

Different building options are provided for the user to choose from including input delays, two-pole Laugerre formulation of DABNet model, and optimization of model parameters. 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 the state-space based DABNet 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 are displayed in the main text window of the D-RM Builder, which can be saved to a log file for future reference. The internal parameters of the generated D-RM can be exported to a MATLAB script file, which can be run in MATLAB to initialize a D-RM object instantiated from a MATLAB class. 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. Details about D-RM Builder including its software structure and its sampling and training algorithms can be found in [3].

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. D-RM Builder Tutorial

Several tutorials are given in this section to demonstrate the use of D-RM Builder to generate data-driven D-RMs. The first tutorial in Section 2.1 contains general descriptions of the features and GUIs of the D-RM Builder, while the rest of the tutorials demonstrate the individual features and options. It is strongly recommended that a new user of D-RM Builder go through the first tutorial to understand the basic features of the program.

Example files used for the tutorials are provided with the installer and are typically installed at: "C:\Program Files (x86)\CCSI\DRMBuilder\examples". The three folders named "VdV", "pH", and "BFB" contain three ACM examples, one for each folder. The folder named "User_Specified" contains two sets of user specified experimental data for a reactor. The folder named "Matlab" contains Matlab classes that represent D-RMs and a script to test generated D-RMs.

2.1. Building a D-RM Using DABNet Framework and BP-Based ANN Training

To demonstrate the feature, an example named Van-de-Vusse reactor is provided and installed in the "VdV" folder. 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 ACM file of the Van-de-Vusse reactor "VdV_Reactor.acmf" in "C:\Program Files (x86)\CCSI\DRMBuilder\examples\VdV" to a working directory. Comfirm that the user of the D-RM Builder has write permissions to this folder.

2. Launch SinterConfigGUI from Window's "Start" menu or its shortcut on the desktop. The GUI window as shown in Figure 1 displays after a splash window.

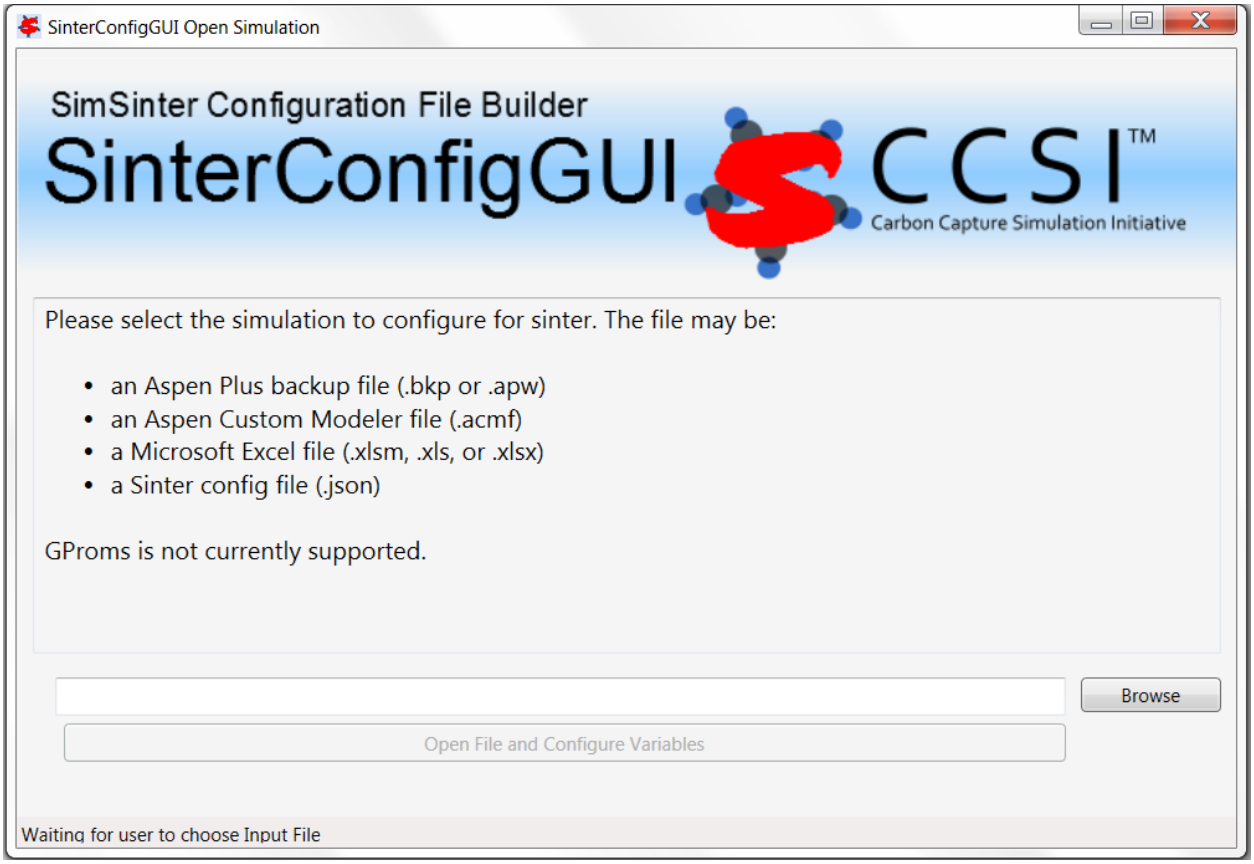

Figure 1: Open SimulationWindow for SinterConfigGUI

3. Click the Browse button and browse to the "VdV_Reactor.acmf" file in the working directory. Then click the "Open File and Configure Variables" button. It takes a few seconds for the

SinterConfigGUI to display a “SinterGonfigGUI Simulation Meta-Data” window as shown in Figure 2. An ACM window with VdV reactor model inside also displays as shown in Figure 3.

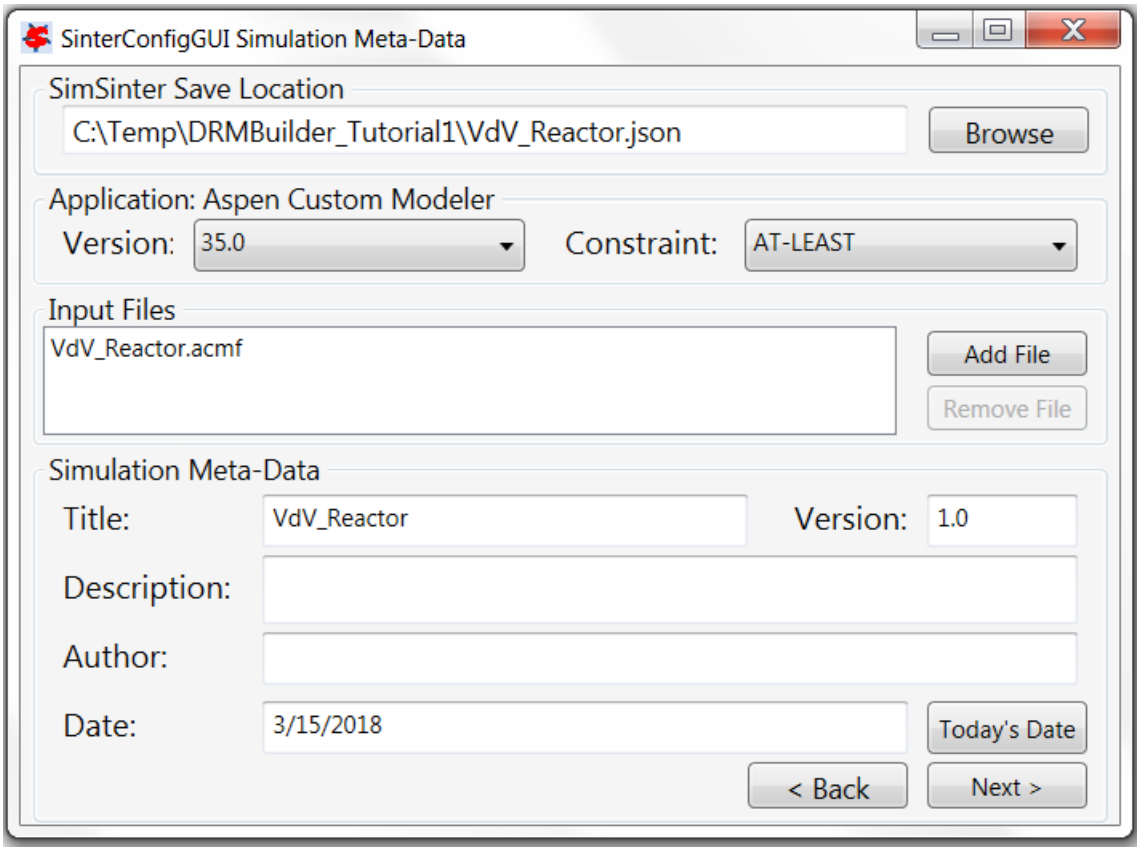

Figure 2: Dialog Window for the Simulation Meta-Data of VdV Reactor Model

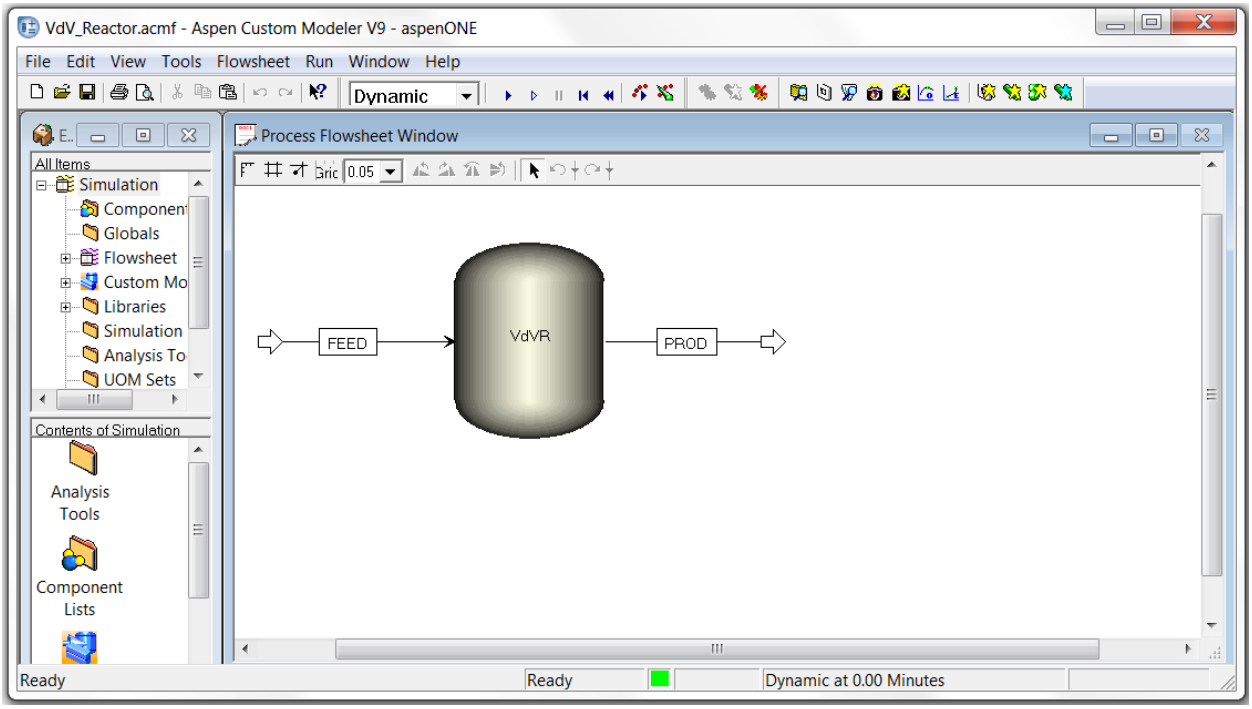

Figure 3: ACM Window Displayed for VdV Reactor Model by SinterConfigGUI

4. Enter “VdV Reactor Example” in the “Title” text box, “My first DRMBuilder tutorial” in the “Description” text box, and user’s name in the “Author” text box. Then click the “Next” button. A “SinterConfigGUI Variable Configuration Page” window displays as shown in Figure 4Figure 4.

Warning: Please keep the “SimSinter Save Location” text box unchanged since the configuration file should be saved in the same directory as the ACM file for the D-RM Builder program. If a D-RM Builder case is moved to run in a different directory or on a different Windows computer, the SimSinter configuration file (to be generated with an extension “json”) and the D-RM Builder case file (to be generated with an extension of “drmb”) have to be copied to the same directory along with the ACM file.

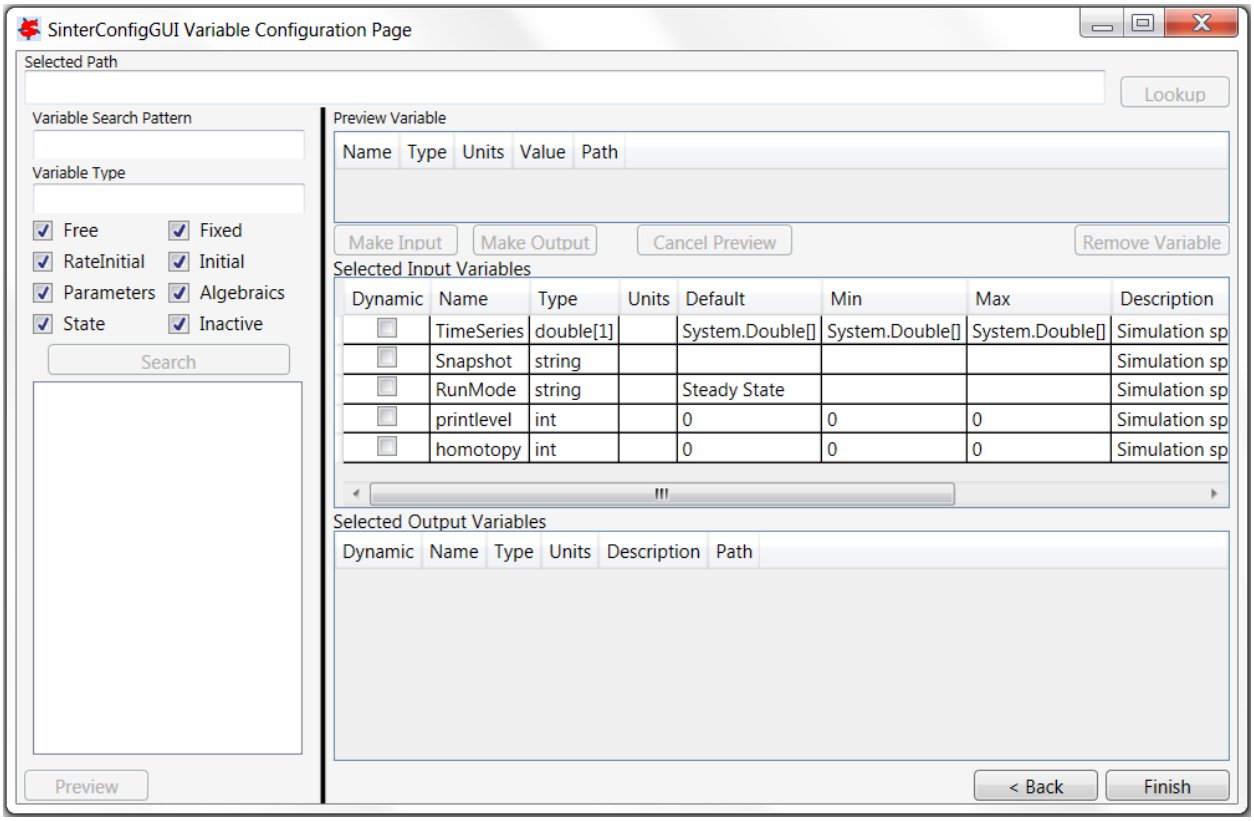

Figure 4: Variable Configuration Page of SinterConfigGUI

5. Enter "~" in the "Variable Search Pattern" text box and click the "Search" button. All of the variables in the VdV reactor model are listed in the list box at the lower left corner of the window as shown in Figure 5Figure 5: Available Dynamic Variables Listed by SinterConfigGUI.

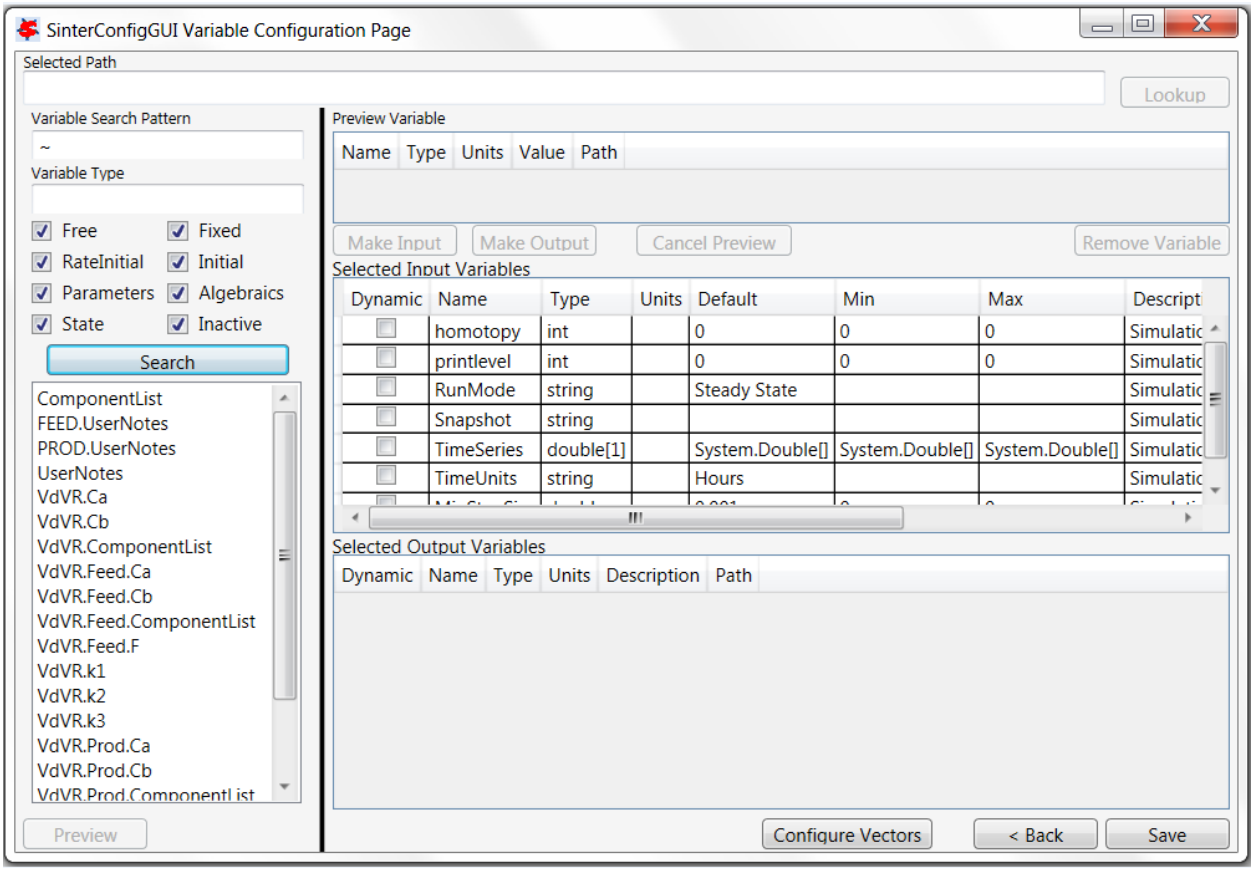

Figure 5: Available Dynamic Variables Listed by SinterConfigGUI

6. Select the “VdVR.Feed.Ca” in the list box and click the “Lookup” button at the upper right corner. The variable is displayed in the “Preview Variable” section. Click the “Make Input” button and the variable is appended to the end of the “Selected Input Variables” table. Change the name of the variable to “Ca_Feed” and select the checkbox in the “Dynamic” column. This makes the selected variable a dynamic input variable.

7. Select the “VdVR.Feed.F” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “F_Feed”.

8. Select the “VdVR.V” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “V_Reactor”.

9. Change the default value of the input variable named “RunMode” in the “Selected Input Variables” table from “Steady State” to “Dynamic”. Please do not enter quotes.

10. Select the “VdVR.Prod.Ca” in the list box and click the “Preview” button. Then click the “Make Output” button and the variable is added to the “Selected Output Variables” table. Change the name of the variable to “Ca_Product” and select the checkbox in the “Dynamic” column. This makes the selected variable a dynamic output variable.

11. Select the “VdVR.Prod.Cb” in the list box and repeat Step 10 to make it a dynamic output variable and change the name to “Cb_Product”.

12. This concludes the configuration of ACM variables. Figure 6 shows the SinterConfigGUI window after the configuration. Click the Next button at the lower right corner and a “SinterConfigGUI Vector Default Initialization” dialog window displays as shown in Figure 7. Leave the default time series data unchanged.

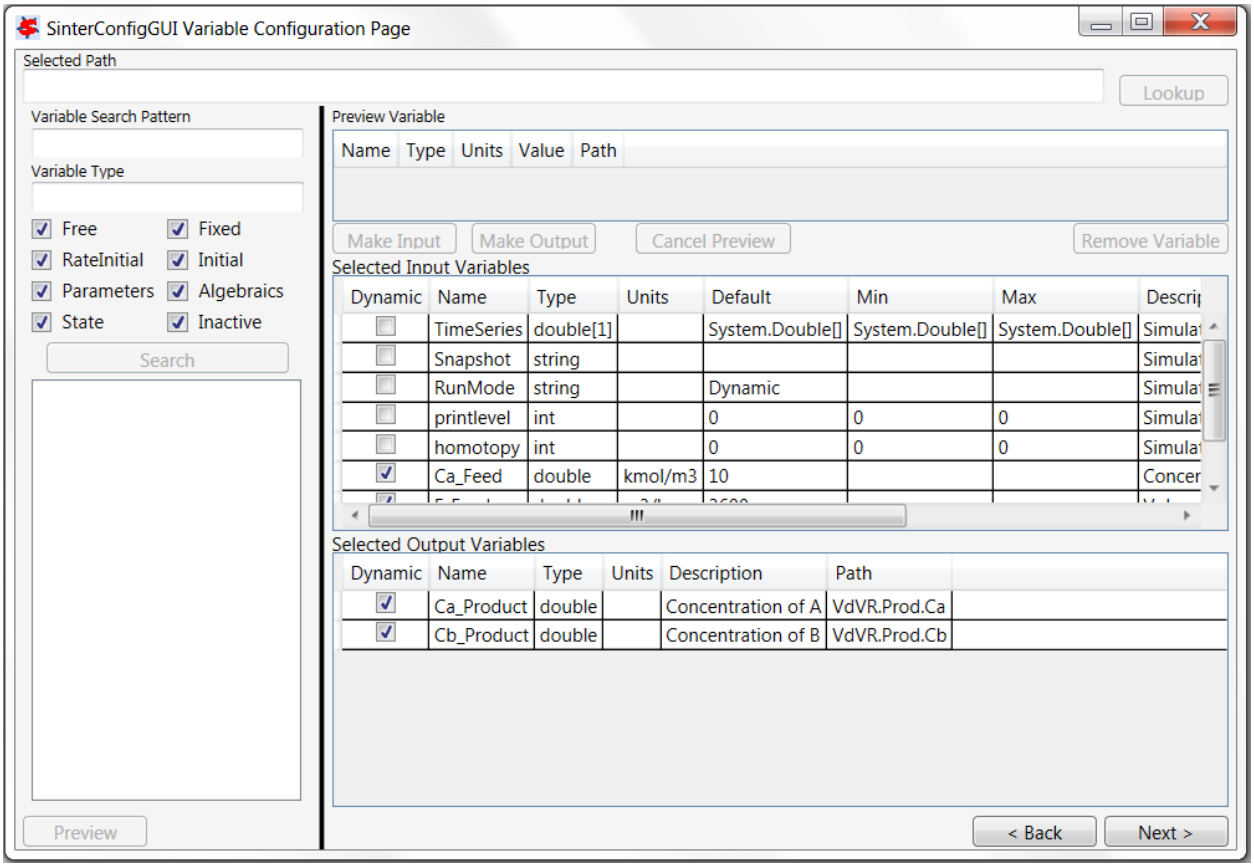

Figure 6: Configured Dynamic Variables by SinterConfigGUI

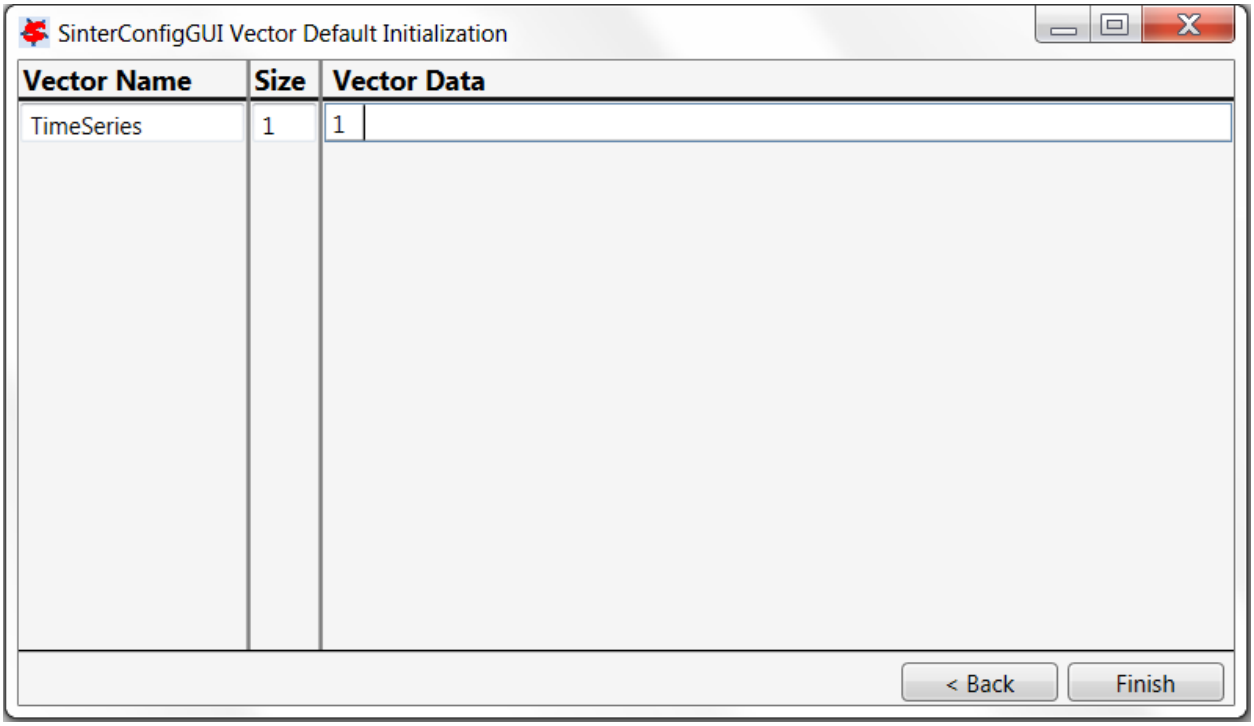

Figure 7: Time Series Vector SpecificationWindow of SinterConfigGUI

13. Click the "Finish" button to exit the "SinterConfigGUI". A file named "VdV_Reactor.json" is created in the working directory. Close the ACM window.

14. After the dynamic input and output variables are configured and the JSON file is created, start D-RM Builder. If it is the first time to run the D-RM Builder on the computer, some configuration will be setup by the program. D-RM Builder needs to find the full path of an executable file "ConsoleSinter.exe" on your computer since it will call this executable to launch ACM dynamic simulations. The "ConsoleSinter.exe" should be installed with "SinterConfigGUI" in the "SimSinter" CCSI tool package. When configured, the full path of the "ConsoleSinter.exe" will be saved in "C:\ProgramData\CCSI\DRMBuilder\config.txt" file such that the executable file will be found immediately when the D-RM Builder is started next time. If the SimSinter tool package is installed in the default directory "C:\Program Files (x86)\CCSI\SimSinter", D-RM Builder will configure and create the "config.txt" automatically. Otherwise, a popup window named "SinSimter Installation Status" will be displayed as shown in Figure 8. If the "SinSimter" has not been installed, click the "No" button and the "D-RM Builder" program will exit. Otherwise, click the "Yes" button and a dialog window named "Select SimSinter File" will be displayed as shown in Figure 9. Browse to the folder where the "ConsoleSinter.exe" is located and select the file and click the "Open" button to close the dialog window. If a valid "ConsoleSinter.exe" is selected, the full path of the file will be written to the "config.txt" file. Meanwhile, the main window of the D-RM Builder will be displays as shown in Figure 10.

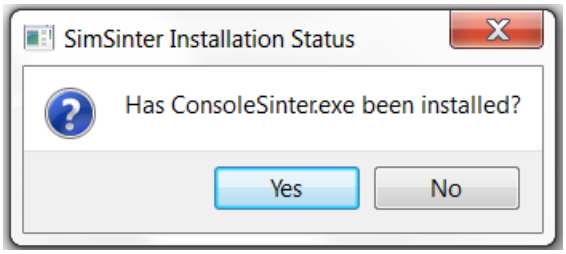

Figure 8: Dialog Window Asking If SimSinter Has Been Installed

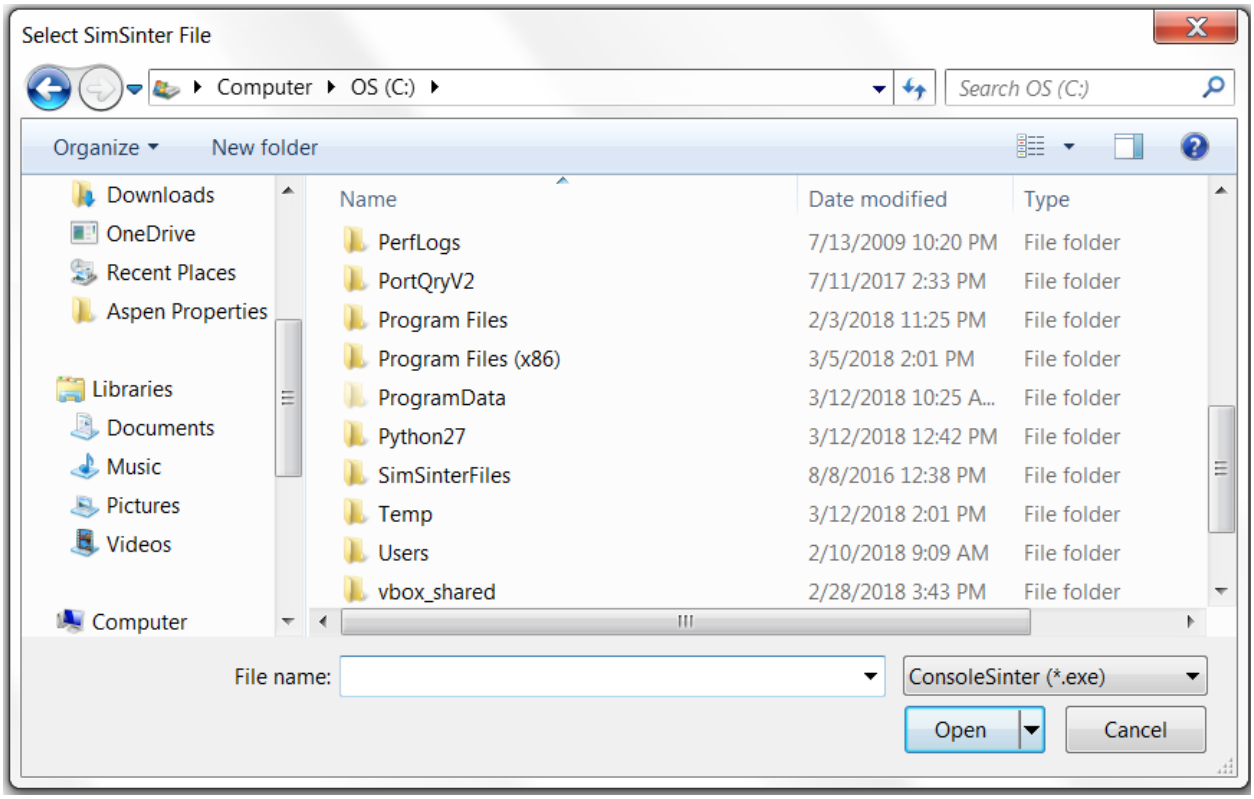

Figure 9: Dialog Window for Selecting ConsoleSinter.exe File

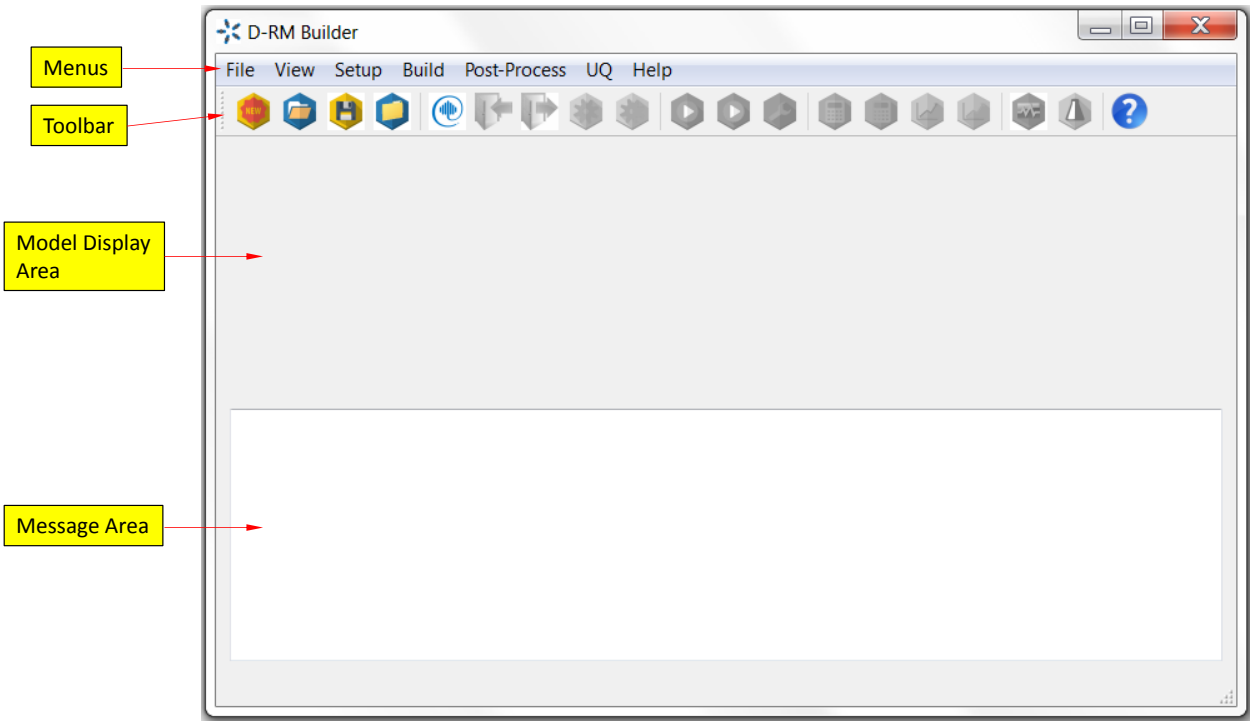

Figure 10: D-RM Builder Main Window

15. The D-RM Builder UGI is similar to a classical Microsoft Windows application with menus and a toolbar at the top, a gray area for displaying the input and output variables (initially blank), a client area with white background 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 are shaded (disabled) initially. They are enabled later when the status of the D-RM building process is updated. When the mouse is moved over a menu or a toolbar icon, a tip about the command displays over the menu or icon while a short description about the command displays in the status bar at the bottom of the embedded window. If a menu ends with "...", it brings up a dialog window for more user inputs when clicked. Initially a blank project is displayed and the user can start to work on the blank project by choosing a high-fidelity model (ACM model) configured in Steps 1–13 through the **Setup** menu. The **File** menu contains **New**, **Open**, **Close**, **Save**, **Save As**, and **Export** submenus. Add a new case to the main window by issuing the **File→New** command or clicking on the toolbar icon . If the current project is not saved, the D-RM Builder will ask the user to save the project. Open an existing case by issuing the **File→Open** command or clicking the toolbar icon . Again, the D-RM Builder will ask the user to save the project if the current project has been modified. Close the current case by issuing the **File→Close** command or clicking the toolbar icon and a blank project will be displayed. Save a case by issuing the **File→Save** command or clicking the toolbar icon . 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, the **Post-Process** menu, and finally to the **UQ** 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.

16. To start the D-RM building process for a blank case, select a high-fidelity model first. Issue **Setup→Choose High-Fidelity Model** command or click the toolbar icon  . A "SimSinter Configuration File" window displays for file browsing and selecting. Browse and select the "VdV_Reactor.json" file created in Step 12. This loads the configuration file into the DRM Builder. A text message confirming the file selection and loading displays in the client area of the D-RM Builder window.

17. The SinterConfigGUI allows a user to select the dynamic input and output variables. Further configuration among the selected input and output variables are required to setup the D-RM Builder project. To configure the selected ACM input variables, navigate to the **Setup→Configure Input Variables** command or click the toolbar icon  . The "Input Variable Dialog" window displays as shown in Figure 11.

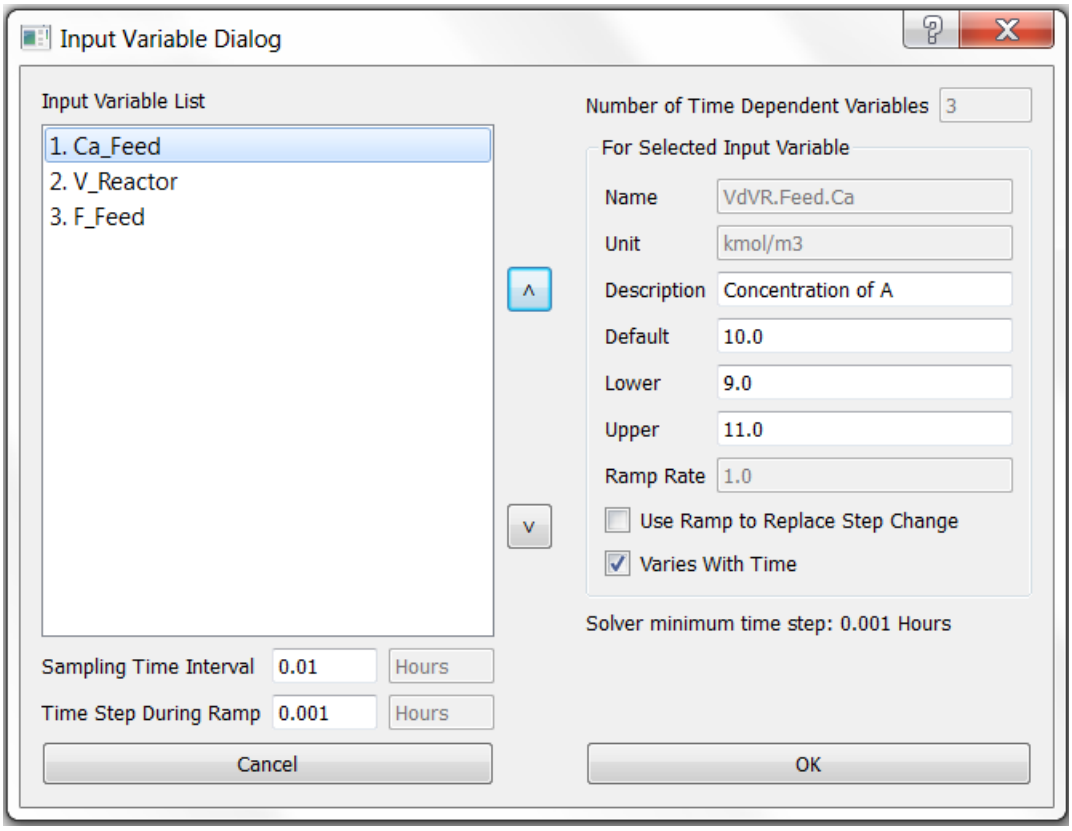

Figure 11: Input variable dialog window for VdV Reactor model

The input variables are listed in the "Input Variable List" list box. Note: The order of the variables listed is not necessarily the order listed in SinterConfigGUI. The user can change the order of the input variables in the list by clicking the "Up" and "Down" arrow buttons near the right side of the list box. For this tutorial, leave the order unchanged. Click each of the variables to see the description, unit, and default ranges (10%) used for conducting training/validation. The "Name" and "Unit" text boxes are for read-only. The name of the variable here represents an ACM's internal variable path with a local variable name appended to a block name. If the input variable is dimensionless, a space is displayed in the "Unit" text box. The user can edit the "Description" text box as desired. 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 second input ("V_Reactor") 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 Variables" 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 default value displayed at the read-only "Default" text box, which is usually the steady-state value used in the original ACM model. Likewise, the upper limit cannot be smaller than the default value. If this rule is broken, a warning message dialog box displays when the user clicks the "OK" button and the user cannot exit the dialog box unless the user clicks the "Cancel" button, 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 input 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. The sampling time interval should be comparable with the time constants of the output variables in response to the change of input variables. The "Solver minimum time step" label displays the minimum integration time step used by the ACM solver. The sampling time interval should be higher than the solver minimum time step. The "Time Step During Ramp" is not applicable in this case since no ramp change is used. If the ramp changes are used to replace step input changes, the "Time Step During Ramp" needs to be specified to a value that is greater than the minimum solver time step and less than the sampling time interval. Leave the default sampling time interval of 0.01 hours unchanged. 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. Two arrows connected to a black box are plotted in the upper half of the D-RM Builder window, representing the two input variables.

18. Configure the output variables by issuing the **Setup→Configure Output Variables** command or clicking the toolbar icon  . The "Output Variable Dialog" window as shown in Figure 12 displays.

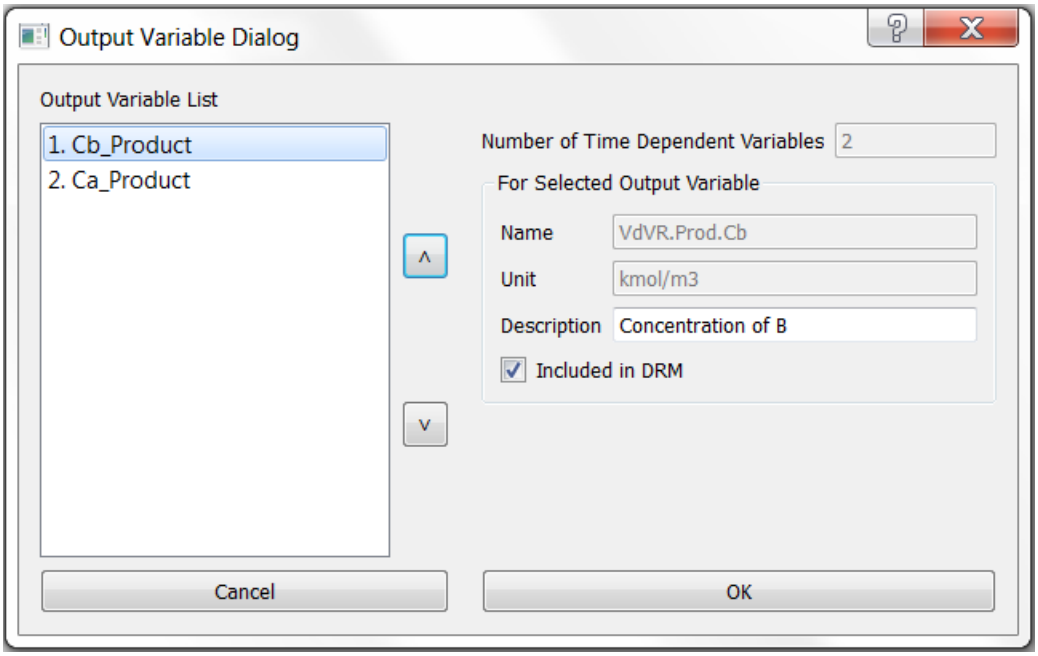

Figure 12: Output Variable Dialog Window

There are two output variables for the reactor. Again, the order of the output variables listed here is not necessarily the same as the order listed in SinterConfigGUI. The user can use the "Up" and "Down" arrow buttons near the list box to change the order of the output variables. 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 inside SinterConfigGUI. 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. Two arrows connected to the right side of the black box are plotted in the upper half of the D-RM Builder window, representing the two output variables.

19. 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 will be simulated by the high-fidelity model. To prepare the training sequence, select the **Setup→Prepare Training Sequence** command or click the toolbar icon  . The "Step Change Sequence Dialog" window displays as shown in Figure 13.

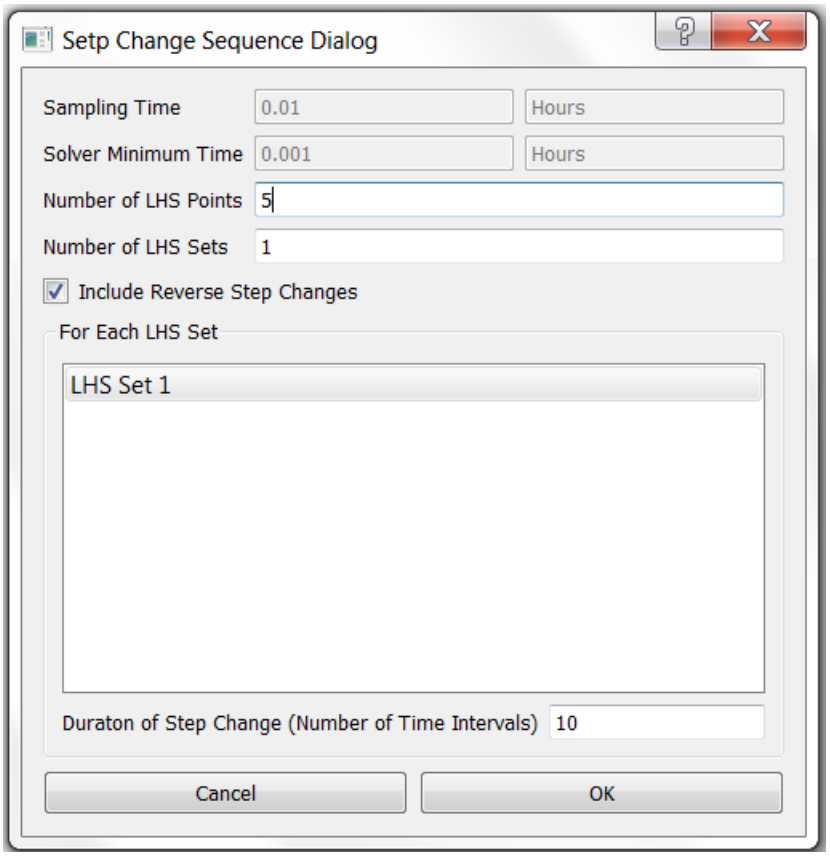

Figure 13: Step Change Sequence Dialog Window

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. Figure 14 shows the dialog window after the user inputs described above. Click "OK" to accept the inputs. A confirmation message displays in the client area of the main window.

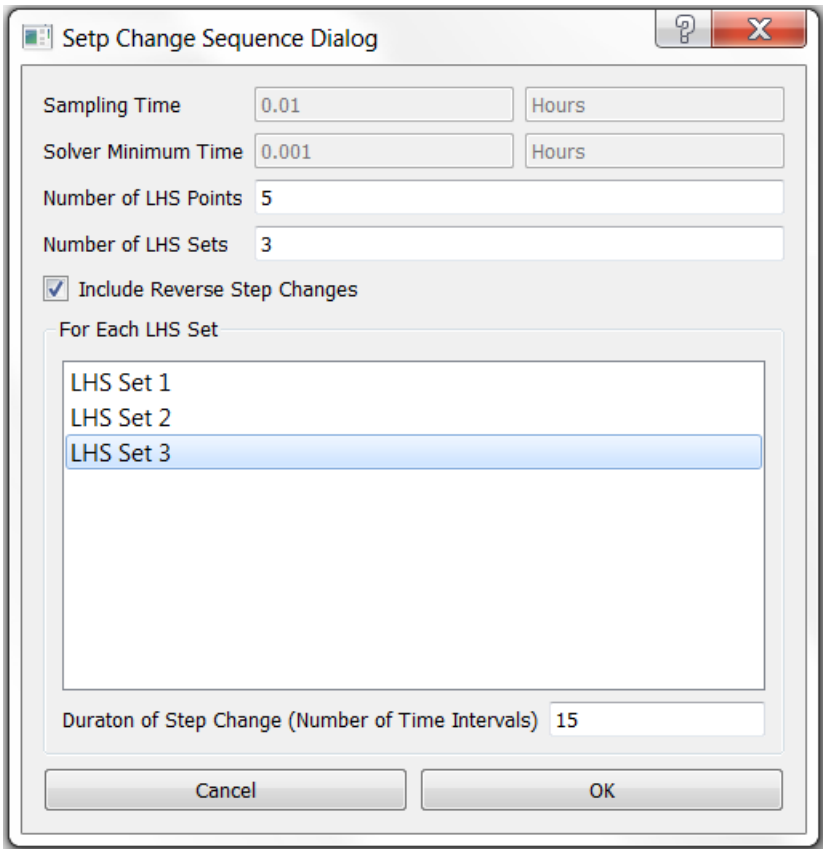

Figure 14: Step Change Sequence Dialog Window for Training After User Modification

20. Prepare an input change sequence for validation by issuing the **Setup→Prepare Validation Sequence** command or clicking the toolbar icon  . The same "Step Change Sequence Dialog" window as shown in Figure 13 displays.

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. Figure 15 shows the dialog window after user modification described above. Click "OK" to accept the specifications. A confirmation text message displays in the client area of the main window.

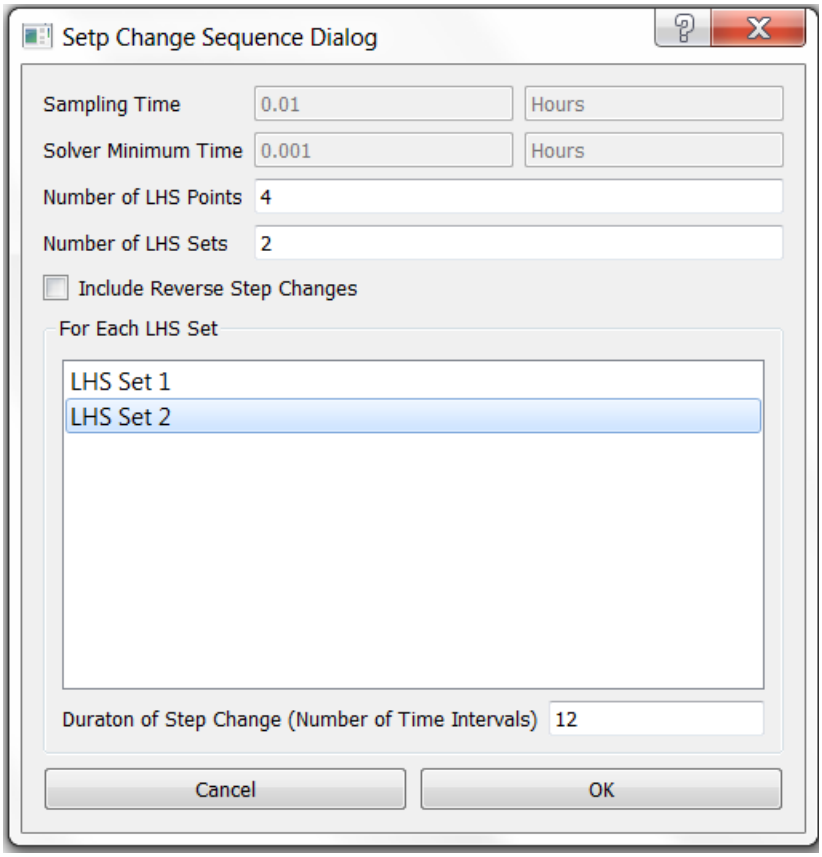

Figure 15: Step Change Sequence Dialog Window for Validation After User Modification

21. Save the file by issuing the **File→Save** or **File→Save As** command or clicking the toolbar icon  . The "Save D-RM Builder File" dialog window displays. Enter "case1_VdV" in the "File name" text box and then click "OK". A text file, "case1 VdV.drmb", is written to the current working directory. Note: The extension of the D-RM Builder case file is "drmb". The file contains the user inputs that have been entered so far in JSON format. 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 he or she left when the case was saved.

Warning: If the **File→Save As** command is issued and the D-RM Builder case file is saved to a new working directory. Please also copy the ACM file (with the ".acmf" extension) and the SimSinter configuration file (with the "json" extension) to the same directory. The two files have to be in the same working directory such that the ACM solver can find them and run dynamic simulations.

22. Launch the ACM high-fidelity model simulation for training by issuing the **Build→Perform Training Simulation** command or clicking the toolbar icon  . This creates temporary files and a folder in the current working directory and brings up an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACMs "Simulation Message" window is turned on. The mouse icon is switched from normal (an arrow) to busy (rotating circle). Wait without closing the ACM window. After the high-fidelity model simulation is completed and the ACM window is closed, a message displays, confirming the successful completion of the high-fidelity model simulation. The mouse icon becomes a normal arrow icon. Note: The D-RM Builder calls "ConsoleSinter.exe" to perform dynamic ACM simulation and the path to the executable file is configured when the D-RM Builder is launched the first time after its installation.

23. Launch the ACM high-fidelity model simulation for validation by issuing the **Build→Perform Validation Simulatio**n command or clicking the toolbar icon  . Note: The background color of toolbar buttons for training are red and those for validation are green. Wait until the ACM simulation is completed and the message confirming the successful completion of the simulation displays.

24. 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 16. 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.

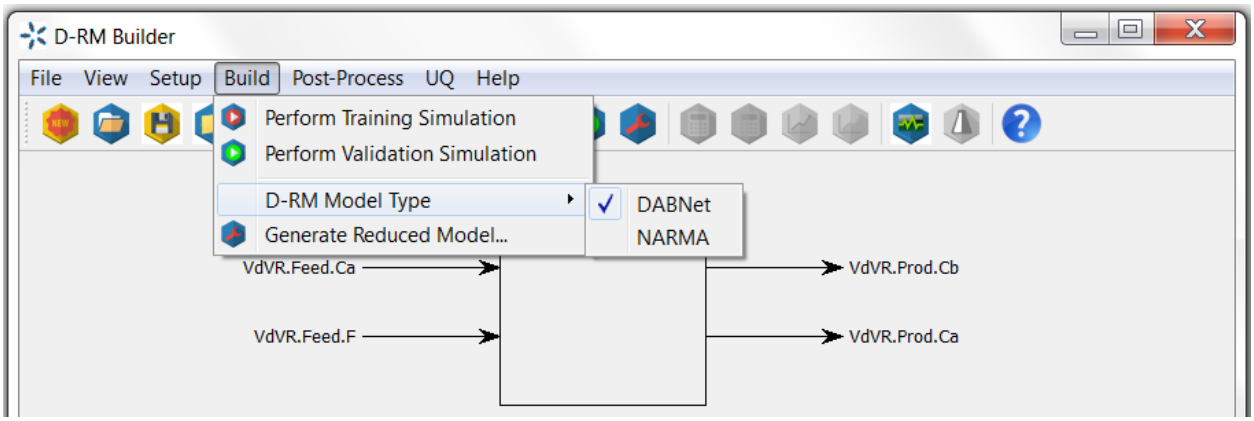

Figure 16: DABNet Submenu for Selecting a D-RM Model Type

25. Build the DABNet model by issuing the Build→Generate Reduced Model command or clicking the toolbar icon  . The "DABNet DRM Parameter Dialog" window displays as shown in Figure 17.

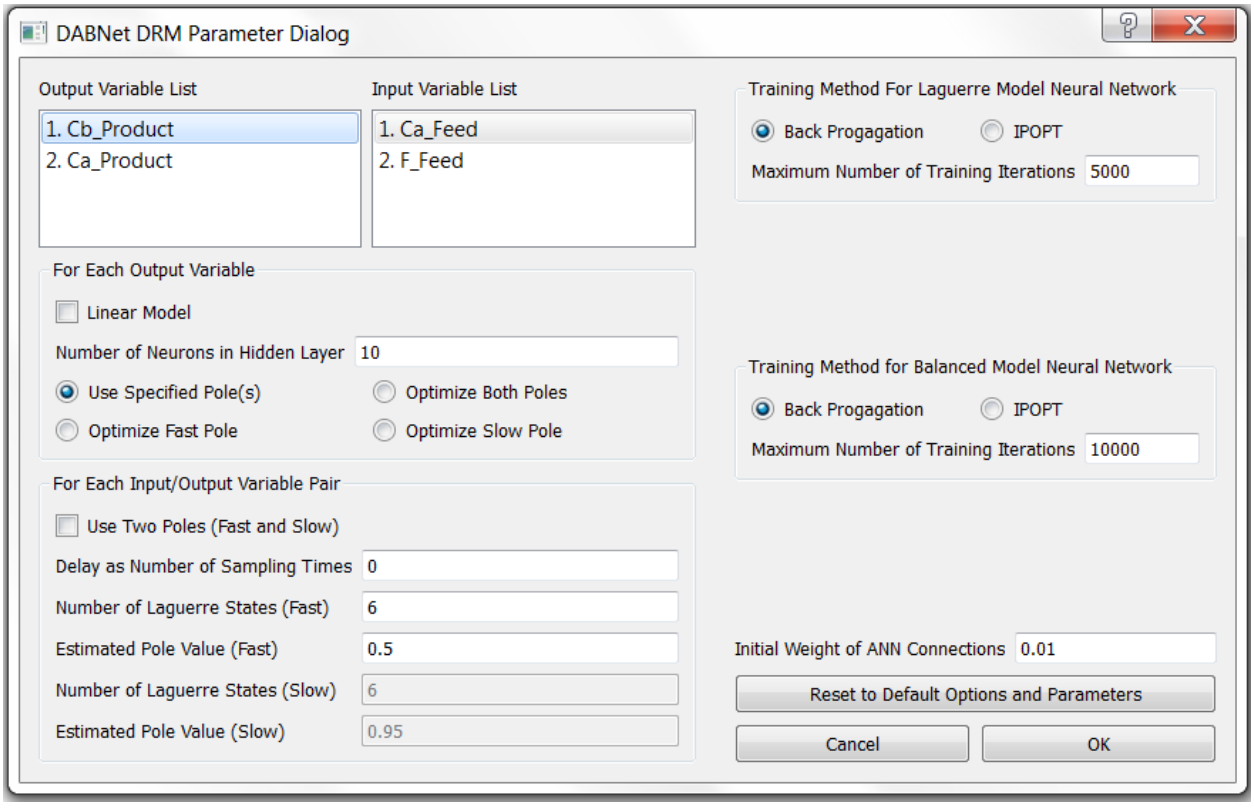

Figure 17: DABNet DRM Parameter Dialog Window for VdV Reactor Model

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

models is generated if multiple output variables exist. In this case, there are two output variables and, therefore, two DABNet models are generated. There are different options for building DABNet D-RM for each output variable. In the "For Each Output Variable" section, the check box "Linear Model" is for the option to build linear D-RM using linear activation function in the ANN that maps the state-space variables to the output. If this check box is selected, the "Number of Neurons in Hidden Layer" will be automatically set to 1. In this case, leave that check box cleared, indicating that a nonlinear D-RM is to be built. Leave the "Number of Neurons in Hidden Layer" to the default value of 10. Select the default button of "Use Specified Pole Value". The other three buttons are those for optimizing the pole values. One major enhancement of D-RM Builder on the original DABNet model is the new formulation using two poles to handle a process with both fast and slow dynamics. If the double-pole option is needed for those two-time-scale systems, there are three options to optimize the pole values including both fast and slow poles, fast poles only, and slow poles only. If the double-pole option is not selected, the user can select either the Optimize Both Poles button or the Optimize Fast Pole button to allow D-RM Builder to optimize the poles. For each input/output pair, the model parameters include the option to use double-pole formulation, time delay in terms of number of sampling time intervals, and the number of Laguerre states. If double-pole option is selected, two sets of pole values and number of Laguerre states need to be specified, one for fast and the other for slow dynamics. 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 on 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 method 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 its console window. Read the messages and pay 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 (Cb_Product) is approximately 10^{-5} (based on the balanced model) and that for the second output (Ca_Product) is approximately 1.14×10^{-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.

26. Save the case by issuing the **File→Save** command or clicking the toolbar icon  . 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.

27. 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.

28. 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 the toolbar icon  . 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.

29. With both ACM and D-RM predictions for the response of the training sequence available, issue the **Post-Process→Plot Training Responses** command or click the toolbar icon  to visualize the results. The "Result Plotting Dialog" window as shown in Figure 18 displays.

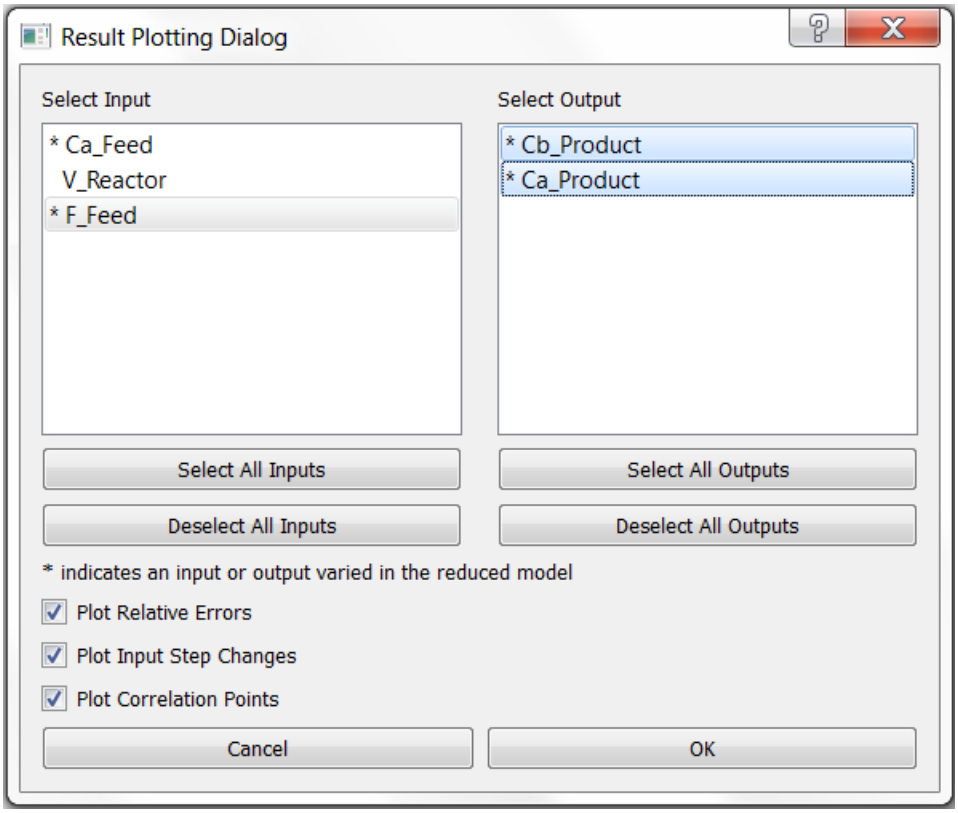

Figure 18: Result Plotting Dialog Window for VdV Reactor Model

The "Result Plotting Dialog" window lists the input and output variables for the user to plot. 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 and the third input variables and the two output variables as shown in Figure 18. 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 buttons in the lower part of the window. Accept the defaults with all of the check boxes selected and then click "OK". A Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 19.

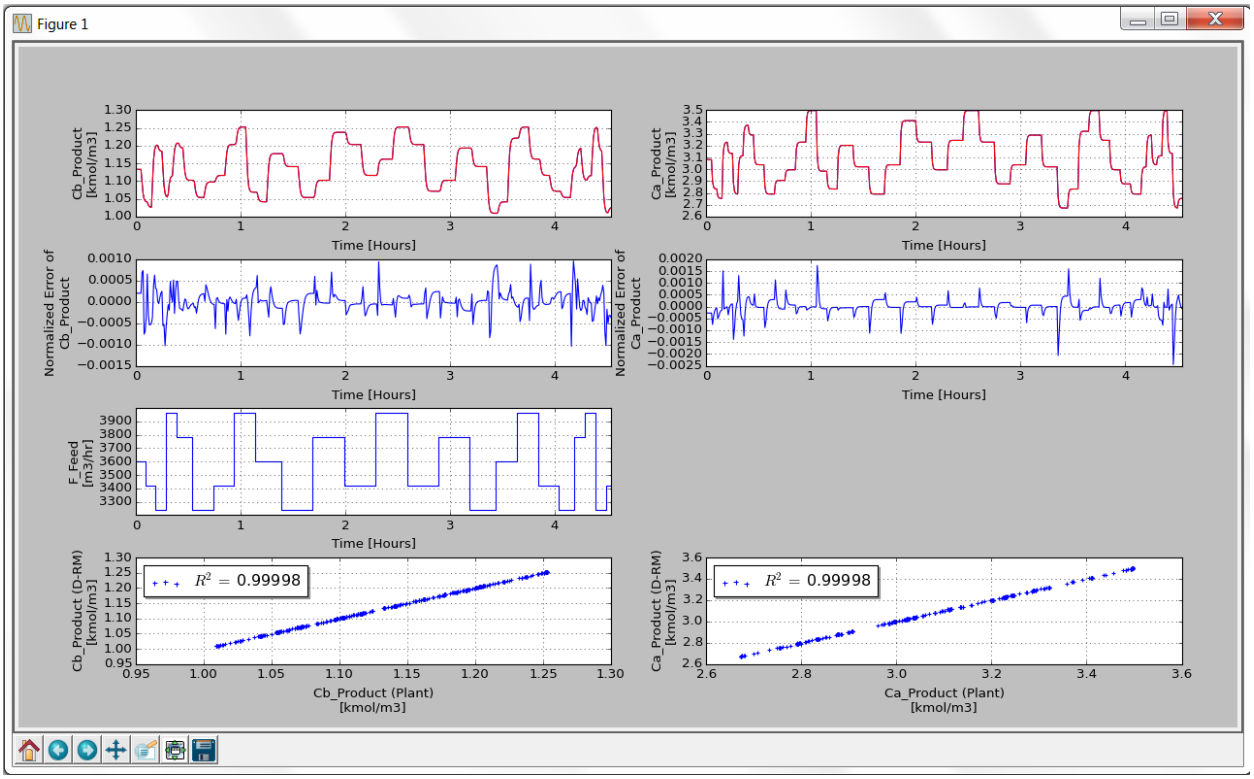

Figure 19: Plots of Input and Output Variables as Functions of Time for Training Data of VdV Reactor Model

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 blue 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 blue 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 19, 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 window by clicking the "x" at the upper right corner. If needed, issue the **Post-Process→Plot 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 18 to remove some of the rows of figures so the user can see the larger figures of interest.

30. 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 the toolbar icon  .

31. Issue the **Post-Process→Plot Validation Responses** command or click the toolbar icon  . The "Result Plotting Dialog" window as shown in Figure 18 displays. Select the first and the third input variables and the two output variables and click "OK". A Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 20.

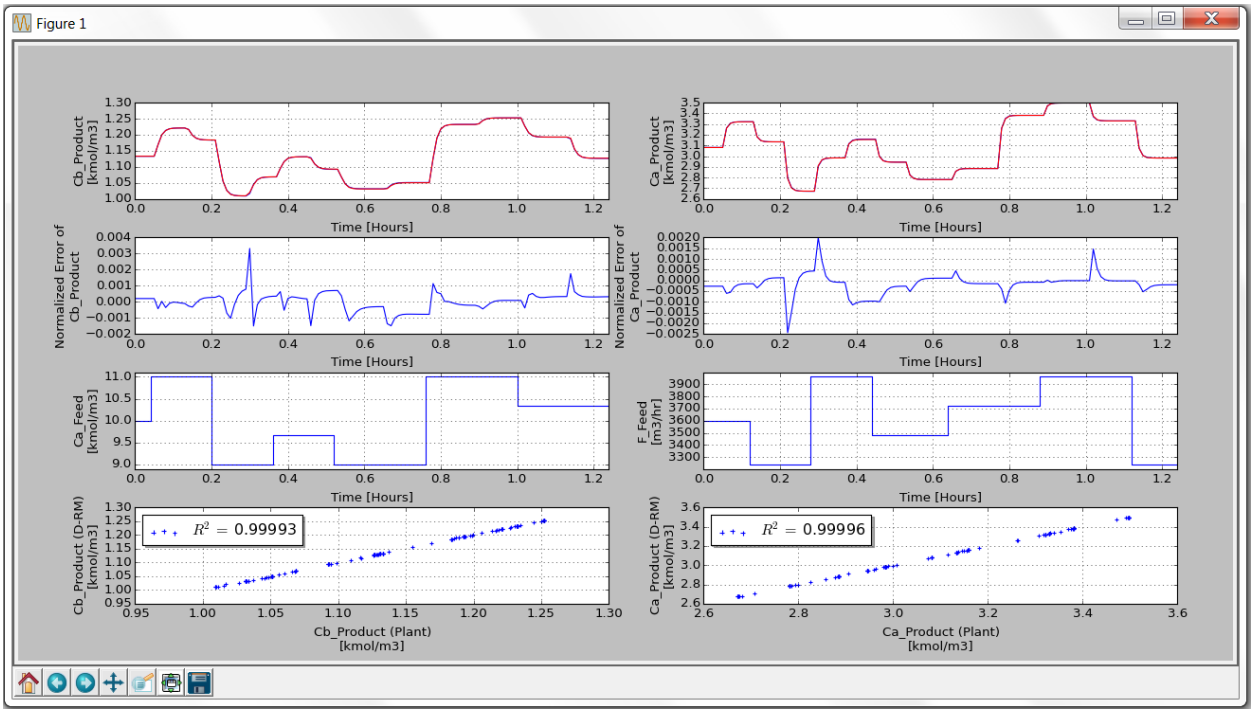

Figure 20: Plots of Input and Output Data as Functions of Time for Validation Data

Notice that the red curves are still on the top of the blue curves for both the Ca_Product and Cb_Product outputs, indicating a quite accurate D-RM generated by the D-RM Builder. Close the plot window by clicking the "x" at the upper right corner.

32. Save the file by clicking the toolbar icon  . The case file contains the data predicted by the D-RM generated for both the training and validation sequences.

33. 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→Specify Noise** command or click the toolbar icon  . The "Noise Specification Dialog" window as shown in Figure 21 displays.

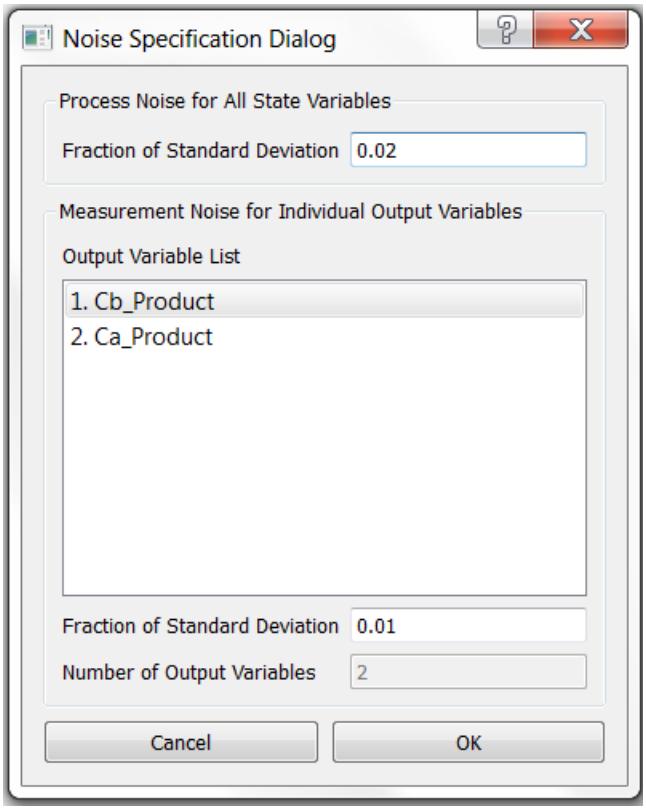

Figure 21: Process and Measurement Noise Dialog Window for VdV Reactor Model

Specify the noise as the fraction of the standard deviation of the state variables in the training sequence. Note: Since the state variables could be either positive or negative in the normal operating conditions, the standard deviation of the state variables calculated in the training data is used as the basis for the noise level of the state variables. In the current version, the ratio of the noise to the standard deviation of the 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 the current version of D-RM Builder. The user can specify a single number for the noise level. For this tutorial, use the default value of 0.02 in the "Fraction of Standard Deviation" text box in the "Process Noise for All State Variables" section. This means that the noise defined as the standard deviation of each state-space variable is 2% of the standard deviation of the variable itself found in the training sequence. 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. Cb_Product" in the "Output List" and then enter 0.05 in the "Fraction of Standard Deviation" text box in the "Measurement Noise For Individual Output Variables" section. Then, select the second variable "2. Ca_Product" and then enter 0.05. Click "OK" to accept the noise specifications.

34. Ask the D-RM Builder to perform 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 to 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→Analyze** command or click the toolbar icon  . The "Result Plotting Dialog" window as shown in Figure 18 displays, asking the user to specify the options for plotting the results of the UQ analysis. Since the volume of the reactor is constant, the user can select only the first and the third 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 22.

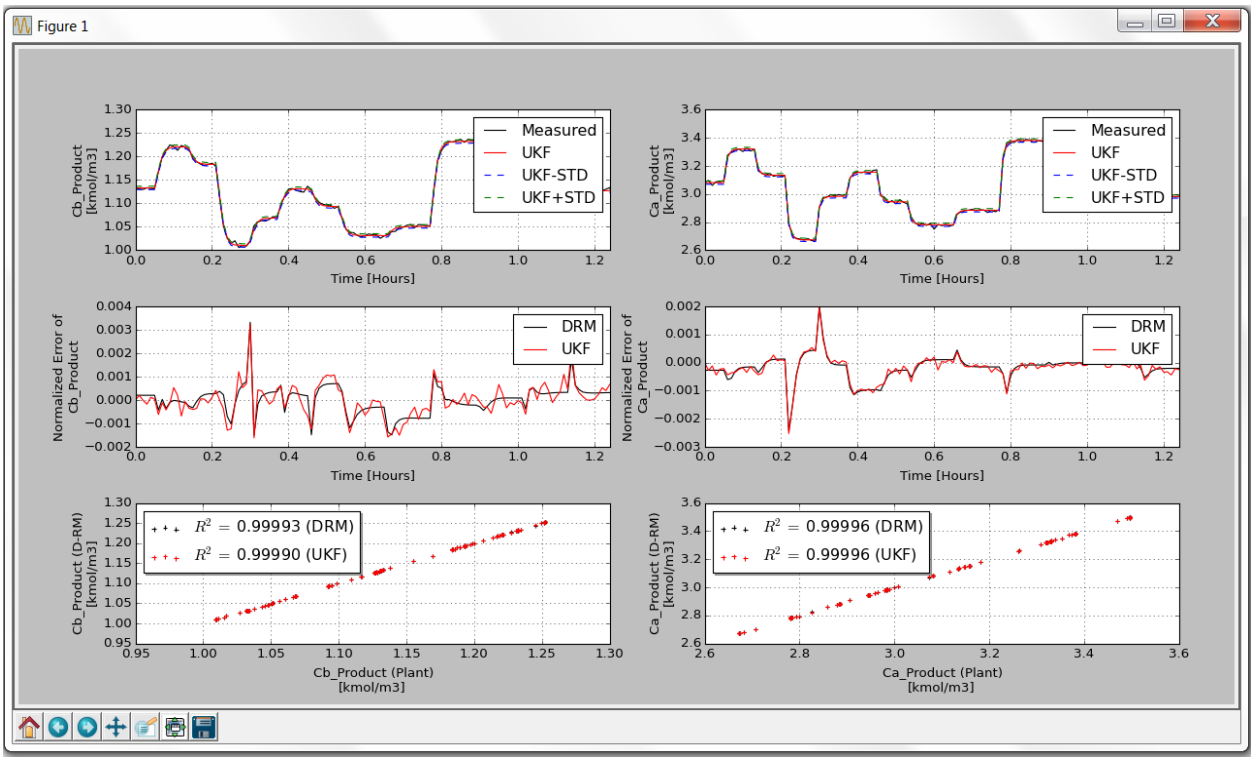

Figure 22: Plots Showing the Results of UQ Analysis for the Validation Sequence

Note: Due to the random nature of the noise added to the ACM prediction as the measurement data, the plotted figures could be slightly different from those shown in Figure 22. Adjust the size of the plot window. 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 (+/- the standard deviation from the calculated covariance matrix). The relative errors and correlation plots are similar to those shown in Figure 20 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 5% of the standard deviation of the output variable in the training data.

35. Save the case by clicking the toolbar icon  . Export the covariance matrices at the end of the validation sequence by issuing the **File→Export→Covariance Matrices** command. The "Save Covariance Matrices" dialog window displays. Click Save to accept the default name for the file. The exported matrices are written in "csv" format.

36. Export the generated D-RM to a MATLAB script file by issuing the **File→Export→DRM As Matlab Script File** command. The "Save D-RM As Matlab Script" dialog window displays. Use the default file name "case1_VdV.m" and then click "OK" to save the D-RM to the

"case1_VdV.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.

37. Save the log messages that have been displayed in the client area of the main window by selecting the **File→Export→Messages** to Log File command. The "Save Message Log" dialog window displays. Use the default file name "case1_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.

38. 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.

39. Close the main window of the D-RM Builder by issuing the **File→Close** command or by clicking the toolbar icon  to complete the model generation process.

2.2. Building a D-RM Using DABNet Framework and IPOPT Based ANN Training

In the previous example, the default back propagation (BP) 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 BP 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. The Van-de-Vusse reactor example in the previous tutorial is modified in this tutorial. The following are the steps to build the D-RM using DABNet framework and IPOPT-based ANN training:

1. Copy the "case1_VdV.drmb" file created in the previous tutorial to a new working directory and rename it as "case2_VdV.drmb". This tutorial uses the case file "case2_VdV.drmb" as the starting point and revises the D-RM training options. As a good practice, copy the ACM file "VdV_Reactor.acmf" and the SimSinter configuration file "VdV_Reactor.json" to the new

directory. Again the three files have to be in the same directory in case ACM dynamic simulations need to be performed.

2. Start D-RM Builder form Windows "Start" button or the corresponding shortcut. The main window of the D-RM Builder displays as shown in Figure 10.

3. Issue the **File→Open** command or click the toolbar icon  to bring up the "Open D-RM Builder File" dialog. Browse to the working directory for this tutorial, select the "case2_VdV.drmb" file, and click the "Open" button.

4. Select the **Build→Generate Reduced Model** command or click the toolbar icon  . The "DABNet DRM Parameter Dialog" window as shown in Figure 17 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 is 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 console window. The mouse icon is switched from a normal (an arrow) icon to a busy (rotating circle) 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 to the normal arrow icon.

5. Use the commands under the **Post-Process** menu to predict the training and validation responses using the new D-RM and visualize the results. 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.

6. 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 under the UQ menu.

7. Export the new D-RM to a file with an ".m" extension. Save the case file and then close the D-RM Builder application.

2.3. Building a D-RM Using DABNet Framework With Pole Values Optimized

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. [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 τ 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 nonlinear systems, 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.

To demonstrate the feature, an example named "pH Neut" is provided in a subfolder named "pH" in the D-RM Builder's "examples" folder. This example models the two-tank pH neutralization reactor described in [1]. The default options are applied first 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 a data-driven D-RM using DABNet framework with pole values optimized:

1. Copy the ACM file of the pH Neut model "pH_Neut.acmf" in the "examples\pH" folder of the D-RM Builder installation directory to a new working directory. Confirm the user of the D-RM Builder has write-permissions to this working directory.

2. Launch SinterConfigGUI. The GUI window as shown in Figure 1 displays.

3. Click the "Browse" button and browse to the "pH_Neut.acmf" file in the working directory. Then click the "Open File and Configure Variables" button. It takes a few seconds for the

SinterConfigGUI to display a “SinterGonfigGUI Simulation Meta-Data” window as shown in Figure 23. An ACM window with pH Neut model inside also displays as shown in Figure 24.

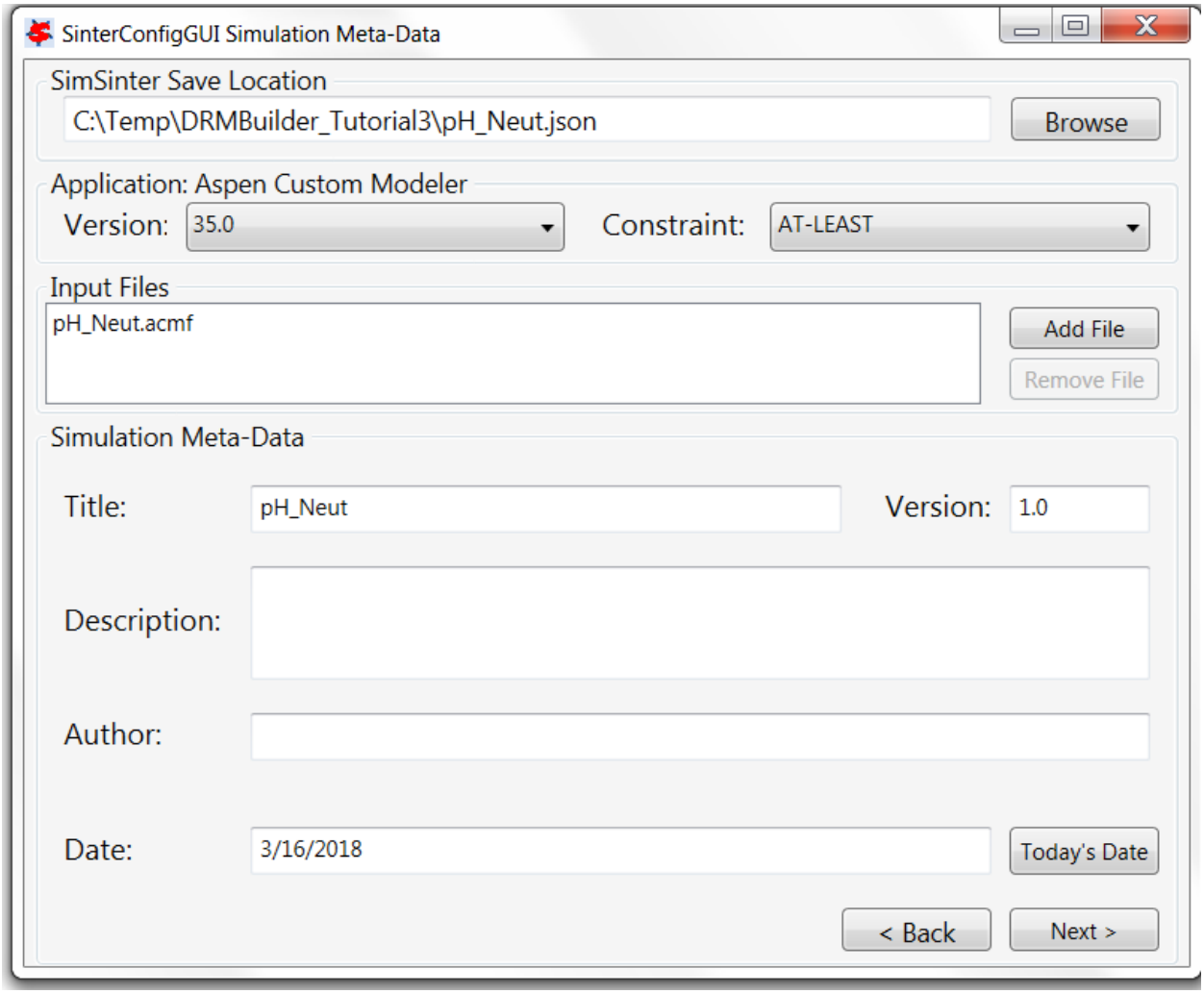

Figure 23: Dialog Window for the Simulation Meta-Data of pH Neut Model

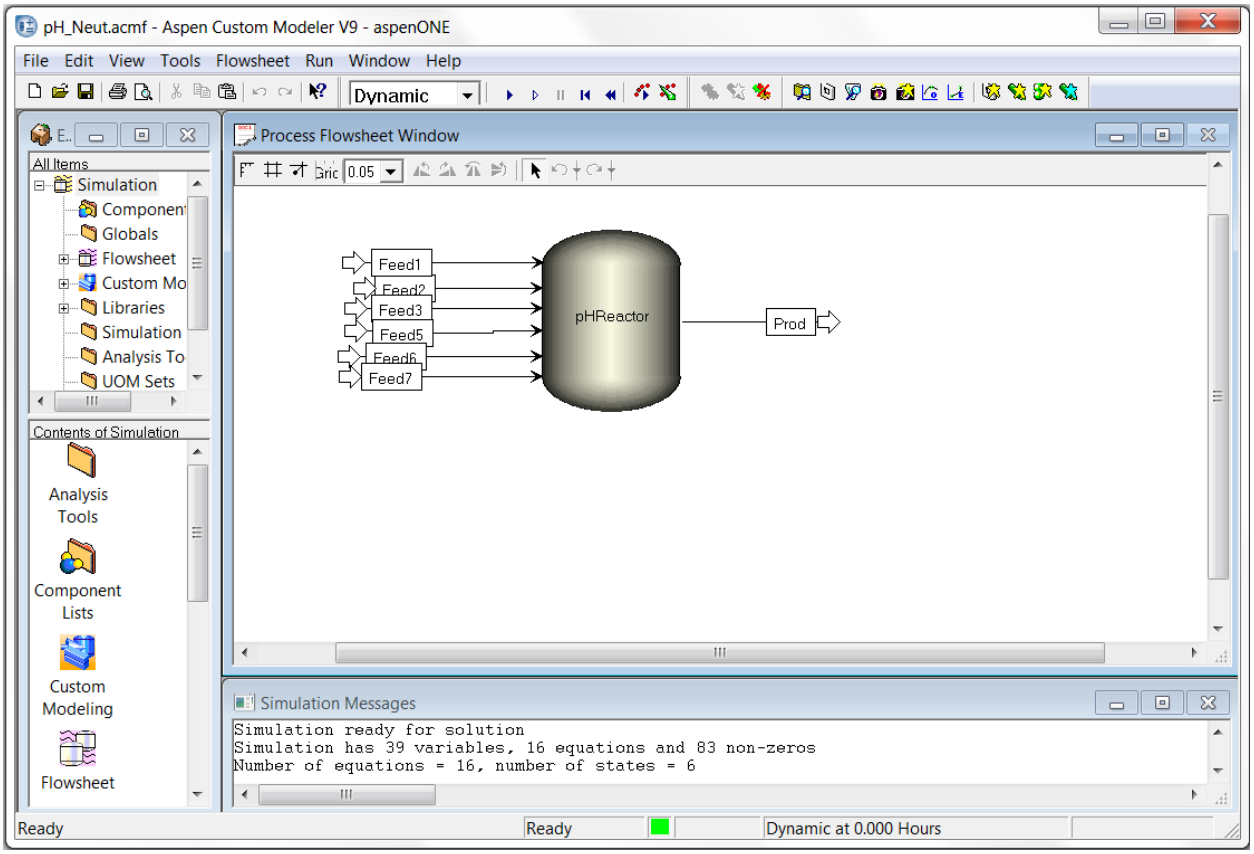

Figure 24: ACM Window for pH Neut Model Displayed by SinterConfigGUI

4. Enter “pH Neutralization Example” in the Title text box, “My third DRMBuilder tutorial” in the Description text box, and the user’s name in the Author text box. Then click the “Next” button. A “SinterConfigGUI Variable Configuration Page” window similar to Figure 4 displays.

5. Enter “~” in the “Variable Search Pattern” text box and click the “Search” button. All the variables in the pH Neut model are listed in the list box at the lower left corner of the window as shown in Figure 25.

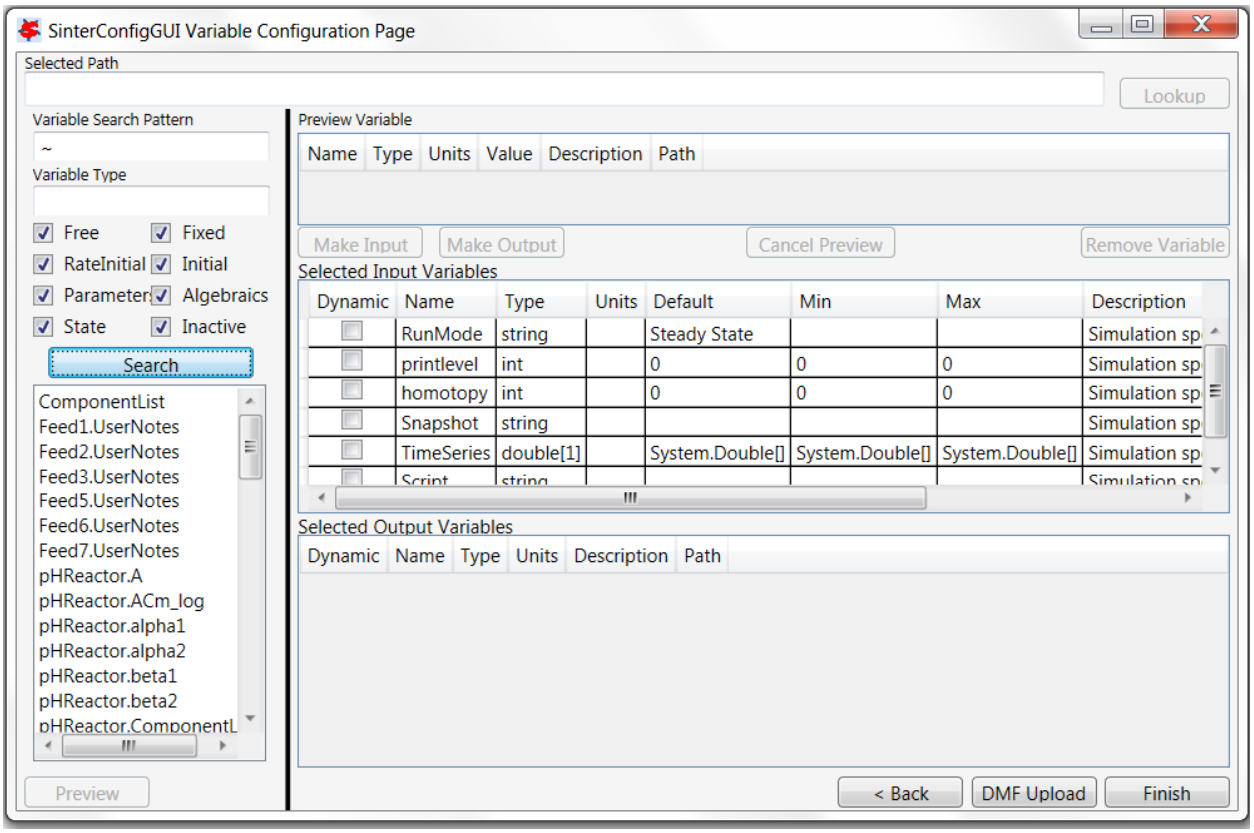

Figure 25: Variable List on Configuration Page of pH Neut Model

6. Select the “pHReactor.Feed1.F” in the list box and click the Lookup button at the upper right corner. The variable is displayed in the “Preview Variable” section. Click the “Make Input” button and the variable is appended to the end of the “Selected Input Variables” table. Change the name of the variable to “F_Feed1” and select the checkbox in the Dynamic column. This makes the selected variable a dynamic input variable.

7. Select the “pHReactor.Feed2.F” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “F_Feed2”.

8. Select the “pHReactor.Feed3.F” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “F_Feed3”.

9. Change the default value of the input variable named “RunMode” in the “Selected Input Variables” table from “Steady State” to “Dynamic”.

10. Select the “pHReactor.pH” in the list box and click the “Preview” button. Then click the “Make Output” button and the variable is added to the “Selected Output Variables” table. Change the

name of the variable to “pH_Product” and select the checkbox in the “Dynamic” column. This makes the selected variable a dynamic output variable.

11. Select the “pHReactor.Prod.Ca” in the list box and repeat Step 10 to make it a dynamic output variable and change the name to “Ca_Product”.

12. This concludes the configuration of ACM variables. Figure 181 shows the SinterConfigGUI window after the configuration. Click the “Next” button at the lower right corner and a “SinterConfigGUI Vector Default Initialization” dialog window display as shown in Figure 26. Leave the default time series data unchanged.

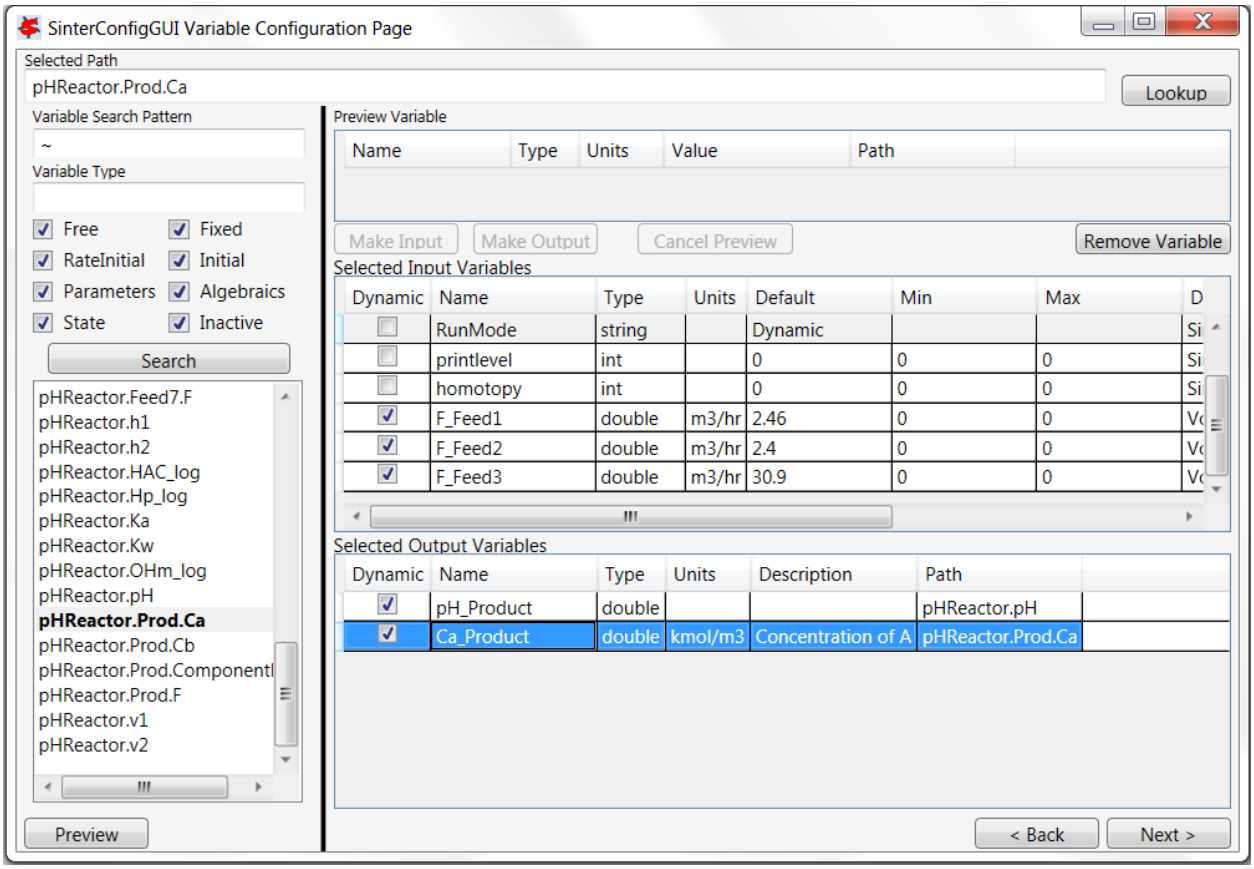

Figure 26: Configured Dynamic Variables of pH Neut Model by SinterConfigGUI

13. Click the Finish button to exit the “SinterConfigGUI”. A file named “pH Neut.json” is created in the working directory. Close the ACM window.

14. After the dynamic input and output variables are configured and the JSON file is created, start D-RM Builder. The D-RM Builder main window containing a blank project displays as shown in Figure 10.

15. To start the D-RM building process, a high-fidelity model (the pH Neut ACM model in this case) needs to be selected. Issue **Setup→Choose High-Fidelity Model** command or click the toolbar icon  . A "SimSinter Configuration File" window displays for file browsing and selecting. Browse and select the "pH_Neut.json" file created in Step 12. This loads the configuration file into the D-RM Builder. A text message confirming the file selection and loading displays in the client area of the D-RM Builder window.

16. Configure the input variables. Navigate to the **Setup→Configure Input Variables** command or click the toolbar icon  . The "Input Variable Dialog" window displays as shown in Figure 27.

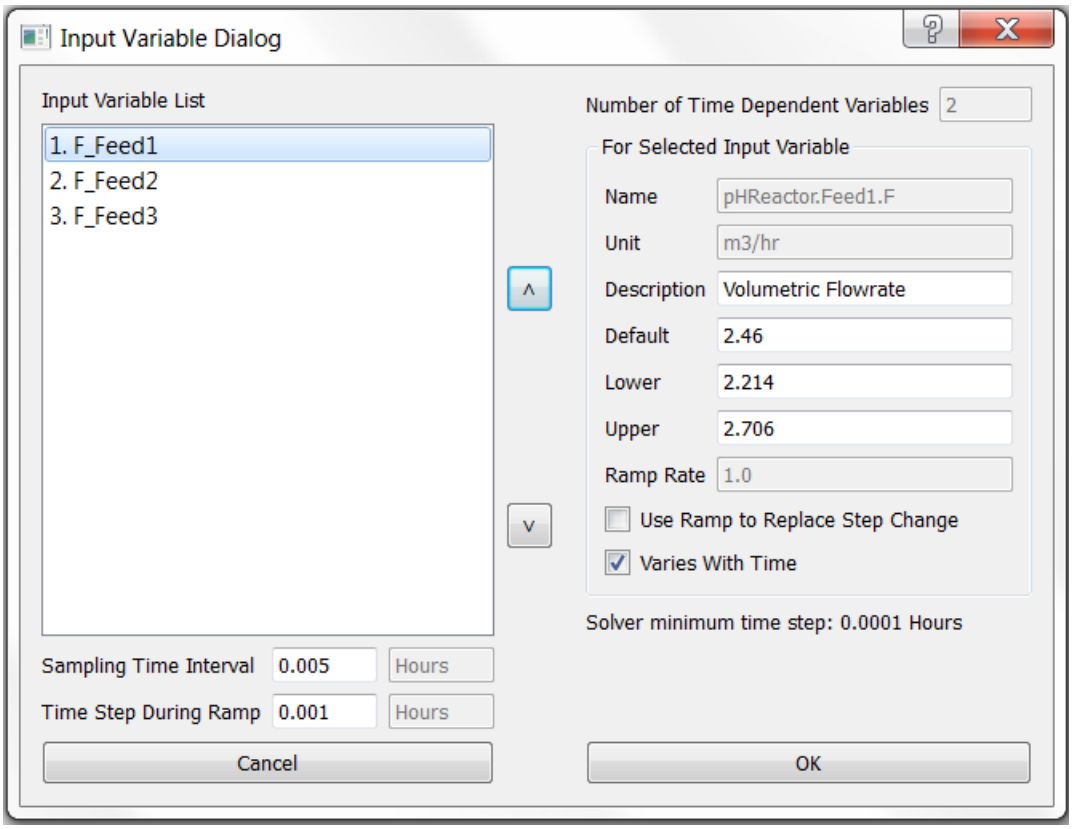

Figure 27: Input Variable Dialog Window for pH Neut Model

For this example, leave the default values of the lower and upper limits for the first and third variables unchanged. Click the "F_Feed2" 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). Enter 0.005 in the "Sampling Time Interval" text box. Click "OK" to accept the modifications and exit the window.

17. Configure the output variables by issuing the **Setup→Configure Output Variables** command or clicking the toolbar icon  . The "Output Variable Dialog" window displays. Accept the default options by clicking "OK".

18. To prepare the training sequence, select the **Setup→Prepare Training Sequence** command or click the toolbar icon  . The "Step Change Sequence Dialog" window displays. For this example, enter 5 in the "Number of LHS Points" text box and then enter 5 in the "Number of LHS Sets" text box. Five 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. Select the fourth row "LHS Set 4" item in the list box and then enter 5 in the "Duration of Step Change" text box. Select the fifth row "LHS Set 5" item in the list box and then enter 10 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.

19. Prepare an input change sequence for validation by issuing the **Setup→Prepare Validation Sequence** command or clicking the toolbar icon  . The "Validation Sequence Dialog" window displays. Enter 5 in the "Number of LHS Points" text box and then enter 2 in the "Number of LHS Sets" text box. Two 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. Click "OK" to accept the change and then close the window.

20. Save the file by issuing the **File→Save As** command. The "Save D-RM Builder File" dialog window displays. Enter "case3 pH Neut" in the File name text box and then click OK. A text file, "case3_pH_Neut.drmb", is written to the current working directory.

21. Launch the ACM high-fidelity model simulation for training by issuing the **Build→Perform Training Simulation** command or clicking the toolbar icon  . This creates temporary files and a folder in the current working directory and displays an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACMs "Simulation Message" window is activated. The mouse icon is switched from normal (an arrow) to busy (rotating circle). Wait without closing the ACM window. After the high-fidelity model simulation is completed and the ACM window is closed, a message displays in the client window, confirming the successfully completion of the high-fidelity model simulation. The mouse icon becomes a normal arrow icon.

22. Launch the ACM high-fidelity model simulation for validation by issuing the **Build→Perform Validation Simulation** command or clicking the toolbar icon  . Wait until the ACM simulation is completed and the message confirming the successful completion of the simulation displays.

23. 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 16.

24. Build the DABNet model by issuing the **Build→Generate Reduced Model** command or clicking the toolbar icon  . The "DABNet DRM Parameter Dialog" window displays as shown in Figure 28.

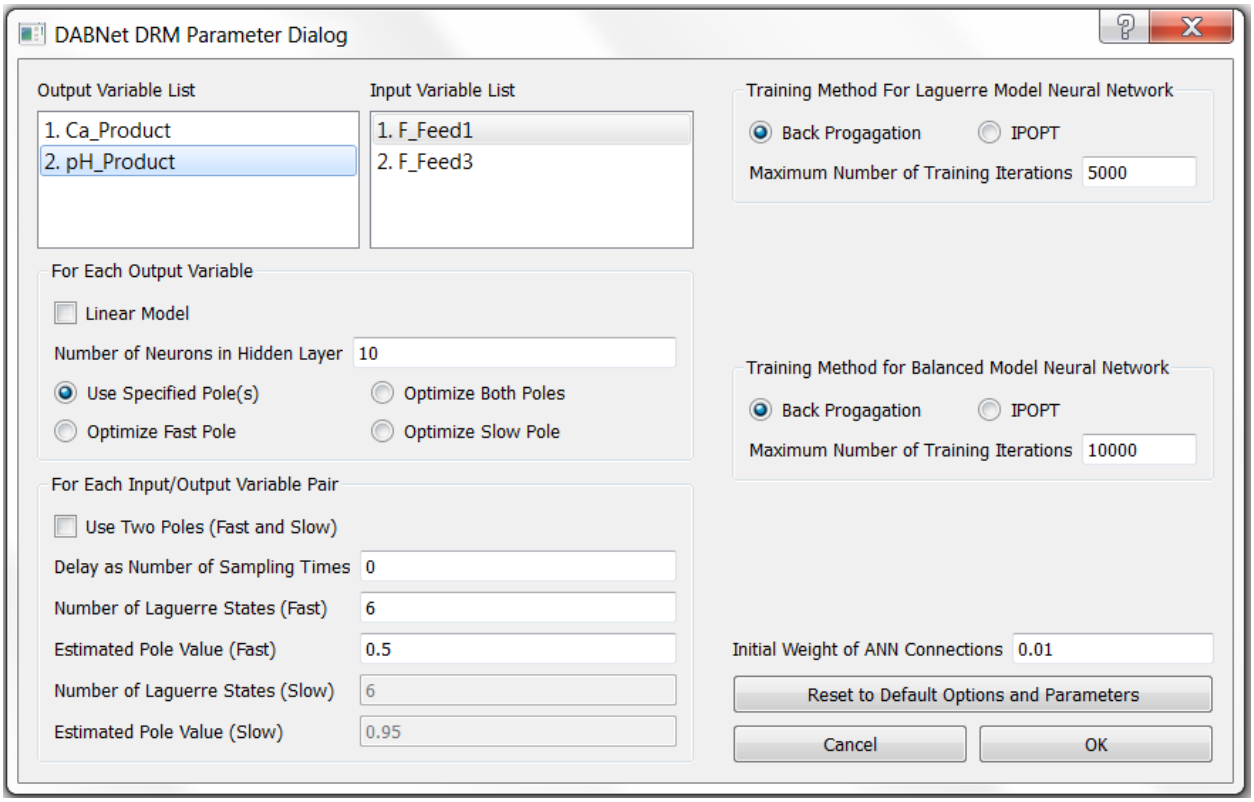

Figure 28: DABNet Parameter Dialog Window for pH Neut Model

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 console window of D-RM Builder to find that the relative error for training the balanced model of the first output variable is approximately 0.05475 while the error for the second variable is approximately 0.0017695. These errors are quite high since the default pole values of 0.5 are not good for the dynamic process. 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.

25. Save the case by issuing the **File→Save** command or clicking the toolbar icon  . 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.

26. 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.

27. 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 the toolbar icon  .

28. Issue the **Post-Process→Plot Training Responses** command or click the toolbar icon  to visualize the results. The "Result Plotting Dialog" window as shown in Figure 29 displays. 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 Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 30.

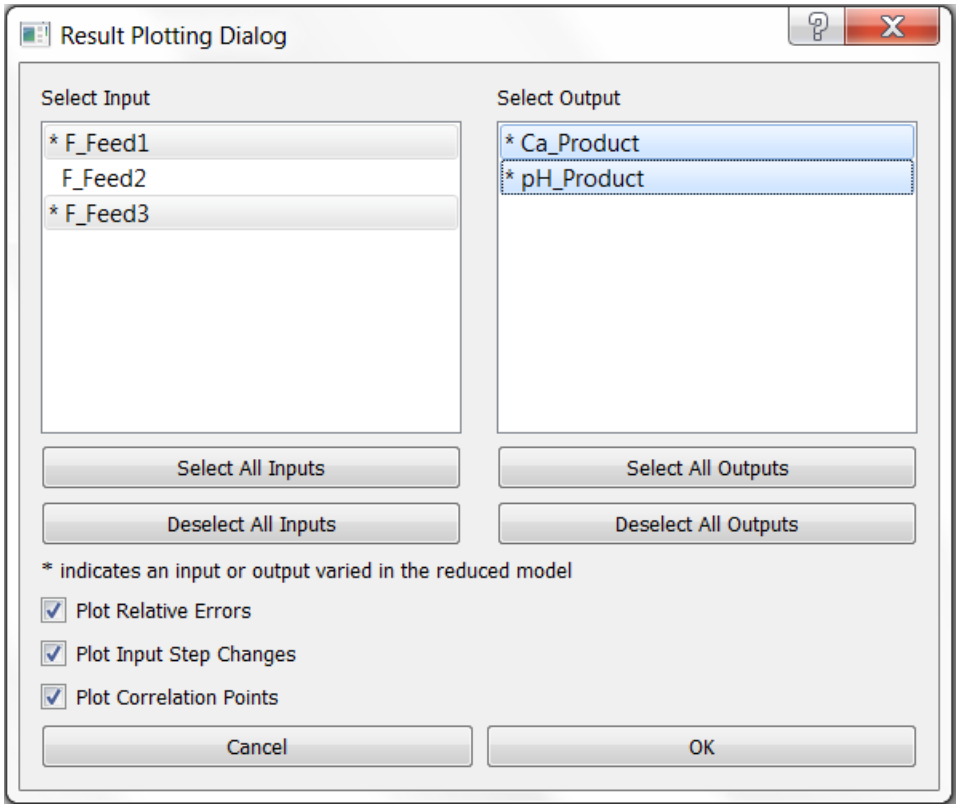

Figure 29: Result Plotting Dialog Window for pH Neut Model

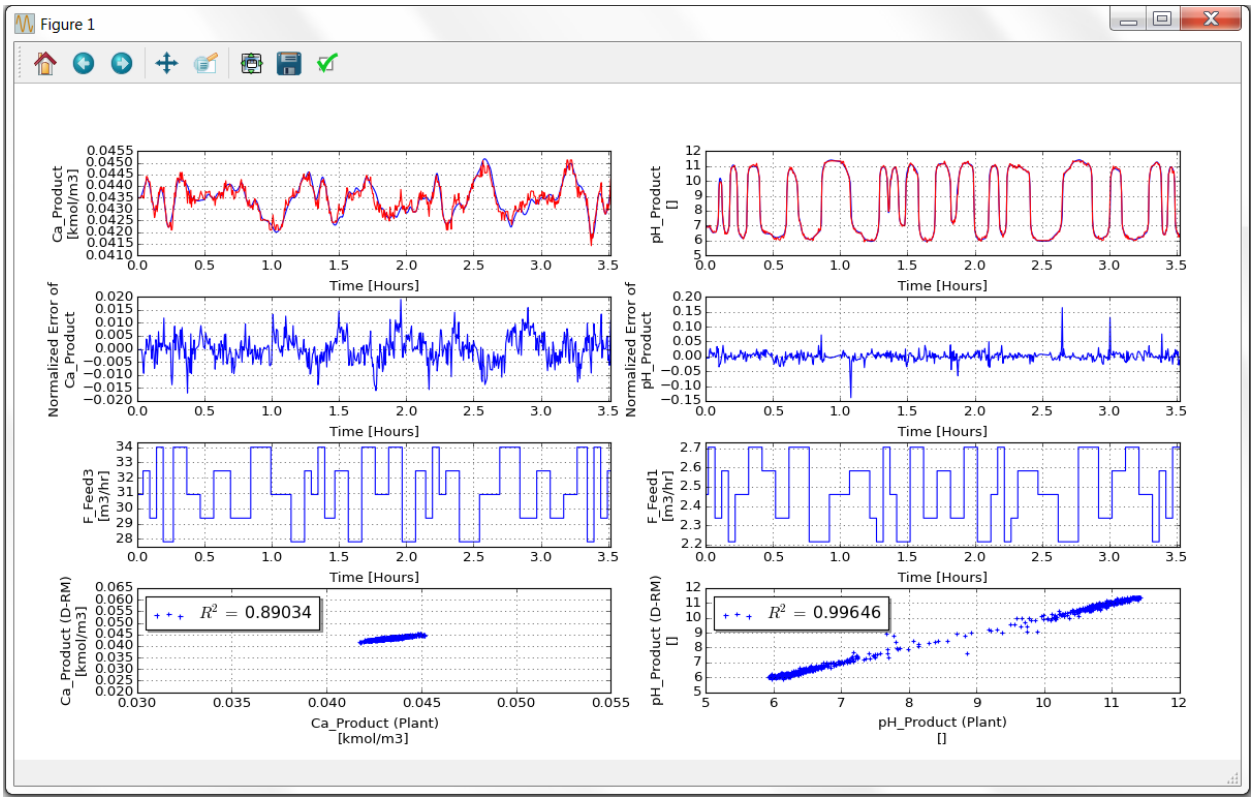

Figure 30: Plots of Training Data of pH Neut Model with Default DABNet Parameters

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 blue 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 first variable (Ca_Product) with a low R^2 value around 0.89. The normalized error plots also show relative errors over 15% in some regions. Close the plot. 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.

29. 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 the toolbar icon  .

30. Issue the **Post-Process→Plot Validation Responses** command or click the toolbar icon  . The "Result Plotting Dialog" window as shown in Figure 29 displays. Select the first and the third input variables and all of the output variables and then click "OK". A Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 31. 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" to close the plot window.

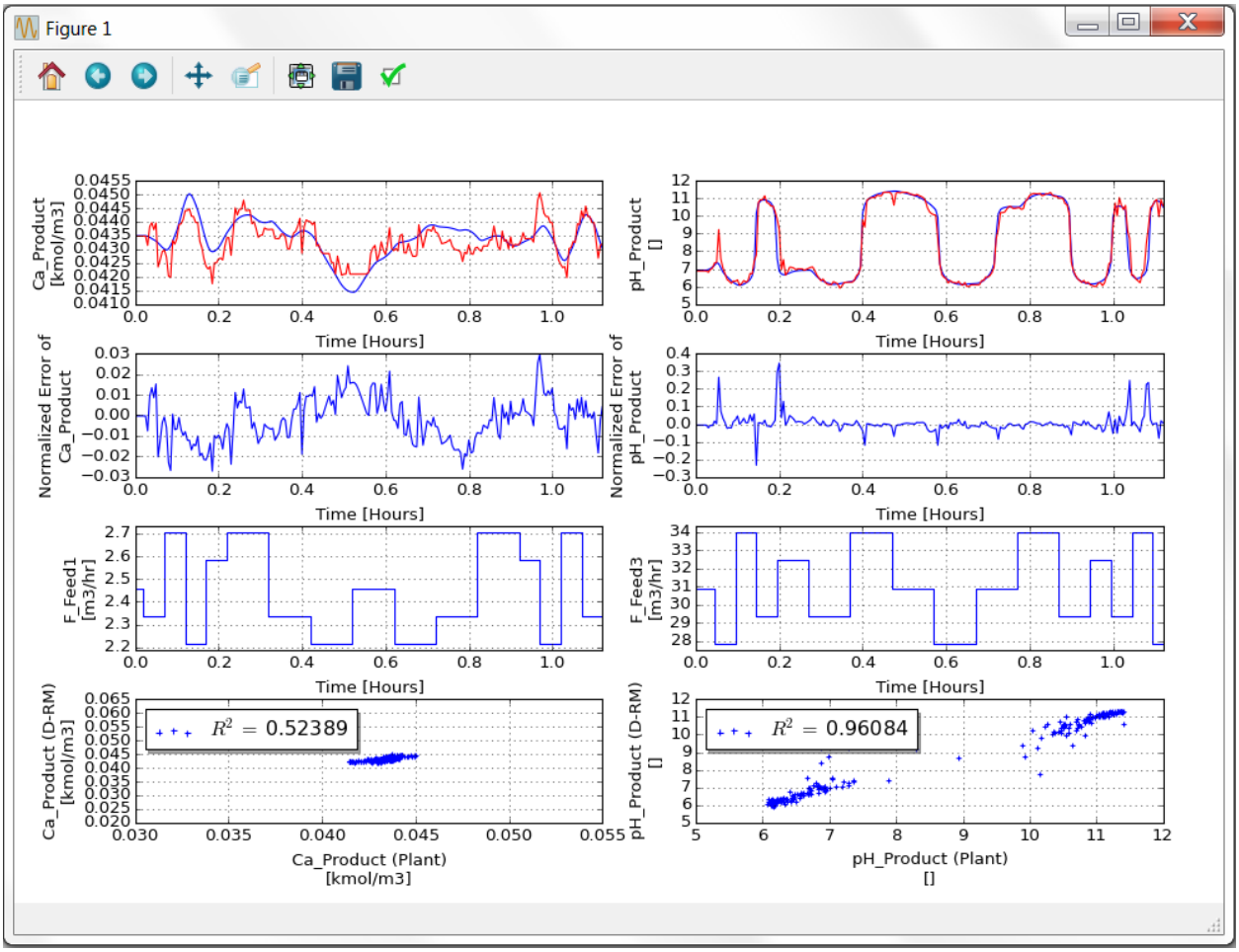

Figure 31: Plots of Validation Data of pH Neut Model with Default DABNet Parameters

31. 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 the toolbar icon  . 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 Fast Pole" option in the "For Each Output Variable" section. Meanwhile, enter 5 in the Number of Neurons in Hidden Layer text box. Click the second output variable and then click the "Optimize Fast Pole" option in the "For Each Output Variable" section. Enter 5 in the "Number of Neurons in Hidden Layer" text box. Select the first input variable in the "Input Variable List" list box and set the "Number of Laguerre States (Fast)" text box to 7. Repeat it for the second input variable. In the "Training Method for Laguerre Model Neural Network" section, enter 7000 in the "Maximum Number of Training Iterations" text box. Click "OK" 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. Text messages are displayed in D-RM Builder console window, including the optimized pole values and average training errors. For example,

the optimized pole values for the second output variable are 0.845556 and 0.864447 with respect to the first and second input variables. The optimized pole values can also be displayed by issuing the **Build→Generate Reduced Model** command again. In the "DABNet DRM Parameter Dialog" window, select an output variable and then an input variable in the two list boxes and the corresponding pole value displays in the "Estimated Pole Value (Fast)" text box. Click "Cancel" to exit the dialog window since the D-RM has been generated.

32. Repeat Steps 27–30 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 32 shows the comparison of the training data. Notice that the R^2 values are very close to 1, indicating a good fit of the DABNet model to the ACM model.

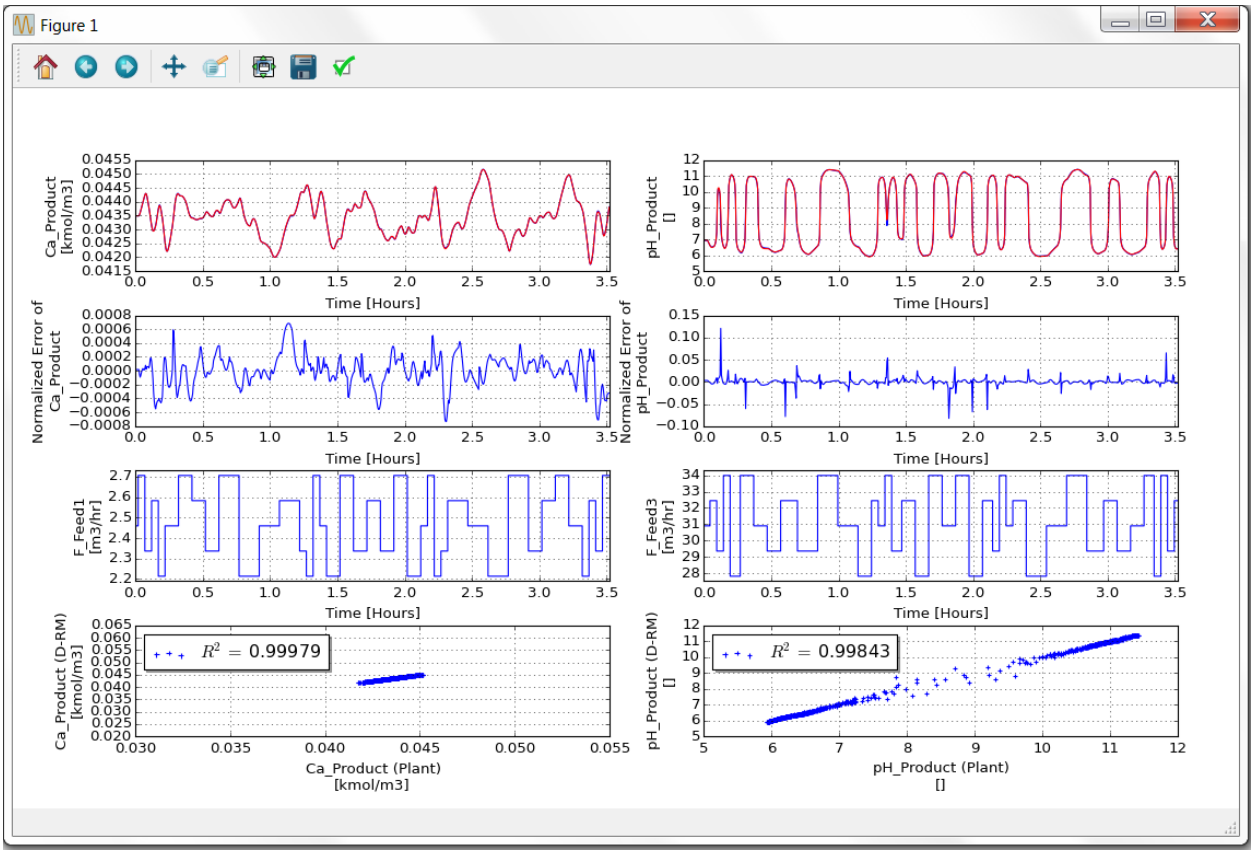

Figure 32: Plots of Training Data of pH Neut Model with Optimized Pole Values

Figure 33 shows the comparison of the validation data. Again the R^2 values for the validation data are much larger than those shown in Figure 31. Therefore, the accuracy of the D-RM has been improved substantially after the pole value optimization.

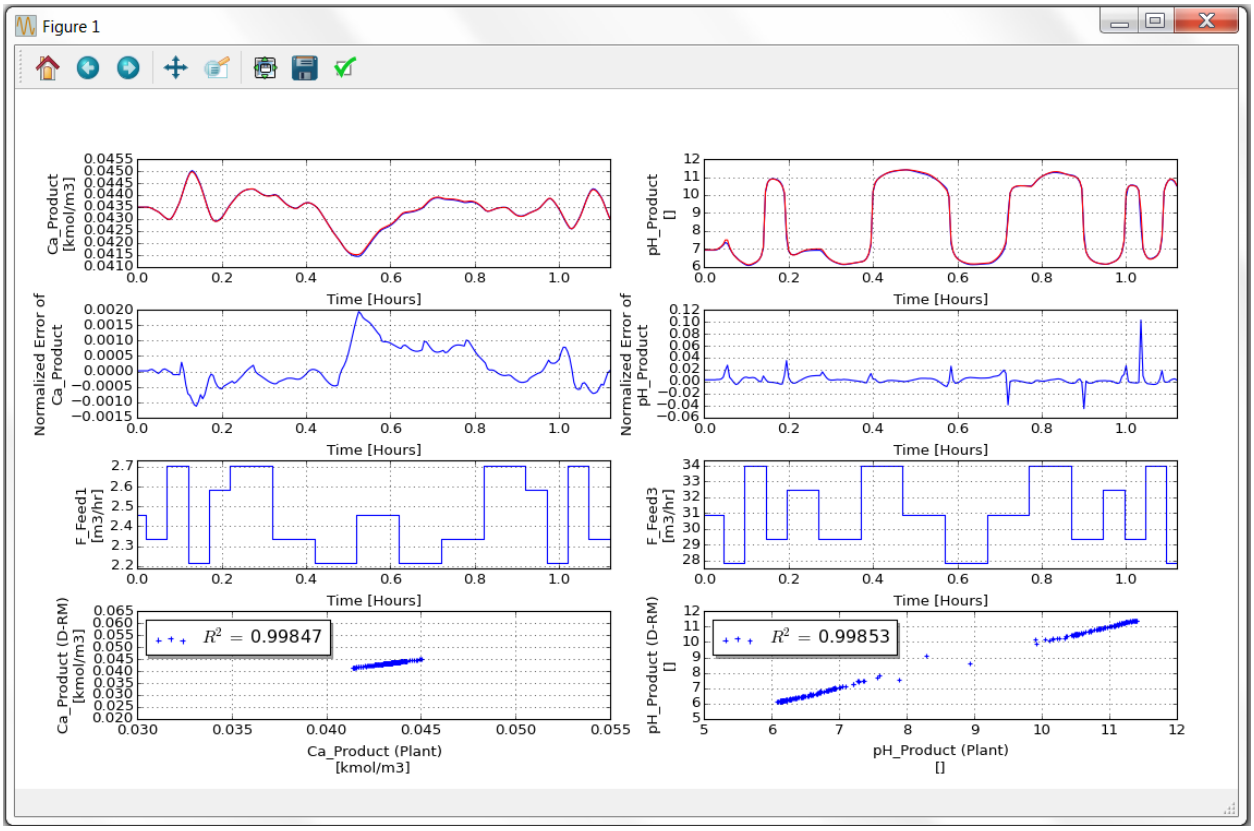

Figure 33: Plots of Validation Data of pH Neut Model with Optimized Pole Values.

33. Perform the UQ analysis. Specify the process and measurement noise levels by issuing the **UQ→Specify Noise** command or clicking the toolbar icon . The "Noise Specificaton Dialog" window as shown in Figure 34 displays.

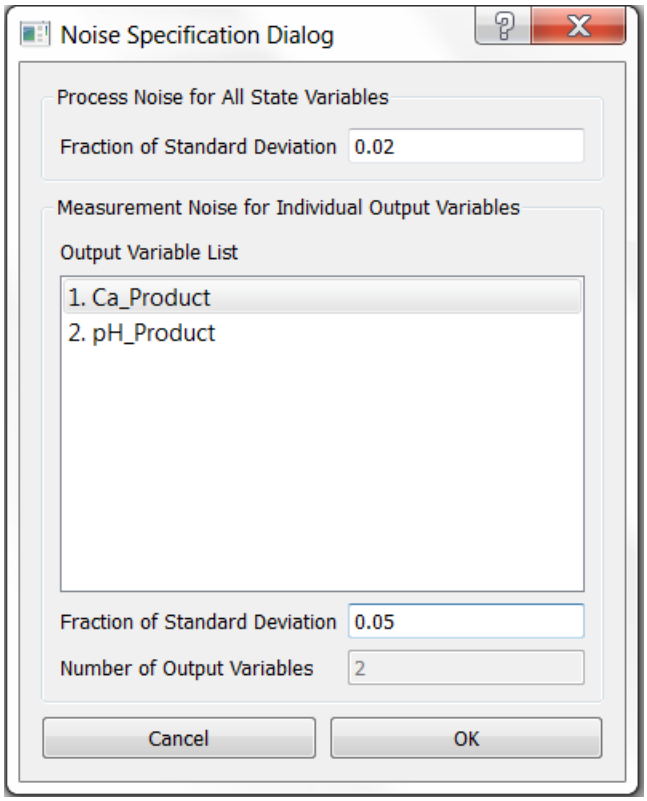

Figure 34: Dialog Window for Specifying the Noise of pH Neut Model

Set the "Fraction of Standard Deviation" in the "Process Noise for All State Variables" section to 0.02. In the section of "Measurement Noise for Individual Output Variables", click each output variable and set the Fraction of Standard Deviation to 0.05. 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 29. Select the two input variables with "*" before the variable names and the two output variables. 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 window as shown in Figure 35.

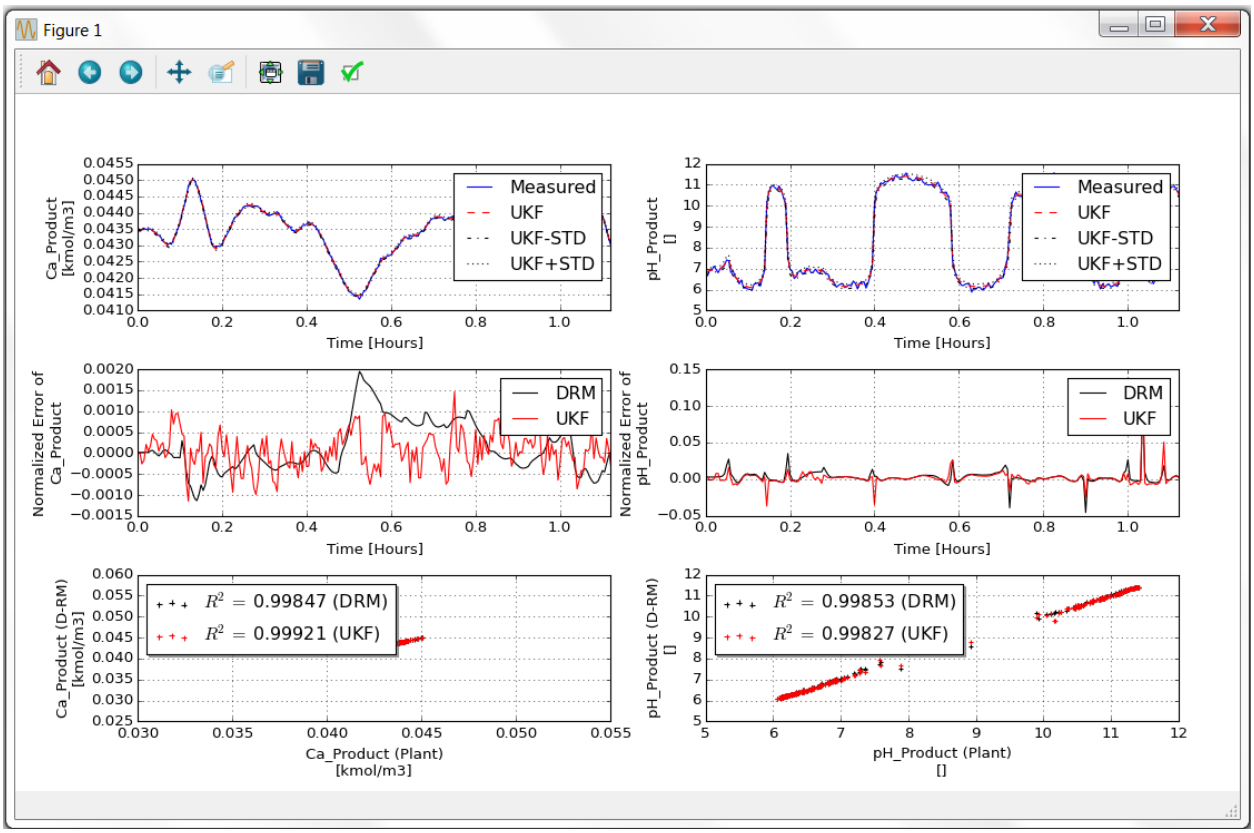

Figure 35: Plots of UQ Analysis for pH Neut Model

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

2.4. Building a D-RM Using DABNet Framework With Ramp Changes For Hard-to-Converge Stiff Models

Depending on the dynamic system to be modeled, a high-fidelity model containing tens of thousands of DAEs could be hard to solve 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.

To demonstrate the feature, an example is provided in a subfolder named “BFB” in the D-RM Builder’s “examples” folder. This example models the sorbent-based two-bed bubbling fluidized bed (BFB) CO_2 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_2 mole fraction in the inlet flue gas could also cause the DAE solver to fail. Therefore, we have to 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 ACM file of the BFB model “BFB.acmf” in the “examples\BFB” folder of the D-RM Builder’s installation directory. Confirm the user of the D-RM Builder has write-permissions to this folder.

2. Launch SinterConfigGUI. The GUI window as shown in Figure 1 displays.

3. Click the “Browse” button and browse to the “BFB.acmf” file in the working directory. Then click the “Open File and Configure Variables” button. It takes a few seconds for the SinterConfigGUI to display a “SinterGonfigGUI Simulation Meta-Data” window as shown in Figure 36. An ACM window with BFB model inside also displays as shown in Figure 37.

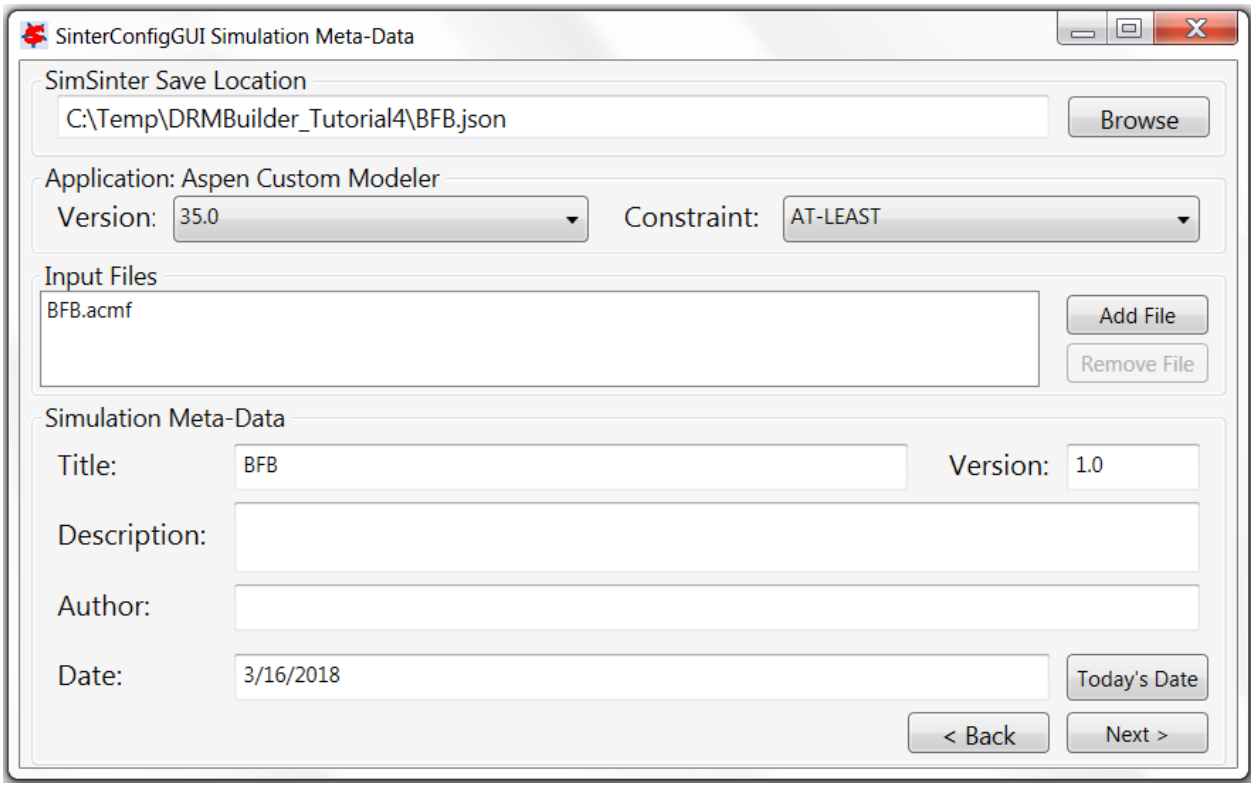

Figure 36: Dialog Window for the Simulation Meta-Data of BFB Model

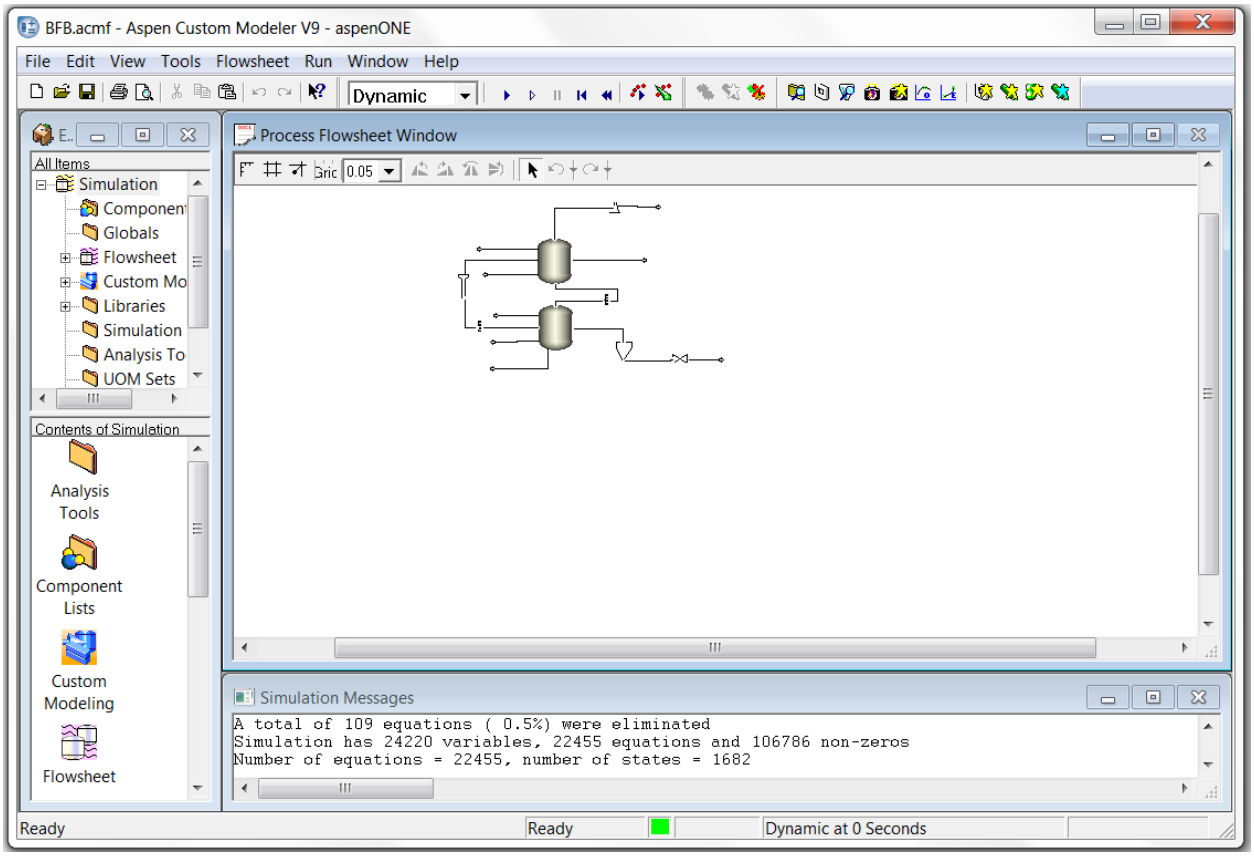

Figure 37: ACM Window for BFB Model Displayed by SinterConfigGUI

4. Enter “BFB Example” in the “Title” text box, “My fourth DRMBuilder tutorial” in the “Description” text box, and the user’s name in the “Author” text box. Then click the “Next” button. A “SinterConfigGUI Variable Configuration Page” window similar to Figure 4 displays.

5. Enter “ADSB.~” in the “Variable Search Pattern” text box and click the “Search” button. All the variables related to ADSB block are listed in the list box at the lower left corner of the window as shown in Figure 38.

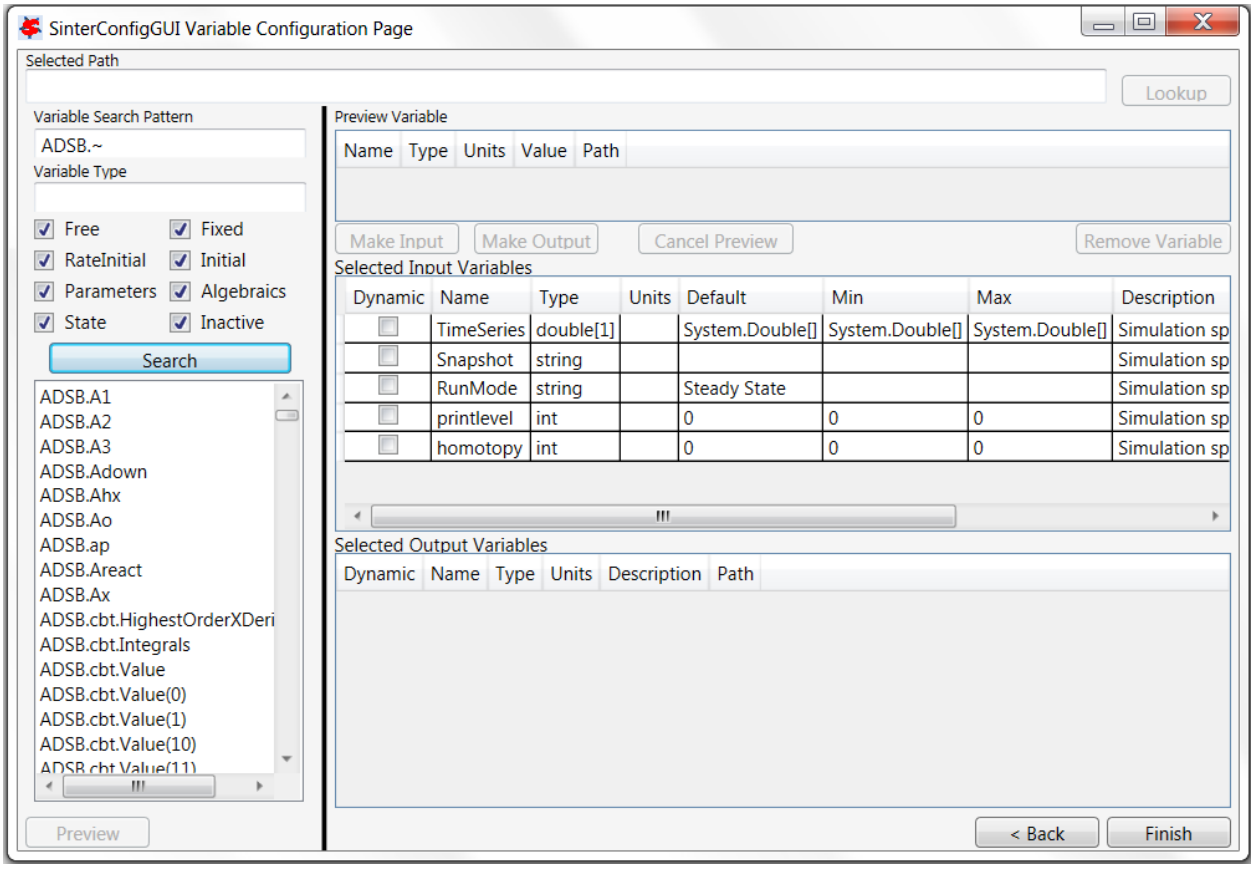

Figure 38: Variable List on Configuration Page of BFB Model

6. Select the “ADSB.GasIn.F” in the list box and click the “Lookup” button at the upper right corner. The variable is displayed in the “Preview Variable” section. Click the “Make Input” button and the variable is appended to the end of the “Selected Input Variables” table. Change the name of the variable to “F_GasIn” and select the checkbox in the “Dynamic” column. This makes the selected variable a dynamic input variable.

7. Select the “ADSB.GasIn.T” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “T_GasIn”.

8. Enter “zco2” in the “Variable Search Pattern” text box and click the “Search” button. Select the “ZCO2” in the list box and repeat Step 6 to make it a dynamic input variable and change the name to “Z_CO2”.

9. Enter “ADSA.~” in the “Variable Search Pattern” text box and click the “Search” button. All the variables related to ADSA block are listed in the list box at the lower left corner of the window. Find “ADSA.SorbIn.Fm” in the list and double-click the item. The variable is shown in the “Preview Variable” table. Click the “Make Input” button and the variable is added to the “Selected Input Variables” table. Make it a dynamic input variable and change the name to “Fm_SorbentIn”.

10. Change the default value of the input variable named “RunMode” in the “Selected Input Variables” table from “Steady State” to “Dynamic”.

11. Enter “co2removal” in the “Variable Search Pattern” text box and click the Search button. Then click the “Make Output” button and the variable is added to the “Selected Output Variables” table. Change the name of the variable to “CO2_Removal” and select the checkbox in the “Dynamic” column. This makes the selected variable a dynamic output variable.

12. Enter “STACK.~” in the “Variable Search Pattern” text box and click the “Search” button. Select the “STACK.Out2.F” in the list box and repeat Step 10 to make it a dynamic output variable and change the name to “F_GasOut”.

13. Select the “STACK.Out2.T” in the list box and repeat Step 10 to make it a dynamic output variable and change the name to “T_GasOut”.

14. Select the “STACK.Out2.Z(”CO2”)” in the list box and repeat Step 10 to make it a dynamic output variable and change the name to “CO2_GasOut”.

15. This concludes the configuration of ACM variables. Figure 39 shows the SinterConfigGUI window after the configuration. Click the “Next” button at the lower right corner and a “SinterConfigGUI Vector Default Initialization” dialog window display as shown in Figure 7. Leave the default time series data unchanged.

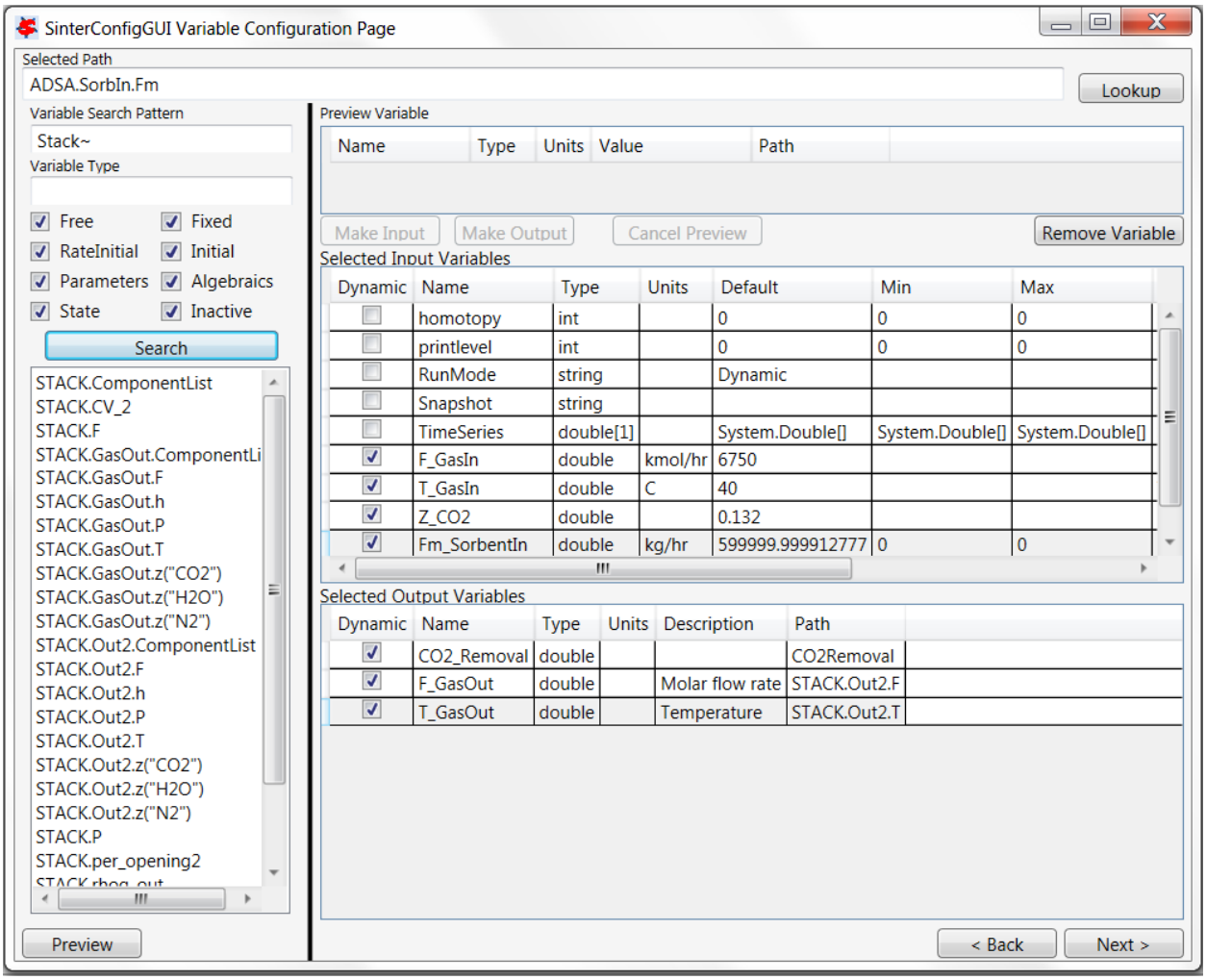

Figure 39: Configured Dynamic Variables of BFB Model by SinterConfigGUI

16. Click the “Finish” button to exit the “SinterConfigGUI”. A file named “BFB.json” is created in the working directory. Close the ACM window.

17. After the dynamic input and output variables are configured and the JSON file is created, start D-RM Builder from Windows “Start” button or shortcut. The main window of D-RM Builder with a blank project displays as shown in Figure 10.

18. To start the D-RM building process, a high-fidelity model (the BFB ACM model in this case) needs to be selected. Issue **Setup→Choose High-Fidelity Model** command or click the toolbar icon . A “SimSinter Configuration File” window displays for file browsing and selecting. Browse and select the “BFB.json” file. This loads the configuration file into the D-RM Builder. A text message confirming the file selection and loading displays in the client area of the D-RM Builder window.

19. Configure the input variables. Navigate to **Setup→Configure Input Variables** or click the toolbar icon  . The "Input Variable Dialog" window displays as shown in Figure 40.

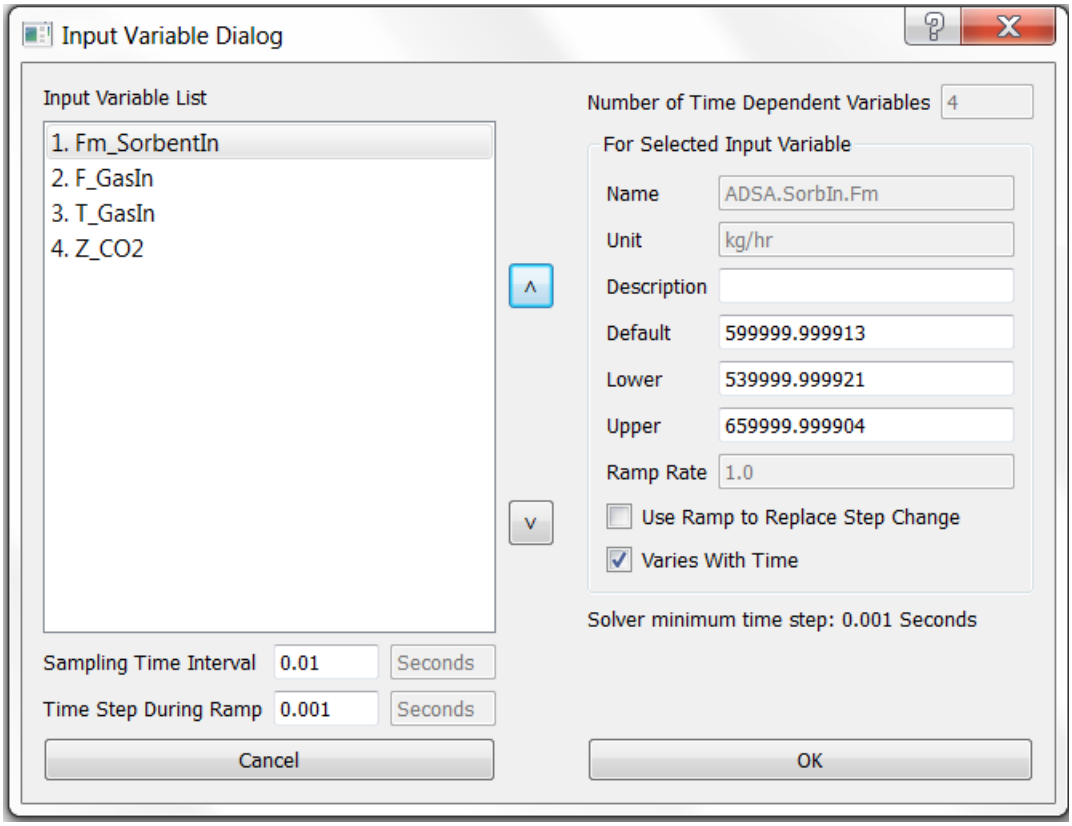

Figure 40: Input Variable Dialog Window for BFB Model

For this example, the first input variable "Fm_SorbentIn" is kept as constant while the other three input variables are varied with time. Click the first variable in the "Input Variable List" list box and clear the "Varies With Time" check box. Change the "Sampling Time Interval" from the 0.01 second to 0.2 second. In the "Time Step During Ramp" text box, enter 0.005. Next, enable the "Use Ramp To Replace Step Change" option for the other three input variables in the "Input Variable List" list box. Click the second item "F_GasIn" in the list box and then select the "Use Ramp To Replace Step Change" check box. The "Ramp Rate" text box is enabled. Enter 13501 in the "Ramp Rate" text box. These are the user inputs for the flow rate of the feed flue gas stream. Click the third item "T_GasIn" 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 fourth item "Z_CO2" 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_2 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. Click “OK” to accept the modifications and exit the window.

20. Configure the output variables by issuing the **Setup→Configure Output Variables** command or clicking the toolbar icon  . For this example, build the D-RM for all three output variables. Click “OK” to accept the default inputs.

21. Prepare the training sequence, by selecting the **Setup→Prepare Training Sequence** command or clicking the toolbar icon  . The “Step Change 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.

22. Prepare an input change sequence for validation by issuing the **Setup→Prepare Validation Sequence** command or clicking the toolbar icon  . The “Step Change 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.

23. Save the file by issuing the **File→Save** command. The “Save As” dialog window displays. Enter “case4_BFB” in the “File name” text box and then click “OK”.

24. Launch the ACM high-fidelity model simulation for training by issuing the **Build→Perform Training Simulation** command or clicking the toolbar icon  . This creates temporary files and a folder in the current working directory and displays an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACMs “Simulation Message” window is enabled. The mouse icon is switched from normal (an arrow) to busy (rotating circle). Wait for at least one hour without closing the ACM window. After the high-fidelity model simulation is completed and the ACM window is closed, a message displays, confirming the successfully completion of the high-fidelity model simulation. The mouse icon becomes a normal arrow icon.

25. Launch the ACM high-fidelity model simulation for validation by issuing the **Build→Perform Validation Simulation** command or clicking the toolbar icon  . Wait until the ACM simulation is completed and the message confirming the successful completion of the simulation displays.

26. 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 16.

27. Build the DABNet model by issuing the **Build→Generate Reduced Model** command or clicking the toolbar icon  . The "DABNet DRM Parameter Dialog" window displays. Enable the pole value optimization option for each output variable. Select an output variable in the "Output Variable List" box and then click either the "Optimize Both Poles" or the "Optimize Fast Pole" button in the "For Each Output Variable" section. Repeat this for all the output variables. Click "OK" in the window to accept all of the modifications. The DRM 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 three output variables. The mouse icon changes to the busy (rotating circle) icon until the model generation process is completed.

28. Save the case by issuing the **File→Save** command or clicking the toolbar icon  . 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.

29. 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.

30. 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.

31. Issue the **Post-Process→Plot Training Responses** command. The "Result Plotting Dialog" window as shown in Figure 41 displays.

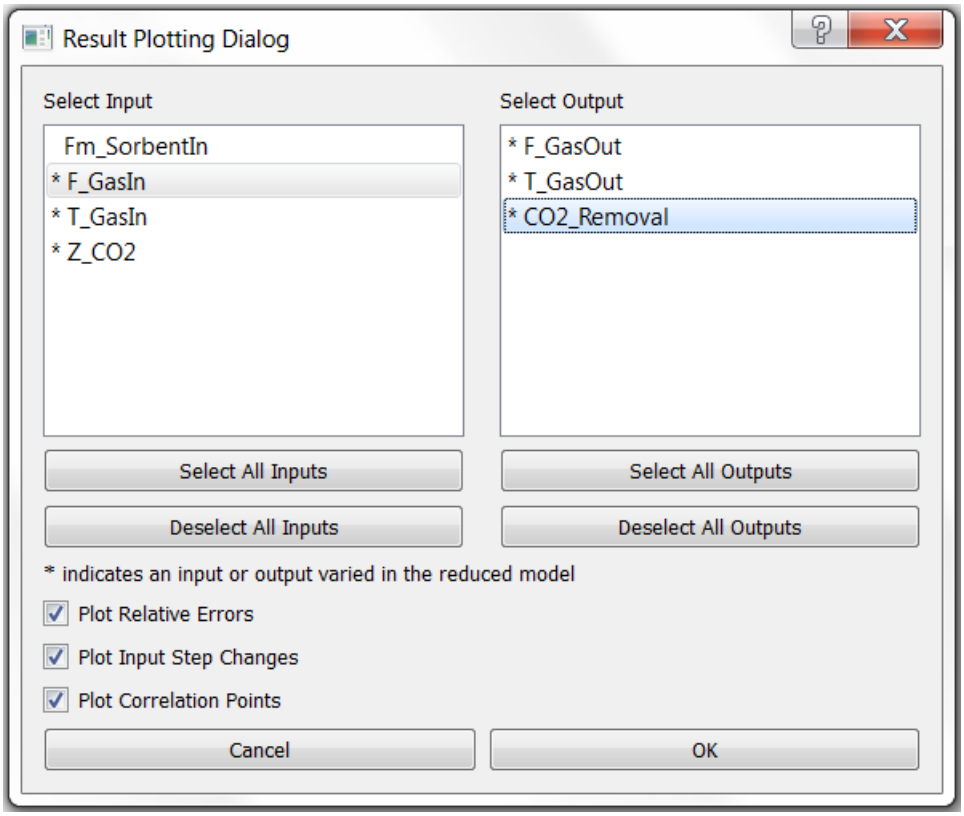

Figure 41: Dialog Window for Plotting Training Data of BFB Model

Select the input variable “F_GasIn” in the “Select Input Variables” list, select the output variable “CO2_Removal” in the “Select Output Variables” list, and then click “OK”. A plot window as show in Figure 42 displays.

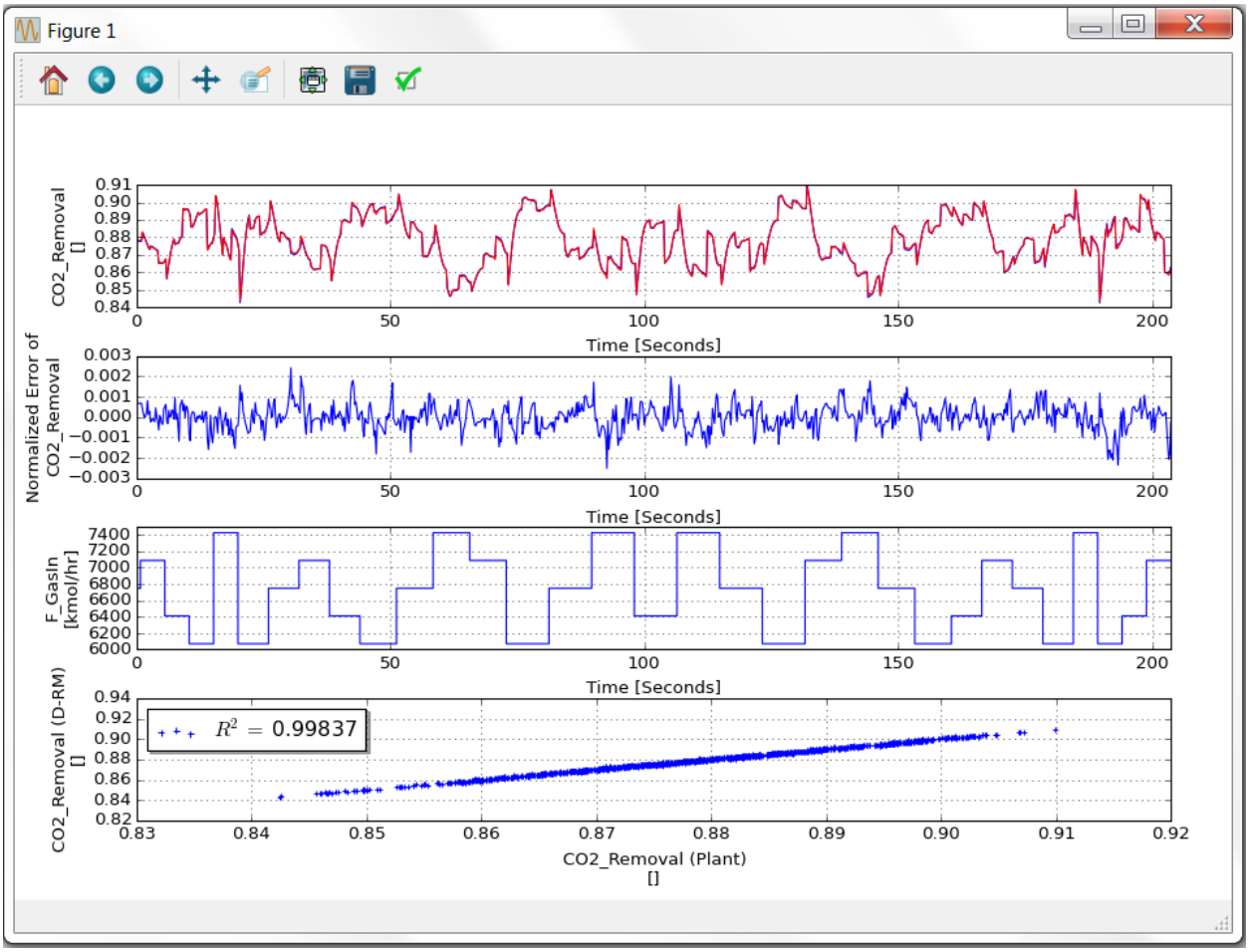

Figure 42: Response of Training Data of BFB Model

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 blue is the high-fidelity model (ACM model) simulation result and the other in red is the D-RM prediction. Close the plot window after examining the results for the training sequence. If needed, issue the **Post-Process→Plot Training Responses** command again to view the plot for other output variables.

32. 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 the toolbar icon .

33. Issue the **Post-Process→Plot Validation Responses** command or click on the toolbar . The "Result Plotting Dialog" window as shown in Figure 41 displays. Select one or a few input variables and one or a few output variables and then click "OK". A plot window displays. Close the plot window after examining the plots.

34. 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. Finally, close the main window of D-RM Builder.

2.5. Building a D-RM Using DABNet for Two-Time-Scale Process

The previous tutorial demonstrates the ability of DABNet D-RM to model the fast dynamics of CCSI's BFB adsorber-reactor since the sampling time interval of 0.2 second is short and so is the duration between two step changes. With short durations, the process is always in transient mode without reaching steady-state conditions. As a matter of fact, the BFB adsorber-reactor is a system that has two time scales, a fast one due to fast fluid dynamics and a slow one due to the reactivity of solid sorbent. This tutorial demonstrates how to model a two-time-scale process using double-pole DABNet D-RM.

This tutorial uses the same BFB adsorber-reactor ACM model in the previous tutorial and the same JSON file configured by SinterConfigGUI. The following are the steps to build the D-RM.

1. Copy the "BFB.acmf" file and the "BFB.json" file created in the previous tutorial in a new working directory. Start D-RM Builder and issue **Setup→Choose High-Fidelity Model** command to select the "BFB.json" file.

2. Configure the input variables. Navigate to **Setup→Configure Input Variables** or click the toolbar icon  . The "Input Variable Dialog" window as shown in Figure 40 displays.

3. With the first input variable "Fm_SorbentIn" selected in the list box, click the "Down" arrow button. The variable becomes the second item in the list box.

4. In this tutorial, the feed flue gas temperature and the mole fraction in the flue gas are kept constant. Select the "T_GasIn" in the list box and make sure the "Varies With Time" check box is cleared. Repeat for the "Z_CO2" input variable. Since flue gas temperature and mole fraction are kept constant, the "Use Ramp to Replace Step Change" check box is cleared for the first two input variables that vary with time. Enter 1 in the "Sampling Time Interval" text box. Accept all other default inputs and click "OK".

5. Configure the output variables by issuing the **Setup→Configure Output Variables** command or clicking the toolbar icon  . For this example, build the D-RM for "CO2_Removal" only. Select "F_GasOut" and clear the "Varies With Time" check box. Repeat for the output variable "T_GasOut". Click "OK" to accept the changes.

6. Prepare the training sequence, by selecting the **Setup→Prepare Training Sequence** command or clicking the toolbar icon  . The "Step Change 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 display in the "For Each LHS Set" list box. Select the first row "LHS Set 1" item in the list box and then enter 20 in the "Duration of Step Change" text box. Select the second row "LHS Set 2" item in the list box and then enter 40 in the "Duration of Step Change" text box. Select the third row "LHS Set 3" item in the list box and then enter 60 in the "Duration of Step Change" text box. Select the fourth row "LHS Set 4" item in the list box and then enter 80 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.

7. Prepare an input change sequence for validation by issuing the **Setup→Prepare Validation Sequence** command or clicking the toolbar icon  . The "Step Change Sequence Dialog" window displays. Enter 5 in the "Number of LHS Points" text box and then enter 2 in the "Number of LHS Sets" text box. Select the first row "LHS Set 1" item in the list box and then enter 20 in the "Duration of Step Change" text box. Select the second row "LHS Set 2" item in the list box and then enter 40 in the "Duration of Step Change" text box. Click "OK" to accept the change and then close the window.

8. Save the file by issuing the **File→Save** command. The "Save As" dialog window displays. Enter "case5_BFB" in the File name text box and then click "OK".

9. Launch the ACM high-fidelity model simulation for training by issuing the **Build→Perform Training Simulation** command or clicking the toolbar icon  . This creates temporary files and a folder in the current working directory and displays an ACM window. The ACM simulation starts immediately. The ACM iteration message can be viewed inside the ACM window if the ACMs "Simulation Message" window is enabled. The mouse icon is switched from normal (an arrow) to busy (rotating circle). Wait for at least one hour without closing the ACM window. After the high-fidelity model simulation is completed and the ACM window is closed, a message displays, confirming the successfully completion of the high-fidelity model simulation. The mouse icon becomes a normal arrow icon.

10. Launch the ACM high-fidelity model simulation for validation by issuing the **Build→Perform Validation Simulation** command or clicking the toolbar icon  . Wait until the ACM simulation is completed and the message confirming the successful completion of the simulation displays.

11. 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 16.

12. Build the DABNet model by issuing the **Build→Generate Reduced Model** command or clicking the toolbar icon  . The "DABNet DRM Parameter Dialog" window displays. Select the only output variable "CO2_Removal" in the "Output Variable List" box. Enter 2 in the "Number of Neurons in Hidden Layer" text box. Click the "Optimize Both Poles" button in the "For Each Output Variable" section. Select the "Use Tow Poles (Fast and Slow)" check box for each input variable. Leave the default value of 0.5 for the fast pole unchanged. Enter 0.99 for the slow pole for each input variable as shown in Figure 43. Click "OK" 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. The mouse icon changes to the busy (rotating circle) icon until the model generation process is completed.

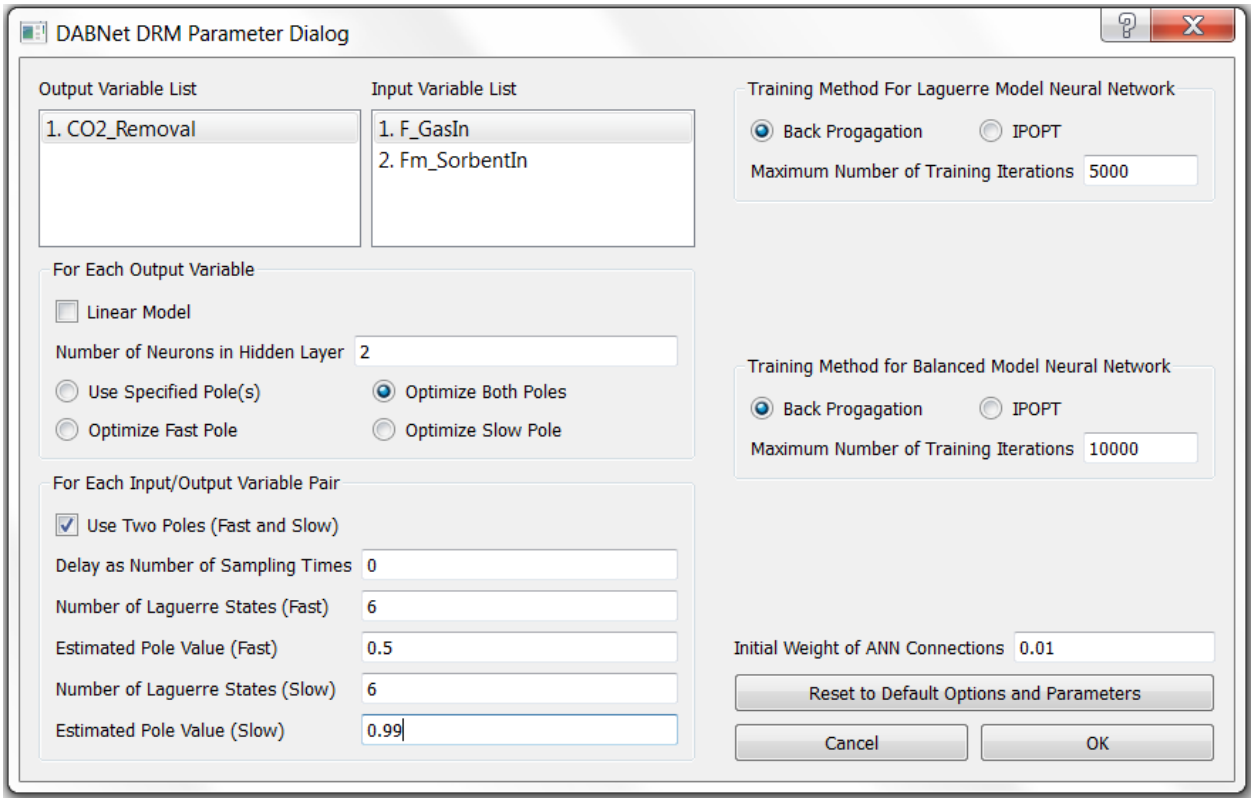

Figure 43: DABNet Parameter Dialog Window for Two-Time-Scale BFB Model

13. Save the case by issuing the **File→Save** command or clicking the toolbar icon  . 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.

14. 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.

15. 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.

16. Issue the **Post-Process→Plot Training Responses** command. The "Result Plotting Dialog" window as shown in Figure 44 displays.

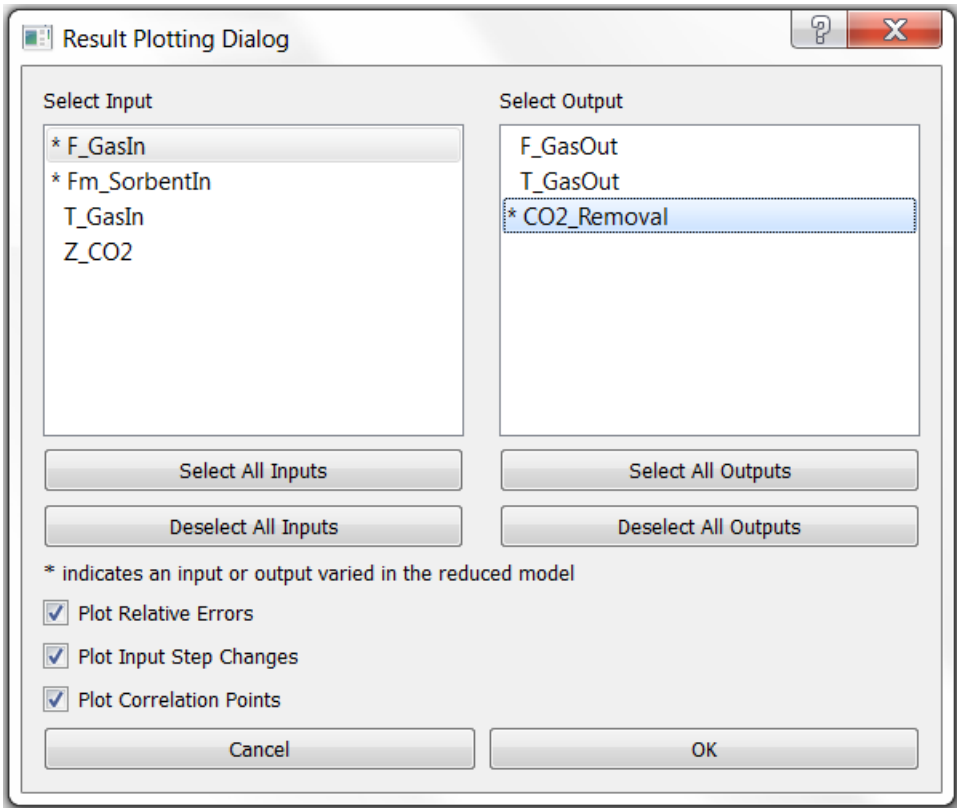

Figure 44: Dialog Window for Plotting Training Data of Two-Time-Scale BFB Model

Select the input variable "F_GasIn" in the "Select Input Variables" list, select the output variable "CO2_Removal" in the "Select Output Variables" list, and then click "OK". A plot window as show in Figure 45 displays.

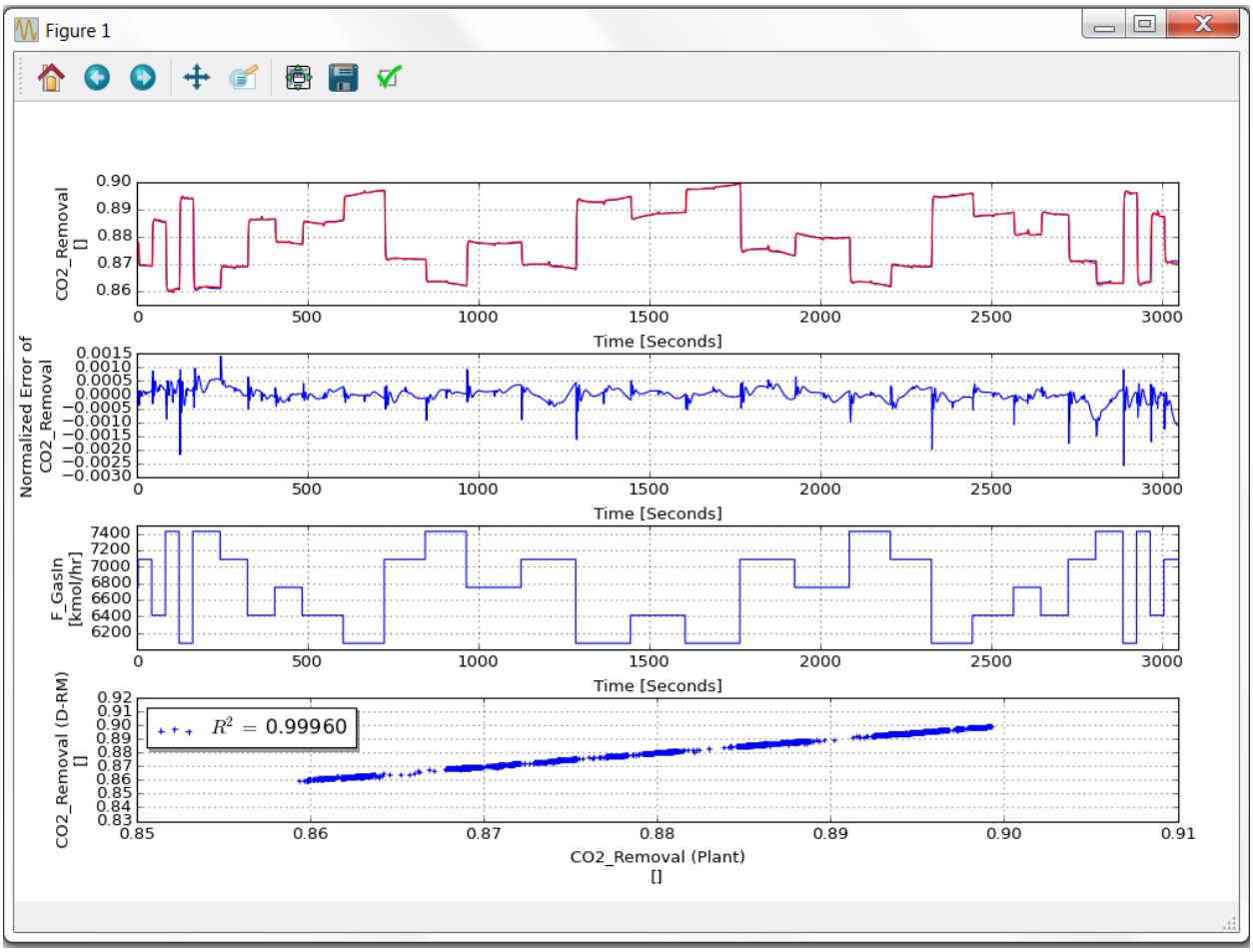

Figure 45: Response of Training Data of Two-Time-Scale BFB Model

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 blue is the high-fidelity model (ACM model) simulation result and the other in red is the D-RM prediction. It can be seen from the figure that there is a quick response in a few seconds followed by a slow response whenever there is an input step change. Close the plot window after examining the results for the training sequence. If needed, issue the **Post-Process→Plot Training Responses** command again to view the plot for other output variables.

17. 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 the toolbar icon .

18. Issue the **Post-Process→Plot Validation Responses** command or click on the toolbar . The "Result Plotting Dialog" window as shown in Figure 44 displays. Select one or a few input

variables and one or a few output variables and then click “OK”. A plot window displays. Close the plot window after examining the plots.

19. 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. Finally, close the main window of D-RM Builder.

2.6. Building a D-RM Using NARMA Framework

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 [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. The Van-de-Vusse reactor model created in the first tutorial of this manual is used as the high-fidelity model for this tutorial. Since the high-fidelity model simulations have been performed in the first tutorial, those results saved in the case file “case1_VdV.json” can be used to build the NARMA model. The following are the steps to generate the NARMA-based D-RM.

1. Open Windows Explorer and browse to the working directory for the tutorial for Section 2.1 and copy the “case1_VdV.drmb” file and rename it as “case5_VdV.drmb”. This tutorial uses the case file “case5_VdV.drmb” as the starting point and revise the D-RM training options. Note: If the “case5_VdV.drmb” is copied to a different working directory, the “VdV_Reactor.json” and “VdV_Reactor.acmf” files also need to be copied.

2. Start DRM-Builder through Windows “Start” button or the shortcut icon for D-RM Builder.

3. Issue the **File→Open** command or click the toolbar icon  to display the “Open D-RM Builder File” dialog. Browse to the working directory, select the “case5_VdV.drmb” file, and click the “Open” button.

4. Select the **Build→D-RM Model Type→NARMA** command. This switches the D-RM model type from DABNet to NARMA.

5. Select the **Build→Generate Reduced Model** command or click the toolbar icon  . The "NARMA DRM Parameter Dialog" window as shown in Figure 46 displays.

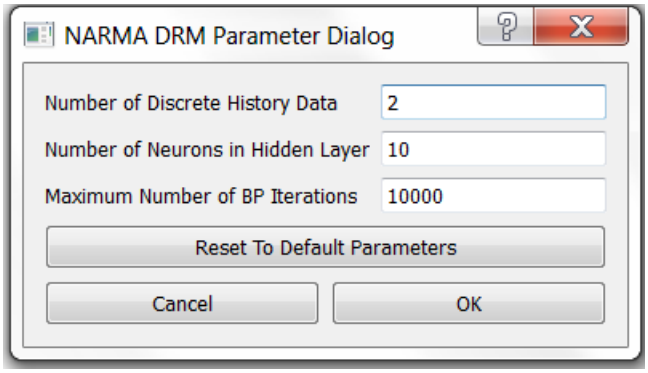

Figure 46: Dialog Window for Building a NARMA Model

Notice that the dialog 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 (rotating circle) icon until the training process is completed.

6. 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 Builder 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.

7. 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 47 shows the plots of the validation data.

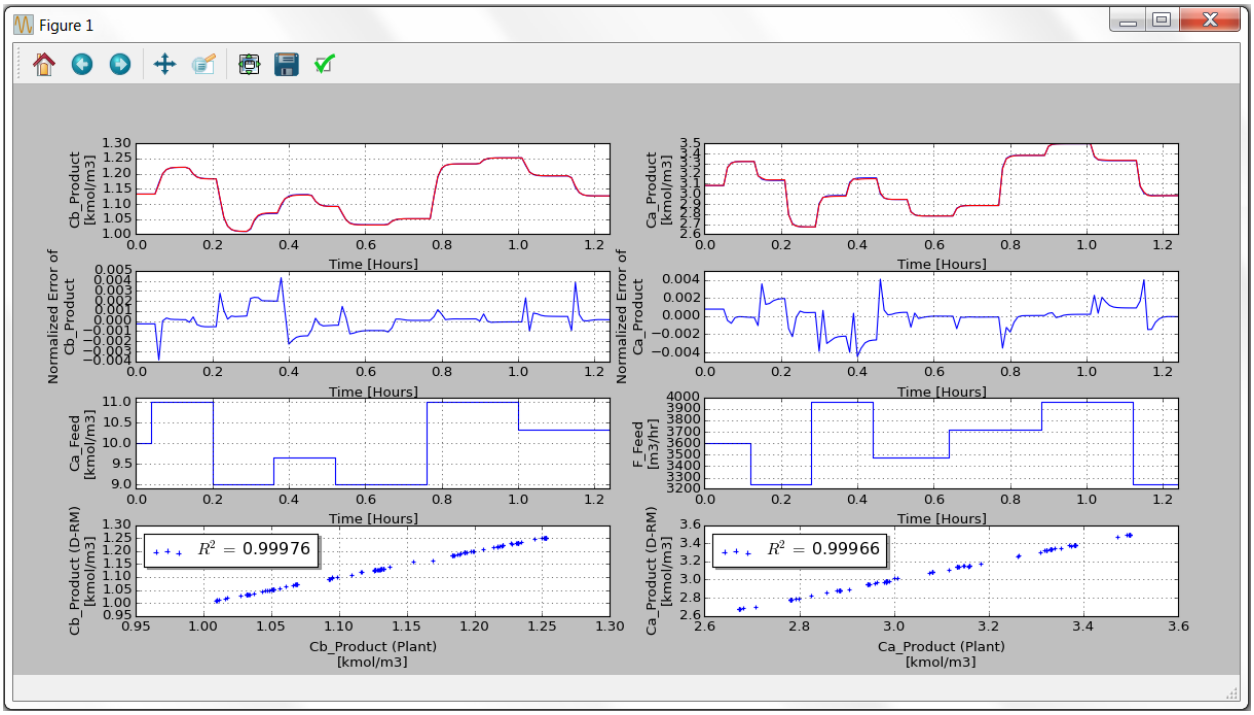

Figure 47: Predictions of Validation Sequence Using High-Fidelity Model History Data

Notice that when the "true" 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.

8. 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.

9. Repeat the commands in Step 7 by clicking the icon  followed by the icon  on the toolbar. Figure 48 shows the plots of the validation sequence.

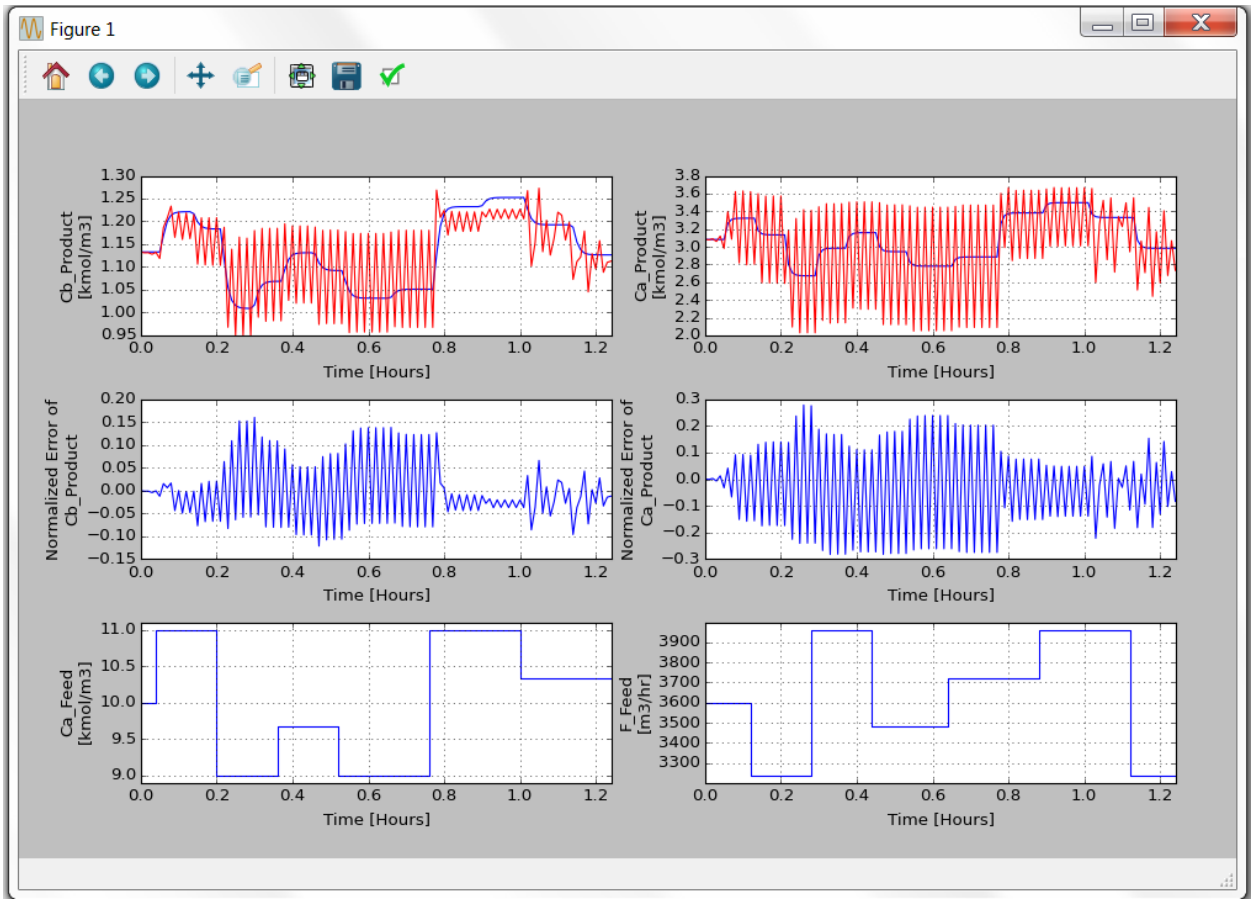

Figure 48: Predictions for Validation Sequence Using D-RM Predicted History Data

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.

10. 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. Building a D-RM Using Custom Transient Data

The tutorials in previous sections demonstrate the generation of data-driven D-RMs from high-fidelity models. The dynamic simulations of the high-fidelity models are performed to provide the dynamic responses of output variables when input variables are changed. Currently only ACM dynamic model is supported as the high-fidelity model by D-RM Builder to simulate the dynamic process. If a user has the dynamic data available either from experiments on an existing plant or from the simulations of other high-fidelity models, D-RM Builder is able to use the user provided custom data to generate a D-RM. The user needs to provide the transient data in comma-separated values (csv) files. This section gives a tutorial to use the user-specified data to generate a D-RM through D-RM Builder.

1. Copy two files named "VdV_Train_Plant.csv" and "VdV_Valid_Plant.csv" from the "examples\custom" folder under D-RM Builder's installation directory and paste them to a working directory of your choice. Conform that the working directory is allowed to read and write files. Open the "VdV_Train_Plant.csv" file by Microsoft Excel. This file contains the user specified data corresponding to the Van-de-Vusse reactor described in previous sections. The format of the data are described below.

Line 1 has two columns. The first one is the sampling time interval, which is the discretization size in time for the D-RM to be generated. If the data are from an existing plant through its data collection system, please make sure the sampling frequency matches the required sampling time interval for the D-RM to be created. The second column contains the unit of time.

Line 2 has two columns. The first column contains the number of input variables and the second column contains the number of output variables.

Line 3 contains the names of input variables followed by the names of output variables. Blank spaces in the name strings are not allowed. The number of columns should be the same as the total number of input and output variables.

Line 4 contains the internal high-fidelity model variable names or tag names. They are used for display only. The entries in Line 4 could be the same as those in Line 3.

Line 5 contains the descriptions of those input and output variables. Again, blank spaces are not allowed in the description.

Line 6 contains the units of the input and output variables. A blank space should be used for a dimensionless variable.

Line 7 contains the number of transient data sets included in the "csv" file. Note that each set of transient data should start from a steady-state condition. In this example, 2 sets of data are tabulated.

Lines 8 contains the number of data points (lines) for the first set of transient data.

Lines 9 contains the number of data points (lines) for the second set of transient data.

Note: Since the example contains only two sets of transient data, only two lines are needed to list the numbers of data points. If there are n sets of transient data, n lines are needed in the "csv"file.

Lines 10-224 contain the first set of the transient data for input and output variables. The number of columns should match the numbers of input and output variables. In this case, there are two input and two output variables and, therefore, each line contains 4 data columns. There are a total of 215 lines since the entry in Line 8 is 215.

Line 225-549 contain the second set of the transient data for input and output variables. There are a total of 325 lines since the entry in Line 9 is 325.

Note: If there are more than 2 sets of transient data, more data lines need to be appended to the "csv" file.

2. Start D-RM Builder from Windows "Start" button or from the desktop shortcut. The main window of the D-RM Builder displays.

3. Issue the **Setup→Read Custom Training Data** command. A "Custom training data file" dialog window displays. Browse to the working directory and choose the "VdV_Train_Plant.csv" file. A message is displayed in the message box of the main window to confirm the successful read of the "csv" file. The inlet and outlet streams connected to a box is also displayed in the upper section of the D-RM Builder window.

4. Issue the **Setup→Read Custom Validation Data** command. A "Custom validation data file" dialog window displays. Browse to the working directory and choose the "VdV_Valid_Plant.csv" file. A message is displayed in the message box of the main window to confirm the successful read of the "csv" file.

5. Click the **Build→D-RM Model Type** menu and make sure "DABNet" is checked.

6. Issue the **Build→Generate Reduced Model** command. A "DABNet DRM Parameter Dialog" window displays as shown in Figure 17. Select the "1. Ca_Product" in the "Output Variable List"

list box and click the "Optimize Fast Pole" radio button. Likewise, select the "2. Cb_Product" in the "Output Variable List" list box and click the "Optimize Fast Pole" radio button. Keep all other inputs unchanged and click "OK". The training process starts immediately. The mouse icon is switched to a busy (rotating circle) icon until the training process is completed. It may take a few minutes for D-RM Builder to optimize the pole values in the training process.

7. Save the case by issuing the **File→Save** command or clicking the toolbar icon  . Since the dynamic reduced model has been generated, it is time to save the case file. In the "Save D-RM Builder File" dialog window, give a file name and click "Save" to save the file.

8. To find how good the D-RM model prediction is compared to the provided transient response, use the commands under the **Post-Process** menu. First confirm the "Use Balanced Model for Prediction" option under the **Post-Process** menu is selected. Issue the **Post-Process→Predict Training Response** command or click the toolbar icon  . 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.

9. Issue the **Post-Process→Plot Training Responses** command or click the toolbar icon  to visualize the results. The "Result Plotting Dialog" window as shown in Figure 49 displays. Select the two input variables and the two output variables listed in the "Select Input" and "Select Output" list boxes. Accept the defaults with all of the check boxes selected and then click "OK". A Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 50. Note that the displayed data contains two sections with the first section before 2.15 hour based on the first set of the custom data while the second section after 2.15 hour based on the second set of the custom data.

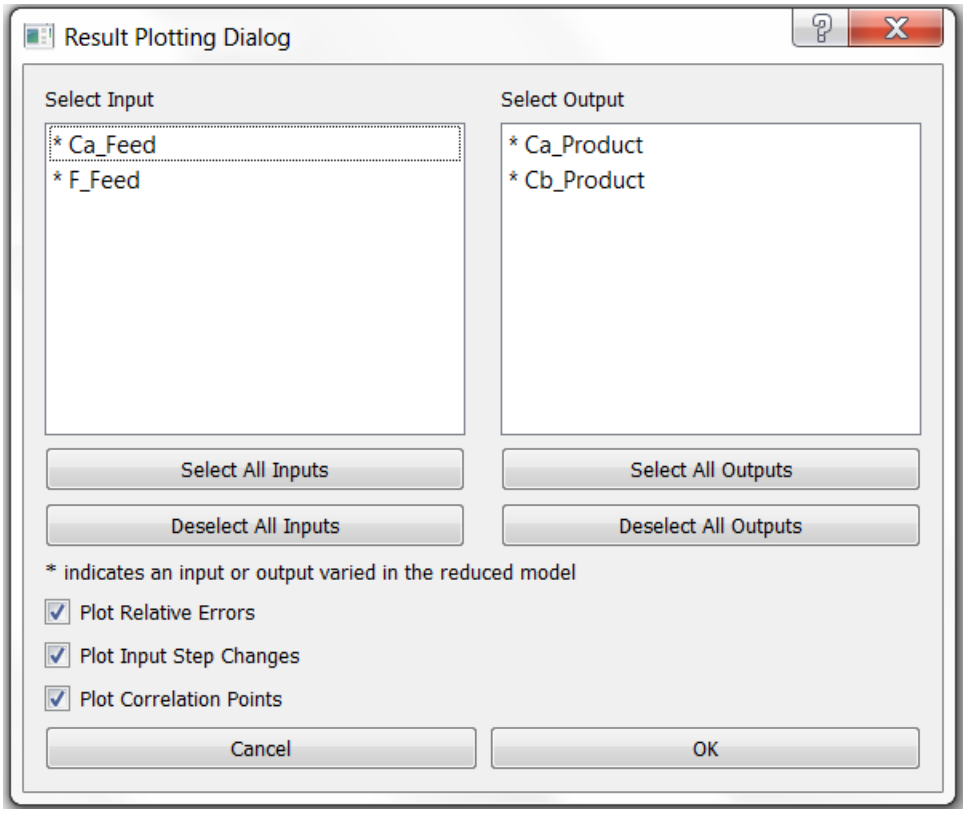

Figure 49: Result Plotting Dialog Window For Custom Input/Output Variables

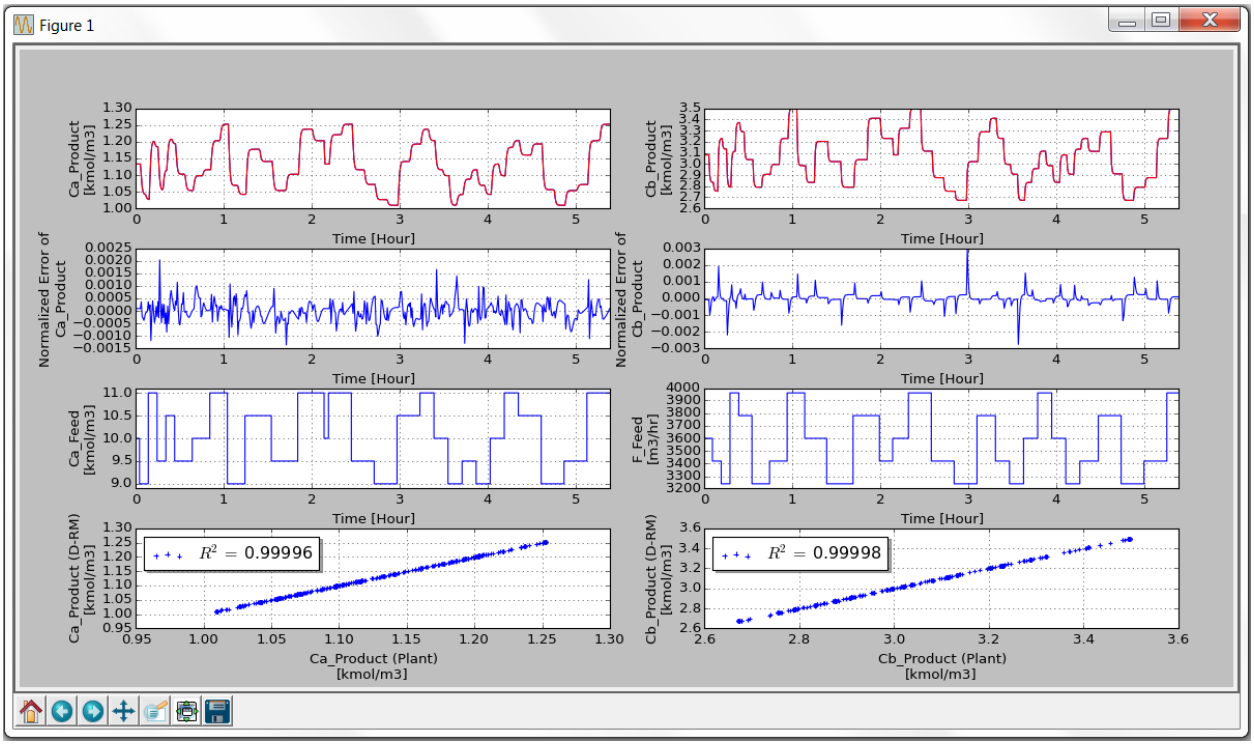

Figure 50: Plots of D-RM Predicted and Provided Custom Data For Training

10. Issue the **Post-Process→Predict Validation Response** command.

11. Issue the **Post-Process→Plot Validation Responses** command to visualize the results for the validation data. The "Result Plotting Dialog" window as shown in Figure 49 displays. Select the two input variables and the two output variables listed in the "Select Input" and "Select Output" list boxes. Accept the defaults with all of the check boxes selected and then click "OK". A Matplotlib window named "Figure 1" displays in a separate window as shown in Figure 51. It can be seen that the generated D-RM is quite accurate with D-RM model prediction matching the custom data quite well. Note that there is only one set of transient data for validation (see the contents of "VdV_Valid_Plant.csv"). Close the "Figure 1" window.

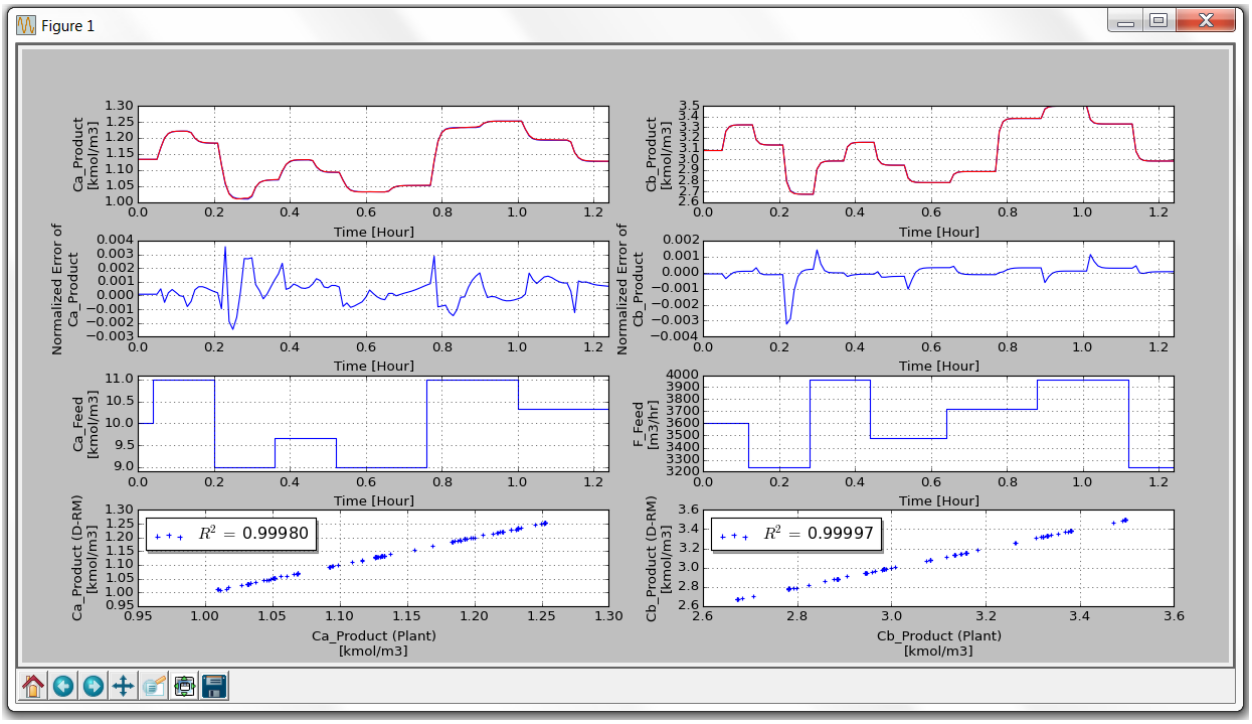

Figure 51: Plots of D-RM Predicted and Provided Custom Data For Validation

12. Save the D-RM Builder case by issuing the **File→Save** command and then close the D-RM Builder window.

2.8. Testing Generated D-RM for Dynamic Simulation in MATLAB

After a D-RM is generated by the D-RM Builder, it 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 installed as a part of D-RM Builder release. 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 "examples\Matlab" directory. Also installed in the "Matlab" folder is a MATLAB driver code named "Test_DRM.m", which contains MATLAB commands to read a MATLAB script file exported from the D-RM Builder, construct the corresponding MATLAB object 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. 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 launching D-RM Builder, opening the case file "case1_VdV.drmb" and then issuing the **File→Export→D-RM As MATLAB Script File** command. The "Save DRM As Matlab Script" dialog window displays. Accept the default file name as "case1_VdV.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 Training Data" window, enter the file name as "case1_VdV_training.csv" and then click "Save".

3. Find the MATLAB files in the D-RM Builder installation directory. Browse to the "examples\Matlab" 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 "examples\Matlab" 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 review the commands in the script file. The script enables the user to open a ".m" file that is exported from the D-RM Builder and contains the D-RM parameters. It then runs the ".m" file to assign the variables to the MATLAB workspace, creates either a "DRM_DABNet" object or a "DRM_NARMA" object, and then passes the variables in the workspace as parameters of the class constructor. It allows the user to 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 the Run icon on the toolbar. A "Choose D-RM Model Parameter File" window displays.

5. Browse to the directory where the "case1_VdV.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 "case1_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 "case1_VdV_training D-RM", and then click "Save". A file named "case1_VdV_training_D-RM.csv" is created in the directory. Meanwhile, a MATLAB plot window as shown in Figure 52 displays.

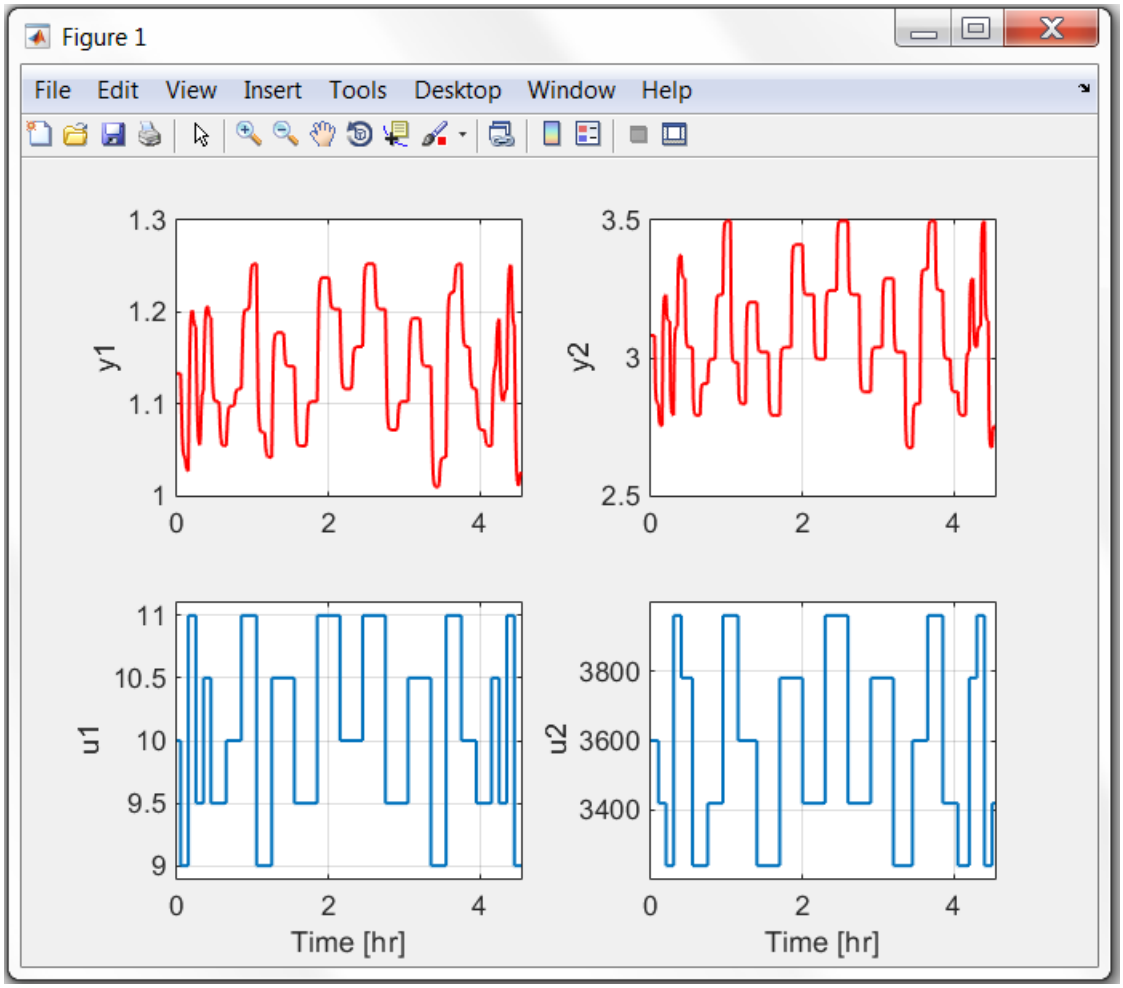

Figure 52: Predicted Response by D-RM Object in MATLAB

Compare the plots with those in Figure 19; the curves of the output variable are essentially the same, indicating the MATLAB object for the D-RM predicts the same responses as the Python 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. Support Information

To obtain support for this tool, send an email to ccsi-support@acceleratecarboncapture.org.

4. References

[1] G. B. Sentoni, L. T. Biegler, J. B. Guiver, and H. Zhao, "State-space nonlinear process modeling: Identification and universality," AIChE Journal, vol. 44, pp. 2229–2239, (1998).

[2] K. S. Narendra, "Adaptive control using neural networks and approximate models," IEEE Transaction on Neural Networks, vol. 8, no. 3, pp. 475–485, (1997).

[3] Jinliang Ma, Priyadarshi Mahapatra, Stephen E. Zitney, Loren T. Biegler, and David C. Miller, "D-RM Builder: A software tool for generating fast and accuratenonlinear dynamic reduced models from high-fidelity models," Computers and Chemical Engineering, 94, 60-74 (2016).