A Globally Convergent Method for Finding All Steady-State Solutions of Distillation Columns

Ali Baharev and Arnold Neumaier
Faculty of Mathematics, University of Vienna, Vienna, Austria

DOI 10.1002/aic.14305
Published online December 11, 2013 in Wiley Online Library (wileyonlinelibrary.com)

Significance

A globally convergent method is proposed that either returns all solutions to steady-state models of distillation columns or proves their infeasibility. Initial estimates are not required. The method requires a specific but fairly general block-sparsity pattern; in return, the computational efforts grow linearly with the number of stages in the column. The well-known stage-by-stage (and the sequential modular) approach also reduces the task of solving high-dimensional steady-state models to that of solving a sequence of low-dimensional ones. Unfortunately, these low-dimensional systems are extremely sensitive to the initial estimates, so that solving them can be notoriously difficult or even impossible. The proposed algorithm overcomes these numerical difficulties by a new reparameterization technique. The successful solution of a numerically challenging reactive distillation column with seven steady-states shows the robustness of the method. No published software known to the authors could compute all solutions to this difficult model without expert tuning. © 2013 American Institute of Chemical Engineers *AIChE J*, 60: 410–414, 2014

Keywords: distillation, mathematical modeling, numerical solutions, simulation, process

Structure of the Problems Considered

The proposed method assumes that the Jacobian can be permuted to lower block Hessenberg form with many square, structurally nonsingular, superdiagonal blocks, see Figure 1. The goal is to solve the bound-constrained nonlinear system

$$F(x)=0, \quad \underline{x} \le x \le \overline{x}\,, \tag{1}$$

where $F:\mathbb{R}^n \mapsto \mathbb{R}^n$ is a continuously differentiable vector-valued function; $\underline{x}$ and $\overline{x}$ denote the vector of lower and upper bounds on the variables x, respectively. The variables are partitioned as

$$x=\begin{pmatrix} x_0 \\ \vdots \\ x_N \end{pmatrix} \tag{2}$$

into subvectors $x_i \in \mathbb{R}^{d_i}$ $(i=0\ldots N)$, so that $n=d_0+\cdots+d_N$. Similarly to the variables, F is partitioned as

$$F(x)=\begin{pmatrix} F_1(x) \\ \vdots \\ F_{N+1}(x) \end{pmatrix} \tag{3}$$

into subfunctions $F_i(x) \in \mathbb{R}^{d_i}$ $(i=1\ldots N+1)$; the trailing dimension must be $d_{N+1} := d_0$ as the system is square. The lower block Hessenberg structure says that only variables from subvectors $x_0,\ldots,x_i (i \le N)$ can appear in $F_i(x)$

$$F_i(x)=F_i(x_0,\ldots,x_i) \quad \text{for } i=1,\ldots,N. \tag{4}$$

Equations 2–4 describe the block sparsity pattern shown in Figure 1. The square blocks form the so-called upper envelope. In practice, the lower triangle is sparse.

At first glance, Figure 1 may look like the Dulmage–Mendelsohn decomposition[1,2] but this is not the case. The Dulmage–Mendelsohn decomposition is inconclusive when applied to the steady-state model of distillation columns as it returns the original system as a single large block.

The sparsity pattern shown in Figure 1 is equivalent to the bordered block triangular form (BBTF) as can be seen by moving the columns corresponding to x_0 to the last position. (Often, the lower triangular part is very sparse and the

Additional Supporting Information may be found in the online version of this article.

Correspondence concerning this article should be addressed to A. Baharev at ali.baharev@gmail.com.

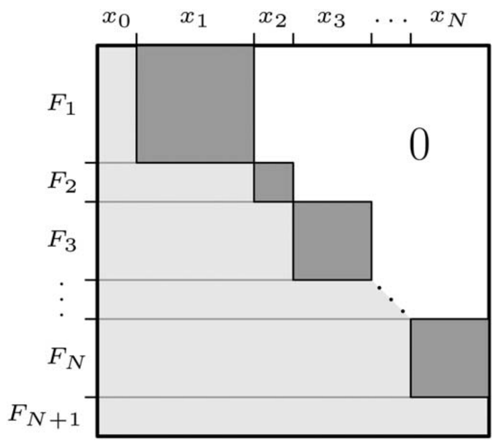


Figure 1. The desirable block sparsity pattern of the Jacobian, the lower block Hessenberg form, see text at Eqs. 1–4.

sparsity pattern can even be permuted to the recursive border block diagonal form as well.[3]) In the tearing approach,[4] the column border corresponds to the "tear" variables and the row border corresponds to the equations that are used to iterate on the tear variables. The tear variables x_0 play a special role; all variables are eliminated in terms of them. The proposed method is fundamentally different from the tearing methods: The proposed method does not have any tear variables, x_0 does not play a special role, and the lower block Hessenberg form suggests exactly this.

Hereafter, we assume that the equations and variables (corresponding to the rows and columns of the Jacobian) have been permuted in such a way that the Jacobian of the problem is in lower block Hessenberg form. Finding an optimal BBTF and hence, an optimal block decomposition is a difficult task in general but powerful heuristics are available, for example, MC33 from the Harwell Subroutine Library.[5]

The Proposed Algorithm

A sophisticated block elimination is performed along the upper envelope (square superdiagonal blocks), working from the left to the right, and from the top to the bottom. The following notation will be used throughout the next subsection: $x_k = x_k^{(i)}(s_i)$. Here, $x_k^{(i)}$ on the right hand side denotes a function: The lower index k tells that the value of the function determines variable x_k; the upper index i is the loop counter of the block elimination algorithm. Similarly, in the equation $s_{i-1} = s_{i-1}(s_i)$, the s_{i-1} on the left hand side is the value of the function s_{i-1} on the right hand side evaluated at s_i.

Formal statement

Input: A system of Eq. 1 in lower block Hessenberg form as shown in Figure 1; lower and upper bounds ($\underline{x}$ and $\bar{x}$, respectively) of the variables.

Output: All solutions to the input.

External dependency: An auxiliary algorithm to parameterize the solution set of an underdetermined system of equations. Here, the input of the auxiliary algorithm is (6) and its output is the parameterization (8) together with the transition map (9).

Initialization: For $i = 0$, apply linear parameterization

$$x_0 = x_0^{(0)}(s_0) = \underline{x}_0 + (\bar{x}_0 - \underline{x}_0)s_0, \qquad s_0 \in [0,1]^{d_0}. \tag{5}$$

Forward sweep: For $i = 1 \ldots N$, apply the auxiliary algorithm to the i-th block, a $d_i \times (d_0 + d_i)$ underdetermined system

$$F_i(x_0^{(i-1)}(s_{i-1}), \ldots, x_{i-1}^{(i-1)}(s_{i-1}), x_i) = 0, \tag{6}$$

considering, s_{i-1} and x_i as variables, ($\dim F_i = d_i$; $\dim s_{i-1} = d_0, \dim x_i = d_i$; d_0 superficial degrees of freedom). The auxiliary algorithm returns the variables s_{i-1} and x_i reparameterized in terms of new parameters s_i, in a well-conditioned, explicit fashion

$$s_{i-1} = s_{i-1}(s_i), \quad x_i = x_i^{(i)}(s_i), \quad s_i \in [0,1]^{d_0}. \tag{7}$$

From (7): $x_k = x_k^{(i-1)}(s_{i-1}) = x_k^{(i-1)}(s_{i-1}(s_i))(k = 0, \ldots, i-1)$ and $x_i = x_i^{(i)}(s_i)$; or simply

$$x_k = x_k^{(i)}(s_i) \quad \text{for} \quad k = 0, \ldots, i, \quad s_i \in [0,1]^{d_0}. \tag{8}$$

Save (8) together with the transition map

$$s_{i-1} = s_{i-1}(s_i) \tag{9}$$

for later use.

Solving the final system: For $i = N+1$, find all solutions s_N^* of the $d_0 \times d_0$ system

$$F_{N+1}(x_0^{(N)}(s_N), \ldots, x_N^{(N)}(s_N)) = 0. \tag{10}$$

Backward sweep: The solutions are obtained by simple substitutions

$$x_0^* := x_0^{(N)}(s_N^*), \ \ldots, \ x_N^* := x_N^{(N)}(s_N^*).$$

In practice, the sparsity of the Jacobian is exploited for efficiency reasons, and not all $x_k^{(i)}(s_i)$ are computed and saved in (8; see the Exploiting the sparsity section). In this case, recover recursively s_{i-1} from (9) for $i = N, \ldots, 1$, then x_{i-1} from (8).

Numerical stability

If the reparameterization steps (8) are skipped and all the variables are parameterized in terms of s_0, we get the final system

$$F_{N+1}(x_0(s_0), \ldots, \ x_N(x_{N-1}(\ldots x_0(s_0)\ldots))) = 0 \tag{11}$$

which is likely to be extremely ill-conditioned due to the deep nesting of reasonably conditioned transformations. This equation is compared to the final system (10) of the forward sweep in which the reparameterization steps are not skipped. For example, if one compares the term $x_N(x_{N-1}(\ldots x_0(s_0)\ldots))$ of (11) to the corresponding term $x_N^{(N)}(s_N)$ in (10), the difference becomes clear. The former is usually extremely ill-conditioned, whereas the latter is reasonably conditioned by construction, as a result of the reparameterization. The stage-by-stage method[6] and the sequential modular approach[4] proceed just in this fashion and do not involve any reparameterization. The resulting major flaw of these methods is well-known: they can be extremely sensitive to the initial estimates, up to a point where solving the final system of equations is notoriously difficult or even impossible.

Reparameterization is the key point to achieve numerical stability. The large ill-conditioned problem is split into smaller, reasonably conditioned ones. The idea of the proposed method was abstracted from the multiple shooting method[7] for boundary value problems, which mitigates in a similar way the conditioning problems of the unstable single shooting method. Implementing the reparameterization (8) is the most challenging step of the algorithm, see the Implementation details section.

Exploiting the sparsity

So far, a dense lower block Hessenberg form has been assumed: For a given i in (8), all parameterizations are computed and stored for $k=0,\dots,i$. It is easy to show that we would compute and store altogether $N(N+1)/2+N$ parameterizations in (8) during the forward sweep ($i=1,\dots,N$). That is, the work is proportional to N^2; formally, the work is $\Theta(N^2)$. Luckily, the Jacobian of the distillation columns is not only a lower block Hessenberg structure but also a (bordered) block band matrix and the work can be made proportional to the number of stages. Formally, the work can be reduced to O(N).

In case of distillation columns, the following recursive formula is proposed to determine which variables have to be reparameterized

$$\begin{aligned} &z_1=x_0,\\ &z_i=\text{ the subvector of } \begin{pmatrix} z_{i\text{-}1}\\ x_{i\text{-}1}\end{pmatrix} \text{ consisting of all variables}\\ &\qquad \text{appearing in some } F_j \ \ (j\geq i) \quad (i=2\dots N+1). \end{aligned} \tag{12}$$

Basically, Eq. 12 ensures that we pass on every variable that will be needed in later blocks, even if they do not appear in some of the intermediate blocks. Relations (12) tell which variables have to be reparameterized in (8); apart from this, no other changes are necessary to exploit the sparsity, the algorithm is executed as it is presented in the Formal statement subsection.

EXAMPLE. *The following system of equations is being solved*

$$\begin{aligned} F_1(x_0,\quad x_1 \qquad\qquad &)=0\\ F_2(\qquad x_1,\quad x_2 \qquad &)=0\\ F_3(\qquad x_1, \qquad\quad x_3&)=0\\ F_4(x_0, \qquad\qquad\quad x_3&)=0. \end{aligned} \tag{13}$$

The block elimination is performed with block size one, that is, single variables are eliminated in the following order: x_1, x_2, x_3 (upper envelope). The corresponding z_i are

$$z_1=x_0,\quad z_2=\begin{pmatrix}x_0\\x_1\end{pmatrix},\quad z_3=\begin{pmatrix}x_0\\x_1\end{pmatrix},\quad z_4=\begin{pmatrix}x_0\\x_3\end{pmatrix}. \tag{14}$$

Note that, x_0 appears in z_2 and *z_3 even though the corresponding equations do not depend on x_0: We have to pass on x_0 because it is needed later, when solving $F_4(x_0,x_3)=0$.*

For distillation columns, formula (12) does not result in any unnecessary reparameterizations, hence, it is optimal. However, if, for example, the last equation involves all variables, (12) does not reduce the computational work, and some other formula would be needed.

Implementation details

The C++ implementation is available in Supporting Information. Reparameterization at (8) is the most challenging part of the implementation. The solution set of the underdetermined system of Eq. 6 defines a manifold ("hyper-surface") of dimension d_0. Parameterizing multidimensional manifolds can be done[8] but is significantly more complex. However, the case $d_0=1$ is easier to handle, as a one-dimensional (1-D) manifold is just a union of curves and the arc length is the natural parameter. Therefore, the current implementation works only for 1-D manifolds, parameterized by arc length. A 1-D manifold consists of infinitely many points but it has to be approximated with finitely many parameters in practice. That is, the manifold has to be discretized. A simple piecewise linear approximation to 1-D manifolds is implemented: An ordered set of points and linear interpolation between the neighboring points is used. Our current research focuses on the general higher-dimensional case.

The Computed Reactive Distillation Column

The rigorous steady-state model is taken from Ciric and Miao,[9] corresponding to the cost-optimal column of Ciric and Gu.[10] In the column, ethylene glycol is produced from ethylene oxide and water

$$\begin{aligned} C_2H_4O+H_2O &\rightarrow C_2H_6O_2 \qquad \text{(ethylene glycol)}\\ C_2H_4O+C_2H_6O_2 &\rightarrow C_4H_{10}O_3 \quad \text{(diethylene glycol)} \end{aligned} \tag{15}$$

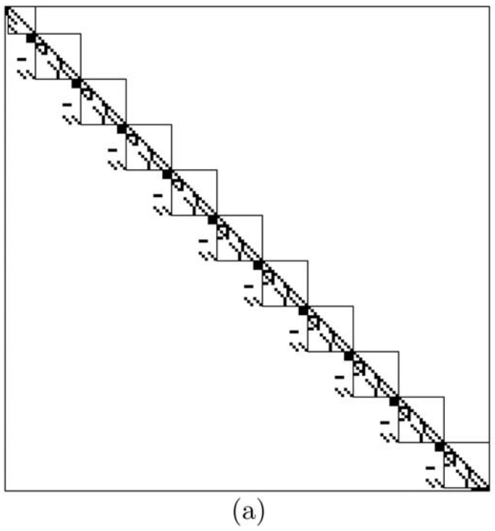
Ethylene glycol is the product and diethylene glycol is an unwanted byproduct. The column can have multiple steady-states (three or even nine) depending on the hold-up volume on the reactive stages. The model equations and parameters, the elimination order, the permutation to lower block Hessenberg form, the AMPL[11] model of the example, and the C++ implementation are all available in Supporting Information. This suffices to reproduce all computations. Due to the strict space limitations, no further details are given here.

The column of Ciric and Miao[9] lacks its distillate stream. Based on the conservation laws and the necessary conditions of equilibrium, one can show that d_0 equals the number R of independent reactions for a column lacking its distillate stream. (The steady-state simulation of such a column only makes sense in the presence of reactions.) Because $R = 2$ in (15) but the current reparameterization step only works for $d_0=1$, as discussed in the Implementation details subsection, a simplifying assumption had to be made. In the 7.3 Multiple Reactions subsection,[9] it is shown that the high multiplicities remain even if the side reaction is neglected and equimolar feed streams are specified. Therefore, the formation of the diethylene glycol is ignored in our simplified model. However, we emphasize that the algorithm is not limited to the $d_0=1$ case, only the current implementation of the reparameterization step is.

Results and Discussion

Block decomposition

The algorithm requires that its input system of equations has been permuted in such a way that the Jacobian is in



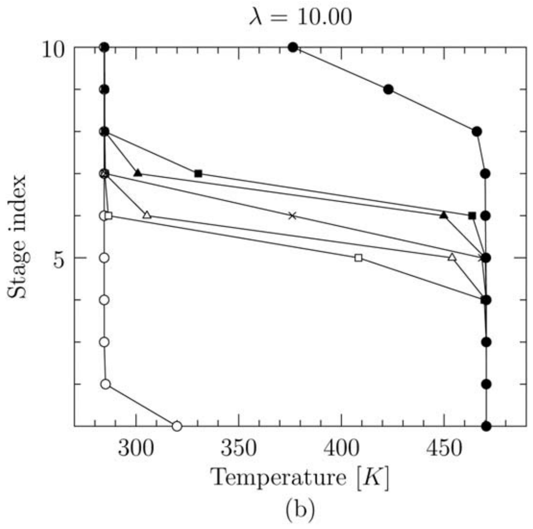


Figure 2. (a) Sparsity pattern of the reactive distillation column of Ciric and Miao,[9] high-resolution image in the supplement.

(b) Seven temperature profiles corresponding to the seven steady-state solutions of the same column, λ is proportional to the hold-up volume on the reactive stages, see in Supporting Information.

lower block Hessenberg form, see Figure 1. The permutation, determining the elimination order, was obtained by inspection for the column referred to in The Computed Reactive Distillation Column section. (As aforementioned, powerful methods are available to generate good decompositions automatically, e.g., MC33.[5]) It is clear from Figure 2a that the Jacobian has the desired lower block Hessenberg structure after permutation; the algorithm can be applied.

Reactive distillation

The steady-state model of the reactive distillation column of Ciric and Miao[9] is a real numerical challenge. IPOPT[12] is well-known as a state-of-the-art solver for optimization. An apparently little known feature of IPOPT is the restoration phase for solving systems of nonlinear equations. The restoration phase is remarkably robust:[13] IPOPT often converges even if tailored-made methods for distillation columns fail. Significant efforts were made to compute this column with IPOPT and with various problem reformulations.[14] These attempts failed; our study showed that the Jacobian is singular to working precision. The failure is not the defect of IPOPT; the equation-oriented approaches[14] cannot succeed under these circumstances. Indeed, a small change in the temperature or the composition of the input streams of a stage can cause a 10^6 time larger change in the output streams of the same stage. In other words, even the individual stages are not well-conditioned.

The proposed algorithm reduces the original $n \times n$ problem (1) to a much smaller $d_0 \times d_0$ problem given by (10), and the final system (10) is reasonably conditioned by construction. Because $d_0 \ll n$, the savings in computational efforts are significant.

As discussed in the Implementation details subsection, the solution set of (6) is discretized as a piecewise linear manifold. We measured the computational efforts in the number of pieces m of the piecewise linear manifold. This is a good measure of work: using m pieces ($m+1$ breakpoints) is roughly the same work as $m+1$ function evaluations of $F(x)$ in the original system (1). Additionally, m does not depend on the software and hardware environment, thus, it enables fair comparisons.

The steady-state model of the column of Ciric and Miao[9] is a 70×70 system ($n = 70$); the proposed method reduces this system to a well-conditioned, univariate equation, corresponding to the final system (10) with $d_0 = 1$. Despite the unusually sensitive stage computations, $m = 10,000$ linear pieces guarantee a sufficient piecewise linear approximation to perform the reparameterization at (6)–(8). The proposed method is numerically stable, consistent results are computed under small perturbations. This would not be the case if the method were numerically unstable. Temperature profiles, corresponding to the seven steady states, are given in Figure 2b. Our results are in good qualitative agreement with Ciric and Miao[9] but there are quantitative differences. Honest efforts have been made to resolve these. Unfortunately, the reason remains unknown; perhaps some of the model parameters contain a misprint.

Further, numerical experiments (not detailed here) with other columns show that less than 50 breakpoints ($m < 50$) are sufficient for columns with $d_0 = 1$ even with highly nonideal mixtures as long as the individual stages are well-conditioned. In other words, the computational costs of the method is typically less than 50 function evaluations of $F(x)$ in (1) if the stages are not ill-conditioned ($d_0 = 1$). The reason why $m = 10,000$, linear pieces were required if the stage computations are unusually sensitive is due to details of the current reparameterization algorithm. Different ways have been identified how these extremely sensitive cases can be handled more efficiently but these ideas are not yet implemented.

Linear time complexity

All current general-purpose methods that are guaranteed to find all solutions have exponential worst case time complexity in the number of variables. Unfortunately, in practice, the actual execution time (per solution) also tends to grow exponentially with the problems size as well. The proposed method requires a specific but fairly general block sparsity pattern. In return, the computational efforts (per solution) of the proposed method grow linearly with the number of stages in case of distillation columns, that is, the time complexity of the method is linear in the number of stages.

The first step in proving linear time complexity is to show that the computational efforts to eliminate one block can be safely bounded with a constant that does not grow with N. A stage in the distillation column corresponds to a block in (6). The time needed to eliminate a block, that is, to solve the model equations of a single stage can be safely bounded by a constant that is independent of the number of stages. Solving the model equations of a stage is basically solving a bubble point calculation (coupled with the reaction rate equations in case of a reactive stage) and this work can be safely bounded by a constant, that is, independent of the number of stages. The number of parameterizations at (8) does not grow with N either: the model equations of a stage only contain variables that are associated with its immediate neighbors, and the number of these variables does not depend on the total number of stages.

It is clear from the Formal statement subsection that the time execution time is proportional to the number of stages: both the forward and the backward sweep take N steps and the time to execute one step is bounded by a constant independent of N. Thus, the proposed method has linear time complexity in N (per solution) for distillation columns.

In the worst case, the computational work grows exponentially with d_0 as the reparameterization step involves discretization of a manifold of dimension d_0. Based on the conservation laws and the necessary conditions of equilibrium, one can show that for well-determined steady-state models of columns with two product streams (distillate and bottoms) and with arbitrarily many but fixed (known) feed streams $d_0=C-1$, where C is the number of components. The number of reactions, if any, does not influence d_0. The execution time is expected to remain acceptable even for multicomponent mixtures with this type of columns, assuming that C is not too large (e.g., $C \leq 6$).

General applicability of the method

The origin of the problem is irrelevant from the point of view of the algorithm. The only relevant aspect is that the Jacobian can be permuted to lower block Hessenberg form. It is irrelevant whether the problem is the steady-state model of a reactive distillation column or the model of an electric circuit and so forth; only the block sparsity pattern matters.

Acknowledgments

The research was funded by the Austrian Science Fund (FWF): P23554. The authors wish to thank Máté Gábor (ÖMV), Endre Rév (BME), and Michael Sielemann (DLR) for their valuable suggestions for clarification of the ideas in this paper.

Literature Cited

1. Dulmage AL, Mendelsohn NS. Two algorithms for bipartite graphs. *J Soc Ind Appl Math.* 1963;11:183–194.
2. Pothen A, Fan CJ. Computing the block triangular form of a sparse matrix. *ACM Trans Math Softw.* 1990;16: 303–324.
3. Abbott KA, Allan BA, Westerberg AW. Global preordering for Newton equations using model hierarchy. *AIChE J.* 1997;43:3193–3204.
4. Biegler LT, Grossmann IE, Westerberg AW. *Systematic Methods of Chemical Process Design*. Upper Saddle River, NJ: Prentice Hall PTR, 1997:243–294.
5. HSL. *A Collection of Fortran Codes for Large Scale Scientific Computation*. 2013. Available at: http://www.hsl.rl.ac.uk. Accessed on November 29, 2013.
6. Lewis WK, Matheson GL. Studies in distillation. *Ind Eng Chem.* 1932;24:494–498.
7. Betts JT. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia, PA: SIAM, 2010:95–97.
8. Henderson ME. Higher-dimensional continuation. In: Krauskopf B, Osinga HM, Galán-Vioque J, editors. *Numerical Continuation Methods for Dynamical Systems*. Dordrecht, The Netherlands: Springer, 2007:77–115.
9. Ciric AR, Miao P. Steady state multiplicities in an ethylene glycol reactive distillation column. *Ind Eng Chem Res.* 1994;33:2738–2748.
10. Ciric AR, Gu D. Synthesis of nonequilibrium reactive distillation processes by MINLP optimization. *AIChE J.* 1994;40:1479–1487.
11. Fourer R, Gay DM, Kernighan BW. *AMPL: A Modeling Language for Mathematical Programming*. Belmont, CA: Thomson Brooks/Cole, 2003.
12. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program.* 2006;106:25–57.
13. Baharev A, Kolev L, Rév E. Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation. *AIChE J.* 2011;57:1485–1495.
14. Biegler LT. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia, PA: SIAM, 2010:181–209.

Manuscript received Feb. 19, 2013, and revision received Nov. 11, 2013.