Novel algorithms for simulation and optimisation of periodic processes operating at cyclic steady state

Susana Medinas

November, 2016

Abstract

The study of novel algorithms for simulation and optimisation of periodic processes operating at cyclic steady state (CSS) is of major importance. This state is the steady state of a periodic process, such as Pressure Swing Adsorption (PSA), often used in the industry for gas separation.

In this thesis, the examples used consist on Rapid Pressure Swing Adsorption (RPSA) when the steps of the cycle are very fast. An already existent example of RPSA for production of enriched oxygen air in gPROMS® ModelBuilder is converted into gPROMS® ProcessBuilder and validated displaying a small deviation in the two KPIs monitored (Purity and Recovery).

An algorithm was developed in Python, ensuring the communication between the two software, gPROMS® and Python. Wegstein method is implemented in the algorithm and applied to models with simple functions in gPROMS® ProcessBuilder and to the one bed RPSA example gPROMS® ProcessBuilder that was developed during this thesis project. Wegstein method shows better convergence when compared to successive substitution.

A Jacobian-free Newton Krylov (JFNK) method was applied to the RPSA model Builder example with the help of a cyclic solver developed at PSE. JFNK method shows much faster convergence using a good initial guess ensured by doing some Successive Substitutions prior to using the method, and using Precondition.

Keywords – gPROMS, Pressure Swing Adsorption, Cyclic-steady state, Wegstein method, Jacobian-Free Newton-Krylov method, Python

1. Introduction

Periodic Adsorption Processes (PAPs) are preferred to other separation techniques in processes such as gas purification, gas drying, solvent vapour recovery applications and bulk gas separation [1]. Therefore, improving modelling and optimisation techniques of these type of processes, with very specific characteristics requiring a special treatment, is very important.

Looking at Periodic processes, instead of considering a steady behavior of the system, a constant periodical behavior repeated over each cycle is desired. Periodic Processes are processes whose state can be modelled by a periodic function (Figure 1). This type of chemical periodic processes are operated cyclically and when subjected to a periodic input signal for a certain amount of time, they converge to a cyclic-steady-state (CSS). CSS is reached when the state values at the start of cycle k are equal to the ones at the start of cycle k+1. This is a quite interesting condition in the operation, therefore the convergence to this state needs to be well accomplished. However, the convergence to CSS is highly time consuming (can take several hours).

Modelling and simulation of PAPs needs to address the intrinsic complexity of these processes, which manly arises from their periodic nature. For instance, considering direct substitution, also known as successive substitution, which is using the final value of an iteration as the initial guess for the next iteration, seeing that most of these processes need many previous cycles until reaching this state, the convergence to the CSS can be a huge drawback in the process. A more efficient way to deal with this issue is required and in consequence was appointed as the motivation to this work. Therefore, the process may be accelerated in many different ways considering that the simulation of a single cycle is a nonlinear function mapping from the initial state to the final state (system state at end of the cycle is a function of system state at start of the new cycle). A new approach can be considered by this function using a sequence of cycles as an iteration which converges to the CSS, and then, regarding the results of previous iterations, the state at the start of each iteration can be estimated, accelerating the convergence.

Figure 1 - Sates of a system disturbed by a Periodic Input
1.1 Adsorption and Isotherms
Adsorption is a phenomenon that allows separation of species in a fluid mixture, a molecule from a fluid phase experiences a spontaneous attraction close to the surface of a solid (adsorbent, normally a porous solid). The nature of the surface and species involved affect the type of interaction. This process can be ruled by either differences in the adsorption equilibrium or adsorption rates [3]. Adsorption is a very complex mechanism, so several models have been developed to describe it. Isotherms are curves that can relate, in equilibrium, the amount of a substance from a fluid phase held in a solid-phase.

1.2 Adsorbent
The main task of defining an adsorption unit is selecting the adsorbent to be employed correctly, since the behaviour of the PSA unit is mainly determined by the adsorbent employed for the separation [4]. Defining an effective strategy to regenerate the adsorbent is also a very important task, and there is where engineering takes place. The most common adsorbents are activated aluminas, activated carbons and zeolites. The bed can also be layered. The adsorbents and their adsorption properties can be used to combine and improve the process performance by utilizing the adsorbents’ inherent potentials [5].

1.3 Periodic Adsorption Processes
Periodic Adsorption Process (PAPs) are used to separate a one or more components from a mixed stream, requiring at least one adsorbent that preferentially adsors one component. They are characterized by two main steps Adsorption and Desorption. In the Adsorption step, the preferentially adsorbed species are gathered from the feed, while in the Desorption step, these species are removed from the adsorbent. This step is also called Regeneration. [6] There are different types of PAPs depending on the technique used to regenerate the adsorbent. The desorption can be managed by changing process parameters such as Temperature and Pressure, since that affects the adsorption equilibrium. In the Pressure Swing Adsorption (PSA) process this occurs due to the reduction of the partial pressure of the gas phase. Compared to Temperature Swing Adsorption, PSA cycles are usually shorter but require more adsorbent per unit time. Studies of Periodic Processes modelling and optimisation (such as PSA) have been reported in the literature. [7] [8] [9]

1.4 Pressure Swing Adsorption
Only one of the two streams can generally be obtained with the desired purity in PSA. A vast majority of PSA systems are “Stripping-type Pressure Swing Adsorption” (S-PSA) and “Rectifying-type Pressure Swing Adsorption” (R-PSA). However, the Dual Reflux Pressure Swing Adsorption (DR-PSA) process allows enrichments in the light and heavy components. [10] The stripping type are based on Skarstrom cycle [11]. Stripping PSA processes are capable of producing only the light product at high purity from a binary feed gas mixture, since the purity of the heavy product is confined by thermodynamic constraints. [12][13] [14] There are various modifications to the typical cycle. PSA is a technique with multiple possibilities where some of them may have impact on energy requirements and even on the efficiency of the unit. More varieties were introduced in early 1960’s but only in 1970’s there was an increase in scale and range of these processes. The adsorption in commercial scale requires availability of adsorbent in tonnage quantities with economic cost and this was only managed some years after the first PSA applications.

1.5 Operation of PSA cycles
1.5.1 Skarstrom cycle
The Skarstrom process is the basis of actual processes. This process includes two adsorbing beds and involves four steps: Pressurization, Adsorption (Feed), Countercurrent Blowdown and Countercurrent Purge. The Skarstrom Cycle, is presented by Skarstrom in 1957 as a process to dry air or other gaseous materials. It is the most well-known PSA process and it does not need an external heat to regenerate the adsorbent, which reduces the requirement for adsorbent material and provides equipment that allows the production of effluent streams rich in, at least, one component. When the feed stream enters the column what happens is that the less adsorbed (light) component passes through the column faster than the other(s). The compounds that are more adsorbed (heavy) will stay in the adsorbent. Therefore, the feed should be stopped before the adsorbent capacity has been exceeded, so that it can be regenerated (by desorbing) avoiding the heavy compounds to travel through the column.

1.5.2 Innovations of the Skarstrom cycle
Some steps have been added to the typical Skarstrom cycle such as blowdown, purge, rinse and pressure equalization.

1.5.3 PSA applications
PSA technology has been used in various applications: air separation, CO2 removal, noble gases (He, Xe, Ar) purification, CH4 upgrading [15], n-isoparaffin separation, hydrogen purification, and so forth. So, the increase in the demand for this gas resulted in a strong motivation for the development of new PSA processes. The need for these processes to be economical resulted in a strong motivation for the development of new PSA processes and promoted the development of multi-bed processes. [16] [17]

1.6 Mathematical Solving Approach for Adsorption beds
Normally the applications of Adsorption deal with non-isothermal systems and non-spatially constant velocity flows during the adsorption and purge operations, which may lead to numerically intensive solutions. [1] Partial differential equations, boundary and initial conditions are crucial to solve systems with adsorption beds, since these are normally a sequence of elementary steps which require appropriate boundary conditions. [18] The development of specialised algorithms for solving bed models is required for simulation. These models are described by partial differential-algebraic equations (PDAEs) in space and time, derived from heat, mass and momentum balances plus transport and equilibrium equations. Numerical methods that are suitable to solve a given PDAEs system are not necessarily good to solve another system accurately. These equations can lead to a completely different behaviour for each change in either one parameter or boundary condition. Solving PDAEs is inextricable difficult, and there is a lack of universally applicable solution methods in spite of all the effort applied in this subject. Among the numerical methods devoted to solve the PDE and PDEAs systems there are different types of methods.
Numerical methods may be based on: [19]
- Method of lines (MOL) [20] [21]
- Finite difference Methods
- Weighted Residuals Methods
• Finite Element Methods
• Finite Volume Methods
• Adaptive Grid Methods
• Moving grid Methods

In previous Model simulation carried out in the gPROMS®-modelling tool, the method of lines adopted consists of two steps:
1) The discretization of the continuous spatial domains into finite grid of points, resulting in a set of differential algebraic equations (DAEs), and
2) Integration of the DAEs over time with DASOLV integrator based on backward differentiation formulae (BDF).
This method minimizes an integral error efficiently by this DAE integration technique, that being its main advantage. Cantered finite difference method (CFDM) in the context of the MOL is used for the discretization. [19]

1.7 Mathematical Solving Approach for Periodic Adsorption Processes

To be able to understand the dynamics of the systems, mathematical models are required to help predicting and explaining the separation results. However, these processes are quite particular since they do not operate in steady state conditions, but in cyclic steady state conditions (CSS) caused by periodic transient conditions with each bed repeatedly undergoing a sequence of steps. In each cycle concentration profiles change dramatically. The technical and economic performance of the operation can only be fully determined after this state is guaranteed.

Moreover, the multi-objective optimisation of PSA operations has been addressed by few studies. However, taking into account the large number of the degrees of freedom, it is desirable to allow the enhancement of the performance of PSA cycles, and subsequently to expand the application of the process, which is possible with a mathematical programming approach to the optimisation of PSA processes. [7] [22]

In conclusion, some of the solution strategies may include PDE discretization, CSS convergence acceleration, sensitivity evaluation and optimization.

1.8 Mathematical formulation of Periodical Adsorption Processes and CSS

Resulting from a spatial discretization of the bed model, an objective function \( f(x, y, u, t) = 0 \), describing a periodic adsorption process can be considered. Where \( f: \mathbb{R}^{n_x+n_y+n_u} \rightarrow \mathbb{R}^{n_x+n_y+n_u} \). A state variable is one of the set of variables that are used to describe the mathematical "state" of a dynamical system. The state of a system describes the system at a certain point, and so, it is enough to determine its future behaviour in the absence of any external forces affecting the system. In this system \( x \) are the state variables, \( y \) are the algebraic variables, and \( u \) are the input variables.

The sum of the time of all the steps that consist one cycle is the cycle time. A forced periodic operation can be taken into account over a cycle time, \( T_c \). This leads to a forced periodic system, in which the inputs \( u \) cycle continuously over a period \( T_c \). In fact, this is what happens when valves are opened or closed during a cycle of a periodic process, inputs to the system cycle over a period.

\[
u(t) = u(t + T) \quad \forall t
\]

In this type of processes, the final state of one cycle is a function of the initial state of the cycle.

\[
x(T) = \phi(x(0))
\]

Therefore, function \( \phi \) is a mapping from the initial state to the final state, with \( \phi: \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x} \). At cyclic steady state, the final state of a system after a cycle is identical to that at start of the cycle.

\[
x(T) = x(0)
\]

According to the former, reaching CSS is done finding values \( x^* \) such that:

\[
x^* = \phi(x^*)
\]

Or even, with the nonlinear system \( R: \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}:\)

\[
R(x) = \phi(x) - x
\]

In CSS \( x^* \) is the solution of \( R(x^*) = 0 \), meaning that the difference between the system state at the beginning and end of one cycle, or residuals must all be very close to zero (with very small \( \delta \)).

\[
\|R(x^*)\|_w < \delta
\]

1.9 Acceleration

Convergence can be enhanced by applying Wegstein method. This iteration method is the default method in Aspen Plus. However, methods such as the direct method, Broyden's method and Newton's method can be used.

1.9.1 Wegstein method

The Wegstein or secant convergence method is used for convergence and is very easy to implement in a computer program. It requires the objective function in the form \( x = g(x) \) and two initial estimates of the solution \( x^{(1)} \) and \( x^{(2)} \), that the function is evaluated at these two points \( g_1 \) and \( g_2 \). The third point is calculated where the secant intersects the line at \( x = g(x) \).

1.9.2 Newton method

Newton method can also be applied to this type of problems. In fact, with \( R(x) = \phi(x) - x \), Newton method can be used to update the solution \( x^k \) by a Newton correction \( \Delta x^k \). However a Jacobian needs to be calculated, or the derivative of \( R \) with respect to the state:

\[
J^k \Delta x^k = -R(x^k)
\]

\[
x^{k+1} = x^k + \Delta x^k
\]

\( \phi \) is the result of the DAE integration over one cycle. However considering the following:

\[
J^k \frac{\partial R}{\partial x}|_{x^k} = \frac{\partial \phi}{\partial x}|_{x^k} - I = x^k - I
\]
there is the Preconditioned Jacobian. A matrix of size $N_a$ by $N_s$, the sensitivity equations (given by the following equations) must be integrated. As expected, this can be very time consuming, in fact, Jiang Fox and Biegler reported that 90% of the computational time was spent integrating the sensitivity equations, meaning. 2 hours to calculate the Jacobian matrix for relatively small example with just 226 states. [24] [25]

1.9.3 Jacobian-Free Newton Krylov method
Jacobian-free Newton Krylov (JFNK) method can be used for computing the CSS condition. It uses an iterative linear solver for the Newton steps, the Generalized minimal residual method (GMRES). This method does not require the Jacobian matrix since it only requires the product of the Jacobian with various vectors, being Jacobian-free. [26]

This method solves a linear system of type $Au = b$ and considers that the linear residual at iteration $j$ is:

$$r_j = Au_j - b$$  \hspace{1cm} (11)

Then, approximates the solution $u_j$ with linear combinations of matrix-vector products. At each GMRES iteration coefficients are chosen to minimize the 2-norm of the residual. In order to compute CSS of a periodic adsorption process, the same nonlinear system is required as equation (6). In this system, the Newton correction is a linear system of equations, $A$ is the Jacobian, $u$ is the Newton update, and $b$ is minus the residual, meaning:

$$J^k \Delta x^k = -R(x^k)$$  \hspace{1cm} (12)

In order to solve this linear system, the evaluation of $J^k v$ Jacobian-vector products is required, where $v$ are various vectors. However, the product of the Jacobian with an arbitrary vector $v$ can be obtained simply doing a single finite perturbation, and simulating one perturbed cycle to obtain the approximate Jacobian vector product, without ever computing the actual Jacobian.

$$J^k v \approx \frac{R(x^k + ev) - R(x^k)}{e}$$  \hspace{1cm} (13)

Since GMRES solves a linear system $Au = b$ via an iterative method, the speed of convergence becomes a major consideration that needs to be improved. In fact, the success of the JFNK method depends on how few iterations it takes to converge or on how rapidly the linear system converges. Each GMRES iteration requires the simulation of a perturbed cycle, which is the limiting step of the method. So, to fasten the convergence, the matrix $A$ must suffer some changes. In order to help the convergence, the eigenvalues of $A$ must be clustered away from the origin. The more clustered the eigenvalues, the more rapidly the linear residual will converge. This operation is called “Preconditioning”. It transforms matrix $A$ with the following equations.

$$(AP^{-1})w = b$$  \hspace{1cm} (14)

$$u = P^{-1}w$$  \hspace{1cm} (15)

A new linear system that consists in solving this linear system for $w$ arises. The system matrix is now multiplied by the preconditioner. Multiplying the preconditioner by the solution $w$ allows finding $u$.

The perfect preconditioner matrix turns all eigenvalues into 1, so it would simply be the inverse of the system matrix. However, this matrix is unavailable. If the eigenvalues are randomly distributed around the origin, the results show a very slow convergence. In fact, previous work reported that an RPSA example with one adsorption bed and 364 state variables, the residual does not decrease much after 100 GMRES iterations. However for eigenvalues clustered tightly around a real value of 2, the result is a very fast convergence rate of the linear residual as it converges to an acceptable accuracy in about 10 iterations. Preconditioning can influence the clustering of the eigenvalues, and consequently, the convergence rate of GMRES. [24]

In order to find the Jacobian, there is the Preconditioned Newton correction:

$$(J^k P^{-1})w = -R(x^k)$$  \hspace{1cm} (16)

$$\Delta x^k = P^{-1}w$$  \hspace{1cm} (17)

The considered linear system required for the JFNK method is the Jacobian times the preconditioner times a vector $w$ equals the residual. Then, the Newton method update is calculated by multiplying the vector by the preconditioner, after the linear system is solved for $w$.

Preconditioned Jacobian-vector products can be obtained simply multiplying the arbitrary vectors $v$ by the preconditioner and then simulating the perturbed cycle:

$$J^k P^{-1}v \approx \left[ R(x^k + eP^{-1}v) - R(x^k) \right] / e$$  \hspace{1cm} (18)

Then, $P^{-1}$ is chosen to be such that $|P^{-1}| \approx 1$, however, the Jacobian still needs to be found and, in fact, given a sequence of points $x_i$, $i = 0, 1, \ldots, n$ and the corresponding values of some functions $R_i \equiv R(x_i)$, Jacobian approximations can be constructed $J_i \approx J_i = \frac{\partial R_i}{\partial x}(x_i)$. Jacobian is approximated by Least Change Secant Updates: taking a sequence of states $\Delta x_i$ and the corresponding values of the function evaluations $\Delta R_i$, it considers how the residuals change as the points $x$ change. Regarding that the quality of approximation improves gradually from one point to the following, as the points change and the functions are evaluated, in the absence of any other information, starting with $A_0 = I$ is not a problem. Therefore, defining $\Delta x_i = x_i - x_{i-1}$ and $\Delta R_i = R(x_i) - R(x_{i-1})$, Jacobian approximation may be obtained via:

$$P_i = P_{i-1} + \frac{(\Delta R_i - P_{i-1}^{-1}\Delta x_i)}{\|\Delta x_i\|_2^2}\Delta x_i^T$$  \hspace{1cm} (19)

$P_i^{-1}$ is obtained directly via Sherman-Morrison formula:

$$P_i^{-1} = P_{i-1}^{-1} + \frac{(\Delta x_i - P_{i-1}^{-1}\Delta R_i)}{\Delta x_i^TP_{i-1}^{-1}\Delta x_i}$$  \hspace{1cm} (20)
2. CSS acceleration for simple gPROMS® models

2.1 Wegstein method

A CSS acceleration was implemented for a gPROMS® model containing simple mathematical functions (not shown in this article) and for an adsorption bed gPROMS® model (shown in section 3.1).

Considering the objective function in the form \( x = g(x) \) and two initial estimates of the solution \( x^{(1)} \) and \( x^{(2)} \) and the function evaluated at these two points \( g_1 \) and \( g_2 \), the third point is calculated where the secant intersects the line where \( x = g(x) \). So, \( g(x^{(k+1)}) = g(x^{(k)}) \), the convergence rate depends on the gradient of \( g(x) \).

From \( k=1 \), the gradients are computed:

\[
s_i = \frac{g_i(x^{(k)}) - g_i(x^{(k-1)})}{x_i^{(k)} - x_i^{(k-1)}}, \quad i = 1, \ldots, n
\]

(21)

Subsequent estimates of solution are given by:

\[
x_i^{(k+1)} = g_i(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) (1 - q_i) + x_i^{(k)} q_i
\]

(22)

Where

\[
q_i = \frac{s_i}{s_i + 1}, \quad i = 1, 2, \ldots, n, \quad k = 2, 3
\]

(23)

Table 1 - Convergence of Wegstein method accord to \( q_i \)

<table>
<thead>
<tr>
<th>Value of ( q_i )</th>
<th>Expected convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 &lt; q_i &lt; 1 )</td>
<td>Slow stable convergence</td>
</tr>
<tr>
<td>( q_i = 1 )</td>
<td>Regular successive substitutions</td>
</tr>
<tr>
<td>( q_i &lt; 0 )</td>
<td>Accelerated successive substitutions: may cause instabilities</td>
</tr>
</tbody>
</table>

2.2 Examples and validation

One bed RPSA developed in gPROMS® ModelBuilder, previously at PSE, was accelerated with the prototype solver with JFNK method developed at PSE, described in section 2.4. Another One bed RPSA gPROMS® example was developed during this thesis through the conversion of the previous one into ProcessBuilder, and validated afterwards with the former. This example was not accelerated with the prototype solver developed at PSE, but with the Wegstein method implemented in Python during this thesis.

2.2.1 One Bed gPROMS® ModelBuilder example

The bed contains a single adsorbent, Zeolite 5A, and the operation is done cyclically. Each cycle is composed by Pressurization and Depressurization, each during 1.5 seconds, as described in Figure 3, in order to produce oxygen enriched air. In order to structure the model to simulate this process, the following model assumptions were considered:

- The operation of the bed is isothermal.
- There are no radial variations in the bed.
- The fluid phase follows Ideal Gas Law.
- The adsorption bed parameters (i.e. bed void fraction, bed bulk density, and particle size) are uniform and constant.
- The flow in the bed is axially dispersed.
- The axial pressure drop can be described in a satisfactory way by Darcy’s law.
- The mass transfer rate is described by a linear driving force model.

The model of adsorption bed for a mixture separation of \( n \) components could be described by the following equations: Considering \( i = 1 \) to the number of components (\( n \)), the component mass balance is done for each component inside the bed, not including its boundaries.

\[
\varepsilon_{tot} \frac{\partial C_i(z)}{\partial t} = - \frac{\partial C_i(z)u}{\partial z} \left( D_i \frac{\partial C_i(z)}{\partial z} - \rho_{bed} \frac{\partial q_i}{\partial z} \right)
\]

(24)

With:

\[
\varepsilon_{tot} = \varepsilon_{bed} + \varepsilon_p (1 - \varepsilon_{bed})
\]

(25)

The ideal gas law includes the end bound, \( z = L \):

\[
\frac{P}{RT} = \sum_{i=1}^{n} C_i(z)
\]

(26)

Meaning that filling the bed with atmospheric air equals:

\[
P_{atm} \times Y_{feed} = \sum_{i=1}^{n} C_i(z) \times R \times T_{feed}
\]

(27)

For the pressure drop, Darcy’s pressure drop equation can be considering for the steady state momentum balance of gas flow at low velocity through a packed bed:

\[
\frac{\partial P}{\partial z} = \frac{180 \mu (1 - \varepsilon_{bed})^2}{\varepsilon_{bed}^3 d_p^2}
\]

(28)

As the Equilibrium isotherm, an approximate Langmuir isotherm, a simple linear isotherm, is taken into account for each component and the whole bed:

\[
q_{eq,i} = m_i P_i = m_i C_i R T
\]

(29)

The adsorption rate equation considered is linear for all bed and each component.

\[
\frac{\partial q_i}{\partial t} = D_i \frac{\partial q_i(r)}{\partial r} \mid_{r=r_p} = k_i (q_{eq,i} - q_i(r_p))
\]

(30)

This equation means that the concentration of specie \( i \) in the bed is proportional to the difference between the adsorbed
amount at equilibrium $q_{eq,i}$ in the macropores and the current value of the adsorbed phase. $K_i$ or $k_{LDF}$ is the mass transfer coefficient and is a proportionality constant. Heat of adsorption is considered to be zero. Starting with the adsorbent at equilibrium with gas phase equals:

$$\frac{\partial q_i}{\partial t} = 0$$  \hspace{1cm} (31)

The mass transfer coefficient is given by the following equation:

$$k_i = \frac{15 \varepsilon^2 (1 - \varepsilon_b)}{r_p^2 \rho_{bed} m_i R T \tau}$$ \hspace{1cm} (32)

The amount of material fed to the column, amount of material escaping in the product stream and of material escaping in the waste stream were integrated in order to calculate purities and recoveries.

The following boundary conditions were considered:

<table>
<thead>
<tr>
<th>Pressurization</th>
<th>Depressurization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{feed} = C(L,0)$ R $T_{feed}$</td>
<td>$\frac{dC(L,0)}{dz} = 0$</td>
</tr>
<tr>
<td>$P_0 = P_{Feed}$</td>
<td>$P_0 = P_{Waste}$</td>
</tr>
</tbody>
</table>

2.2.2 One Bed gPROMS® ProcessBuilder example

The example in gPROMS® ModelBuilder was considered and converted to a gML (gPROMS Model Library) included in the libraries of the recently launched gPROMS® ProcessBuilder package.

Several main units were considered in this example (table 3) to obtain the flowsheet presented in Figure 5.

Table 2 - Boundary conditions for operation steps

<table>
<thead>
<tr>
<th>Units in ModelBuilder</th>
<th>gML used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Adsorption_bed_gML</td>
</tr>
<tr>
<td>Feed</td>
<td>Source_material_reversible_gML</td>
</tr>
<tr>
<td>Product</td>
<td>Source_material_reversible_gML</td>
</tr>
<tr>
<td>Waste</td>
<td>Source_material_reversible_gML</td>
</tr>
<tr>
<td>Feed_Valve</td>
<td>Valve_reversible_gML</td>
</tr>
<tr>
<td>Product_Valve</td>
<td>Valve_reversible_gML</td>
</tr>
<tr>
<td>Waste_Valve</td>
<td>Valve_reversible_gML</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Scheduling_gML</td>
</tr>
</tbody>
</table>

Other units were added in order to replicate the example, since the volumetric flow at the product end of the bed was set to be constant: Stream_analyzer_reversible_gML AS Stream_analyzer_reversible_gML - with the reporting (Overall) set to Volumetric flowrate, and Adj_spec_gML AS Adj_spec_gML. For replicating the Langmuir isotherm equation in gPROMS® ProcessBuilder, two custom models were created during this theses. A model called isotherm_section_custom_gML with the isotherm equation, as well as mass_transfer_coefficient_adsorption_custom_gML, containing the mass transfer equation.

In order to assure the operation, the openings and closures of the valves are managed by a schedule model:

This is very important to be well assign for a PSA process. In fact, the results depend a lot on the valve behavior along time, meaning step durations and cycle sequence for instance.

Figure 4 - Valve positions specified in the Schedule
2.3 Python Program

A Python program was created in order to apply Wegstein acceleration to the convergence to the solution of simple functions in gPROMS® and for the convergence to the CSS of gPROMS® one bed gPROMS® ProcessBuilder example. In order to do this, communication between Python and gPROMS® should be established. A Foreign Process Interface (FPI) available in gPROMS® allows the exchange of data and other information between executing gPROMS® simulations and external software. This allows the interaction with an external software such as python.

The interaction takes the form of a special set of elementary actions within the gPROMS® Task language. It is in gPROMS® that the user must specify the correct points of the simulation in which to save and send the initial or final values of each cycle to a text file to be used by Python.

So, this communication happens at discrete time points throughout the simulation and the user can determine the frequency and content of the exchanges. The user can also determine how the information should be exchanged (in this case it is done with text files). To execute the program. After doing the model, the user only has to export gPROMS® model and specify the number of Wegstein method applications to do.

The algorithm of the program is shown bellow in Figure 8.

3. Results and discussion

3.1. Python program - Wegstein Method

This example was considered with five types of state variables, totaling 85, to which Wegstein method was applied.

While performing the Wegstein acceleration, when the Residual reaches $10^{-6}$, the Key Performance Indicators (KPI’s) obtained after that last acceleration are very similar to the ones estimated with Successive Substitution for the same accuracy.

In fact, only 4 Wegstein method iterations were enough to obtain the residual of the order of $10^{-5}$. This program took 41 seconds to display the result with four iterations.

In fact, this was the only acceleration that was successfully applied to the one Bed RPSA gPROMS® ProcessBuilder example – the prototype cyclic solver described in section 2.4 was not yet robust enough to be used with this example.

3.2. Cyclic Solver – Jacobian-Free Newton Krylov Method

The first acceleration results, with the cyclic solver with JFNK method, for the one Bed RPSA gPROMS® ProcessBuilder example were the following:

Column.Purity  = 0.920422
Column.Recovery = 0.0341193

The values are equal to the ones obtained with the successively substitution in the same example, which were:

0.920421 and 0.0341193 respectively.

3.2.1 Sensitivity analysis

In this case, a study was made in order to understand the effects of the following solution parameters detailed in the configuration file of the solver:

- Maximum number of successive substitutions prior to Newton-Krylov iteration
- Stopping tolerance on f-norm
- Right preconditioning method
- Method for epsilon of finite-difference perturbation
- Constant $\epsilon_r$ used for computing epsilon
- Selection of forcing term eta
- Initial value of eta in (0, 1) for forcing term eta

Figure 9 - Logarithm of the residual for each iteration using both Successive Substitution and Wegstein method

While performing the Wegstein acceleration, when the Residual reaches $10^{-6}$, the Key Performance Indicators (KPI's) obtained after that last acceleration are very similar to the ones estimated with Successive Substitution for the same accuracy.

In fact, only 4 Wegstein method iterations were enough to obtain the residual of the order of $10^{-5}$. This program took 41 seconds to display the result with four iterations.

In fact, this was the only acceleration that was successfully applied to the one Bed RPSA gPROMS® ProcessBuilder example – the prototype cyclic solver described in section 2.4 was not yet robust enough to be used with this example.

Figure 8 - Framework of Python and gPROMS® interaction to accelerate performance of gPROMS® models

2.4 Cyclic Solver

This cyclic solver is applied to the state variables of the system. It can execute sensitivity analysis and monitor key performance parameters.

A file with the initial state variables named CSS_Progress needs to be available. This file must be available each time the solver runs, and it will be used by the solver to save the variables throughout its execution.

Another file, a configuration file, needs to be available for the solver as well. This is the file that includes specifications the solver needs to access the correct input files (the gENCRIPT for instance), as well as some solving parameters including the following:
The effect of right preconditioning must be studied, since from previous work, inclusively at PSE it is known to accelerate the convergence of the method. [25]

The method for epsilon (ε) for finite-perturbation should be tested in order to know how the solver behaves accordingly to this parameter. The selection of the values of perturbation parameter ε is important because it must be large enough to prevent numerical noise effecting the accuracy of the derivative, but it still needs to be small enough to accurately approximate the derivative. In JFNK there is only one perturbation parameter that must be selected for all states (rather than finite differences, where an appropriate perturbation value can be found for each state). So this is crucial to properly selecting the magnitude of the perturbation parameter.

The selection of forcing term ξk, the initial value of ξ and the methods used to obtain the forcing term are also very important parameters for the solver performance. Instead of considering the normal Newton correction, the Newton correction can be relaxed to an inexact Newton condition: Where the forcing term τk ∈ [0,1] can be specified in various ways leading to a different convergence.[27] An initial s0 satisfying the previous equation is determined using a Krylov subspace method to solve the newton correction equation approximately that is why this method is a Newton-Krylov method. In this case, the Krylov subspace method used is GMRES. Once an initial s0 has been determined, it is tested and, if necessary reduced in length through a safeguarded backtracking until an acceptable step is obtained. Choosing the forcing terms can lead to different results. In fact, desirably fast local convergence can be obtained by using suitably small values for the initial forcing terms near a solution. The initial forcing terms can also affect the performance of the algorithm away from a solution, in case of considering an initial τk too small, the Newton equation can be over solved, meaning imposing an accuracy on an approximate solution s0 of the newton equation that leads to significant disagreement between \( F(x_k + s_k) \) and its local linear model \( F(x_k) + F'(x_k) \cdot s_k \). This may result in little or even no progress towards a solution, involving probably a pointless expense since a less accurate solution may be both cheaper and more effective in reducing the norm of F. However, after analysing the results, this has not been registered. The various methods used worked in a relatively similar way.

Regarding the performance of the solver there are nine, that can be analysed while changing the parameters specified in the configuration file of the solver referred in section 2.4:

- Number of function evaluations
- Number of Jv - ( Jacobian × vector ) - evaluations
- Number of P( inverse v ) -( Preconditioner\(^{-1}\) × vector ) - evaluations
- Number of linear iterations
- Number of nonlinear iterations
- Number of backtracks
- CPU time on successive substitution
- CPU time on Newton-Krylov iteration
- Total CPU time: CPU time on successive substitutions plus CPU time on Newton-Krylov iteration.

In order to study the nine performance indicators described above, in the one bed RPSA gPROMS® ModelBuilder example, the following parameters were studied and assigned with the values mentioned in each case represented:

(1) Maximum number of successive substitutions prior to Newton-Krylov iteration: 10, 20, 40, 100:

Figure 10 shows that the number of linear iterations and Jacobian product’s evaluations, as well as the Preconditioner’s and the function evaluations in non-linear iterations vary inversely with the number of successive substitutions. However, with 100 successive substitutions, the results are not as better when compared with 40 substitutions, as it was expected. In fact, the number of function evaluations with 100 substitutions is much more ‘expensive’ computationally.

![Figure 10 - Number of iterations regarding the number of Successive Substitutions prior to JFNK method](image)

In fact a reduction in the computational time is observed as the number of successive substitutions prior to Newton-Krylov iterations increase, this was expected due to the fact that this allows a better initial point (solution), accelerating the convergence to the CSS. With 100 successive substitutions, CPU time increases to 191 seconds (almost one minute more than the lowest value).

(2) Right preconditioning method: 0, 1, 2

Table 4 – Preconditioning methods

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type of Precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No precondition</td>
</tr>
<tr>
<td>1</td>
<td>Precondition without updated precondition matrix during successive substitution</td>
</tr>
<tr>
<td>2</td>
<td>Precondition with updated precondition matrix during successive substitution</td>
</tr>
</tbody>
</table>

The Precondition without updated precondition matrix during successive substitution problem turned the problem more suitable for numerical solving methods, which is reflected in the decrease of all the evaluations represented in figure 11. (The solver had less ‘effort’ to return the solution). Without preconditioning, the number of evaluations of the product of the Preconditioner’s inverse is zero, since there is no Preconditioner. Studying the results, it is noticeable that increases from no preconditioning to preconditioning type 1 and 2 respectively.

The time spent on updating the precondition matrix during successive substitution (Table 5) is very high, because it leads to an increase in the number of function evaluations and other operations (Figure 11).
ing errors that could lead to inaccurate

Builder example

mpared to no

iptions),

or 20,

2

st

cessive substitution for the convergence of

d fastest

accurate

that some took hours instead of minutes, all of them were

parameters combinations lead to the CSS, and besides the fact

in the

computing epsilon, in the fastest solver performance is

obtained, for this parameter’s sensitivity analysis as well.

After studying the 20 results, in the majority of the cases,

no linear iterations and Jv evaluations

The two parameters whose results were shown are the ones

in order to do a mixed sensitivity analysis, parameters were

change randomly. This analysis was also performed for the

one bed RPSA ModelBuilder example. The faster result was

for 20 successive substitutions with the following values:

Table 6 - Specified parameters for the fastest study in
mixed sensitivity analysis.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum number of successive substitutions</td>
<td>20</td>
</tr>
<tr>
<td>maximum number of Newton-Krylov iterations</td>
<td>2 000</td>
</tr>
<tr>
<td>function norm tolerance</td>
<td>1e-6</td>
</tr>
<tr>
<td>step length tolerance</td>
<td>1e-10</td>
</tr>
<tr>
<td>right precondition</td>
<td>1</td>
</tr>
<tr>
<td>method for epsilon of finite difference</td>
<td>2</td>
</tr>
<tr>
<td>constant $\epsilon_r$ used for computing $\epsilon$</td>
<td>1e-6</td>
</tr>
<tr>
<td>forcing term</td>
<td>1</td>
</tr>
<tr>
<td>initial eta</td>
<td>0.5</td>
</tr>
</tbody>
</table>

After studying the 20 results, in the majority of the cases, method 2 for finite difference is the faster while method 4 is the slower. The fastest examples of this mixed sensitivity analysis all hold for Preconditioning method 1 either for 20, 40 or 100 successive substitutions. The constant $\epsilon_r$, used for computing epsilon, in the fastest solver performance is $10^{-6}$, the same was concluded while performing sensitivity analysis to this parameter separately. The initial $\xi$ obtained in the study with the previous parameters was the fastest obtained, for this parameter’s sensitivity analysis as well. Concerning the method used for computing $\epsilon$ and the forcing term method, this study showed the fastest results for method 2, which was the second fastest while performing sensitivity analysis to this parameter.

After this sensitivity analysis, the majority of the solver parameters combinations lead to the CSS, and besides the fact that some took hours instead of minutes, all of them were accurate when compared to the ones obtained for the KPI’s (Purity and Recovery) during successive substitution, since all of them converged to the same CSS.

4. Conclusion

Simulating bed models is a difficult task due to all the Partial Differential Algebraic Equations that described these models, which need a specific treatment while being solved. In fact, after converting a Rapid Pressure Swing Adsorption (RPSA) example from the gPROMS® ModelBuilder into Process Builder obtained results were not exactly the same. This fact enhances the idea that the results of the simulation of this type of processes highly depend on the solving method and the models used. Considering successive substitution for the convergence of this type of processes is incompatible with the user expectations since becomes too time consuming. However, for a one bed Rapid Pressure Swing Adsorption gPROMS® ProcessBuilder example using the Wesgein method this convergence can be accelerated showing more promising results in terms of computational time without putting in danger the accuracy of the solution.

Considering the usage of the cyclic solver with Jacobian-Free Newton Krylov (JFNK) method, the ModelBuilder example displays much faster convergence to the Cyclic Steady State with JFNK than successive substitution. The Cyclic Solver’s results for the Cyclic Steady State are ‘accurate’ since the Key Performance Indicators obtained for the CSS are the same as with successive substitution. After performing the sensitivity analysis, the following conclusions were taken: A good initial guess for the solver is easily found with 10 to 40 Successive Substitutions prior to JFNK method. This can prevent some solving errors that could lead to inaccurate solutions, without compromising the computational effort required to converge Moreover, in this case, Precondition without updated precondition matrix during successive substitution shows even faster results when compared to no Precondition and Precondition with updated precondition matrix during successive substitution. The other parameters, such as the method for finite differences do not influence the performance of the solver as much.

Taking into account the one bed RPSA Process Builder example validated in this project, nor the solver are yet prepared to be used together. The solver requires specific operations in the gPROMS® models that are not easily accomplished by the user. Future works would be to investigate optimisation for dynamic problems such as Pressure Swing Adsorption and other periodic processes with simple examples using gPROMS®. In order to do so, improving the solver configuration file, and output layout (errors descriptions), turning it more convenient and easy to use is fundamental.

5. Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Bed area</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$C$</td>
<td>Concentration</td>
<td>mol $m^{-3}$</td>
</tr>
<tr>
<td>$d_b$</td>
<td>Bed diameter</td>
<td>$m$</td>
</tr>
<tr>
<td>$d_p$</td>
<td>Adsorbent particle diameter</td>
<td>$m$</td>
</tr>
<tr>
<td>$D_i$</td>
<td>Dispersion coefficient of component $i$</td>
<td>$m/s$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Mass Transfer coefficient of component $i$</td>
<td>$s^{-1}$</td>
</tr>
<tr>
<td>$m_i$</td>
<td>Coefficient of Langmuir isotherm</td>
<td>mol/(kg.Pa)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Unit</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>(\epsilon_{\text{bed}})</td>
<td>Bed void fraction</td>
<td>(m_{\text{void}}/m_{\text{bed}})</td>
</tr>
<tr>
<td>(\epsilon_{\text{pel}})</td>
<td>Particle void fraction</td>
<td>(m_{\text{void}}/m_{\text{pel}})</td>
</tr>
<tr>
<td>(\epsilon_{\text{tot}})</td>
<td>Total void fraction</td>
<td>(m_{\text{void}}/m_{\text{tot}})</td>
</tr>
<tr>
<td>(\rho_{\text{bed}})</td>
<td>Bed density</td>
<td>(kg_{\text{bed}}/m_{\text{bed}}^3)</td>
</tr>
<tr>
<td>(\rho_{\text{wall}})</td>
<td>Wall density</td>
<td>(kg/\text{m}^3)</td>
</tr>
<tr>
<td>(\tau)</td>
<td>Tortuosity factor</td>
<td>-</td>
</tr>
</tbody>
</table>

6. Bibliography


