Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation

Ali Baharev,[1,3] Lubomir Kolev,[2] Endre Rév[1]

(1) Budapest University of Technology and Economics
Department of Chemical and Environmental Process Engineering
1521 Budapest, Pf. 91, Hungary

(2) Technical University of Sofia
Department of Theoretical Electrical Engineering
Sofia, 8 Kliment Ohridski Blvd., Block 12, Floor 4 and 5.

(3) Author to whom all correspondence should be addressed. E-mail: ali.baharev@gmail.com

This is a preprint of an article accepted for publication in
AIChE Journal

http://www.interscience.wiley.com

Abstract

Multiple steady states are typically discovered by tracing a solution path, including turning points. A new technique is presented here that does not follow this approach. The original problem is solved directly, without tracing a solution path. The proposed branch and prune algorithm is guaranteed to find all solutions *automatically*. Core components of the framework are affine arithmetic, constraint propagation and linear programming. The C++ implementation is available as an open source solver and has an interface to the AMPL® modeling environment.

In certain difficult cases, only continuation methods have been reported to find the unstable solution automatically. The proposed method seems to be the first published alternative method in those cases. Although this paper focuses mainly on distillation, the presented framework is fairly general and applicable to a wide variety of problems. Further computational results are given to demonstrate this.

Keywords: interval methods, output multiplicities, multiple solutions, bifurcation, MSS

1. Introduction

Multiple steady states (MSS) in distillation columns have been studied extensively during the past 50 years. The reader is referred to the work of Güttinger[1] for an excellent review. Identifying multiple steady states is critical[2] to proper design, simulation, control and operation of distillation columns. Professional simulators can return one solution at a time without indicating possible existence of other solutions. As a consequence, finding multiple solutions or proving uniqueness of a solution requires further efforts. Surprisingly enough, the computational tools to find multiple solutions *automatically* have received little attention in the literature. Continuation methods[3-5] (pp. 304–310 in 5) seem to be the only successfully applied methods in certain difficult cases.[6-11]

Conventional inside-out[12] and simultaneous correction[13] procedures were reported to miss[14-15] the unstable steady state solution of a homogeneous ternary azeotropic distillation problem (all input variables were specified; output multiplicity). All steady state branches were computed with an appropriate continuation method.[14-17] A solution path was traced with respect to a user selected parameter (bifurcation parameter). The initial estimates were carefully chosen,[18-19] special attention was paid to the turning points and branch switching.

A fundamentally different approach, a branch-and-prune algorithm is presented here. The proposed method is guaranteed to find all solutions automatically. The method works without initial estimates and without any assistance from the user. Exact computations are assumed, *i.e.* rounding errors are neglected. The algorithm requires that each variable has reasonable (not big M) lower and upper bounds. This also ensures that the search procedure is restricted to the domain of interest. Physically meaningless solutions can be cut out *a-priori* by imposing proper variable bounds. To the best of the authors' knowledge, the proposed algorithm is the first alternative to continuation methods that is capable to find all solutions to the homogeneous azeotropic distillation problem[14-16] discussed above. The applicability of the proposed method is not limited to distillation. Further computational results are given to demonstrate the generality of the method.

Several general purpose solvers exist that are guaranteed to find all solutions automatically, assuming exact computations (so-called *complete* solvers). The reader is referred to the survey of Neumaier[20] for an exhaustive overview. The paper of Belotti *et al.*[21] gives a thorough review of the latest results achieved since the paper of Neumaier. Impressive results[22-25] have been achieved with complete methods for a vast number of chemical engineering problems. The characteristic components of the method proposed here are (*i*) the linearization technique, (*ii*) the bound tightening method based on linear programming and probing, (*iii*) the local search heuristic and (*iv*) the AMPL[26] interface.

Algorithms guaranteed to find all solutions tend to require drastically increasing computational efforts as the number of variables increases. The problem size is not necessarily a good measure of the intrinsic difficulty of a problem.[27-28] Still, the major source of difficulties in solving the MESH equations of the homoazeotropic distillation column seems to be the size of the problem and the wide initial intervals of the variables. Solving this *large-scale* system of nonlinear equations is considered the main achievement of the paper. There can be several other sources of numerical difficulties when solving systems of nonlinear equations. The problem may be ill-conditioned, badly scaled, or have singular / nearly singular solutions, nonisolated solution set, *etc*. The latter issues are not addressed in the present study.

2. Methodology

Given

$$f(x) = 0, \qquad \text{where } f\text{: } R^n \rightarrow R^n,\ x \in \boldsymbol{X} \tag{1}$$

the goal is to bound all solutions of (1) or prove their absence, using a first order interval method. Interval data types are typeset in boldface, for example $x_j \in \boldsymbol{x}_j = [\underline{x}_j, \overline{x}_j]$, where $\underline{x}_j$, $\overline{x}_j$ are the lower and upper bound, resp., of variable x_j. The interval vector of the domain of the variables is called box.
The algorithm is presented only at conceptual level. Reference to the deep theoretical background is indicated at each component. For further, mainly implementation details, the reader is referred to the source code.[29] The aim of this section is to point out the components that make the method effective.

2.1. Linearization

One of the characteristic components of the proposed algorithm is the linearization technique. Numerical evidence published in the literature[30-42] seem to indicate superiority of the linear enclosure

$$f(x) \in Ax + \boldsymbol{b}, \quad x \in \boldsymbol{X}, \qquad A \text{ is a } \textit{real} \text{ matrix}, \tag{2}$$

compared to the traditional one

$$f(x) \in \boldsymbol{A}(x - z) + f(z),\ x \in \boldsymbol{X}, \qquad \boldsymbol{A} \text{ is an } \textit{interval} \text{ matrix}, \tag{3}$$

such as the interval Newton methods[43] (Chapter 7, pp. 123–157). The linear enclosure (2) is automatically obtained with the mixed affine arithmetic and interval arithmetic model.[44]

2.2. Pruning based on linear programming

Another characteristic component of the proposed method is the *efficient implementation* of the pruning procedure based on linear programming (LP pruning). The original non-linear function $f(x)$ in (1) is enclosed by the linear enclosure (2). Tight bounds on the solution set of (2), hence on the solution set of (1), is computed by solving the linear programming subproblems for all j-s:

$$\begin{aligned} &\min/\max\ x_j \\ &\text{subject to} \\ &\quad b_L \le Ax \le b_U \\ &\quad x_L \le x \le x_U. \end{aligned} \tag{4}$$

The constraints in (4) are directly obtained by linearization, as discussed in the previous subsection, and the rest are the bounds on the variables. The constraints remain unchanged during the pruning procedure, only the objective function is changed iteratively. The sequence in which the subproblems in (4) are processed has a great impact on the required computational efforts.[45] Achterberg's heuristic[45] for subproblem selection is followed. Other LP pruning methods have been proposed by several authors.[21,33,36,46-54]

2.3. Pruning based on directed acyclic graph representation

The system of equations can be represented as directed acyclic graphs (DAG). Each variable, each constant, each operation and each equation is represented by a node of the DAG. Edges represent the flow of the computation. The output of the AMPL modeling environment is basically the DAG representation of the equations.
The reader is referred to *Section 5* of the paper of Schichl and Neumaier,[55] and to the work of Kearfott[56] and Beelitz *et al.*[57] for proper mathematical and algorithmic underpinnings of this domain reduction technique. The basic idea is demonstrated here by an example.[58] The goal is to derive bounds on the solution set of the system of equations

$$\begin{aligned} x^2 + y^2 - 1 &= 0 \\ x^2 - y &= 0 \end{aligned} \tag{5}$$

where $x \in [-\infty, \infty]$, $y \in [-\infty, \infty]$ initially. A new variable and equation is introduced for each elementary operation as follows:

$$v_1 = x^2 \tag{6a}$$
$$v_2 = y^2 \tag{6b}$$
$$v_1 + v_2 = 1 \tag{6c}$$
$$y = x^2 \tag{6d}$$

where $x \in [-\infty, \infty]$, $y \in [-\infty, \infty]$, $v_1 \in [-\infty, \infty]$, $v_2 \in [-\infty, \infty]$ initially. As $x^2 \in [\,0, \infty]$, (6a) implies that $v_1 \in [\,0, \infty]$. Similarly (6b) implies that $v_2 \in [\,0, \infty]$. As just deduced $v_1 \ge 0$, therefore (6c) yields that $v_2 = 1 - v_1 \le 1$, *i.e.* $v_2 \in [\,0, 1]$. Similarly, $v_1 \in [\,0, 1]$. Because $x^2 \in [\,0, \infty]$, it follows from (6d) that $y \in [\,0, \infty]$. From (6b): $y \in [-\sqrt{v_2}\,, \sqrt{v_2}]$, so $y \in [\,0, \infty] \cap [-1, 1] = [\,0, 1]$. From (6a): $x \in [-\sqrt{v_1}\,, \sqrt{v_1}] = [-1, 1]$. The new domain of the original variables are $x \in [-1, 1]$ and $y \in [\,0, 1]$, still enclosing *all* the solutions in the initial

domain. Note that this example can be solved by replacing y in the first equation by x^2 to obtain a univariate quadratic equation in x^2. The steps performed by the solver have been presented here, without making this substitution.

2.4. Probing (optional component)

This component is obtained by incorporating the previous components. For a given index j, the domain of variable x_j is split into k equal-width parts ($k = 8$ in the current implementation). In this way k adjacent subboxes are formed from the original box. The above DAG and LP based pruning steps are applied to each subbox individually. Then, the (hopefully) contracted subboxes are all merged into one box again by computing their hull. These splitting-pruning-merging steps are repeated for all variables ($j = 1 \dots n$) in the authors' *brute-force* implementation. Note that the current probing method is eligible for parallel processing. The idea of the algorithm is inspired by the presolving technique probing widely used in mixed integer programming.[59] One can easily derive alterations to the above naïve algorithm by translating more elaborate probing techniques from mixed integer programming to interval methods.

2.5. Local search heuristic (optional component)

A judiciously chosen practical method[5] (pp. 292–304), a local solver, is applied to minimize either the ℓ^1 or the ℓ^2 norm of $f(x)$ (constraint violation) in $f(x) = 0$, $x \in \boldsymbol{X}$. The local solver usually finds solutions in a far earlier stage of the iteration (*i.e.* in wide boxes) than the branch-and-prune procedure would find them. When a solution is found in a box by the local solver, the branch and prune procedure will still continue to search that box for other possible solutions. The only intent of the optional local search procedure is to deliver solutions earlier, it does not influence the branch and prune procedure.

The trivial benefit of getting solutions quickly is that they can be analyzed while the usually time consuming computations are still in progress. Even if the branch-and-prune procedure fails to deliver any result in acceptable time because the problem is too complicated, the local solver may still find solutions. In the latter case, the proposed method degrades to a 'smart' multistart method.

The local solver has to tolerate poor initial points because the midpoint of the actual box is chosen as initial point in each iteration. Bound constraints on the variables have to be handled by the local solver. This ensures that only the current box is searched, prevents the method from converging to solutions that are not of interest (*e.g.* physically meaningless). A robust scaling method is also crucial.

All of these requirements are satisfied by the restoration phase in IPOPT.[60] The general purpose solver IPOPT is remarkably robust. It is able to converge to the unstable steady state solution of homogeneous azeotropic distillation problem from a poor starting point (given in the corresponding AMPL model file `mss20`[29]). As discussed in the introduction, this solution is missed by inside-out and simultaneous correction procedures even when supplied with excellent initial guesses.

2.6. Splitting

The box is split into smaller parts if its width is above a certain threshold but it cannot be (sufficiently) reduced further in size. The pruning steps are repeated on each smaller part obtained by splitting. The choice of the variable(s) along which to split the box is a hard and open question in general. A reasonable rule is implemented in the solver: we split the widest

interval component of the box. This plain rule may not be robust enough in general. It was reported to fail[45] for extractive distillation problems. The simple, problem specific bisection rule proposed for extractive distillation[45] is also applicable to homogeneous azeotropic distillation (`mss20`). In case of `mss20`, the widest interval corresponding to the bulk composition is bisected exclusively as long as its width is above 0.05.
The current implementation uses a simple stopping criterion. The box is not split any further if the absolute or the relative width of the box drops below a pre-defined threshold. The box is printed instead, it may contain a solution or solutions. The threshold is currently set to 10^{-4}, it is sufficient for most engineering applications.

2.7. Algorithm to bound all solutions

Given the system of nonlinear equations $f(x) = 0$ and variable bounds $x \in \boldsymbol{X}$, the goal is to bound all solutions or prove their absence.

Step 0. Initialize the stack of boxes with the initial box $\boldsymbol{X}$.

Step 1. If the stack is empty then exit else pop the top-most box off the stack.

Step 2. Apply DAG based pruning as long as the box is sufficiently reduced in size but at most n times (n denotes the number of variables). If an empty interval is obtained then discard the box and go to step 1.

Step 3. Linearize f over the (hopefully contracted) box. If there is at least one component of the linear enclosure whose range does not contain zero then discard the box and go to step 1.

Step 4. Apply LP pruning. If the first LP problem is infeasible then discard the box and go to step 1.

Step 5. Repeat just step 2, then continue with step 6.

Step 6. Repeat steps 2−5 as long as the box is sufficiently reduced in size but at most 10 times.

Step 7. (*Optional*) Apply brute-force probing for each variable, use step 6 to prune the subboxes.

Step 8. (*Optional*) Perform local search in the current box. If a new solution is found then print it immediately.

Step 9. If the widest component of the contracted box is below a pre-defined threshold then print the box (it may contain a solution or solutions) else bisect it along the domain of the widest component and push the resulting two boxes to the stack. Go to step 1.

Note. It is difficult to establish a rule to decide if the box is ‘sufficiently’ reduced in size. Different parameter values were tried, and the first found set that worked acceptably with the most difficult benchmark `mss20`[29] was chosen and used for each benchmark. Even though these parameters can have significant influence on the computation time, no considerable efforts have been made to refine them, partly because a test run lasts for several hours. Another reason for not tuning these parameters is that developing an adaptive rule for

measuring the rate of progress seems to be superior to fixed parameter values. Such an adaptive rule can be based on the dimensionality of the problem, *e.g.* Hansen's proposal,[61] but other properties should also be considered. Fixed, hard-coded parameters ignore the properties of the problem at hand.

3. Results

In order to maintain brevity, only a short description of the problems is given here. The model equations and parameters are given in the corresponding AMPL model files.[29] The software and hardware environment is given in the *Appendix*. The computational efforts are summarized in subsection *3.5*.

3.1. Multiple steady states in homogeneous azeotropic distillation

Column profiles of a distillation column with specified methanol-methyl butyrate-toluene feed composition, reflux ratio and vapor flow rate are computed (benchmark name: `mss20`). Except the number of stages and the feed stage location, the model and its parameters correspond to the Auto model.[16] There are three steady state branches: two stable steady state branches with an unstable branch in between.[16,62]
Multiple steady states can be predicted by analyzing columns with infinite reflux and infinite length.[18,19,63] These predictions for such infinite columns have relevant implications for columns of finite length operated at finite reflux. The existence and stability of these steady states was experimentally verified in an industrial pilot column operated at finite reflux.[16,62]
As for the simulation, conventional inside-out[12] and simultaneous correction[13] procedures were reported to miss[14-15] the unstable steady state solution. In contrast, the general purpose solver IPOPT is able to converge to the unstable steady state solution from a poor starting point (given in `mss20`[29]). The proposed branch and prune method successfully finds all solutions. The computed results are presented in *Figure 1* and *2*, they are in excellent agreement with the previously published ones.[14-16]

3.2. Multiple steady states in ideal two-product distillation

Perhaps surprisingly, multiple steady states can arise in simple distillation columns with ideal vapor-liquid equilibrium.[64] The model equations are taken from the paper of Jacobsen and Skogestad (benchmark name: `Jacobsen96_3` and `Jacobsen59_2`). Specifications are: methanol-propanol feed composition, *mass* reflux flow rate and vapor molar flow rate of the boilup. Heat balances are included in the model.
Two types of multiplicities can be distinguished. The first type can occur when some flows are specified on a mass basis and are due to the nonlinear mass to molar flow transformation.[64] *Figure 3a* shows an example for this. There are three steady state branches: two stable steady state branches with an unstable branch in between.
The second type occurs when energy balances are included in the model. This type of multiplicity does not occur for the case of constant molar flows. The region of the different type of multiplicities can overlap, leading to two stable and two unstable solutions.[64] *Figure 3b* illustrates this phenomenon. The model equations have five distinct solutions in a certain range of the relux flow rate. One of the solutions is infeasible in practice because it would result in negative flow rates. Nevertheless, the proposed algorithm finds all the five solutions, the infeasible one is represented by a dashed line in *Figure 3b*.

3.3. Extractive distillation

MESH equations of columns hosting continuous extractive distillation of acetone and methanol with water as entrainer are solved (benchmark name: `extr22` and `extr30`). Multiplicity is not observed in case of this benchmark, it has a unique solution. Previously, the largest reported column had 22 theoretical stages.[45] A problem specific bisection rule was also needed earlier[45] in order to solve the problem in acceptable time. That column with 22 stages and a larger one having 30 theoretical stages are computed here. The problem specific bisection rule is not needed if probing is enabled, the models are solved without bisection in both cases. The column composition and temperature profiles are shown in *Figure 4*.

3.4. Further practical problems

The first problem[51] (having 30 variables) is usually called the Bratu problem in the literature and is part of the MINPACK-2 test problem collection[65] (`Bratu30`). The benchmark is derived from the steady-state heat balance equation applied to model thermal ignition in a vessel. The theoretical foundation of the model was given by Frank–Kamenetskii[66] (Chap. VI–VII). The main result on the existence and multiplicity of solutions shows that there is a critical value δ_c of the so-called Frank–Kamenetskii parameter δ such that the steady state model has one stable and one unstable solution if $\delta < \delta_c$, and there is no solution if $\delta > \delta_c$. The latter case, when there is no steady state solution, corresponds to the thermal runaway. The benchmark `Bratu30` has two solutions.
The second benchmark[67-69] is an economic modeling problem, having 16 solutions (`eco9`). This problem seems to be inherently difficult for branch-and-prune interval methods; this is why it is considered here.

3.5. Computational efforts for each problem

As shown in *Table 1*, the first solution is immediately found in the initial box by the local solver. The local solver finds many / all of the solutions before the branch-and-prune method itself would find the first solution. The computational overhead of the local search is negligible.
The *brute-force* probing procedure is computationally demanding. These efforts pay off in the significantly reduced number of splits. It is interesting to note that except `mss20` and `eco9`, the problems are solved by performing the least possible splits. It is impossible to find m solutions in less than $m-1$ splits, the solutions have to be separated by splits.
As shown in *Table 2*, the reduced / minimal number of splits does not necessarily mean reduced computation time. See for example `eco9`. The homogeneous azeotropic and the extractive distillation problems are not shown is *Table 2*. Probing and/or problem specific splitting rule is required to solve the latter problems.
These results are achieved with the same internal parameters (*e.g.* threshold for the rate of progress). Different parameters can give significantly better performance for low dimensional problems but those parameters may not necessarily work for high dimensional problems. See the note at subsection *2.7*.

4. Limitations of the current implementation

These limitations apply to the *current implementation only*, the proposed branch and prune algorithm is not involved.

Rounding errors are ignored during the computations. This can lead to erroneous bounds, which in turn may result in lost solutions, especially for ill-conditioned problems. All solutions are found for each of the benchmarks considered here (and for others computed but not presented here). Still, this issue must be fixed in the future. Rounding errors can be automatically handled in all components of the algorithm: (*i*) the interval arithmetic computations[70] (DAG based pruning step), (*ii*) the affine arithmetic computations[44] and (*iii*) the LP pruning step.[46,71,72] The entire implementation can be made rigorous, providing both mathematical and computational guarantees, by following the previous guidelines.

"Unbounded variables are perhaps the dominant reason for failure of current complete global optimization codes on problems with few variables."[20] The same holds for systems of equations. Aggressive use of probing and pruning methods during the presolve phase can ameliorate this, provided the problem inherently implies useful finite bounds on the variables. The current implementation requires reasonable (not big M) lower and upper bounds as a presolve phase is not implemented yet.

The output of the branch and prune method is a list of boxes. Each box of this list may contain a solution or solutions. Parasitic solutions are boxes that the pruning method failed to discard but they do not contain solutions. They are typically close to a true solution. Rigorous verification of existence and / or uniqueness of a solution in the boxes of the result list is not implemented yet. This flaw can be fixed by implementing a post-solve phase, similar to the `VerificationStep` in C-XSC.[70] Other, more efficient post-solve methods are also available.[73-77]

5. Conclusions and future work

To the best of the authors' knowledge, the proposed algorithm is the first alternative to continuation methods that is capable to find all solutions to the homogeneous azeotropic distillation problem[14-16] automatically. Solving the corresponding large-scale system of nonlinear equations is considered the main achievement of the paper. The proposed method is guaranteed to find all solutions without any assistance from the user. The method does not depend on any initial estimates. The algorithm handles variable bounds to ensure that only the domain of interest is considered during the search procedure. The proposed method is fairly general and applicable to a wide variety of problems.

The current C++ implementation should be considered as a *proof-of-concept* and not as an industrial strength solver. There are several ways to improve the implementation. For example the *brute-force* probing method is the bottleneck of the algorithm. The probing step is eligible for parallel processing. More effective probing can be derived by analogy with mixed integer probing techniques.

Incorporating consistency techniques[43,78-82] (Chap. 10 in Hansen, Walster[43]), *state-of-the-art* propagation methods[83] and box consistency techniques[84] into the solver is expected to significantly increase both efficiency and robustness. The cutting edge consistency techniques can exploit common subexpressions,[85] monotonicity[86] and the structure of the problem.[87] Numerical evidence confirm[88] that symbolic preprocessing and automatic generation of redundant constraints for pruning can speed up the search procedure by several orders of magnitude in time.

Except the benchmark `eco9`, the studied problems show some similarity in their structure. The equations are loosely coupled by a few variables in a certain pattern ('cascade-like pattern'). The numerical evidence presented here and published for other large-scale problems with similar sparsity pattern[49] raises the question whether a tractable problem class can be identified based on the structure of the problem. Answering this question is the authors' ongoing research.

Appendix

The computations have been carried out with the following hardware and software configuration. Processor: Intel® Core™ 2 Duo E6750 @ 2.66 GHz. Memory: 2 GB RAM. Operating system: Debian with Linux kernel 2.6.26-2-amd64. Compiler: GCC 4.3.2 x86_64-linux-gnu. Version of the relevant software libraries: IPOPT 3.6.1, C-XSC 2.4.0, GNU GLPK 4.39.

References

1. Güttinger TE. Multiple Steady States in Azeotropic and Reactive Distillation. PhD Thesis; ETH No. 12720, ETH Zürich, Switzerland. 1998.

2. Bekiaris N, Morari M. Multiple Steady States in Distillation: ∞/∞ Predictions, Extensions, and Implications for Design, Synthesis, and Simulation. Ind Eng Chem Res. 1996;35:4264–4280.

3. Wayburn TL, Seader JD. Homotopy Continuation Methods for Computer-Aided Process Design. Comp Chem Eng. 1987;11:7–25.

4. Seydel R, Hlavacek V. Role of continuation in engineering analysis. Chemical engineering science. 1987;42:1281–1295.

5. Nocedal J, Wright SJ. Numerical Optimization. New York: Springer, 1999.

6. Perry RH, Green DW. Perry's Chemical Engineers' Handbook. 8th edition. Section 13: Distillation. McGraw-Hill Professional, 2007:33–34.

7. Lin WJ, Seader JD, Wayburn TL. Computing Multiple Solutions to Systems of Interlinked Separation Columns. AIChE Journal. 1987;33:886–897.

8. Kovach JW, Seider WD. Heterogeneous Azeotropic Distillation: Homotopy-Continuation Methods. Comput Chem Eng. 1987;11:593–605.

9. Kovach JW, Seider WD. Heterogeneous Azeotropic Distillation: Experimental and Simulation Results. AIChE Journal. 1987;33:1300–1314.

10. Chang YA, Seader JD. Simulation of continuous reactive distillation by a homotopy-continuation method. Comp Chem Eng. 1988;12:1243–1255.

11. Chen F, Huss RS, Doherty MF, Malone MF. Multiple steady states in reactive distillation: kinetic effects. Computers and Chemical Engineering. 2002;26:81–93.

12. Boston JF, Sullivan SL. A new class of solution methods for multicomponent, multistage separation processes. Can J Chem Eng. 1974;52:52–63.

13. Naphthali LM, Sandholm DP. Multicomponent separation calculations by linearization. AIChE J. 1971;17:148–153.

14. Vadapalli A, Seader JD. A generalized framework for computing bifurcation diagrams using process simulation programs. Computers and Chemical Engineering. 2001;25:445–464.

15. Kannan A, Joshi MR, Reddy GR, Shah DM. Multiple-Steady-States Identification in Homogeneous Azeotropic Distillation Using a Process Simulator. Ind Eng Chem Res. 2005;44:4386–4399.

16. Güttinger TE, Dorn C, Morari M. Experimental Study of Multiple Steady States in Homogeneous Azeotropic Distillation. Ind Eng Chem Res. 1997;36:794–802.

17. Doedel EJ, Wang XJ, Fairgrieve TF. AUTO94: Software for Continuation and Bifurcation Problems in Ordinary Differential Equations. Technical Report CRPC-95-1, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125, 1995.

18. Bekiaris N, Meski GA, Radu CM, Morari M. Multiple Steady States in Homogeneous Azeotropic Distillation. Ind Eng Chem Res. 1993;32:2023–2038.

19. Güttinger TE, Morari M. Comments on “Multiple Steady States in Homogeneous Azeotropic Distillation”. Ind Eng Chem Res. 1996;35:2816–2816.

20. Neumaier A. Complete Search in Continuous Global Optimization and Constraint Satisfaction. Acta Numerica. Cambridge: University Press, 2004:271–369.

21. Belotti P, Lee J, Liberti L, Margot F, Waechter A. Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software. 2009;24:597–634.

22. Gwaltney CR, Lin Y, Simoni LD, Stadtherr MA. Interval Methods for Nonlinear Equation Solving Applications. Handbook of Granular Computing. Chichester, UK: John Wiley & Sons Ltd. 2008:81–96.

23. Adjiman CS, Dallwig S, Floudas CA, Neumaier A. A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs - I. Theoretical Advances. Computers and Chemical Engineering. 1998;22:1137–1158.

24. Adjiman CS, Androulakis IP, Floudas CA. A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs - II. Implementation and Computational Results. Computers and Chemical Engineering. 1998;22:1159–1179.

25. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J. Global optimization in the 21st century: Advances and challenges. Computers and Chemical Engineering. 2005;29:1185–1202.

26. Fourer R, Gay DM, Kernighan BW. A Modeling Language for Mathematical Programming. Management Science. 1990;36:519–554.

27. Gau CY, Stadtherr MA. Deterministic Global Optimization for Error-in-Variables Parameter Estimation. AIChE J. 2002;48:1192–1197.

28. Kearfott RB, Hongthong S. Validated Linear Relaxations and Preprocessing: Some Experiments. SIAM Journal on Optimization. 2005;16:418–433.

29. ASOL – Affine Solver. https://sourceforge.net/projects/asol/

30. Kolev LV. A general interval method for global nonlinear dc analysis. Proc. of the 1997 European Conf. on Circuit Theory and Design, ECCD'97. Technical University of Budapest, 30th August - 3rd Sept., Budapest, Hungary. 1997;3:1460–1462.

31. Kolev LV. A New Method for Global Solution of Systems of Non-Linear Equations. Reliable Computing. 1998;4:125–146.

32. Kolev LV. An efficient interval method for global analysis of non-linear resistive circuits. Int J Circuit Theory Appl. 1998;26:81–92.

33. Kolev LV, Mladenov VM. A linear programming implementation of a interval method for global non-linear DC analysis. IEEE International Conference on Electronics, Circuits and Systems. 1998;1:75–78.

34. Kolev LV. An Improved Method for Global Solution of Non-Linear Systems. Reliable Computing. 1999;5:103–111.

35. Kolev LV. An Interval Method for Global Nonlinear Analysis. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications. 2000;47:675–683.

36. Kolev L, Nenov I. A Combined Interval Method for Global Solution of Nonlinear Systems. Proc. XXIII Int. Conf. Fundamentals of Electronics and Circuit Theory SPETO 2000. Gliwice, Poland, 2000:365–368.

37. Kolev LV. An improved interval linearization for solving non-linear problems. Numerical Algorithms. 2004;37:213–224.

38. Nenov IP, Fylstra DH. Interval Methods for Accelerated Global Search in the Microsoft Excel Solver. Reliable Computing. 2003;9:143–159.

39. Yamamura K, Kumakura T, Inoue Y. Finding All Solutions of Nonlinear Equations Using Inverses of Approximate Jacobian Matrices. IEICE Trans Fundamentals. 2001;E84-A:2950–2952.

40. Yamamura K, Tanaka K. Finding all solutions of weakly nonlinear equations using the dual simplex method. Electronics and Communications in Japan (Part III: Fundamental Electronic Science). 2006;89:1–7.

41. Miyajima S, Kashiwagi M. Existence test for solution of nonlinear systems applying affine arithmetic. J Comput Appl Math. 2007;199:304–309.

42. Baharev A, Rév E. Reliable Computation of Equilibrium Cascades with Affine Arithmetic. AIChE J. 2008;54:1782–1797.

43. Hansen ER, Walster GW. Global Optimization Using Interval Analysis. New York: Marcel Dekker, Inc., 2004.

44. Stolfi J, Figueiredo LH. Self-Validated Numerical Methods and Applications. Monograph for the 21st Brazilian Mathematics Colloquium (CBM'97), IMPA. Rio de Janeiro, Brazil, 1997:75–76.

45. Baharev A, Achterberg T, Rév E. Computation of an extractive distillation column with affine arithmetic. AIChE Journal. 2009;55:1695–1704.

46. Lin Y, Stadtherr MA. LP Strategy for Interval-Newton Method in Deterministic Global Optimization. Ind Eng Chem Res. 2004;43:3741–3749.

47. Lin Y, Stadtherr MA. Advances in Interval Methods for Deterministic Global Optimization in Chemical Engineering. J Global Optimization. 2004;29:281–296.

48. Yamamura K, Kawata H, Tokue A. Interval solution of nonlinear equations using linear programming. BIT. 1998;38:186–199.

49. Yamamura K, Suda K, Tamura N. LP narrowing: A new strategy for finding all solutions of nonlinear equations. Applied Mathematics and Computation. 2009;215:405–413.

50. Lebbah Y, Michel C, Rueher M, Daney D, Merlet JP. Efficient and Safe Global Constraints for Handling Numerical Constraint Systems. SIAM J Numer Anal. 2004;42:2076–2097.

51. Vu XH, Sam-Haroud D, Faltings B. Enhancing numerical constraint propagation using multiple inclusion representations. Annals of Mathematics and Artificial Intelligence. 2009;55:295–354.

52. Kearfott RB. Preconditioners for the Interval Gauss-Seidel Method. SIAM J Numer Anal. 1990;27:804–822

53. Kearfott RB, Hongthong S. On preconditioners and splitting in the interval Gauss–Seidel method. Technical report, University of Louisiana at Lafayette. 2005.

54. Kearfott RB. A comparison of some methods for bounding connected and disconnected solution sets of interval linear systems. Computing. 2008;82:77–102.

55. Schichl H, Neumaier A. Interval Analysis on Directed Acyclic Graphs for Global Optimization. J Glob Optim. 2005;33:541–562.

56. Kearfott RB. Decomposition of Arithmetic Expressions to Improve the Behavior of Interval Iteration for Nonlinear Systems. Computing. 1991;47:169–191.

57. Beelitz T, Frommer A, Lang B, Willems P. Symbolic-Numeric Techniques for Solving Nonlinear Systems. PAMM. 2005;5:705–708.

58. Goualard F. Interval Multivalued Inverse Functions: Relational Interval Arithmetic and its Use. Invited talk, SCAN’08, El Paso, TX, 2008.

59. Achterberg T. Constraint Integer Programming. PhD Thesis, TU Berlin, 2007:154–157.

60. Wächter A, Biegler LT. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming. 2006;106:25–57.

61. Hansen ER. Global Optimization Using Interval Analysis. New York: Marcel Dekker; 1992:98–100.

62. Dorn C, Güttinger TE, Wells GJ, Morari M. Stabilization of an Unstable Distillation Column. Ind Eng Chem Res. 1998;37:506–515.

63. Petlyuk FB. Distillation Theory and Its Application to Optimal Design of Separation Units. Cambridge University Press, Cambridge, UK, 2004.

64. Jacobsen EW, Skogestad S. Multiple steady states in ideal two-product distillation. AIChE Journal. 1991;37:499–511.

65. Averick BM, Carter RG, Moré JJ, Xue GL. The MINPACK-2 test problem collection. Preprint MCS-P153-0694, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992.

66. Frank-Kamenetskii DA. Diffusion and Heat Exchange in Chemical Kinetics. Princeton Univ. Press, Princeton, NJ, 1955.

67. Morgan A. Solving polynomial systems using continuation for engineering and scientific problems. Prentice-Hall, Englewood Cliffs, New Jersey, 1987:148–148.

68. Van Hentenryck P, Michel L, Deville Y. Numerica: a modeling language for global optimization. Massachusetts: The MIT Press, 1997:144–144.

69. Van Hentenryck P, McAllester D, Kapur D. Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. 1997;34:797–827.

70. Hammer R, Hocks M, Kulisch U, Ratz D. C++ Toolbox for Verified Computing I, Basic Numerical Problems. Berlin: Springer-Verlag, 1995.

71. Neumaier A, Shcherbina O. Safe bounds in linear and mixed-integer programming. Mathematical Programming. 2004;99:283–296.

72. Jansson C. Rigorous Lower and Upper Bounds in Linear Programming. SIAM J Optim. 2004;14:914–935.

73. Jansson C. On self-validating methods for optimization problems. pp. 381–438 in: Topics in validated computation (J. Herzberger, ed.). Amsterdam: Elsevier, 1994.

74. Van Iwaarden RJ. An improved unconstrained global optimization algorithm. PhD. Thesis, University of Colorado at Denver, CO, 1996.

75. de Figueiredo LH, Van Iwaarden R, Stolfi J. Fast interval branch-and-bound methods for unconstrained global optimization with affine arithmetic. Technical report IC-97-08, Institute of Computing, Univ. of Campinas, Brazil, 1997.

76. Kearfott RB. Empirical Evaluation of Innovations in Interval Branch and Bound Algorithms for Nonlinear Systems. SIAM Journal on Scientific Computing. 1997;18:574–594.

77. Schichl H, Neumaier A. Exclusion regions for systems of equations. SIAM J Numer Anal 2004;42:383–408.

78. Benhamou F, McAllester D, Van Hentenryck P. CLP(Intervals) revisited. Procs Intl Symp on Logic Prog. The MIT Press, 1994:124–138.

79. McAllester DA, Van Hentenryck P, Kapur D. Three cuts for accelerated interval propagation. Technical Memo AIM-1542, MIT, AI Lab., May 1995.

80. Walster GW, Hansen E. Using pillow functions to efficiently compute crude range tests. Numerical Algorithms. 2004;37:401–415.

81. Benhamou F, Older WJ. Applying interval arithmetic to real, integer, and boolean constraints. The Journal of Logic Programming. 1997;32:1–24.

82. Benhamou F, Goualard F, Granvilliers L, Puget JF. Revising hull and box consistency. Proceedings of the International Conference on Logic Programming. Las Cruces, New Mexico: MIT Press, 1999:230–244.

83. Vu XH, Schichl H, Sam-Haroud D. Interval propagation and search on directed acyclic graphs for numerical constraint solving. J Glob Optim. 2009;45:499–531.

84. Goldsztejn A, Goualard F. Box Consistency through Adaptive Shaving. Proceedings of the 25th Annual ACM Symposium on Applied Computing (Constraint Solving and Programming track). ACM. 2010.

85. Araya I, Neveu B, Trombettoni G. Exploiting Common Subexpressions in Numerical CSPs. Lecture Notes In Computer Science. Vol. 5202. Principles and Practice of Constraint Programming. Berlin: Springer-Verlag, 2008:342–357.

86. Araya I, Neveu B, Trombettoni G. An Interval Constraint Propagation Algorithm Exploiting Monotonicity. International workshop IntCP, interval analysis, constraint propagation, applications, at CP conference, 2009:65–83.

87. Neveu B, Trombettoni G, Chabert G. Improving inter-block backtracking with interval Newton. Constraints. 2010;15:93–116.

88. ALIAS library, the COPRIN project, “Difficult problems” page
http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/node6.html

Table 1. Computational results and the efficiency of local search. Probing is enabled.

Problem name	Number of variables	Splits	First solution found in[1]	Found before the first[2]	Number of all solutions	Computation time (s)	
						Local search disabled	Local search enabled
mss20[3]	140	86	initial box	3	3	11475	11586
extr22	387	0	initial box	1	1	363	364
extr30	523	0	initial box	1	1	678	679
Jacobsen96_3	19	4	initial box	2	5	3	3
Jacobsen59_2	19	2	initial box	2	3	2	3
Bratu30	30	1	initial box	1	2	2	3
eco9	8	751	initial box	13	16	44	50

(1) Found by the local solver.

(2) Number of solutions found by the local solver before the branch-and-prune method returns the first solution.

(3) Problem specific bisection rule is used, see *subsection 2.6*.

Table 2. Computational efforts with and without probing. Local search is disabled.

Problem name	Number of splits		Computation time (s)	
	Probing enabled	Probing disabled	Probing enabled	Probing disabled
Jacobsen96_3	4	86	3	2
Jacobsen59_2	2	438	2	5
Bratu30	1	21	2	1
eco9	751	14873	44	27

List of Figure Captions

Figure 1a. Steady state distillate compositions as the function of the distillate flow rate.

Figure 1b. Steady state bottom compositions as the function of the distillate flow rate.

Figure 2a. Liquid phase composition profiles of all three steady states at a distillate flow rate of D = 0.455 mol/s. Components: ○ methanol, Δ methyl-butyrate, × toluene.

Figure 2b. Temperature profiles of all three steady states at a distillate flow rate of D = 0.455 mol/s.

Figure 3a. Distillate composition and reflux molar flow rate as a function of mass reflux flow rate. Boilup V_N = 2.0 kmol/min. (1 kg/min = 1/60 kg/s; 1 kmol/min = 1/60 kmol/s)

Figure 3b. Distillate composition and reflux molar flow rate as a function of mass reflux flow rate. Boilup V_N = 3.0 kmol/min. Dashed: infeasible in practice. (1 kg/min = 1/60 kg/s; 1 kmol/min = 1/60 kmol/s)

Figure 4a. Liquid phase composition profiles. Components: ○ acetone, Δ methanol, × water.

Figure 4b. Temperature profiles.

Figure 1.

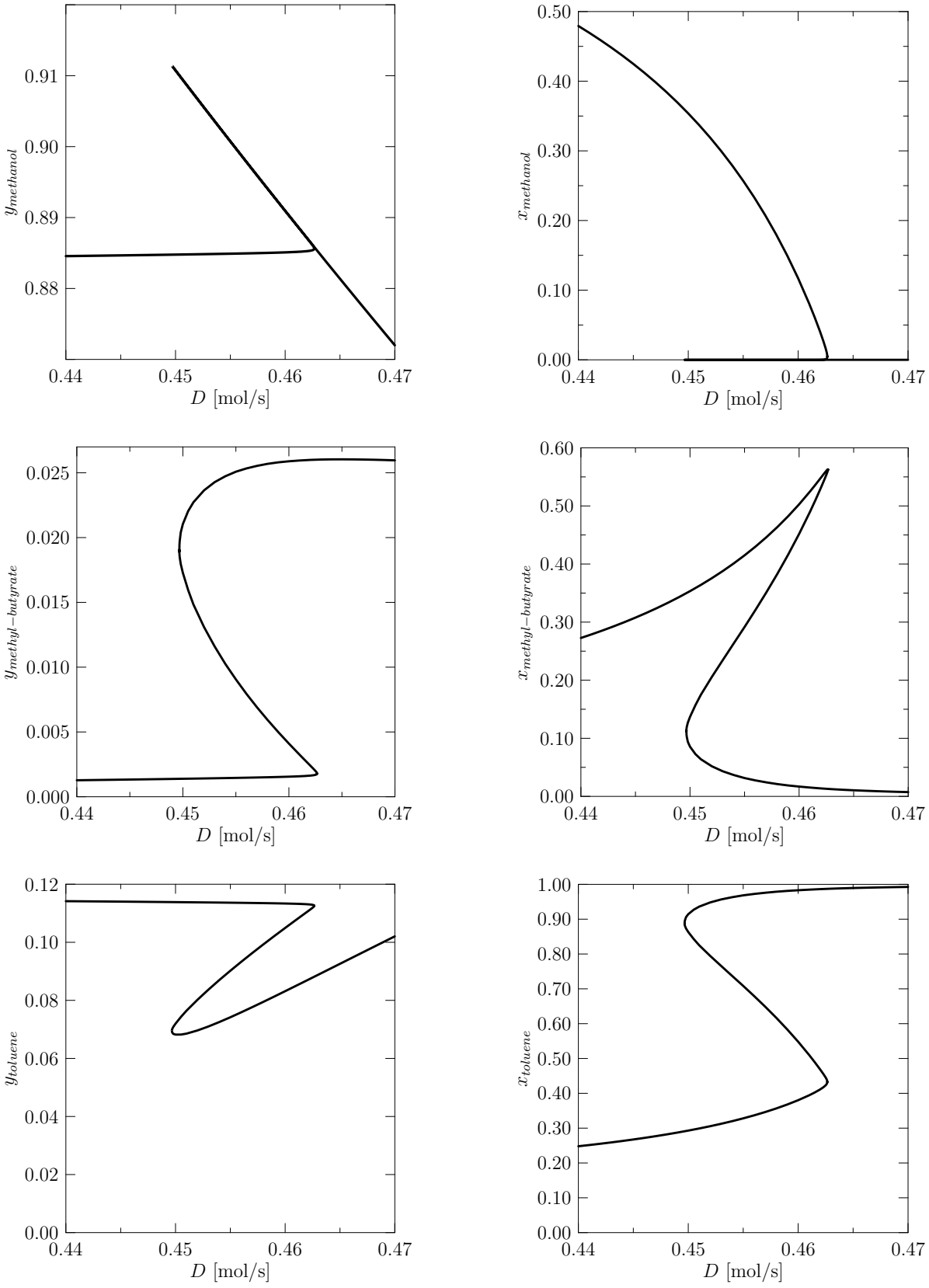

$y_{methanol}$
$x_{methanol}$
$y_{methyl-butyrate}$
$x_{methyl-butyrate}$
$y_{toluene}$
$x_{toluene}$
D [mol/s]

Figure 2.

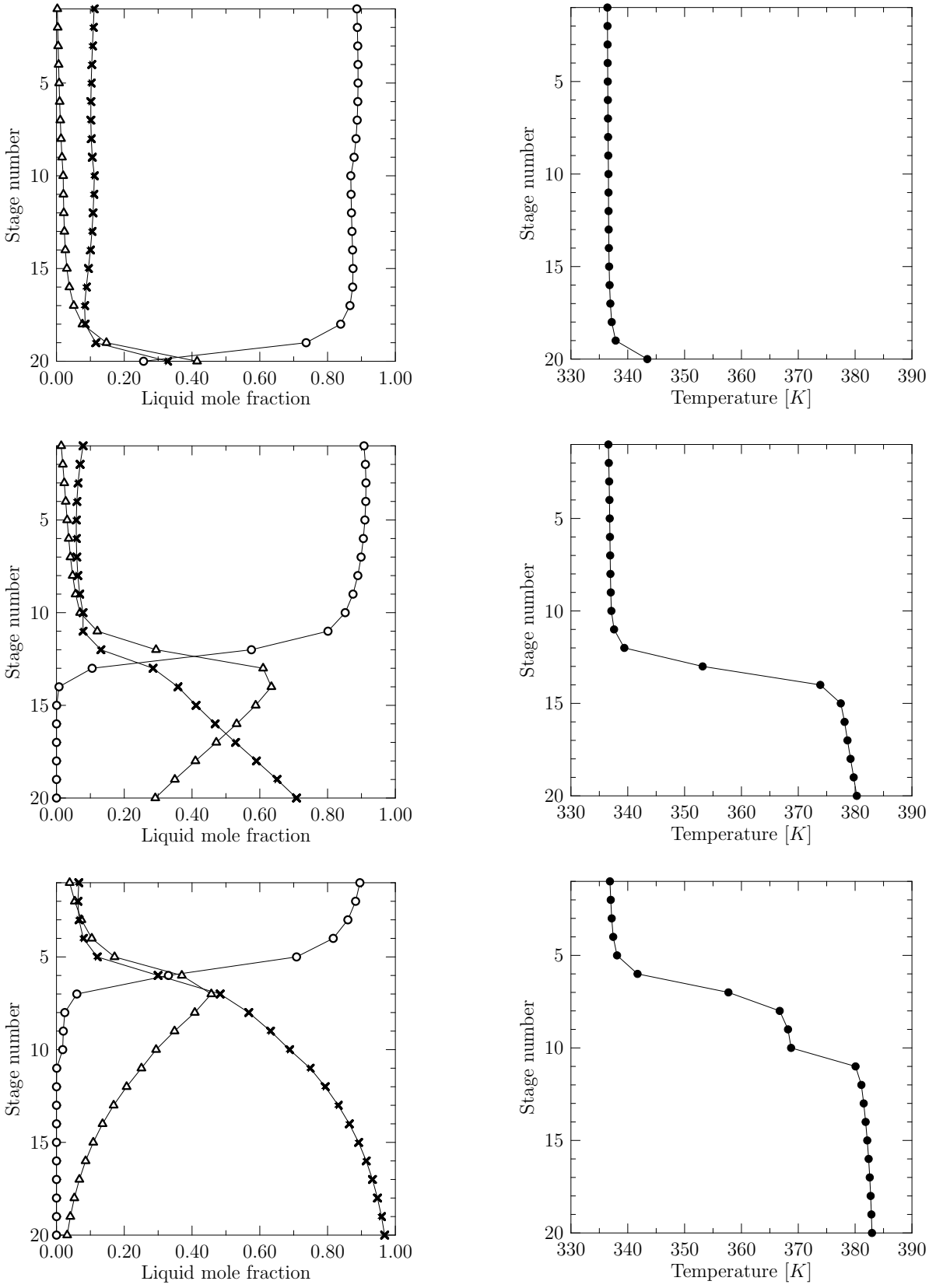

Stage number
Liquid mole fraction
Temperature [K]
0.00
0.20
0.40
0.60
0.80
1.00
330
340
350
360
370
380
390
5
10
15
20

Figure 3.

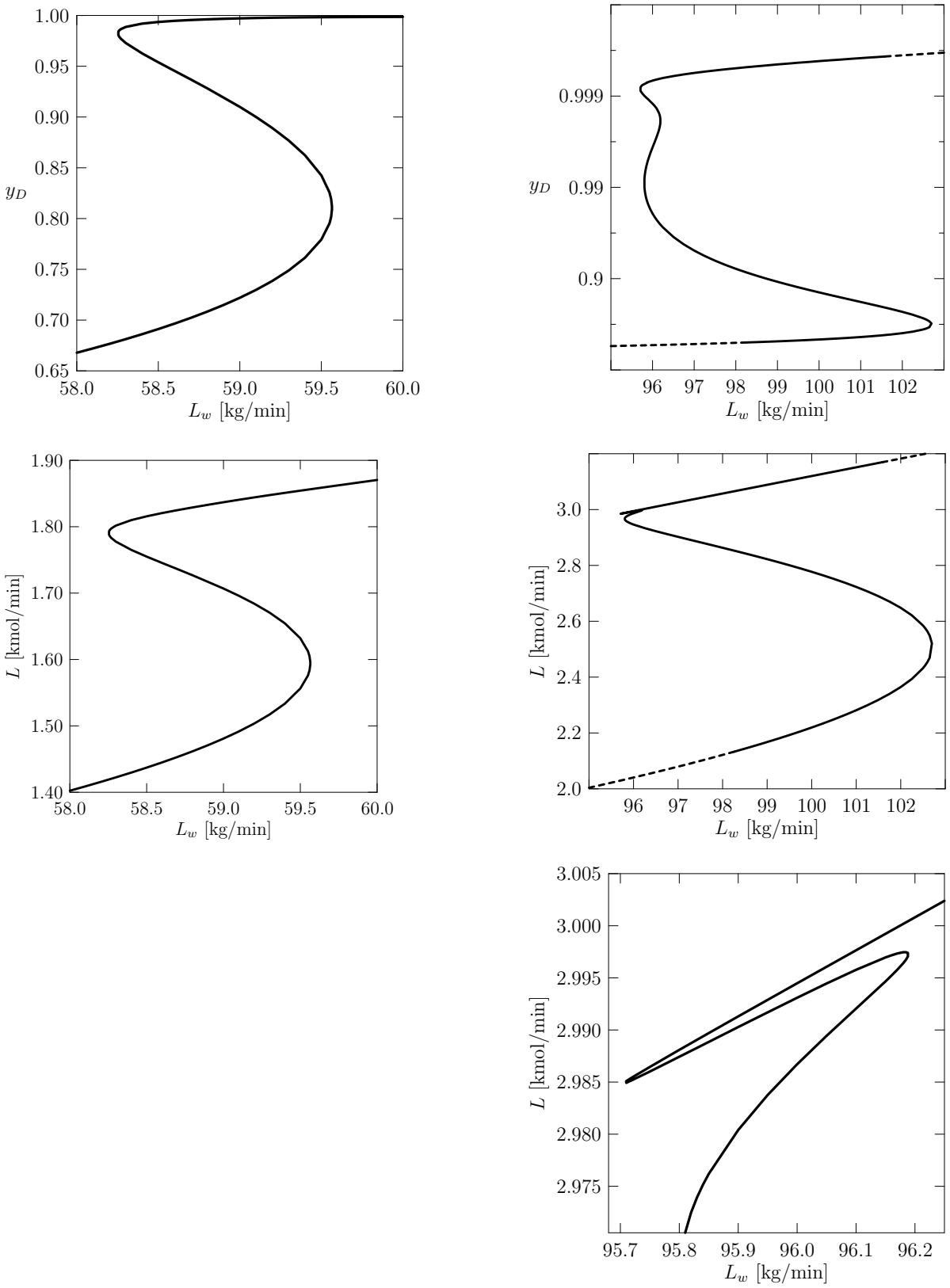
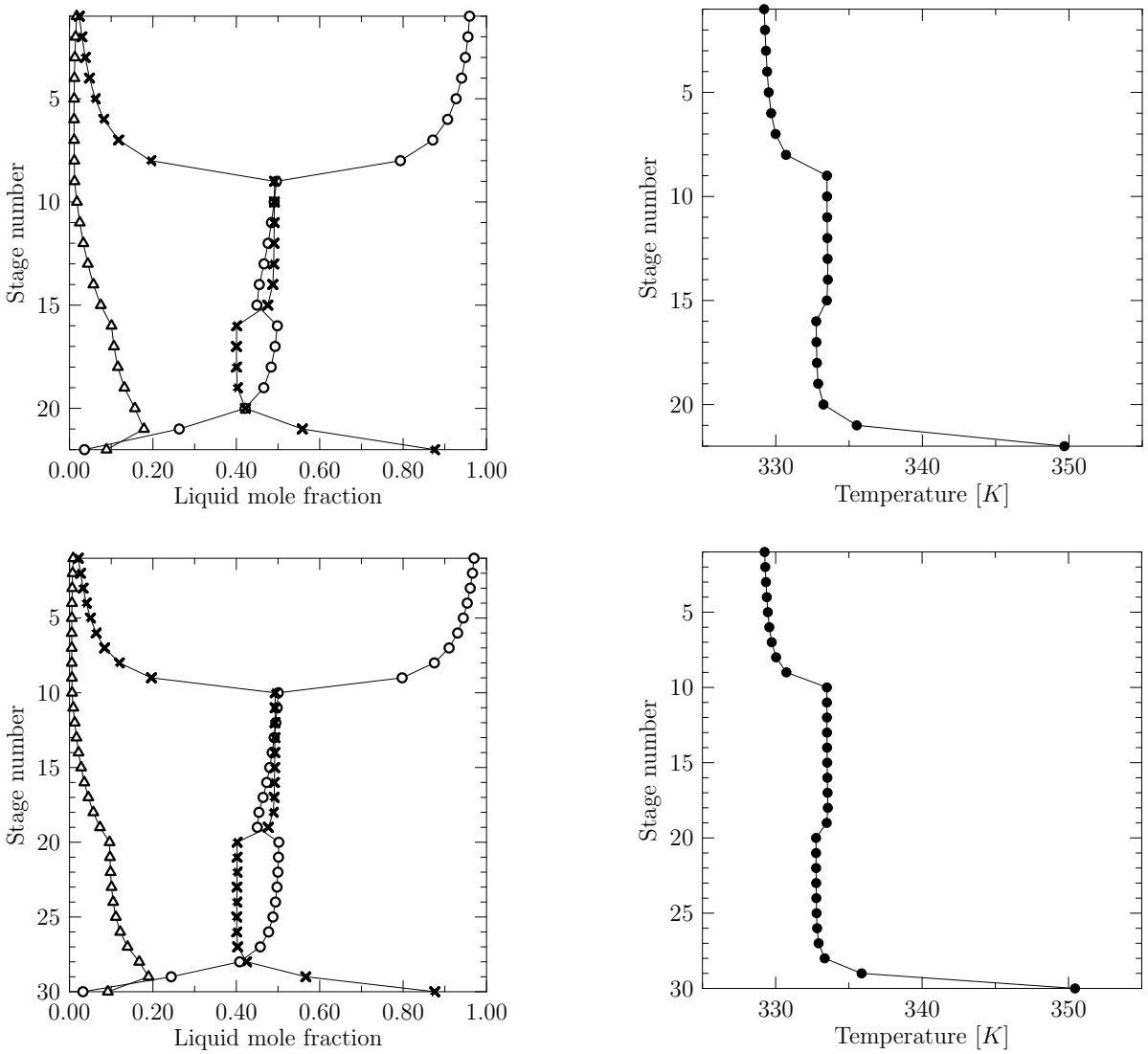

1.00
0.95
0.90
0.85
y_D
0.80
0.75
0.70
0.65
58.0
58.5
59.0
59.5
60.0
L_w [kg/min]
0.999
y_D
0.99
0.9
96
97
98
99
100
101
102
L_w [kg/min]
1.90
1.80
1.70
L [kmol/min]
1.60
1.50
1.40
58.0
58.5
59.0
59.5
60.0
L_w [kg/min]
3.0
2.8
2.6
L [kmol/min]
2.4
2.2
2.0
96
97
98
99
100
101
102
L_w [kg/min]
3.005
3.000
2.995
L [kmol/min]
2.990
2.985
2.980
2.975
95.7
95.8
95.9
96.0
96.1
96.2
L_w [kg/min]

Figure 4.


Stage number
Liquid mole fraction
Temperature [K]
0.00
0.20
0.40
0.60
0.80
1.00
330
340
350
5
10
15
20
25
30