Scott Research Group

toPSAil: Totally Open Pressure Swing Adsorption Intensification Laboratory

Taehun Kim and Joseph K. Scott

April 30, 2021

Georgia Institute of Technology, Atlanta, GA.

Copyright 2021 Georgia Tech Research Corporation ©

Release Notes

- *Totally Open Pressure Swing Adsorption Intensification Laboratory* (toPSAil), herein referred to as *the simulator*, was developed by researchers from *Scott Research Group* in *Chemical and Bio-molecular Engineering department* at *Georgia Institute of Technology* under the support from *Rapid Advancement in Process Intensification Deployment* (RAPID) *Manufacturing Institute's Center for Process Modeling* (CPM).
- This is the beta version (release date: 4/30/2021) of the simulator. More capabilities will be added continuously throughout 2021. The purpose of this user manual is to help enhance the user experience in simulating a variety of gas separation applications involving fixed bed pressure swing adsorption (PSA) process technology. This user manual is intended to be a short document primarily focusing on how to use the simulator for the aforementioned use cases.
- The simulator is being developed at the Georgia Institute of Technology in the department of Chemical and Bio-molecular Engineering (ChBE) by Dr. Joseph K. Scott and Taehun Kim and is owned by Georgia Tech Research Corporation. The work was supported by RAPID-CPM led by Dr. Chau-Chyun Chen, Dr. Maximilian B. Gorensek, and Dr. Joseph K. Scott.
- The simulator is available to academic research and noncommercial purposes within the RAPID Community for free. Any external publications of the disclosed models are prohibited until the IP generator publishes corresponding research papers in a public domain.
- Any commercial usage of the simulator will require the member to negotiate a non-exclusive license from Georgia Tech Office of Technology Licensing (GT-OTL). For information on the licensing, please contact Dr. Terry Bray, the director of GT-OTL, at terry.bray@industry.gatech.edu.

Contents

---

1. Service Information

Please contact the following individuals for the product inquiry and support:

- Taehun Kim (Graduate Student): taehun.kim@gatech.edu.
  - contact for scheduling a training session
- Joseph K. Scott (Associate Professor): joseph.scott@chbe.gatech.edu.
  - principal investigator

2. System & Software Requirement

In principle, the simulator will run with a little to no problem when the following hardware specifications are available:

- a multiple core CPU (e.g. Intel core i5 equivalent or better)
- more than 8 GB of *free* RAM memory
- Windows or Mac OS (Linux does not support Microsoft® Excel®)
- At least 2 GB free disk space after the installation

We recommend using Windows as the operating system because the development work was carried on a Windows machine. However, Mac OS should also work without trouble as well. For the software requirement, we do have the followings:

- Microsoft® Excel®: please enable the macro function
- MathWorks® MATLAB®: no additional toolbox is needed

For the software programs, please use the latest versions as much as possible to receive the most up-to-date supports from the vendors.

The simulator performs demanding computations during the course of running a simulation. Therefore, you should not multitask when running a simulation.

3. Simulation Package

In this section, we provide an overview on the simulation framework employed by the simulator. We briefly review the overall framework in terms of the model and sub-models that are included in the simulator. Also we provide a big-picture view on a typical workflow in using the simulator.

3.1. Bird's-eye View

From Figure 1, we observe that the simulator is equipped with four main unit blocks including sub-models with a varying degree of complexity. The main physics of a fixed-bed adsorber is captured with the most details. Interactions between multiple units in a simulation are also well described. In contrast, pressure changers (e.g. valve, pump, or compressor) are modeled with the least details among the available unit models.

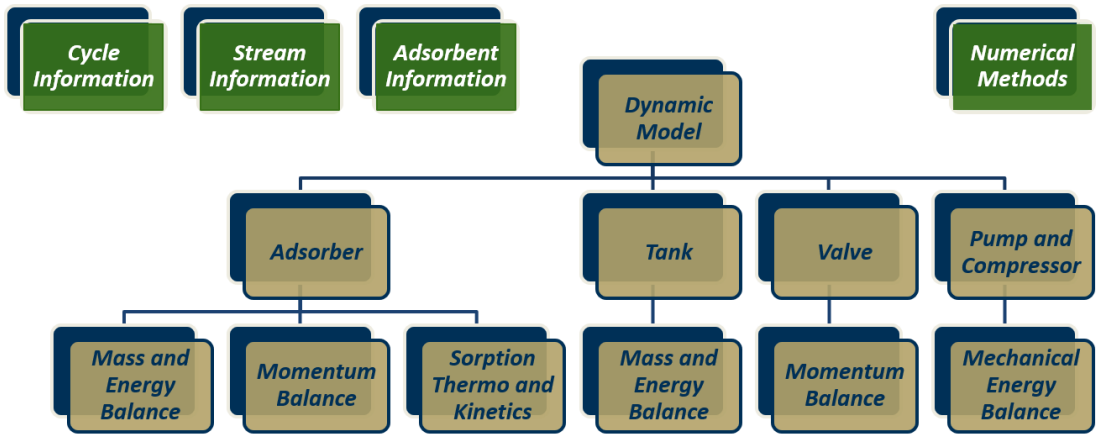

Figure 1: A bird's-eye view of the simulation framework

It is also important to recognize the green blocks in Figure 1; these blocks represents user-inputs to the simulation environment and numerical methods employed to simulate

PSA processes. The user must decide which adsorbent-adsorbates system to simulate, how to design and operate a PSA process and cycle, and a desired accuracy of the numerical approximation to the solution given by the simulator.

3.2. Summary of Models

In this section, we introduce the CSTR model and show how it is used to approximate the dynamics of adsorption columns and tanks. In a nut-shell, an adsorption column is modeled as a series of sequenced CSTRs. This is motivated from the fact that if you take the number of CSTRs to approach infinity, we practically have a plug flow reactor (PFR) behavior. In contrast to and adsorption column, each tank is modeled using a single CSTR.

We would like to emphasize that this *user-manual* is intended to be a tutorial document focusing on how to use the simulator that is contained in the released package. Nevertheless, we cover a basic modeling information in this document to enhance the user's understanding on the working principles of the simulator. Therefore, we omit a lot of details and leave the readers to refer to our upcoming publications regarding the development of this simulator. Finally, we note that all the models that are implemented inside the simulator are *nondimesionalized* to enhance numerical stability and efficiency.

3.2.1. Continuously Stirred Tank Reactor

We are about to describe a single continuously stirred tank reactor (CSTR) model derived from the first principle modeling methodology involving the conservation laws. We present a schematic diagram of a CSTR as shown below in Figure 2.

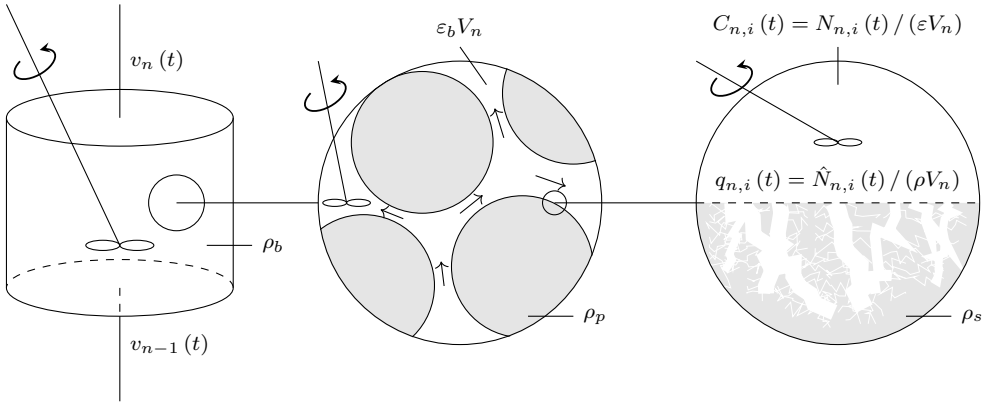

Figure 2: Depiction of nth CSTR and its internal; $\forall n \in \mathcal{N} = \{1, ..., n_c\}$ and $\forall i \in \mathcal{I} = \{1, ..., n_s\}$. ε and ρ are general and specified based on the controlling resistance.

For each CSTR, the rate of change in the gas phase concentration of species i is described by (1) as below:

$$\frac{dC_{n,i}(t)}{dt} = C_{n-1,i}(t)\frac{v_{n-1}(t)}{\varepsilon V_n} - C_{n,i}\frac{v_n(t)}{\varepsilon V_n} - \rho\left(\frac{1-\varepsilon}{\varepsilon}\right) r_i\left(\mathbf{C}_n(t), \mathbf{q}_n(t)\right). \tag{1}$$

Practically, the first two terms on the right hand side describes the convective flow in and out of a given CSTR for a co-current flow direction. Also, the last term on the right signifies the rate of adsorption inside the nth CSTR. The balance equation can easily be applied for a counter-current flow direction by manipulating the sub-scripts for the indices.

In case of adsorption column, a CSTR has mass exchange going on inside itself due to the adsorption of species. In case of tanks, however, since there are only void spaces, no adsorption happens internally. To describe the mass exchange between adsorbed and gas phases, we may write any explicit form of rate expression as below:

$$\frac{dq_{n,i}(t)}{dt} = r_i\left(\mathbf{C}_n(t), \mathbf{q}_n(t)\right). \tag{2}$$

For instance, the right hand side of (2) can be described as a linear driving force (LDF) type rate model where we have $r_i\left(\mathbf{C}_n(t), \mathbf{q}_n(t)\right) = k_i\left(q_i^*\left(\mathbf{C}_n(t)\right) - q_{n,i}(t)\right)$. Here, k_i is the mass transfer coefficient, $q_i^*\left(\mathbf{C}_n(t)\right)$ is a hypothetical adsorbed phase concentration of species i in equilibrium with the current gas phase inside nth CSTR, and $q_{n,i}(t)$ is the current adsorbed phase concentration of species i in nth CSTR.

When it comes to heat exchange modes that are associated with a single CSTR, we first consider an interior heat effects. In this case, the main underlying assumption of the model is that everything inside a CSTR is well mixed and there exists an instantaneous thermal equilibrium between different phases. The rate of change in the temperature of nth CSTR is given by the following:

$$\begin{aligned}\frac{dT_n(t)}{dt} =& \frac{1}{C_{p,n}(t)}\left[\varepsilon V_n\frac{dP_n(t)}{dt} + (1-\varepsilon)V_n\rho\sum_{i=1}^{n_s}\left(q_i^{st}\left(\mathbf{q}_n(t), T_n(t)\right)\frac{dq_{n,i}(t)}{dt}\right)\right.\\ &\left. + \frac{dQ_n(t)}{dt} - \sum_{i=1}^{n_s}\Delta h_{n,i}(t)\, C_{n-1,i}(t)\, v_{n-1}(t)\right],\end{aligned} \tag{3}$$

where $C_{p,n}(t)$ is defined as below:

$$\begin{aligned}C_{p,n}(t) \equiv& \left[\varepsilon V_n\sum_{i=1}^{n_s} C_{n,i}(t)\, C_{p,f,i}\left(T_n(t)\right)\right.\\ &\left. + (1-\varepsilon)V_n\rho\left(\sum_{i=1}^{n_s} q_{n,i}(t)\, C_{p,f,i}\left(T_n(t)\right) + C_{p,s}^{'}\left(T_n(t)\right)\right)\right],\end{aligned} \tag{4}$$

and $\Delta h_{n,i}(t)$ is defined as below:

$$\Delta h_{n,i}(t) \equiv \int_{T_{n-1}(t)}^{T_n(t)} C_{p,f,i}(T)\, dT. \tag{5}$$

Now, an important aspect to take account is modeling heat effects in a CSTR wall of a given thickness. The heat transfer between cell-interior-to-cell-wall and cell-wall-to-environment is modeled with a linear driving force assuming respective heat transfer coefficients, as depicted in Figure 3 as below:

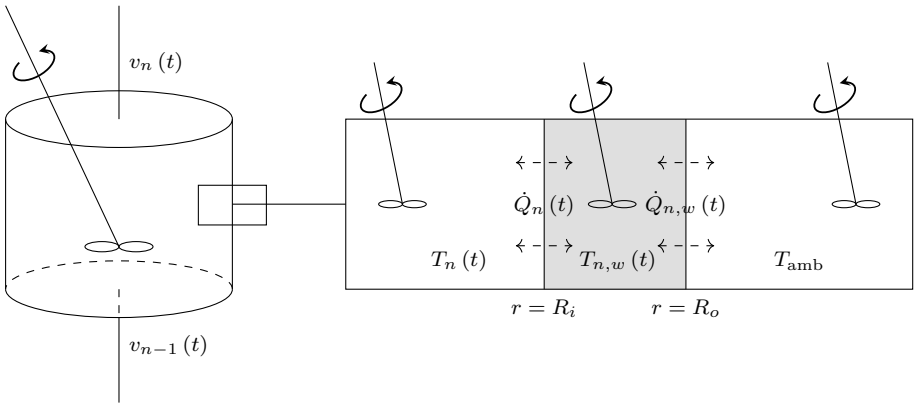

Figure 3: Depiction of heat transfer mechanism inside nth CSTR. Heat accumulates at the wall and convective heat transfer happens at the outer wall.

Note that heat will accumulate inside the column wall. This requires an additional energy balance over the column wall. To do so, let us first consider the heat transfer between the bulk gas phase inside CSTR to the column wall.

$$\frac{dQ_n(t)}{dt} = h_i A_{n,i}\left(T_{n,w}(t) - T_n(t)\right), \tag{6}$$

where $A_{n,i} = 2\pi R_i L_n$, and h_i represents an interior convective heat transfer coefficient.

To describe the heat transfer between the column wall and the environment, we also recourse to the convective heat transfer relationship.

$$\frac{dQ_{n,w}(t)}{dt} = h_o A_{n,o}\left(T_{\text{amb}} - T_{n,w}(t)\right), \tag{7}$$

where $A_{n,o} = 2\pi R_o L_n$, T_{amb} represents an ambient temperature, and h_o represents an exterior convective heat transfer coefficient.

Now, by doing a simple energy balance over the column wall (i.e. a closed system depicted in Figure 3), we may write the following balance equation:

$$\frac{dT_{n,w}(t)}{dt} = \frac{1}{\rho_w C'_{p,w} A_w L_n} \left(\frac{dQ_{n,w}(t)}{dt} - \frac{dQ_n(t)}{dt} \right), \tag{8}$$

where $A_w \equiv \pi \left(R_o^2 - R_i^2 \right)$, ρ_w is the wall density, $C'_{p,w}$ is the wall heat capacity, and L_n is a height of nth CSTR.

Now that we have clearly described the model for a single CSTR, we are ready to discuss more complicated models involving one or more of the CSTRs.

3.2.2. Fixed-bed Adsorber

The governing equations for a single column take the form of partial differential equations (PDEs) describing variations of the column states with respect to time and axial dimension. Our models approximate the column as a sequence of well-mixed tanks (i.e., CSTRs) aligned in the column axial dimension (Figure 2). As a consequence, the resulting models take the form of coupled systems of differential algebraic equations (DAEs) that are analogous to a simple first-order finite-volume discretization of the governing PDEs.

The most important feature of the adsorption column model is determining how to solve for the unknowns for the DAEs system. The unknowns of the system are the volumetric flow rates associated with the CSTRs. By imposing assumptions such as (i) axially constant and time invariant pressure or (ii) axially constant but time variant pressure, we can derive algebraic equations that can be solved for the unknown volumetric flow rates.

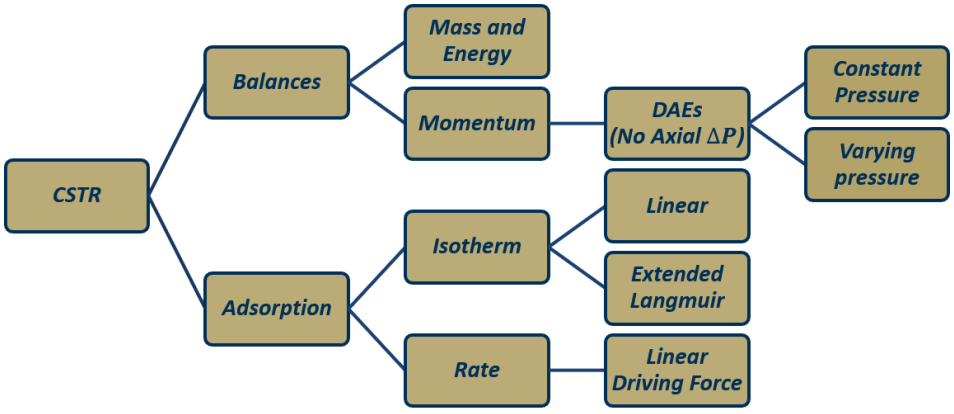

Figure 4: A summary of supported models for an adsorption column

The set of ODEs from the conservation laws and the algebraic equations from the first assumption, i.e., (i), form a constant pressure DAE model; These models are useful for modeling high pressure feed or low pressure purge steps. Similarly, the set of ODEs from the conservation laws and the algebraic equations from the second assumption, i.e., (ii), form a

time varying pressure DAE model; these models are useful for modeling re-pressurization, depressurization, or equalization steps.

The summary of currently supported models for fixed-bed adsorber is Figure 4.

3.2.3. Feed and Product Receiver Tanks

Feed and product receiver tanks are modeled with a single CSTR respectively. Basically, the conservation laws for a single CSTR described in Section 3.2.1 hold true except that we do not need (2) for the tanks. Typically, a tank has much larger volume than a single column void volume, i.e. $\varepsilon V_c/V_t << 1$. This is important because if tank volume is too small, than we may not have enough product or feed gas to re-pressurize an adsorption column.

For the summary of currently supported models for feed and product receiver tanks, please refer to below.

Figure 5: A summary of supported models for a feed or product receiver tank

3.2.4. Pressure Changers

The pressure changers (e.g. valves, pumps, or compressors) are not explicitly modeled in the simulator. Pressure changers simply achieve the desired pressure change. In other words, before and after a pressure changer, no physical properties of the stream changes. Instead, we do compute isentropic compression or vacuum energy consumption as a part of our model. Finally, for the current version of the simulator, we only support a linear valve which relates the molar flow rate across the valve by a valve constant and the pressure difference, i.e. $F(t) = C_v \Delta P(t)$.

For the summary of currently supported models for the pressure changers, please refer to below.

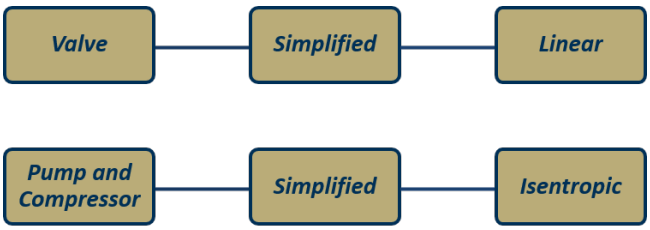

Figure 6: A summary of supported models for pressure changers

4. Simulation Environment

Using our simulator, we can simulate a PSA process with one or more adsorption columns connected with a feed tank and a product tank through a network of pipes, valves, and pumps. Practically, the simulation environment represents a real world PSA process system closely by mimicking realistic features of a such system for a given gas separation application. *The goal of this section is to get the user familiarized with the actual conceptual picture of the system that we are simulating.*

4.1. Generality

Some of the most notable generality of the simulation environment can be summarized as below:

- uni-bed and poly-bed process simulations are possible
- a PSA cycle can be organized for a given process and we support the following steps:
  - re-pressurization (w/ feed or product)
  - de-pressurization (on either ends)
  - pressure equalization (on either ends)
  - constant pressure steps (high pressure feed or low pressure purge)
- without loss of generality, feed stream containing any number of components can be simulated
- an arbitrary number of adsorption columns can be simulated (but cycle organization gets more difficult)

On top of this, the simulation framework is flexible in that, while it supports a limited set of standard isotherm and rate models, the framework allows us to easily add different sub-models. More information on the model selection can be found later in Section 5.

4.2. Process Flow Diagram

Figure 7 shows the overall process flow diagram (PFD) for the PSA process system that we are simulating. We are about to describe a detailed features of each units on the PFD. Correctly understanding the PFD will enhance the user experience when using the simulator to run a simulation on various PSA processes.

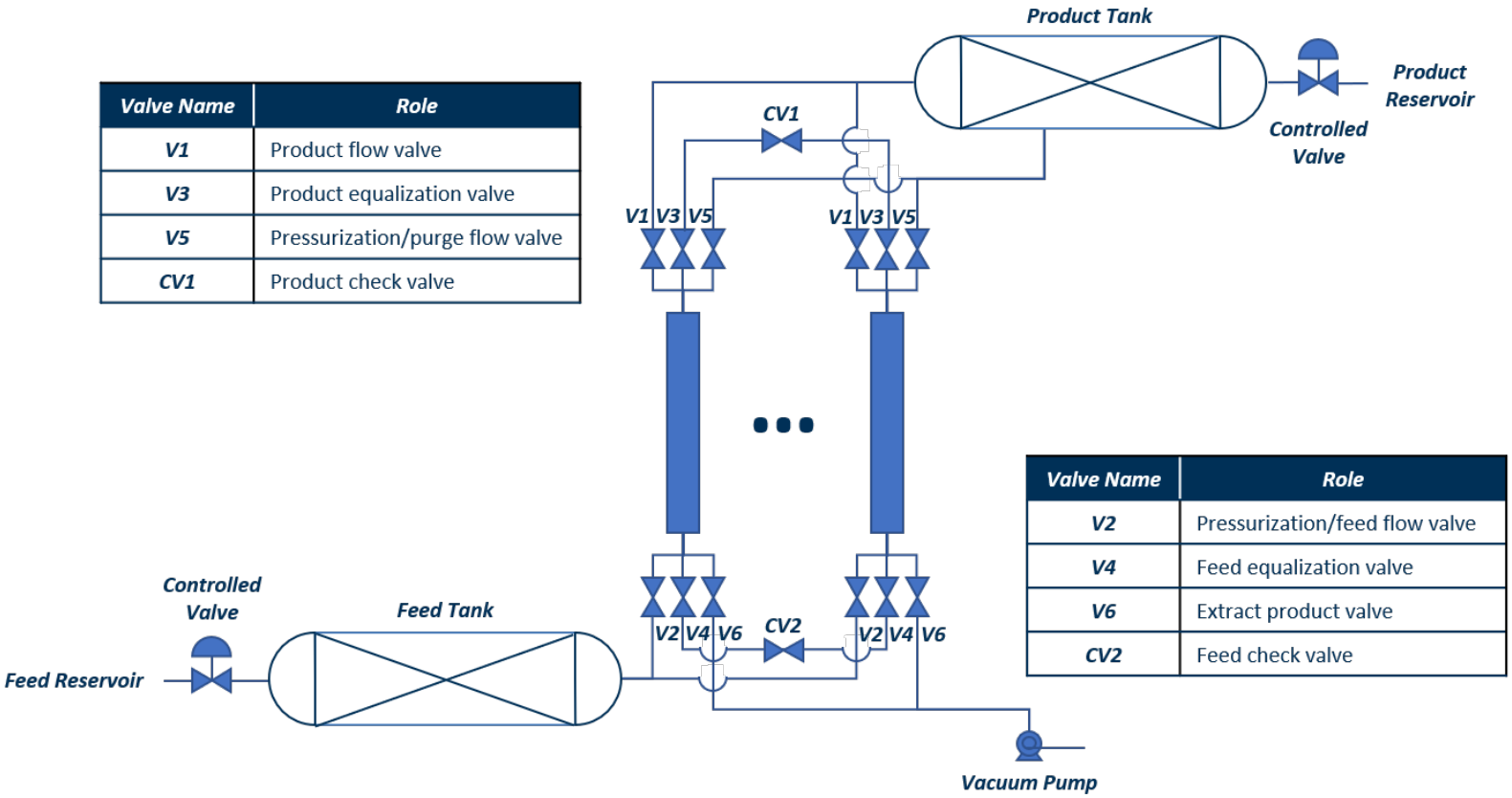

Figure 7: A general pressure swing adsorption (PSA) process simulation process flow diagram

4.2.1. Adsorption Columns

There are 6 valves per an adsorption column. For each end of and adsorption column, only one or less number of valves can be opened at the same time. The different purpose and functionalities of the valves are summarized in the tables in Figure 7. Nevertheless, it is important to note that valve 3 and 4 can have flows in both directions while all other valves are uni-directional; valve 1 and 2 allows co-current flows only as opposed to valve 5 and 6 which allows counter-current flows only.

It is also important to understand the importance of check valves. They are placed in between adjacent adsorption columns, in case there are more than one adsorption columns. The check valves prevent any unwanted interactions between adsorption columns in the network of the PSA process system and assumes a value of either 0 (closed) or 1 (open).

4.2.2. Feed and Product Receiver Tanks

Note that the mass flow rates into the feed tank and out of the product tank are regulated by respective mass flow controllers. For the feed tank, the pressure is always maintained at a pressure higher than any pressure levels that would ever be reached inside any adsorption column or the product tank; we name the feed pressure as P^f and $P^f \geq \hat{P}$ where $\hat{P}$ is any other pressure in the system.

For the product tank, the valve located at the exit of the product tank is fully opened when the pressure inside the product tank (P^p) becomes equal to the high pressure of the adsorption column (P^h), i.e. $P^p = P^h$. Then the valve controls the outflow from the product tank to maintain the high pressure inside the product tank.

4.2.3. Pressure Changers

While it is important to consider pressure changes in the feed, raffinate, and extract streams, no pressure changers, e.g. pumps or compressors, are explicitly modeled. For the case of the feed compressor, we calculate the energy consumed based on the assumption that the feed has been compressed from a standard pressure (P°) to the feed pressure (P^f). For the vacuum pump for the extract stream, we also just compute the energy consumed based on the pressure difference between the current adsorption column pressure to the low pressure.

4.3. Cyclic Steady State Convergence

At the beginning of each PSA cycle, except for the very first cycle, we may compute the difference between the current initial condition vector ($\mathbf{x}_t$) and the previous initial condition vector ($\mathbf{x}_{t-1}$), i.e. $\mathbf{x}_t - \mathbf{x}_{t-1}$. Then, by taking a measure of distance, typically a squared $L2$ norm, we may assess the cyclic steady state (CSS) convergence of our simulation at a current iteration t. In other words, for some positive number ε, typically $\varepsilon \approx 10^{-9}$, we can check to see if the following condition holds true:

$$e_t := \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^2 < \varepsilon. \tag{9}$$

While it is possible to take the entire initial condition vector and compute the CSS convergence, we could also pick state variables related to adsorption columns and compare those as well to determine the CSS convergence status. Typically, if the simulation is well put together, then, e_t in overall trend will continuously decrease in its value as the cycle number increases.

4.4. Performance Metrics

Some of the main outputs that one can obtain from the simulation environment are the process performance metrics. Accurate evaluation of these metrics are critical when it comes to designing PSA processes and comparing them with conventional or alternate gas separation processes. In this section, we clearly define all important process performance metrics that the simulator is capable of calculating as simulation outputs.

One important notion to realize here is that process performance metrics can be evaluated in a cycle basis. In other words, at the end of each PSA cycle, we can readily evaluate performance metrics based on the cumulative amount of gas that were processed or harnessed, as well as total amount of energy used to either compress or vacuum the gas in the system. We reserve more detailed definition of the performance metrics in our publications but provide a simpler definition in this user-manual.

4.4.1. Product Purity

From Figure 7, we see that we can either collect the product from the top (i.e. raffinate product) or from the bottom (i.e. extract product). The product purity of species i in a given product stream is simply defined as a mole fraction of species i in that stream.

4.4.2. Product Recovery

From Figure 7, we see that we have a flow coming into the system from the feed tank. By dividing the amount of product of species i *produced* in a given PSA cycle by the corresponding amount of species i *processed* or *fed*, we obtain the product recovery fraction of species i.

4.4.3. Productivity

The total amount of species i recovered in a given PSA cycle divided by the cycle time is defined as the productivity of species i in a given PSA cycle.

4.4.4. Energy Consumption

The total amount of species i recovered in a given PSA cycle divided by the total energy expenditure from the sum of compression and vacuum is defined as the energy consumption of species i in a given PSA cycle.

5. User Interface

In this section, we cover the A-to-Z on how to use the simulator to run a supplied library of simulations in this model release. From the folder-tree structure, input specification, to actually running a simulation, we concisely provide a step-by-step instruction on how to use the simulator.

5.1. Folder-tree

Perhaps, the most important message here is that *the user should not attempt to change any codes written in saved in .m-files!* The simulator is fully capable of running simulations with different user-specified inputs without having to change any pieces of codes that are released. Nevertheless, we have the following folder-tree for the main folder.

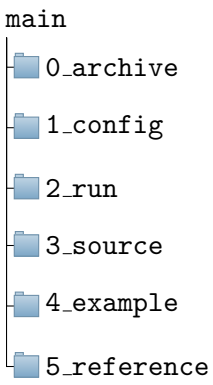

Figure 8: Main folder tree

We again note that the user should only be concerned with 2_run, 4_example, and 5_reference where the last folder contains helpful references. In the remainder of this section, we introduce folder hierarchy for the run folder (Section 5.1.1) as well as the example folder (Section 5.1.2).

5.1.1. Run Folder

First, we present the overall folder structure of the 2_run folder shown in Figure 9.

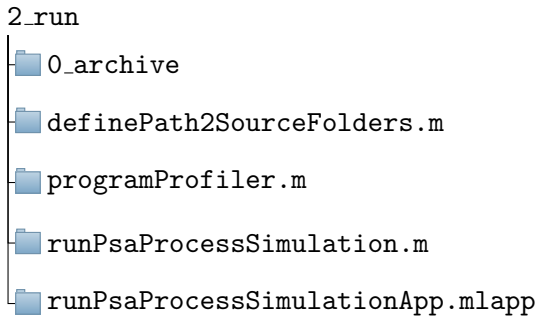

Figure 9: Run folder tree

defineSourceFolders.m is a MATLAB® function that adds relevant paths for the folders containing source codes and example files located inside the main folder. Therefore, the user should not worry about opening up the file. Also, programProfiler.m is used only for profiling the performance of the code and the user should not worry about it as well.

Now, runPsaProcessSimulation.m is a .m function file that is used to run a user specified simulation using the simulator. Note that the function requires an input which should be a string variable denoting the directory of the excel file (specifySimulationParameters.xlsm) that contains input parameters required for a given simulation. The user should not be required to run an instance of the function file. Instead, the user is directed to use GUI for the purpose of running a simulation by specifying a folder directory where the example excel spreadsheet is located and simulation outputs will be saved. More on this will be covered in Section 5.3.

5.1.2. Example Folder

First, we present the overall folder structure in 4_example folder shown in Figure 10. In terms of the folders inside 4_example, we have three folders as the following: 1_breakThroughExp, 2_valvePsaCycle, and 3_valveFreePsaCycle. Note that 3_valveFreePsaCycle is *work in progress* and is not a part of the current release of the simulator.

Perhaps, the most important information to convey here is that each of the example folders contains specifySimulationParameters.xlsm file with a cycle *pre-configured* for the user for a particular PSA process or a breakthrough experiment. Therefore, the user simply has to supply basic information about the system without having to worry about constructing a PSA cycle or a breakthrough simulation oneself. More details on how to specify simulation

parameters can be found in Section 5.3. Also, customization of a simulation through the input excel file is covered in Section 8.

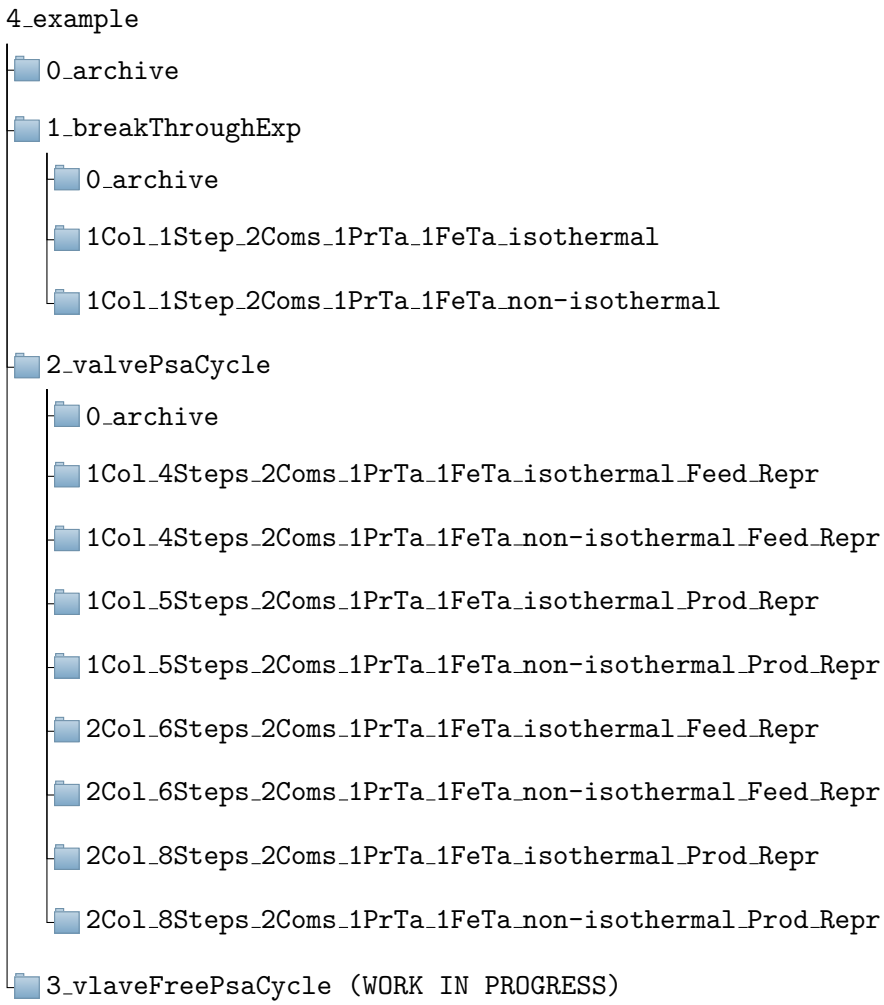

Figure 10: Example folder tree

Inside 1_breakThroughExp, we have two folders created for running an isothermal and a non-isothermal version of the fixed bed adsorber breakthrough experiment simulation respectively. A breakthrough simulation by definition is a single column and single step PSA cycle where the adsorption column is undergoing a high pressure feed step. The initial condition of the column can be chosen appropriately by the user and the simulation proceeds until a breakthrough happens at the product-end of the adsorption column.

Inside 2_valvePsaCycle, at the moment, we have 8 sub-folders that each contains specifySimulationParameters.xlsm file with a pre-configured PSA cycle. Also, from the sub-folder names, it is easy to identify the number of steps, the number of components, the number of feed tanks, and the number of product tanks. Furthermore, the sub-folder names also indicate whether the simulation is configured as isothermal vs. non-isothermal and from which end of an adsorption column the re-pressurization is happening.

We stress that putting together a PSA process with a working PSA cycle is a non-trivial job for any PSA process simulators whether they are open sources or commercial ones. Therefore, we went ahead and provided a pre-configured PSA cycle setups for the user so that the user of the simulator has a solid starting point for further exploring a PSA simulation with different input parameters.

5.2. Workflow

In this section, we discuss a typical work flow of using the simulator. An important part of the work flow involves: specifying simulation inputs, running a small bits of simulation using the simulator, tuning inputs, obtaining simulation outputs, and analyzing the results. We present the overview of the work flow in the following diagram as below.

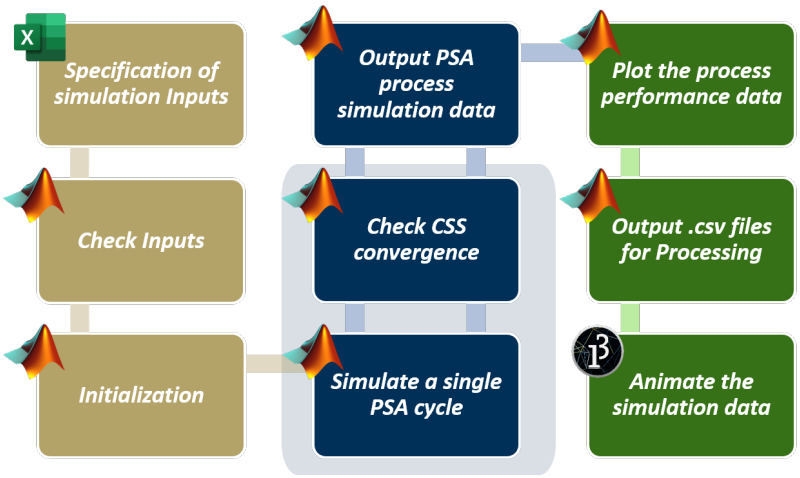

Figure 11: A schematic workflow diagram for using the simulator

From Figure 11, once the user specifies an input, then the simulator checks for the correctness of the supplied inputs. Also, based on the user supplied information, the simulator makes high level decisions regarding which sub-models to use, how to configure a PSA cycle, types of events that would decide termination of each step, and so on. Indeed, the first three blocks of Figure 11 make sure that integral steps are taken cared so that the simulator can function properly.

Once initialization finishes, the simulator will simulate a user specified number of PSA cycles. We recommend the user to simulate only a handful number of PSA cycles for the first pass. This is because there inevitably will be a trial-and-error procedure tuning relevant input parameters to obtain a desired performance of a given PSA process. As the simulation proceeds, the simulator checks the CSS convergence at the end of each PSA cycle. If a CSS is attained before the specified number of cycles, then the simulation terminates early.

After when the simulation terminates, we obtain outputs in the forms of either plots or .csv data files. For the plots, the user can specify options inside specifySimulationParameters.xlsm file in a desired example folder for the current simulation. Also, at the end of ths simulation the .csv files will be saved inside *the same directory* as where the corresponding specifySimulationParameters.xlsm file is located. Finally, we note that another future model release will include a data visualization tool for reading in those .csv files and converting those to animations that can be use to diagnose and compare dynamics and performance of a given PSA cycle to the other.

5.3. Specifying Simulation Inputs

As shown in Figure 11, to use the simulator, we see that the very first thing the user has to do is to specify inputs to the simulator. Doing so is exclusively taken care by the specifySimulationParameters.xlsm file located in the sub-folders inside the example folder; please refer to Section 5.1 for more details. Now, go ahead and open one of the sub-folder which contains a PSA process of your choice and open specifySimulationParameters.xlsm. We will walk through how to specify all the inputs using specifySimulationParameters.xlsm in the following sections.

5.3.1. Checklist

When specifying simulation inputs using specifySimulationParameters.xlsm, be sure to check out the following items and make sure that we are on the safe side.

- please enable visual basic (VBA) macro feature inside Microsoft® Excel®
- before running a simulation, make sure to save any changes you made inside specifySimulationParameters.xlsm

VBA macro is needed to initialize, push, or clear data to MATLAB®. Also, if you do not save changes made in specifySimulationParameters.xlsm, the information you supplied will not be read by MATLAB®. Other than this, there really shouldn't be too much difficulties when specifying the inputs as long as you follow the instruction correctly; we will try our best to provide a clear instruction.

5.3.2. Color Codes

When specifying inputs in specifySimulationParameters.xlsm, throughout the process, please keep in mind that the name of the game here is to provide inputs for the yellow highlighted cells. Once you specify an input to a yellow highlighted cell, the cell will turn into green in color; please refer to Figure 12 and Figure 13.

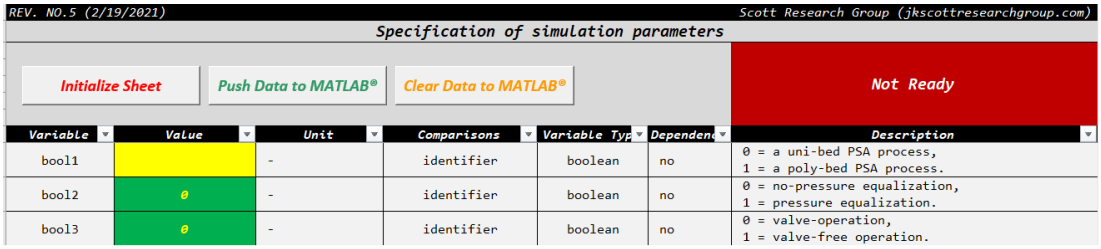

Figure 12: An example of color code: the tab is not fully specified specifySimulationParameters.xlsm; the overall status is not ready (red)

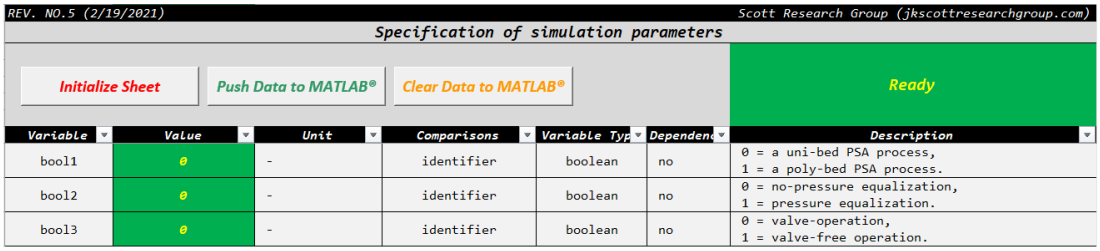

Figure 13: An example of color code: the tab is fully specified specifySimulationParameters.xlsm; the overall status is ready (green)

5.3.3. Input Types

Each input type for the inputs to the simulator is represented by each tab in specifySimulationParameters.xlsm. The same color code rules apply here; yellow tab means the tab needs all the inputs specified. We provide the following big picture summary for the tabs as below:

- **Models**: select the scope of the simulation to be performed along with appropriate sub-models used in the simulator
  - make sure that the boolean variable takes a value of 0 or 1
  - make sure that the model specification takes a zero or a positive integer value
  - be careful not to choose unsupported models or scope of simulation

- **Plotting**: select plots to be generated at the end of the simulation
  - adjust as needed for the simulation
- **Numerical Methods**: define parameters related to numerical methods
  - a recommended value for the number of CSTRs is 200 (i.e. nVols = 200)
    - increasing the number of CSTRs increases computational time
    - decreasing the number of CSTRs intensifies numerical diffusion
    - a set number of CSTRs within a mass transfer zone is needed
  - increasing the number of time points for each step can slow down a simulation
- **Natural Constants**: define parameters for natural constants
  - there are no reasons to change any values in this tab
- **Adsorbent Properties**: define parameters for adsorbent physical characteristics, equilibrium, and kinetics
  - please make sure that the units are consistent and are correct
- **Cycle Organizer**: define information needed to synthesize a working PSA cycle
  - please carefully follow the instructions in the description column
  - when specifying multiple numerical values as string variables, make sure that there is an empty space between the numbers
  - when specifying a single numerical value as a string variable, provide an equivalent complex number involving imaginary number; e.g. for 1, specify $1 + 0i$ instead
  - for certain definitions, please refer to the "Extended Definition" tab
- **Packed Bed Properties**: define physical properties of a packed bed adsorber
  - please make sure that the units are consistent and are correct
- **Tank Properties**: define physical properties of a tank
  - please make sure that the units are consistent and are correct
- **Stream Properties**: define parameters for the feed and the raffinate stream
  - please make sure that the units are consistent and are correct

5.3.4. Saving the Inputs

Once all the yellow cells are converted to green, we should have all the tabs still remaining in yellow while the status bar on the top right corner of all tabs should be green with *ready* sign now. Then, click ***Push Data to MATLAB®*** as shown in either Figure 12 or Figure 13. Once the macro finishes, you should be able to see a pop up message saying ***Data was pushed to MATLAB® successfully!***. Also, all the tabs should be green now.

If you made mistake or you wish to specify a new input, then you may push ***Clear Data to MATLAB®*** button. The macro will clear up any previously pushed data and will turn the tabs back to yellow, indicating that you are free to edit or change input parameters. Once you are ready, you can hit ***Push Data to MATLAB®*** again.

The last and the most important step in the input specification is to actually save the spreadsheet by either hitting the floppy disc icon on the top left corner or going to *File* menu and saving the file. This will ensure that the simulator will be able to read in all the inputs that you've specified.

5.4. Running a Simulation

Please recollect from Section 5.1.1 on where to locate runPsaProcessSimulationApp.mlapp. When you open the app, you will have an instance of MATLAB® opened along with the GUI as shown in Figure 14.

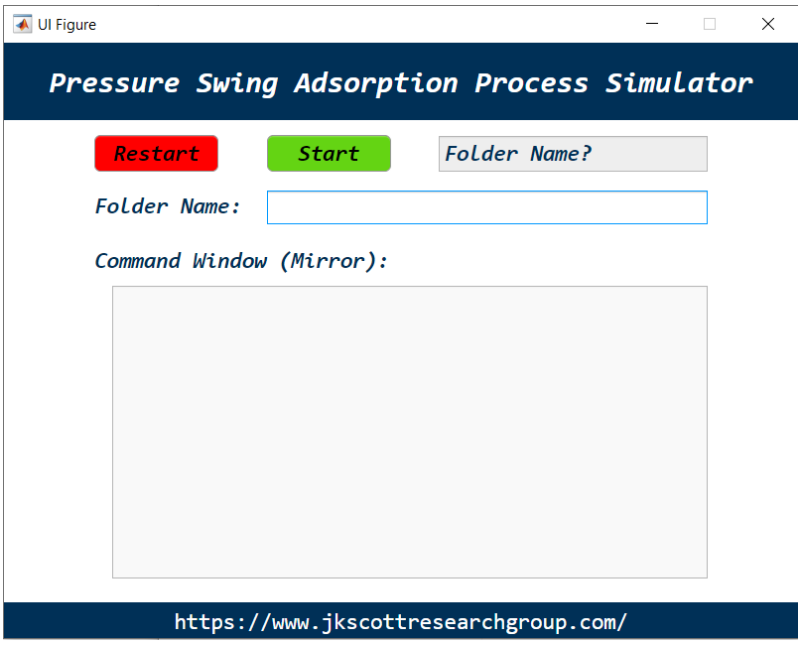

Figure 14: A snapshot of the GUI right after being launched

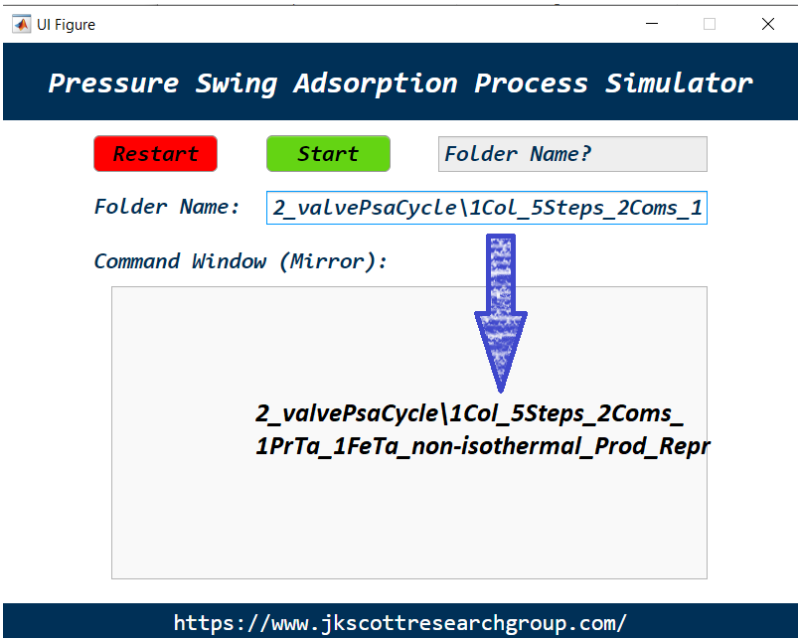

Figure 15: A snapshot of the GUI right after a folder directory is specified

Note from Figure 14 that there are two buttons: *Restart* and *Start*. The *Restart* button closes the current GUI and launches a brand new instance of it; this is useful for the case when GUI freezes or stops working. The *Start* button starts the simulation when pressed. However, the button requires that a proper folder name is supplied.

When it comes to specifying a proper folder name, one can either copy and paste a part of directory *after* 4_example or actually type the directory manually. For example, as shown in Figure 15, the user can specify the following strings into the folder name field: 2_valvePsaCycle_1Col_5Steps_2Coms_1PrTa_1FeTa_non-isothermal_Prod_Repr. Again, we reiterate that we want to specify a part of directory *after* 4_example.

Once you supply the folder name, now, you can go ahead and press the start button on the GUI. Then, the simulation will run and you should be able to see the command window output in the opened instance of MATLAB® as the simulation progresses. At the end of the simulation, you will obtain the following output in GUI, as shown in Figure 16 below.

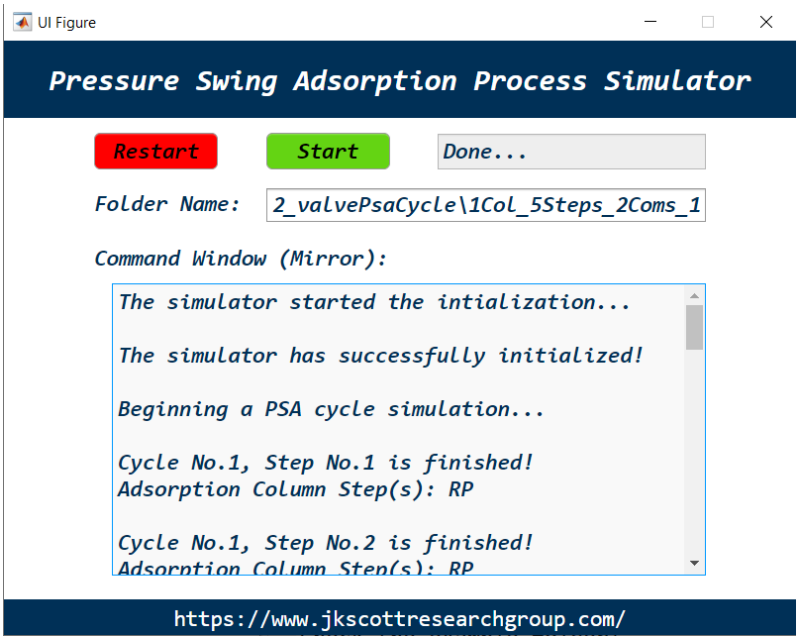

Figure 16: A snapshot of the GUI after a simulation finishes

We highlight that the GUI is definitely something that may be annoying if you are already familiar with MATLAB® to an extent. However, to make sure that we are not changing any internal of the codes written in MATLAB®, we decide to go ahead and implement the GUI. *So, please give the GUI a chance and give it a try!* Lastly, this officially sums up how to run a simulation from the example folder.

5.5. Helpful Tips

There are a few helpful tips that may be useful for running, monitoring, and finalizing the simulation.

- to monitor the progress of the simulation, simply observe the *Command Window* outputs as the simulation runs
- after activating the *Command Window* by clicking on it and pressing ctrl+c will terminate a running simulation
- to gauge if your system is over burdened, follow the steps below (for Windows)
  - press ctrl+alt+delete and open the *Task Manager*
  - go to *Performance* tab and monitor the real time CPU and memory usages
- the saved simulation outputs will overwrite any existing files in the example folder

6. Tuning a Simulation

For any modern day's PSA process simulators, identifying an optimal performance metrics require a kind of trial-and-error simulation procedure. In this section we cover details on how to use the simulator to fine tune the performance metrics under a given operation policy. In this model release, we exclusively give details on how to tune conventional simulations involving valve based operation policy.

6.1. Valve Operation Policy

In this section, we discuss how to identify an optimal operation condition using the simulator, especially using valve operation policy. The valve policy represents an operational paradigm where the user specified valve constants predominantly influences the overall performance of a PSA cycle. In this case, there has to be a careful tuning process with regards to the duration of the step and the valve constant values.

6.1.1. Step Duration or Event

For a given step in a PSA cycle of our interest, we may have a specified value of a predetermined duration from an experiment. For instance the high pressure feed may have taken 100 seconds for the experiment or it took 35 seconds to depressurize an adsorption column. In that case, it is fairly simple to specify the duration of each step; just follow the experimental values to begin with (since it is a feasible PSA cycle) and adjust the duration of the steps as we tune the simulation.

On the other hand, it may be much more simpler to tune the duration when we have an information about certain events that may happen inside a PSA cycle. For instance, we may wish to perform a high pressure feed step until a breakthrough happens whence we stop the step and perform regeneration of the adsorption column. In this case, the duration of each step is determined by the event values and user plays a passive role of determining the event values.

It is the case that there isn't always an explicit event that is relevant for a given step in a PSA cycle. Therefore, we cannot always rely on the event function and we need both techniques to be able to fine tune the performance of a PSA process.

6.1.2. Valve Constants

In this section, we illustrate the importance of tuning the value of a valve constant. Suppose we have a valve constant specified for a depressurization step (i.e. a pressure changing step). Since the goal of the depressurization is to achieve the pressure change within a certain period of time, a reasonable thing to do here is to specify a fixed duration of the step. However, the complication here is that depending on the valve constant value, we may depressurize too much or too less.

In the above case, it is instructive to run the step simulation using an event function. In other words, when the adsorption column pressure reaches a low pressure, we would want to terminate the step. This means, regardless of whatever valve constant value the user may supply, the step will be simulated until the event happens (or it may be that the event did not happen within a specified duration of the step).

In sum, the user should have some sort of preconceived duration of the step in mind. Then, by tuning the input valve constant, the user can tune the simulation of the step so that a desired event take place around at the preconceived duration of the step. This is what we mean by careful tuning of the valve constant to achieve a desired event at a reasonable duration of the step.

6.2. Overall Tuning Process

We propose the following overall tuning process/tips for a given PSA process simulation.

- start with an equivalent isothermal simulation
  - the dependence of adsorption affinity constant on the temperature for non-isothermal simulation makes the simulation much longer to reach a CSS
  - an isothermal simulation can be used first to tune the system to maximize process performance metrics
- start with a known set of duration of the steps in a given PSA cycle
  - one may have experimental results on a working PSA cycle that can be used for this purpose
  - duration can be tuned as needed
- estimate valve constants based on the pressure difference before and after the valve
  - an expression derived for isothermal linear valve model can be used to predict order of magnitude of a valve constant that is needed to achieve a desired pressure change in a given duration of a step

7. Example Simulations

In this model release, we loaded up our example folders with examples with differently organized PSA cycles (maximum of 3 PSA cycles) exclusively using the parameters from Kayser's experimental system[1]. We have done our best to tune the single column simulations. For the multiple column simulations, we recommend running the following example: 2Col_8Steps_2Coms_1PrTa_1FeTa_isothermal_Prod_Repr.

7.1. Kayser's Air Separation Experiment

Herein, we present some of the simulation outputs from running the example simulation stored inside 2_valvePsaCycle \1Col_5Steps_2Coms_1PrTa_1FeTa_isothermal_Prod_Repr. Some of the key features in the simulation setup is described as below:

- a single column 5 step PSA cycle: re-pressurization with product, re-pressurization with feed, high pressure feed, depressurization to a sub-ambient pressure, and low pressure purge
- isothermal operation at 45 degree centigrade with adsorption equilibrium described by linear decoupled isotherm model
- binary separation (oxygen as a light key and nitrogen as a heavy key)
- adsorption rate described by decoupled linear driving force adsorption rate model

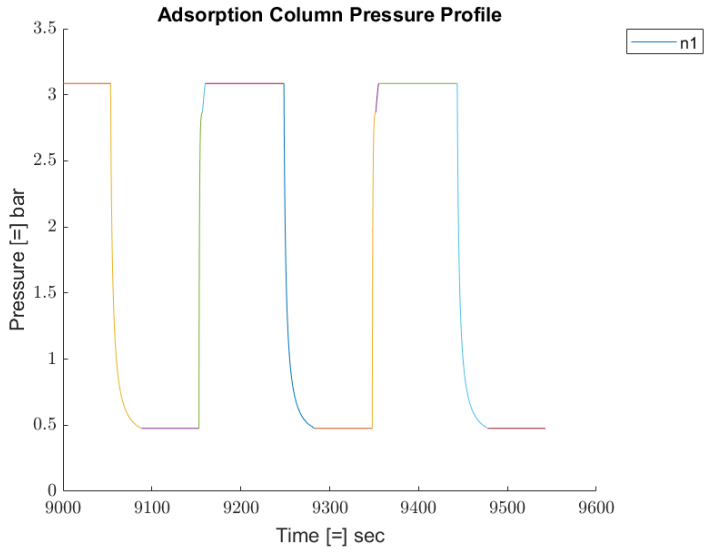

Figure 17: A simulation results on total pressure of an adsorption column during Kayser's PSA experiment

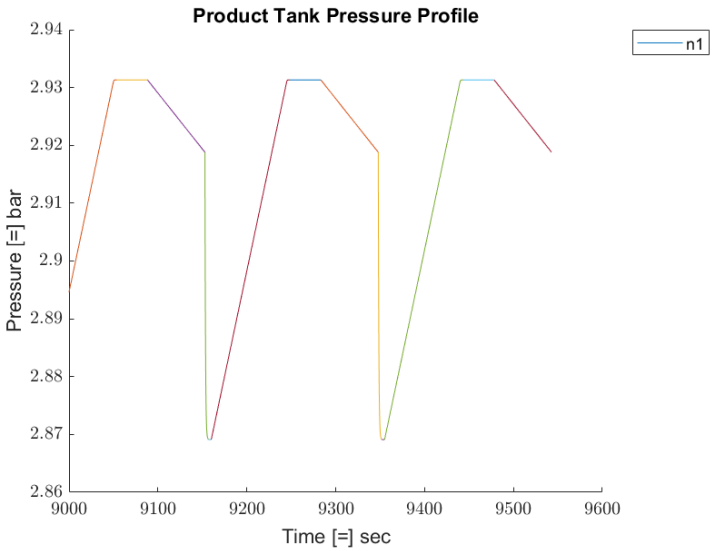

Figure 18: A simulation results on total pressure of a product tank column during Kayser's PSA experiment

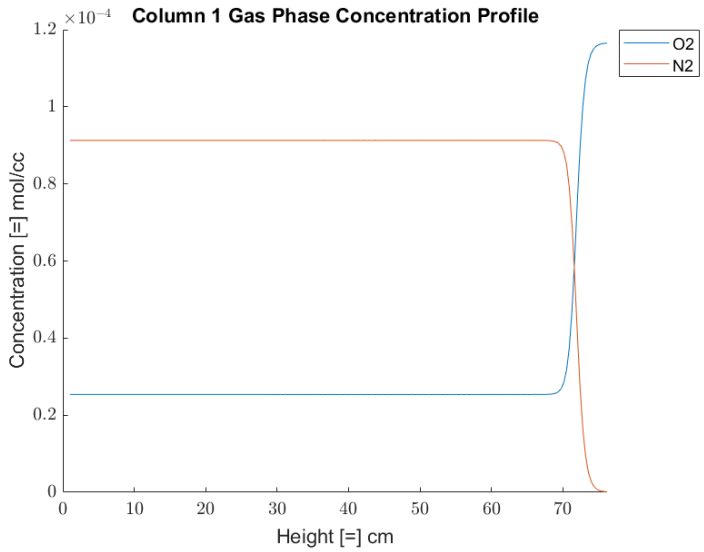

Figure 19: A simulation results on gas phase species concentrations of Kayser's PSA experiment at the breakthrough

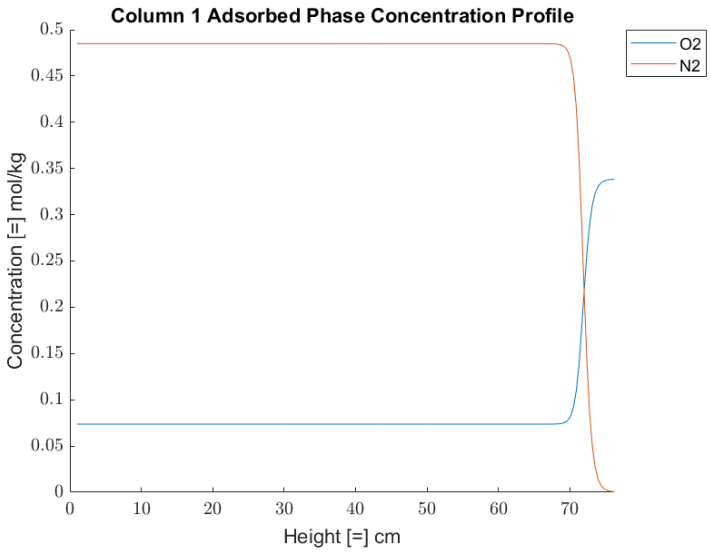

Figure 20: A simulation results on adsorbed phase species concentrations of Kayser's PSA experiment at the breakthrough

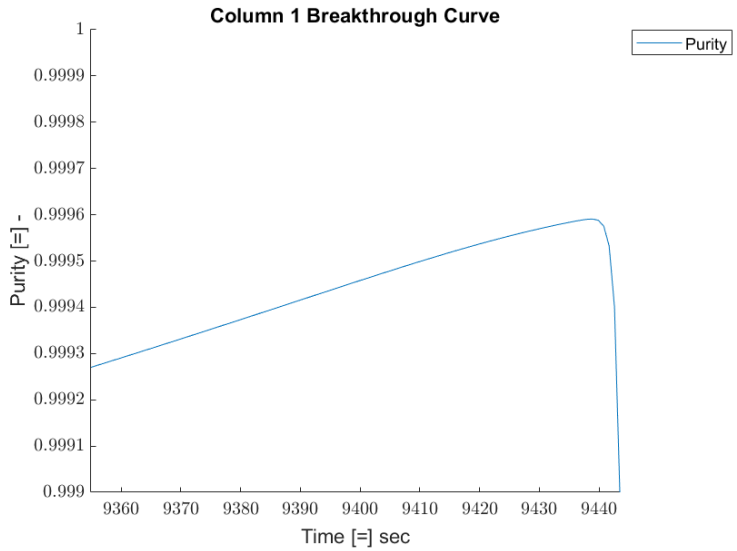

Figure 21: A simulation results on breakthrough during high pressure feed of Kayser's PSA experiment

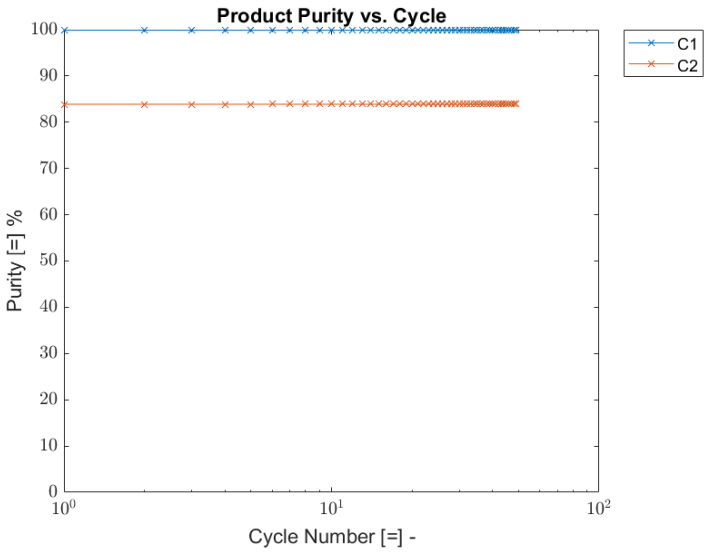

Figure 22: A simulation results on product purity of Kayser's PSA experiment

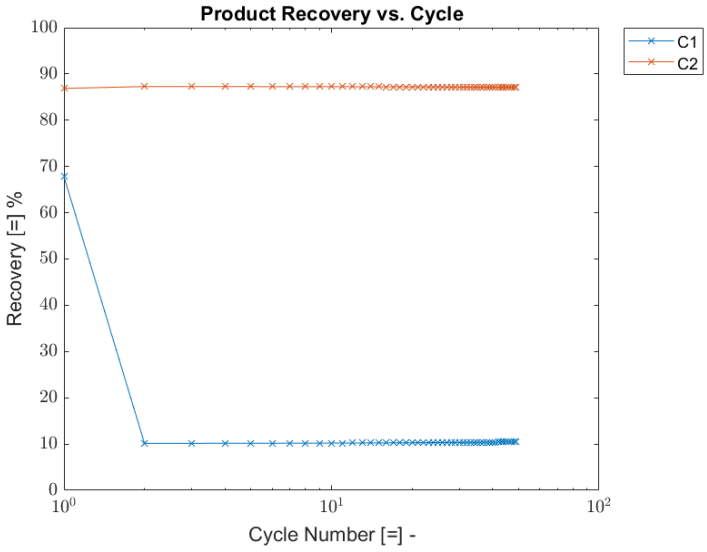

Figure 23: A simulation results on product recovery of Kayser's PSA experiment

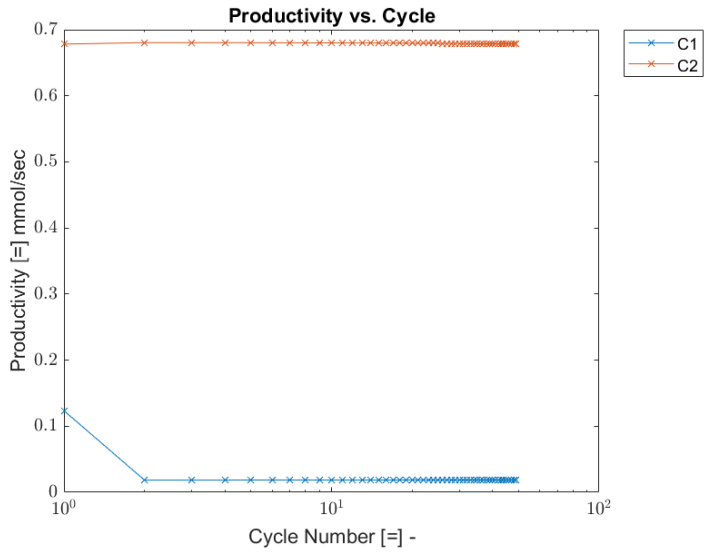

Figure 24: A simulation results on productivity of Kayser's PSA experiment

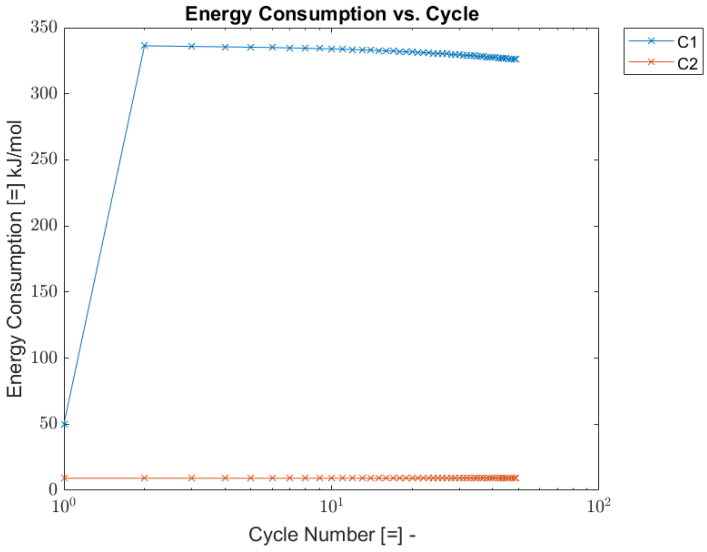

Figure 25: A simulation results on energy consumption of Kayser's PSA experiment

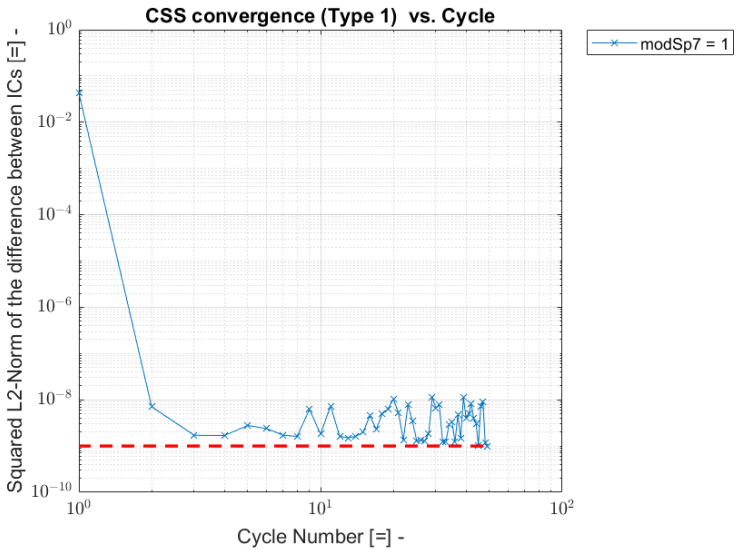

Figure 26: A simulation results on CSS convergence of Kayser's PSA experiment

8. Customizing Simulations

When the user wants to put together new PSA process simulations other than the examples that are supplied, there is a way to do this in our simulator. We can use the excel spreadsheet that and specify the information for the new PSA process simulation accordingly. In this section, we cover the most basic approach on putting together a customized PSA process simulation, using our simulator. Since this section is intended for the advanced users, if you need supports, please contact people shown in Section 1 for a training session.

8.1. Customization Procedure

In the most broad sense, we follow the customization procedure below:

- under the example folder, create a new folder containing a copy of specifySimulationParameters.xlsm
- open specifySimulationParameters.xlsm in the folder that was just created and update the input parameters
- push ***Clear Data to MATLAB®*** button and clear any previous data in the entire workbook
- Initialize tabs that contains different number of components

- Initialize the cycle organization tab with a newly specified step numbers
- specify all the input parameters for the tabs.
- push ***Push Data to MATLAB®*** to secure the data supplied so that MATLAB® can read the data
- save the spreadsheet, close the spreadsheet, and use the GUI to run the simulation

8.1.1. Initialization

From Figure 12 and Figure 13, we see that there is a button called ***Initialize Sheet***. Hitting this button will practically erase all the entries in the *Value* column of the spreadsheet of the current tab. However, there are more to the initialization functionalities. For instance, before initializing, if the user specify a new number of components, the new spreadsheet after the initialization will have the updated number of components. This also holds true for the number of steps in a given PSA cycle (i.e., the cycle Organization tab), the number of columns (i.e., the Packed Bed Properties tab) and the number of tanks (i.e., the Tank Properties tab) as well.

It is important to initialize the spreadsheets that contain quantities that are dependent on the number of components. For instance, by inspecting the *Dependency* column, we can see if a given input parameter depends on the number of components (e.g., isotherm parameters, kinetics parameters, etc.). Then, we must initialize tabs that contain such parameters and provide correct value of the parameters. Likewise, it is also important to initialize and specify relevant parameters for the cycle organization as well.

Release History

- beta version release: 4/30/2021

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office Award Number DE-EE0007888.

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or

responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Appendices

A. List of Available Sub-Models

For the beta version of the simulator, the following sub-models are available. To run a working simulation, the user must choose sub-models from the list of available sub-models.

A.1. Adsorption Isotherm

- linear decoupled isotherm
- extended Langmuir isotherm

A.2. Adsorption Rate

- linear driving force (LDF) decoupled adsorption kinetics model

A.3. Equation of State

- ideal gas

A.4. Valve

- linear valve

A.5. Heat Capacity

- a constant heat capacity model

A.6. Pressure Drop

- no pressure drop model

A.7. Cyclic Steady State

- overall states convergence criteria
- 1st single column states convergence criteria
- all column states convergence criteria
- 1st product tank convergence criteria

References

[1] Kayser, J. and Knaebel, K. (1986). Pressure swing adsorption - experimental study of an equilibrium theory. *Chemical Engineering Science*, 41(11):2931–2938.