Jacobian-based Model Diagnostics and Application to Equation Oriented Modeling of a Carbon Capture System

Douglas A. Allan[a,b], Anca G. Ostace[a,b] and Tanner Polley[a,b]

[a]National Energy Technology Laboratory Pittsburgh PA USA
[b]NETL Support Contractor Pittsburgh PA USA

ARTICLE INFO

Keywords:
Pyomo
Optimization
Carbon Dioxide Capture
Jacobian
Modelling

ABSTRACT

It can be difficult to identify the specific variables or equations responsible for convergence issues in large mathematical programming models. The Institute for the Design of Advanced Energy Systems Integrated Platform (IDAES-IP) contains a tool to identify poorly scaled constraints and variables by searching for rows and columns of the Jacobian matrix with small L2-norms. A singular value decomposition is then performed to identify degenerate sets of equations and remaining scaling issues. This work presents a flowsheet developed for post-combustion carbon capture using a monoethanolamine (MEA) solvent system as a case study. This work takes the reader through the entire process of model diagnostics and reformulation, from a basic introduction to the mathematics behind these model diagnostics to the reformulations necessary to make the model numerically robust, including a significantly modified enhancement factor model.

1. Introduction

To ensure energy abundance, a wide variety of energy systems must be deployed. Process optimization can help efficiently allocate resources to effect the changes in the U.S. and world economies needed to achieve this goal. The Institute for the Design of Advanced Energy Systems (IDAES) was founded in 2015 to study advanced energy systems and to develop the IDAES Integrated Platform (IDAES-IP) to facilitate their development and optimization [1]. IDAES-IP is based on the Pyomo modeling language [2, 3] in Python and has been used to simulate a wide variety of chemical and energy process systems.

Because IDAES-IP is equation-oriented (EO), it can provide improved convergence when closing recycle loops over the sequential-modular (SM) approach that is used in popular commercial process modeling and simulation tools like Aspen Plus, CHEMCAD, and Pro/II. However, it also requires more user skill to ensure that the model is well-formulated to benefit from these theoretical convergence improvements. IDAES has developed and implemented a diagnostics toolbox leveraging several years of experience of many experts in EO modeling, debugging, and optimization, making it available to the public [4]. This toolbox features methods to determine a model's structural issues, such as determining whether a system of equations is structurally singular or if a mismatch exists in the units of measurement, as well as a model's numerical issues, such as whether the Jacobian matrix of the system of equations is rank deficient or whether variables and constraints are badly scaled.

Previous work on model diagnostics for EO models focused primarily on diagnosing structural issues in a model. The Dulmage-Mendelsohn partition [5] of EO models into overconstrainted, well-constrained, and underconstrained systems was implemented in the AMOEBA environment in Modelica by Bunus and Fritzson [6]. Soares and Secchi [7] provide a variation on this method using directed graphs instead of bidirectional graphs. Olsson et al. [8] highlight the syntax introduced in Modelica 3.0 to allow checking individual model components for structural singularity. Broman et al. [9] propose a method of integrating checking for the number of degrees of freedom into the class system of object-oriented modeling languages and apply it to a derivative of Modelica.Bublitz et al. [10] implemented both structural and numerical diagnostics for the MOSAICmodeling platform. Wang et al. [11] propose a hierarchical structural analysis method to verify EO models without having to flatten large-scale equation systems, thereby improving the efficiency of analyzing singularities in complex models. The Dulmage-Mendelsohn partition was implemented in Pyomo by Parker et al. [12] and is available through the IDAES diagnostics toolbox.

douglas.allan@netl.doe.gov (D.A. Allan)
ORCID(s): 0009-0008-8232-6691 (D.A. Allan); 0009-0008-5957-4152 (T. Polley)

In addition to serving as a diagnostics tool, the Dulmage-Mendelsohn factorization of the model Jacobian is a convenient solution method because it brings the system into a block triangular form. The system of equations can then be solved by applying, e.g., Newton's method to the blocks on the main diagonal, then using backwards substitution for the nondiagonal elements. However, while powerful, block triangularization often does not work as intended in many systems of interest to chemical engineers. When applied to a process model with countercurrent flow or a flowsheet with a long recycle loop, it is not uncommon for block triangularization to create a diagonal block containing almost all the variables and equations. Bublitz et al. [10] are able to overcome this problem by automatically creating tears in the (linear) system, but the convergence rate can suffer. Similarly, when applied to model diagnostics, the overconstrained and underconstrained subsystems frequently contain too many equations to adequately diagnose. Furthermore, systems without a structural singularity can have a global numerical singularity.

One technique for identifying numerical singularity is the Degeneracy Hunter algorithm proposed by Dowling and Biegler [13] and available through the IDAES Diagnostics Toolbox. It performs a QR factorization of the model Jacobian matrix, then solves a series of mixed integer linear programs (MILPs) to find irreducible degenerate sets of equations. However, the success of this algorithm strongly depends on the availability and quality of the MILP solver used. When commercial solvers like CPLEX or Gurobi are not available, Degeneracy Hunter can be slow or fail completely.

Another method to diagnose numerical singularities is by using the singular value decomposition (SVD) of the Jacobian. The SVD of the Jacobian creates an orthogonal vector basis for the spaces of both variables and constraints, which are linked via scaling by the singular values. If the variable and constraint spaces are linked by a zero or near-zero singular values, it shows that the problem is ill-conditioned and may be badly posed. While the singular vectors do not specify individual variables and constraints as problematic, but rather linear combinations of variables and constraints, sets of potentially problematic variables and constraints can be extracted for user inspection. However, before SVD analysis is effective, variables and constraints must be effectively scaled.

Chemical process models frequently contain variables such as mole fractions that are on the order of 10^{-6} or smaller, as well as enthalpy flows which, when specified in SI units for industrial systems, can be 10^{6} or larger. Making sure that these variables and constraints are properly scaled is necessary not just for solution speed but also solution accuracy. Osborne [14] performed an early investigation of scaling on the solution of eigenvalue problems. Van der Sluis [15] gives a method to provide near-optimal scaling for symmetric positive definite matrices. Van der Sluis [16] provides an in-depth analysis about how various automatic scaling schemes effect the error in the solution of linear equations via both Gaussian elimination and QR factorization. Braatz and Morari [17] demonstrate that optimal row and column scaling of a matrix can be generated by solving a convex semidefinite program, for which there are off-the-shelf solvers available. A point made by both Van der Sluis [16] and Braatz and Morari [17] is that minimizing the matrix condition number may not provide the most accurate solutions, sometimes increasing the solution error in typical use cases.

This work demonstrates the use of the IDAES diagnostics toolbox through its application to an existing IDAES model. A solvent-based capture flowsheet using monoethanol amine (MEA) as the solvent previously presented in Akula et al. [18] was chosen for robust optimization (RO) in PyROS [19], a Pyomo tool for robust optimization. A numerically robust flowsheet model is required as it needs to be solved for the hundreds of realization of uncertain parameters, so we applied model diagnostics and reformulation to it. In Section 2, a high-level overview of relevant concepts like Newton's Method, variable and constraint scaling, the matrix condition number, and the SVD is given. In Section 3, these methods are applied to the MEA flowsheet as a tutorial on how to use the IDAES toolbox. Issues with both variable scaling and model formulation are identified and solved. In Section 4, an approximate form of the enhancement factor model is derived that is much more numerically robust. Finally, the results are summarized in Section 5.

2. Methods

Some are born modelers, some become modelers through careful study, others have modeling thrust upon them. Therefore, the understanding of linear algebra, numerical methods, or mathematical programming of modelers varies greatly. In this section, we begin with a basic description of Newton's method, the matrix condition number, and the SVD. Then model scaling and SVD analysis are discussed in greater depth.

2.1. Preliminaries

When prototyping a model, it is typically best to solve a "square problem," i.e., one in which there are an equal number of free variables and equality constraints. The number of degrees of freedom can be checked by the function

```
idaes.core.util.model_statistics.degress_of_freedom
```

There are zero degrees of freedom in a square problem. Variable bounds can be present, as they are often helpful to keep the nonlinear solver from exploring areas with non-physical solutions or areas in which equations become undefined. For example, if a logarithm of a variable is taken, its lower bound should be set to zero. Inequality constraints can also be used for purposes like root selection for an equation of state [20].

However, neither variable bounds nor inequality constraints should be active at a solution. The inclusion of such a bound is effectively an equality constraint, reducing the number of degrees of freedom by 1. The system of equations then either becomes infeasible, which precludes a solution, or degenerate, which means that some of the equations are redundant. Different nonlinear solvers have different ways of handling degeneracy, but it either prevents or dramatically slows convergence to a solution. For this reason, "performance" inequality constraints and variable bounds that are not necessary to achieve a physical solution should be avoided.

The principal diagnostic methods used in this work utilize the Jacobian. When solving a root-finding problem

$$\begin{bmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{bmatrix} \coloneqq \mathbf{f}(\mathbf{x}) = 0 \tag{1}$$

the Jacobian matrix of f(x) is given by

$$\boldsymbol{J}(\mathbf{x}) \coloneqq \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2}$$

For a square problem, the Jacobian matrix is square. Each row of $\boldsymbol{J}(\mathbf{x})$ corresponds to an equality constraint and each column corresponds to a free variable. Therefore, we can examine $\boldsymbol{J}(\mathbf{x})$ to find clues about problems in equations and constraints. In the past, calculation of the Jacobian was difficult because derivatives had to be calculated either by hand or by finite differences. However, advances in algorithmic differentiation (AD) enable automatic, precise calculation of derivatives. Pyomo offers access to the AD capacities of the AMPL solver library (ASL) [21] through the PyNumero interface [22].

The most common methods of solving multivariate root-finding problems of the form of (1), like the one utilized in IPOPT [23], which is freely available and was used as a nonlinear solver in this paper, are variations on Newton's Method. The method approximates $f(\cdot)$ as linear and then repeatedly solves the linearized equation

$$\mathbf{f}(\mathbf{x}_k) + \boldsymbol{J}(\mathbf{x}_k)\Delta\mathbf{x}_k = 0 \tag{3}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k \tag{4}$$

until the condition

$$\left\|\mathbf{f}(\mathbf{x}_{k+1})\right\|_2 \leq \varepsilon \tag{5}$$

in which $\|\cdot\|_2$ is the vector 2-norm (Euclidean norm), is satisfied for some chosen tolerance ε. In a practical implementation, the Newton step (4) may be truncated either to account for variable bounds or because $\left\|\mathbf{f}(\mathbf{x}_{k+1})\right\|_2 > \left\|\mathbf{f}(\mathbf{x}_k)\right\|_2$, but it is desirable to take the full step when possible. A sign of a well-formulated method working on a well-formulated problem is when the full step (4) is frequently taken until a solution is reached.

A condition number of a matrix is a measure of the sensitivity of changes in the solution to a system of linear equations. For example, for the Newton step-finding problem

$$\boldsymbol{J}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k) \tag{6}$$

suppose we perturb the function output by $\delta\mathbf{f}_k$ and want to estimate the perturbation in the resulting step $\delta\mathbf{x}_k$

$$J(\mathbf{x}_k)(\Delta\mathbf{x}_k + \delta\mathbf{x}_k) = -(\mathbf{f}(\mathbf{x}_k) + \delta\mathbf{f}_k) \tag{7}$$

Use of the condition number for the 2-norm $\kappa_2(J(\mathbf{x}_k))$ gives us the bound

$$\left\|\delta\mathbf{x}_k\right\|_2 \leq \kappa_2(J(\mathbf{x}_k)) \left\|\delta\mathbf{f}_k\right\|_2 \tag{8}$$

In practice, such perturbations in the function output always exist from the round-off errors in floating point arithmetic. The relative error inherent in double precision floating point arithmetic is on the order of 10^{-16}. Furthermore, the termination condition (5) means that we can expect $\left\|\delta\mathbf{f}_k\right\|_2 \approx \varepsilon$ at the final iteration. Additionally, a large condition number $\kappa_2(J(\mathbf{x}_k))$ (greater than about 10^8) makes it difficult to solve the problem (6) numerically, increasing the time that Newton iterations take to solve or even causing it to diverge.

The condition number of $J(\mathbf{x})$ is given by

$$\kappa_2(J(\mathbf{x})) = \|J(\mathbf{x})\|_2 \,/\, \left\|J(\mathbf{x})^{-1}\right\|_2 \tag{9}$$

in which $\|\cdot\|_2$ is the operator norm induced by the vector 2-norm, provided $J(\mathbf{x})$ is full rank. A more useful formula comes from the singular value decomposition (SVD). We factorize

$$J(\mathbf{x}) = U\Sigma V^T \tag{10}$$

in which U and V are orthogonal matrices and

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix} \tag{11}$$

in which $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. If $J(\mathbf{x})$ is full rank, then $\sigma_n > 0$ and

$$\kappa_2(J(\mathbf{x})) = \sigma_1/\sigma_n \tag{12}$$

Therefore, we can improve the problem formulation by reducing the condition number by bringing the values of σ_1 and σ_n closer together. This can be done by appropriate scaling of variables and constraints.

2.2. Scaling

When working in SI units, it is common to have variables and equations that vary over many orders of magnitude, from mole fractions of trace components with magnitudes 10^{-6} to enthalpy flow values that can be up to 10^6. IDAES-IP offers a variety of tools to assist in scaling both variables and constraints. Scaling serves three purposes: first, to ensure that the convergence criterion (5) guarantees that all equations are satisfied without being unduly difficult to satisfy, second, to ensure that variables are sufficiently far from bounds so their bounds are not mistakenly identified as active constraints, and, third, to reduce the condition number of the Jacobian. A method exists [17] for calculating scaling factors for variables and constraints that minimize the condition number of the Jacobian by solution of a convex program. While this method achieves the third goal, there is no guarantee that it will achieve the first and second goals. Furthermore, as pointed out by both Van der Sluis [16] and Braatz and Morari [17], scaling to reduce uncertainty in the solution to a system of linear equation depends on the relative magnitude of the elements $\delta\mathbf{f}_k$, as well as any uncertainty in the matrix $J(\mathbf{x}_k)$.

Consider two equations, extracted from a larger system of equations that may occur in, for example, a heater unit model:

$$y_{\text{trace,in}} - y_{\text{trace,out}} = f_1(\mathbf{x}) \tag{13}$$

$$H_{\text{in}} + Q - H_{\text{out}} = f_2(\mathbf{x}) \tag{14}$$

$$\vdots \qquad = \quad \vdots$$

in which $y_{\text{trace,in}}$ and $y_{\text{trace,out}}$ are the inlet and outlet mole fractions of a trace component, H_{in} and H_{out} are the inlet and outlet enthalpy flows, and Q is the heat duty. If the initial values of $y_{\text{trace,out}}$ and H_{out} were off by a relative error of 100%, the error in (13) would be of the order 10^{-6} and the error in (14) of the order 10^{6}. If equations of these magnitudes are loaded into the same $\mathbf{f}(\cdot)$, significant error can exist in the values of mole fractions with (5) satisfied while the solver would strain to keep reducing the error in (14) far beyond what is significant. IDAES and Pyomo allow us to give the first equation a scaling factor of 10^{6} and the second one of 10^{-6} to bring their residuals to the same order of magnitude.

Let us note the effect of such scaling on the function and the Jacobian. We would replace (13) and (14) with

$$10^{6}(y_{\text{trace,in}} - y_{\text{trace,out}}) = \tilde{f}_1(\mathbf{x}) \tag{15}$$

$$10^{-6}(H_{\text{in}} + Q - H_{\text{out}}) = \tilde{f}_2(\mathbf{x}) \tag{16}$$

and the corresponding rows in the Jacobian would be replaced by

$$\boldsymbol{J}(\mathbf{x}) = \begin{bmatrix} 10^{6}\frac{\partial f_1}{\partial x_1} & \cdots & 10^{6}\frac{\partial f_1}{\partial x_n} \\ 10^{-6}\frac{\partial f_2}{\partial x_1} & \cdots & 10^{-6}\frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{17}$$

Now the coefficients in the Jacobian have values of extremely different orders of magnitude. Row 1 has norm of $10^{6}\sqrt{2}$ and row 2 has norm of $10^{-6}\sqrt{3}$. We can derive a lower bound for the condition number in terms of rows with largest and smallest 2-norm. Rearrange Jacobian rows so that they are sorted in descending order of 2-norm magnitude

$$\boldsymbol{J}(\mathbf{x}) = \begin{bmatrix} r_{\max} \\ \vdots \\ r_{\min} \end{bmatrix} \tag{18}$$

Then we have that

$$\left\|r_{\max}\right\|_2 = \left\| \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} r_{\max} \\ \vdots \\ r_{\min} \end{bmatrix} \right\| \leq \sigma_1 \tag{19}$$

and

$$\left\|r_{\min}\right\|_2 = \left\| \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{\max} \\ \vdots \\ r_{\min} \end{bmatrix} \right\| \geq \sigma_n \tag{20}$$

so we have that

$$\kappa_2(\boldsymbol{J}(\mathbf{x})) = \sigma_1/\sigma_n \geq \left\|r_{\max}\right\|_2 / \left\|r_{\min}\right\|_2 \tag{21}$$

Presently, the Jacobian matrix has a condition number greater than $\sqrt{6}/3 \cdot 10^{12}$. To reduce it, we also need to set scaling factors for variables. We define

$$\tilde{y}_{\text{trace,in}} \coloneqq 10^{6} y_{\text{trace,in}} \tag{22}$$

$$\tilde{H}_{\text{out}} \coloneqq 10^{-6} H_{\text{out}} \tag{23}$$

Suppose we had $y_{\text{trace,in}} = x_1$ and $H_{\text{out}} = x_n$. The Jacobian now looks like

$$\boldsymbol{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & 10^{12}\frac{\partial f_1}{\partial x_n} \\ 10^{-12}\frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{24}$$

The entry involving 10^{12} is not problematic because $\partial f_1/\partial x_n = 0$. In short, scaling a constraint *multiplies* the corresponding Jacobian row by that scaling factor, and scaling a variable *divides* the corresponding Jacobian column by that scaling factor.

It is not necessary to manually create transformed variables and constraints because Pyomo allows the user to store scaling factors associated with variables and constraints as part of the model. The unscaled model can then be transformed into the scaled model. This scaled model can then either be passed to the nonlinear solver or used for diagnostic purposes. In a model with thousands of variables and constraints, however, it is difficult to keep track of which variables and constraints have been scaled and whether the scaling factors assigned are appropriate. IDAES allows scaling methods associated with unit models and property models which scale internal constraints and variables so long as the user sets scaling factors for "base" quantities like flow rate, temperature, pressure, molar enthalpy, and component mole fractions.[1] However, due to the modular nature of IDAES models, it is difficult for a model developer to create a scaling scheme that works in every situation, and the user may need to manually set scaling factors for additional variables and equations.

The IDAES diagnostics toolbox, described in [4], has the methods

```
display_constraints_with_extreme_jacobians
display_variables_with_extreme_jacobians
```

that iterate over Jacobian rows and columns, respectively, to display those with 2-norms larger and smaller than some user-specified tolerances (by default 10^4 and 10^{-4}). These methods highlight areas of the model that require work, but the user still needs to determine appropriate values for scaling factors, as well as which variables and constraints need to be scaled. For example, rows 1 and 2 would be highlighted in the Jacobian of (17), when it is the variables, not the constraints, that require scaling.

Scaling is often an iterative process. Once appropriate scaling factors are assigned to the variables, bringing the Jacobian into the form (24), large or small entries can appear in additional rows, corresponding to other constraints, which would then need to be scaled. However, a large entry can correspond to a model that is intrinsically ill-conditioned. For example, when modeling a process with a large recycle ratio, large terms can appear in the corresponding Jacobian. Small impurities in the feed stream can become concentrated by factors of 100 or 1000 as a feature of the process. Scaling might be able to hide this feature, but it can cause a loss in accuracy.

2.3. SVD Analysis

Even when rows and columns with inappropriately large or small norms have been removed by scaling, the Jacobian's condition number, which can be revealed through the SVD of the Jacobian, can still be extremely large. Both σ_1 and σ_n appear directly in the expression for the condition number, so reducing σ_1 and increasing σ_n should help reduce it. In practice, σ_1 is usually reduced to a reasonable level by scaling rows and columns, so we will focus on increasing σ_n instead.

Consider the solution to the Newton step problem:

$$\boldsymbol{J}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k) \tag{6}$$

By decomposing $\boldsymbol{J}(\mathbf{x})$ using the SVD and using that to solve (6), we obtain

$$U\Sigma V^T\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k) \tag{25}$$

$$(-V\Sigma^{-1}U^T)(U\Sigma V^T)\Delta\mathbf{x}_k = (-V\Sigma^{-1}U^T)(-\mathbf{f}(\mathbf{x}_k)) \tag{26}$$

$$\Delta\mathbf{x}_k = -V\Sigma^{-1}U^T\mathbf{f}(\mathbf{x}_k) \tag{27}$$

In solving the perturbed problem, the constraint residual is projected by U^T into orthogonal components in the space of the singular values, scaled by the inverse singular values, and those scaled values are projected by V into the space of variables.

Looking at (27) another way

$$-U^T\mathbf{f}(\mathbf{x}_k) = \Sigma V^T\Delta\mathbf{x}_k \tag{28}$$

[1] At the time of publication, the IDAES scaling toolbox is undergoing a transition to a new API. The most up-to-date information can be found at https://idaes-pse.readthedocs.io/en/stable/explanations/scaling_toolbox/index.html, but, as of yet, new style scaler objects do not exist for many models.

Each column of U (row of U^T) is an orthogonal vector, associating elements of $\mathbf{f}(\mathbf{x}_k)$, i.e., particular constraints, with a singular value. Likewise, each column of V (row of V^T) is a vector associating elements of $\Delta\mathbf{x}_k$, i.e., particular variables, with a singular value. The SVD thus shows which constraints are being satisfied using which variables and the sensitivity of the variables with respect to constraints. If all constraints were linear, the relationship between variables and constraints would be global and hold for all $\mathbf{x}$. Since we are considering a nonlinear problem, this relationship holds only in a neighborhood of $\mathbf{x}_k$. Nevertheless, the SVD is good at identifying both local and global numerical singularities.

If the smallest singular value $\sigma_n << 1$, a small change in certain function values requires a large change in variable values. By looking at the constraints involved in the n^{th} left-singular vector u_n and the variables involved in n^{th} right singular vector v_n, we can attempt to determine the cause of this dysfunctional relationship. Because of the dense linear algebra involved, we expect most entries of u_n and v_n to be populated by nonzero numbers. A crude but effective method is to filter for indices that have values greater than some tolerance. We have found that the absolute values of 0.1 to 0.3 work well for this. If too many variables and constraints appear, increase the tolerance, and if too few appear, decrease it.

Ideally, we would want the condition number of $\boldsymbol{J}(\mathbf{x}_k)$ to be relatively small. In practice, however, we typically accomplish a condition number on the order of 10^6–10^8. That means filtering for singular values smaller than 10^{-8}–10^{-6} and attempting to remedy them. In general, there are four causes of small singular values:

1. Incorrect scaling of variables or constraints
2. Redundant equations, hinting at a problem that over-specifies some variables and under-specifies others, i.e., global degeneracy
3. A local singularity caused by evaluating $\boldsymbol{J}(\mathbf{x})$ at a point where it loses rank, i.e., local degeneracy
4. Attempting to solve a problem that is inherently ill-conditioned

Incorrect scaling is probably the most common cause of small singular values of values 10^{-12}–10^{-6}, followed by inherent ill-conditioning. For singular values smaller than 10^{-12}, the cause is typically local or global degeneracy. The next section presents a carbon capture application in which these concepts can be demonstrated.

3. Case Study: Post Combustion Carbon Capture Flowsheet

Analysis of a flowsheet, shown in Figure 1, in which CO_2 is captured from flue gas from a 740 MWe Natural Gas Combined Cycle power plant by absorption into monoethanolamine (MEA) provides several examples in which these diagnostic tests were of great assistance in model refinement and increasing robustness. A version of this flowsheet was first presented in [18], but despite the techniques used to provide robust solutions, such as a four-stage initialization routine for the absorption and stripping columns, it remained fragile and prone to failure. It was desirable to improve convergence and numerical robustness so that the flowsheet could be used for robust design optimization. The flowsheet consists of an absorption column, stripper column, lean-rich heat exchanger, and balance-of-plant equipment. It is divided into two sub-flowsheets, an absorber section and a stripper section, which are solved independently before being linked and solved together. We begin with a flowsheet that has some partial scaling applied to the column model but does not have scaling for any other models. The column model has already been reformulated to remove division in constraints where possible, which helps avoid bad numeric behavior when the denominator of an expression is nearly zero at intermediate Newton iterations. For an unscaled or partially-scaled model, the output of the diagnostic toolbox can be hundreds of lines long. The output length is mostly the result of scaling issues in indexed variables or constraints resulting in those variables or constraints being printed for each value of their indices. Since the column model is discretized into 40 nodes along its length, 10 badly-scaled equations result in 400 entries. Therefore, the full output cannot be displayed here, but some representative entries can be shown.

3.1. Jacobian Analysis

Here, we run the diagnostic tools after a successful solution of the flowsheet formulated as a square problem. They can be run even after failures to solve a flowsheet, but care should be taken as variables may not have realistic values, which is reflected in the model's Jacobian. The following is one line (out of hundreds) produced by the `display_variables_with_extreme_jacobians` command from the `DiagnosticsToolbox`:

```
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0].temperature: 1.997E+08
```

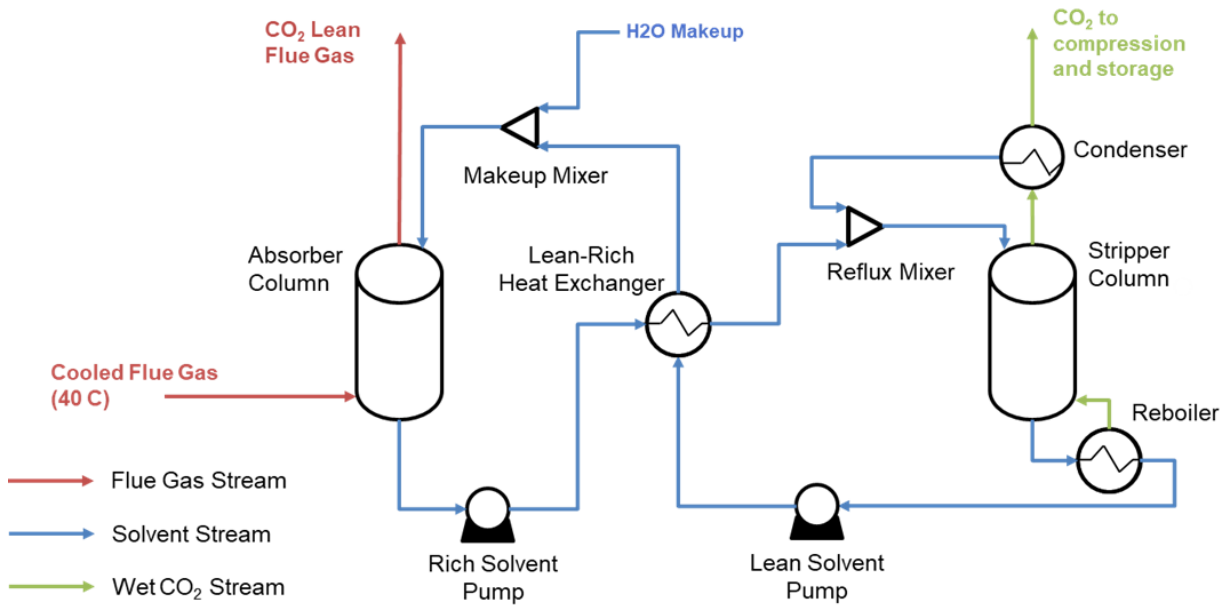


Figure 1: Process flow diagram of MEA-based carbon capture flowsheet.

This entry indicates a column norm of $2 \cdot 10^8$. We can inspect the associated column of the Jacobian to find which rows are associated with this large value. In IDAES-IP, we can do this using the `display_constraints_including_variable` function in the `SVDToolbox`.

```
fs.stripper_section.reflux_mixer.enthalpy_mixing_equations[0.0]: 1.997e+08
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_k_eq_constraint[bicarbonate]: 5.364e+00
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_k_eq_constraint[carbamate]: 6.551e+00
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,MEACOO_-]: 1.514e+00
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,HCO3_-]: 2.333e-01
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,MEA_+]: 1.748e+00
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,H2O]: 3.183e+01
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,MEA]: 1.129e+00
fs.stripper_section.reflux_mixer.rich_solvent_state[0.0]
    .log_conc_mol_phase_comp_true_eq[Liq,CO2]: 5.685e-03
fs.rich_temperature: 1.000e+02
```

From inspection, the problematic Jacobian entry is associated with a mixer enthalpy mixing equation that is not yet scaled.

To fix the problem, we need to determine an appropriate scaling factor. We can calculate such a value from scaling factors for both the molar flow and the specific molar enthalpy. Finding the molar flow scaling factor is easy; it is generally observed that the molar flow rates are on the order of 10^3–10^4 throughout the flowsheet, so we choose $3 \cdot 10^{-4}$ as a scaling factor for the molar flow rate. Determining a scaling factor for the specific molar enthalpy is harder. Enthalpy can be either positive or negative in different areas of the flowsheet, so its order of magnitude does not make a good scaling factor. By evaluating the liquid-phase enthalpy in the reflux mixer, the water makeup mixer,

and the reboiler, we get values of −42200, −43600, and −38800. With an apparent range of 4800, a scaling factor of $3 \cdot 10^{-4}$ is also a good choice for this variable. Values for scaling factors do not have to be exact.

The process of removing this scaling issue took us through at least three Pyomo sub-models (and associated Python files) to determine a process through which the scaling factor for one equation was calculated. However, had reasonable default scaling factors for the property sub-model been set by the model developer/user ahead of time, this issue would never have arisen. Note that these default values are application specific. Use of a different amine solvent would require the user to revisit these values.

Next, the `display_row/column_with_extreme_jacobians` methods called these equations describing heat transfer in the column to our attention:

$$h_{\mathrm{V}} a_e \mathrm{Ack} = -(\underline{C}_{p,\mathrm{CO_2,V}} N_{\mathrm{CO_2,V}} + \underline{C}_{p,\mathrm{H_2O,V}} N_{\mathrm{H_2O,V}}) \tag{29}$$

$$h'_{\mathrm{V}} a_e (1 - \exp(\mathrm{Ack})) = \underline{C}_{p,\mathrm{CO_2,V}} N_{\mathrm{CO_2,V}} + \underline{C}_{p,\mathrm{H_2O,V}} N_{\mathrm{H_2O,V}} \tag{30}$$

in which h_{V} is the heat transfer coefficient from liquid to vapor, a_e is the effective area of heat transfer per unit column volume, $\underline{C}_{p,i,\mathrm{V}}$ is the heat capacity of species i, $N_{i,\mathrm{V}}$ is the molar flux of species i from vapor to liquid, and Ack is the Ackerman factor, which is a measure of how much diffusive heat fluxes are distorted by convective heat transfer. It was discovered that $|\mathrm{Ack}| \leq 10^{-2}$ for operating conditions of interest and was frequently on the order of 10^{-5}. This causes ill-conditioning in both equations because the factors Ack and $1 - \exp(\mathrm{Ack})$ became close to zero. Because Ack can vary over several orders of magnitude depending on location and operating conditions, assigning a consistent scaling factor is difficult.

If we first rearrange (28) and (29) to make the relationship between h_{V} and h'_{V} more clear

$$h'_{\mathrm{V}} = h_{\mathrm{V}} \frac{\mathrm{Ack}}{\exp(\mathrm{Ack}) - 1} \tag{31}$$

several solutions are possible. The function

$$\theta(x) = \frac{x}{\exp(x) - 1} \tag{32}$$

has an indeterminate form at $x = 0$, but a well-defined Taylor expansion

$$\theta(x) = 1 - \frac{x}{2} + \frac{x^2}{12} + O(x^4) \tag{33}$$

A Taylor approximation could be substituted for $\theta(Ack)$ in (31), but $\theta(x)$ could also be implemented as an external function in Pyomo and IDAES, switching between the full form (32) and Taylor form (33) based on $|x|$. Both options were eventually implemented. An external function for $\theta(x)$ is implemented in IDAES with a sixth-order Taylor approximation, but, because not every solver or solver interface in Pyomo supports external functions, the trivial Taylor approximation $\theta(x) \approx 1$ is presently used in the MEA flowsheet. This trivial approximation introduces some error, but, since the effect of the Ackerman correction on the heat transfer coefficient is 1% or less in this application, the loss of accuracy was deemed acceptable.

3.2. SVD Analysis

The SVD also tells us valuable information about the state of the flowsheet. Its condition number is $9.2 \cdot 10^{17}$, so the matrix is singular to machine precision. The smallest singular value is $\sigma_n = 3.1 \cdot 10^{-10}$. Using a tolerance for the singular vectors of 0.1, we find the following variables involved:

```
fs.stripper_section.reflux_mixer.reflux_state[0.0].log_conc_mol_phase_comp_true[Liq,HCO3_-]
fs.stripper_section.reflux_mixer.reflux_state[0.0].log_conc_mol_phase_comp_true[Liq,MEA_+]
fs.stripper_section.reflux_mixer.reflux_state[0.0].log_conc_mol_phase_comp_true[Liq,MEA]
fs.stripper_section.reflux_mixer.reflux_state[0.0].log_conc_mol_phase_comp_true[Liq,MEACOO_-]
```

and the following constraints involved:

```
fs.stripper_section.condenser.liquid_phase[0.0].appr_to_true_species[Liq,MEA]
fs.stripper_section.condenser.liquid_phase[0.0].true_mole_frac_constraint[Liq,HCO3_-]
fs.stripper_section.condenser.liquid_phase[0.0].true_mole_frac_constraint[Liq,MEA_+]
fs.stripper_section.condenser.liquid_phase[0.0].true_mole_frac_constraint[Liq,MEA]
fs.stripper_section.condenser.liquid_phase[0.0].log_conc_mol_phase_comp_true_eq[Liq,HCO3_-]
fs.stripper_section.condenser.liquid_phase[0.0].log_conc_mol_phase_comp_true_eq[Liq,MEA_+]
fs.stripper_section.condenser.liquid_phase[0.0].log_conc_mol_phase_comp_true_eq[Liq,MEA]
fs.stripper_section.condenser.liquid_phase[0.0].appr_to_true_species[Liq,HCO3_-]
fs.stripper_section.condenser.liquid_phase[0.0].appr_to_true_species[Liq,MEA_+]
```

All these variables and constraints occur in the condenser and the condensate stream to the reflux mixer, which makes interpretation of the problem easier. However, if multiple degeneracies are present, they can mix between different tiny singular values, so sometimes additional analysis is necessary to separate different degeneracies.

Because the property model in this flowsheet does not account for amine volatility, the mole fraction of MEA in the condenser is effectively zero.[2] All the variables implicated here are dissociation species of MEA, which similarly have concentrations of effectively zero. The constraints are likewise the disassociation equations for MEA. Thus, the absence of MEA causes degeneracy in the system of equations governing the dissociation reactions. Different strategies can be employed to overcome this degeneracy. The easiest is to increase the mole fraction of MEA to around 10^{-4}. However, this merely mitigates the ill-conditioning—it does not remove it—and the addition of extra MEA in the system could cause problems with material balances converging.

IDAES-IP allows the user to define property packages that bundle together thermodynamic calculations into a single sub-model that can then be employed in different unit models. The solution we employed was to create a duplicate property package without the dissociation reactions and use it for the condenser and reflux mixer. That solution works only because the liquid phase property sub-model does not rely on ion concentrations for the calculation of enthalpy. For one that requires ion concentrations to calculate the enthalpy of mixing, like eNRTL, another solution would need to be devised.

3.3. Limits of Scaling and Reformulation

Not every problem that can be discovered using these diagnostic tools has a simple solution. Such is the case with the system of equations for the enhancement factor for material transfer in reactive systems. The enhancement factor is the ratio of material transport that occurs when chemical reaction with the solvent is considered to that which would occur without chemical reaction. The enhancement factor model, taken from [25] for the MEA-CO_2 system, has ten tightly coupled equations, given in Section 4.1. However, the core numerical issues derive from two equations:

$$E = 1 + (E^*_\infty - 1)\frac{1 - \Upsilon^i_{\mathrm{MEA}}}{1 - \Upsilon^b_{\mathrm{CO_2}}} \tag{34}$$

$$E = \mathrm{Ha}\sqrt{\Upsilon^i_{\mathrm{MEA}}}\frac{1 - \Upsilon^*_{\mathrm{CO_2}}}{1 - \Upsilon^b_{\mathrm{CO_2}}} \tag{35}$$

in which E is the enhancement factor, E^*_∞ is the "instantaneous enhancement factor" (a theoretical maximum value for the enhancement factor if the absorption reaction occurred instantaneously), $\Upsilon^i_{\mathrm{MEA}}$ is a dimensionless concentration of MEA at the vapor/liquid interface, $\Upsilon^*_{\mathrm{CO_2}}$ is the dimensionless concentration of CO_2 that would be at chemical equilibrium with the other species at the interface, $\Upsilon^b_{\mathrm{CO_2}}$ is the dimensionless concentration of CO_2 in the bulk liquid, and Ha is the Hatta number (the ratio of reaction film to the rate of diffusion in the film). Absorption of CO_2 happens when $\Upsilon^b_{\mathrm{CO_2}} < 1$, desorption of CO_2 happens when $\Upsilon^b_{\mathrm{CO_2}} > 1$, and equilibrium occurs when $\Upsilon^b_{\mathrm{CO_2}} = 1$.

The problem occurs near equilibrium because the expression $1 - \Upsilon^b_{\mathrm{CO_2}}$ is nearly equal to zero and the quotients in (34) and (35) are nearly singular. At solutions to the system of equations, numerical tests show that $1 - \Upsilon^*_{\mathrm{CO_2}}$ and $1 - \Upsilon^i_{\mathrm{MEA}}$ also approach zero, so the quotient remains defined. However, since we now must deal with a multivariate function, we have been unsuccessful at removing the singularity by use of a Taylor series like we did with the Ackerman

[2] In reality, some amine would be present in the condenser, but in a small enough amount that these equations would remain problematic. Discussion of amine emissions is beyond the scope of this paper, but interested readers can refer to [24] for an extensive review.

factor. The expressions $1-\Upsilon^{b}_{CO_2}$ and $1-\Upsilon^{*}_{CO_2}$ have the same sign at any solution to the activity factor system. However, contrary to what is stated in [25], there exists at least one scenario in which $1-\Upsilon^{b}_{CO_2}$ has the opposite sign as $1-\Upsilon^{i}_{MEA}$. When CO_2 is desorbing at low temperatures, the rate of reaction can become slow enough that the rate of desorption is lower with reaction than it would be without reaction, and the enhancement factor drops below one. That fact limits possible reformulations of these two equations.

The equations were ultimately reformulated in several steps. First, a log variable S_{CO_2} was introduced for one of the singular ratios:

$$(1-\Upsilon^{b}_{CO_2})\exp(S_{CO_2}) = 1-\Upsilon^{*}_{CO_2} \tag{36}$$

so that (35) could be rewritten as the linear equation

$$log\, E = log\, Ha + \frac{1}{2} log \Upsilon^{i}_{MEA} + S_{CO_2} \tag{37}$$

in which $log\, E$, $log\, Ha$, and $log \Upsilon^{i}_{MEA}$ are additional log variables defined using equations of the form

$$\exp(log\, x) = x \tag{38}$$

To be completely clear, $log\, x$ is a single variable, not an expression denoting the logarithm of the variable x, which we denote $\log(x)$. We have that $log\, x = \log(x)$ at solutions of the model, i.e., when $\mathbf{f}(\mathbf{x}) = 0$, but equality does not necessarily hold at intermediate Newton iterates. The benefit of implicitly taking the logarithm in equations like (36) and (38) instead of using a log function in (37) is that the variable $log\, x$ can maintain numerical accuracy even when the values of x are extremely small or extremely large.

It is possible to use the same technique as (37) for the ratio of MEA to CO_2 in (34). However, that implicitly assumes that $1-\Upsilon^{*}_{CO_2}$ and $1-\Upsilon^{i}_{MEA}$ have the same sign, which is true for most operating conditions of interest but is not true for desorption at low temperatures. The resulting reformulation of (34) is then

$$\exp(log\, E)(1-\Upsilon^{b}_{CO_2}) = \exp(\eta)(1-\Upsilon^{i}_{MEA}) \tag{39}$$

$$\exp(\eta) = E^{*} - 1 \tag{40}$$

in which the log variable η has been introduced for the instantaneous enhancement factor minus one because there is a simple expression for η in terms of logarithms of other variables.

Unfortunately, both (34) and (39) become degenerate when $\Upsilon^{b}_{CO_2} = 1$. However, this reformulated enhancement factor model is reliable enough to use as for deterministic optimization. Figure 2 shows the convergence of the original and reformulated stand-alone column models, while Figure 3 shows the convergence of the original and reformulated full flowsheet models (with all changes, not just those to the enhancement factor calculations). The numerical robustness of the reformulated and scaled models is significantly improved over that of the original model.

4. Equilibrium Enhancement Factor Model

The reformulation method presented in Section 3.3 is adequate for deterministic optimization. However, robust optimization (RO) requires solutions of not one single optimization problem, but hundreds with different values of uncertain parameters. A more direct approach to remove the degenerate equations was needed. The strategy we used was to derive a function giving the limit of the enhancement factor as the vapor partial pressure approaches the equilibrium pressure and use that explicit function for the equilibrium enhancement factor as the enhancement factor model rather than solving the system of degenerate equations.

4.1. Gaspar and Fosbøl Model

We start with a general absorption chemical reaction:

$$\nu_A\,A + \nu_B\,B \rightleftharpoons \nu_C\,C + \nu_D\,D \tag{41}$$

in which the volatile species A reacts with the solvent B to create products C and D. The rate of reaction is governed by the kinetics

$$\text{rate} = k(C^{b}_{A})^{m}(C^{b}_{B})^{n} \tag{42}$$

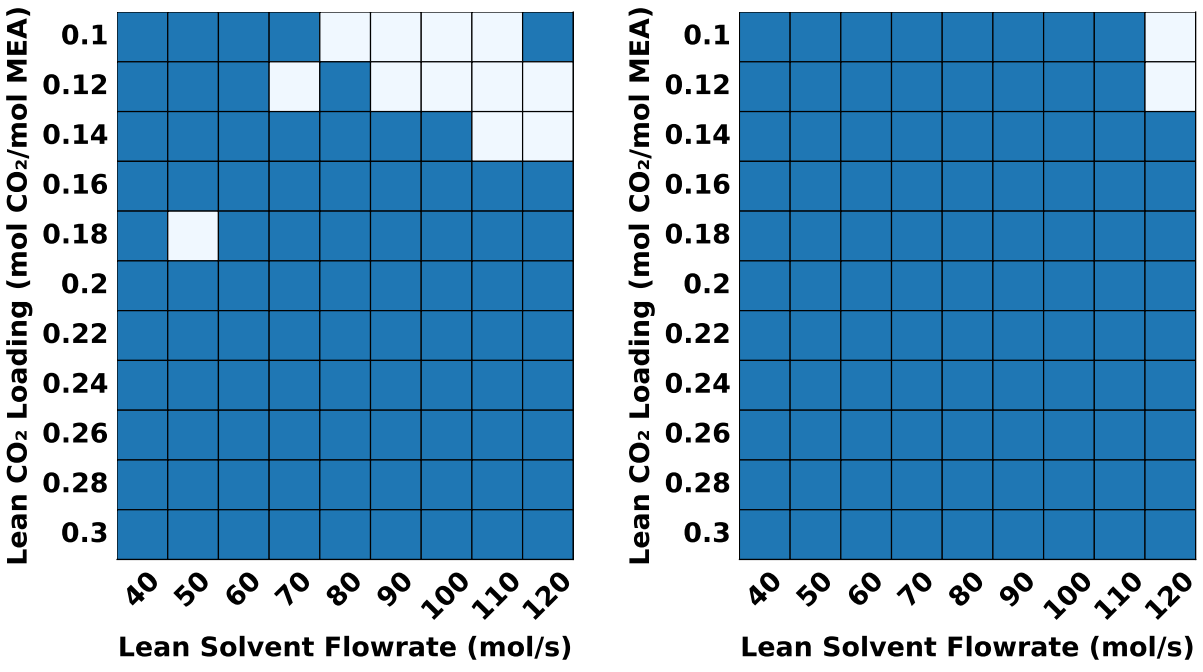


Figure 2: Convergence of the pilot-scale stand-alone column model depending on inlet parameters (a) before and (b) after the scaling and reformulations detailed in Section 3. A filled square indicates it converged and an unfilled square indicates it did not converge. In the reformulated model, divergence typically is a result of $\Upsilon^*_{CO_2}$ incorrectly converging to 1, resulting in (36) and (39) becoming degenerate.

in which k is the rate constant, C_A^b and C_B^b are the molar concentrations of species A and B, respectively, in the bulk fluid, and m and n are reaction orders with respect to A and B, respectively. Furthermore, $m \geq 1$ must be an integer. [3]

The Gaspar and Fosbøl (GF) enhancement factor model presented in [25] takes the form of a tightly-coupled system of 10 equations and 10 unknowns:

$$E = 1 + (E^*_\infty - 1)\frac{1 - \Upsilon^i_B}{1 - \Upsilon^b_A} \tag{43}$$

$$E = \mathrm{Ha}\sqrt{\frac{\nu_A}{m}}\left(\frac{1 - \Upsilon^*_A}{1 - \Upsilon^b_A}\right)\sqrt{\left(\Upsilon^i_B\right)^n Q_m\left(\Upsilon^*_A\right)} \tag{44}$$

$$\left(\Upsilon^*_A\right)^{\nu_A} = \left(\Upsilon^b_A\right)^{\nu_A}\left(\frac{\left(\Upsilon^i_C\right)^{\nu_C}\left(\Upsilon^i_D\right)^{\nu_D}}{\left(\Upsilon^i_B\right)^{\nu_B}}\right) \tag{45}$$

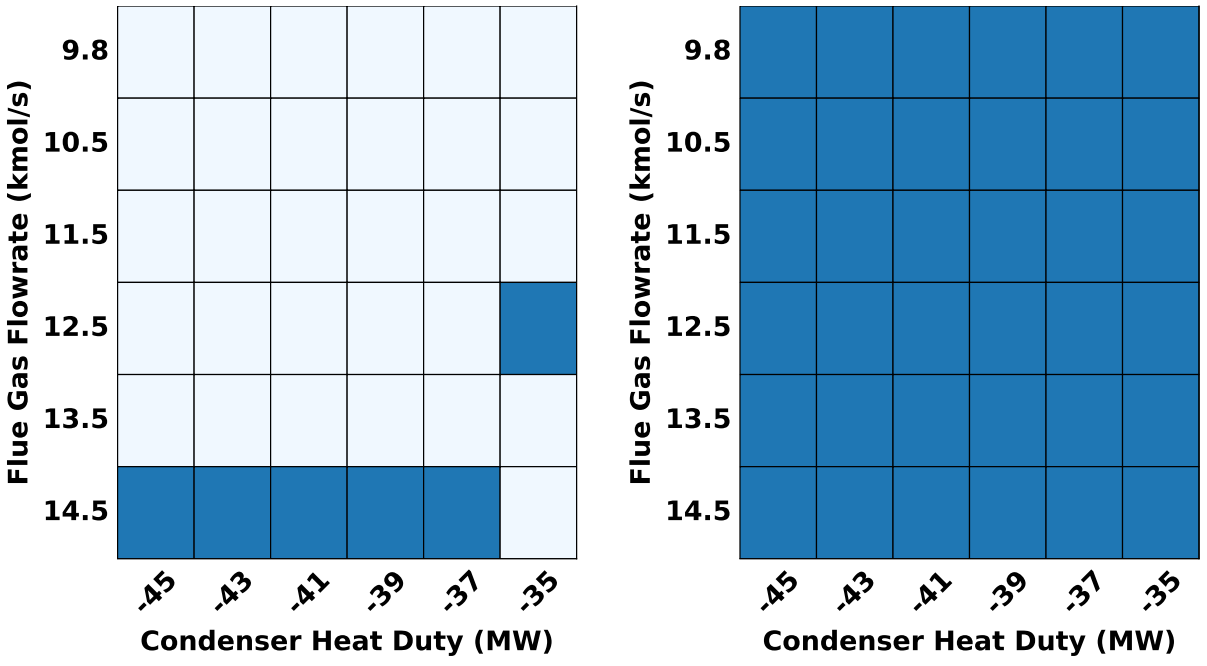
$$\Upsilon^i_C = 1 + R_C(1 - \Upsilon^i_B) \tag{46}$$

$$\Upsilon^i_D = 1 + R_D(1 - \Upsilon^i_B) \tag{47}$$

$$E^*_\infty = 1 + R_A \tag{48}$$

$$\mathrm{Ha} = \frac{1}{k_L}\sqrt{\frac{2}{m+1}k(C^b_\mathrm{A})^{m-1}(C^b_\mathrm{B})^n D_\mathrm{A}} \tag{49}$$

[3] Restrictions on m and n are not stated in [25]. These equations remain sensible for any real value of n. If m takes on some noninteger value strictly greater than zero, the function $Q_m(x)$ cannot be represented as a polynomial, but is instead a rational function with fractional powers and a removable singularity at $x = 1$.



(a) (b)

Figure 3: Convergence of the plant-scale full flowsheet model depending on inlet parameters (a) before and (b) after the scaling and reformulations detailed in Section 3. A filled square indicates it converged and an unfilled square indicates it did not converge.

$$\Upsilon_A^b = \frac{C_{\mathrm{A}}^b}{C_{\mathrm{A}}^i} \tag{50}$$

$$Ek_L(C_{\mathrm{A}}^i - C_{\mathrm{A}}^b) = k_V(P_{\mathrm{A}}^b - P_{\mathrm{A}}^i) \tag{51}$$

$$P_{\mathrm{A}}^i = \mathrm{He} C_{\mathrm{A}}^i \tag{52}$$

in which E is the enhancement factor, Ha is the Hatta number, C_j^b is the bulk liquid concentration of species j, C_j^i is the interfacial liquid concentration of species j, D_j is the liquid-phase diffusion coefficient for species j, k_L and k_V are the liquid and vapor mass transfer coefficients, P_{A}^b is the bulk vapor phase partial pressure of species A, P_{A}^i is the interfacial partial pressure of species A, He is the Henry's Law constant, and

$$Q_m(x) \coloneqq \sum_{j=0}^{m-1} (m-j)x^j \tag{53}$$

$$R_j \coloneqq \frac{\nu_j D_B C_B^b}{\nu_B D_j C_j^b} \text{ for } j \in \{A, C, D\} \tag{54}$$

The unknowns solved for in this system of 10 equations are E, E_∞^*, Υ_A^b, Υ_A^*, Υ_B^i, Υ_C^i, Υ_D^i, Ha, C_{A}^i, and P_{A}^i. Many of these equations can be eliminated by simple variable substitution (and, indeed, are implemented in the IDAES-IP model as Pyomo `Expressions` instead of `Constraints`), but (43) and (44) cannot be handled in such a fashion.[4]

[4]Note that (44) has a different form than the corresponding equation 22 in [25]. The equation there, as written, results in negative values for the enhancement factor during desorption. This is avoided by factoring out $(1-\Upsilon_A^*)^2$ from the polynomial under the square root and using replacing $\sqrt{(1-\Upsilon_A^*)^2}$ with $1-\Upsilon_A^*$ instead of $\left|1-\Upsilon_A^*\right|$. as indeed is done in [25] when the equation is applied to the example of absorption of CO_2 in MEA. This reformulation is not merely a point of pedantry, but required in order to obtain a physical solution to (44) in any numerical solution method.

At equilibrium, we have $P^i_{\mathrm{A}} = P^b_{\mathrm{A}}$ and $C^i_{\mathrm{A}} = C^b_{\mathrm{A}}$, thus (50) shows that $\Upsilon^b_A = 1$. In [25], Υ^i_B is defined $\Upsilon^i_B \coloneqq C^i_{\mathrm{B}}/C^b_{\mathrm{B}}$, so it also equals one at equilibrium. Substitution of (46) and (47) into (45) shows that $\Upsilon^*_A = 1$ as well. That causes (43) and (44) to contain the indeterminate forms of $0/0$. They cannot be directly evaluated for the enhancement factor E at equilibrium, so in order to obtain a value a limit must be evaluated as an appropriate perturbation from equilibrium approaches zero.

We rewrite the bulk vapor phase partial pressure of A as a perturbation p from the partial pressure in equilibrium with the bulk liquid phase

$$p \coloneqq P^b_A - \mathrm{He} C^b_A \tag{55}$$

Then we can combine (51) and (52) to obtain

$$C^i_{\mathrm{A}} = C^b_{\mathrm{A}} + \frac{k_V p}{E k_L + k_V \mathrm{He}} \tag{56}$$

The convenience of this perturbation is not just in the expression for C^i_{A}, however, but in the fact that many quantities in Equations (43) to (52) are not functions of p. As a result, the process of taking the limit as $p \to 0$ is much simpler. We define

$$\hat{E} \coloneqq \lim_{p\to 0} E \tag{57}$$

to simplify notation.

By rearrangement of (43), we have that

$$\lim_{p\to 0} \frac{E-1}{E^*_\infty - 1} = \frac{\hat{E}-1}{E^*_\infty - 1} = \lim_{p\to 0} \frac{1-\Upsilon^i_B}{1-\Upsilon^b_A} \tag{58}$$

The quantity on the right hand side reduces to the indeterminate form $0/0$ when $p = 0$. Therefore, we apply L'Hôpital's rule to obtain

$$\frac{\hat{E}-1}{E^*_\infty - 1} = \lim_{p\to 0} \frac{d\Upsilon^i_B/dp}{d\Upsilon^b_A/dp} \tag{59}$$

To solve for these quantities, we next take the limit of (44)

$$\hat{E} = \lim_{p\to 0} \mathrm{Ha} \sqrt{\frac{\nu_A}{m}} \left(\frac{1-\Upsilon^*_A}{1-\Upsilon^b_A}\right) \sqrt{\left(\Upsilon^i_B\right)^n Q_m\left(\Upsilon^*_A\right)} \tag{60}$$

$$= \mathrm{Ha} \sqrt{\frac{\nu_A}{m}} \left(\lim_{p\to 0} \frac{1-\Upsilon^*_A}{1-\Upsilon^b_A}\right) \left(\lim_{p\to 0} \sqrt{\left(\Upsilon^i_B\right)^n Q_m\left(\Upsilon^*_A\right)}\right) \tag{61}$$

We have that $Q_m(1) = m(m+1)/2$, and therefore

$$\hat{E} = Ha \sqrt{\frac{\nu_A(m+1)}{2}} \left(\lim_{p\to 0} \frac{1-\Upsilon^*_A}{1-\Upsilon^b_A}\right) \tag{62}$$

so all that remains is evaluation of the limit of the indeterminate form. We once again apply L'Hôpital's rule to obtain

$$\lim_{p\to 0} \frac{1-\Upsilon^*_A}{1-\Upsilon^b_A} = \lim_{p\to 0} \frac{d\Upsilon^*_A/dp}{d\Upsilon^b_A/dp} \tag{63}$$

To relate $d\Upsilon^*_A/dp$, $d\Upsilon^b_A/dp$, and $d\Upsilon^i_B/dp$, we use (45). Taking the logarithm of both sides and then differentiating, we obtain

$$\nu_A \log\left(\Upsilon^*_A\right) = \nu_A \log\left(\Upsilon^b_A\right) + \nu_C \log\left(\Upsilon^i_C\right) + \nu_D \log\left(\Upsilon^i_D\right) - \nu_B \log\left(\Upsilon^i_B\right) \tag{64}$$

$$\frac{\nu_A}{\Upsilon^*_A}\frac{d\Upsilon^*_A}{dp} = \frac{\nu_A}{\Upsilon^b_A}\frac{d\Upsilon^b_A}{dp} + \frac{\nu_C}{\Upsilon^i_C}\frac{d\Upsilon^i_C}{dp} + \frac{\nu_D}{\Upsilon^i_D}\frac{d\Upsilon^i_D}{dp} - \frac{\nu_B}{\Upsilon^i_B}\frac{d\Upsilon^i_B}{dp} \tag{65}$$

Differentiating (46) and (47) give us expressions for $d\Upsilon^i_C/dp$ and $d\Upsilon^i_D/dp$:

$$\frac{d\Upsilon^i_C}{dp} = -R_C\frac{d\Upsilon^i_B}{dp} \tag{66}$$

$$\frac{d\Upsilon^i_D}{dp} = -R_D\frac{d\Upsilon^i_B}{dp} \tag{67}$$

Substituting these expressions into (65) and rearranging yields

$$\frac{d\Upsilon^*_A}{dp} = \frac{\Upsilon^*_A}{\Upsilon^b_A}\frac{d\Upsilon^b_A}{dp} - \frac{\Upsilon^*_A}{\nu_A}\frac{d\Upsilon^i_B}{dp}\left(\frac{\nu_C R_C}{\Upsilon^i_C} + \frac{\nu_D R_D}{\Upsilon^i_D} + \frac{\nu_B}{\Upsilon^i_B}\right) \tag{68}$$

This expression, in turn, can be substituted into (63) to obtain

$$\lim_{p\to 0}\frac{1-\Upsilon^*_A}{1-\Upsilon^b_A} = \lim_{p\to 0}\frac{\Upsilon^*_A}{\Upsilon^b_A} - \frac{\Upsilon^*_A}{\nu_A}\left(\frac{\nu_C R_C}{\Upsilon^i_C} + \frac{\nu_D R_D}{\Upsilon^i_D} + \frac{\nu_B}{\Upsilon^i_B}\right)\left(\frac{d\Upsilon^i_B/dp}{d\Upsilon^b_A/dp}\right) \tag{69}$$

$$= 1 - \frac{1}{\nu_A}\left(\nu_C R_C + \nu_D R_D + \nu_B\right)\left(\lim_{p\to 0}\frac{d\Upsilon^i_B/dp}{d\Upsilon^b_A/dp}\right) \tag{70}$$

We then use (59) to eliminate the limit of derivatives to obtain

$$\lim_{p\to 0}\frac{1-\Upsilon^*_A}{1-\Upsilon^b_A} = 1 - \frac{1}{\nu_A}\left(\nu_C R_C + \nu_D R_D + \nu_B\right)\left(\frac{\hat{E}-1}{E^*_\infty - 1}\right) \tag{71}$$

Finally, substituting this equation into (61) we obtain

$$\hat{E} = Ha\sqrt{\frac{\nu_A(m+1)}{2}}\left(1 - \frac{1}{\nu_A}\left(\nu_C R_C + \nu_D R_D + \nu_B\right)\left(\frac{\hat{E}-1}{E^*_\infty - 1}\right)\right) \tag{72}$$

This equation can be solved for $\hat{E}$, yielding

$$\hat{E} = \frac{\mathrm{Ha}\sqrt{\frac{m+1}{2\nu_A}}\left(\nu_A + \frac{\nu_C R_C + \nu_D R_D + \nu_B}{E^*_\infty - 1}\right)}{1 + \mathrm{Ha}\sqrt{\frac{m+1}{2\nu_A}}\left(\nu_C R_C + \nu_D R_D + \nu_B\right)} \tag{73}$$

4.2. Application to MEA-CO_2 System

A pseudo-second order rate law is used for the MEA-CO_2 system:

$$\text{rate} = k_{\text{eff}} C_{\text{MEA}} C_{\text{CO}_2} \tag{74}$$

in which

$$k_{\text{eff}} = k_{\text{MEA}} C_{\text{MEA}} + k_{\text{H}_2\text{O}} C_{\text{H}_2\text{O}} \tag{75}$$

which takes into account both kinetically-important reaction mechanisms

$$\text{CO}_2 + 2\,\text{MEA} \longrightarrow \text{MEAH}^+ + \text{MEACOO}^- \tag{76}$$

$$\text{CO}_2 + \text{MEA} + \text{H}_2\text{O} \longrightarrow \text{MEAH}^+ + \text{HCO}_3{}^- \tag{77}$$

Therefore, we have $\nu_A = \nu_B = \nu_C = \nu_D = 1$ and $m = n = 1$. Thus, (73) becomes

$$\hat{E} = \frac{\mathrm{Ha}\left(1 + \frac{R_{\mathrm{MEAH^+}} + R_{\mathrm{MEACOO^-}} + 1}{E_\infty^* - 1}\right)}{1 + \mathrm{Ha}\left(R_{\mathrm{MEAH^+}} + R_{\mathrm{MEACOO^-}} + 1\right)} \tag{78}$$

This explicit formula is significantly more numerically robust than even the first reformulation presented in Section 3.3. Using this formula, the flowsheet model can be reliably solved by PyROS in the context of a large-scale optimization problem, with preliminary results presented in Sherman et al. [19]. However, this formula computes the enhancement factor at the equilibrium pressure of CO_2, not the true partial pressure in the vapor phase, so using it for the entire column model is an approximation. Since we expect the difficulty of material transfer at the pinch pressure to dominate the column design (where the enhancement factor is close to the equilibrium enhancement factor), we expect it to accurately predict overall CO_2 capture in the case of high capture.

To test for the MEA-CO_2 system, a series of simulations were carried out on the MEA flowsheet while varying the liquid inlet temperature (T_L), the lean CO_2 loading (ratio of moles absorbed CO_2 to moles MEA, denoted as α), and the ratio of the liquid molar flow to the vapor molar rate flow (L/G). Simulations were performed for both the original enhancement factor model and the new explicit model, allowing a comparison between the enhancement factor values, CO_2 vapor pressure, and equilibrium pressure. Figure 4 shows plots with different values of (L/G).

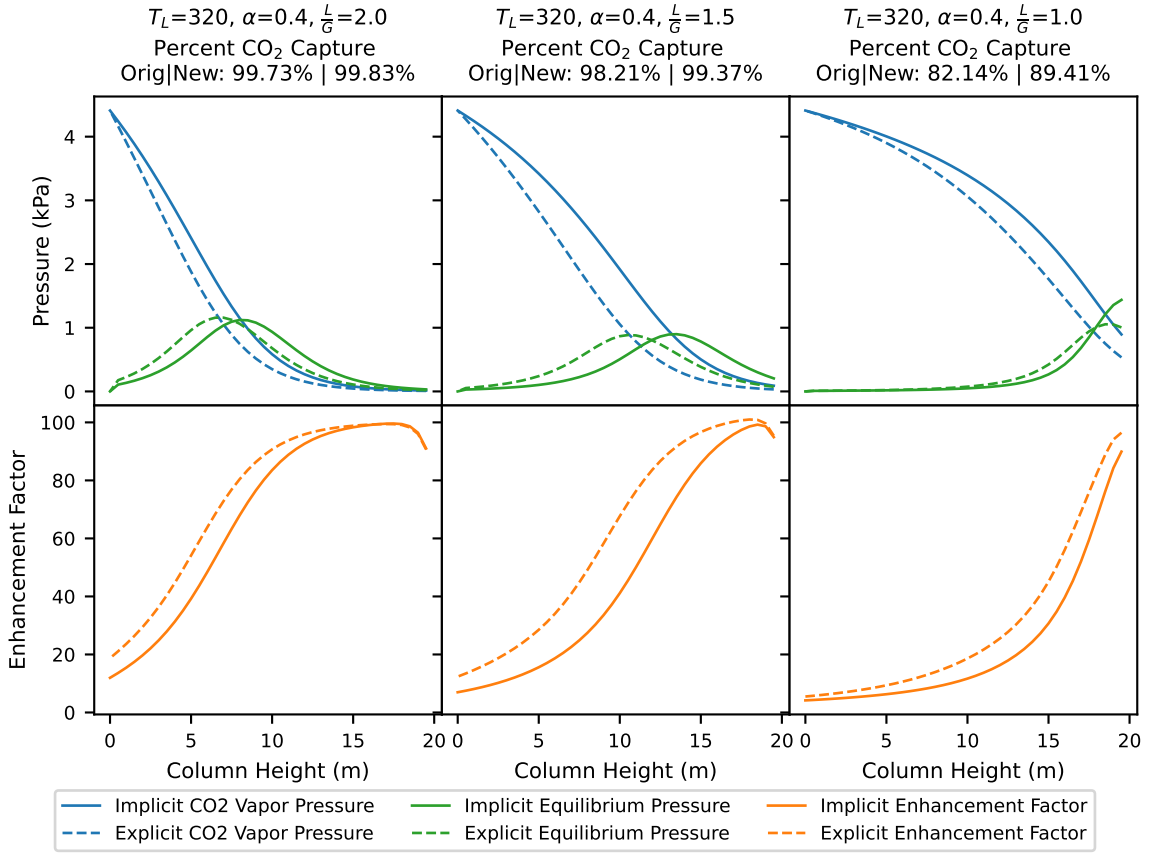


Figure 4: Comparison between the full (implicit) and equilibrium (explicit) enhancement factor models at varying levels of liquid-to-gas (L/G) molar flow ratio in the absorber column model. Note that the lean solvent inlet temperature was fixed at 320 K and the lean MEA loading was fixed at $\alpha = 0.4$. In all three scenarios, the percent CO_2 captured is overpredicted. However, at extremely high capture rates, the overprediction is only 0.1%, but at lower capture rates the overprediction is as high as 7%.

At higher capture percentages, the equilibrium enhancement factor model can predict a capture percentage similar to that of the original model. However, the enhancement factor given by the equilibrium model is consistently greater than that of the original model. The CO_2 vapor pressures and the equilibrium pressures also converge at high capture percentages, but the equilibrium enhancement factor model gives lower values of vapor pressure than the original model. Note that these simulations were carried out with a fixed column design. Optimization should push the system towards pinch conditions in which this approximation performs better.

5. Conclusions

We have outlined model diagnostic methods using tools available in Pyomo and IDAES-IP. As illustrated through the example of the MEA flowsheet, these tools can greatly assist the user by helping to point out likely problems. However, they do not solve them for the user. Finding problems in a model with tens of thousands of constraints and variables is a major service, but user insight and modeling expertise are still needed to solve them. Nevertheless, these tools and techniques can help make EO modeling frameworks more accessible and facilitate the use of advanced optimization techniques in process design.

In addition, we have presented a modified enhancement factor model that is explicit in the variables on which it depends. Although EO modeling frameworks like IDAES can handle implicit models, having an approximate explicit model is better for initializing the column model. Furthermore, the particular implicit model presented in [25] becomes degenerate as the system approaches equilibrium because it contains the indeterminate form $0/0$, so replacing it with an explicit model makes it much more numerically robust. A comparison between the implicit and explicit models found that, in the case of an absorption column, the enhancement factor is overestimated in the explicit model, which can lead to overprediction of the total amount of CO_2 captured. Nevertheless, it is hoped that this approximation is useful in situations where numerical robustness is key.

Acknowledgments

This work was conducted as part of the U.S. Department of Energy's Institute for the Design of Advanced Energy Systems (IDAES) supported by the Office of Fossil Energy and Carbon Management through the Simulation-based Engineering/Crosscutting Research and Carbon Capture Programs. It was conducted under the Strategic Analysis contract, number 23CFE000075. We would like to thank Andrew Lee, Miguel Zamarripa, Grigorios Panagakos, and Chrysanthos Gounaris for their helpful comments while revising this work.

Conflict of interest declaration

This project was funded by the Department of Energy, National Energy Technology Laboratory an agency of the United States Government, through a support contract. Neither the United States Government nor any agency thereof, nor any of its employees, nor the support contractor, 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.

References

[1] A. Lee, J. H. Ghouse, J. C. Eslick, C. D. Laird, J. D. Siirola, M. A. Zamarripa, D. Gunter, J. H. Shinn, A. W. Dowling, D. Bhattacharyya, L. T. Biegler, A. P. Burgard, D. C. Miller, The IDAES process modeling framework and model library—flexibility for process simulation and optimization, Journal of Advanced Manufacturing and Processing 3 (2021) 1–30.

[2] W. E. Hart, J.-P. Watson, D. L. Woodruff, Pyomo: modeling and solving mathematical programs in Python, Mathematical Programming Computation 3 (2011) 219–260.

[3] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, D. L. Woodruff, Pyomo-optimization modeling in python, volume 67, Springer, 2021.

[4] A. Lee, R. B. Parker, S. Poon, D. Gunter, A. W. Dowling, B. Nicholson, Model diagnostics for equation-oriented models: roadblocks and the path forward, Breckenridge, Colorado, USA, 2024, pp. 966–974. URL: https://PSEcommunity.org/LAPSE:2024.1632. doi:10.69997/sct.147875.

[5] A. L. Dulmage, N. S. Mendelsohn, Coverings of bipartite graphs, Canadian Journal of Mathematics 10 (1958) 517–534.

[6] P. Bunus, P. Fritzson, Automated static analysis of equation-based components, Simulation 80 (2004) 321–345.

[7] R. d. P. Soares, A. R. Secchi, Structural analysis for static and dynamic models, Mathematical and Computer Modelling 55 (2012) 1051–1067.

[8] H. Olsson, M. Otter, S. E. Mattsson, H. Elmqvist, Balanced models in Modelica 3.0 for increased model quality (2008).

[9] D. Broman, K. Nyström, P. Fritzson, Determining over-and under-constrained systems of equations using structural constraint delta, in: Proceedings of the 5th international conference on Generative programming and component engineering, 2006, pp. 151–160.

[10] S. Bublitz, E. Esche, G. Tolksdorf, V. Mehrmann, J.-U. Repke, Analysis and decomposition for improved convergence of nonlinear process models in chemical engineering, Chemie Ingenieur Technik 89 (2017) 1503–1514.

[11] C. Wang, L. Wan, T. Xiong, Y. Xie, S. Wang, J. Ding, L. Chen, Hierarchical structural analysis method for complex equation-oriented models, Mathematics 9 (2021) 1–23.

[12] R. B. Parker, B. L. Nicholson, J. D. Siirola, L. T. Biegler, Applications of the Dulmage–Mendelsohn decomposition for debugging nonlinear optimization problems, Computers & Chemical Engineering 178 (2023) 108383.

[13] A. W. Dowling, L. T. Biegler, Degeneracy Hunter: an algorithm for determining irreducible sets of degenerate constraints in mathematical programs, volume 37, Elsevier, 2015. URL: http://dx.doi.org/10.1016/B978-0-444-63578-5.50130-4. doi:10.1016/B978-0-444-63578-5.50130-4, publication Title: Computer Aided Chemical Engineering Issue: June ISSN: 15707946.

[14] E. Osborne, On pre-conditioning of matrices, Journal of the ACM (JACM) 7 (1960) 338–345.

[15] A. Van der Sluis, Condition numbers and equilibration of matrices, Numerische Mathematik 14 (1969) 14–23.

[16] A. Van der Sluis, Condition, equilibration and pivoting in linear algebraic systems, Numerische Mathematik 15 (1970) 74–86.

[17] R. D. Braatz, M. Morari, Minimizing the euclidean condition number, Siam Journal on Control and Optimization 32 (1994) 1763–1768.

[18] P. Akula, J. Eslick, D. Bhattacharyya, D. C. Miller, Model development, validation, and optimization of an MEA-based post-combustion CO_2 capture Process under part-load and variable capture operations, Industrial & Engineering Chemistry Research 60 (2021) 5176–5193.

[19] J. A. F. Sherman, N. M. Isenberg, J. D. Siirola, C. E. Gounaris, Recent advances of PyROS: A Pyomo solver for nonconvex two-stage robust optimization in process systems engineering, Breckenridge, Colorado, USA, 2024, pp. 201–207. URL: https://PSEcommunity.org/LAPSE:2024.1528. doi:10.69997/sct.142058.

[20] R. S. Kamath, L. T. Biegler, I. E. Grossmann, An equation-oriented approach for handling thermodynamics based on cubic equation of state in process optimization, Computers & Chemical Engineering 34 (2010) 2085–2096.

[21] D. M. Gay, Hooking your solver to AMPL, Technical Report, Bell Laboratories, 1997.

[22] J. S. Rodriguez, R. Parker, C. D. Laird, B. Nicholson, J. D. Siirola, M. Bynum, Scalable parallel nonlinear optimization with PyNumero and Parapint, Technical Report, Sandia National Laboratories, 2021.

[23] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming 106 (2006) 25–57.

[24] G. T. Rochelle, Air pollution impacts of amine scrubbing for CO2 capture, Carbon Capture Science & Technology 11 (2024) 100192.

[25] J. Gaspar, P. L. Fosbøl, A general enhancement factor model for absorption and desorption systems: A CO2 capture case-study, Chemical Engineering Science 138 (2015) 203–215. Publisher: Elsevier Ltd.

Nomenclature

Section 2	
$\mathbf{f}(\cdot)$	Generic function
$\mathbf{x}$	Generic unknown
$\boldsymbol{J}(\cdot)$	Jacobian Matrix of $\mathbf{f}(\cdot)$
$\Delta\mathbf{x}$	Newton step in $\mathbf{x}$
$\|\cdot\|_2$	Vector/Matrix 2-norm
ε	Convergence tolerance
$\kappa_2(\cdot)$	2-norm condition number
$\delta\mathbf{f}$	Perturbation in $\mathbf{f}(\cdot)$
$\delta\mathbf{x}$	Perturbation in $\mathbf{x}$
$\boldsymbol{U}$	Matrix of left singular vectors
$\boldsymbol{V}$	Matrix of right singular vectors
$\boldsymbol{\Sigma}$	Diagonal matrix of singular values
σ_k	kth singular value
y	Mole fraction
H	Enthalpy
Q	Heat duty
r_k	kth largest row of $\boldsymbol{J}(\cdot)$
Section 3	
h_V	Vapor heat transfer coefficient
h'_V	Corrected vapor heat transfer coefficient
a_e	Heat transfer area per unit column volume
Ack	Ackermann factor
$\underline{C}_{p,k,V}$	Molar heat capacity of component k vapor
$N_{k,V}$	Molar flux of component k from vapor to liquid
E	Enhancement factor
E^*_∞	Instantaneous enhancement factor
Ha	Hatta number
Υ^i_{MEA}	Dimensionless concentration of MEA at interface
$\Upsilon^b_{\text{CO}_2}$	Dimensionless concentration of CO_2 in bulk liquid
$\Upsilon^*_{\text{CO}_2}$	Dimensionless equilibrium concentration of CO_2 at interface
S_{CO_2}	Singular ratio of CO_2 concentrations
$log\ E$	Variable representing the logarithm of the enhancement factor
$log\ Ha$	Variable representing the logarithm of the Hatta number
$log\ \Upsilon^i_{\text{MEA}}$	Variable representing the logarithm of Υ^i_{MEA}
$log\ \eta$	Variable representing the logarithm of $E^*_\infty - 1$
Section 4	
ν_k	Stoichiometric coefficient of species k
k	Rate constant
C^b_k	Molar concentration of species k in bulk liquid
m	Reaction order with respect to species A
n	Reaction order with respect to species B
Υ^b_A	Dimensionless concentration of species A in bulk liquid
Υ^*_A	Dimensionless equilibrium concentration of species A
Υ^i_k	Dimensionless concentration of species k at interface
k_L	Liquid mass transfer coefficient of species A
k_V	Vapor mass transfer coefficient of species A
D_k	Diffusion coefficient of species k
He	Henry's law constant in the bulk
$Q_m(x)$	Polynomial function
R_j	Weighted ratio of mass transfer rates in film
p	Component A partial pressure perturbation from equilibrium
$\hat{E}$	Enhancement factor at equilibrium (i.e., $p = 0$)