CCSI Advanced Process Control Framework

User Manual

Version 2.0.0

March 2018

Table of Contents

List of Figures

CCSI Advanced Process Control Framework

1.0 ABSTRACT

The Advanced Process Control (APC) Framework is an integrated software framework in the Carbon Capture Simulation Initiative (CCSI) Toolset for advanced process control of energy and chemical systems, including CO_2 capture units. This framework enables efficient dynamic transition of the process to a desired setpoint and effective mitigation of process uncertainties by leveraging “fast” D-RMs as predictive models. In recent years, model predictive control (MPC) has been the most successful APC technique applied in the process and energy industries. The MPC formulation naturally handles process time-delays, multivariable interactions, and constraints. However, the performance of industrial MPC applications has been limited by the use of linear models and the standard “additive output disturbance” assumption to compensate for plant-model mismatch. In the presence of strong nonlinearity within processes along with significant increase in computational power over the past decade, use of nonlinear model predictive control (NMPC) is no longer a hurdle for achieving real-time controller performance. Motivated by this, the APC Framework was implemented in MATLAB and provides a suite of nonlinear model-based controllers, each with core characteristics based on the type of D-RM, nonlinear programming solver, and disturbance estimation technique used.

2.0 REPORTING ISSUES

To report an issue, please send an e-mail to ccsi-support@acceleratecarboncapture.org.

3.0 VERSION LOG

Product	Version Number	Release Date	Description
APC Framework	**2.0.0**	**3/31/2019**	**Open Source Release**
CCSI Advanced Process Control Framework	**2015.10.00**	**11/20/2015**	**2015 November IAB Release** Major code revision and feature additions including: • APC Framework Simulation Tool GUI • Event-Driven Run-Time APC Tuning • Inclusion of numerous open-source optimizers including IPOPT, NLOPT, OOQP, etc. • State and Disturbance Estimation through Extended and Unscented Kalman Filtering
APC Framework	2015.06.0	06/30/2015	2015 June Release – Major revision including speed improvement fixes.
APC Framework	2014.10.0	10/31/2014	2014 October IAB Release – First release featuring DABNet-based NMPC and MMPC control toolboxes for nonlinear control. Framework includes APCConfiguration and APCSimulation Tools.

APC Framework

1.0 INTRODUCTION

The APC Framework is a software tool used to implement APC methodologies for CO_2 capture systems. This unified platform is written in MATLAB and can be used to setup an APC object specific to a "plant", which can either be a high-fidelity dynamic model of the plant or an actual CO_2 capture plant. The APC object can be generated using either a Dynamic Reduced Order Model (D-RM), obtained from CCSI's D-RM Builder tool or a state-space model generated from step tests performed on the "plant". The APC Framework also provides a dynamic simulation platform where the APC object, mentioned above, is linked to the high-fidelity "plant" model for testing and implementing real-time optimal control. The APC object uses sophisticated model-based control algorithms, such as model-predictive-control (MPC).

1.1 Motivating Example

Many of the CO_2 capture systems have significant nonlinear dynamics due to nonlinear behavior of various hydrodynamic factors, time-delays involved in adsorbing medium circulation, recycle loops, and nonlinear changes in physical and chemical material characteristics, especially those nearing the physical degradation limit. Therefore, a model-based controller can dynamically and efficiently move the system from one setpoint to another, based on its predictive capabilities, while meeting design and operating constraints. Any measured disturbance modeled within the controller can be rejected in a very effective fashion, due to predictive knowledge of the effect the disturbance could have on the system.

The solid sorbent-based bubbling fluidized bed (BFB) CO_2 adsorber-reactor for post-combustion carbon capture is a good example to which the APC Framework tool can be applied. The dynamics of the adsorber-reactor were 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. With a classical control approach using Proportional-Integral-Derivative (PID) control at the supervisory control layer, a trade-off between a significant process settling time or the presence of large overshoots (or undershoots) was observed when the overall CO_2 capture rate was controlled during flue-gas flowrate changes. The predictive knowledge of the process behavior can play a substantial role in mitigating either over-aggressive or sluggish behavior of controller dynamics. Implementing multiple-model predictive control with additive-output-disturbance rejection [8] resolved both the settling time and overshoot/undershoot issues simultaneously, yielding a significantly better dynamic behavior to flue-gas flowrate change as process disturbance. This will be demonstrated through one of the examples later in this manual.

1.2 Features List

The APC Framework is a MATLAB-based application with separate scripts for configuring the controller object (specific to a "plant") and simulating the closed-loop plant. A GUI based interface is also available for closed-loop plant simulation and controller testing. In the current version, the following controllers and features are included:

1. Nonlinear Model Predictive Control (NMPC) with additive output disturbance using DABNet based D-RMs (generated from the D-RM Builder Tool) as the control-model.
2. Multiple Model Predictive Control (MMPC) with additive output disturbance using multiple linear state-space models to span the nonlinear regime, in a "model-bank", as the control-model used for predictions.
3. Nonlinear State Estimation using Unscented and Extended Kalman Filter.
4. Nonlinear Disturbance Estimation using a suite of estimated disturbance formulation including random-step input, random-ramp input, and periodic forms.
5. Suite of quadratic-programming (QP) optimizers for MMPC, including Mehrotra Predictor Corrector, Wright and Hildreth methods, and nonlinear-programming (NLP) solvers for DABNet-NMPC, including IPOPT and NLOPT.

1.3 Organization of the Manual

The goal of this APC Framework User Manual is to provide a thorough understanding of various usage, features, and limitations of the APC Framework by providing examples and detailed rationale involved at each stage of setup, configuration, and control implementation. After reading and utilizing various test-cases provided in the tutorial, a user can determine whether the framework proves useful for a specific application.

In the tutorial section (Section 2.0 Tutorial), sub-section 2.1 Toolbox Setup provides instructions for Installation/Uninstallation of the APC Framework. While this may seem trivial to those familiar with MATLAB, these instructions provide steps for the smooth implementation of various scripts, an example demonstration, and important information on clean usage.

The subsequent tutorial sub-sections present individual features available in the framework, through specific example cases. These example cases are based on various combinations of control-algorithms and "plant" interfaces. Currently, ACM-based and MATLAB DAE solver-based interfaces are supported. In general, a short description and theoretical basis of the specific feature are provided before the example cases are presented.

Although each example case is unique, it follows a defined conceptual path using the following steps:

1. Setup a Working Folder.
2. Define "plant" and I/O variables. Then setup plant's communication with MATLAB.
3. Define plant's initialization process and marching script.
4. Generate training-set(s) using input perturbation(s) for building D-RM and then obtain D-RM/control-model(s) from training data.
5. Configure controller (offline) using *APCConfigTool*.
6. Initialize "plant" and then run closed-loop simulation using *APCSimulationTool* or *APCSimulationToolGUI*.

Usage information for the software (Section 3.0 USAGE Information) includes environment/ prerequisites, support, and software restrictions. Advanced features (Section 4.0 Advanced Features) includes information on third-party software required to run the framework. The manual concludes with debugging information (Section 5.0 Debugging) which includes known issues and how to report software bugs.

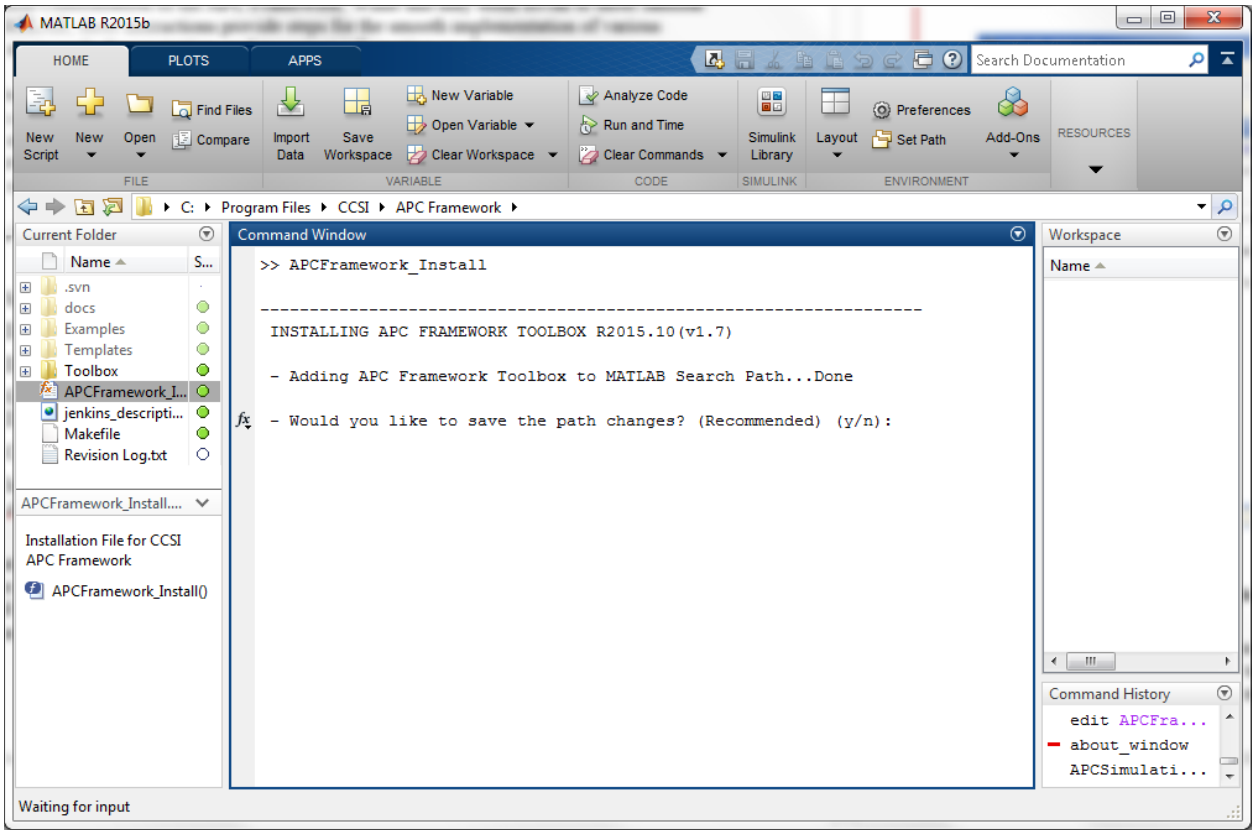
2.0 TUTORIAL

2.1 Toolbox Setup

The main APC Framework program files are provided in a compressed (`.zip`) format. Extract the contents of the .zip file to a file location determined by the user. It is recommended that all of the contents are extracted, including subfolders in "`C:\Program Files\CCSI\APC Framework`". For 64-bit OS, this subfolder may be "`C:\Program Files (x86)\CCSI\APC Framework`". This parent-folder location, determined by the user, is referred to as `<APCBaseFolder>` for the remainder of this manual.

Installation Instructions

In the parent folder, search for a MATLAB script-file named, "`APCFramework_Install.m`". Open MATLAB and then browse to `<APCBaseFolder>`. Launch the installer by typing "`APCFramework_Install`" in the MATLAB command window. The MATLAB Command Window output screen should be displayed similar to the example in Figure 1.



```
>> APCFramework_Install

----------------------------------------------------------------------
 INSTALLING APC FRAMEWORK TOOLBOX R2015.10(v1.7)

 - Adding APC Framework Toolbox to MATLAB Search Path...Done

 - Would you like to save the path changes? (Recommended) (y/n):
```

Figure 1: Installer script running in MATLAB Command Window.

If the user receives a warning message stating that a 32-bit MATLAB and/or Windows® OS is not detected, be aware that the user may not be able to run APC applications for ACM-based plants on their machine. The user may continue to use the framework for MATLAB DAE-based plants, similar to the MMPC Example 1. If the user's APC application does not involve Aspen-based dynamic model(s), select "y". **Note:** None of the framework capabilities itself will be limited, except that the user may not be able to test many of the examples provided with the tool.

The installer adds `<APCBaseFolder>` and its subfolder in the MATLAB search path for the current session only (i.e., as long as MATLAB is running). To add the `<APCBaseFolder>` and its subfolder to the MATLAB search path permanently, type "y". This action requires administrative privileges on the machine.

After the installation is complete it is important **not** to modify the contents of the "`<APCBaseFolder>\Toolbox`" subfolder or to change its location. In such an event, run the installer again to re-register the new location into the MATLAB search paths. To prevent this from happening accidentally, it is highly recommended that the user define the "`<APCBaseFolder>\Toolbox`" subfolder as read-only.

Uninstallation Instructions

To uninstall the APC Framework software tool, the user deletes the `<APCBaseFolder>`. To save the path changes permanently, search for a file named, "`pathdef.m`" in the `<MATLAB>\toolbox\local` folder, where `<MATLAB>` is the base location of the MATLAB installation. The absolute path of the file is displayed similar to the following:

```
C:\Program Files (x86)\MATLAB\R2015b\toolbox\local\pathdef.m
```

Open this file in a text editor program and then remove entries corresponding to the APC Framework location. This action requires administrative privileges on the machine. Save the file and then exit the system.

If all of the above uninstall changes were made while a MATLAB session was open, issue a "`rehash`" command at the MATLAB command window to rebuild the path.

2.2 Decoupled A-B Net-Based Nonlinear Model Predictive Control

Description

The Decoupled A-B Net (DABNet)-based NMPC strategy uses a fast and accurate nonlinear D-RM named, "DABNet" [1], at its core for control-predictions. The CCSI's D-RM Builder tool is used for generating such models from ACM-based "plant" or high-fidelity process models.

The methodology supporting DABNet dynamic reduced models is detailed in Sentoni et al., 1998 [1]. This model essentially is composed of a decoupled linear dynamic system followed by a neural-network based nonlinear static mapping of state variables to outputs. The linear dynamic system is initially spanned by a set of discrete Laguerre systems and then cascaded with a single hidden layer perception based on a feed-forward Artificial Neural Network (ANN) as shown in Equations (1) and (2).

$$\begin{aligned} \mathbf{x}_k^{ij} &= \mathbf{A}_{ij}\mathbf{x}_{k-1}^{ij} + \mathbf{B}_{ij}u_{k-1}^{j} \\ y_k^i &= NN_i\left(\mathbf{x}_k^i\right) \end{aligned} \qquad \textbf{(1)}$$

with,

$$\mathbf{x}_k^i = \begin{bmatrix} \mathbf{x}_k^{i1} \\ \mathbf{x}_k^{i2} \\ \mathbf{x}_k^{i3} \\ \vdots \\ \mathbf{x}_k^{i,nu} \end{bmatrix} \qquad \textbf{(2)}$$

where, given the current time step *k*, the following notations are used:

u_{k-1}^{j}	: jth input (scalar) between time-step k-1 and k
$\mathbf{x}_k^{ij}$	: decoupled process states at time-step k corresponding to input j and output i
$\mathbf{x}_k^{i}$	: augmented states at time-step k corresponding to output i given by Equation (2)
y_k^{i}	: ith output (scalar) at time-step k

$\mathbf{A}_{ij}, \mathbf{B}_{ij}$ are decoupled state-space parameters corresponding to input *j* and output *i*. $NN_i(\cdot)$ is the ANN function, mapping the *i*th augmented state vector to *i*th output. A model reduction technique (linear balancing) helps reduce the dimensionality of the perceptron input, although the user is also given the option to use the Laguerre states directly (without balancing) as the neural network input within the D-RM Builder.

Within the APC Framework, a state-estimation feature provides enhance control performance during noisy measurements (both outputs and disturbances). The Extended Kalman Filter (EKF) [6] and Unscented Kalman Filter (UKF) [2] based state-correction technique is utilized to filter and update the augmented states once new "plant" measurements are available. The corrected states are thereafter used in the control-model for better predictions.

Detailed descriptions of individual commands and screenshots are provided in the following section. It is highly recommend that the user completes this tutorial before using the tool for a personal, specific APC application.

Example 1 – NMPC for ACM-Based Plant (pH Neutralization Reactor)

A pH Neutralization Reactor example, provided in the "`<APCBaseFolder>\Examples\pH_Reactor`" subfolder, demonstrates the NMPC feature of the APC Framework for "plants" based on ACM. This nonlinear reactor model is a benchmark process popular in control literature (see Reference [1] for more details on the reactor model). This ACM dynamic model is considered the high-fidelity "plant" model used to test the APC Framework. The workflow involved in setting up a detailed control simulation to ACM-based "plant" models, in particular, is as follows:

1. Copy the entire example subfolder of pH Neutralization Reactor "`<APCBaseFolder>\Examples\pH_Reactor`" to a location determined by the user. This action requires the user to have write-permissions to this location. Browse to the "`pH_Reactor`" subfolder. The "plant" model is available for review by opening the file "`pH_Neut_2Tank.acmf`" in ACM. Streams Q1, Q2, Q5, and Q6 are acidic streams (Q2 and Q6 are constant auxiliary streams), whereas streams Q3 and Q7 are basic streams used to neutralize pH in both the reactors. In Example 1, the pH is controlled by manipulating the basic streams (Q3 and Q7) in the face of fluctuating acidic streams (Q1 and Q5), serving as disturbance inputs. For consistency, **do not** save any changes made to the ACM file. Verify that the "Run Mode" drop-down list in the toolbar is set to "Dynamic". If not, select "Dynamic" either in the toolbar or from the Run → Mode menu and then save the file. If "Dynamic" mode was not pre-selected, please send an e-mail to ccsi-support@acceleratecarboncapture.org.

   In addition, within ACM navigate to Tools → Settings from the main menu and then navigate to the Preferences tab. Confirm the "Allow setting of working folder location" check box is cleared. If not, clear the check box, click OK, and then save/close the flowsheet. This setting enables initialization and multiple simulation runs to occur in a single ACM instance and prevents any erratic Simulink-ACM communication behavior in the future steps.

2. Open MATLAB (if not already done) and browse/set the "Current Folder" to the newly copied "`pH_Reactor`" subfolder, which becomes the *working directory* in MATLAB.

   Double-click the "`pH_Neut.mdl`" Simulink model file to display a Simulink model as shown in Figure 2.

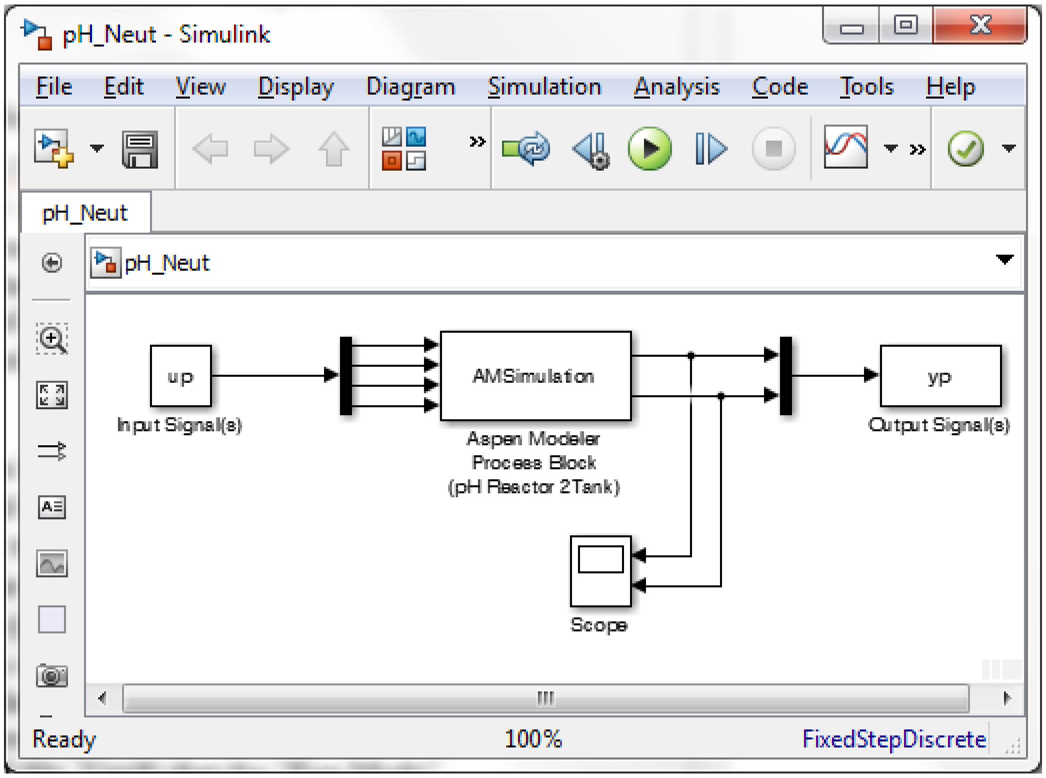
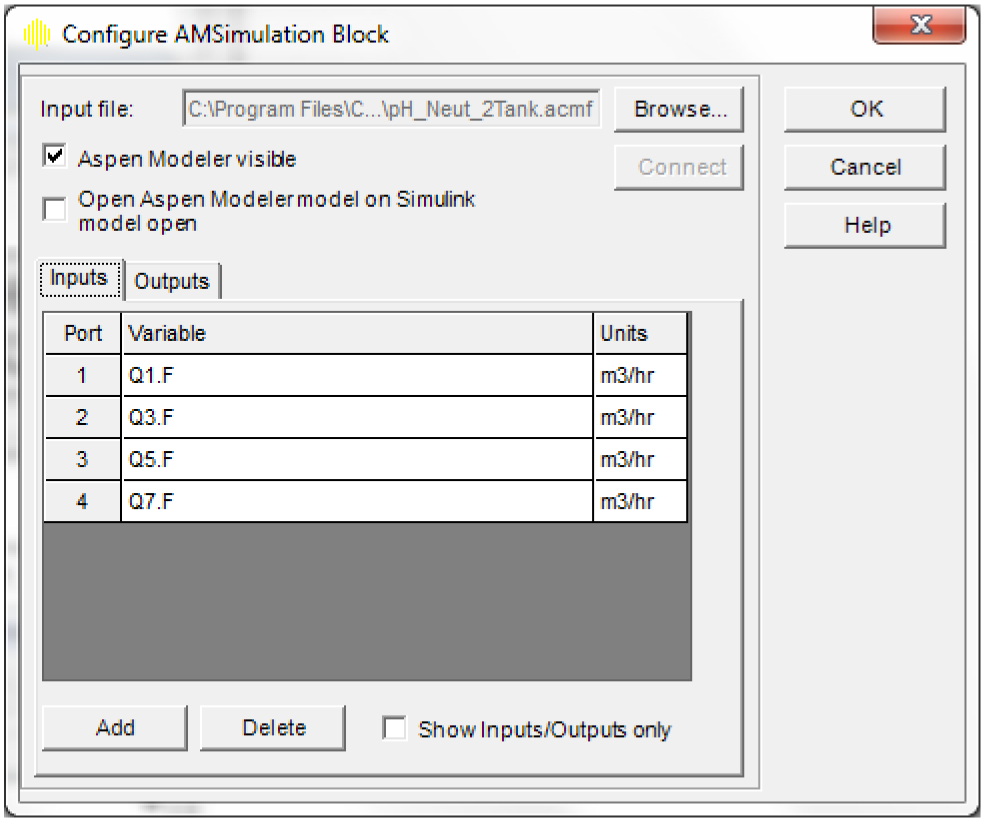


Figure 2: Simulink model with embedded ACM-Based 2-tank pH Neutralization Reactor.

Double-click “AMSimulation” to display the “Configure AMSimulation Block” window, as shown in Figure 3.



Input file: C:\Program Files\C...\pH_Neut_2Tank.acmf

[✓] Aspen Modeler visible

[ ] Open Aspen Modeler model on Simulink model open

Port	Variable	Units
1	Q1.F	m3/hr
2	Q3.F	m3/hr
3	Q5.F	m3/hr
4	Q7.F	m3/hr

[ ] Show Inputs/Outputs only

Figure 3: Configure AMSimulation Block window for pH Neutralization Reactor.

Click “Browse” and then select the ACM file in the working folder to launch the ACM. The input/output port variables should populate automatically. **Note:** Failure to do this step keeps the working folder linked to the original APC Framework Installation folder. Check for these input/output “I/O” variables:

Inputs: `Q1.F, Q3.F, Q5.F, Q7.F`

Outputs: `pH_Tank1.pH, pH_Tank2.pH`

If all the variables are consistent, click OK, and then save and exit the Simulink model. The ACM window may automatically close. **Note:** Never close the accompanying ACM window manually because it may lead to file corruption. Any ACM window invoked from within MATLAB/ Simulink automatically closes upon MATLAB exit.

3. Initially, any "plant" is preferably operating at a nominal operating "steady-state" condition. Although this is not a strict requirement, following this guidance makes the control response more tractable, by avoiding any unexpected jumps when the control loop is closed. For Example 1, the plant is initialized at nominal operating "steady-state" conditions in this step.

   There is no shortcut to do the initialization for an ACM-based "plant" from within MATLAB as opposed to MATLAB DAE-based "plant" models, where a separate initialization function/method may be coded. The basic way to do the initialization for an ACM-based "plant" at any operating condition is given as follows, with specific references to Example 1. **Note:** The following steps are provided as guidelines for initialization at new conditions. No actions are needed at this time, unless the user modified the ACM flowsheet from the steady-state conditions provided in a fresh installation.

   a. Open the pH Reactor flowsheet "`pH_Neut_2Tank.acmf`" in ACM.
   b. Define the new operating condition by modifying the input values. For example, to change the acid flowrate (Q1), double-click the corresponding streams. Make sure the flowrate field is a "fixed" variable and enter the new value. Similarly, make changes to the other variables which define the new condition.
   c. Conduct a steady-state run by changing the run mode to "Steady State". After the run is complete (and converged), note the relevant output values corresponding to those provided in the Output port specification within Simulink. These output values have to be provided as "`y_plant`" variable for many APC Framework scripts to work. For the D-RM Builder to build D-RM around this steady-state value, a snapshot file must be generated by selecting Tools → Snapshots from the menu (or pressing Alt-F3). Select the above steady-state run from the list of available snapshots and then select Export.
   d. Change the "Run Mode" to "Dynamic". This is an important step.
   e. Save the flowsheet.

   Typically, in a real plant or in simulator based plant-models, the system is physically or dynamically moved from one steady-state operating condition to the next. This is what has been virtually done using the steps above for ACM-based "plants".

   A MATLAB script "`Initialize_pH_Reactor.m`" is provided which can obtain output values "`y_plant`" quickly, assuming the ACM file is saved at the desired steady-state. This script dynamically marches the "plant" a few steps to automatically extract the steady-state output information using the Simulink interface.

   The simulation of the plant is completed using the marching script, "`SimulatePlant.m`". To successfully launch and march an ACM-based "plant" within APC Framework, two things are required: (1) the "plant" should be initialized at desired nominal-conditions within ACM, as highlighted in the above sub-steps and (2) the nominal I/O values, corresponding to the specified I/O ports to/from the AMSimulation Block in the previous step, should be present in "`u_plant`" and "`y_plant`" variables for the APC Framework scripts (which depend on "plant" initialization) to run properly. To view the code, navigate to the *working directory* and then open this file in MATLAB or issue the command, "`edit SimulatePlant.m`". A `sim` command is used to forward-march the "`pH_Neut`" Simulink model (configured in Step 2) by one sampling time from time `t_plant` to (`t_plant + Ts_plant`) taking `u_plant` as "plant" input and obtaining `y_plant` at the next time-step.

Important Note: This script can be named anything and must hold the marching script for the specific "plant" that the user investigates. It is clearly mentioned later (see sub-section 3.3 Restrictions), that the end-user is responsible for setting up the communication link between the controller and "plant". This link could be in the form of a high-fidelity simulation embedded within Simulink (similar to Example 1), a marching script for MATLAB-based "plant" objects (see the MMPC pH Reactor example), or hardware interface modules which could communicate with MATLAB through the MATLAB Data Acquisition Toolbox, LabVIEW interface, etc.

4. Now that the "plant" has been defined, the control-model is built using a D-RM for this plant. The APC Framework's NMPC algorithm uses DABNet-based data-driven D-RM for fast control predictions, obtained from the CCSI D-RM Builder tool. This step demonstrates how this is done within the scope of this framework. It is strongly recommended that the D-RM Builder User Manual is read for a thorough understanding of the user-interface. **Note:** A pre-built D-RM parameter file, "`pH_Neut_DRMParameters.m`", is provided in the example folder. The user may choose to skip the steps involving D-RM Builder training and D-RM generation and move to the end of this step (refer to the [*redirect*] comments, following Step 4.j. below) for generating an APC Framework compatible D-RM from a D-RM parameter file.

   The following enumerates the necessary information to set-up the D-RM training runs:

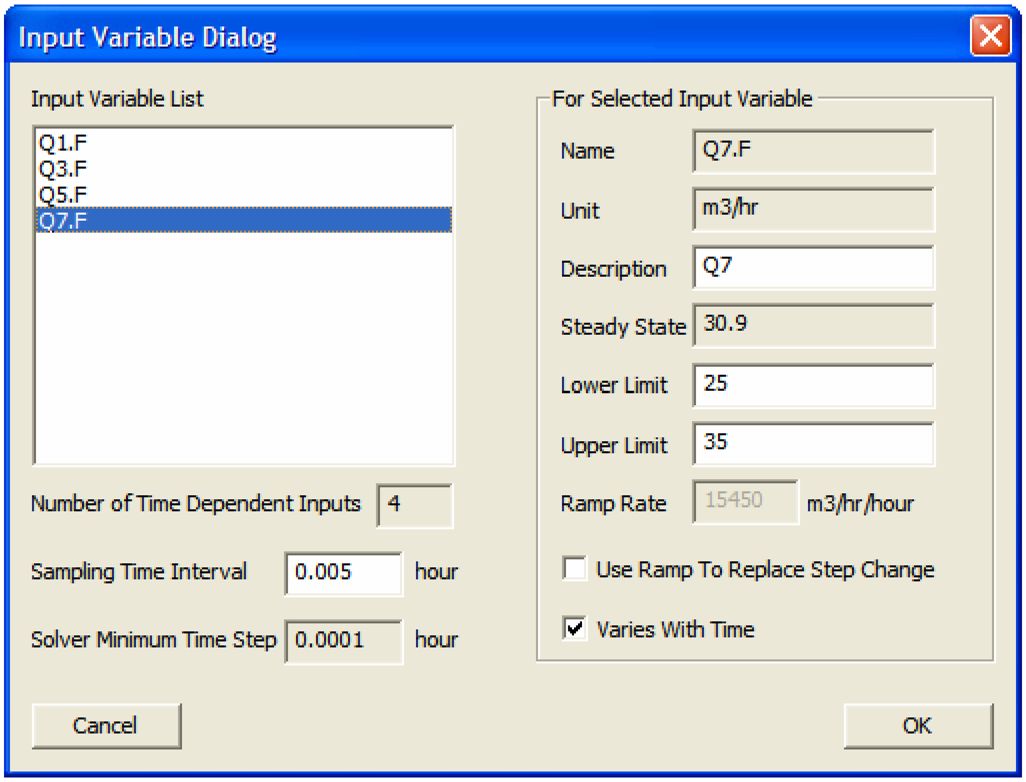
   a. Open the CCSI D-RM Builder tool. Specify the ACM/high fidelity model file as "`pH_Neut_2Tank.acmf`" in the current working folder.

   b. Specify the snapshot file as "`pH_Neut_2Tank_SS.asnp`". This snapshot file contains instantaneous values of all state-variables when the snapshot was taken. The number of such state-variables might range from hundreds to tens of thousands (depending on scale of process). The pH Neutralization process contains approximately 150 such variables. D-RM Builder uses the snapshot file to extract nominal I/O values corresponding to variables provided as I/O ports in the Simulink file (specified below). These nominal values are used as an operating point around which the DABNet-based D-RM is built; therefore, it is important for the ACM file to be last-saved at operating conditions corresponding to the snapshot file provided in this manual. If this is not the case, please refer to guidelines for plant initialization in Step 3.

   c. Specify the Simulink file as "`pH_Neut.mdl`". This file was previously configured in Step 2.

d. Configure Input Variables as shown in Figure 4. Enter a Sampling Time Interval of "0.005 hour". Enter Lower/Upper Limits of both acid flowrates (Q1 and Q5) as "2" and "3", respectively. Similarly, enter Lower/Upper Limits for base flowrates (Q3 and Q7) as "25" and "35", respectively. For all four variables, select the "Varies with Time" check box and then clear the "Use Ramp to Replace Step Change" check box.

   Process heuristics suggests a sampling time of at-most a tenth of the process-time is a good starting point. For pH reactor, a settling time of 5–10 minutes is observed, which gives a sampling time of any number lower than 0.5 minutes (~0.008 hour) adequate time to capture most of the high-frequency dynamics. Enter "0.005 [hour]" as the Sampling Time Interval.


Input Variable Dialog

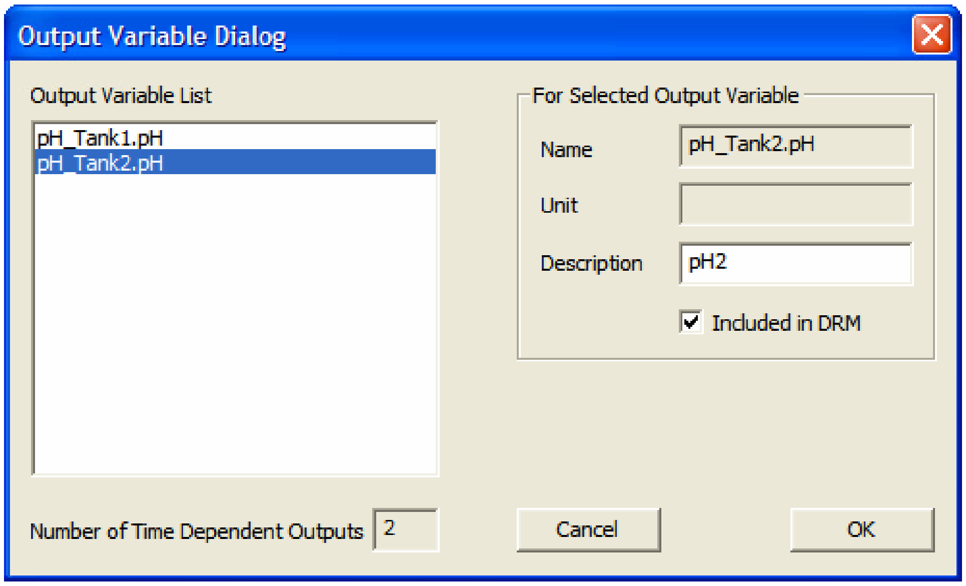
Input Variable List
Q1.F
Q3.F
Q5.F
Q7.F

Number of Time Dependent Inputs: 4
Sampling Time Interval: 0.005 hour
Solver Minimum Time Step: 0.0001 hour

For Selected Input Variable
Name: Q7.F
Unit: m3/hr
Description: Q7
Steady State: 30.9
Lower Limit: 25
Upper Limit: 35
Ramp Rate: 15450 m3/hr/hour
☐ Use Ramp To Replace Step Change
☑ Varies With Time

Figure 4: D-RM Builder Input Variable Dialog window for pH Neutralization Reactor.

e. Configure the output variables as shown in Figure 5. Include both the pH variables in the D-RM.


Output Variable Dialog

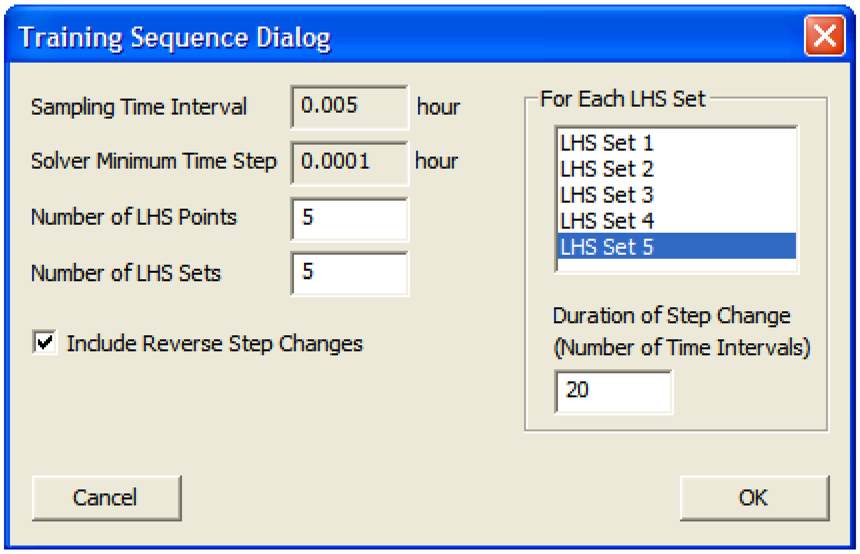
Output Variable List
pH_Tank1.pH
pH_Tank2.pH

For Selected Output Variable
Name: pH_Tank2.pH
Unit:
Description: pH2
☑ Included in DRM

Number of Time Dependent Outputs: 2

Figure 5: D-RM Builder Output Variable Dialog window for pH Neutralization Reactor.

f. The training sampling needs to be determined. Reference [1] provides information that this pH reactor process is a highly nonlinear system. Therefore, the large sample space should include broad frequency ranges into the D-RM training process. In the Training Sequence Dialog shown in Figure 6, set the "Number of LHS Sets" to a value of 5 and the "Number of LHS Points" to 5. This corresponds to using five Latin Hypercube Sampling (LHS) sets with five points in each set. Specify the "Duration of Step Change (Number of Time Intervals)" as 2, 5, 10, 15, and 20 to assign time-intervals ranging from 36 seconds to 6 minutes. Click OK when finished. **Note:** Providing large sets of training samples may tend to generate an over-fitted D-RM.


Training Sequence Dialog

Sampling Time Interval: 0.005 hour
Solver Minimum Time Step: 0.0001 hour
Number of LHS Points: 5
Number of LHS Sets: 5
☑ Include Reverse Step Changes

For Each LHS Set
LHS Set 1
LHS Set 2
LHS Set 3
LHS Set 4
LHS Set 5

Duration of Step Change (Number of Time Intervals): 20

Figure 6: D-RM Builder's Training Sequence Dialog window for defining training sets for pH Neutralization Reactor D-RM.

g. Perform the training simulation, by clicking Perform Training Simulation in the Build menu, to launch the MATLAB command and ACM window. Completing the large training runs may take a few minutes. Once the run is complete, a quick MATLAB plot shows the input sequence and plant responses.

h. Verify that “DABNet” is selected in Build → D-RM Model Type from the menu. Open the “DABNet DRM Parameter” Dialog Box by selecting “Generate Reduced Model” from the Build menu. Select “Optimize Pole Value” for all outputs; keep default values for other fields. Once OK is clicked, the optimization runs begin and the system provides the best pole value for each output. Refer to the D-RM Builder documentation for more details. When the run is complete the D-RM should be successfully generated, click Save.

i. To verify the qualitative accuracy of the generated D-RM, a comparative plot of the ACM plant versus D-RM predictions is visualized. Verify “Used Balanced Model for Predictions” is selected under the PostProcess menu, and then generate the D-RM predictions by selecting “Predict Training Responses” under the same menu. The dynamic responses can be compared by using “Plot/Compare Training Responses” which displays the comparative plot similar to Figure 7.

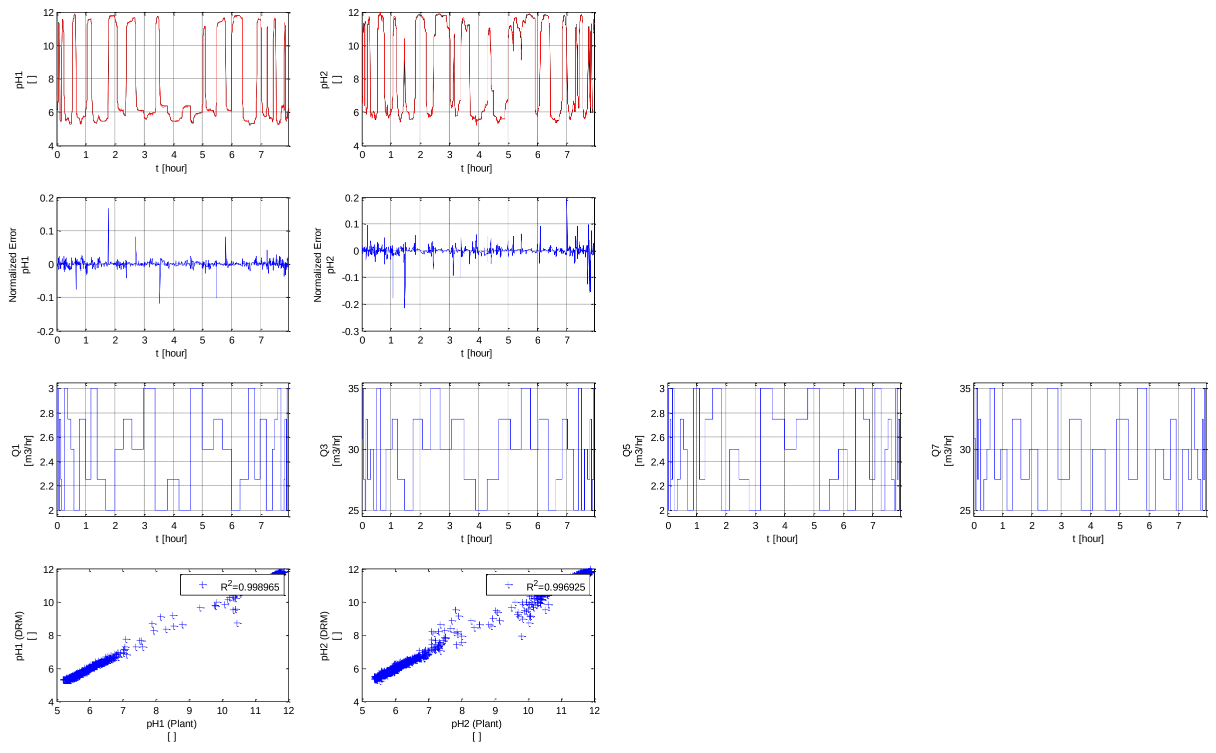


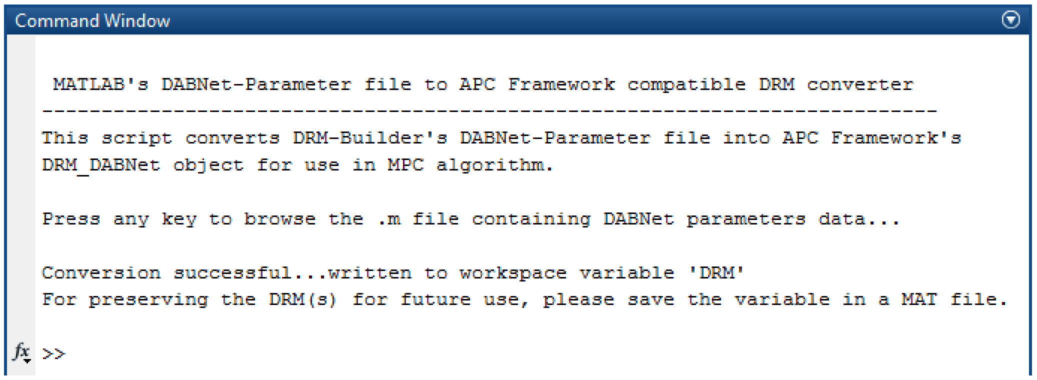
Figure 7: Comparative responses of ACM-Based “plant” versus DABNet D-RM for training data.

Note: The validation runs are the real-test for quality of generated D-RM. Refer to documentation on how to build, predict, and plot validation set data and remember to save the D-RM case file.

j. The most important step, in context of the current framework, is to export the D-RM in a format compatible with APC Framework. Navigate to File → Export → D-RM as a MATLAB script file. This step enables the user to save the D-RM parameters in a MATLAB .m format. Enter a filename determined by the user. As mentioned earlier, a pre-built file "`pH_Neut_DRMParameters.m`" is provided in the example folder. The user may use this file or the one that was created.

[Redirect here if using the pre-built D-RM parameters file]

Now that the D-RM parameter file is built, a DABNet-based D-RM object in MATLAB is created. Launch the APC Framework's DABNet-parameter to the D-RM converter script by issuing the command, "`DABNetParameter2DRM`," in the MATLAB command window. Browse to "`pH_Neut_DRMParameters.m`" or the one specified above, as shown in Figure 8.



```
 MATLAB's DABNet-Parameter file to APC Framework compatible DRM converter
----------------------------------------------------------------------------
This script converts DRM-Builder's DABNet-Parameter file into APC Framework's
DRM_DABNet object for use in MPC algorithm.

Press any key to browse the .m file containing DABNet parameters data...

Conversion successful...written to workspace variable 'DRM'
For preserving the DRM(s) for future use, please save the variable in a MAT file.

fx >>
```

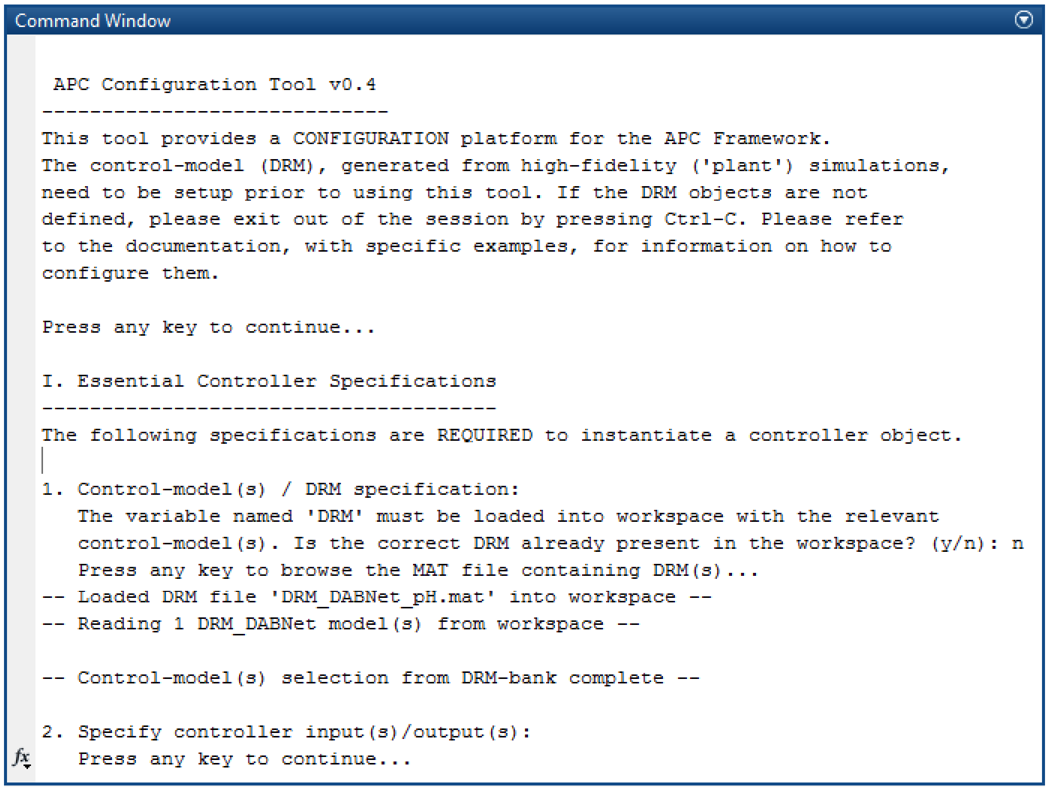
Figure 8: APC framework's DABNetParameter2DRM script.

After the conversion is complete, a workspace variable named, "`DRM`", is created. Save this variable in a MATLAB data file, by right-clicking the variable name and then selecting "Save As". A preloaded MATLAB data file, "`DRM_DABNet_pH.mat`", is already created in the example folder, although this file can be overwritten for consistency with the user's MATLAB version. **Note:** This file can be named anything, although this file is referred to as the D-RM data-file or "`DRM_DABNet_pH.mat`" interchangeably throughout the rest of Example 1. This step concludes with a demonstration on how to create control-model or D-RM for use within the APC object.

5. This step includes instructions on configuring the controller (offline) for later use alongside plant simulations (online), which is crucial for any control-implementation. Launch the APC Configuration Script by issuing the command, "`APCConfigTool`." The Welcome window, with the current version information, displays a short summary and prerequisites for running this tool. The primary requirement is the availability of the "`DRM`" variable in the workspace. If the "`DRM`" variable is not present, the tool prompts the user to browse for a MATLAB data (.mat) file containing this variable, similar to "`DRM_DABNet_pH.mat`" created/mentioned in the previous step. To skip this step of creating an APC object, advance to the closed-loop simulation section (Step 6 below); a preconfigured APC object "`APC_DABNet_pH.mat`" is present in the example folder.

a. The availability of the "DRM" variable is required. If the user continued directly from Step 4 above, this variable (in its correct form) should already be present in the workspace. In this case, type "y".

   If the workspace was cleared, the variable was modified in any way, or MATLAB was re-launched, type "n" here. The user is prompted to browse the data file. Select "DRM_DABNet_pH.mat". The tool will confirm that a DABNet D-RM was successfully read, as shown in Figure 9.



Command Window

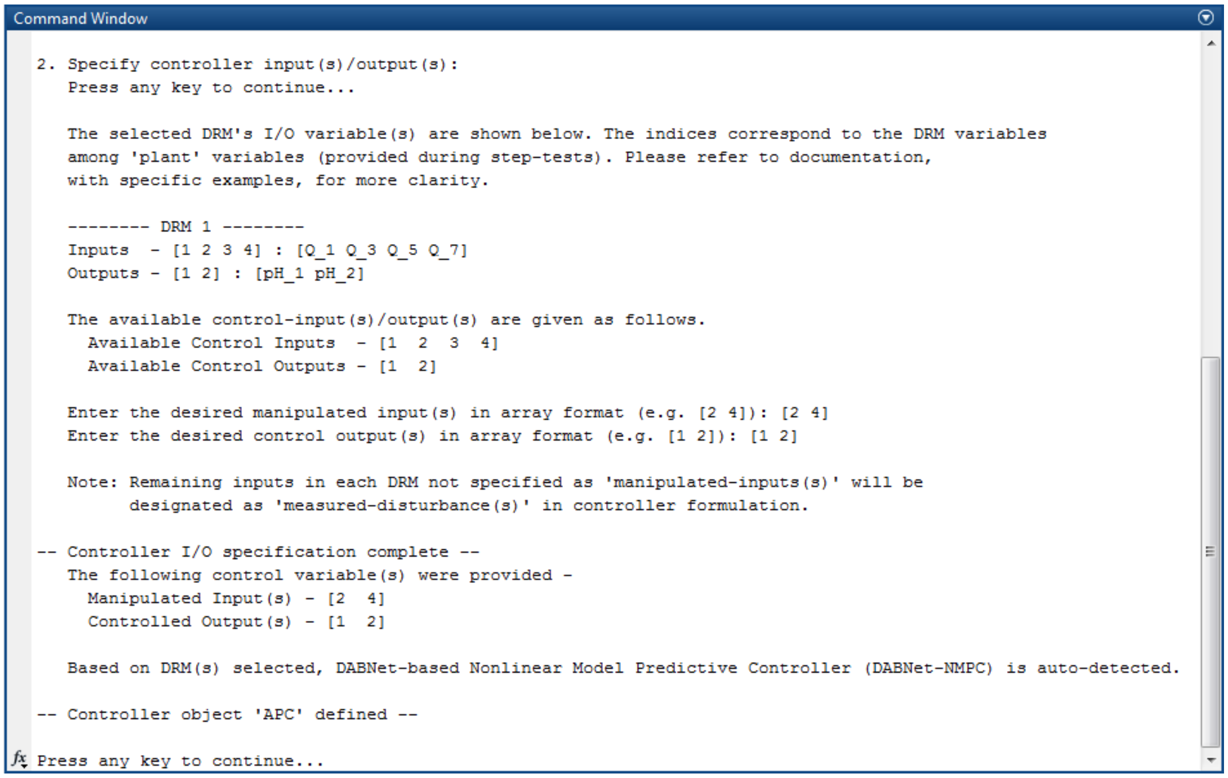
```
 APC Configuration Tool v0.4
-----------------------------
This tool provides a CONFIGURATION platform for the APC Framework.
The control-model (DRM), generated from high-fidelity ('plant') simulations,
need to be setup prior to using this tool. If the DRM objects are not
defined, please exit out of the session by pressing Ctrl-C. Please refer
to the documentation, with specific examples, for information on how to
configure them.

Press any key to continue...

I. Essential Controller Specifications
--------------------------------------
The following specifications are REQUIRED to instantiate a controller object.

1. Control-model(s) / DRM specification:
   The variable named 'DRM' must be loaded into workspace with the relevant
   control-model(s). Is the correct DRM already present in the workspace? (y/n): n
   Press any key to browse the MAT file containing DRM(s)...
-- Loaded DRM file 'DRM_DABNet_pH.mat' into workspace --
-- Reading 1 DRM_DABNet model(s) from workspace --

-- Control-model(s) selection from DRM-bank complete --

2. Specify controller input(s)/output(s):
   Press any key to continue...
```

Figure 9: D-RM specification for DABNet-Based NMPC within the APC Configuration tool.

b. The second step details controller I/O specifications. The available control inputs and outputs, along-with indices corresponding to plant ports (see Step 4.d.) are shown. The manipulated variables(s) are chosen from within these available control inputs. “Availability” of control inputs and outputs is solely based on those control variables that were modeled in the D-RM.

Enter the input(s) from among the available inputs shown in the form of MATLAB array (enclosed within square brackets, separated by either “space”, “comma”, or “semi-colon”). **Note:** The index number(s) correspond to the plant’s input-port sequence defined during plant specification. Enter “[2 4]” for Manipulated Inputs, which corresponds to Q3 and Q7 flowrates.

Similarly, enter the indices corresponding to the controlled output(s) or commonly referred to as Controlled Variable (CV). Enter “[1 2]” for Controlled Output(s), corresponding to pH1 and pH2 values (refer to Figure 10 for guidance).



```
2. Specify controller input(s)/output(s):
   Press any key to continue...

   The selected DRM's I/O variable(s) are shown below. The indices correspond to the DRM variables
   among 'plant' variables (provided during step-tests). Please refer to documentation,
   with specific examples, for more clarity.

   -------- DRM 1 --------
   Inputs  - [1 2 3 4] : [Q_1 Q_3 Q_5 Q_7]
   Outputs - [1 2] : [pH_1 pH_2]

   The available control-input(s)/output(s) are given as follows.
     Available Control Inputs  - [1  2  3  4]
     Available Control Outputs - [1  2]

   Enter the desired manipulated input(s) in array format (e.g. [2 4]): [2 4]
   Enter the desired control output(s) in array format (e.g. [1 2]): [1 2]

   Note: Remaining inputs in each DRM not specified as 'manipulated-inputs(s)' will be
         designated as 'measured-disturbance(s)' in controller formulation.

-- Controller I/O specification complete --
   The following control variable(s) were provided -
     Manipulated Input(s) - [2  4]
     Controlled Output(s) - [1  2]

   Based on DRM(s) selected, DABNet-based Nonlinear Model Predictive Controller (DABNet-NMPC) is auto-detected.

-- Controller object 'APC' defined --

fx Press any key to continue...
```

Figure 10: Controller I/O specification for DABNet-Based NMPC within the APC Configuration tool.

Note: The message stating that the remaining inputs are designated as “measured-disturbances” in control calculation is displayed in the MATLAB command window. The knowledge of measured-disturbance, modeled in the prediction-model, is significantly beneficial to control response for disturbance-rejection and is traditionally analogous to feed-forward information in a Feed-Forward Feed-Back (FF-FB) controller.

After the I/O specifications are complete, the controller object is instantiated. This signifies that the controller has the necessary ports to communicate with the specific plant for which the controller was configured. Additional controller “setup” and “initialization” steps are required before the control-loop can be closed for online-control.

c. Additional controller specifications may be completed. These configurations consist of the typical controller parameters including constraint specification, input-output weight(s), and control/prediction horizon information. Type "n" to exit. Run the pre-build script file to setup the controller, as described below.

   If the user exited in the previous step, additional controller specifications can still be provided in the MATLAB command window. For example, to specify a prediction horizon, just invoke the command "`APCParameters.P = 20`". Once all the controller parameters are defined, issue the following command:

```
APC.setupController(APCParameters);
```

   The following variables **must** be specified to fully define the controller object:

   - Input Absolute and Rate Constraints (u_{min}, u_{max}, Δu_{min}, Δu_{max})
   - Output [Hard/Soft] Constraints (y_{min}, y_{max})
   - Input Weights (w_u)
   - Output Weights (w_y)
   - Prediction Horizon (P)
   - Control Horizon (M)

   A pre-built script named, "`SetupDemoAPCPars_pH.m`", is provided in the working folder, which when run automatically updates the parameters into the controller. Run this file by opening it in MATLAB editor and then selecting "Run" or issuing the command "`SetupDemoAPCPars_pH`" at the MATLAB command window. To save, right-click the "`APC`" variable in the workspace and then choose an appropriate filename determined by the user, or overwrite the "`APC_DABNet_pH.mat`" file.

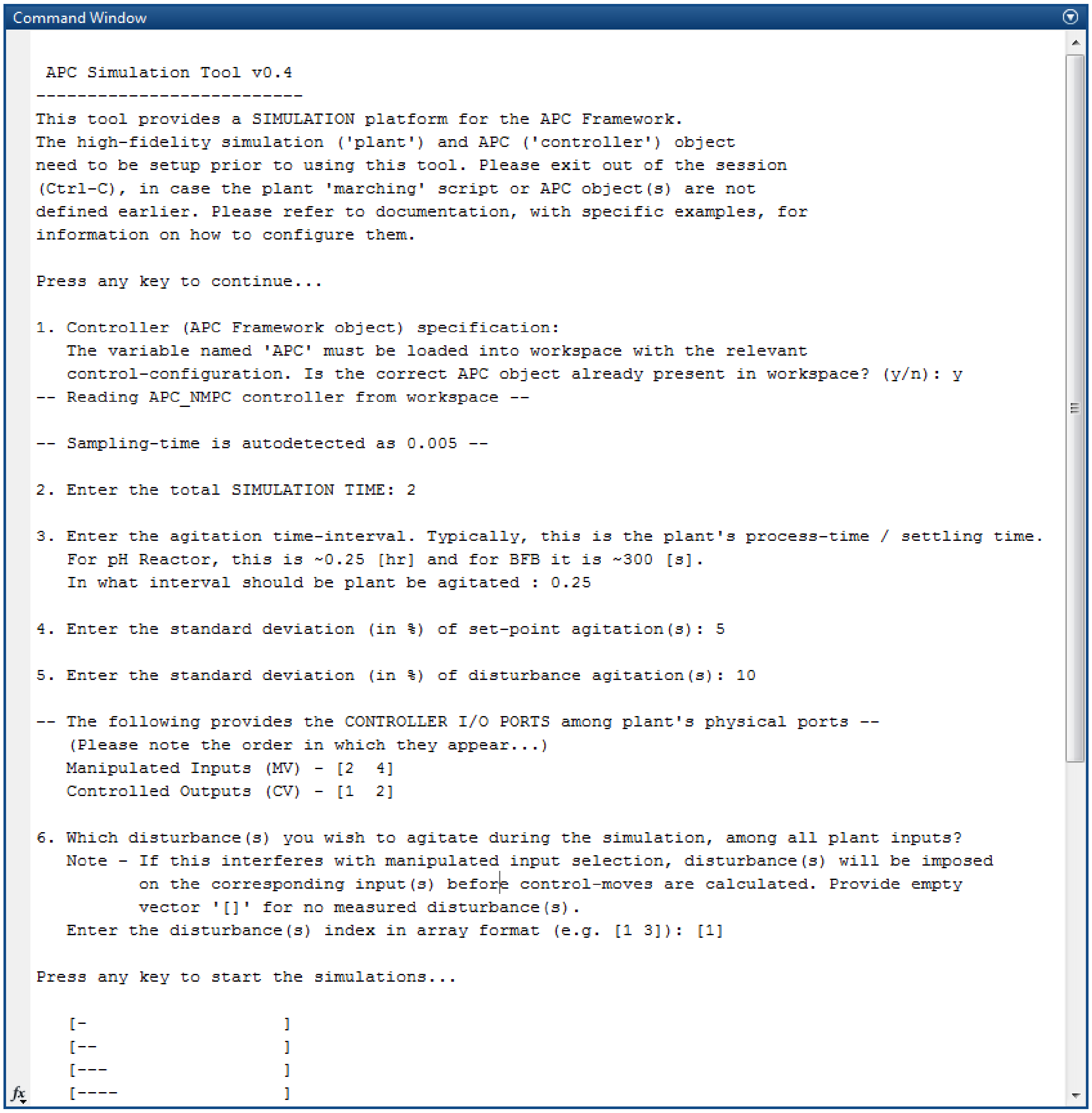
   Important Note: Every time a controller parameter is changed, verify the `APC.setupController` command has been issued (as shown above) to update the controller internal parameters/weights, etc. Not doing so always results in incorrect closed-loop responses, since the older parameter(s) would not be updated.

6. Simulating the pH Neutralization Reactor "plant" with control implemented or in "closed-loop" mode is detailed in this section. As in a real-plant scenario, the pH Reactor "plant" must be operating in some nominal state. The input-output information needs to be made available to the controller. The controller object can then, with this information, initialize the embedded D-RMs within it.

   The plant initialization, and making the initial plant values available to the controller, is the responsibility of the plant operators or control engineers; therefore, it is kept separate from the Simulation Tool. For Example 1, the "`Initialize_pH_Reactor.m`" script is run or commands are invoked, as given in Step 3 above.

   Launch the APC Simulation Script by issuing the command, "`APCSimulationTool`", in the MATLAB command window. This script uses the plant's marching script `SimulatePlant.m` and fully-configured APC object to operate. Verify that both items are in the current working folder and in the workspace (or in a pre-saved .mat file), respectively. Refer to Step 3 of Example 1 for information on how to define the plant's marching script. Figure 11 presents a guide for the following steps:

a. The availability of the "`APC`" variable in the workspace is required. If the user continued directly from Step 5, this variable (in its correct form) should already be present in the workspace (variable "`APC`" can be located in the MATLAB workspace). In this case, type "y".



```
 APC Simulation Tool v0.4
--------------------------
This tool provides a SIMULATION platform for the APC Framework.
The high-fidelity simulation ('plant') and APC ('controller') object
need to be setup prior to using this tool. Please exit out of the session
(Ctrl-C), in case the plant 'marching' script or APC object(s) are not
defined earlier. Please refer to documentation, with specific examples, for
information on how to configure them.

Press any key to continue...

1. Controller (APC Framework object) specification:
   The variable named 'APC' must be loaded into workspace with the relevant
   control-configuration. Is the correct APC object already present in workspace? (y/n): y
-- Reading APC_NMPC controller from workspace --

-- Sampling-time is autodetected as 0.005 --

2. Enter the total SIMULATION TIME: 2

3. Enter the agitation time-interval. Typically, this is the plant's process-time / settling time.
   For pH Reactor, this is ~0.25 [hr] and for BFB it is ~300 [s].
   In what interval should be plant be agitated : 0.25

4. Enter the standard deviation (in %) of set-point agitation(s): 5

5. Enter the standard deviation (in %) of disturbance agitation(s): 10

-- The following provides the CONTROLLER I/O PORTS among plant's physical ports --
   (Please note the order in which they appear...)
   Manipulated Inputs (MV) - [2  4]
   Controlled Outputs (CV) - [1  2]

6. Which disturbance(s) you wish to agitate during the simulation, among all plant inputs?
   Note - If this interferes with manipulated input selection, disturbance(s) will be imposed
          on the corresponding input(s) before control-moves are calculated. Provide empty
          vector '[]' for no measured disturbance(s).
   Enter the disturbance(s) index in array format (e.g. [1 3]): [1]

Press any key to start the simulations...

   [-                    ]
   [--                   ]
   [---                  ]
   [----                 ]
```

Figure 11: APC Framework Simulation Tool user interface workflow.

If the workspace was cleared, the variable was modified in any way, or MATLAB was re-launched, type "n" here. The user is prompted to browse the data file. Select the preconfigured APC file "`APC_DABNet_pH.mat`" or any file which the user saved in the previous step. The tool confirms that an APC controller of type "DABNet" was successfully read.

b. Enter “2” in the total simulation time. The sample time of 0.005 [hour] provides enough time for conducting set-point tracking and disturbance-rejection tests.

c. Enter the agitation time-interval. In the present version, the simulation script is hard-coded to randomly modify the disturbance(s) and set-point(s) alternatively in a random step fashion. Close observation of open-loop step-response(s), obtained in Step 4, show that the pH Reactor settles to the new steady-state in an interval close to 0.25 hour (i.e., after 50 sampling-intervals). This defines the process-time or settling time; therefore, enter “0.25” to define the agitation time-interval.

d. The simulation agitates the set-point in a random step fashion. This prompt requires the user to quantify how large of a step set-point change is induced during the simulation. The percentage change is defined on the nominal operating conditions. As an example, the pH1 is maintained in an approximate neutral state of 7. To quantify a random set-point change with a Standard Deviation (SD) of 7±5% to this value (and any other controlled outputs); enter “5” for the nominal operating conditions. A separate agitation magnitude for each CV is not implemented in the script.

e. Enter “10” for the SD of input disturbance(s) value.

f. The user determines the disturbance used to agitate, following the same index convention previously used. The acid flow Q1 is used as the main disturbance. Q1 holds an index 1 among all the plant ports [Q1 Q3 Q5 Q7]; therefore, enter “[1]” for the disturbance value.

   Note: If an index which corresponds to a manipulated input is provided, it will be ignored (overwritten by the controller). To not include any disturbance agitation, enter empty square-brackets [] or click “Enter”.

The simulations runs and the script selects the random agitation to access the controller performance. The MATLAB plots show the progress of simulation with current values of CV and MV. The end of simulation runs provides the total-simulation time. The workspace variable “`APCResponse`” provides additional insight into computational time such as initialization, plant-simulation times, etc. The Total Cumulative Residual (*R*) is provided as shown in Equation (3). This quantity is defined as:

$$R = \sum_{k=1}^{N} \left(\mathbf{r}_k - \mathbf{y}_k\right)^T \mathbf{W}_\mathbf{y} \left(\mathbf{r}_k - \mathbf{y}_k\right) \qquad \textbf{(3)}$$

where, $\mathbf{r}_k$ and $\mathbf{y}_k$ denote the setpoint and plant’s output, $\mathbf{W}_y$ denotes the output weighing matrix. Results, as shown in Figures 12–14, are obtained.

Figure 12 shows the plots of controlled variable(s) (CV) in the top row and manipulated variable(s) (MV) in the bottom row. The red-dotted lines represent the setpoint values. The top two plots correspond to outputs [pH1 pH2] and the bottom two rows represent manipulated inputs [Q3 Q7].

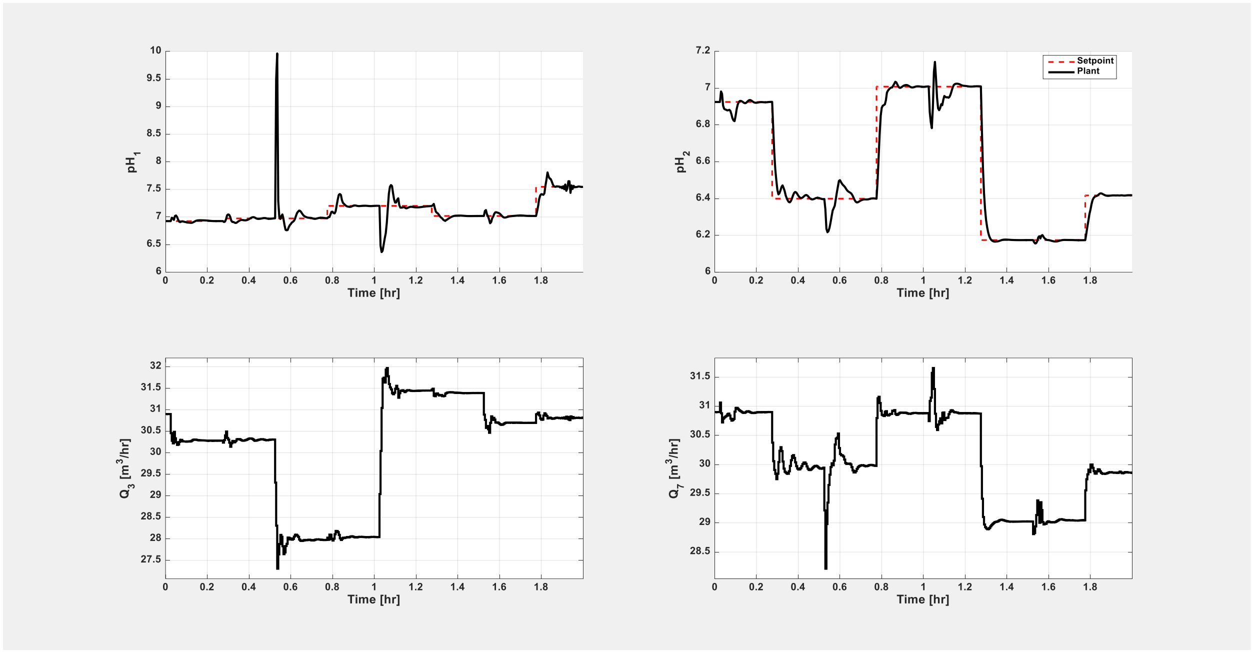


Figure 12: DABNet-NMPC control responses (CV, MV only) for ACM-Based pH Neutralization Reactor example.

Figure 13 provides information on each of the D-RM output(s) along-side the plants outputs (top-row of graphs). In addition, it also shows plots for all the plant's inputs (bottom-rows). The plant has very good response in face of large acid flowrate fluctuations. Typically, this is a difficult problem to address, since not only does the acid-base fluctuations get transferred to the second-tank resulting in a second-third order concentration and volume dynamics; the pH measurement has logarithmic dependence on the concentrations which makes the measurement highly nonlinear to the MVs.

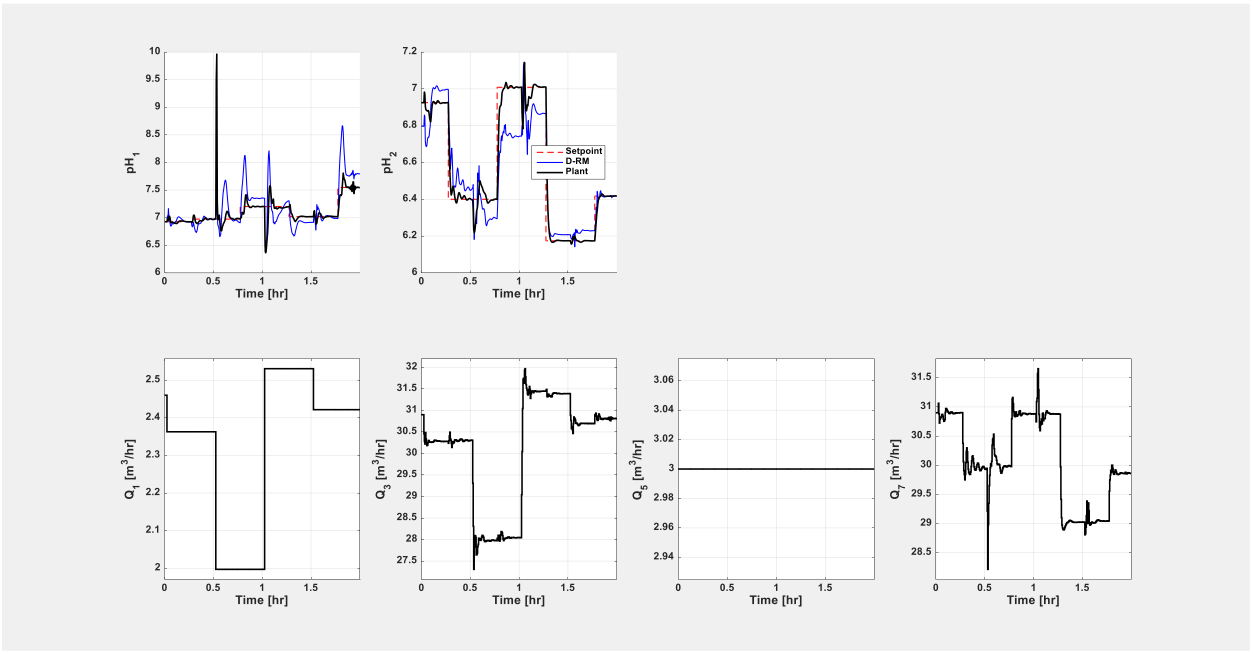


Figure 13: DABNet-NMPC control responses (all) for ACM-Based pH Neutralization Reactor example.

Figure 14 shows the computational cost involved at each time-step. The left axis and the corresponding plot in “blue” represent the CPU-time required for control-calculation at each step, including time to:

- Correct the states, based on new “plant” data, if state-estimation is enabled (not shown in Example 1).
- Solve the constrained nonlinear optimization problem based on set-point (over the entire prediction horizon) and update disturbance input information.
- Forward-march the D-RM one-step with the calculated controlled move.

The right axis with the “green” plot shows the CPU-time needed for “plant” calculations. This involves MATLAB/Simulink ACM communication-time and the time needed by the ACM solver for integrating the pH Neutralization process. **Note:** The first step calculations typically take longer than subsequent ones. This step generally allocates memory, initializes various matrices, and pre-caches workspace variables.

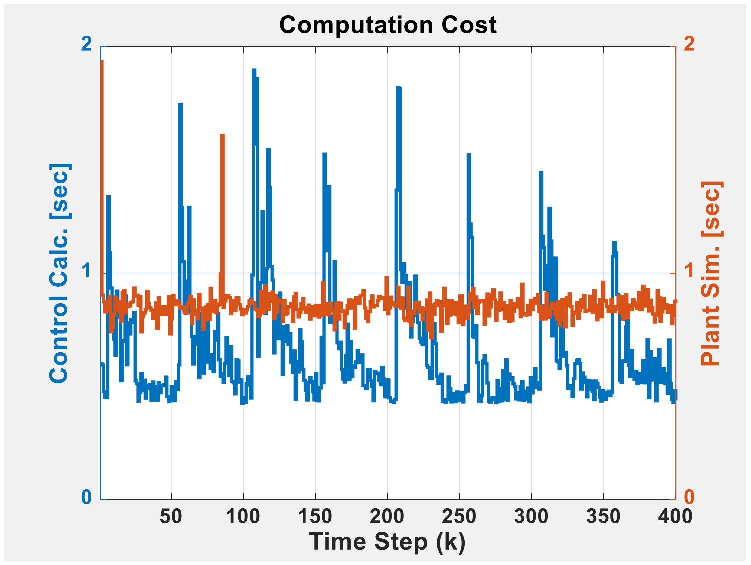


Figure 14: DABNet-NMPC computational cost plots for ACM-Based pH Neutralization Reactor example.

For real-time applicability, it is essential for the controller to return the optimal control-moves before the next sampling-time, i.e., the control calculation at each step should be less than 18 seconds (for Example 1, Ts = 0.005 and hr = 18 sec). The maximum control-calculation time in this case is ~8 seconds (in this particular sequence of random set-point and disturbance changes) which makes it reasonable to use this algorithm for real-time optimal control of this particular example case.

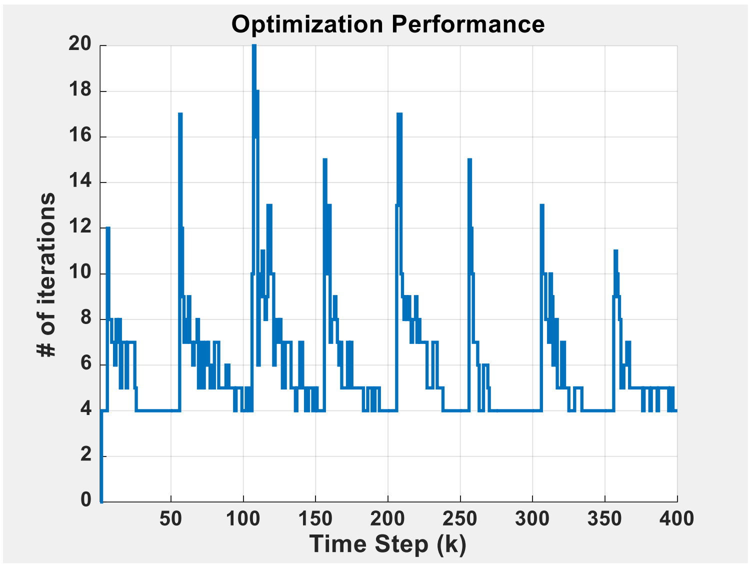


Figure 15: DABNet-NMPC optimization performance for ACM-Based pH Neutralization Reactor example.

Plot (Figure 15) shows the optimization solver performance in terms of number of iterations required to solve the optimal control problem. This technical response plot can be utilized to determine the operating conditions in which APC reaches its solution bottleneck and helps user choose an alternative optimizer.

2.3 Multiple-Model Predictive Control (MMPC) with Additive Output Disturbance

Description

The multiple model predictive control strategy is based on the use of *n* models in the model bank that have the general state-space form given in Equation (4).

$$\begin{aligned} {}^{i}\hat{\mathbf{x}}_{k|k-1} &= {}^{i}\mathbf{\Phi}\,{}^{i}\hat{\mathbf{x}}_{k-1|k-1} + {}^{i}\mathbf{\Gamma}^{u}\mathbf{u}_{k-1} + {}^{i}\mathbf{\Gamma}^{l}\mathbf{l}_{k-1} + {}^{i}\mathbf{\Gamma}^{d}\,{}^{i}\hat{\mathbf{d}}_{k-1|k-1} \\ {}^{i}\hat{\mathbf{y}}_{k|k-1} &= {}^{i}\mathbf{C}\,{}^{i}\hat{\mathbf{x}}_{k|k-1} + {}^{i}\mathbf{D}^{u}\mathbf{u}_{k-1} + {}^{i}\mathbf{D}^{l}\mathbf{l}_{k-1} + {}^{i}\mathbf{D}^{d}\,{}^{i}\hat{\mathbf{d}}_{k-1|k-1} \end{aligned} \tag{4}$$

where, given the current time step *k*, the following notations are used:

$\mathbf{u}_{k-1}$	: manipulated-input vector between time-step *k*-1 and *k*
$\mathbf{l}_{k-1}$	: measured and modeled disturbance vector between time-step *k*-1 and *k*
$\hat{\mathbf{d}}_{k-1\|k-1}$	: unmeasured (estimated) but modeled disturbance vector between time-step *k*-1 and *k* for model *i*
$\hat{\mathbf{x}}_{k\|k-1}$	: process states at time-step *k* given the plant measurements at time-step *k*-1(predicted) for model *i*
$\hat{\mathbf{x}}_{k\|k}$	: process states at time-step *k* given the plant measurements at time-step *k* (corrected) for model *i*
$\hat{\mathbf{y}}_{k\|k-1}$	: measured outputs at time-step *k* given the plant measurements at time-step *k*-1 (predicted) for model *i*
$\hat{\mathbf{y}}_{k\|k}$	: measured outputs at time-step *k* given the plant measurements at time-step *k* (corrected) for model *i*
$\mathbf{\Phi}, \mathbf{\Gamma}^{u}, \mathbf{\Gamma}^{l}, \mathbf{\Gamma}^{d}, \mathbf{C}, \mathbf{D}^{u}, \mathbf{D}^{l}, \mathbf{D}^{d}$	: various state-space parameters for *i*th model

The left superscript "*i*" denotes the model number, with *i* ranging from 1 to *n* models. Although the plant being controlled in practice is highly nonlinear, the models in Equation (4) are all linear. Each linear model is chosen to represent a discrete subspace of the overall nonlinear operating space. When all *n* models are linearly combined, the resulting bank of linear models spans the entire nonlinear operating space. Figure 16 shows that the models within the model bank are updated in parallel with the plant.

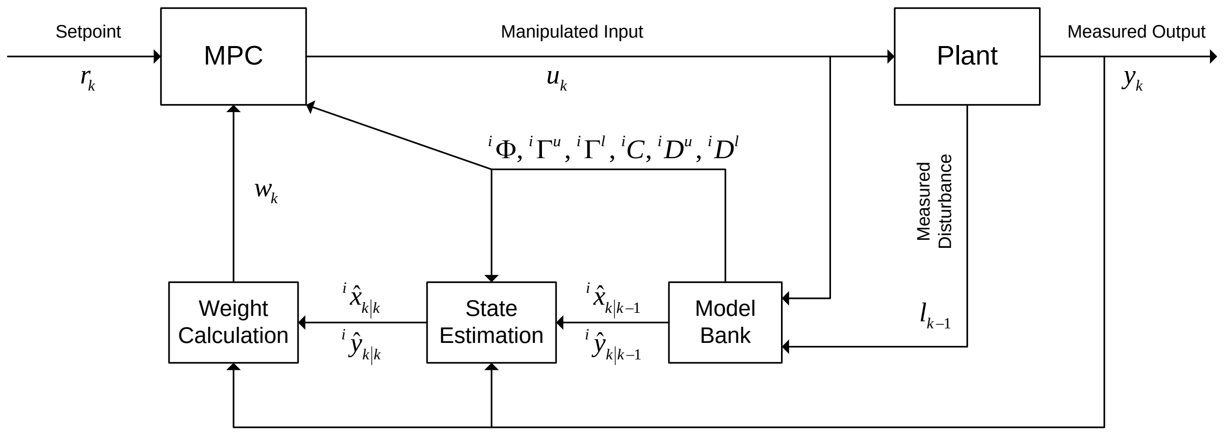


Figure 16: Block diagram showing the generic multiple model predictive control (MMPC) formulation.

The state-estimation involves a correction step, which modifies/corrects the plant states based on plant measurements at current time-step. For achieving this, an appended state formulation is used. The theoretical details of these approaches are given in Kurre-Kinsey and Bequette, 2009 and 2010 [3,4]. The application of MMPC approach an ACM-based Air Separation Unit (ASU) plant is provided in Mahapatra and Zitney, 2012 [5].

Detailed descriptions of individual commands and screenshots are provided. It is highly recommend that this tutorial is read in its entirety before using the tool for a specific APC application.

Example 2 – MMPC for MATLAB ODE-Based Plant Model (pH Neutralization Reactor)

To demonstrate the MMPC feature of the APC Framework for “plants” based on MATLAB ODE solvers, a pH Neutralization Reactor example has been provided in the “`<APCBaseFolder>\Examples\pH_Reactor_MATLAB`” subfolder. This nonlinear reactor model is a benchmark process popular in control literature. See Reference [1] for more details on the reactor model. This detailed ordinary differential equation (ODE) model is the high-fidelity “plant” model on which the framework is tested. The following steps enumerate the workflow involved in setting up a detailed control simulation, in particular, to MATLAB ODE-based “plant” models.

1. Copy the entire example subfolder of pH Neutralization Reactor “`<APCBaseFolder>\Examples\pH_Reactor_MATLAB`” to a location determined by the user. The user requires write-permissions to this folder. Then browse to the “`pH_Reactor_MATLAB`” subfolder.
2. Open MATLAB (if not already done) and then browse to the newly copied “`pH_Reactor_MATLAB`” subfolder. This becomes the *working directory* in MATLAB. Double-click the `pH_Reactor.m` code in MATLAB or issue the command, “`edit pH_Reactor.m`”, to learn more about the process being used in Example 2.

   Note: Plant inputs correspond to acid and base flowrates to each tank given by Q1, Q3 for Tank 1 and Q5, Q7 for Tank 2, respectively. The outputs correspond to pH1 and pH2 which denote the pH values in corresponding tanks.
3. The following steps instantiate the reactor object and initialize the plant at a certain condition. To skip these steps, a quick-initialize MATLAB script is provided with the filename “`Initialize_pH_Reactor.m`”. The user can open the file in MATLAB editor and then “Run” it or issue the command “`Initialize_pH_Reactor`” at the MATLAB command window.
   a. Define the plant by issuing the command, “`pH_Neut_Plant = pH_Reactor;`” in the command window for the pH Neutralization Reactor.
   b. Set an initial value of the flowrates (inputs) Q1, Q3, Q5, and Q7 as 2.46 m^3/hr, 30.9 m^3/hr, 3 m^3/hr, and 30.9 m^3/hr, respectively. These figures are taken directly from the steady-state conditions detailed in Reference [1].

      “`u_plant = [2.46; 30.9; 3.0; 30.9];`”
   c. Initialize the plant at these inputs by issuing the command, “`pH_Neut_Plant.initialize(u_plant);`”

      Verify the pH in either reactor for the above values of acid and base flowrates, by issuing the command, “`pH_Neut_Plant.pH1`” or “`pH_Neut_Plant.pH2`” in the command window, both should be 6.9251. Verify the output y, which is a vector of both outputs using “`pH_Neut_Plant.y`”.

d. Extract the nominal outputs from the plant using "`y_plant = pH_Neut_Plant.y;`"

Simulation of the plant is completed using a marching script named, "`SimulatePlant.m`". Navigate to this file in the *working directory* and then open it to view the code by double-clicking it within MATLAB or issuing the command, "`edit SimulatePlant.m`". The `pH_Neut_Plant.evalNextStep(u_plant,Ts_plant)` can calculate the next discretized results/outputs (`y_plant`) at the next time-step (after `Ts_plant` units), given any set of inputs `u_plant`.

4. Now that the "plant" is defined, the user builds the control-models or D-RMs for this plant. The APC Framework's MMPC algorithm uses multiple control-models linearized at different "plant" operating conditions. The following text demonstrates how the control-models or D-RMs are built within the scope of the framework.

To build D-RMs around a specific operating point, a time-series of input-output data needs generated from the "plant", this is also known as dynamic step-responses. Specify the desired input-outputs wants to model in each D-RM. The APC Framework's MMPC algorithm can handle multiple D-RMs that may have different sets of input-output variables. The plant's operating condition around which each D-RM is to be generated is specified later in this manual.

To generate step-responses, navigate to the working folder and issue the command "`GetStepTestData`" in the MATLAB command window to run the file "`GetStepTestData.m`". The "Step-Response Data Generator" script displays.

Note: To skip these steps, browse to the preloaded data file provided in the "`Step_Tests`" subfolder within MATLAB and then load the results directly into the workspace by double-clicking "`pHStepResponseData.mat`" or issuing the command "`load('pHStepResponse.mat')`" in the MATLAB command window. Remember to go back to the parent directory (the user's working folder) after completing this action. Skip to the *[Redirect here if using pre-built "`pHStepResponseData.mat`" file]* section under Step 4 after the step-response generation portion.

a. Specify the number of training-sets to be generated. This depends on the number of control-model(s) the user wants to define within the model-bank. Having more models will help span more operational regimes but may slow down control-computation. It is highly recommended that the user set the number of control-models to the minimal amount necessary for capturing nonlinearity adequately. For Example 2, 3 models are used in the bank.

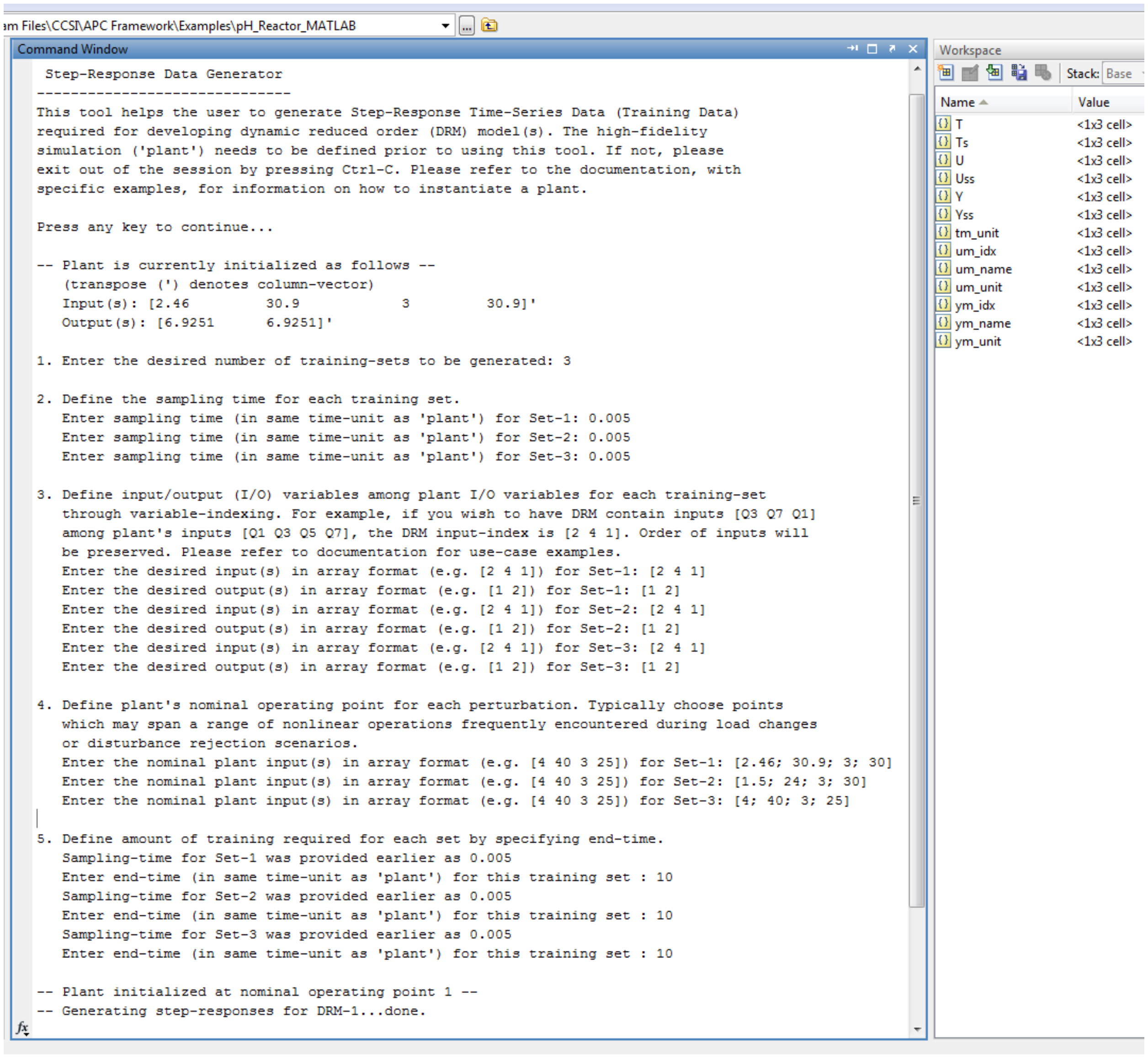
b. Enter "0.005" as the sampling time for each training set. **Note:** The time-unit corresponding to the plant is hr. See the detailed rationale for selecting 0.005 as the sampling time in Example 1. Since it is intended to use all of these models in a model-bank for APC, using a common sampling interval is strongly advised, which will avoid issues at controller configuration steps.

c. Select the inputs to model in the D-RM(s). A close look at physical pH Reactor shows that the control objective is to control pH in both the reactors by manipulating the base flow-rates to both the tanks. The acid inflow stream flowrate to reactor-1 serves as the measured disturbance to the process. Therefore, flowrates of manipulated inputs (Q3 and Q7) and disturbance (Q1) into the user's D-RMs need to be considered. **Note:** The flowrate of acid-stream to reactor 2 (Q5) is considered an unmeasured disturbance. Since this flowrate cannot be measured, any time-response data corresponding to this variable cannot be extracted; therefore, cannot be modeled in the D-RMs. Manipulated inputs should be lined-up first for brevity sake. The plant is modeled to have the inputs ports listed in order as [Q1 Q3 Q5 Q7]. If [Q3 Q7 Q1] are selected as inputs for proposed D-RMs, type "[2 4 1]" as the corresponding indices. Both plant outputs [pH1 pH2] need to be controlled and modeled in the D-RM. Enter "[1 2]" as indices for outputs. Enter the same inputs/outputs for all second and third D-RM as well.

d. Specify the operating points around which perturbations are completed for generating step-responses. Since the current example involves a mathematical DAE model in MATLAB with an initialization (steady-state solver) script defined, it is trivial to have the plant initialized at these points (in backend, the script invokes `pH_Neut_Plant.initialize(nominal_input)`) and perform small perturbations. Enter the current nominal value "[2.46; 30.9; 3; 30]", one with a low acid flowrate "[1.5; 24; 3; 30]", and another with a high acid-flowrate "[4; 40; 3; 25]". Different values of acid-flowrate address the range of load-change operation in the system. The base-flowrates are decided heuristically so that the approximate pH in each tank remains close to the original value (~7). For example, for low acid-flowrate (Q1), the base-flowrate is decreased to reactor-1 (Q3) to avoid drastic changes in pH1. During close-loop operation, where pH1 holds the value (at ~7), each of the model predictions remain as close to this "held" value as possible. As a counter justification, if the plant operates very rarely at very low/high pH (once the loop is closed), there is no significant reason of having a D-RM in the model bank which was linearized around low/high pH operating point.

e. Define the amount of training required for each set by specifying the time of simulation (in the same time-units as "plant"). Divide this by the sampling time to determine the number of samples involved in the corresponding run. The more data provided, the better the training. Training is done once offline. **Note:** A good prediction model is at hand when the control-loop is closed. A large simulation time of 10 (hrs.) is provided for each training-set, resulting in 2,000 samples.

Testing this large number of samples requires a couple of minutes before the step-response data is generated. The entire workflow for Example 2, as it should display on the MATLAB command window, is shown in Figure 17. Various post-simulation variables may display in the user's workspace; if not visible, type "`whos`" in the MATLAB command window.



```
 Step-Response Data Generator
------------------------------
This tool helps the user to generate Step-Response Time-Series Data (Training Data)
required for developing dynamic reduced order (DRM) model(s). The high-fidelity
simulation ('plant') needs to be defined prior to using this tool. If not, please
exit out of the session by pressing Ctrl-C. Please refer to the documentation, with
specific examples, for information on how to instantiate a plant.

Press any key to continue...

-- Plant is currently initialized as follows --
   (transpose (') denotes column-vector)
   Input(s): [2.46        30.9          3         30.9]'
   Output(s): [6.9251     6.9251]'

1. Enter the desired number of training-sets to be generated: 3

2. Define the sampling time for each training set.
   Enter sampling time (in same time-unit as 'plant') for Set-1: 0.005
   Enter sampling time (in same time-unit as 'plant') for Set-2: 0.005
   Enter sampling time (in same time-unit as 'plant') for Set-3: 0.005

3. Define input/output (I/O) variables among plant I/O variables for each training-set
   through variable-indexing. For example, if you wish to have DRM contain inputs [Q3 Q7 Q1]
   among plant's inputs [Q1 Q3 Q5 Q7], the DRM input-index is [2 4 1]. Order of inputs will
   be preserved. Please refer to documentation for use-case examples.
   Enter the desired input(s) in array format (e.g. [2 4 1]) for Set-1: [2 4 1]
   Enter the desired output(s) in array format (e.g. [1 2]) for Set-1: [1 2]
   Enter the desired input(s) in array format (e.g. [2 4 1]) for Set-2: [2 4 1]
   Enter the desired output(s) in array format (e.g. [1 2]) for Set-2: [1 2]
   Enter the desired input(s) in array format (e.g. [2 4 1]) for Set-3: [2 4 1]
   Enter the desired output(s) in array format (e.g. [1 2]) for Set-3: [1 2]

4. Define plant's nominal operating point for each perturbation. Typically choose points
   which may span a range of nonlinear operations frequently encountered during load changes
   or disturbance rejection scenarios.
   Enter the nominal plant input(s) in array format (e.g. [4 40 3 25]) for Set-1: [2.46; 30.9; 3; 30]
   Enter the nominal plant input(s) in array format (e.g. [4 40 3 25]) for Set-2: [1.5; 24; 3; 30]
   Enter the nominal plant input(s) in array format (e.g. [4 40 3 25]) for Set-3: [4; 40; 3; 25]

5. Define amount of training required for each set by specifying end-time.
   Sampling-time for Set-1 was provided earlier as 0.005
   Enter end-time (in same time-unit as 'plant') for this training set : 10
   Sampling-time for Set-2 was provided earlier as 0.005
   Enter end-time (in same time-unit as 'plant') for this training set : 10
   Sampling-time for Set-3 was provided earlier as 0.005
   Enter end-time (in same time-unit as 'plant') for this training set : 10

-- Plant initialized at nominal operating point 1 --
-- Generating step-responses for DRM-1...done.
```

Name	Value
T	<1x3 cell>
Ts	<1x3 cell>
U	<1x3 cell>
Uss	<1x3 cell>
Y	<1x3 cell>
Yss	<1x3 cell>
tm_unit	<1x3 cell>
um_idx	<1x3 cell>
um_name	<1x3 cell>
um_unit	<1x3 cell>
ym_idx	<1x3 cell>
ym_name	<1x3 cell>
ym_unit	<1x3 cell>

Figure 17: Example workflow for step-response data generator.

Many variables are available as type "`1x3 cell`" which means that each of such variable contains responses for all three models, one in every cell. For instance, for accessing output data [pH1 pH2] for 3rd model at 450th sampling-time, issue "`Y{3}(:,450)`" at the command window, where ':' denotes both the outputs as column-vector. `Uss` and `Yss` provide the nominal values for each set. Variables `um_idx` and `ym_idx` are the I/O indices, as mentioned in Step c, linking the D-RM to the plant. Additionally, variable names (`um_name`, `ym_name`) and units (`um_unit`, `ym_unit` and `tm_unit`) are automatically extracted from plant simulations using each model indices.

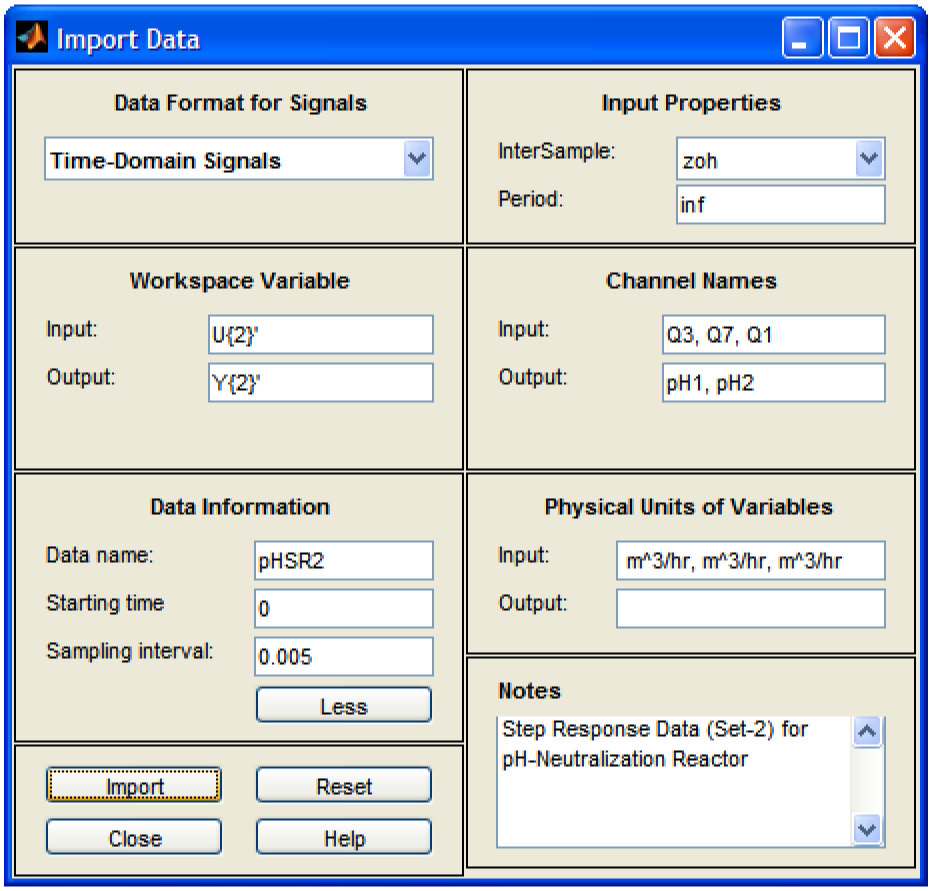
[Redirect here if using pre-built "`pHStepResponseData.mat`" file]

The previously generated step-responses are utilized to obtain state-space based D-RMs. This requires knowledge and use of the MATLAB System Identification Toolbox and are not covered in-depth here. The following provides screenshots and a quick guide for doing the same. If the user does not have a license for the System Identification Toolbox, see the **Note** below. If the user closed MATLAB or cleared the workspace, refer to the "**Note**" in Step 4, regarding loading the "`pHStepResponseData.mat`" file.

Note: Typically, the methodology for generating the D-RM model bank is determined by the user, as long as the final D-RM(s) obtained are compatible with the APC Framework's D-RM class-structure. This section gives guidance on one of the methodologies. As a shortcut, preloaded D-RM objects containing the three control-model(s) discussed in previous steps are available in the MATLAB data-file "`DRM_SSBank_pH.mat`". The results can be loaded directly into the workspace by double-clicking this file within MATLAB or by issuing "`load('DRM_SSBank_pH.mat')`" in the MATLAB command window; proceed to Step 6.

a. Browse to the "`Step-Test`" subfolder and then load the identified model into the workspace either by double-clicking the "`pH_IdentData.mat`" file inside MATLAB or by issuing the command "`load ('pH_IdentData.mat')`". The variables `A, B, C, D, K, x0` (each of type `1x3 cell`) contains the fitted state-space information for the three training-sets. The nominal values, I/O indices, variable names, and units have been carried forward from earlier. Navigate back to the parent directory (working folder). The following describes details on how the identification was performed. To skip this step, advance to Step b.

   Launch the System Identification GUI by issuing the command "`ident`". Import time-domain data using Figure 18 as a guide.



Data Format for Signals

Time-Domain Signals

Input Properties

InterSample: zoh

Period: inf

Workspace Variable

Input: U{2}'

Output: Y{2}'

Channel Names

Input: Q3, Q7, Q1

Output: pH1, pH2

Data Information

Data name: pHSR2

Starting time 0

Sampling interval: 0.005

Physical Units of Variables

Input: m^3/hr, m^3/hr, m^3/hr

Output:

Notes

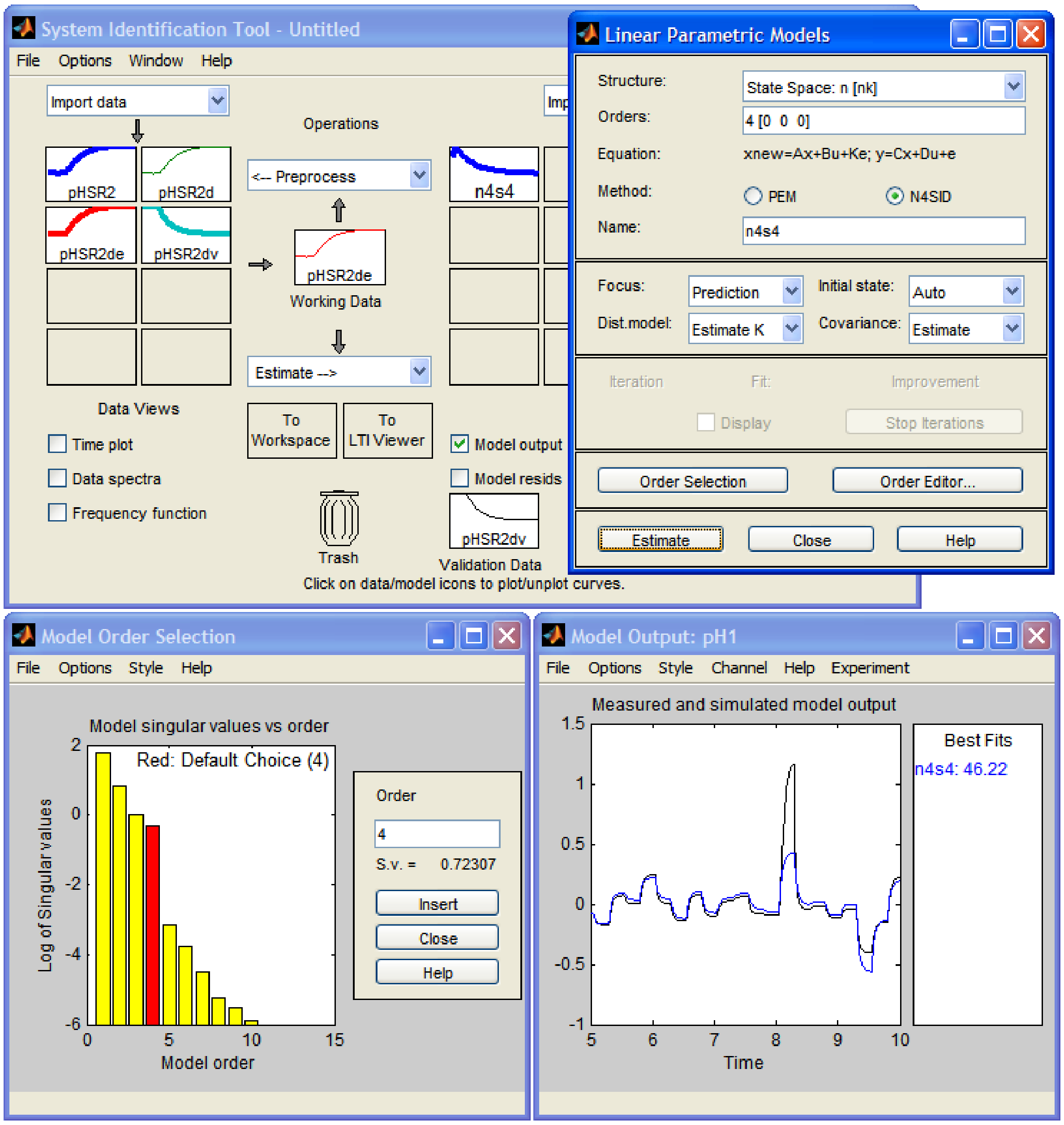
Step Response Data (Set-2) for pH-Neutralization Reactor

Figure 18: "ident" Import Data window showing import of set-2 step-response data.

MATLAB made significant improvements in the identification interface over the past few years. The interface shown here (in R2011b) might be different from what is displayed for the user. To summarize, the entire workflow is similar to the following:

- Import the time-series data for each set, as shown in Figure 18.
- Preprocess data to transform it into its deviation form.
- Separate the data into estimation and validation sets. "ident" made all the steps hassle-free for the user. Single-click "Quick Start" at the "Preprocess" stage.
- Select Linear Parametric Model/State-Space Model for estimation.
- Specify the order using order-dependent singular value analysis; "ident" suggests an order.

When the fitted D-RM is used as control-model(s) for accurate prediction, it is pertinent to keep the focus setting to “Prediction”. In addition, “ident” estimates K (disturbance model) which is used in disturbance estimation. With all the relevant settings in place, click “Estimate” to run the SS parameter estimation routine. “ident” provides the fitted model and shows the quality of fit. Figure 19 provides guidance for most of these steps.



Structure: State Space: n [nk]
Orders: 4 [0 0 0]
Equation: xnew=Ax+Bu+Ke; y=Cx+Du+e
Method: PEM [✓ N4SID]
Name: n4s4
Focus: Prediction
Initial state: Auto
Dist.model: Estimate K
Covariance: Estimate
[✓ Model output]
Order: 4
S.v. = 0.72307
n4s4: 46.22

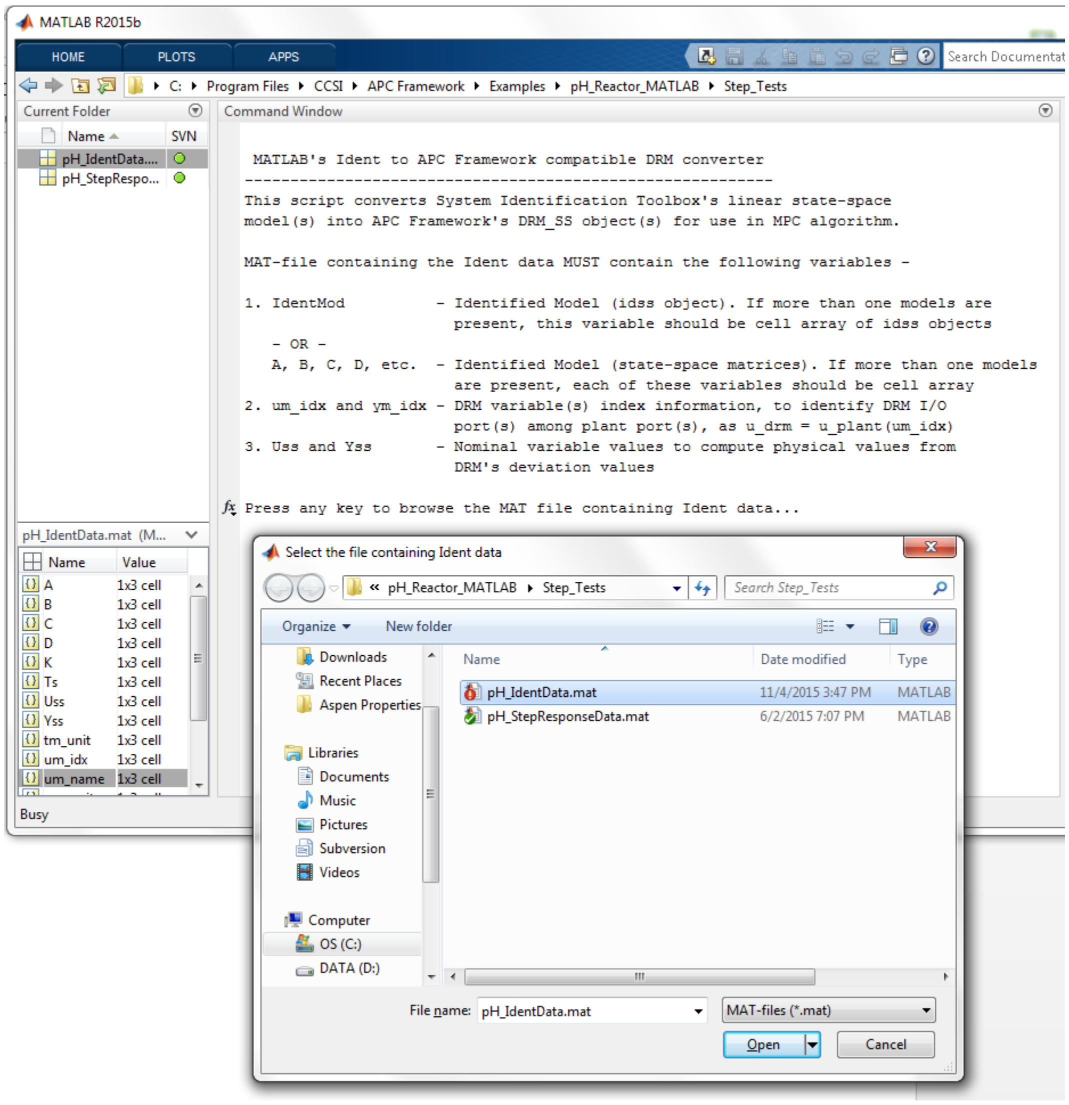
Figure 19: “ident” State-Space Model configuration windows showing typical settings and workflow.

The model can be transferred to the workspace by dragging the fitted model to the "To Workspace" box and thereafter to the `IdentMod` variable. As an example for 2nd identified model in Figures 18 and 19, the MATLAB command will be:

```
IdentMod{2} = n4s4;
```

For cross-compatibility between different MATLAB versions, a tool "`Ident2SS`" has been provided which extracts state-space variables from the System Identification Toolbox specific objects and stores them in a more readable, cross-version format. For this issue the command "`Ident2SS`" in the MATLAB command window and point to the "`IdentMod`" variable.

b. Launch the APC Framework's "ident" to the D-RM converter script by issuing the command "`Ident2DRM`" in the MATLAB command window. For .MAT files containing the identified data, browse to the "Step-Tests" subfolder and then select the "`pH_IdentData.mat`" file, as shown in Figure 20.



```
 MATLAB's Ident to APC Framework compatible DRM converter
 ----------------------------------------------------------
This script converts System Identification Toolbox's linear state-space
model(s) into APC Framework's DRM_SS object(s) for use in MPC algorithm.

MAT-file containing the Ident data MUST contain the following variables -

1. IdentMod          - Identified Model (idss object). If more than one models are
                       present, this variable should be cell array of idss objects
   - OR -
   A, B, C, D, etc.  - Identified Model (state-space matrices). If more than one models
                       are present, each of these variables should be cell array
2. um_idx and ym_idx - DRM variable(s) index information, to identify DRM I/O
                       port(s) among plant port(s), as u_drm = u_plant(um_idx)
3. Uss and Yss       - Nominal variable values to compute physical values from
                       DRM's deviation values

fx Press any key to browse the MAT file containing Ident data...
```

Name	Value
A	1x3 cell
B	1x3 cell
C	1x3 cell
D	1x3 cell
K	1x3 cell
Ts	1x3 cell
Uss	1x3 cell
Yss	1x3 cell
tm_unit	1x3 cell
um_idx	1x3 cell
um_name	1x3 cell

Name	Date modified	Type
pH_IdentData.mat	11/4/2015 3:47 PM	MATLAB
pH_StepResponseData.mat	6/2/2015 7:07 PM	MATLAB

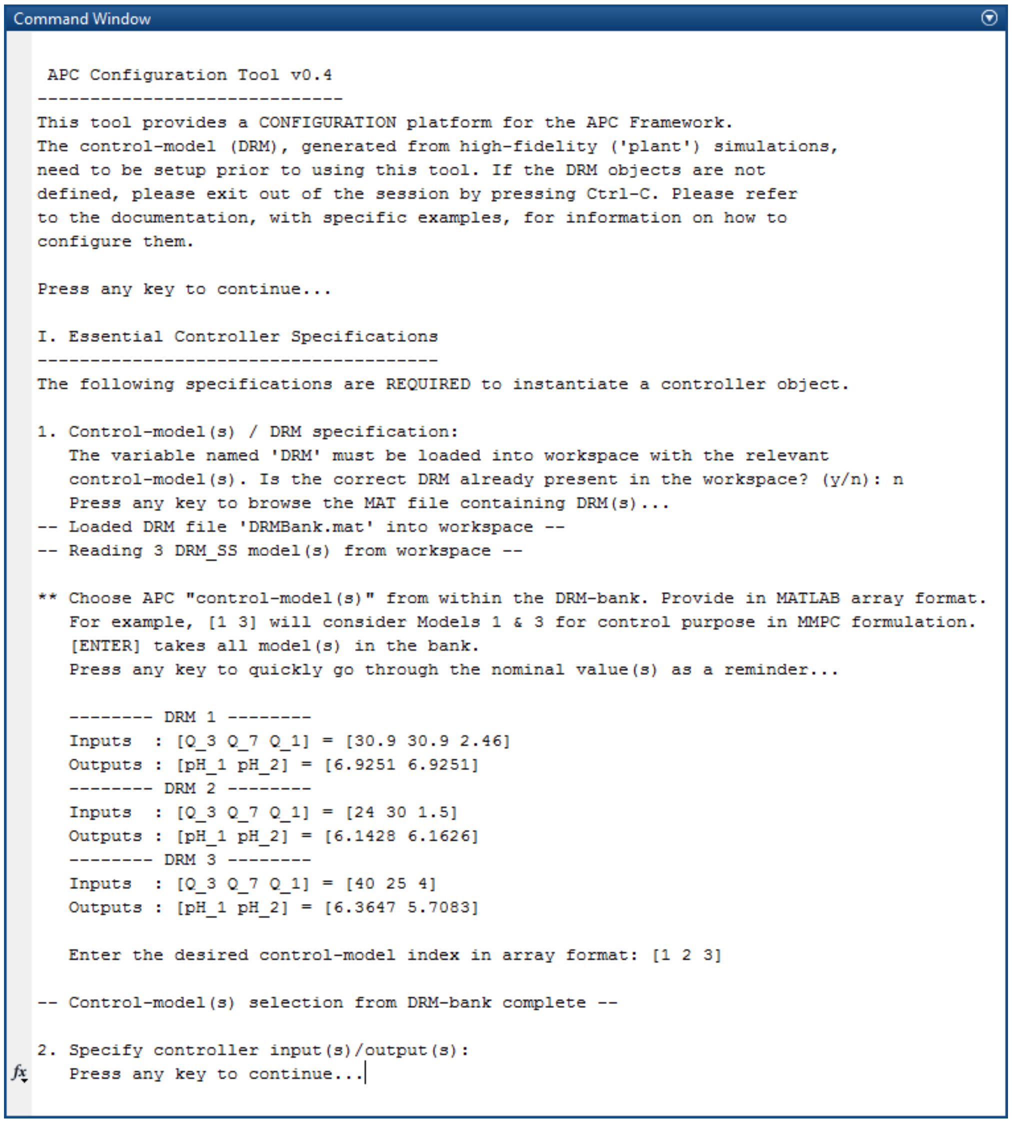
File name: pH_IdentData.mat — MAT-files (*.mat)

Figure 20: APC Framework's Ident2DRM Browse window to select MATLAB "ident" data for conversion to APC Framework-Friendly D-RM.

Note: After the conversion is complete, a workspace variable named, "`DRM`", is created. Save this variable as a MATLAB data file, by right-clicking this variable in the workspace and then selecting "Save As". A pre-loaded file "`DRM_SSBank_pH.mat`" is already created, which can be overwritten for consistency with the MATLAB version. **Note:** This file can be named anything, although it is referred to as the D-RM data file, model-bank, and "`DRM_SSBank_pH.mat`" interchangeably for the remainder of this manual.

5. In this step, an APC object is configured for use with the data-bank. Launch the APC Configuration Script by issuing the command "`APCConfigTool`". To skip this step, refer to the preconfigured APC object "`APC_MMPC_pH.mat`" in the working folder.

    a. If the relevant "`DRM`" variable (upon continuing from the previous step) is present in the workspace, type "y", if not type "n", and then browse to "`DRM_SSBank_pH.mat`". The tool confirms that three D-RMs of type state-space (SS) were successfully read, as shown in Figure 21.

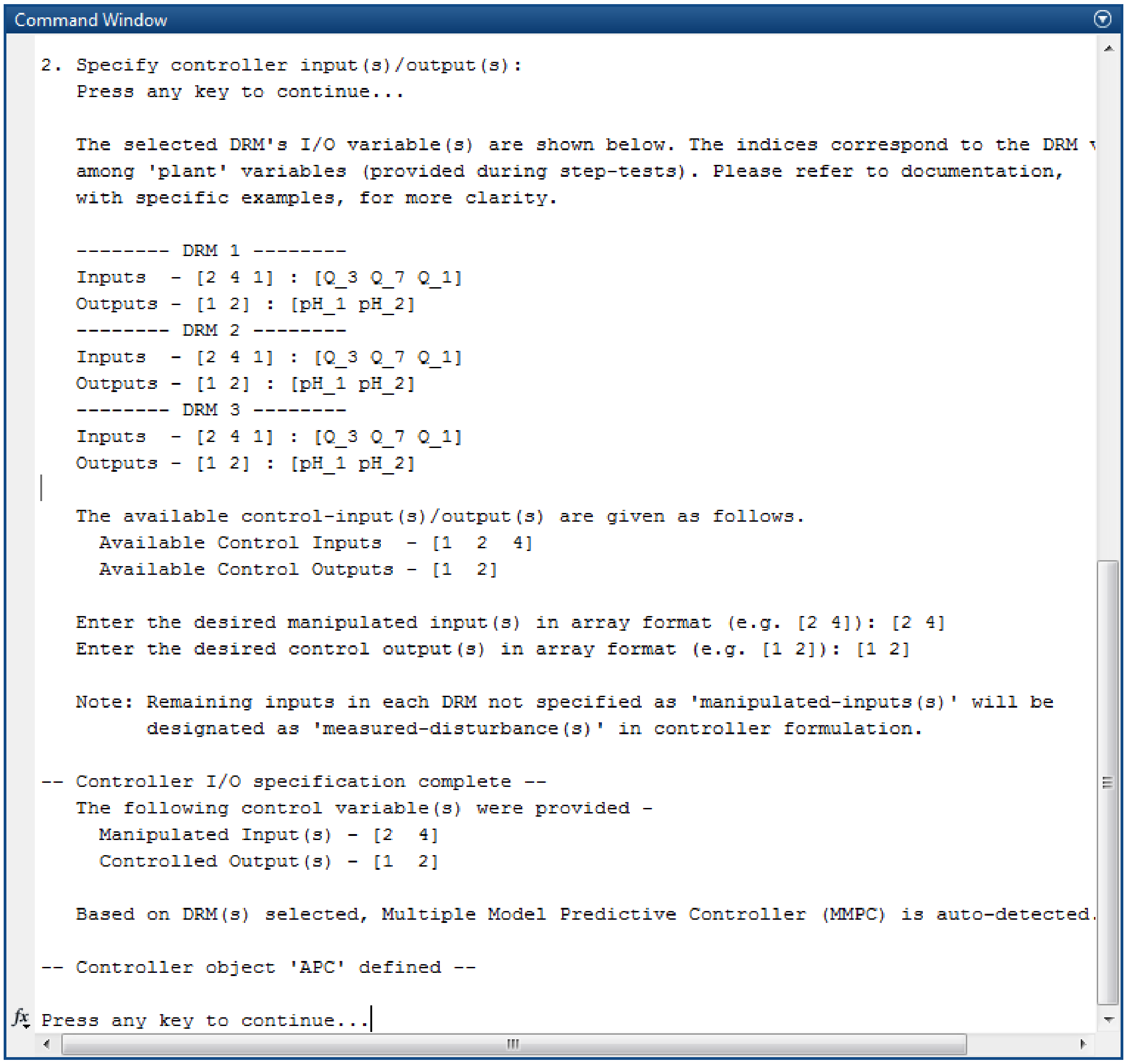

Command Window

```
 APC Configuration Tool v0.4
-----------------------------
This tool provides a CONFIGURATION platform for the APC Framework.
The control-model (DRM), generated from high-fidelity ('plant') simulations,
need to be setup prior to using this tool. If the DRM objects are not
defined, please exit out of the session by pressing Ctrl-C. Please refer
to the documentation, with specific examples, for information on how to
configure them.

Press any key to continue...

I. Essential Controller Specifications
--------------------------------------
The following specifications are REQUIRED to instantiate a controller object.

1. Control-model(s) / DRM specification:
   The variable named 'DRM' must be loaded into workspace with the relevant
   control-model(s). Is the correct DRM already present in the workspace? (y/n): n
   Press any key to browse the MAT file containing DRM(s)...
-- Loaded DRM file 'DRMBank.mat' into workspace --
-- Reading 3 DRM_SS model(s) from workspace --

** Choose APC "control-model(s)" from within the DRM-bank. Provide in MATLAB array format.
   For example, [1 3] will consider Models 1 & 3 for control purpose in MMPC formulation.
   [ENTER] takes all model(s) in the bank.
   Press any key to quickly go through the nominal value(s) as a reminder...

   -------- DRM 1 --------
   Inputs  : [Q_3 Q_7 Q_1] = [30.9 30.9 2.46]
   Outputs : [pH_1 pH_2] = [6.9251 6.9251]
   -------- DRM 2 --------
   Inputs  : [Q_3 Q_7 Q_1] = [24 30 1.5]
   Outputs : [pH_1 pH_2] = [6.1428 6.1626]
   -------- DRM 3 --------
   Inputs  : [Q_3 Q_7 Q_1] = [40 25 4]
   Outputs : [pH_1 pH_2] = [6.3647 5.7083]

   Enter the desired control-model index in array format: [1 2 3]

-- Control-model(s) selection from DRM-bank complete --

2. Specify controller input(s)/output(s):
   Press any key to continue...
```

Figure 21: APC Framework's Configuration Tool specification workflow.

b. The control-model(s) have to be chosen from within the ones available in the D-RM bank. Ideally, use all D-RM model(s) to span as much nonlinearity as possible. Select all models by entering “[1 2 3]” at the prompt or press “Enter”.

c. I/O variables, as provided in Step 4.c, are listed for each of the D-RM. The available control-inputs (i.e., the input ports which are common to all D-RMs) are displayed, among which the manipulated variable(s) are to be chosen. If a manipulated input is not modeled in all of the D-RMs, it cannot be used to setup the control objective function since the prediction(s) from the un-modeled D-RM is not available for this input; otherwise, the manipulated inputs in a “plant” are well-defined. For implementing a model-based control, the control-model/D-RMs should adequately capture or “model” all of these inputs; otherwise, the D-RM is of no use. It is the responsibility of the control-engineer tasked with implementing model-based control to model all the manipulated-inputs, in addition to as many measured disturbance as possible to keep the D-RM tractable.



```
2. Specify controller input(s)/output(s):
   Press any key to continue...

   The selected DRM's I/O variable(s) are shown below. The indices correspond to the DRM v
   among 'plant' variables (provided during step-tests). Please refer to documentation,
   with specific examples, for more clarity.

   -------- DRM 1 --------
   Inputs  - [2 4 1] : [Q_3 Q_7 Q_1]
   Outputs - [1 2] : [pH_1 pH_2]
   -------- DRM 2 --------
   Inputs  - [2 4 1] : [Q_3 Q_7 Q_1]
   Outputs - [1 2] : [pH_1 pH_2]
   -------- DRM 3 --------
   Inputs  - [2 4 1] : [Q_3 Q_7 Q_1]
   Outputs - [1 2] : [pH_1 pH_2]

   The available control-input(s)/output(s) are given as follows.
     Available Control Inputs  - [1  2  4]
     Available Control Outputs - [1  2]

   Enter the desired manipulated input(s) in array format (e.g. [2 4]): [2 4]
   Enter the desired control output(s) in array format (e.g. [1 2]): [1 2]

   Note: Remaining inputs in each DRM not specified as 'manipulated-inputs(s)' will be
         designated as 'measured-disturbance(s)' in controller formulation.

-- Controller I/O specification complete --
   The following control variable(s) were provided -
     Manipulated Input(s) - [2  4]
     Controlled Output(s) - [1  2]

   Based on DRM(s) selected, Multiple Model Predictive Controller (MMPC) is auto-detected.

-- Controller object 'APC' defined --

fx Press any key to continue...
```

Figure 22: APC Configuration Tool workflow for specification of D-RM inputs/outputs.

Enter “[2 4]” as the desired manipulated input corresponding to base flowrates, Q3 and Q7. Similarly, enter “[1 2]” as desired control outputs corresponding to pH of both tanks. Refer to Figure 22 for guidance. The controller object “APC” is now defined. Additional controller setup and initialization steps are required before online-control can be implemented.

d. In the next window, after pressing a key to continue, additional controller specifications may be completed. In addition to the regular controller parameters, mentioned in the previous NMPC section, the Model Covariance Coefficient (λ) **must** be specified to fully define the MMPC controller object. Type “n” to exit. A pre-built script named, “`SetupDemoAPCPars_pH.m`”, is provided in the working folder, which when run automatically updates the parameters into the controller. Run this file by opening it in the MATLAB editor and then selecting “Run” or issuing the command “`SetupDemoAPCPars_pH`” at the MATLAB command window.

   Right-click the “`APC`” variable in the workspace, select “Save As”, and then provide an appropriate filename determined by the user, or overwrite the “`APC_MMPC_pH.mat`” file.

6. Simulating the pH Neutralization Reactor “plant” with control implemented or in “closed-loop” mode is detailed in this section. As in a real-plant scenario, the pH Reactor “plant” must be operating in some nominal state. The input-output information needs to be made available to the controller. The controller-object can then, with this information, initialize the embedded D-RMs within it.

   The plant initialization, and making the initial plant values available to the controller, is the responsibility of the plant operators or control engineers; therefore, this is kept separate from the Simulation Tool. For Example 2, open the file “`Initialize_pH_Reactor.m`” in MATLAB editor and then “Run” it or issue the command, “`Initialize_pH_Reactor`,” at the MATLAB command window.

   Launch the APC Simulation Script by issuing the command “`APCSimulationTool`” in the MATLAB command window. This script uses the plant’s marching script `SimulatePlant.m` and fully-configured APC object to operate. Verify that both items are in the current working folder and in the workspace, respectively. Refer to Figure 22 as a guide to help with the entries required in the tool. **Note:** There might be slight differences in the user’s display and what is shown in Figure 22.

   a. The availability of the “`APC`” variable in the workspace is required. If the user continued directly from Step 6, this variable (in its correct form) should already be present in the workspace (variable “`APC`” should be located in the MATLAB workspace), if so type “y”.

   b. If the workspace was cleared, the variable was modified in any way, or MATLAB was re-launched, type “n” and then browse the data file. Select the preconfigured APC file “`APC_MMPC_pH.mat`” or any file which was saved in the previous step. The tool confirms that an APC controller of type MMPC was successfully read.

   c. Enter “2” in the total simulation time. The sample time of 0.005 [hr] allows enough time for conducting set-point tracking and disturbance-rejection tests.

d. Enter the agitation time-interval. In the present version, the simulation script is hard-coded to randomly modify the disturbance(s) and set-point(s) alternatively in a random step fashion. Close observation of open-loop step-response(s), obtained in Step 4, show that the pH Reactor settles to the new steady-state in an interval close to 0.25 hr. (i.e., after 50 sampling-intervals). This defines the process-time or settling time; therefore, enter “0.25” to define the agitation time-interval.

e. The simulation agitates the set-point in a random step fashion. This prompt requires the user to quantify how large of a step set-point change is induced during the simulation. The percentage change is defined on the nominal operating conditions. As an example, the pH1 is maintained in an approximate neutral state of 7. To quantify a random set-point change with a Standard Deviation (SD) of 7±5% to this value (and any other controlled outputs), enter “5” for the nominal operating conditions. A separate agitation magnitude for each CV is not implemented in the script.

f. Enter “10” for the SD of the input disturbance(s) value.

g. The user determines the disturbance used to agitate, following the same index convention previously used. The acid flow Q1 is used as the main disturbance. Q1 holds an index 1 among all the plant ports [Q1 Q3 Q5 Q7], therefore enter “[1]” as the disturbance value.

   Note: If an index which corresponds to a manipulated input is provided, it will be ignored (overwritten by the controller). To not include any disturbance agitation, enter empty square-brackets “[]” or simply click “Enter”.

The simulations run and the script selects the random agitation to access the controller performance. The MATLAB plots show the progress of simulation with current values of CV and MV. Results, as shown in Figures 23–26, are obtained.

Figure 23 shows the plots of controlled variable(s) (CV) in the top row and manipulated variable(s) (MV) in the bottom row. The red-dotted lines represent the setpoint values. The top two plots correspond to outputs [pH1 pH2] and the bottom two rows represent manipulated inputs [Q3 Q7].

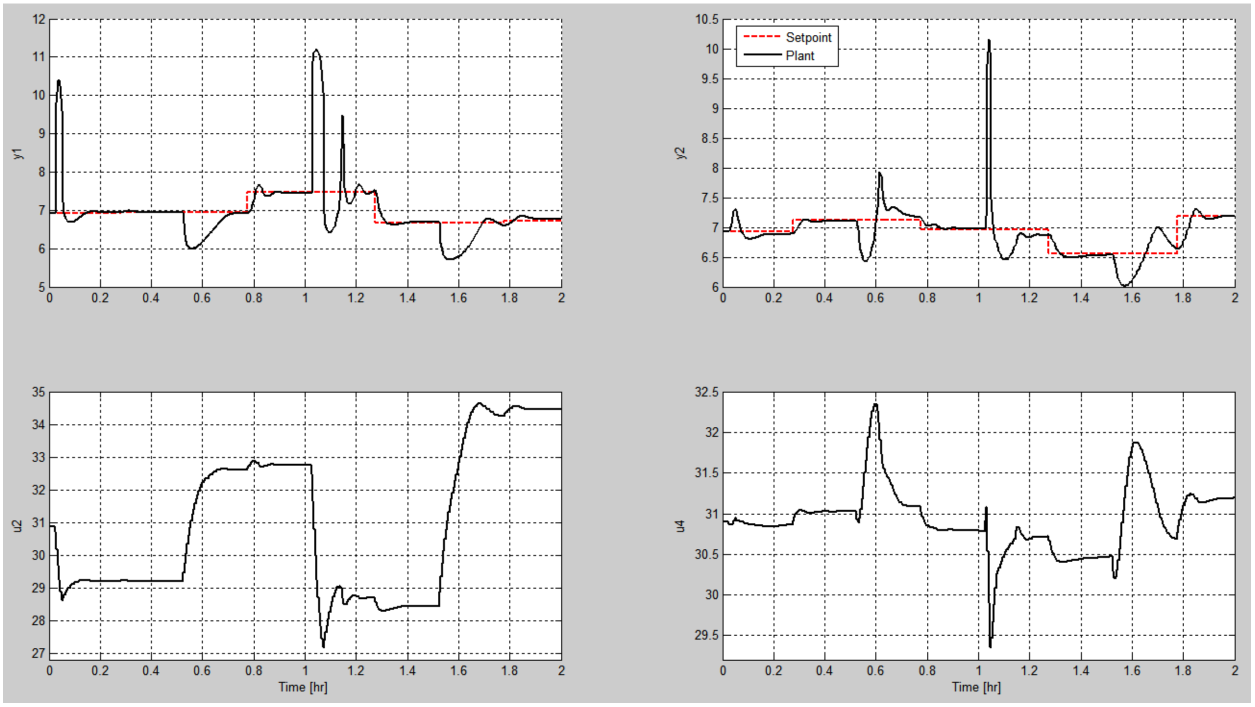


Figure 23: MMPC control responses (CV, MV only) for MATLAB-DAE-Based pH Neutralization Reactor example.

Figure 24 provides information on each of the D-RM output(s) along-side the plants outputs (top-row of graphs). In addition, it also shows plots for all the plant's inputs (bottom-rows). The plant has very good response in face of large acid flowrate fluctuations. Typically, this is a difficult problem to address, since not only does the acid-base fluctuations gets transferred to the second-tank resulting in a second-third order concentration and volume dynamics; the pH measurement has logarithmic dependence on the concentrations which makes the measurement highly nonlinear to the MVs.

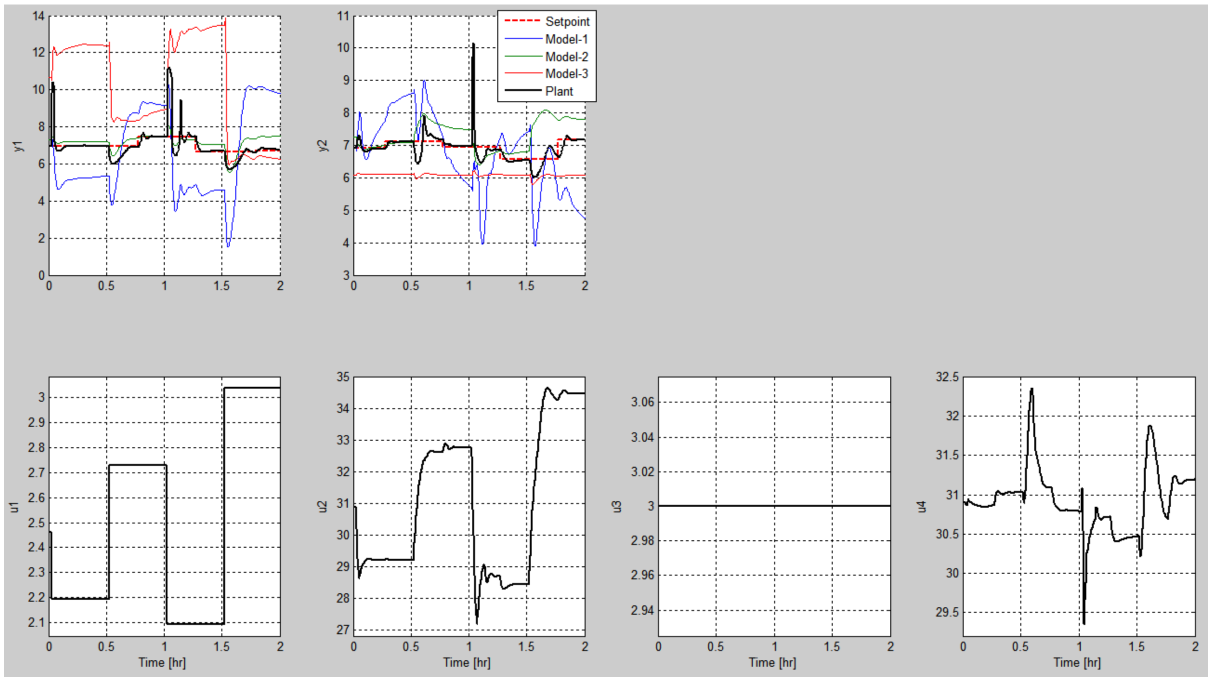


Figure 24: MMPC control responses (all) for MATLAB-DAE-Based pH Neutralization Reactor example.

Figure 25 shows the model-weights during control-transients. It is very clear that at t = 1.6 hr, Model-2 grows inaccurate in predicting responses corresponding to the operating conditions at that time. The MMPC algorithm automatically switches over to Model-3 for better predictions. **Note:** Model-1 was obtained by linearizing around an operating condition which was the same as the nominal operating point for the current run. Such an approach is typically adapted in a single model MPC, where the model is built around a base-load condition. In the particular set of randomly induced setpoints and disturbances, corresponding to Figures 22–26, this base-load model (Model 1) maintained a zero model-weight, indicating inferior prediction performance compared to the other two models over a period of time; therefore, the benefit of using MMPC is evident when capturing nonlinearities and off-design conditions.

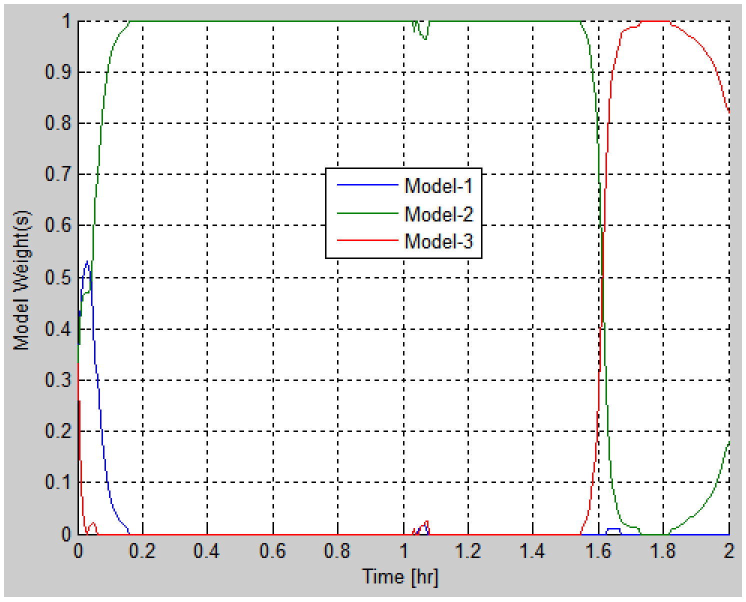


Figure 25: MMPC adaptive-evolution of model-weights for MATLAB-DAE-Based pH Neutralization Reactor example.

Figure 26 shows the computational cost involved at each time-step. The left axis and the corresponding plot in “blue” gives the CPU-time required for control-calculation at each step. This time is significantly less compared to using the DABNet-based NMPC formulation, due to a much faster constrained quadratic programming approach with a model probability and weight update strategy; making MMPC a much faster alternative approach to NMPC. The plant computation time for the exact same process (pH Neutralization Reactor), differs significantly between MATLAB-DAE and ACM based “plant” processes. MATLAB DAE takes about 0.02 sec average CPU time compared to 0.5 sec in ACM-based process. Typically, commercial process simulators including ACM are very-fast when performing numerical integration. This 20–25 times slower computation-time can be attributed to a MATLAB/Simulink – ACM communication delay, not the respective solver evaluations.

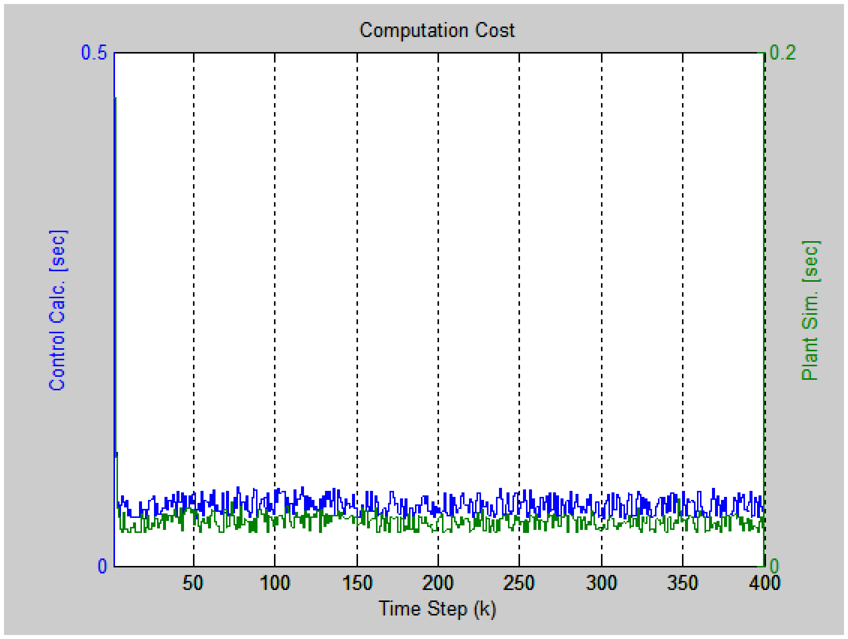


Figure 26: MMPC computation cost plots for MATLAB-DAE-Based pH Neutralization Reactor example.

Example 3 – MMPC for ACM-Based Plant (Bubbling Fluidized Bed Adsorber)

This example demonstrates the MMPC feature of the APC Framework for “plant-models” based on ACM or Aspen Plus Dynamics (APD). A BFB model has been provided in the “`<APCBaseFolder>\Examples\BFB_Adsorber`” subfolder as an example. This file models the sorbent-based two-bed BFB CO_2 adsorber-reactor system developed by the CCSI team. The current version of the BFB adsorber-reactor model (`BFBAdsorberDyn_v5.1.1.dynf`) contains over 20,000 DAEs. The input variables include the flow rate of feed flue-gas stream and sorbent flowrate. The output variable is the overall CO_2 capture. The minimum integration time step is 0.001 seconds within the ACM solver. The process time ranges from 150–200 seconds and 10 seconds is considered the sampling time without reasonable loss of high-frequency band information. The following steps illustrate the workflow involved in setting up a detailed control simulation, in particular, to ACM-based “plant” models. The explanation of each step is limited for this example. Refer to Example 1 – pH Neutralization Reactor for more narrative.

1. Copy the entire example subfolder of pH Neutralization Reactor “`<APCBaseFolder>\Examples\BFB_Adsorber`” to a location determined by the user. The user requires write-permissions to this folder. Browse to the “`BFB_Adsorber`” subfolder.
2. Open MATLAB (if not already done) and then browse to the newly copied “`BFB_Adsorber`” subfolder. This becomes the *working directory* in MATLAB. Double-click the “`BFBAdsorber.mdl`” Simulink model file.

   This displays a Simulink model as shown in Figure 27. Double-click the “AMSimulation” block. The AMSimulation Block configuration window displays, as shown in Figure 28.

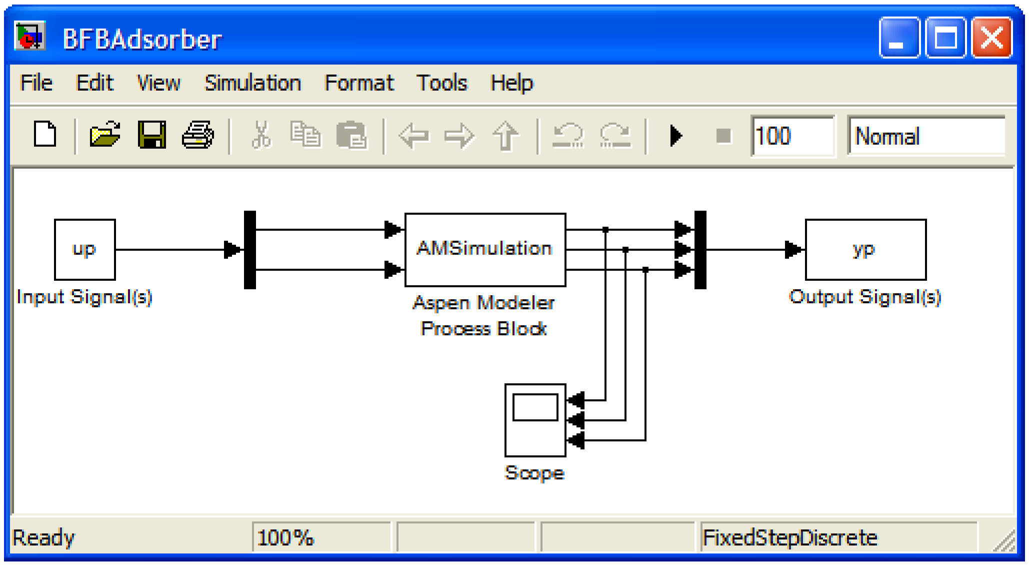
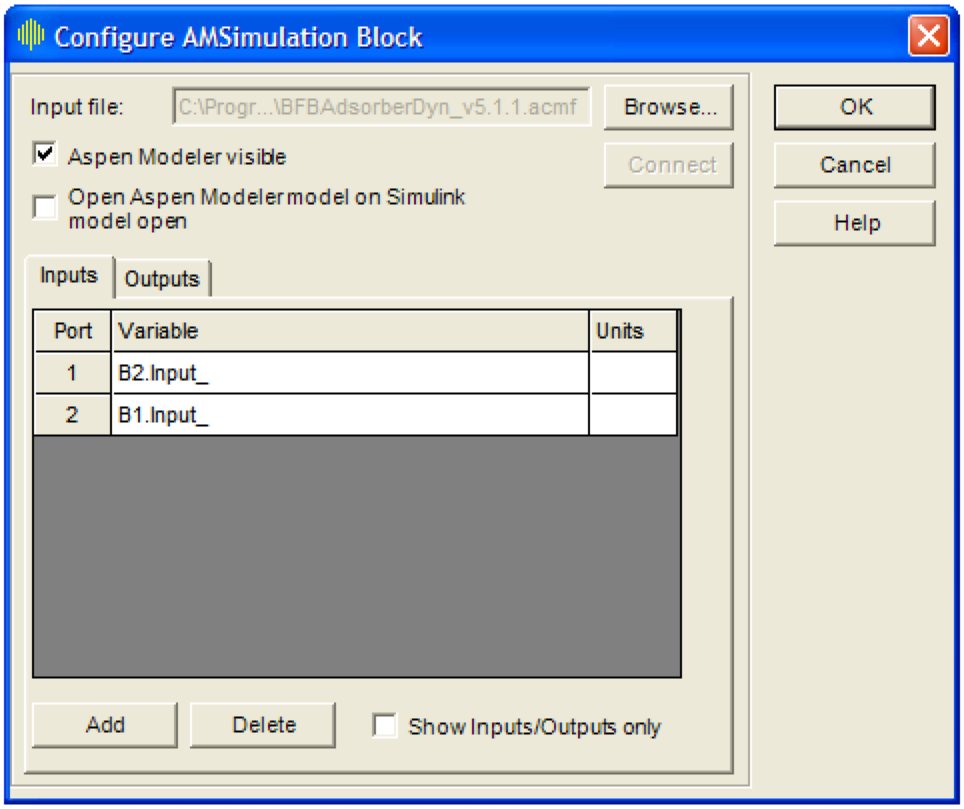


Figure 27: Simulink model with embedded ACM-Based BFB Adsorber model.


Configure AMSimulation Block
Input file:
C:\Progr...\BFBAdsorberDyn_v5.1.1.acmf
Browse...
OK
Cancel
Help
Connect
Aspen Modeler visible
Open Aspen Modeler model on Simulink model open
Inputs
Outputs
Add
Delete
Show Inputs/Outputs only

Port	Variable	Units
1	B2.Input_	
2	B1.Input_	

Figure 28: Configure AMSimulation Block window for BFB Adsorber.

Click “Browse” and then select the ACM file in the working folder to launch the ACM. The input/output port variable should populate automatically. Check for these I/O variables:

Inputs:	`B2.Input_`, `B1.Input_`
Outputs:	`CO2_Capture_Percent.Output_`, `Valve3.SorbIn.Fm`, `Valve1.GasIn.F`

If all the variables are consistent, click OK, and then save and close the Simulink model. **Note:** Never close the accompanying ACM window manually because this may lead to file corruption. This window automatically closes upon MATLAB exit.

The nominal values in ACM at which the pre-configured BFB Adsorber model was provided is as follows, in the order as above:

Inputs:	600000 [kg/hr], 6750 [kmol/hr]
Outputs:	87.7645 [%], 600000 [kg/hr], 6750 [kg/hr]

Note: The second and third outputs are the actual physical values of the flowrates, whereas the inputs are, in reality, the setpoints to the regulatory-layered flow-controller.

3. As opposed to the pH Neutralization Reactor case, there is no shortcut to initialize the plant from within MATLAB. One option is to open the flowsheet in ACM, make an initialization and steady-state run within ACM, quickly note the output values, and then save the flowsheet. More about this process is discussed in Step 4. Simulation of the plant is completed through the marching script, "`SimulatePlant.m`". To successfully launch and march an ACM-based "plant" within the APC Framework, two things are required: (1) the "plant" should be initialized within Aspen and (2) the initial I/O values should be present in "`u_plant`" and "`y_plant`" variables for many APC Framework scripts (which depend on "plant" initialization) to run properly.

4. Generate the step-test-results. To skip this step, refer to the pre-built step-test results "`BFBStepResponseData.mat`" located in the "Step-Tests" subfolder. Double-click this file within MATLAB to load the three step-test set data into the workspace. Proceed to the end of this step. **Note:** The following steps may take as long as 2 hours of simulation time for generating the three step-response sets. Use the pre-built file mentioned above to avoid this wait time.

   Verify that ACM is initialized at "[600000, 6750]" and is currently in Dynamic run-mode. Verify the first line in the "`Initialize_BFB.m`" script (open in MATLAB editor) corresponds to

   ```
       u_plant = [600000; 6750];
   ```

   Invoke this script in the MATLAB command window displaying ACM. Run the simulation for a few time-steps and obtain the value(s) of outputs.

   Invoke the "`GetStepTestData.m`" Step-Response Data Generator script in the MATLAB command window. Unlike the previous example, the step-responses have to be generated one at a time for different nominal points, because of manual initialization of the "plant" from within ACM. **Note:** If this is the first launch of this script for this example, clear the workspace by issuing the following command, "`clearvars -except u_plant y_plant u_name y_name u_unit y_unit t_unit`". This resolves possible conflicts with already existing variables in the workspace.

   a. Enter "1" for the training-set no.
   b. Enter "10" as the sampling time in plant's units [s].
   c. Enter "[1 2]" for the D-RMs desired inputs.
   d. Enter "[1]" for the D-RMs desired outputs.
   e. Enter "14000" as the end-time in plant's units [s].

   After the first set of step-responses is completed (~30–50 minutes real-time), the lot of variables (`T`, `U`, `Y`, `Uss`, `Yss`, `ym_idx`, etc.) is present in the workspace. Find more narrative on the variables in Step 4 of Example 2. Save/backup all of the variables in the workspace area to a file determined by the user by using "`BFB_SRData.mat`" or issuing the command "`save('BFB_SRData.mat')`" in the command window. A best practice is to exit MATLAB at this point which cleans-up any communication objects with ACM and automatically close any open ACM window(s).

   Create a backup of "`BFBAdsorberDyn_v5.1.1.dynf`" and "`BFBAdsorber.mdl`" files. Open "`BFBAdsorberDyn_v5.1.1.dynf`" in ACM. To advance to a new steady-state operating point for step-tests, change the run mode to "Steady State" (from "Dynamic") by selecting Run → Mode → Steady State or from the drop-down menu in the toolbar.

Navigate to a Time Delay (ΔT) block on the right named "B2". Double-click to open the B2.Results Table. The input-signal "Input_" corresponds to the sorbent-flowrate to the adsorber unit. Increase this value by 30 percent, up to 780000 [kg/hr]. Double-click the Multiplier (X) block named "CO2_Capture_Percent". The output signal "Output_" represents the net CO_2 capture in the adsorber.

Make the steady-state run to advance to the new operating point with increased inflow of sorbent by clicking "Run" (▶) in the toolbar or pressing "F5". After the run is complete, the CO_2 capture value increases to 90.14 percent.

Change the run mode back to "Dynamic" by selecting Run → Mode → Dynamic or from the drop-down menu in the toolbar. This step is critical. Save the Simulation.

Open MATLAB and browse to the working folder. Load the previously saved Set-1 results file "BFB_SRData.mat". Double-click the "Initialize_BFB.m" file to open in MATLAB editor and modify the first line as follows:

```
u_plant = [780000; 6750];
```

Invoke this script in the MATLAB command window and repeat the steps of invoking the Step-Response Data Generator script. Repeat similar steps as Set-1, remembering to enter "2" as the training-set number. After the run is complete (~30–50 minutes real-time), the workspace variables are concatenated with new data from Set-2 runs. Continue repeating this Set-3 with 30 percent decrease in nominal sorbent flowrate where the following line is to be entered in initialization script.

```
u_plant = [420000; 6750];
```

Obtain the D-RMs corresponding to the above three step tests. The detailed explanation of obtaining the D-RMs from Step-Response Data is given in Example 2. This involves the usage of the System Identification Toolbox, generation of three "ident" data-objects, converting to state-space data-sets (for cross-compatibility), and finally using the "Ident2DRM" script to convert the identified data-sets into APC Framework compatible State-Space D-RMs. All of the D-RMs are made available in the "DRM_SSBank_BFB.mat" file in the working folder. Advance to the APC configuration step (Step 5).

5. Configure the APC object. If the user may choose to skip this step, a preconfigured file "APC_MMPC_BFB.mat" is made available. Launch the APC Configuration Script by issuing the command "APCConfigTool". It is recommended to clear the workspace before doing this.
   a. Select the "DRM_SSBank_BFB.mat" file when prompted for D-RM information.
   b. Press "Enter" to accept all three models in the model-bank.
   c. Enter "[1]" as the desired manipulated input.
   d. Enter "[1]" as the desired control variable.
   e. Enter "n" to exit the APC Configuration Tool.
   f. Run the file "SetupDemoAPCPars_BFB.m".
   g. Save the "APC" variable for online simulation runs.

6. Conduct closed-loop simulations. Initialize the BFB plant at the original operating point [600000; 6750]. Refer to the processes in Step 4 or restore the original version from the backed up file. Double-click the "`Initialize_BFB.m`" file to open it in the MATLAB editor and verify the first line as follows:

    ```
    u_plant = [600000; 6750];
    ```

    Invoke this script in the MATLAB command window to obtain the outputs, "`y_plant`".

    Launch the APC Simulation Script by issuing the command "`APCSimulationTool`" in the MATLAB command window.

    a. Specify the file containing the previously saved "`APC`" variable or use the preconfigured "`APC_MMPC_BFB.mat`" file.
    b. Enter "1400" as the total simulation time.
    c. Enter "300" as the agitation interval.
    d. Enter "0" SD for setpoint agitation; only disturbance rejection performance is considered for Example 3.
    e. Enter "10" for the SD for disturbance agitation; denoting "±10%" SD in disturbance from the current nominal value (6750 kmol/hr).
    f. Enter "[2]" as the disturbance index.

    The five MATLAB figure windows that display, show the controller responses for random agitation of flue-gas flowrates. Detailed interpretation of these figures is provided in Example 1 and Example 2. This concludes the BFB Adsorber example case.

2.4 State and Disturbance Estimation

Description

Model-based controllers are prone to modeling uncertainties within the predictive D-RM and "plant" measurement errors due to sensor noise, which deteriorate controller performance significantly. Quantifying such uncertainties and errors online as the "plant" data are obtained may become necessary to greatly improve the performance of an online control algorithm. APC Framework offers many sophisticated methodologies and algorithms for doing the same. This section describes such methods and demonstrates their applicability on previous benchmark problems. The framework utilizes the Kalman Filtering approach for state and disturbance estimation in the face of process noise.

If $\mathbf{w}_k^x$ and $\mathbf{w}_k^d$ quantify the uncertainties in states and unmeasured disturbances respectively, the following equations provide the model-evolution over subsequent time-intervals in the face of process and measurement uncertainties:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{B_d}\mathbf{d}_k + \mathbf{w}_k^x$$
$$\mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{w}_k^d$$
$$\mathbf{y}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1}) + \mathbf{v}_k$$

where, $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^1 \\ \mathbf{x}_k^2 \\ \vdots \\ \mathbf{x}_k^{ny} \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{ny} \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_{ny} \end{bmatrix}$ for DAB-Net D-RMs.

For state-space based D-RMs, $\mathbf{h}(\mathbf{x}_k) = \mathbf{C}\mathbf{x}_k$, whereas for DAB-Net based D-RMs,

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} NN^1(\mathbf{x}_k^1) \\ NN^2(\mathbf{x}_k^2) \\ NN^3(\mathbf{x}_k^3) \\ \vdots \\ NN^{ny}(\mathbf{x}_k^{ny}) \end{bmatrix}$$

Here, $\mathbf{d}_k$ is the unmeasured disturbance which is modeled into the process through $\mathbf{B_d}$. Typically, $\mathbf{d}_k$ may be considered as a sub-set of process inputs ($\mathbf{u}_k$) where estimates of $\mathbf{d}_k$ are evaluated as feedback from "plant" measurements ($\mathbf{y}_k$) via the filtering algorithm.

The disturbance model can take many different forms among which random-step input disturbance (RSID), random-ramp input disturbance (RRID), and periodic disturbance (PD) are very common and present in the APC Framework. Choosing among these disturbances requires a good heuristic knowledge of the process and type of operation (near-nominal, ramp, etc.) at which state/disturbance estimation is required, although an RSID formulation is a good-start.

An *augmented* form of the formulation is used for consistency with state-space based Kalman filter algorithm. This form treats the disturbances as part of augmented states enabling a convenient handling of states and disturbances estimation, either together or independently. For instance, the RSID formulation mathematically evolves into the following augmented form:

$$\mathbf{x}_{k+1}^{aug} = \mathbf{A}^{aug}\mathbf{x}_k^{aug} + \mathbf{B}^{aug}\mathbf{u}_k + \mathbf{w}_k$$
$$\mathbf{y}_{k+1} = \mathbf{h}^{aug}(\mathbf{x}_{k+1}^{aug}) + \mathbf{v}_k \quad .$$

where, $\mathbf{x}_k^{aug} = \begin{bmatrix} \mathbf{x}_k^1 \\ \mathbf{x}_k^2 \\ \vdots \\ \mathbf{x}_k^{ny} \\ \hdashline \mathbf{d}_k \end{bmatrix}$, $\mathbf{A}^{aug} = \begin{bmatrix} \mathbf{A} & \mathbf{B_d} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$, $\mathbf{B}^{aug} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}$ and $\mathbf{w}_k = \begin{bmatrix} \mathbf{w}_k^x \\ \mathbf{w}_k^d \end{bmatrix}$.

For state-space D-RMs, $\mathbf{h}^{aug}(\mathbf{x}_k^{aug}) = [\mathbf{C} \quad \mathbf{0}]\mathbf{x}_k^{aug}$, whereas for DAB-Net D-RMs, $\mathbf{h}^{aug}(\mathbf{x}_k^{aug}) = \mathbf{h}(\mathbf{x}_k)$.

Ideally, the covariance for process (both states and disturbance) and measurement uncertainty is $\mathbf{Q} = E(\mathbf{w}\mathbf{w}^T)$ and $\mathbf{R} = E(\mathbf{v}\mathbf{v}^T)$. The APC Framework assumes that the process noise covariance can be adequately captured using the following expressions:

$\mathbf{Q} = \begin{bmatrix} \mathbf{0}_{nx} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q_d} \end{bmatrix}$, for disturbance estimation only, $\mathbf{Q} = \begin{bmatrix} \mathbf{Q_x} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{nd} \end{bmatrix}$ for state-estimation only and $\mathbf{Q} = \begin{bmatrix} \mathbf{Q_x} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q_d} \end{bmatrix}$ for both state and disturbance estimation, where $\mathbf{Q_x}$ and $\mathbf{Q_d}$ are noise covariance of states and disturbances respectively.

Based on the above-mentioned assumptions, the user is required to provide covariance of states, disturbances, and measured outputs along-with indices of disturbance variables among D-RM inputs as specifications for setting up state-estimation. In addition, filtering algorithm (such as UKF, EKF) and state-estimation method (such as RSID, RRID, or periodic) has to be specified. These have been demonstrated through the following examples.

Example 4 – Disturbance Estimation using Unscented Kalman Filtering (UKF) for pH Neutralization Reactor (MATLAB ODE-Based Plant Model)

This example serves as an extension to Example 1, with MATLAB ODE-based "plant" instead of ACM-based high-fidelity "plant". Here a disturbance estimation in the pH Neutralization Reactor will be demonstrated. As a reminder, flowrate of acidic streams (Q1 and Q5) served as a process disturbances to the pH 2-Tank system. The following steps should be self-explanatory. Please refer to Example 1 and Example 2 – pH Neutralization Reactor for more narrative.

1. Copy the entire example subfolder of pH Neutralization Reactor "`<APCBaseFolder>\Examples\pH_Reactor_MATLAB`" to a location determined by the user. The user requires write-permissions to this folder. If this was already done in context of Example 2, this step is not required. Browse to the "`pH_Reactor_MATLAB`" subfolder.
2. Open MATLAB (if not already done) and then browse to the newly copied "`pH_Reactor_MATLAB`" subfolder. This becomes the *working directory* in MATLAB. Double-click the `pH_Reactor.m` code in MATLAB or issue the command, "`edit pH_Reactor.m`". Enforce some white noise with SD of 0.1% nominal value in the acid flowrates, Q1 and Q5 (which hold index 1 and 3 among all plant inputs) by modifying line 73 from "`noise_u = [0; 0; 0; 0]`" to "`noise_u = [0.001; 0; 0.001; 0]`". **Note:** These are *unmeasured disturbances* and the effect is only visible indirectly through measured outputs i.e., pH measurements. Similarly, force white-noise of 1% SD in measurements by changing content of line 74 from "`noise_y = [0; 0]`" to "`noise_y = [0.001; 0.001]`".
3. Initialize the plant by running the "`Initialize_pH_Reactor.m`", similar to Example 2.
4. Launch the APC Framework's DABNet-parameter to the D-RM converter script by issuing the command, "`DABNetParameter2DRM`," in the MATLAB command window. Browse to "`pH_Neut_DRMParameters.m`", similar to that highlighted in Example 1. After the conversion is complete, a workspace variable named, "`DRM`," is created.

   Note: The content of DABNet-parameters file for both ACM-based "plant" (present in the "`Examples\pH_Reactor`" folder) and MATLAB ODE-based "plant" for current example are exactly the same.

At this point, the state-estimation can be configured using the following steps:

a. Use the following set of commands on the MATLAB command window to form the state-estimation parameters structure. The `FilterMethod` field specifies that an UKF estimation method is desired. The `d_idx` field provides the indices corresponding to the estimated disturbances among the D-RM inputs (**Note:** This is not among the plant inputs, contrary to many of input-indices specifications earlier). The `qd` field specifies the SD of the estimated disturbances. Since a scalar is used, it will be used as SD for both acidic flowrates. If separate SD for each disturbance is desired, an array of SD can be provided corresponding to this field such as "`SEPars.qd = [0.001; 0.0025];`". Similarly, SD for each or all measured outputs are provided through the `r` field via an array or scalar respectively. Finally, the disturbance estimation method is specified by the `SEFormulation` field. Here random-step input disturbance estimation will be used (also takes `RSI` or `RSID` as field entries):

```
SEPars.FilterMethod = 'UKF';
SEPars.d_idx = [1 3];
SEPars.qd = 0.001;
SEPars.r = 1;
SEPars.SEFormulation = 'Random-Step Input Disturbance';
```

Alternatively, all of these fields can be entered in a single statement such as the following:

```
SEPars = struct('FilterMethod','UKF','d_idx',[1
3],'qd',0.001,'r',1,'SEFormulation','RSI');
```

Note: `qd` and `r` specification are tuning parameters and determines how much confidence can be placed on the D-RM (specifically the disturbance-model, **Bd**) as opposed to measurement data. This is determined by the identification methodology. D-RM builder provides analysis tools for quantifying the D-RM uncertainty. More importantly, it must be realized that `qd` and `r` specification may not be directly related to noise enforced earlier within the plant.

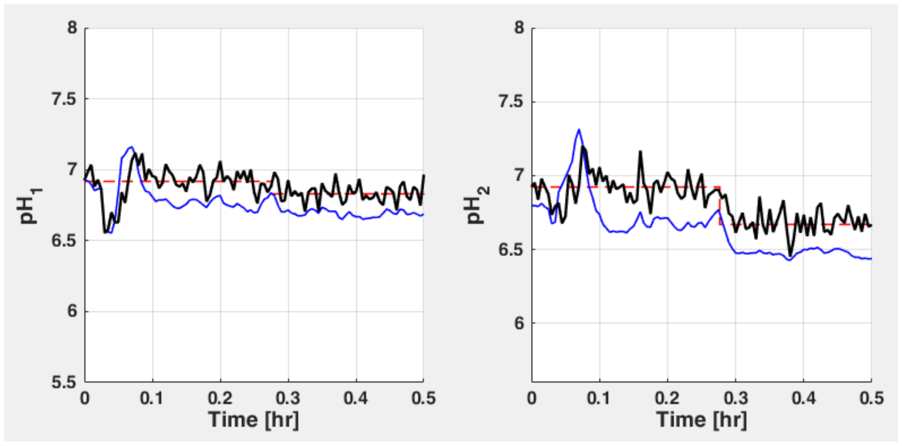
b. Use the state-estimation parameters structure for configuring the state-estimation within the D-RM using the following command in MATLAB command window:

```
DRM.setupStateEstimation(SEPars);
```

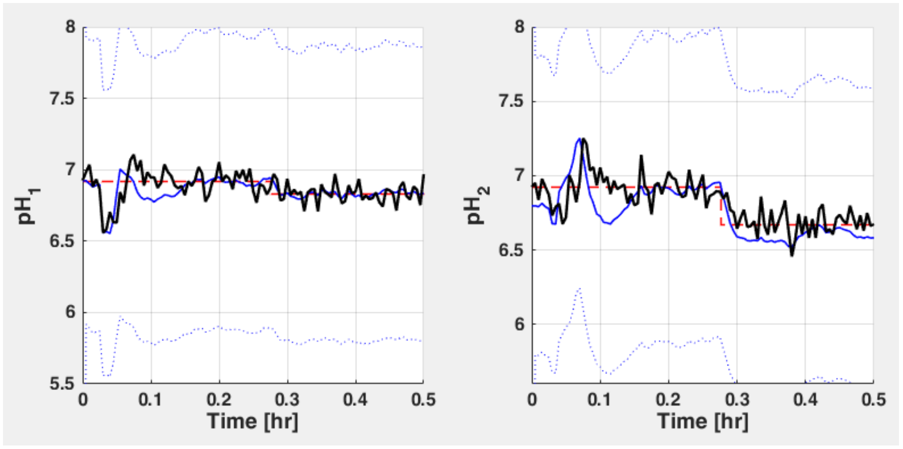
This updates the D-RM and enables state-estimation with the provided parameters above.

Save the variable "`DRM`" in a MATLAB data file, by right-clicking the variable name and then selecting "Save As". A preloaded MATLAB data file, "`DRM_DABNetSE_pH.mat`," is already created in the example folder, although this file can be overwritten for consistency with the user's MATLAB version. **Note:** This file can be named anything, although this file is referred to as the D-RM data-file or "`DRM_DABNetSE_pH.mat`" interchangeably throughout the rest of this example.

5. Configure the APC object. If the user may choose to skip this step, a preconfigured file "`APC_DABNetSE_pH.mat`" is made available. Launch the APC Configuration Script by issuing the command "`APCConfigTool`". It is recommended to clear the workspace before doing this.
    a. Select the "`DRM_DABNetSE_pH.mat`" file when prompted for D-RM information.
    b. Enter "[2 4]" as the desired manipulated input.
    c. Enter "[1 2]" as the desired control variable.
    d. Enter "n" to exit the APC Configuration Tool.
    e. Run the file "`SetupDemoAPCParsSE_pH.m`".
    f. Save the "`APC`" variable for online simulation runs.
6. Conduct "closed-loop" simulation by referring to Step 6 of Example 1. Provide the preconfigured APC data-file "`APC_DABNetSE_pH.mat`" or the file used to save the `APC` object in the previous step. After the simulation, various plots are displayed similar to the previous examples. The plot showing all I/O variables show some additional dotted lines above and below the D-RM responses when state-estimation is enabled. These are confidence lines for the D-RM predictions. As **Q**/**R** decreases, signifying greater trust with the measured data, the confidence band gap shrinks and the D-RM responses get closer to the "plant" data.



(a)



(b)

Figure 29: Comparison plot showing advantages of using disturbance estimation using UKF.

Figure 29 shows clear advantages of using disturbance estimation (Figure 29b) as opposed to utilizing NO estimation in the previous examples (Figure 29a). The D-RM are much smoother and closer to the plant responses and hence lead to lower setpoint residuals and better controller responses. This improvement becomes very obvious in the face of large measurement noise.

Example 5 – State Estimation using Extended Kalman Filtering (EKF) for pH Neutralization Reactor (MATLAB ODE-Based Plant Model)

This example demonstrates further capabilities of state and disturbance estimation features within the APC Framework. Here the state-estimation feature without any disturbance estimation will be highlighted. Refer to Example 4 for most of the steps. Only the instructions specific to this example are given here. Do not use any preconfigured file for this example.

It is recommended to clear the workspace to remove remnants of previous parameters, D-RM and APC objects by issuing "`clear all`" in the MATLAB command window. Since **no** disturbance estimation will be considered here, an accurate disturbance measurement will be assumed. In Step 2 – Plant Specification, modify the "`noise_u`" back to "`[0; 0; 0; 0]`" keeping the measurement noise (`noise_y`) the same as the previous example. Initialize the "plant" and run the "`DABNetParameter2DRM`" script. Once the "`DRM`" variable is defined and present in the workspace, implement the following in the MATLAB command window:

```
SEPars.FilterMethod = 'EKF';
SEPars.IncludeStates = true;
SEPars.d_idx = [];
SEPars.qx = 0.001;
SEPars.r = 1;
SEPars.SEFormulation = 'Random-Step Input Disturbance';
DRM.setupStateEstimation(SEPars);
```

Alternatively, all of these fields can be entered in a single statement such as the following:

```
SEPars =
struct('FilterMethod','EKF','IncludeStates',true,'d_idx',[],'qx',0.001
,'r',1,'SEFormulation','RSI');
DRM.setupStateEstimation(SEPars);
```

The `FilterMethod` field specifies that an Extended Kalman Filter (EKF) estimation method is desired. For state-estimation, the `IncludeStates` field is specified to be "true". The `d_idx` field is left empty since no disturbance estimation is desired. **Note:** If the `d_idx` field is omitted, all D-RM inputs will be picked up as estimated disturbance by default. The `qx` field specifies the SD of the estimated states. This only takes a single scalar value, corresponding to SD for all states. DABNet states correspond to Laguerre functions and do not vary significantly in magnitude.

Configure the controller following Step 5 of the previous example and conduct a "closed-loop" simulation run (Step 6). Similar to the previous example, all I/O plot shows a +/- SD lines corresponding to each D-RM output, showing confidence of the estimation formulation.

2.5 APC Framework Simulation Tool Graphics User Interface

Release 2015.10.00 features the APC Framework Simulation Tool GUI for user-friendly implementation of closed-loop simulation within the CCSI APC Framework toolset. The user can specify the "plant", define the controller (either from a saved MATLAB data file or MATLAB workspace), initialize the "plant" and controller, and implement closed-loop simulated runs from within the GUI. Once the APC object is specified, the controller parameters can be viewed and modified in a control panel designed very similar to a real-plant control-room panels. Upon starting the closed-loop run, an animated dynamic plot shows various controller-relevant I/O. The user may pause, restore, and can save the history data, in between the simulation runs. On the fly modification of APC parameters is possible including the optimizer used for solving the optimal control problem. The GUI also features many APC response visualization options, which may help the user access performance of the controller in a simulated environment and later utilize them in their specific application.

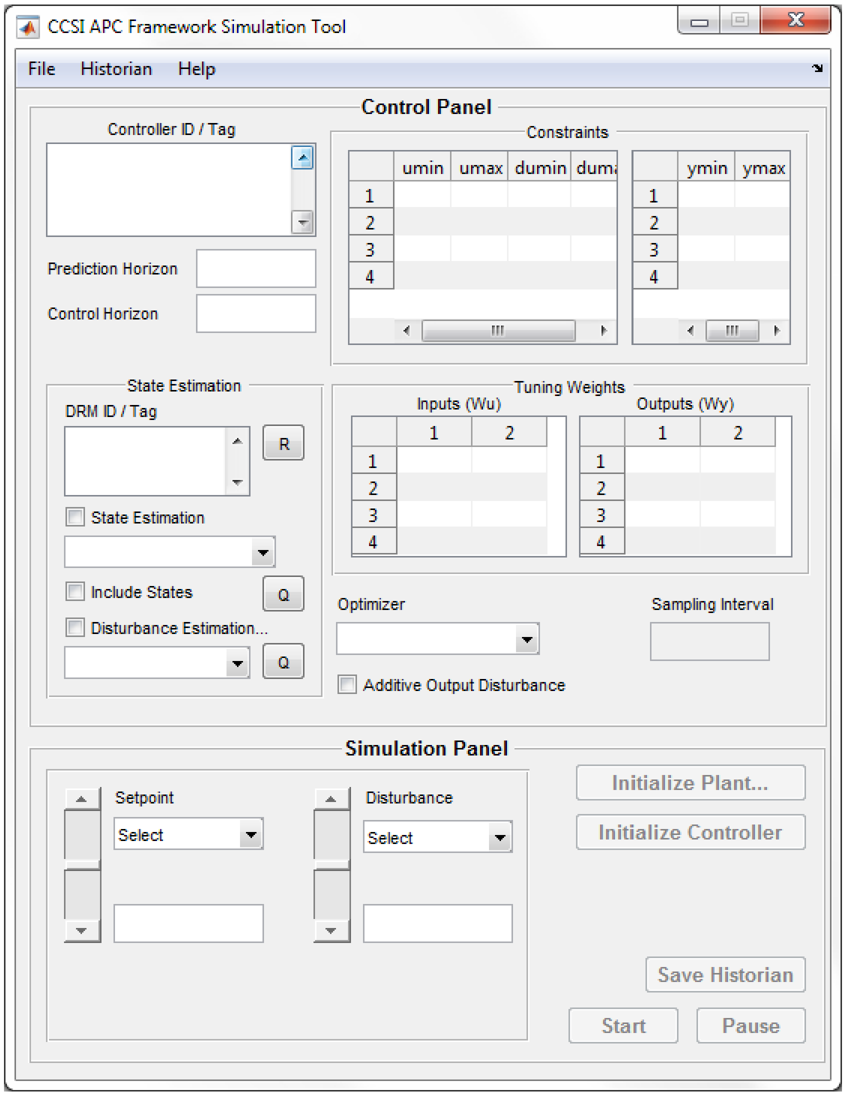


Figure 30: APC Framework Simulation Tool graphics user interface upon first start.

The GUI can be invoked by issuing the command “`APCSimulationToolGUI`” in the MATLAB command window. The interface looks similar to that shown in Figure 30. The “File” menu can be used to define a “plant” and a controller. The “Historian” menu can be used to load a historian data-base and thereafter plot various responses. The “Help” menu provides link to this documentation.

The GUI contains two functional panels – the *Control Panel* and the *Simulation Panel*. The control panel contains various GUI elements that correspond to different controller parameters. For MPC based controllers such as that present in the APC Framework, these parameters include prediction and control horizons, input absolute and rate constraints (umin, umax, dumin, dumax), output constraints (ymin, ymax), input and output tuning weights (Wu and Wy, respectively), optimizer used, and option for enabling/disabling additive output disturbance [8] to be used as a feedback mode. The state-estimation sub-panel includes details for one of more underlying D-RM(s) involved the APC. This subpanel displays whether state/disturbance estimation is enabled, type of filtering algorithm (UKF, EKF) and the involved disturbance-estimation methodology (RSI, RRI, periodic, etc.), if utilized. In the current version, this sub-panel is informative and cannot be modified on-the-fly.

The simulation panel, on the other hand, contains GUI elements directly related to closed-loop runs. The “Initialize” buttons are used to initialize plant and controller, both of which are necessary steps to start a closed-loop run. Once the plant is initialized, the setpoint and disturbance variables (specified at the Controller configuration step) automatically become available for run-time manipulation. If the simulation is paused/stopped, the “Save Historian” button becomes active, where the user may specify a MATLAB data file to save the historical data starting from time t = 0. This can be loaded anytime via the Historian → Load menu which can thereafter be used to plot various controller responses via the Historian → Plot menu.

The following examples demonstrate the usage of the APC Framework Simulation Tool GUI via two familiar examples that have been presented earlier – MMPC of ACM-based BFB Adsorber (Example 3) and DABNet-NMPC with UKF-Based disturbance estimation for MATLAB ODE-Based pH Neutralization Reactor (Example 4).

Example 6 – MMPC Simulation for ACM-Based BFB Adsorber using GUI

This example shows how to run the APC Framework Simulation Tool in context of MMPC as an alternative to the text-based APC Framework Simulation Tool in Step 6 of Example 3. The user must be familiar with Example 3 and must have generated D-RM and APC objects, by following Steps 1–5 of Example 3, before continuing with the following steps.

1. It is strongly advised to clear the MATLAB workspace by issuing the "`clear all`" command in the MATLAB command window and then close any previously opened instance of the APC Framework Simulation Tool GUI. Browse to the "`BFB_Adsorber`" subfolder that was used in Step 1 of Example 3. Invoke the APC Framework Simulation Tool GUI by issuing the command "`APCSimulationToolGUI`" in the MATLAB command window.
2. Navigate to File → Define Plant and then select "`Use SimulatePlant.m`". The GUI reports a successful assignment of "plant" marching script. The user can also choose "Select Marching Script" and browse to the "SimulatePlant.m" script file but that approach is slower due to the way MATLAB inherently invokes external scripts. Instead, it is strongly recommended to rename the marching script to "SimulatePlant.m" and choose the same from the menu. This message is also displayed if "Select Marching Script" is selected. **Note:** As soon as a "plant" script is chosen, the "Initialize Plant" button in the Simulation Panel becomes active.
3. Navigate to File → Define Controller and then select "From MAT File". Select the preconfigured APC data-file "`APC_MMPC_BFB.mat`" or the file used to save the APC object in Step 5 of Example 3. The GUI confirms the successful import of one controller object. The Control Panel is updated with the controller details such as ID and other controller parameters along with the three D-RM used within the APC object.
4. Initialize the "plant" by clicking "Initialize Plant" in the Simulation Panel. Select "Initialization Script", select the "`Initialize_BFB.m`" script file, and then click "Open". Wait for some time as the ACM window is invoked and the "plant" is run for 20 sampling-time (coded within the initialization script) to extract the nominal output values. After successful initialization, the setpoint variable (CO_2 Capture Percentage) and disturbance variable (Fluegas Flowrate) along-with their nominal values are displayed. The "Initialize Controller" button becomes active.
5. Initialize the controller by clicking "Initialize Controller". This essentially uses the initialized "plant" I/O values (`u_plant` and `y_plant`) obtained from the previous step, to initialize the D-RMs within the APC object. After the controller is successfully initialized, the "Start" button on the Simulation Panel becomes active as shown in Figure 31.

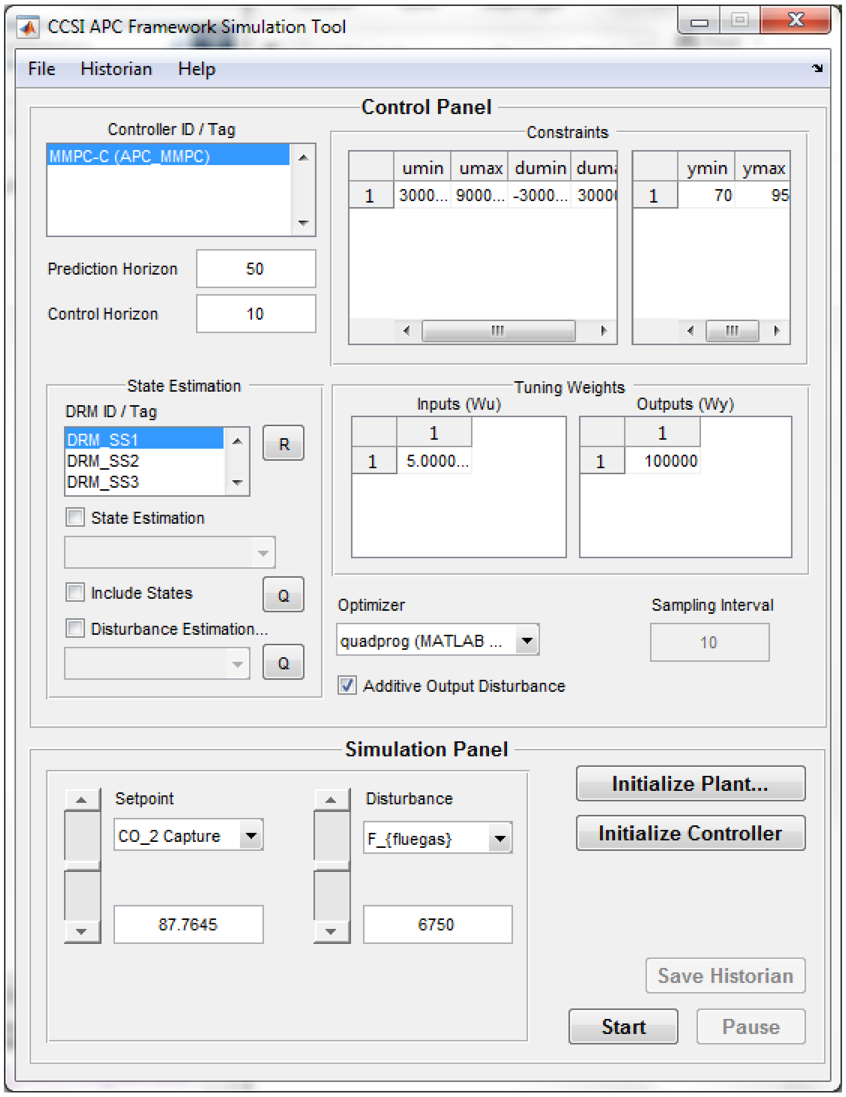


Figure 31: APC Framework Simulation Tool GUI window displaying MMPC for the BFB Adsorber after the plant/controller initialization step.

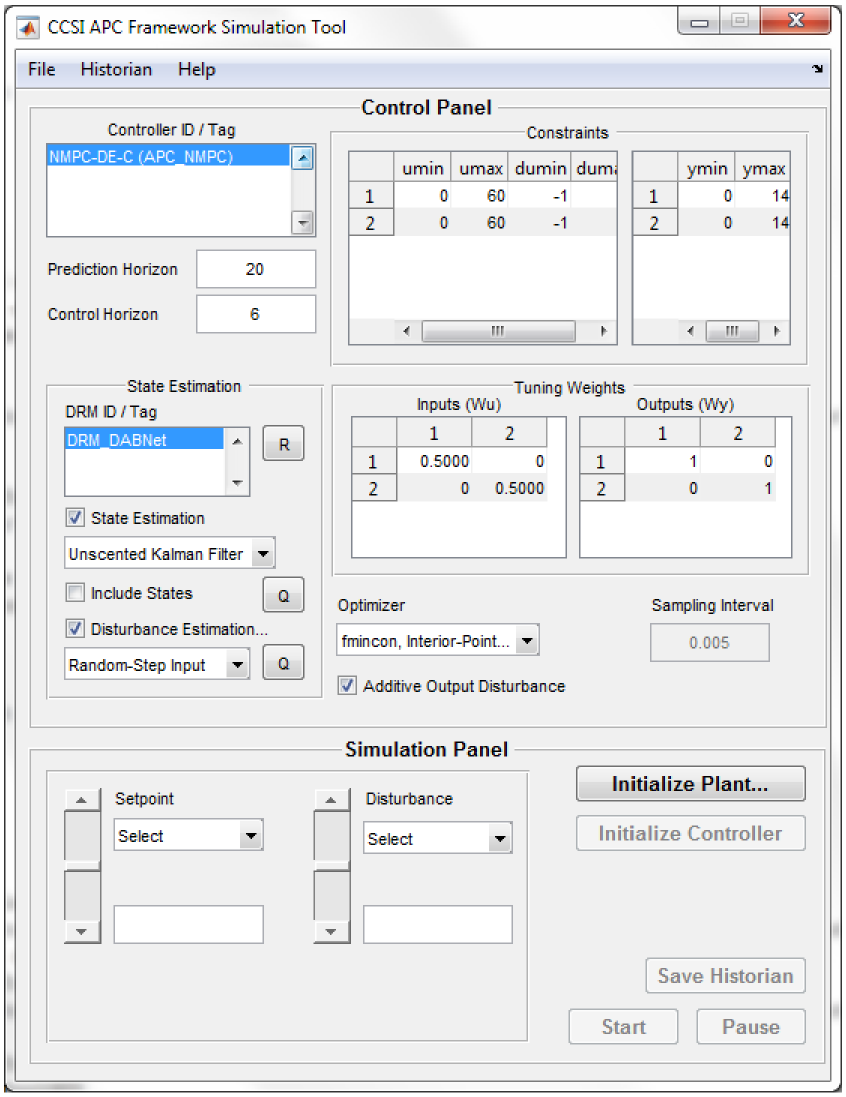
6. Click “Start” to initiate the simulation run. A dynamic plot displays showing the controller response. Use the slider buttons for setpoint and disturbance to manipulate these variables for testing the controller response. The text-box can also be used to type-in the entry. **Note:** This interactive feature offers an inherent benefit over the test-based APC Framework Simulation Tool. The user may change the controller parameters on-the-fly to test the controller performance, including changing constraints, prediction, and control horizon, and even the quadratic programming optimizer (for this MMPC example). Feel free to experiment with these settings. Once done click “Pause” to pause the simulation. The “Save Historian” button is now active.

7. Click “Save Historian” which prompts the user for saving the historian data for the simulation run (until now) in a .MAT file. Save the historian with the default historian name “`APCResponse.mat`”. Click “Resume” to continue. Once done, click “Stop”. The GUI will prompt the user to save the historian before it is permanently deleted from memory. Select “Yes” to save the historian, otherwise select “No” to finish the simulation run. The dynamic plot will close.
8. Navigate to Historian → Load → From MAT File and then select “`APCResponse.mat`”. After the historian is imported, navigate to the Historian → Plot menu to select various plots. These plots are similar to the plots displayed at the end of Step 6 in Example 3.

Example 7 – NMPC with Disturbance Estimation for MATLAB ODE-Based pH Neutralization Reactor

This example shows how to run the APC Framework Simulation Tool in context of NMPC with state estimation as an alternative to the text-based APC Framework Simulation Tool in Step 6 of Example 4. The user must be familiar with Example 4 and must have generated D-RM and APC objects, by following Steps 1–5 of Example 4, before continuing with the following steps. Please refer to previous example for a detailed narrative for each step.

1. Clear the MATLAB workspace by issuing the “`clear all`” command in MATLAB command window and then close any previously opened instance of the APC Framework Simulation Tool GUI. Browse to the “`pH_Reactor_MATLAB`” subfolder that was used in Step 1 of Example 4. Invoke the APC Framework Simulation Tool GUI by issuing the command “`APCSimulationToolGUI`” in the MATLAB command window.
2. Navigate to File → Define Plant and then select “Use SimulatePlant.m”.
3. Navigate to File → Define Controller and then select “From MAT File”. Select the preconfigured APC data-file “`APC_DABNetSE_pH.mat`”. Note how the state-estimation (using UKF) and disturbance-estimation (utilizing RSI formulation) is selected within “State Estimation” sub-panel as shown in Figure 32.



Controller ID / Tag: NMPC-DE-C (APC_NMPC)

Prediction Horizon: 20

Control Horizon: 6

Constraints

	umin	umax	dumin	dum
1	0	60	-1	
2	0	60	-1	

	ymin	ymax
1	0	14
2	0	14

State Estimation

DRM ID / Tag: DRM_DABNet

[✓] State Estimation

Unscented Kalman Filter

[ ] Include States

[✓] Disturbance Estimation...

Random-Step Input

Tuning Weights

Inputs (Wu)

	1	2
1	0.5000	0
2	0	0.5000

Outputs (Wy)

	1	2
1	1	0
2	0	1

Optimizer: fmincon, Interior-Point...

Sampling Interval: 0.005

[✓] Additive Output Disturbance

Setpoint: Select

Disturbance: Select

Figure 32: APC Framework Simulation Tool GUI window displaying the active state estimation for NMPC of pH Neutralization Reactor.

4. Click “Initialize Plant”, select “Initialization Script”, and then select/open the “`Initialize_pH_Reactor.m`” script file.
5. Click “Initialize Controller”.
6. Click “Start” for initiating dynamic “closed-loop” simulation runs. Use the drop-down arrow for selecting setpoint and disturbances. The slider buttons can be used to gradually modify these values or the text-box can be used to step-change directly to a certain value. The APC Framework features a hoard of NLP solvers/algorithms including fast interior-point solvers such as IPOPT [7]. Select/try these in the “Optimizer” drop-down list.
7. Click “Stop” to exit the simulation. Select “Yes” to save the historian as “`APCResponse.mat`”.
8. Manually load the historian by double-clicking the “`APCResponse.mat`” file in the MATLAB folder list panel. Navigate to Historian → Load → From Workspace. After the historian is loaded, navigate to the Historian → Plot menu to select various plots.

3.0 USAGE INFORMATION

3.1 Environment/Prerequisites

The APC Framework requires the MATLAB (R2011b or higher), Simulink, Control System Toolbox, and Optimization Toolbox. Developing the state-space models for MMPC may require a System Identification Toolbox and is highly recommended for using this application. The CCSI D-RM Builder tool (also available as part of the CCSI Framework for Optimization and Quantification of Uncertainty and Sensitivity (FOQUS) tool) is required for generating DABNet-based nonlinear D-RMs from ACM-based "plant" simulations, whereas a standalone version which accepts input/output dynamic data in CSV format can be used to identify DABNet D-RM for MATLAB ODE-based "plant" models. Using ACM/APD as the high fidelity "plant" simulation requires ACM and/or APD licenses. These licenses are also required for running the BFB Adsorber example case.

The current version of APC Framework has been developed and tested on MATLAB V8.6 (R2015b), Simulink V8.6 (R2015b), Control System Toolbox V9.10, Optimization Toolbox V7.3, System Identification Toolbox V9.3, and ACM Versions 7.3 through 8.4. It is highly recommended to use these versions for running the examples and for testing purposes. If the user has multiple versions of ACM and MATLAB installed, double-clicking the ACM files (.acmf), MATLAB function and script files (.m), and Simulink model files (.mdl) should invoke the above-mentioned versions of the software.

3.2 Support

To obtain support for this package, send an e-mail to ccsi-support@acceleratecarboncapture.org and/or complete the "Submit Feedback/Request Support" form available on the product distribution page.

3.3 Restrictions

It is the sole responsibility of the end-user (plant-model developer, plant operator, control engineer) to setup the communication between the plant and the framework. In simple cases of MATLAB DAE based plants, a setup similar to the pH Neutralization Reactor example provided in Section 2.3 Multiple-Model Predictive Control (MMPC) with Additive Output Disturbance [8] may be used. For ACM/APD-based control-driven application (typically used for accessing the control performance and research studies), a setup similar to the BFB Adsorber example in Section 2.4 State and Disturbance Estimation may be used. The templates for linking the ACM/APD process model to the APC Framework simulation are provided in the "`<APCBaseFolder>\Templates`" subfolder.

When using ACM/APD-based simulation, or any commercial simulator, in a controller-driven environment, the "plant" may fail to achieve convergence, in case aggressive control actions were evaluated by the controller. This may be avoided by specifying a reasonable ramp-constraint (`dumin`, `dumax`) during the control setup, based on heuristics within the process simulation.

4.0 ADVANCED FEATURES

An optional requirement for generating state-space linear parametric models using input-output time-series data, for creating MMPC model-bank, is the familiarity with using the MATLAB System Identification Toolbox. In MMPC examples, provided with the framework, "ident" object are pre-generated using the developer's copy of the toolbox. A MATLAB script for converting "ident" objects to D-RM SS objects (required by APC Framework) is provided in "`<APCBaseFolder>\Toolbox\Scripts\Ident2DRM.m`" and may be invoked by issuing the "`Ident2DRM`" command in the MATLAB command window.

5.0 DEBUGGING

Presently, error handling during the controller configuration and simulation stages is not very exhaustive. In case a critical error occurs, such as incorrect specification of D-RM or APC object, incorrect selection of files, incorrect specification of constraints, weighting matrices in proper format, and other parameters, MATLAB script generates an error (either instantaneously or at some later stage whenever an incorrect parameter is being used) and exits to the MATLAB command window prompt, by displaying exception messages. These messages might not be related to APC Framework scripts and can be difficult to debug. In future releases, exhaustive error handling routines will be included, which will enable users to actively diagnose and rectify any incorrect specifications during program runtime.

5.1 How to Debug

As indicated above, mistakes made in APC Framework work flow, are relatively difficult to diagnose. Any errors in the control-algorithms are the sole responsibility of the developer and if reported will be addressed in future releases of this tool.

If no errors are experienced during the configuration or simulation stages, but the simulated plant provides poor control responses or involves large computational delays in spite of various tuning trials, promptly send an e-mail to ccsi-support@acceleratecarboncapture.org to report these issues so that the exact problem can be pin-pointed. In numerous cases, providing the exact plant data or establishing communication might be impossible. In such cases, as much information as possible, including the D-RM(s), configured APC objects, and various parameters used will be helpful to diagnose and fix the issue.

5.2 Known Issues

The software module for the link between MATLAB and ACM was developed by AspenTech and is included as a part of the Aspen installation. However, a DLL named, "stdole.dll", might not be correctly registered by the Aspen installer, which could cause the configuration between Simulink and ACM described in Step 4 of the tutorial in Section 2.2 Decoupled A-B Net-Based Nonlinear Model Predictive Control to fail. Sometimes, installing a version of Microsoft® Office suite fixes the problem. If the user's system already has Microsoft Office or Microsoft Visual Studio installed, the DLL must have been registered correctly.

A 64-bit version of MATLAB is incompatible with implementing the MATLAB – ACM software link. A 32-bit MATLAB version is strongly recommended to run examples utilizing ACM-based "plant" processes.

5.3 Reporting Issues

To report an issue, please send an e-mail to ccsi-support@acceleratecarboncapture.org.

6.0 REFERENCES

[1] Sentoni, G.B., Biegler, L.T., Guiver, J.B., and Zhao, H., "State-Space Nonlinear Process Modeling: Identification and Universality," *AIChE Journal*, 44(10), 1998.

[2] Julier, S.J., and Uhlmann, J.K., "Unscented Filtering and Nonlinear Estimation," *proceedings of the IEEE*, 92(3), 401–422, 2004.

[3] Kurre-Kinsey, M., and Bequette, B.W., "Multiple Model Predictive Control of Nonlinear Systems," *Nonlinear Model Predictive Control, Lecture Notes, Control and Information Sciences*, 384, p. 153–165, 2009.

[4] Kurre-Kinsey, M., and Bequette, B.W., "Multiple Model Predictive Control Strategy for Disturbance Rejection," *Industrial & Engineering Chemistry Research*, 49(17), 7983–7989, 2010.

[5] Mahapatra, P., and Zitney, S.E., "Dynamic Maximization of Oxygen Yield in an Elevated-Pressure Air Separation Unit using Linear Multiple Model Predictive Control (MMPC) Framework," *proceedings of the 29th Annual International Pittsburgh Coal Conference*, Pittsburgh, PA, October 15-18, 2012.

[6] Lee, J.H. and N.L. Ricker, "Extended Kalman Filter based Nonlinear Model Predictive Control," *Industrial & Engineering Chemistry Research*, 33, 1530–1541, 1994.

[7] Wächter, A., and L.T. Biegler, "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, 106(1), 25–57, 2006.

[8] Pannocchia, G. and J.B. Rawlings, "Robustness of MPC and Disturbance Models for Multivariable Ill-conditioned Processes," *Technical Report – Texas-Wisconsin Modeling and Control Consortium*, 2001-2002, May 18, 2001.