Computation of an extractive distillation column with affine arithmetic

Ali Baharev[1,3], Tobias Achterberg[2], Endre Rév[1]

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

(2) Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)
Division Scientific Computing, Department Optimization
Takustr. 7, D-14195 Berlin-Dahlem, Germany

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

This is a preprint of an article published in

AIChE Journal, 2009, 55 (7), 1695-1704

http://www.interscience.wiley.com

DOI 10.1002/aic.11777
http://dx.doi.org/10.1002/aic.11777

Abstract

The need of reliably solving systems of nonlinear equations often arises in the everyday practice of chemical engineering. In general, standard methods cannot provide theoretical guarantee for convergence to a solution, cannot reliably find multiple solutions, and cannot prove non-existence of solutions. Interval methods provide tools to overcome these problems, thus achieving reliability. To the authors' best knowledge, computation of distillation columns with interval methods have not yet been considered in the literature.

This paper presents significant enhancements compared to a previously published interval method of the authors. The proposed branch-and-prune algorithm is guaranteed to converge, and is fairly general at the same time. If no solution exists then this information is provided by the method as a result. Power of the suggested method is demonstrated by solving, with guaranteed convergence, even the MESH equations of a 22 stage extractive distillation column with a ternary mixture.

Topical heading: Separations

Keywords: separation, MESH equations, root finding, affine arithmetic, interval arithmetic

Introduction

Computing steady states of counter-current multistage processes is equivalent to finding solutions of large scale non-linear equation systems. Although a good deal of effort has been made in constructing efficient and robust computation techniques, and impressive results have been achieved[1,2], generally there is no theoretical guarantee for convergence to the true solution. The routines developed for computing steady states are sensitive to initial estimates, and if no solution is achieved after several attempts with different initial points then one does not know whether the initial estimation is poor or simply that no solution exists for the specified circumstances. Moreover, there are specifications that give rise to several solutions (output multiplicity[3,4]) but standard methods cannot guarantee that all solutions are found.

Interval methods provide tools to overcome these problems: these tools either provide all the solutions or prove nonexistence of solution of a general nonlinear equation system with mathematical certainty. The Interval Newton / Generalized Bisection method (IN/GB) has been successfully applied to solve a wide variety of chemical engineering problems[5] such as computation of phase stability with activity coefficient models[6,7], cubic equation-of-state (EOS) models[8,9], modeling liquid-liquid equilibrium of ionic liquid systems[10], calculation of critical points from cubic EOS models[11], location of azeotropes[12], parameter estimation using standard least squares and error-in-variables[13]. Interval arithmetic can also be applied to compute validated solutions of initial value problems for ODEs[14,15], to enclose all solutions of two-point boundary value problems for ODEs[16], and to deterministic global optimization of nonlinear dynamic systems[17].

Interval methods improved considerably during the past few decades. *State-of-the-art* variants of IN/GB, involving advanced preconditioning[18], linear programming[19] and / or constraint propagation on directed acyclic graphs (DAG)[20-22], may be several orders of magnitude faster than the 'textbook' Interval Newton / Gauss-Seidel[23] (IN/GS) algorithm with the so-called midpoint inverse preconditioner.

A new linearization technique, based on affine arithmetic (AA)[24-27], has been proposed recently by Kolev[28-36]. Numerical evidence published in the literature[28-32,34,37-41] suggests that the new technique may be superior to the traditional linearization techniques such as the interval Newton or the Krawczyk[42] method. Linear programming may be preferable as pruning technique for this new linearization in the case of the vapor-liquid equilibrium cascades[41].

The aim of the present work is to combine the above ideas in order to obtain an efficient interval methodology and thus extend the capability of these methods to compute such a complex and large scale chemical engineering problem as the steady state of an industrial scale distillation column. Power of the suggested method is demonstrated by solving a 22 stage extractive distillation column with a ternary mixture. To the authors' best knowledge, computation of industrial scale distillation columns with interval methods have not yet been considered in the literature. It is perhaps so because of the extensive complexity and dimensionality of these problems. The proposed method is also able to provide information on infeasibility if the equation system has no solution, and is able to find several solutions in the studied domain if they exist.

Procedure for locating all solutions

Here the procedure used in this work for locating all the solutions is described. Three major components of the procedure may be distinguished: linearization, pruning (discarding some regions of the variables' domain not containing a solution), and bisection.

Linearization

Given

$$\mathbf{f(x)} = \mathbf{0}, \qquad \text{where } \mathbf{f}\text{: } R^n \to R^n,\ x_j \in X_j = [\underline{x}_j, \overline{x}_j] \tag{1}$$

the goal is to bound all solutions of (1) or prove their absence using a first order interval method. Linearization of (1) with the mixed affine arithmetic and interval arithmetic model[25] (mixed AA/IA, pp. 75-76) yields a linear constraint system in the form of

$$A(\boldsymbol{X})\mathbf{x}+\boldsymbol{B}(\boldsymbol{X}) = \mathbf{0} \qquad \mathbf{x} \in \boldsymbol{X} \tag{2}$$

which must be satisfied by any of the solution vector(s) $\mathbf{x}^* \in \boldsymbol{X}$; where $A(\boldsymbol{X})$ is a real $n\times n$ matrix and $\boldsymbol{B}(\boldsymbol{X})$ is an interval vector.

The mixed AA/IA was used only at the critical parts (where otherwise division by zero or calling the logarithm function with negative argument would have occurred) in the previous work[41] of the authors due to implementation design flaws. Based on the conclusions of the previous work, the affine class has been redesigned and implemented in C++. The mixed AA/IA is used during the entire solution process in the present work. All the optimization techniques proposed in the monograph[25] are incorporated (pp. 79-83); most noticeably the affine class uses a memory pool which is automatically managed by the constructors and destructors of the affine class.

Pruning based on constraint propagation

Two methods are used for discarding from the box some regions not containing a solution (shortly: for pruning). One of them is based on equationwise constraint propagation[32]: the formula

$$X_j^{new} = X_j \cap \left(\frac{1}{a_{ij}} \left(B_i - \sum_{k \neq j} a_{ik} X_k \right) \right) \tag{3}$$

is evaluated equation by equation, and for each variable in the actual equation. In formula (3), a_{ij}, X_j and B_i are the corresponding elements of the real matrix A, interval vector $\boldsymbol{X}$ and $\boldsymbol{B}$ in (2), respectively. Redundant equations can also be involved in the above propagation.

Equation (3) is the affine analogue of the well-known Interval Newton Gauss-Seidel iteration, in (3) the denominator is real whereas in the Gauss-Seidel iteration division by an interval is required.

Pruning based on linear programming

The other method of pruning is based on linear programming; this is the so-called LP pruning. The original non-linear function (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 below:

$$\begin{aligned} &\min / \max\ x_j \quad \text{for all } j \\ &\text{subject to} \\ &\qquad A\mathbf{x} = -\boldsymbol{B} \\ &\qquad \mathbf{x} \in \boldsymbol{X} \end{aligned} \tag{4}$$

where the constraints are the same as (2) and remain unchanged during the entire pruning procedure.

At first glance it seems as if $2n$ LP subproblems have to be solved (n denotes the number of variables) but this is not the case. The minimization / maximization subproblem for x_j can be skipped if x_j equals its lower / upper bound, respectively, in *any* of the primal feasible solution vectors obtained during the pruning procedure. The gain is obvious.

Any primal feasible solution of the LP subproblems (4) remains primal feasible after manipulating the objective arbitrarily. It follows that only the first LP subproblem has to be solved from scratch; all other subproblems should use the optimal solution of the preceding subproblem as a primal feasible basis and run only Phase II of the primal simplex algorithm, thus hopefully reducing the computational efforts. The naïve sequence[41] to process the x_j variables would be min x_1, max x_1, min x_2, max x_2, … *etc* but the subproblems min x_1 and max x_1 are likely to produce completely different solutions. As a consequence, this is expected to result in a lot of simplex iterations when using the optimal solution of subproblem min x_1 as the initial primal feasible basic solution when solving the subproblem max x_1. Considering this idea, a simple heuristic is proposed for selecting the subsequent subproblem. Find that variable which is the closest to its lower / upper bound and has not yet been considered in the pruning step; then solve the corresponding optimization problem (min x_j if x_j is close to its lower bound, or max x_j if x_j is close to its upper bound). The assumption behind this heuristic is that the current primal feasible solution should not be far from the optimal solution for that variable. The enhancements presented in this subsection will be referred to as Achterberg's heuristic. Numerical examples, suggesting the superiority of this heuristic to the previous implementation[41], will be presented after the subsection *Separation problem* which describes the numerical examples used for comparison.

Bisection

A simple rule is used in the present paper: bisect the box along the domain of the widest component. If the problem is badly scaled globally, it is desirable to choose the scale factors after the first LP pruning step so as all edges of the box equal unity. Unfortunately, this bisection rule may not be robust enough in general as it is demonstrated at the *Numerical examples*. The choice of the variable along which to bisect the box is a hard and open question in general. In the case of the *Separation problem* below, however, a simple yet efficient problem specific bisection rule can be constructed if the above simple rule fails; numerical examples are also presented at subsection *Problem specific bisection rule*.

Branch–and–prune algorithm to bound all solutions

Step 0. Initialize a stack of boxes with the original box.

Step 1. If the stack is empty then exit else pop the top-most box $X^{(k)}$ off the stack.

Step 2. Linearize the system of equations in $X^{(k)}$ with the mixed affine arithmetic and interval arithmetic model. If the obtained lower and upper bounds on the range of **f** do not enclose **0** then discard the box and go to step 1.

Step 3. Apply equationwise constraint propagation; if an empty interval is obtained then discard the box and go to step 1.

Step 4. Re-linearize **f** in the (hopefully) contracted box, and apply LP pruning. If the first LP problem is infeasible then discard the box and go to step 1.

Step 5. If the widest component of the contracted box is below a pre-defined threshold then print the box (it may contain a solution) else bisect it along the domain of the widest component, push the resulting two boxes to the stack, and go to Step 1.

Note. Steps 2, 3, and 4 may be repeated if the box could be sufficiently reduced in size (analogous to the idea of Hansen[43], pp. 98-100). However it is not straightforward how to quantify that the box is 'sufficiently' contracted. Instead, one can simply repeat steps 2, 3, and 4 for a fixed number of times irrespective of the rate of pruning; this is obviously not the most effective way but is easy to implement.

Effect of the enhancements concerning the linearization

As discussed earlier at subsection *Linearization*, the present implementation uses the mixed affine arithmetic / interval arithmetic (mixed AA/IA) model during the entire computations, and the affine class uses a memory pool. The expected effects of these enhancements are first drawn up, and then comparisons to the previous results[41] are given. The numerical results confirm the expectations.

Using the Chebyshev approximation (Stolfi and Figueiredo[25], pp. 56-57), the difference between the mixed AA/IA and the pure affine arithmetic is that the so-called overshoot (p. 63) is cut off in the mixed AA/IA leading to better approximation. Note that the overshoot is unwanted because it can result in division by zero, or calling the logarithm function with negative argument[41]. The mixed AA/IA with Chebyshev approximation gives identical results with respect to the range of the enclosure as the pure affine arithmetic with min-range approximation (pp. 64-65); while the enclosure of the mixed AA/IA is mostly tighter than the pure affine arithmetic with min-range approximation. See also Figures 1a-d. The better / mostly better approximation is expected to give smaller number of iterations.

The previous implementation was based on the `map` container of the C++ Standard Template Library, the present one uses arrays arranged in a memory pool. This enhancement obviously cannot influence the number of iterations but is expected to give at least an order of magnitude speed-up (Item 10 in the book of Meyers[44]). The numerical examples with the AA/CP method of Baharev and Rév[41] are re-computed; these examples do not involve linear programming which is important to maintain comparability since the LP pruning part is also changed in the current implementation. The software and hardware environment is given in the appendix. As shown in Tables 1 to 3, the numerical examples confirm the expectations.

Liquid phase split

Given the gross composition, the goal is to determine the relative amounts and compositions of at most two phases in equilibrium. The equifugacity conditions are solved at constant pressure and temperature.

Counter-current equilibrium cascade

The steady state of one theoretical stage between a reboiler and a condenser (VLE cascade with 3 stages) is computed by solving the MESH equations (component **m**aterial balances, **e**quilibrium conditions, **s**ummation equations, and **h**eat balance equations) simultaneously.

Separation problem

Steady state of continuous extractive distillation of acetone and methanol with water as entrainer is computed. The equilibrium stage used in the current work is shown in Figure 2; the scheme of the studied distillation column is given in Figure 3.

Variables and specifications

Specifications are the reflux ratio R, distillate flow rate D, composition and flow rate of the solvent feed and the main feed, total number of stages, and feed stage locations. As shown in Figure 3, total condenser and total reboiler is used. Variables are listed in Table 4.

Enthalpy model

A fairly simplified enthalpy model is used: the molar enthalpy of the vapor phase is the mole fraction weighted average of the constant molar heat of vaporization (λ_i) of the components, $H = \Sigma\ \lambda_i y_i$. Other heat effects are neglected. The molar heat of vaporization values are given in Table A1. The assumption behind this model is that the heat of vaporization is at least an order of magnitude higher than the other enthalpy changes in the liquid or vapor phase, which is reasonable for distillation in practice. This model may seem rough but when comparing the computational results performed with this model to those obtained with commercial simulators using detailed and thermodynamically consistent enthalpy model, the result are the same up to 2-3 digits.

Equations related to the variables involved in the pruning

These are the so-called MESH equations.

Component material balance (**M**) equations:

$$l_{i,j-1} + v_{i,j+1} + f_{i,j} - (l_{i,j} + v_{i,j}) = 0 \qquad i = 1 \ldots C;\ j = 1 \ldots N$$

$$l_{i,j} = L_j\, x_{i,j} \qquad i = 1 \ldots C;\ j = 0 \ldots N$$

$$v_{i,j} = V_j\, y_{i,j} \qquad i = 1 \ldots C;\ j = 1 \ldots N+1$$

where $f_{i,j}$ is the specified molar flow rate of component i in feed stream to stage j.

Vapor-liquid equilibrium (**E**) equations:

$$y_{i,j} = K_{i,j}\, x_{i,j} \qquad i = 1 \ldots C;\ j = 1 \ldots N$$

Summation (**S**) equations:

$$\Sigma\, x_{i,j} = 1 \qquad j = 1 \ldots N$$

$$\Sigma\, y_{i,j} = 1 \qquad j = 1 \ldots N$$

Heat balance (**H**) equations:

$$Q_j = Q_{j+1} \qquad j = 1 \ldots N$$

Auxiliary equations:

$$H_j = \Sigma\, \lambda_{i,j} y_{i,j} \qquad j = 2 \ldots N+1$$

$$Q_j = V_j\, H_j \qquad j = 2 \ldots N+1$$

$$K_{i,j} = \gamma_{i,j}\, p_{i,j}/P \qquad i = 1 \ldots C;\ j = 1 \ldots N \qquad (5)$$

When computing the linearized form of $K_{i,j}$ according to equation (5) with the mixed AA/IA model, both for equationwise constraint propagation and linear programming-based pruning, all noise variables (basic entities of the AA model, see Stolfi and Figueiredo[25], pp. 43-44) not corresponding to **x** or T are combined into a single noise variable (Stolfi and Figueiredo[25], pp.

81-82). This is obviously a loss of information but it makes the structure of the LP problem simpler and improves the scaling.

Vapor-liquid equilibrium conditions are modeled with a modified Raoult-Dalton equation. Liquid phase activity coefficients are modeled by the 2-parameter Wilson equations

$$\ln \gamma_i = -\ln\left(\sum_{a=1}^{C} x_a \Lambda_{ia}\right) + 1 - \sum_{b=1}^{C} \frac{x_b \Lambda_{bi}}{\sum_{a=1}^{C} x_a \Lambda_{ia}} \qquad i = 1 \dots C \qquad (6)$$

$$\Lambda_{ab} = \frac{V_b^m}{V_a^m} \exp\left(-\frac{k_{ab}}{R_G T}\right) \qquad a = 1 \dots C;\ b = 1 \dots C \qquad (7)$$

where model parameters k_{ab} and V_i^m are given in Table A2 and A3, respectively; R_G is the general (*Regnault*'s) gas constant, given at Table A2; T is the temperature; pure components vapor pressures p_i are computed by the *Antoine* equation

$$\ln p_i = A_i - \frac{B_i}{C_i + T} \qquad (8)$$

with coefficients A_i, B_i, C_i given in Table A4; pressure P at each stage is specified to be 101325 Pa, *i.e.* no pressure drop is taken into account for simplicity.

Note that equations (6)-(8) are not directly involved in the pruning but only the linearized equation (5) is.

Equations related to variables not involved in the pruning

These variables are computed by substitution to the following equations

At stage $j = 0$ (total condenser):

$$x_{i,0} \leftarrow y_{i,1} \qquad i = 1 \dots C$$

$$l_{i,0} \leftarrow RDy_{i,1} \qquad i = 1 \dots C$$

At stage $j = 1$ (upmost equilibrium stage):

$$v_{i,1} \leftarrow (R+1)Dy_{i,1} \qquad i = 1 \dots C$$

$V_1 \leftarrow (R+1)D$

$H_1 \leftarrow \Sigma\, \lambda_i y_{i,1}$

$Q_1 \leftarrow Q_2$

At stage $j = N+1$ (reboiler):

$y_{i,N+1} \leftarrow x_{i,N} \qquad\qquad i = 1 \ldots C$

Numerical examples

Specifications

Components are (1) acetone, (2) methanol and (3) water ($C = 3$). Specifications are $R = 5$, $D = 0.73$ mol/s, solvent (entrainer) feed is 2.0 mol/s pure water, main feed is 0.783 mol/s acetone and 0.217 mol/s methanol. The location of each feed tray is specified. The column has N stages, plus a total condenser and a total reboiler (stage 0 and stage N+1, respectively).

Purity restriction

The purity restriction on the mol fraction of the acetone in the distillate is varied; three cases are discussed: $0.96 \le x_{acetone}$, $0.92 \le x_{acetone}$, $0.78 \le x_{acetone}$. Note that the last restriction is included barely for testing numerically the proposed method; it is too permissive and is not meaningful from engineering aspect because distillate stream significantly richer in acetone than the azeotropic composition is to be produced in this operation.

Preparation of the initial box

The initial intervals of V_j are obtained by a reasonable assumption from engineering aspect: $\mathrm{abs}((V_j - V_1)/V_1) \le 0.32$ where $V_1 = (R+1)D = 4.38$ mol/s as it follows from the specifications. Note that the constant molar overflow assumption would involve $\mathrm{abs}((V_j - V_1)/V_1) = 0.0$.

The initial intervals for all the $K_{i,j}$ values are chosen to be $K_{1,j} \in [0.98, 38.97]$, $K_{2,j} \in [0.80, 7.53]$, and $K_{3,j} \in [0.26, 1.01]$; these properly enclose the range of K in the

$\mathbf{x} \in [0, 1]^C$ space. These intervals of the $K_{i,j}$ values are read from the graph of the corresponding K_i implicit function over $\mathbf{x} \in [0, 1]^C$. These values are then verified by proving the infeasibility of the corresponding system of equations with the proposed method. For example to verify the lower bound 0.98 on K_1, the interval equation $K_1 = [-\infty, 0.98]$ is appended to the system of equations describing the bubble point of the mixture, and then its infeasibility is proven by the proposed branch-and-prune method. Verifying all the bounds similarly totals less than a second. The initial T_j values are [327, 374]; this interval properly encloses all possible bubble points at the specified pressure. All mole fractions are assumed to be not less than 0.01. All other initial intervals are chosen to be non-informative, *i.e.* $[-\infty, +\infty]$.

Results

The results are given in Table 5. Example profiles are shown in Figures 4 and 5. The restriction $0.96 \leq x_{acetone}$ proves to be too strict in case $N = 12$ and $N = 16$, the separation problem is infeasible. This information is provided by the method as a result. The problem becomes feasible with the more permissive but still reasonable $0.92 \leq x_{acetone}$ constraint, and the solution is successfully found. In the practically meaningless case of the $0.78 \leq x_{acetone}$, used here for numerical testing only, the solution cannot be found in 3 hours with the machine used for computation (software and hardware environment is given in the appendix, the used LP solver is GNU GLPK 4.28[45]). Note however that a general splitting rule is applied here, and the problem can be solved in a few minutes with a problem specific splitting rule, as discussed in the next subsection.

Problem specific bisection rule

Splitting the box along its widest component proved to be unsatisfactory in case of the rather loose and practically meaningless restriction $0.78 \le x_{acetone}$. The problem can be solved in minutes by multiplying the mol fraction of the acetone in the distillate by an appropriate weight w when applying the bisection rule. This is shown in Table 6. If the multiplier is too big, *e.g.* 50 or larger in our case, then always the interval being multiplied by w is bisected which is obviously not the ideal solution, hence the slightly worse results. However, the problem is solved in minutes even in that case.

This problem specific rule generally produces worse results for the reasonable purity restrictions, as it is shown in Table 7, but the computation time remains acceptable. Note that the original bisection rule corresponds to the case $w = 1$.

Effect of the enhancements concerning the LP pruning

The Achterberg's heuristic presented at subsection *Linear programming based pruning* is expected to reduce the number of simplex iterations, thus the overall computation time. The numerical examples of Baharev and Rév[41] can be solved in less then a second with the current implementation, as it was presented earlier in Tables 1-3, which makes those examples unsuitable for testing. The *Separation problem* is chosen instead. As shown in Table 8, the Achterberg's heuristic makes the pruning roughly 5 times faster, which is in line with the reduced number of simplex iterations.

Finding multiple solutions

The ability of the proposed method to prove non-existence of solutions is already demonstrated above. The capability of finding multiple solutions is presented here. Since the separation problem has a single solution, the test problem of Meintjes and Morgan[46], chemical equilibrium of hydrocarbon combustion, is computed instead. All the four solutions are found; they are shown in Table 9. This requires 1.46 seconds and 22219 boxes to be examined.

Variables

$x_1 \dots x_5$ all in [-1.0E+1, 1.0E+8],

Constants

$R = 10$,

$R_5 = 0.193$,

$R_6 = 0.002597/\sqrt{40}$,

$R_7 = 0.003448/\sqrt{40}$,

$R_8 = 0.00001799/40$,

$R_9 = 0.0002155/\sqrt{40}$,

$R_{10} = 0.00003846/40$;

Equations

$3x_5 = x_1(x_2 + 1)$,

$x_3(x_2(2x_3+R_7) + 2R_5x_3 + R_6) = 8x_5$,

$x_4(R_9x_2 + 2x_4) = 4Rx_5$,

$x_2(2x_1 + x_3(x_3+R_7) + R_8 + 2R_{10}x_2 + R_9x_4) + x_1 = Rx_5$,

$x_2(x_1 + R_{10}x_2 + x_3(x_3+R_7) + R_8 + R_9x_4) + x_1 + x_3(R_5x_3 + R_6) + x_4^2 = 1$;

Limitations of the current implementation

The proposed method is fairly general, and the numerical results are promising. However, the implementation is still in its infancy. The C++ source code of the affine class consists of approximately 3000 lines although only the bare minimum of the functions are implemented. The C++ code of the distillation column is approximately 2300 lines. The source code was developed solely for experimental purposes, *i.e.* to study the numerical capabilities of the proposed algorithm, thus it is hard to extend or modify. Interfacing the solver with a modeling language would make the work drastically easier. Debugging of the C++ source code is difficult since the authors do not know any mixed affine arithmetic / interval arithmetic implementation that could give correct reference values in case of a suspected bug.

Summary

Generally there is no theoretical guarantee for convergence to the true solution at computing steady states of counter-current multistage processes with conventional methods. The routines are usually sensitive to initial estimates, and if no solution is achieved after several attempts with different initial points then one does not know whether the initial estimation is poor or simply that no solution exists for the specified circumstances. Moreover, there are specifications that give rise to several solutions but standard methods cannot guarantee that all solutions are found.

Interval methods provide tools to overcome these problems: these tools either provide all the solutions or prove nonexistence of solution of a general nonlinear equation system with mathematical certainty. This paper presents significant enhancements compared to a previously published interval method[41] of the authors: both the linearization and the linear programming based pruning step are revised. The effect of each enhancement is demonstrated on the corresponding numerical examples of the previous work, namely the *Liquid phase split* with binary and ternary mixtures, and the *Counter-current equilibrium cascade* with one theoretical stage is re-computed.

The above mentioned improvements make it possible to compute industrial scale distillation columns: MESH equations of columns, number of theoretical stages varying from 12 to 22, hosting continuous extractive distillation of acetone and methanol with water as entrainer are successfully solved. The authors consider this as the main achievement of the paper: to the authors' best knowledge, computation of distillation columns with interval methods has not yet been considered in the literature.

Further numerical examples presented in the paper lead to the following observations. The computation with the simple 'split the widest interval' bisection rule is rather slow if unreasonably loose product purity is specified for the distillation column, but can be shortened to a few minutes by applying a problem specific weight factor in the bisection step. Applicability to proving non-existence of solutions is also demonstrated, in case of specifications that cannot be met. Capability of the method to find multiple solutions is illustrated on a problem of chemical equilibria.

The proposed method is fairly general, and the numerical results are promising. The implementation is, however, still in its infancy, and it is very difficult to code the nonlinear equation systems in C++ programming language. The authors plan to hook the solver to an appropriate modeling language to make the usage of the solver easier.

Acknowledgements

This work was supported by the Hungarian Scientific Research Foundation ("OTKA") K062099. The authors are grateful to Professor Lubomir Varadinov Kolev for reading an earlier draft of this paper and for making valuable comments and suggestions.

References

1. Pratt HRC. *Countercurrent Separation Processes*. Amsterdam, Netherlands: Elsevier; 1967.
2. King CJ. Separation Processes. New York: McGraw-Hill; 1980.
3. 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.
4. Müller D, Marquardt W. Experimental Verification of Multiple Steady States in Heterogeneous Azeotropic Distillation. *Ind Eng Chem Res.* 1997;36:5410–5418.
5. Lin Y, Gwaltney CR, Stadtherr MA. Reliable Modeling and Optimization for Chemical Engineering Applications: Interval Analysis Approach. *Reliable Computing.* 2006;12:427–450.
6. Tessier SR, Brennecke JF, Stadtherr MA. Reliable Phase Stability Analysis for Excess Gibbs Energy Models. *Chem Eng Sci*. 2000;55:1785-1796.
7. Xu G, Haynes WD, Stadtherr MA. Reliable Phase Stability Analysis for Asymmetric Models. *Fluid Phase Equilib.* 2005;235:152-165.
8. Burgos-Solórzano GI, Brennecke JF, Stadtherr MA. Validated Computing Approach for High-Pressure Chemical and Multiphase Equilibrium. *Fluid Phase Equilib*. 2004;219:245-255.
9. Hua JZ, Brennecke JF, Stadtherr MA. Enhanced Interval Analysis for Phase Stability: Cubic Equation of State Models. *Ind Eng Chem Res*. 1998;37:1519-1527.
10. Simoni LD, Lin Y, Brennecke JF, Stadtherr MA. Modeling Liquid-Liquid Equilibrium of Ionic Liquid Systems with NRTL, Electrolyte-NRTL, and UNIQUAC. *Ind Eng Chem Res.* 2008;47: 256-272.

11. Stradi BA, Brennecke JF, Kohn JP, Stadtherr MA. Reliable Computation of Mixture Critical Points. *AIChE J.* 2001;47:212-221.
12. Maier RW, Brennecke JF, Stadtherr MA. Reliable Computation of Homogeneous Azeotropes. *AIChE J.* 1998;44:1745-1755.
13. Gau CY, Stadtherr MA. Deterministic Global Optimization for Error-in-Variables Parameter Estimation. *AIChE J.* 2002;48:1192-1197.
14. Nedialkov NS, Jackson KR, Corliss GF. Validated solutions of initial value problems for ordinary differential equations. *Appl Math Comput.* 1999;105:21–68.
15. Lin Y, Stadtherr MA. Validated Solutions of Initial Value Problems for Parametric ODEs. *Appl Numer Math.* 2007;58:1145–1162.
16. Lin Y, Enszer JA, Stadtherr MA. Enclosing All Solutions of Two-Point Boundary Value Problems for ODEs. *Comput Chem Eng.* 2008;32:1714–1725.
17. Lin Y, Stadtherr MA. Deterministic global optimization of nonlinear dynamic systems. *AIChE J.* 2007;53:866–875.
18. Kearfott, RB. Preconditioners for the Interval Gauss-Seidel Method. *SIAM J Numer Anal.* 1990;27:804–822.
19. Lin Y, Stadtherr MA. LP Strategy for Interval-Newton Method in Deterministic Global Optimization. *Ind Eng Chem Res.* 2004;43:3741–3749.
20. Kearfott RB. Decomposition of Arithmetic Expressions to Improve the Behavior of Interval Iteration for Nonlinear Systems. *Computing* 1991;47(2):169–191.
21. Beelitz T, Frommer A, Lang B, Willems P. Symbolic-Numeric Techniques for Solving Nonlinear Systems. *PAMM.* 2005;5:705–708.
22. Schichl H, Neumaier A. Interval Analysis on Directed Acyclic Graphs for Global Optimization. *J Glob Optim.* 2005;33:541–562.

23. Hammer R, Hocks M, Kulisch U, Ratz D. *C++ Toolbox for Verified Computing I, Basic Numerical Problems.* Berlin: Springer-Verlag; 1995.
24. Comba JLD, Stolfi J. Affine arithmetic and its applications to computer graphics. *Proc. VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'93)*; 1993:9–18.
25. 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.
26. Stolfi J, Figueiredo LH. An Introduction to Affine Arithmetic. *TEMA Tend Mat Apl Comput.* 2003;4:297–312.
27. Figueiredo LH, Stolfi J. Affine Arithmetic: Concepts and Applications. *Numerical Algorithms.* 2004;37:147–158.
28. Kolev LV. A New Method for Global Solution of Systems of Non-Linear Equations. *Reliable Computing.* 1998;4:125–146.
29. Kolev LV. An efficient interval method for global analysis of non-linear resistive circuits. *Int J Circuit Theory Appl.* 1998;26:81–92.
30. 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.
31. Kolev LV. An Improved Method for Global Solution of Non-Linear Systems. *Reliable Computing.* 1999;5:103–111.
32. Kolev LV. An Interval Method for Global Nonlinear Analysis. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications.* 2000;47:675–683.
33. Kolev LV. Automatic Computation of a Linear Interval Enclosure. *Reliable Computing.* 2001;7:17–28.

34. Kolev LV. An improved interval linearization for solving non-linear problems. *Numerical Algorithms.* 2004;37:213–224.
35. Kolev LV. New Formulae for Multiplication of Intervals. *Reliable Computing.* 2006;12:281–292.
36. Kolev LV. Optimal multiplication of G-intervals. *Reliable Computing.* 2007;13(5):399–408.
37. Nenov IP, Fylstra DH. Interval Methods for Accelerated Global Search in the Microsoft Excel Solver. *Reliable Computing.* 2003;9:143–159.
38. Yamamura K, Kumakura T, Inoue Y. Finding All Solutions of Nonlinear Equations Using Inverses of Approximate Jacobian Matrices. *IEICE Trans. Fundamentals.* 2001;E84-A(11):2950–2952.
39. 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.
40. Miyajima S, Kashiwagi M. Existence test for solution of nonlinear systems applying affine arithmetic. *J Comput Appl Math*. 2007;199:304–309.
41. Baharev A, Rév E. Reliable Computation of Equilibrium Cascades with Affine Arithmetic. *AIChE J.* 2008;54(7):1782-1797.
42. Moore RE. Methods and Applications of Interval Analysis. Philadelphia: SIAM; 1979.
43. Hansen ER. Global Optimization Using Interval Analysis. New York: Marcel Dekker; 1992.
44. Meyers S. Effective C++: 50 Specific Ways to Improve Your Programs and Design (2nd Edition), Addison-Wesley Professional Computing Series, 1997.
45. GNU Linear Programming Kit, http://www.gnu.org/software/glpk

46. Meintjes K, Morgan AP. Chemical equilibrium systems as numerical test problems. *ACM TOMS*. 1990;16(2):143—151.

Appendix

Table A1

Heat of vaporization at normal boiling point. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.). Computations were carried out with the values in cal/mol.

	λ_i	
Component *i*	(cal/mol)	(kJ/mol)
Acetone	6960	29.12
Methanol	8426	35.25
Water	9717	40.66

Table A2

Parameters of the Wilson equation. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.) R_G is the general (*Regnault*'s) gas constant 8.314472 J/(mol K). Computations were carried out with the values in cal/mol and with R_G = 1.98721 cal/(mol K). Components: (1) acetone, (2) methanol, (3) water

i	j	k_{ij} (cal/mol)	k_{ji} (cal/mol)	k_{ij} / R_G (K)	k_{ji} / R_G (K)
1	2	-157.981	592.638	-79.4989	298.226
1	3	393.27	1430.0	197.90	719.60
2	3	-52.605	620.63	-26.472	312.31

Table A3

Liquid molar volume. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.)

Component i	V_i^m (cm^3/mol)
Acetone	74.05
Methanol	40.729
Water	18.069

Table A4

Parameters of the Antoine equation $\ln p = A - B/(T + C)$ where vapor pressure p is in mmHg, temperature T is in degrees Kelvin. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.) Computations were carried out with the *A*, *B*, *C* values; to get the vapor pressure in Pa use A^* instead of *A*.

Component i	A_i (for mmHg)	B_i (K)	C_i (K)	A_i^* (for Pa)
Acetone	16.732	2975.9	-34.523	21.625
Methanol	18.51	3593.4	-35.225	23.40
Water	18.304	3816.4	-46.13	23.197

Software and hardware environment

The computations are carried out with the following hardware and software configuration. Processor: Intel Pentium 4 530 Prescott at 3.00 GHz, L1 cache 16 KB, L2 cache 1024 KB; memory: 2×512 MB PC3200 DDR RAM 400 MHz (dual channel interleaved), bus speed 800 MHz; chipset: Intel i915P; operating system: Kubuntu 5.04 (in text mode) with Ubuntu kernel 2.6.10-5-686-smp; compiler: Intel C++ Compiler for Linux 8.1, compiler flags: -O2 -ip -static –xP.

List of Figure Captions

Figure 1. Illustration of the linear enclosure of function x^2 computed by (*a*) ordinary interval arithmetic (IA), (*b*) affine arithmetic (AA) and min-range approximation, (*c*) AA and Chebyshev approximation, (d) mixed AA/IA model with Chebyshev approximation.

The slope of the dashed line computed by AA correlates well with the slope of the approximated function.

Figure 2. Equilibrium stage

Figure 3. Continuous extractive distillation

Figure 4. Composition profile

Figure 5. Temperature profile

Tables

Table 1. Comparison of the previous and current implementation (LLE phase split, binary mixture)

	implementation		
	previous[41]	current	$\frac{\text{previous}}{\text{current}}$
time (s)	1.15	0.010	115
iterations	1407	627	2.24
cycle time (μs)	817	15.9	51.3

Table 2. Comparison of the previous and current implementation (LLE phase split, ternary mixture)

	implementation		
	previous[41]	current	$\frac{\text{previous}}{\text{current}}$
time (s)	23.3	0.790	29.5
iterations	7715	6513	1.18
cycle time (μs)	3010	121	24.9

Table 3. Comparison of the previous and current implementation (VLE cascade with 3 stages)

	implementation		
	previous[41]	current	$\frac{\text{previous}}{\text{current}}$
time (s)	4.44	0.099	44.8
iterations	1687	645	2.62
cycle time (μs)	2630	153	17.1

Table 4. List of variables. The domain of *i* is 1 … *C* in all cases. Stage $j = 0$ is the condenser, stage $j = N+1$ is the reboiler.

		involved in pruning	not involved in pruning
$x_{i,j}$	mole fraction of component *i* in the liquid phase at stage *j*	$j = 1 \ldots N$	$j = 0$
$y_{i,j}$	mole fraction of component *i* in the vapor phase at stage *j*	$j = 1 \ldots N$	$j = N+1$
$K_{i,j}$	equilibrium ratio	$j = 1 \ldots N$	
$l_{i,j}$	molar flow rate of component *i* in the liquid phase at stage *j*	$j = 1 \ldots N$	$j = 0$
$v_{i,j}$	molar flow rate of component *i* in the vapor phase at stage *j*	$j = 2 \ldots N+1$	$j = 1$
V_j	vapor flow rate at stage *j*	$j = 2 \ldots N+1$	$j = 1$
H_j	molar enthalpy of vapor at stage *j*	$j = 2 \ldots N+1$	$j = 1$
Q_j	enthalpy rate carried by the vapor at stage *j*	$j = 2 \ldots N+1$	$j = 1$
T_j	temperature at stage *j*	$j = 1 \ldots N$	

Table 5. Computational results for the extractive distillation column with different specifications and restrictions. General splitting rule is applied.

number of trays	purity restriction on distillate	distillate composition	time (s)	number of boxes	simplex iterations
12[1]	$0.96 \leq x_{acetone}$	infeasible	2.77	3	23891
12	$0.92 \leq x_{acetone}$	[0.923, 0.0430, 0.0342]	22.10	19	247522
12	$0.78 \leq x_{acetone}$	time limit reached	>12000	—	—
16[2]	$0.96 \leq x_{acetone}$	infeasible	9.49	9	78803
16	$0.92 \leq x_{acetone}$	[0.942, 0.0343, 0.0234]	54.15	29	500041
16	$0.78 \leq x_{acetone}$	time limit reached	>12000	—	—
22[3]	$0.96 \leq x_{acetone}$	[0.961, 0.0212, 0.0179]	52.86	15	459290
22	$0.92 \leq x_{acetone}$	[0.961, 0.0212, 0.0179]	92.13	33	709058
22	$0.78 \leq x_{acetone}$	time limit reached	>12000	—	—

(1) Solvent feed to tray 5, main feed to tray 9.
(2) Solvent feed to tray 7, main feed to tray 12.
(3) Solvent feed to tray 9, main feed to tray 16.

Table 6. Effect of changing the multiplier *w* used in the bisection step. Specifications: number of trays: 16, solvent feed to tray 7, main feed to tray 12; purity restriction on the distillate: $0.78 \leq x_{acetone}$

multiplier *w*	time (s)	number of boxes	simplex iterations
1	>12000	—	—
10	>12000	—	—
20	157.5	99	1424169
≥50	187.8	113	1787940

Table 7. Comparison of the computational results using multiplier $w = 1$ and $w = 20$ in the bisection step.

number of trays	purity restriction on distillate	time (s) ($w = 1$)	time (s) ($w = 20$)	number of boxes ($w = 1$)	number of boxes ($w = 20$)	simplex iterations ($w = 1$)	simplex iterations ($w = 20$)	$\frac{\text{time}_{w=1}}{\text{time}_{w=20}}$	$\frac{\text{simp. iter}_{w=1}}{\text{simp. iter}_{w=20}}$
12[1]	$0.96 \le x_{acetone}$	2.77	4.19	3	7	23891	40601	0.66	0.59
12	$0.92 \le x_{acetone}$	22.10	17.37	19	17	247522	200165	1.27	1.24
12	$0.78 \le x_{acetone}$	>12000	67.50	—	67	—	759990	>177.78	—
16[2]	$0.96 \le x_{acetone}$	9.49	16.48	9	13	78803	152683	0.58	0.52
16	$0.92 \le x_{acetone}$	54.15	56.51	29	31	500041	577726	0.96	0.87
16	$0.78 \le x_{acetone}$	>12000	157.5	—	99	—	1424169	>76.19	—
22[3]	$0.96 \le x_{acetone}$	52.86	210.94	15	55	459290	1934330	0.25	0.24
22	$0.92 \le x_{acetone}$	92.13	230.26	33	61	709058	2029564	0.40	0.35
22	$0.78 \le x_{acetone}$	>12000	498.03	—	147	—	3968612	>24.09	—

(1) Solvent feed to tray 5, main feed to tray 9.

(2) Solvent feed to tray 7, main feed to tray 12.

(3) Solvent feed to tray 9, main feed to tray 16.

Table 8. Comparison of the computational results obtained with / without using Achterberg's heuristic. Number of trays is 16.

	without heur.	with heur.	$\frac{\text{without heur.}}{\text{with heur.}}$
$0.96 \leq x_{acetone}$ and $w = 1$			
time (s)	44.0	9.49	4.63
simplex iterations	505424	78803	6.41
$0.92 \leq x_{acetone}$ and $w = 1$			
time (s)	301.43	54.15	5.57
simplex iterations	3376560	500041	6.75
$0.78 \leq x_{acetone}$ and $w = 20$			
time (s)	769.56	157.5	4.89
simplex iterations	9102883	1424169	6.39

Table 9. All real solutions of the chemical equilibrium of hydrocarbon combustion problem in the given search box.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	3.114E-3	2.757E-3	2.471E-3	2.153E-3
x_2	3.460E+1	3.924E+1	4.388E+1	5.055E+1
x_3	6.504E-2	-6.139E-2	5.778E-2	-5.414E-2
x_4	8.594E-1	8.597E-1	-8.602E-1	-8.607E-1
x_5	3.695E-2	3.699E-2	3.697E-2	3.700E-2

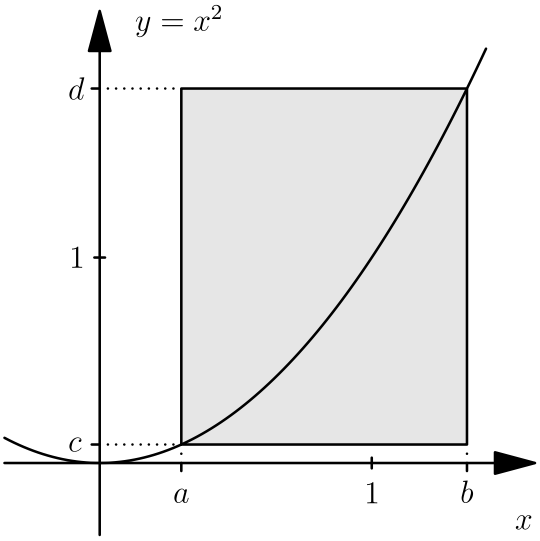


Fig 1a.

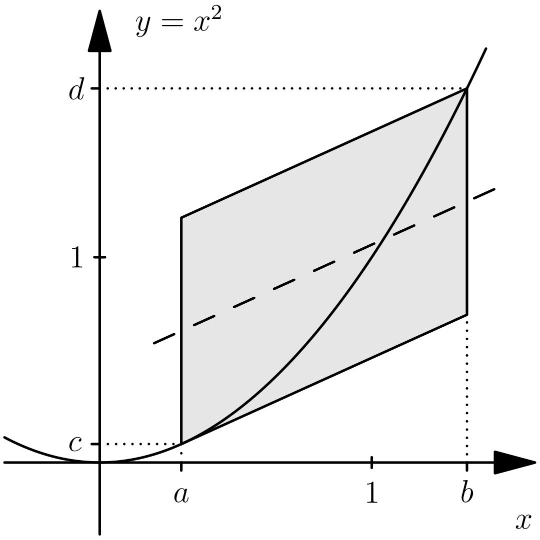


Fig. 1b.

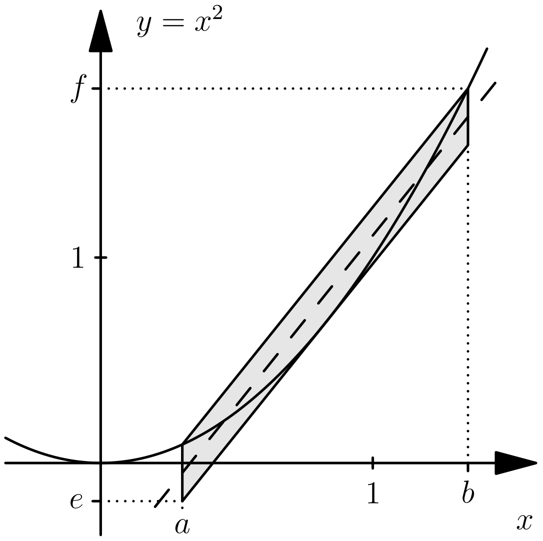


Fig. 1c.

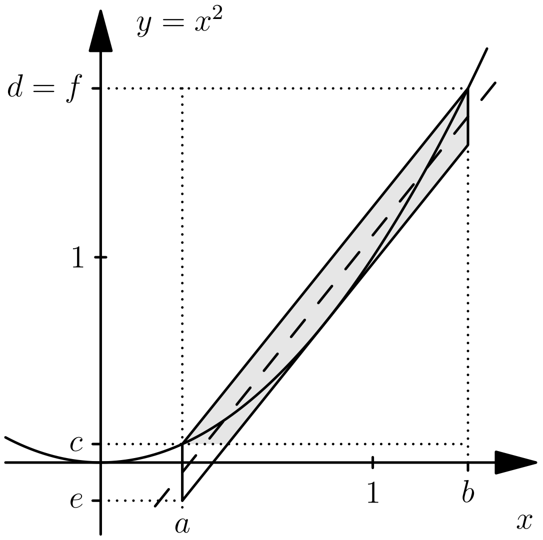


Fig. 1d.

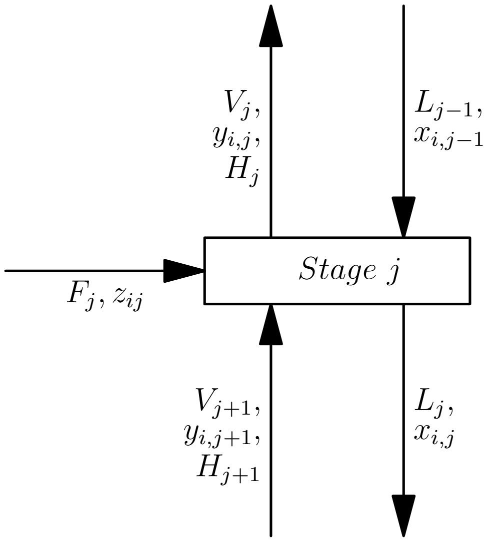


Fig. 2.

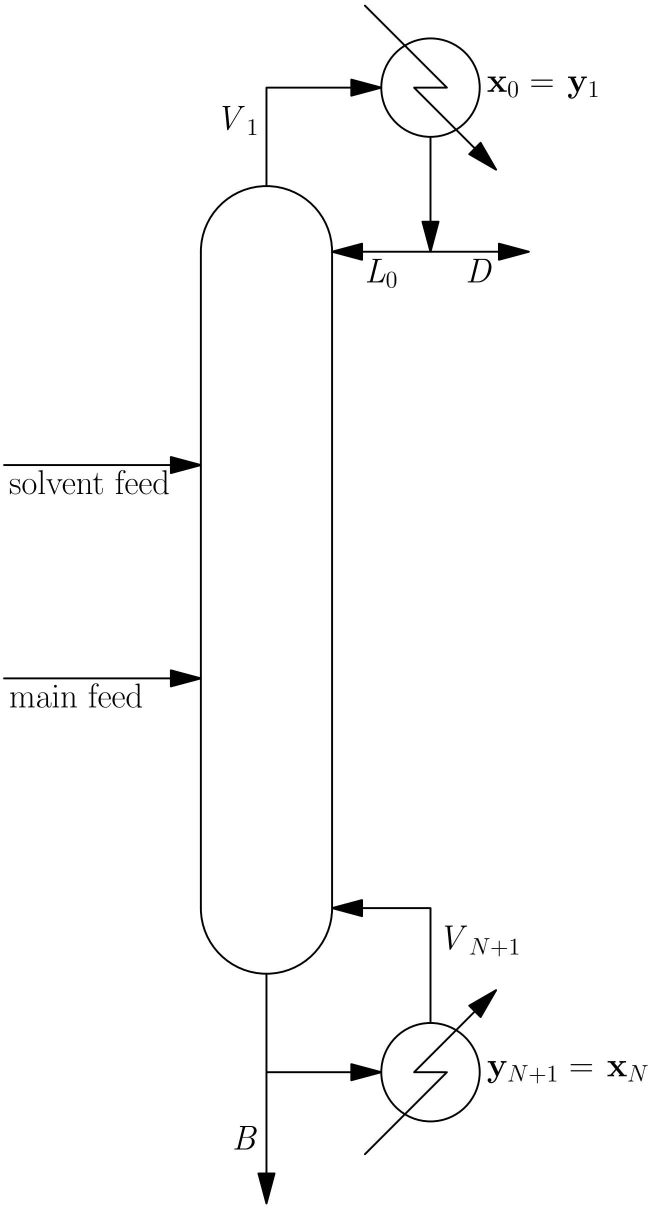

$\mathbf{x}_0 = \mathbf{y}_1$
V_1
L_0
D
solvent feed
main feed
V_{N+1}
$\mathbf{y}_{N+1} = \mathbf{x}_N$
B

Fig. 3.

Fig. 4.

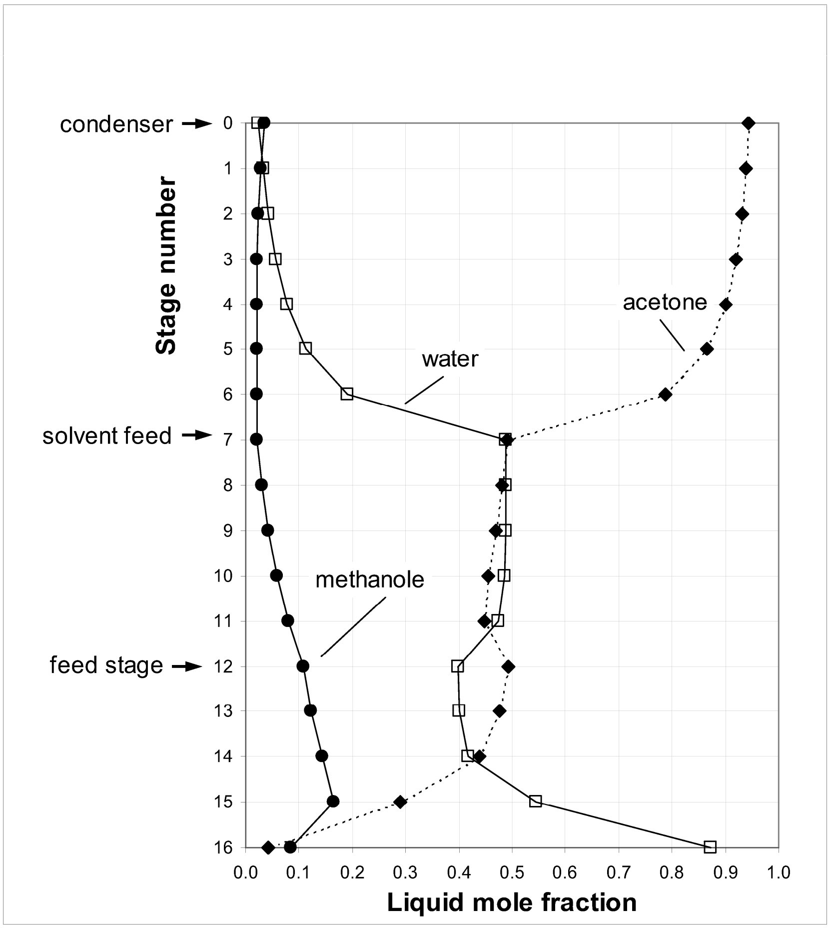
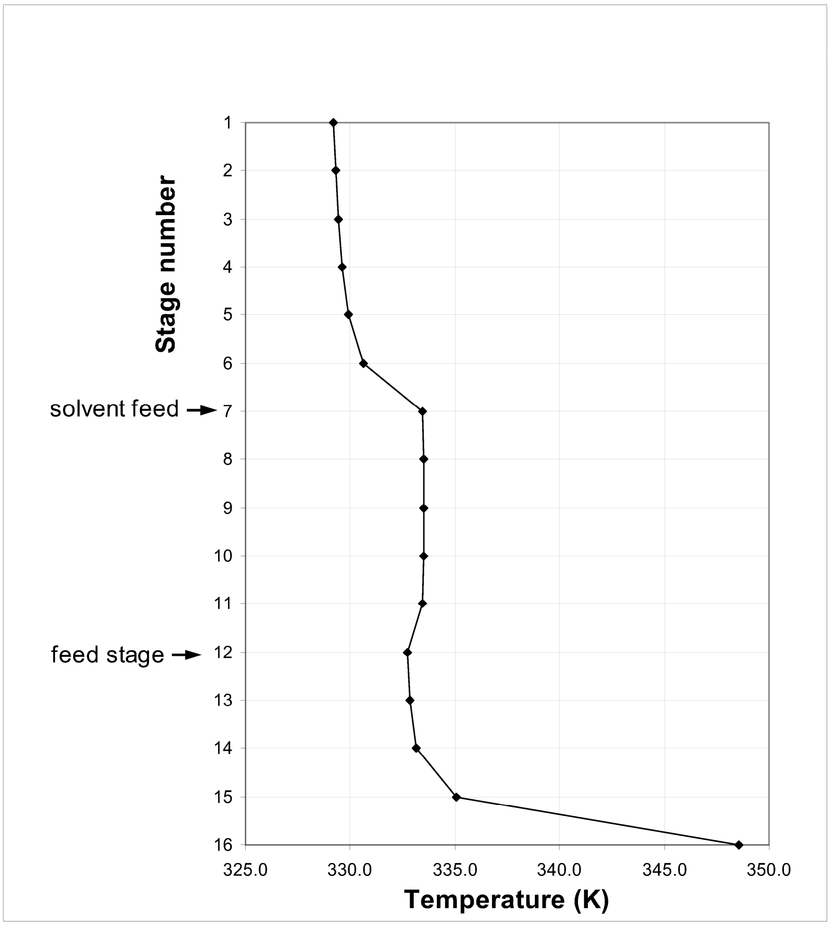

condenser
solvent feed
feed stage
Stage number
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
acetone
water
methanole
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Liquid mole fraction

Fig. 5.



Stage number
1
2
3
4
5
6
solvent feed → 7
8
9
10
11
feed stage → 12
13
14
15
16
325.0
330.0
335.0
340.0
345.0
350.0
Temperature (K)