Reliable Computation of Equilibrium Cascades with Affine Arithmetic

Ali Baharev* and Endre Rév

Budapest Universty of Technology and Economics

Department of Chemical and Environmental Process Engineering

Muegyetem rkp. 3., K. Mf. 56, 1111 Budapest, Hungary

Fax: +36 (1) 463-3197

*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, Volume 54, Issue 7, Pages 1782-1797, July 2008

Online ISSN: 1547-5905

Print ISSN: 0001-1541

http://www.interscience.wiley.com

Published Online: Apr 30 2008 2:05PM

DOI: 10.1002/aic.11490

Abstract

Computing the steady state of multistage counter-current processes like distillation, extraction, or absorption is the equivalent to finding solutions for large scale non-linear equation systems. The conventional solution techniques are fast and efficient if a good estimation is available but are prone to fail, and do not provide information about the reason for the failure. This is the main motive to apply reliable methods in solving them.

Reliable computations are usually realized with interval methods. This paper presents a reliable root finding method based on affine arithmetic (AA), a recently developed linearization technique and interval method. AA is compared here to another linearization technique, the widely known Interval Newton method. The studied examples seem to indicate superiority of the novel method over the traditional one. The comparison is made with a pruning technique not *state-of-the-art* but analogous in the two compared methods.

AA can be combined with constraint propagation (CP) or linear programming (LP) aided CP, as pruning techniques. The combined techniques, AA/CP and AA/LP are studied and compared. AA/LP proves to be preferable because of its robustness.

Short distillation columns are successfully computed with the proposed AA/LP method.

Topical heading: Separations

Keywords: affine arithmetic, interval arithmetic, reliable computation, root finding, MESH equations

Introduction

Design and simulation of phase equilibrium based on separation of chemical components from liquid or gas mixtures such as extraction, absorption, desorption, stripping, and distillation, constitute an important part of the everyday practice in chemical engineering. Computing the steady states of the 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 reliable computation techniques, and powerful results have been achieved[1,2], there is no theoretical guarantee for convergence to the true solution.

Finding the correct number and composition of equilibrium phases at given pressure and temperature when only the gross composition is known is a basic problem in phase equilibrium calculations as it involves numerical difficulties. Even in the case of liquid / liquid phase separation of a binary mixture, the so-called trivial solution is sometimes found instead of the correct one because both solutions satisfy the equality of partial fugacities, a necessary condition of equilibrium. In this case one does not know if the homogeneous mixture is thermodynamically stable or there is phase separation not found numerically. Minimizing the molar Gibbs energy of the system, *i.e.* satisfying the necessary and satisfactory conditions of equilibrium, is equivalent to finding global minimum. Finding the global minimum cannot be guaranteed by local methods; there may be several local optima due to the applied sophisticated thermodynamic nonlinear models. Although local methods work well if a good estimate is available for each mole fraction in the conjugated phases and if the gross composition is far from any plate point (*e.g.* Prausnitz *et al.*[3], p. 128), they do not always work, and do not guarantee correctness. In order to reliably determine the number and

composition of phases, one should find the global minimizer of the molar Gibbs energy function using a proper global optimizing tool[4-13].

Computing steady states of counter-current cascades of phase equilibrium units (so-called equilibrium stages) is a well-developed discipline in chemical engineering[1,2]. However, the numerical problems in this field have not yet been perfectly solved. The routines developed for computing steady state 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 bad or simply that no solution exists for the specified circumstances. Moreover, there are specifications that give rise to several solutions (output multiplicity[14,15]).

The above chemical engineering problems could be overcome by a methodology that reliably explores all the solutions of the modeling nonlinear equation system, or reliably concludes to the non-existence of a solution if it really does not exist. Standard methods neither guarantee that all solutions have been found nor prove that no solution exists. Until now, the only strategy that provides all the solutions of the general form system with mathematical certainty seems to be the interval methods. The Interval Newton / Generalized Bisection method (IN/GB) has been successfully applied to solve a wide variety of problems[16] such as computation of phase stability with activity coefficient models, cubic equation-of-state (EOS) models and statistical associating fluid theory, calculation of critical points from cubic EOS models, location of azeotropes and reactive azeotropes, parameter estimation using standard least squares and error-in-variables. Ordinary interval arithmetic can also be applied to compute validated solutions of initial value problems for ODEs[17,18].

Interval root finding methods based on the traditional and widely used linearization technique in the form

$$\mathbf{f(x)} \in \boldsymbol{A}(\mathbf{x}-\mathbf{z}) + \mathbf{f(z)}$$

(in a given interval box $\boldsymbol{X}$, where $\mathbf{x} \in \boldsymbol{X}$ and $\mathbf{z} \in \boldsymbol{X}$ are real numbers, $\boldsymbol{A}(\boldsymbol{X})$ is an interval matrix, and $\mathbf{f(z)}$ is a real vector) have improved considerably during the past few decades. *State-of-the-art* variants, involving advanced preconditioning[19], linear programming[20] and / or directed acyclic graph (DAG) based constraint propagation techniques[21-23], may be several orders of magnitude faster than the "textbook" Interval Newton / Gauss-Seidel[24] (IN/GS) algorithm with the so-called midpoint inverse preconditioner. It is interesting to note that the idea of the DAG based constraint propagation technique, which is one of the most effective pruning techniques, seems to be first described very early in the independent paper of Kearfott[21].

Roughly speaking, the total time of the interval computation depends, apart from the problem itself to be solved, on three main factors: (1) the technique applied for successively linearizing the functions over the actual box, (2) the way the box is pruned, based on this linearization, and (3) the way the box is subdivided in case the pruning is unsatisfactory. These factors are not independent, of course. One may introduce, for example, specific enhancements of the linearization technique in order to achieve better pruning. Expansive abbreviations are usually applied for distinguishing combinations of the techniques according to the above three factors. For example, a method may be abbreviated as IN/GS/GB, using (1) interval Newton method (IN) as linearization technique, (2) Gauss-Seidel iteration (GS) as a technique for pruning the box, and (3) an appropriate form of Generalized Bisection (GB) as a technique for subdivision. As subdivision strategies are not studied in this paper, a splitting rule based on the idea of Kearfott and Novoa[25] is used in all methods, and the /GB part is dropped from the abbreviations for simplicity.

A new linearization technique, based on affine arithmetic (AA[26-29]), has been proposed recently by Kolev[30-38]. Numerical evidence published in the literature[30-34,36,39-42] suggests that

the new technique is superior to the traditional linearization techniques such as the interval Newton or the Krawczyk[43] method.

The purpose of the current research is to compare the traditional and the new AA-based linearization techniques, and compare different pruning methods applicable with AA, using chemical engineering problems of real complexity. Not the numerical values of the solutions of the studied examples are significant here but the time and number of iterations (cycles) to compute those solutions. The goal of the current implementation is to identify the bottlenecks and to get a better understanding of affine arithmetic, and in this way to construct a good starting point for future development and implementation. For this reason, the example problems in this paper are selected in such a way that they can be solved with professional flowsheet simulators without any difficulty.

Efficiency of the new AA-based linearization technique is compared to the traditional Interval Newton technique with Gauss Seidel iteration[24] (IN/GS). Neither *state-of-the-art* constraint propagation (CP) nor problem specific enhancement is used in the AA/CP method. The major difference between AA/CP and IN/GS lies in their linearization technique (AA *vs.* IN). The applied propagation techniques, namely Procedure 3 below and interval Gauss Seidel iteration are analogous (comparable). The authors think that the applied rules for bisection are also analogous and, as a consequence, AA/CP and IN/GS are comparable in a fair way.

AA can be combined with different techniques for pruning the search intervals. *State-of-the-art* CP techniques are not yet implemented although they are in principle applicable to AA. Instead, an easy to implement constraint propagation technique (AA/CP) and the same technique aided with linear programming (AA/LP) are studied and compared.

The proposed AA/LP method is applicable to solving small scale counter-current cascade problems that have not yet been considered at all in the literature of applying interval arithmetic to chemical engineering problems, according to the authors best knowledge,

perhaps because of their extensive complexity and dimensionality. Based on the hitherto achieved results, the authors expect to solve larger scale counter-current processes with this interval technique in tolerable running time.

Mathematical background

Overestimation in ordinary interval arithmetic

Ordinary Interval Arithmetic[43-48] (IA) provides interval inclusion of the range of real functions over given intervals. When implemented on floating point machines, interval arithmetic is appended with extra rules of rounding outwards in order to prevent losing any element of the proper range. In this way, one can compute rigorous enclosure.

The ranges are usually overestimated by IA. For example, $X - X = [\underline{X} - \overline{X},\ \overline{X} - \underline{X}] \neq 0$ if $\underline{X} \neq \overline{X}$ where $\underline{X}$ and $\overline{X}$ denote the lower and upper endpoint of interval X, respectively. For the range of $f(x) = (x-1)/(x^2+2)$ over interval $[2, 4]$, IA yields $[5/9,\ 1/2] \subseteq [0.055555,\ 0.500000]$; this is approximately 27 times wider than the exact range $[1/6,\ (\sqrt{3}-1)/4] \subseteq [0.166666,\ 0.183013]$. This overestimation is mainly a consequence of neglecting the correlation between the partial results of the computation. For example, $y = x-1$ and $z = x^2+2$ are not independent in the above example; however, they are divided as ratio of intervals of independent variables *y* and *z*.

This phenomenon of neglected correlation is called the "dependency problem" in the literature of IA. One of the main issues with IA is the dependency problem, and affine arithmetic is a method to cope with this.

Notions of affine arithmetic

The aim of this subsection is to give the notations and nomenclature used in this paper. For an introduction and exhaustive overview of affine arithmetic, the reader is referred to the excellent monograph of Stolfi and Figueiredo[27].

Overestimation by IA is mainly due to its inability to keep track of the correlation between the computed partial results. A possible way to overcome this problem is the use of affine arithmetic[26-29] (AA) that operates on linear polynomials of the form

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \ldots + x_n\varepsilon_n ,$$

where the coefficients x_i $(i = 0, \ldots, n)$ are real numbers, and ε_i $(i = 1, \ldots, n)$ denote symbolic real variables, called noise variables. The values of the noise variables are unknown but must lie in the interval $[-1, 1]$.

Apart from the rounding error, the result of the affine operation $\alpha x + \beta y + \gamma$ is exactly computed as

$$\alpha x + \beta y + \gamma = (\alpha x_0 + \beta y_0 + \gamma) + \sum_{i=1}^{n} (\alpha x_i + \beta y_i)\varepsilon_i .$$

To any non-affine operation $f^*(\varepsilon_1, \ldots, \varepsilon_n)$, an affine function

$$f^a(\varepsilon_1, \ldots, \varepsilon_n) = z_0 + z_1\varepsilon_1 + \ldots + z_n\varepsilon_n$$

that approximates $f^*(\varepsilon_1, \ldots, \varepsilon_n)$ is carefully selected and a new term $z_k\varepsilon_k$ is introduced to bound the absolute magnitude of the error of the approximation

$$|z_k| \geq \sup\{|f^*(\varepsilon_1, \ldots, \varepsilon_n) - f^a(\varepsilon_1, \ldots, \varepsilon_n)|\} . \qquad (1)$$

As there are $n+1$ degrees of freedom (z_0 to z_n) in choosing the approximating affine function f^a, it is possible to aim for approximations that are optimal in some sense. An optimal affine approximation is the Chebyshev or minimax approximation that minimizes the maximum absolute error (1). Another approximation is the min-range approximation where

the result has the minimal range if converted to ordinary interval. Chebyshev and min-range approximations of basic non-affine operations are given in Stolfi and Figueiredo[27].

The mixed AA/IA model

The Chebysev approximation preserves more information on the dependency of the computed results than the min-range approximation but has the drawback of overestimating the range, whereas the min-range approximation has no overshoot or undershoot (Stolfi and Figueiredo[27], pages 63 and 68). Overestimation of the range of intermediate results can cause problems such as division by zero or negative argument to the logarithmic function *etc.* that would not be experienced even if using ordinary interval arithmetic (see also Example 3 below). These problems can be avoided by using min-range approximation at the crucial points or using a mixed AA/IA model as proposed by Stolfi and Figueiredo[27] (pp. 75-76).

In the mixed AA/IA model the IA part is responsible for providing tight ranges of the individual variables, and the AA part is optimized to keep track correlations between computed quantities without having to worry about problems caused by over- or undershoot.

The IA, AA min-range, AA Chebysev, and mixed AA/IA, linear enclosures of function x^2 are illustrated in Figure 1. As is shown in Figure 1d (mixed AA/IA model) the range of the affine form should be determined as the intersection of range $[c, d]$ computed by IA and range $[e, f]$ computed by AA when it is needed, *e.g.* for a non-affine operation.

Computing a non-affine function or operation usually starts with converting its affine argument to conventional interval. The range is worth computing both with AA and IA in each non-affine operation, *i.e.* in each non-affine elementary step of the computation. For illustration, see Example 1 below. During the evaluation of expressions, the IA part can store

information (tighter range) obtained by constraint propagation; see Example 2 and 3 for instance.

Example 1

Evaluation of $(x-1)/(x^2+2)$ over $[2, 4]$ with IA yields $[5/9, 1/2] \subseteq [0.055555, 0.500000]$ as discussed above. Evaluating this function with AA over the same domain, using min-range approximation to x^2 and division according to Kolev[36], gives $[0.055555,\ 0.271605]$. Despite the fact that the range of x^2 over domain $[2, 4]$ is overestimated by the Chebysev approximation as to $[3, 16]$ instead of $[4, 16]$, the Chebysev approximation gives a range more than 4 times tighter than the min-range approximation because it has more information on the correlation between $(x-1)$ and (x^2+2). The best result is obtained by the Chebysev approximation and using the true range of the denominator computed by IA when performing the division. The results are summarized in Table 1.

Computing bounds on mole fraction weighted averages for the mixed AA/IA model

In certain practical applications, especially in chemical engineering, computation of weighted sums $\sum_i w_i x_i$ and their reciprocal $1/\sum_i w_i x_i$ are often required, where $\sum_i x_i = 1$, $0 < x_i < 1$, and $w_i > 0$, hold. The ranges of these sums are obviously overestimated if constraint $\sum_i x_i = 1$ is neglected. The most popular activity coefficient models like Wilson[49], NRTL[50], UNIQUAC[51], and UNIFAC[52], involve division with weighted sums of mole fractions. Such summation of weighted mole fractions easily result in intervals containing zero which, in turn,

gives rise to an exception of division by zero. The situation becomes even more problematic if the weights are not *a priori* specified constants but nonlinear functions $w_i = p_i(T)$ of some variable T, and the sums are evaluated in themselves irrespective of the condition $\sum_i x_i = 1$.

Efficiently computing tight bounds on these sums not only solves this problem but may also significantly speed up the computation. However, the tighter bounds can be used by the mixed AA/IA model only, and not by the pure AA model.

Exact bounds (within round out) on the sum $\sum_i w_i x_i$ can be computed with Procedure 1 below, an O(n) algorithm, if the weights are *a priori* specified constant real numbers. Procedure 1 is the affine analogue of the IA algorithm published by Hua, Brennecke, and Stadtherr[10]. Based on analyzing the corresponding LP problem it is shown by them that at most one variable is not at its lower or upper bound if $\sum_i w_i x_i$ reaches its extreme. In Step 2 of Procedure 1 the weighted sum is computed in the following way. The constraint is satisfied through the computation of y_j; at most one variable is not at its lower or upper bound because z_j is independent from x_j, and all the possible cases are considered. Thus, it follows from the results of Hua, Brennecke, and Stadtherr[10] that Procedure 1 gives the exact bounds on the sum. Note that this procedure would fail if performed in IA because IA cannot track the dependence between $w_j y_j$ and $\sum_{\substack{i=1 \\ i \neq j}}^{n} w_i x_i$ occurring in the sum for z_j in Step 2, and thus it overestimates the range of z_j. In contrast, AA gives the true range of z_j because only affine operations are performed.

In Procedure 1, affine form a stores the result of the affine operation $\sum_{i=1}^{n} w_i x_i$; this a can be used later by pruning methods discussed below in the section *Linearization and pruning*

techniques; the tightest bounds on the sum $\sum_{i=1}^{n} w_i x_i$ are given by the ordinary interval r; the function named AA_to_IA converts its affine argument to ordinary interval (Stolfi and Figueiredo[27], pp. 48); x_i, y_i, z_i are affine forms $(i = 1, 2, ..., n)$; for the independent x_i variables $\sum_i x_i = 1$ and $0 < x_i < 1$ hold; back arrow "$\leftarrow$" denotes assignment; "$\cap$" denotes intersection.

Procedure 1

Step 1

$$a \leftarrow \sum_{i=1}^{n} w_i x_i$$

$$r \leftarrow AA\_to\_IA(a)$$

Step 2

For $j = 1$ to n repeat

$$y_j \leftarrow 1 - \sum_{\substack{i \neq j \\ i=1}}^{n} x_i$$

$$z_j \leftarrow w_j y_j + \sum_{\substack{i \neq j \\ i=1}}^{n} w_i x_i$$

$$r \leftarrow r \cap AA\_to\_IA(z_j)$$

Step 3

Finished, tightest bounds are given by r.

Example 2

This example is taken from Hua, Brennecke, and Stadtherr[10]. Compute the tightest possible range for $x_1 + 2x_2 + 3x_3$ where $x_1 + x_2 + x_3 = 1$, $x_1 \in [0.1, 0.4]$, $x_2 \in [0.2, 0.4]$, and $x_3 \in [0.3, 0.5]$. The resulted interval is $[0.1+2\cdot 0.2+3\cdot 0.3,\ 0.4+2\cdot 0.4+3\cdot 0.5] = [1.4, 2.7]$ if condition $x_1 + x_2 + x_3 = 1$ is ignored and the operations are performed in IA. Procedure 1 yields the true range $[1.9,\ 2.4]$, identical to the result of Hua, Brennecke, and Stadtherr.

Tightening bounds on nonlinear functions for the mixed AA/IA model

Procedure 1 can be generalized, and bounds on sophisticated nonlinear functions can also be tightened, by taking into account fairly general constraints. In Procedure 2 below, tight bounds on a non-linear function $f(x_1,\ x_2,\ \ldots,\ x_n)$ are to be determined by taking into account constraint $g(x_1,\ x_2,\ldots,\ x_n) = 0$. Suppose it is possible to express each x_j $(j = 1,\ 2,\ \ldots,\ n)$ from constraint $g(x_1,\ x_2,\ldots,\ x_n) = 0$ in the form $x_j = g_j(x_1,\ x_2,\ldots,\ x_n)$. If it is not possible for some index j then the *j*-th cycle in Step 2 can be skipped in Procedure 2. Note that AA evaluation of $g(x_1,\ x_2,\ldots,\ x_n) = 0$ gives a linear relaxation of the constraint from which it is always possible to express x_j.

Variables $x_i,\ y_i,\ z_i$ $(i = 1, 2, \ldots, n)$ in Procedure 2 are affine forms; for other symbols, see notation of Procedure 1.

Procedure 2

Step 1

$$a \leftarrow f(x_1,\ x_2,\ \ldots,\ x_j,\ \ldots,\ x_n)$$

$r \leftarrow AA\_to\_IA(a)$

Step 2

For $j = 1$ to n repeat

$y_j \leftarrow g_j(x_1, x_2, ..., x_n)$

$z_j \leftarrow f(x_1, x_2, \ldots, y_j, \ldots, x_n)$

$r \leftarrow r \cap AA\_to\_IA(z_j)$

Step 3

Finished, result is r.

Although this procedure can be used to tighten the bounds, the bounds obtained may not compensate the cost of computing function f $n+1$ times with AA.

Example 3

Bounds on sub-expression $\sum_{a=1}^{C} x_a \Lambda_{ia}(T)$ of the Wilson equations (17) and (18) are considered as an example. Let $C = 3$ (number of components), $\Lambda_{ia}(T) = V_a^m / V_i^m \cdot \exp(-k_{ia}/(R_G T))$ where $a, i \in \{1,2,3\}$; model parameters k_{ia} and V_i^m are given in Table A6 and A7 respectively; R_G denotes the (*Regnault*) ideal gas constant (1.98721 cal/(mol·K)); T denotes the temperature in Kelvin. The nonlinear function f of Procedure 2 with substitution $x_4 \equiv T$ is

$$f(x_1, x_2, x_3, x_4) \equiv \sum_{a=1}^{3} x_a \Lambda_{ia}(x_4)$$

for a fixed $i \in \{1,2,3\}$, and constraint g is

$$x_1 + x_2 + x_3 - 1 = 0.$$

The ranges of the variables are

$x_i \in [0, 1]$ for $i = 1, 2, 3$,

$T \in [300, 380]$ (Kelvin).

Results are summarized in Table 2. The range of $f(x_1, x_2, x_3, x_4)$ is also shown as computed with IA and AA when constraint $x_1 + x_2 + x_3 - 1 = 0$ is ignored during the evaluation. The ranges obtained by AA are even worse than by IA because zero is contained properly in all 3 results for AA due to the undershoot problems. The main reduction of the overestimation is reached by considering the constraint. Note that Procedure 1 is also applicable when the weights are not pre-specified real constants but (non-degenerate) intervals.

Linearization and pruning techniques

Given the system of n nonlinear equations

$$\begin{aligned} &f_1(x_1, x_2, ..., x_n) = 0, \\ &f_2(x_1, x_2, ..., x_n) = 0, \\ &\vdots \qquad\qquad \vdots \\ &f_n(x_1, x_2, ..., x_n) = 0, \end{aligned} \tag{2}$$

where f_j $(j = 1, 2, ..., n)$ are real functions, the goal is to bound the solution(s) contained in a rectangular box, *i.e.* an ordinary interval vector $\boldsymbol{X}^{(0)} = [X_1^{(0)}, X_2^{(0)}, ..., X_n^{(0)}]$, without evaluating the functions over any sub-box smaller than the original one (or, in other words, without splitting the box). Let $\boldsymbol{L}(\boldsymbol{X})$ denote the affine extension of $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x})]$ in box $\boldsymbol{X}$, in the form

$$L_j(\boldsymbol{X}) = a_{0,j} + \sum_{i=1}^{n+m} a_{i,j} \cdot \varepsilon_i, \qquad (j = 1, 2, ..., n) \tag{3}$$

where the noise variables ε_i $(i = 1, ..., n)$ are obtained in the course of converting the input interval vector $\boldsymbol{X}$ to the corresponding affine form, ε_i $(i = n+1, ..., n+m)$ are introduced in

the course of computation, $a_{0,j}$ and $a_{i,j}$ ($i = 1, ..., n+m$, $j = 1, 2, ..., n$) are real values also determined in the course of computation. This is an alternative to the traditional and widely used linearization (such as used in the Interval Newton method)

$$\mathbf{f(x)} \in \boldsymbol{A}(\mathbf{x} - \mathbf{z}) + \mathbf{f(z)} \tag{4}$$

in a given interval box $\boldsymbol{X}$, where $\mathbf{x} \in \boldsymbol{X}$ and $\mathbf{z} \in \boldsymbol{X}$ are real numbers, $\boldsymbol{A}(\boldsymbol{X})$ is an interval matrix, and $\mathbf{f(z)}$ is a real vector.

Form (3) in a more general form is

$$\mathbf{f(x)} \in \mathbf{A}\mathbf{x} + \boldsymbol{B} \tag{5}$$

for $\mathbf{x} \in \boldsymbol{X}$, where $\mathbf{A}$ is a real matrix in contrast to (4) where the coefficient matrix is an interval matrix, and $\boldsymbol{B}$ is an interval vector. The applicability of form (5) for root finding and optimization was studied in detail by Kolev[30-38]. Numerical examples published by Kolev[30-34,36], Nenov and Fylstra[39], Yamamura[40,41], Miyajima and Kashiwagi[42], and our results, suggest that (5) is a more efficient linearization than (4).

Let $\boldsymbol{M}$ denote the solution set of the system

$$\boldsymbol{L}(\boldsymbol{X}) = \mathbf{0} \tag{6}$$

derived from (3). $\boldsymbol{M}$ encloses all solutions of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in $\boldsymbol{X}$, but usually $\boldsymbol{M}$ is not contained in $\boldsymbol{X}$. The hull of the intersection of $\boldsymbol{X}$ and $\boldsymbol{M}$ can be computed by solving the following $2n$ LP problems

$$\min / \max\ e_i \qquad \text{for } i = 1 \ldots n$$

s.t.

$$\underline{B}_j \le \sum_{i \in S} a_{i,j} \cdot e_i \le \overline{B}_j \qquad j = 1, 2, ..., n, \tag{7}$$

$$-1 \le e_i \le 1 \qquad i \in S$$

where S denotes the index set of the noise variables shared by the equations in (6), interval $B_j = [\underline{B}_j, \overline{B}_j]$ for a fixed j is calculated by negating the sum of $a_{0,j}$ and those $a_{i,j} \cdot e_i$

products for which $i \notin S$. Use of linear programming is straightforward, and has been proposed in the literature[32] (LP has also been suggested for other linearization techniques[20,53]). Another approach to prune the box is the use of constraint propagation (CP) as proposed by Kolev[34], Algorithm A4 and A5. Procedure 3 below is an affine analogue of Kolev's algorithms.

Procedure 3

Components of the original box $\boldsymbol{X}$ are determined as $X_i = x_{0,i} + x_i \varepsilon_i$ ($i = 1, ..., n$).

Step 1

For $i = 1$ to $n+m$, $i \in S$, initialize e_i-s with $[-1, 1]$

Step 2

For $j = 1$ to n repeat

For $i = 1$ to $n+m$, $i \in S$, repeat

$$e_i \leftarrow e_i \cap \left(\frac{1}{a_{i,j}} \left(B_j - \sum_{\substack{k \neq i \\ k \in S}} a_{k,j} e_k \right) \right) \tag{8}$$

Step 3

The result is the (hopefully) pruned box $\boldsymbol{X}'$, with components for $i = 1, ..., n$

$X_i' \leftarrow x_{0,i} + x_i e_i$

Values $a_{i,j}$ and B_j are computed before starting Procedure 3, and they remain unchanged during the procedure (similarly to Algorithm A4 of Kolev[34]). This approach will henceforth be referred to as "the pre-evaluated variant of Procedure 3".

Note that it is also possible to apply Procedure 3 equationwise (similarly to Kolev's Algorithm A5[34]): $a_{i,j}$ and B_j values are computed for the j-th equation using the current box, then (8) is applied and the box is (hopefully) reduced, and the next iteration for the $j+1$-

th equation uses the reduced box, *etc.* This approach will from now on be referred to as the "AA/CP method" or as "the equationwise variant of Procedure 3". Depending on the number and the computation costs of the shared sub-expressions, the equationwise variant of Procedure 3 can be applied on a subset of the equations only, and the pre-evaluated variant is used for the rest.

Constraint propagation (CP) and linear programming (LP) can also be combined in order to reduce the box by way of first applying Procedure 3 and then using LP as described above. This combination will from now on be referred to as "the AA/LP method".

In the above pruning methods, the number of variables need not equal the number of equations. Redundant equations and inequalities may significantly speed up the computation.

The AA/LP method needs less iteration and is expected to be more robust than the AA/CP method though the computation cost of one iteration in AA/LP is obviously higher. This higher cost per iteration may be compensated by the reduced number of iterations. The relative performance of the AA/CP and AA/LP methods may vary substantially depending on the problem and the implementation of the methods.

The performance of IN/GS may be worsened[21,30,54] by the so called dependency problem (troublesome overestimation of ranges). AA automatically keeps track of the correlation between the partial results obtained during the evaluation of functions; this reduces overestimation. As the examples considered in the present paper are full of dependency, the AA/CP benefits from the partially remedied dependency. The reduced overestimation in AA also influences CP (Procedure 3) favorably. However this CP has a low efficiency compared to the cutting edge CP techniques based on directed acyclic graph (DAG). These *state-of-the-art* CP techniques[21-23] can be easily extended to the mixed AA/IA model and are expected to speed up the computation by orders of magnitude.

Form (5) has a simpler structure than (4) (real *vs.* interval coefficient matrix). If the real coefficient matrix $\mathbf{A}$ is invertible, computing the hull of the solution set $\boldsymbol{W}$ of

$$\mathbf{Ax} + \boldsymbol{B} = \mathbf{0} \tag{9}$$

is straightforward (neglecting rounding)

$$\boldsymbol{Y} = -\mathbf{A}^{-1}\boldsymbol{B}\,, \tag{10}$$

and then the search box $\boldsymbol{X}$ can be pruned as

$$\boldsymbol{X}' = \boldsymbol{X} \cap \boldsymbol{Y}\,. \tag{11}$$

In contrast, IN/GS yields an interval solution usually wider than the hull of the solution set of $\boldsymbol{A}(\mathbf{x}-\mathbf{z}) + \mathbf{f}(\mathbf{z}) = \mathbf{0}$ corresponding to (9). Pruning according to (10) and (11) is not used in this study.

A geometrical illustration of the different pruning techniques is given by Kolev[54]. As it is shown in Figure 2, the hull of the intersection of the search box and the solution set of (9) is computed ($\boldsymbol{X}'' = \mathrm{Hull}(\boldsymbol{X} \cap \boldsymbol{W})$) with the LP pruning technique. In contrast, the intersection of the search box and the hull of the solution set of (9) is computed ($\boldsymbol{X}' = \boldsymbol{X} \cap \mathrm{Hull}(\boldsymbol{W}) = \boldsymbol{X} \cap \boldsymbol{Y}$) with CP (10) and (11). LP is obviously advantageous in pruning (Figure 2a) and discarding (Figure 2b).

Bounding all solutions

The goal is to bound all solutions of (2) in boxes with a diameter smaller than a specified threshold. A recursive procedure is carried out starting with stack D containing the initial box $\boldsymbol{X}^{(0)} = [X_1^{(0)}, X_2^{(0)}, ..., X_n^{(0)}]$ at the beginning. The top element $\boldsymbol{X}^{(k)}$ of D $(k \geq 0)$ is taken out and removed from D, and $\boldsymbol{F}(\boldsymbol{X}^{(k)})$ is evaluated. If $\mathbf{0} \notin \boldsymbol{F}(\boldsymbol{X}^{(k)})$ then there is no solution in $\boldsymbol{X}^{(k)}$, and the box is discarded. If $\mathbf{0} \in \boldsymbol{F}(\boldsymbol{X}^{(k)})$ then tight bounds on the solution set of the relaxed system $\boldsymbol{F}(\boldsymbol{X}^{(k)}) = \mathbf{0}$ are calculated by any of the methods described above. Box $\boldsymbol{X}^{(k)}$

is deleted if an empty interval is obtained during constraint propagation according to Procedure 3, or if the LP problems (7) are infeasible. If the maximum diameter of box $\boldsymbol{X}^{(k)}$ cannot be pruned to drop below a small pre-defined threshold then the box is bisected in a carefully chosen direction, yielding boxes $\boldsymbol{X}^{(k+1)}$ and $\boldsymbol{X}^{(k+2)}$; otherwise $\boldsymbol{X}^{(k)}$ is placed in a list R of the candidate solutions. The steps above are repeated on box $\boldsymbol{X}^{(k+1)}$ (starting with evaluating $\boldsymbol{F}(\boldsymbol{X}^{(k+1)})$); box $\boldsymbol{X}^{(k+2)}$ is pushed to stack D. This recursive method is finished when stack D becomes empty. The result is stored in list R; each solution of (2) in $\boldsymbol{X}^{(0)}$ is contained in one of the boxes in this list, though not all the boxes necessarily contain a solution.

Implementation

Implemented linearization and pruning techniques

One of our aims is to compare the efficiency of the AA-based linearization technique to that of the traditional IN when they are applied to example problems taken from the field of chemical engineering. In order to maintain comparability of the two linearization methods, analogous techniques of box pruning and bisection are applied in both cases (AA/CP and IN/GS). Although *state-of-art* methods[19-23] may be several orders of magnitude faster than the IN/GS implementation[24] used in this study, this version of IN/GS is applied for fair comparison. In the same way as IN based root finding has been made gradually faster in the last few decades, there is still room for improvement left for the AA methods, see section *Further Enhancements*.

In order to compare the efficiency and robustness of CP with and without LP in the case of AA-based linearization, both techniques are implemented. The source code of AA/CP is

directly derived from the source code of AA/LP by commenting out the LP pruning part. AA/LP is compared to AA/CP only; comparison of IN/GS and AA/LP would be unfair because not only the linearization techniques, but also the pruning methods, are different.

Implementation details

IA computations are carried out with the Interval Newton / Gauss Seidel / Generalized Bisection root finding algorithm (IN/GS/GB) implemented in C++ Toolbox 2.0[24]. It has been necessary to use the splitting rule of Kearfott and Novoa[25], that takes into account scaling, instead of the original code but no other change is applied to the source code.

Affine arithmetic is realized using the map container of the C++ Standard Template Library; rigorous operations use the C-XSC 2.0 library[55]. Basic arithmetic operations and 3 non-affine elementary functions, namely $\ln(x)$ using Chebyshev approximation, $\exp(x)$ using Chebysev approximation, and x^2 using min-range approximation, are currently implemented. Multiplication and division rules are applied according to Kolev[36]. As for other implementation issues, the monograph of Stolfi and Figueiredo[27] has been followed. All the arithmetic operations and the functions include rigorous directed rounding.

Procedure 3, the general scheme of constraint propagation (8) analogous to Kolev's Algorithm A5[34], is implemented and used. LP problems are solved by primal simplex algorithm of the general-purpose LP solver[56] GNU GLPK 4.10. As all LP problems in (7) share the same constraints, exactly one basic feasible solution is generated in Phase I of the primal simplex algorithm. This basic feasible solution is used when Phase II of the primal simplex algorithm is called for the first time. All other LP problems in (7) use the optimal solution of the previous LP problem as a basic feasible solution and run only Phase II of the

primal simplex algorithm instead of invoking Phase I repeatedly. After each run of Phase II, the corresponding bound of e_i is set to the found optimal objective value.

There is no guarantee that the exact bounds of the solution set are enclosed by LP pruning because ordinary floating-point arithmetic is used for solving LP problems. Rigorous procedures for determining safe bounds on LP objective value have already been proposed in the literature[20,57,58] but none of them is applied in the current implementation.

A very simple yet efficient splitting rule is implemented: bisect the variable for which the diameter of $a_{i,j} \cdot e_i$ is the largest after applying pruning methods. This is an affine analogue of the well-known splitting rule of Kearfott and Novoa[25].

All computations have been carried out on the same machine 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 (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.

Separation problems

Liquid phase split

Given the gross composition $\mathbf{z}$, the goal is to determine the relative amounts (λ) and compositions ($\mathbf{x}$ and $\mathbf{y}$) of at most two phases in equilibrium. The number of phases is not restricted to just two generally, but only two phases are considered in the present paper. The equifugacity conditions are solved at constant pressure P and temperature T.

A continuous set of trivial solutions

$$z_i = x_i = y_i \qquad (i = 1, 2, ..., C),$$

always exists with the *arbitrary* value of λ, where z_i, x_i, y_i denote the mole fraction of the i-th component in the feed, phase I, and phase II, respectively, and C denotes the number of components. This continuous set of trivial solutions obviously gives rise to difficulty in applying interval methodology. One possibility to overcome this problem is to cut out a narrow interval containing the feed composition z_i from the search intervals of x_i and y_i, and to solve $2C$ sub-problems formed over disjoint sub-domains. In this way non-trivial solutions for which x_i and y_i $(i = 1, 2, ..., C)$ are approximately equal may be lost but this is practically acceptable. Another trick to avoid trivial solutions is detailed below.

The "cut-out method" used for dealing with the problem of trivial solutions is only applicable when the gross composition is known. In the iterative procedure of solving the equation system of a counter-current cascade, such as a counter-current extractor or a distillation column, the gross composition of each equilibrium unit is not *a priori* known. This *a priori* knowledge of the gross composition is not needed in our methodology.

Variables

For $i = 1, 2, ..., C$ (C denotes the number of components)

x_i (mole fraction of component i in phase I)

y_i (mole fraction of component i in phase II)

λ (relative amount of phase I)

r (auxiliary variable)

The number of variables is $2C + 2$, where C is the number of components. The initial search interval for all x_i and y_i variables is $[\delta, 1.0]$ where δ is a small positive value *e.g.* 10^{-6}; the

search interval for λ is $[0.0, 1.0]$. The auxiliary variable r is used in the equation cutting out the continuous set of trivial solutions.

Equations

Sum of the mole fractions equals unity:

$$\sum_{i=1}^{C} x_i = 1$$
$$\sum_{i=1}^{C} y_i = 1 \qquad (12)$$

Component material balances:

$$\lambda \cdot x_i + (1-\lambda) \cdot y_i = z_i \qquad (i = 1, 2, ..., C) \qquad (13)$$

The specified z_i values (feed composition) are given in Table A1.

Equifugacity conditions:

$$\ln \gamma_i(x_1, x_2, ..., x_C) + \ln x_i = \ln \gamma_i(y_1, y_2, ..., y_C) + \ln y_i \qquad (i = 1, 2, ..., C) \qquad (14)$$

where the NRTL equations are used as a liquid phase activity coefficient model

$$\ln \gamma_i(x_1, x_2, ..., x_C) = \frac{\sum_{j=1}^{C} \tau_{ji} G_{ji} x_j}{\sum_{k=1}^{C} G_{ki} x_k} + \sum_{j=1}^{C} \frac{x_j G_{ij}}{\sum_{k=1}^{C} G_{kj} x_k} \cdot \left(\tau_{ij} - \frac{\sum_{l=1}^{C} \tau_{lj} G_{lj} x_l}{\sum_{k=1}^{C} G_{kj} x_k} \right) \qquad (15)$$

$$\tau_{ij} = \frac{B_{ij}}{T}$$

$$G_{ij} = \exp(-\alpha_{ij} \cdot \tau_{ij})$$

Values of the model parameters B_{ij} and α_{ij} are listed in Tables A2 and A3, the specified temperature T is given in Table A1.

The continuous set of trivial solutions is cut out by the following equation

$$\sum_{i=1}^{C-1} (x_i - y_i)^2 = r \qquad (16)$$

where r is an additional variable with search interval $[\Delta_{\min}, C-1]$, and $\Delta_{\min}$ is a small positive number, *e.g.* 10^{-4}. Equation (16) also cuts out non-trivial solutions for which $\sum_{i=1}^{C-1}(x_i - y_i)^2 < \Delta_{\min}$ holds but this is practically acceptable.

Numerical results for L/L phase split problem

The specified feed composition, temperature, and initial box, are given in Table A1. Parameters of the NRTL equations are taken from the databank of ChemCAD 5.5.4 (Chemstations, Inc.), and are listed in Table A2 and A3. Relative tolerance is set to 10^{-3}. The computation results and their comparison are collected in Tables 3 to 5.

Notes

When $C = 2$, equation (15) reduces to the form

$$\ln\gamma_1(x_1, x_2) = x_2^2\left(\tau_{21}\frac{G_{21}^2}{(x_1 + x_2G_{21})^2} + \frac{\tau_{12}G_{12}}{(x_2 + x_1G_{12})^2}\right)$$

$$\ln\gamma_2(x_1, x_2) = x_1^2\left(\tau_{12}\frac{G_{12}^2}{(x_2 + x_1G_{12})^2} + \frac{\tau_{21}G_{21}}{(x_1 + x_2G_{21})^2}\right)$$

where $x_2 = 1 - x_1$. Shared sub-expressions used during the affine computations are

$$(x_1 + x_2G_{21})^2 \text{ and } (x_2 + x_1G_{12})^2.$$

In this special case the nonlinear system of equations (12)-(14) and (16) is expressed in the form

$$z_1 - \lambda x_1 - (1-\lambda)y_1 = 0,$$

$$(\ln\gamma_1(x_1, x_2) + \ln x_1) - (\ln\gamma_1(y_1, y_2) + \ln y_1) = 0,$$

$$(\ln\gamma_2(x_1, x_2) + \ln x_2) - (\ln\gamma_2(y_1, y_2) + \ln y_2) = 0,$$

$$(x_1 - y_1)^2 = r\,,$$

where $x_2 = 1 - x_1$, $y_2 = 1 - y_1$, and the variables are $x_1,\ y_1,\ \lambda,\ r$.

When $C = 3$ equation (15) is expanded and symbolically simplified to give

$$\ln\gamma_1 = \frac{G_{21}^2\tau_{21}x_2^2 + G_{21}G_{31}(\tau_{21}+\tau_{31})x_2x_3 + G_{31}^2\tau_{31}x_3^2}{(x_1 + G_{21}x_2 + G_{31}x_3)^2} + \frac{G_{12}\tau_{12}x_2^2 + G_{12}G_{32}(\tau_{12}-\tau_{32})x_2x_3}{(G_{12}x_1 + x_2 + G_{32}x_3)^2} + \frac{G_{13}\tau_{13}x_3^2 + G_{13}G_{23}(\tau_{13}-\tau_{23})x_2x_3}{(G_{13}x_1 + G_{23}x_2 + x_3)^2}$$

$$\ln\gamma_2 = \frac{G_{21}\tau_{21}x_1^2 + G_{21}G_{31}(\tau_{21}-\tau_{31})x_1x_3}{(x_1 + G_{21}x_2 + G_{31}x_3)^2} + \frac{G_{12}^2\tau_{12}x_1^2 + G_{12}G_{32}(\tau_{12}+\tau_{32})x_1x_3 + G_{32}^2\tau_{32}x_3^2}{(G_{12}x_1 + x_2 + G_{32}x_3)^2} + \frac{G_{23}\tau_{23}x_3^2 + G_{23}G_{13}(\tau_{23}-\tau_{13})x_1x_3}{(G_{13}x_1 + G_{23}x_2 + x_3)^2}$$

$$\ln\gamma_3 = \frac{G_{31}\tau_{31}x_1^2 + G_{31}G_{21}(\tau_{31}-\tau_{21})x_1x_2}{(x_1 + G_{21}x_2 + G_{31}x_3)^2} + \frac{G_{32}\tau_{32}x_2^2 + G_{12}G_{32}(\tau_{32}-\tau_{12})x_1x_2}{(G_{12}x_1 + x_2 + G_{32}x_3)^2} + \frac{G_{13}^2\tau_{13}x_1^2 + G_{13}G_{23}(\tau_{13}+\tau_{23})x_1x_2 + G_{23}^2\tau_{23}x_2^2}{(G_{13}x_1 + G_{23}x_2 + x_3)^2}$$

Shared sub-expressions are

$$x_1^2,\ x_2^2,\ x_3^2,\ x_1x_2,\ x_1x_3,\ x_2x_3\,,$$

$$(x_1 + G_{21}x_2 + G_{31}x_3)^2,\ (G_{12}x_1 + x_2 + G_{32}x_3)^2,\ (G_{13}x_1 + G_{23}x_2 + x_3)^2\,.$$

The above particular forms have been applied all over the numerical calculations.

Discussion

As for comparing linearization techniques, in the binary case AA/CP outperforms IN/GS by an order of magnitude in speed, although the cycle time of IN/GS is approximately a quarter of the other. Three days is not enough time for IN/GS to find the solutions in the ternary case, whereas AA/CP finds all solutions in less than 24 seconds. As it is clear from Table 5, the total computation cost of IN/GS increases much faster with the size and difficulty of the problem than with that of AA/CP.

As for comparing the pruning techniques, no considerable difference between computation time of AA/CP and AA/LP is found in both the binary and the ternary case. AA/LP requires slightly less iterations to converge compared to AA/CP but the cycle time of AA/LP is longer, and this results in an almost equal total computation time.

Counter-current equilibrium cascades

Distillation columns with a total condenser (and liquid distillate product) at the top and a total boiler at the bottom, and with a single theoretical stage or two stages only in between, are shown in Figure 3. The steady state of such processes is determined in the examples below. The MESH equations (component **m**aterial balances, **e**quilibrium conditions, **s**ummation equations, and **h**eat balance equations) are simultaneously solved. The vapor phase is modeled as ideal gas, and a fairly simple enthalpy model is used.

Variables

For $i = 1, 2, ..., C$ and $j = 1, 2, ..., N$ (where C and N denote the number of components and the number stages, respectively)

$x_{i,j}$ (mole fraction of component i in the liquid phase at stage j),

$y_{i,j}$ (mole fraction of component i in the vapor phase at stage j),

V_j (vapor flow rate at stage j),

T_j (temperature at stage j).

Equations

All the equations below apply to $i = 1, 2, ..., C$ and / or $j = 1, 2, ..., N$.

Summation (S_x and S_y) equations:

$$\sum_{i=1}^{C} x_{i,j} = 1$$

$$\sum_{i=1}^{C} y_{i,j} = 1$$

Component material balance (M) equations:

$$L_{j-1}x_{i,j-1} + V_{j+1}y_{i,j+1} + F_j z_{i,j} = L_j x_{i,j} + V_j y_{i,j}$$

where L_j and F_j denote the liquid flow rate and feed flow rate at stage j, respectively; $z_{i,j}$ denotes the specified mole fraction of the i-th component in the feed at stage j.

Heat balance (H) equations using a fairly simple enthalpy model with the constant heat of vaporization of the components (λ_i):

$$V_j \sum_{i=1}^{C} \lambda_i \, y_{i,j} = V_{j+1} \sum_{i=1}^{C} \lambda_i \, y_{i,j+1}$$

Note that the liquid phase is left out of consideration in the heat balances, and the feed is assumed to have zero vapor fraction. The vaporization heat values are given in Table A8.

Vapor-liquid equilibrium conditions are modeled with a modified *Raoult-Dalton* equation (E equations):

$$\ln(\gamma_i(x_{1,j},\, x_{2,j},\, ...,\, x_{C,j},\, T_j)) + \ln(x_{i,j}) + \ln(p_i(T_j)) = \ln(y_{i,j}) + \ln P\,,$$

where the $\ln\gamma_i$ (natural logarithm of the activity coefficient) values are computed by the 2-parameter version of *Wilson* equations, p_i values (vapor pressure of the pure i-th component) are computed by the *Antoine* equations, pressure at each stage is specified, and is denoted by P. The pressure drop is neglected.

Auxiliary functions

The *Wilson* equations are used as liquid phase activity coefficient model:

$$\ln(\gamma_i(x_1,\, x_2,\, ...,\, x_C,\, T)) = -\ln\left(\sum_{a=1}^{C} x_a \Lambda_{ia}\right) + 1 - \sum_{b=1}^{C} \frac{x_b \Lambda_{bi}}{\sum_{a=1}^{C} x_a \Lambda_{ia}}\,, \qquad (17)$$

$$\Lambda_{ab}(T) = \frac{V_b^m}{V_a^m} \exp\left(-\frac{k_{ab}}{R_G T}\right) \quad (a = 1,\, 2,\, ...,\, C;\; b = 1,\, 2,\, ...,\, C)\,. \qquad (18)$$

Model parameters k_{ab} and V_i^m are given in Table A6 and A7 respectively; R_G denotes the general (*Regnault*'s) gas constant (1.98721 cal/(mol·K)); T denotes the temperature at stage j (index j is dropped for simplicity).

Pure components vapor pressures p_i are computed from *Antoine*-equations:

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

Coefficients A_i, B_i, C_i are given in Table A9; T denotes the temperature at stage j (index j is dropped for simplicity).

Flow rates that follow from the specifications of reflux ratio R and reboil ratio B:

$$L_0 = \frac{R}{R+1} V_1$$

$$V_{N+1} = \frac{B}{B+1} L_N$$

Relations for flow rates that follow from total material balance:

$$L_N = (B+1)(-\frac{1}{R+1} V_1 + \sum_{k=1}^{N} F_k),$$

$$L_j = V_{j+1} - \frac{1}{R+1} V_1 + \sum_{k=1}^{j} F_k \qquad (j = 1, 2, ..., N-1).$$

Notes

Shared sub-expressions used in the affine computations for a fixed j are

$$\frac{1}{T},$$

$$\Lambda_{ab} \qquad (a = 1, 2, ..., C;\ b = 1, 2, ..., C),$$

$$\sum_{a=1}^{C} x_a \Lambda_{ia} \qquad (i = 1, 2, ..., C).$$

Here index j is dropped for simplicity.

Cascade with a single theoretical stage between a boiler and a condenser

The studied process is illustrated in Figure 3a. Components are (1) acetone, (2) methanol, (3) water ($C = 3$). Specifications are: reflux ratio $R = 3$, reboil ratio $B = 4$; feed flow rate $F_1 = 1.00$ mol/s; feed composition $z_{1,1} = 0.33$, $z_{2,1} = 0.34$, $z_{3,1} = 0.33$; and column pressure $P = 101325$ Pa. Initial box and solution are given in Table A4.

Cascade with 2 equilibrium stages between a boiler and a condenser

The studied process is illustrated in Figure 3b. Components are (1) acetone, (2) methanol, (3) water ($C = 3$). Specifications are: reflux ratio $R = 3$, reboil ratio $B = 4$; feed flow rates $F_1 = 0.00$ mol/s, $F_2 = 1.0$ mol/s, feed composition $z_{1,2} = 0.33$, $z_{2,2} = 0.34$, $z_{3,2} = 0.33$; and column pressure $P = 101325$ Pa. Computations are carried out with 3 different initial boxes ("small, medium, and large initial boxes"), the solution and the initial boxes are listed in Table A5.

Numerical results for counter-current equilibrium cascades

Parameters of the *Wilson* equations and the *Antoine* constants are taken from the databank of ChemCAD 5.5.4 (Chemstations, Inc.), and are listed in Table A6-A9. Solution and search intervals for variables of the three stages problem, shown in Figure 3a, are listed in Table A4. Solution and search intervals (small, medium, and large size boxes) for variables of the four stages problem, shown in Figure 3b, are listed in Table A5. Relative tolerance is set to 10^{-3} (3 significant digits) in each case. The computation results and their comparison are collected in Tables 6 to 8.

Discussion

As for comparing the linearization techniques, the AA/CP method outperforms IN/GS in the case of three stages by an order of magnitude in speed. AA/CP requires two orders of magnitude less iterations to converge. A negligible clustering effect is observed with both methods.

As for comparing the pruning techniques, the AA/LP method proves to be more than four times faster than AA/CP in the case of three stages, and produces less cluster boxes. In the case of four stages a serious clustering effect is observed with the AA/CP method, and this clustering effect makes it difficult to compare the results with AA/LP in a fair way. The authors presume that the clustering effect is caused by the relative low performance of the applied constraint propagation technique (Procedure 3) compared to that of AA/LP. The results of the four stages case show the robustness and efficiency of the AA/LP method.

Further enhancements to apply

Numerical tests not detailed here suggest that using static arrays for storing the coefficients and indices of the noise variables instead of the map container gives 1-2 orders of magnitude speed-up of the computation.

Optimal multiplication which yields no overestimation has been proposed by Kolev[37,38]. Miyajima, Miyata, and Kashiwagi[59,60] suggest that linear approximation $ax+by+c$ to binary operations $f(x,y)$ should be computed over the joint range of the operands and not over the rectangular domain $X \times Y$. They give a detailed $O(n^2)$ algorithm for multiplication and division.

If the box is "sufficiently reduced" in size after applying the pruning method in a given iteration then, as published by Hansen[46], it is preferable to repeat the pruning step instead of splitting the box.

A hybrid splitting rule may also be easily constructed on the analogy of Beelitz *et al.*[61].

It has been necessary to combine AA and IA during the computations (similarly to Example 1), otherwise the problem of division by zero and negative argument in the logarithm function would have occurred. This combination is applied only at the critical points in the current implementation, where it is necessary. A more convenient and effective way is to implement the mixed AA/IA model in a general manner, following the ideas of Stolfi and Figueiredo[27], pp. 75-76, and using it in each step of the computation.

Static storage allocation for LP problems can be used instead of allocating and de-allocating the corresponding memory area in every iteration step. The solution methodology to solving LP problems can be further optimized *e.g.* by implementing the algorithm of Kolev[62], or by creating more sophisticated heuristics for selecting and sequencing the noise variables that are involved in the LP pruning.

Advanced constraint propagation techniques using directed acyclic graph (DAG) representation of the problem published for the IA case in *e.g.* Kearfott[21], Beelitz *et al.*[22], or Schichl and Neumaier[23], can be generalized by applying them to the mixed AA/IA model, and are expected to decrease the overall computation time significantly.

Well-known decomposition methods developed for solving the MESH equations[1,2] are also applicable in affine arithmetic. For example, it is possible to apply Procedure 3 and LP pruning (7) in the following sequence: (*a*) use Procedure 3 (equationwise manner) to the S equations, then (*b*) use them to the M equations, (*c*) compute tight bounds with LP on the box of the corresponding variables using the S and M equations, (*d*) evaluate all the H and E equations and then apply Procedure 3 (pre-evaluated manner) and, finally, (*e*) use LP pruning involving all the MESH equations.

Conclusions

A new automatic interval linearization technique based on affine arithmetic (AA) is to chemical engineering problems, and its efficiency is compared to the traditional one based on ordinary interval arithmetic (IA) by embedding them in reliable root finding algorithms (AA/CP and IN/GS, respectively). In order to make the comparison as fair as possible, the implemented methods differ in the applied linearization techniques only, the pruning and bisecting rules are (hopefully) analogous. The proposed root finding algorithm AA/CP outperforms the traditional textbook algorithm IN/GS by an order of magnitude in the case of the studied examples. It is important to mention at this point that *state-of-the-art* variants of the Interval Newton methods also outperform the IN/GS used for comparison in the present paper.

Two pruning techniques applicable together with AA are also studied and compared. One is a basic constraint propagation technique (AA/CP), the other is linear programming appended to this constraint propagation technique (AA/LP). There is no significant difference between the two methods except in the case of the most difficult problem considered. In that difficult case AA/CP produces thousands of cluster boxes whereas AA/LP does not show clustering. The clustering effect slows down the AA/CP method drastically. This example demonstrates the robustness and efficiency of the AA/LP method.

Procedure 1 and Procedure 2 are suggested for tightening the range of general non-linear expressions accompanied by general non-linear constraints as a further development of a similar technique[10] for linear functions with linear constraints. This works well in evaluating non-isothermal activity coefficient models like Wilson, NRTL, Uniquac or Unifac.

State-of-the-art CP techniques can be easily extended to the mixed AA/IA model. These methods together with other, problem specific, enhancements are expected to speed up the computation by several orders of magnitude.

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. Prausnitz J, Anderson T, Grens E, Eckert C, Hsies R, O'Connell J. *Computer Calculations for Multicomponent Vapor-Liquid and Liquid-Liquid Equilibria*. Englewood Cliffs, New Jersey: Prentice-Hall Inc; 1980.
4. McKinnon K, Mongeau M. A Generic Global Optimization Algorithm for the Chemical and Phase Equilibrium Problem. *J Glob Optim.* 1998;12:325–351.
5. McDonald CM, Floudas CA. Global optimization for the phase stability problem. *AIChE J.* 1995;41:1798–1814.
6. McDonald CM, Floudas CA. Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation. *Comput Chem Eng.* 1995;19:1111–1139.
7. McDonald CM, Floudas CA. Global optimization and analysis for the Gibbs free energy function using the UNIFAC, Wilson, and ASOG equations. *Ind Eng Chem Res.* 1995;34:1674–1687.

8. McDonald CM, Floudas CA. GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. *Comput Chem Eng.* 1997;21:1–23.
9. Stadtherr MA, Schnepper CA, Brennecke JF. Robust Phase Stability Analysis Using Interval Methods. *AIChE Symp Ser.* 1995;91(304):356–359.
10. Hua JZ, Brennecke JF, Stadtherr MA. Enhanced Interval Analysis for Phase Stability: Cubic Equation of State Models. *Ind Eng Chem Res.* 1998;37:519–1527.
11. Tessier SR, Brennecke JF, Stadtherr MA. Reliable Phase Stability Analysis for Excess Gibbs Energy Models. *Chem Eng Sci.* 2000;55:1785–1796.
12. Xu G, Brennecke JF, Stadtherr MA. Reliable Computation of Phase Stability and Equilibrium from the SAFT Equation of State. *Ind Eng Chem Res.* 2001;41:938–952.
13. Xu G, Haynes WD, Stadtherr MA. Reliable Phase Stability Analysis for Asymmetric Models. *Fluid Phase Equilib.* 2005;235:152–165.
14. 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.
15. Müller D, Marquardt W. Experimental Verification of Multiple Steady States in Heterogeneous Azeotropic Distillation. *Ind Eng Chem Res.* 1997;36:5410–5418.
16. Lin Y, Gwaltney CR, Stadtherr MA. Reliable Modeling and Optimization for Chemical Engineering Applications: Interval Analysis Approach. *Reliable Computing.* 2006;12:427–450.
17. Nedialkov NS, Jackson KR, Corliss GF. Validated solutions of initial value problems for ordinary differential equations. *Appl Math Comput.* 1999;105:21–68.
18. Lin Y, Stadtherr MA. Validated Solutions of Initial Value Problems for Parametric ODEs. Appl Numer Math. 2007;58:1145–1162.

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

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

21. Kearfott RB. Decomposition of Arithmetic Expressions to Improve the Behavior of Interval Iteration for Nonlinear Systems. *Computing* 1991;47(2):169–191.

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

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

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

25. Kearfott RB, Novoa M. ALGORITHM 681, INTBIS, a Portable Interval Newton / Bisection Package. *ACM Transactions on Mathematical Software.* 1990;16:152–157.

26. 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.

27. 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.

28. Stolfi J, Figueiredo LH. An Introduction to Affine Arithmetic. *TEMA Tend Mat Apl Comput.* 2003;4:297–312.

29. Figueiredo LH, Stolfi J. Affine Arithmetic: Concepts and Applications. *Numerical Algorithms.* 2004;37:147–158.

30. Kolev LV. A New Method for Global Solution of Systems of Non-Linear Equations. *Reliable Computing.* 1998;4:125–146.
31. Kolev LV. An efficient interval method for global analysis of non-linear resistive circuits. *Int J Circuit Theory Appl.* 1998;26:81–92.
32. 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.
33. Kolev LV. An Improved Method for Global Solution of Non-Linear Systems. *Reliable Computing.* 1999;5:103–111.
34. Kolev LV. An Interval Method for Global Nonlinear Analysis. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications.* 2000;47:675–683.
35. Kolev LV. Automatic Computation of a Linear Interval Enclosure. *Reliable Computing.* 2001;7:17–28.
36. Kolev LV. An improved interval linearization for solving non-linear problems. *Numerical Algorithms.* 2004;37:213–224.
37. Kolev LV. New Formulae for Multiplication of Intervals. *Reliable Computing.* 2006;12:281–292.
38. Kolev LV. Optimal multiplication of G-intervals. *Reliable Computing.* 2007;13(5):399–408.
39. Nenov IP, Fylstra DH. Interval Methods for Accelerated Global Search in the Microsoft Excel Solver. *Reliable Computing.* 2003;9:143–159.
40. 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.

41. 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.**
42. Miyajima S, Kashiwagi M. Existence test for solution of nonlinear systems applying affine arithmetic. *J Comput Appl Math*. 2007;199:304–309.
43. Moore RE. *Methods and Applications of Interval Analysis.* Philadelphia: SIAM; 1979.
44. Alefeld G, Herzberger J. *Introduction to Interval Computations.* New York: Academic Press; 1983.
45. Neumaier A. *Interval Methods for Systems of Equations.* Cambridge: Cambridge University Press; 1990.
46. Hansen ER. *Global Optimization Using Interval Analysis.* New York: Marcel Dekker; 1992.
47. Kolev LV. *Interval Methods for Circuit Analysis (Advanced Series on Circuits and Systems – Vol 1).* River Edge, NJ: World Scientific Pub Co Inc; 1993.
48. Kearfott RB. *Rigorous Global Search: Continuous Problems.* Dordrecht, The Netherlands: Kluwer Academic Publishers; 1996.
49. Wilson GM. Vapor-Liquid Equilibrium. XI. A New Expression for the Excess Free Energy of Mixing. *J Am Chem Soc.* 1964;86:127–130.
50. Renon H, Prausnitz JM. Local compositions in thermodynamic excess functions for liquid mixtures. *AIChE J.* 1968;14:135–144.
51. Abrams DS, Prausnitz JM. Statistical thermodynamics of liquid mixtures: A new expression for the excess Gibbs energy of partly or completely miscible systems. *AIChE J.* 1975;21:116–128.
52. Fredenslund A, Gmehling J, Rasmussen P. *Vapor-liquid equilibria by UNIFAC. A group contribution method.* Englewood Cliffs, NJ: Elsevier; 1977.

53. Yamamura K, Igarashi N. An interval algorithm for finding all solutions of non-linear resistive circuits. *Int J Circuit Theory Appl.* 2004;32:47–55.
54. Kolev LV. Some ideas towards global optimization of improved efficiency. Presented at *GICOLAG Workshop,* Wien, Austria, 4-8 December 2006.
55. Klatte R, Kulisch U, Wiethoff A, Lawo C, Rauch M. *C-XSC – A C++ Class Library for Extended Scientific Computing* –. Heidelberg: Springer-Verlag; 1993.
56. GNU Linear Programming Kit, http://www.gnu.org/software/glpk
57. Neumaier A, Shcherbina O. Safe bounds in linear and mixed-integer programming. *Math. Programming A.* 2004;99:283–296.
58. Jansson C. Rigorous Lower and Upper Bounds in Linear Programming. *SIAM J. Optim.* 2004;14:914–935.
59. Miyajima S, Miyata T, Kashiwagi M. A New Dividing Method in Affine Arithmetic. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer.* 2003;E86-A(9):2192–2196.
60. Miyajima S, Kashiwagi M. A dividing method utilizing the best multiplication in affine arithmetic. *IEICE Electron. Express.* 2004;1(7):176–181.
61. Beelitz T, Bischof CH, Lang B. A Hybrid Subdivision Strategy for Result-Verifying Nonlinear Solvers. *Proceedings in Applied Mathematics and Mechanics.* 2004;4:632–633.
62. Kolev LV. Minimum-fuel minimum-time control of linear discrete systems. *Int J Control.* 1978;27:21–29.

Appendix

Liquid-liquid phase split problem

Table A1

Search intervals and solutions for the L/L phase split problem with components: (1) methanol, (2) cyclohexane and feed composition [0.5, 0.5] in case $C = 2$; (1) acetone, (2) toluene, (3) water and feed composition [0.2, 0.4, 0.4] in case $C = 3$, at $T = 298.15\,\mathrm{K}$. Only one of the non-trivial solutions is given.

	Case $C = 2$		Case $C = 3$	
Variable	Search interval	Solution (3 digits)	Search interval	Solution (3 digits)
x_1	[1.0e-6, 0.999999]	0.841	[1.0e-6, 1.0]	0.0446
x_2	—	—	[1.0e-6, 1.0]	0.000603
x_3	—	—	[1.0e-6, 1.0]	0.955
y_1	[1.0e-6, 0.999999]	0.108	[1.0e-6, 1.0]	0.311
y_2	—	—	[1.0e-6, 1.0]	0.686
y_3	—	—	[1.0e-6, 1.0]	0.00284
λ	[0.0, 1.0]	0.535	[0.0, 1.0]	0.417
r	[0.0001, 1.00]	0.538	[0.0002, 2.0]	0.541

Table A2

Parameters of the NRTL equations. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.)

Components: (1) methanol, (2) cyclohexane

B_{12} (cal/mol)	B_{21} (cal/mol)	$\alpha_{12} = \alpha_{21}$
593.739	668.941	0.3995

Table A3

Parameters of the NRTL equations. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.)

Components: (1) acetone, (2) toluene, (3) water

i	j	B_{ij} (cal/mol)	B_{ji} (cal/mol)	$\alpha_{ij} = \alpha_{ji}$
1	2	-124.774	366.098	0.295
1	3	377.577	653.885	0.5856
2	3	2839.37	2160.78	0.2

Steady state of vapor-liquid equilibrium stages

Table A4

Search intervals and solution for the cascade with one V/L equilibrium stage (shown in Figure 2a), components: (1) acetone, (2) methanol, (3) water ($C = 3$, $N = 1$). Specifications: reflux ratio $R = 3$, reboil ratio $B = 4$; feed flow rate $F_1 = 1.00$ mol/s; feed composition $z_{1,1} = 0.33$, $z_{2,1} = 0.34$, $z_{3,1} = 0.33$; and $P = 101325$ Pa.

Variable	Search interval	Solution (3 digits)
$x_{1,1}$	[0.02, 0.98]	0.164
$x_{2,1}$	[0.02, 0.98]	0.318
$x_{3,1}$	[0.02, 0.98]	0.517
$y_{1,1}$	[0.02, 0.98]	0.478
$y_{2,1}$	[0.02, 0.98]	0.359
$y_{3,1}$	[0.02, 0.98]	0.162
V_1	[1.5, 2.8]	2.11
T_1	[300.0, 350.0]	337

Table A5

Search intervals and solution for the cascade with two V/L equilibrium stages (shown in Figure 2b), components: (1) acetone, (2) methanol, (3) water ($C = 3$, $N = 2$). Specifications: reflux ratio $R = 3$, reboil ratio $B = 4$; feed flow rates $F_1 = 0.00$ mol/s, $F_2 = 1.0$ mol/s, feed composition $z_{1,2} = 0.33$, $z_{2,2} = 0.34$, $z_{3,2} = 0.33$; and $P = 101325$ Pa.

Variable	Search intervals			Solution (3 digits)
	case *a* ("small initial box")	case *b* ("medium initial box")	case *c* ("large initial box")	
$x_{1,1}$	[0.31, 0.34]	[0.25, 0.40]	[0.02, 0.98]	0.327
$x_{2,1}$	[0.41, 0.44]	[0.35, 0.50]	[0.02, 0.98]	0.428
$x_{3,1}$	[0.22, 0.25]	[0.15, 0.35]	[0.02, 0.98]	0.245
$x_{1,2}$	[0.07, 0.10]	[0.02, 0.15]	[0.02, 0.98]	0.0891
$x_{2,2}$	[0.28, 0.31]	[0.20, 0.40]	[0.02, 0.98]	0.297
$x_{3,2}$	[0.60, 0.63]	[0.50, 0.70]	[0.02, 0.98]	0.614
$y_{1,1}$	[0.52, 0.55]	[0.45, 0.60]	[0.02, 0.98]	0.536
$y_{2,1}$	[0.36, 0.39]	[0.30, 0.45]	[0.02, 0.98]	0.377
$y_{3,1}$	[0.07, 0.10]	[0.02, 0.15]	[0.02, 0.98]	0.0876
$y_{1,2}$	[0.37, 0.40]	[0.30, 0.45]	[0.02, 0.98]	0.382
$y_{2,2}$	[0.40, 0.43]	[0.35, 0.50]	[0.02, 0.98]	0.414
$y_{3,2}$	[0.19, 0.22]	[0.15, 0.25]	[0.02, 0.98]	0.204
V_1	[2.0, 2.3]	[1.6, 2.8]	[1.6, 2.8]	2.16
V_2	[1.9, 2.2]	[1.5, 2.7]	[1.5, 2.7]	2.06
T_1	[330.0, 335.0]	[327.0, 337.0]	[300.0, 350.0]	333
T_2	[339.0, 344.0]	[336.0, 346.0]	[325.0, 375.0]	341

Table A6

Parameters of the Wilson equation. Data is from the databank of ChemCAD 5.5.4 (Chemstations, Inc.)

Components: (1) acetone, (2) methanol, (3) water

i	j	k_{ij} (cal/mol)	k_{ji} (cal/mol)
1	2	-157.981	592.638
1	3	393.27	1430.0
2	3	-52.605	620.63

Table A7

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 A8

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.

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

Table A9

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.)

Component i	A_i	B_i (K)	C_i (K)
Acetone	16.732	2975.9	-34.523
Methanol	18.51	3593.4	-35.225
Water	18.304	3816.4	-46.13

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 Chebysev approximation, (d) mixed AA/IA model with Chebysev approximation. The slope of the dashed line computed by AA correlates well with the slope of the approximated function.

Figure 2. Geometrical illustration of the LP and CP pruning techniques according to Kolev[54]. (a) LP is advantageous in pruning because $\boldsymbol{X}''\subset\boldsymbol{X}'$ (b) LP is advantageous in discarding (feasibility check) as $W\cap X=\varnothing$ whereas CP gives $Y\cap X\neq\varnothing$

Figure 3. Cascades of the total condenser, total boiler and (*a*) a single theoretical stage in between (*b*) two theoretical stages in between. Specified streams are represented by solid lines, whereas unknown streams are dashed.

Tables

Table 1. Results for evaluating $(x-1)/(x^2+2)$ over $[2, 4]$ with various methods.

Method	Computed interval	Width related to the true range
IA	$[0.055555, 0.500000]$	27.19
AA with min-range	$[0.055555, 0.271605]$	13.22
AA with Chebysev	$[0.151207, 0.200000]$	2.985
Mixed AA/IA	$[0.153784, 0.196860]$	2.635
True range	$[0.166666, 0.183013]$	1.000

Table 2. Results for Example 3.

Method	i	Resulted range of $\sum_{a=1}^{3} x_a \Lambda_{ia}$	Width relative to result of Procedure 2 (with the same i)
AA[1]	1	[-0.020901, 1.853065]	2.08
	2	[-0.085791, 2.319595]	3.96
	3	[-0.247357, 2.639316]	3.40
Procedure 2	1	[0.116456, 1.019514]	1.00
	2	[0.436081, 1.043752]	1.00
	3	[0.331738, 1.180088]	1.00
IA[1]	1	[0.000000, 1.861866]	2.06
	2	[0.000000, 2.314005]	3.81
	3	[0.000000, 2.607750]	3.07
Procedure 1[2]	1	[0.106708, 1.028848]	1.02
	2	[0.397333, 1.082773]	1.13
	3	[0.274760, 1.219796]	1.11

[1]Constraint $x_1 + x_2 + x_3 - 1 = 0$ is ignored in the evaluation

[2]Procedure 1 is invoked with the ranges of Λ_{ia} -s as given independent interval weights

Table 3a. Comparison of linearization techniques for $C = 2$

Method	total time [s]	$\frac{\text{total time}}{\text{total time}_{AA/CP}}$	iterations	$\frac{\text{iterations}}{\text{iterations}_{AA/CP}}$	cycle time [μs]	$\frac{\text{cycle time}}{\text{cycle time}_{AA/CP}}$
IN/GS	26.1	22.7	120234	85.5	217	0.27
AA/CP	1.15	1.00	1407	1.00	817	1.00

Table 3b. Comparison of pruning techniques for $C = 2$

Method	total time [s]	$\frac{\text{total time}}{\text{total time}_{AA/LP}}$	iterations	$\frac{\text{iterations}}{\text{iterations}_{AA/LP}}$	cycle time [μs]	$\frac{\text{cycle time}}{\text{cycle time}_{AA/LP}}$
AA/CP	1.15	0.852	1407	1.04	817	0.82
AA/LP	1.35	1.00	1355	1.00	996	1.00

Table 4a. Comparison of linearization techniques for $C = 3$

Method	total time [s]	$\frac{\text{total time}}{\text{total time}_{AA/CP}}$	iterations	$\frac{\text{iterations}}{\text{iterations}_{AA/CP}}$	cycle time [ms]	$\frac{\text{cycle time}}{\text{cycle time}_{AA/CP}}$
IN/GS	>318000	$>1.36 \cdot 10^4$	—	—	—	—
AA/CP	23.3	1.00	7715	1.00	3.01	1.00

Table 4b. Comparison of pruning techniques for $C = 3$

Method	total time [s]	$\frac{\text{total time}}{\text{total time}_{AA/LP}}$	iterations	$\frac{\text{iterations}}{\text{iterations}_{AA/LP}}$	cycle time [ms]	$\frac{\text{cycle time}}{\text{cycle time}_{AA/LP}}$
AA/CP	23.3	1.05	7715	1.33	3.01	0.79
AA/LP	22.1	1.00	5795	1.00	3.82	1.00

Table 5. Comparison of the results of the studied methods

Method	$\frac{\text{total time}_{C=3}}{\text{total time}_{C=2}}$	$\frac{\text{iterations}_{C=3}}{\text{iterations}_{C=2}}$	$\frac{\text{cycle time}_{C=3}}{\text{cycle time}_{C=2}}$
IN/GS	$>1.22\cdot10^{4}$	—	—
AA/CP	20.2	5.48	3.69
AA/LP	16.4	4.28	3.83

Table 6a. Comparison of linearization techniques for the cascade of three stages

Method	time [s]	$\frac{\text{total time}_{IN/GS}}{\text{total time}_{AA/CP}}$	cycles	$\frac{\text{cycles}_{IN/GS}}{\text{cycles}_{AA/CP}}$	cycle time [ms]	$\frac{\text{cycle time}_{IN/GS}}{\text{cycle time}_{AA/CP}}$	Cluster boxes
IN/GS	280.7	63.2	271491	160.9	1.03	0.39	3
AA/CP	4.44	1.00	1687	1.00	2.63	1.00	8

Table 6b. Comparison of pruning techniques for the cascade of three stages

Method	time [s]	$\frac{\text{total time}_{IN/GS}}{\text{total time}_{AA/CP}}$	cycles	$\frac{\text{cycles}_{IN/GS}}{\text{cycles}_{AA/CP}}$	cycle time [ms]	$\frac{\text{cycle time}_{IN/GS}}{\text{cycle time}_{AA/CP}}$	Cluster boxes
AA/CP	4.44	4.23	1687	5.68	2.63	0.74	8
AA/LP	1.05	1.00	297	1.00	3.54	1.00	1

Table 7a. Comparison of pruning techniques, over the small size initial box, for the cascade of four stages

Method	time [s]	$\frac{\text{total time}_{AA/CP}}{\text{total time}_{AA/LP}}$	cycles	$\frac{\text{cycles}_{AA/CP}}{\text{cycles}_{AA/LP}}$	cycle time [ms]	$\frac{\text{cycle time}_{AA/CP}}{\text{cycle time}_{AA/LP}}$	Cluster boxes
AA/CP	633.1	3332	116481	6131	5.43	0.54	3123
AA/LP	0.19	1.00	19	1.00	10.0	1.00	0

Table 7b. Comparison of pruning techniques, over the medium size initial box, for the cascade of four stages

Method	time [s]	$\frac{\text{total time}_{AA/CP}}{\text{total time}_{AA/LP}}$	cycles	$\frac{\text{cycles}_{AA/CP}}{\text{cycles}_{AA/LP}}$	cycle time [ms]	$\frac{\text{cycle time}_{AA/CP}}{\text{cycle time}_{AA/LP}}$	Cluster boxes
AA/CP	1132.2	364	210677	577	5.37	0.63	2496
AA/LP	3.11	1.00	365	1.00	8.52	1.00	1

Table 7c. Comparison of pruning techniques, over the large size initial box, for the cascade of four stages

Method	time [s]	$\frac{\text{total time}_{AA/CP}}{\text{total time}_{AA/LP}}$	cycles	$\frac{\text{cycles}_{AA/CP}}{\text{cycles}_{AA/LP}}$	cycle time [ms]	$\frac{\text{cycle time}_{AA/CP}}{\text{cycle time}_{AA/LP}}$	Cluster boxes
AA/CP	3934.0	14.0	721501	22.2	5.45	0.63	2560
AA/LP	281.5	1.00	32471	1.00	8.67	1.00	0

Table 8. Comparison of the results for the cascade with three stages and the cascade with four stages over the large size initial box

Method	$\frac{\text{total time}_{N=4}}{\text{total time}_{N=3}}$	$\frac{\text{cycles}_{N=4}}{\text{cycles}_{N=3}}$	$\frac{\text{cycle time}_{N=4}}{\text{cycle time}_{N=3}}$
AA/CP	886.0	427.7	2.07
AA/LP	268.1	109.3	2.45

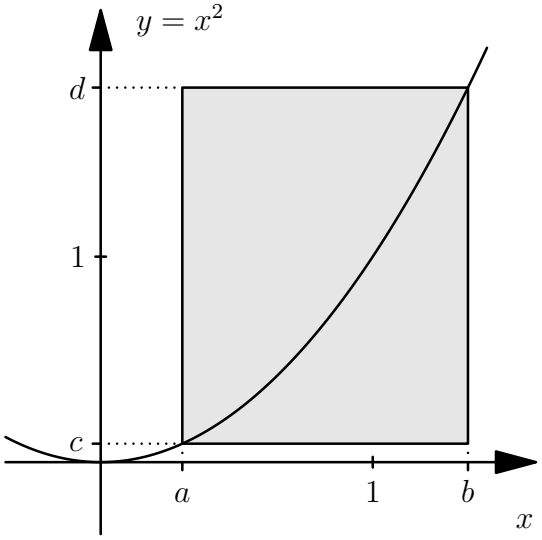

Fig. 1a.

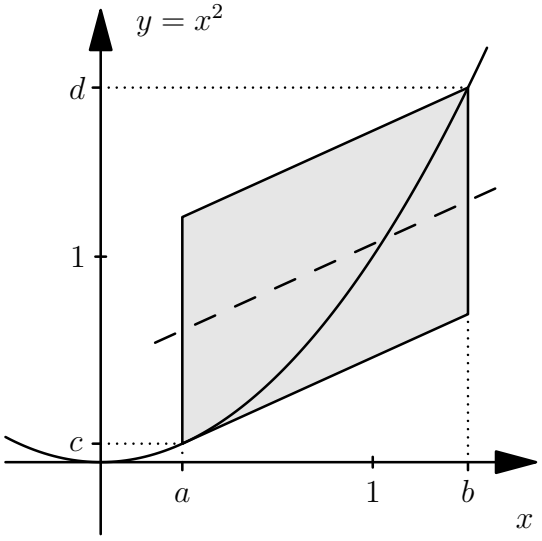

Fig 1b.

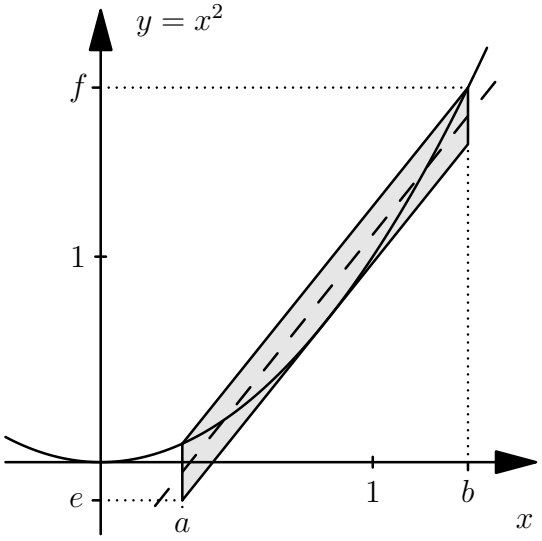

Fig. 1c.

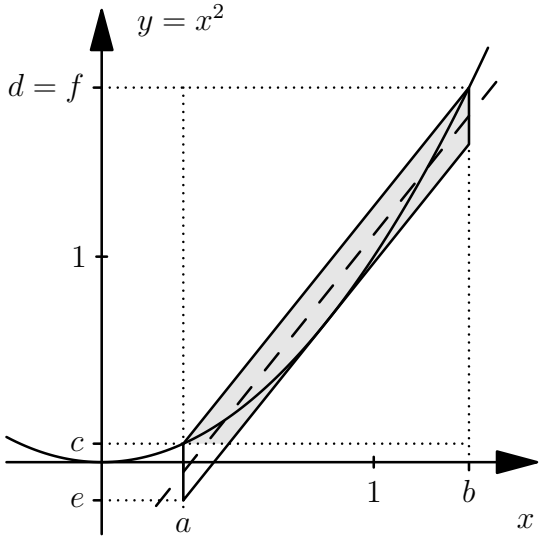

Fig. 1d.

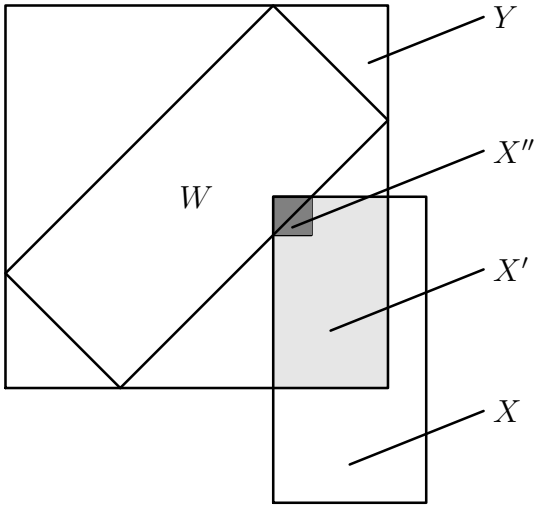

Fig. 2a.

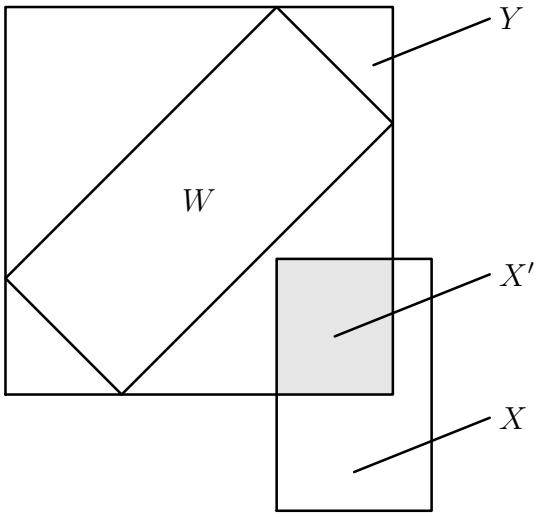

Fig. 2b.

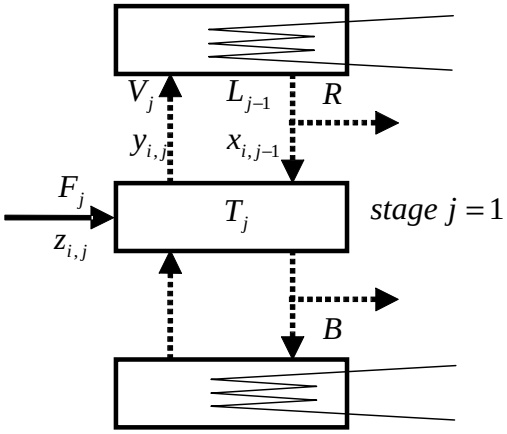

Fig. 3a.

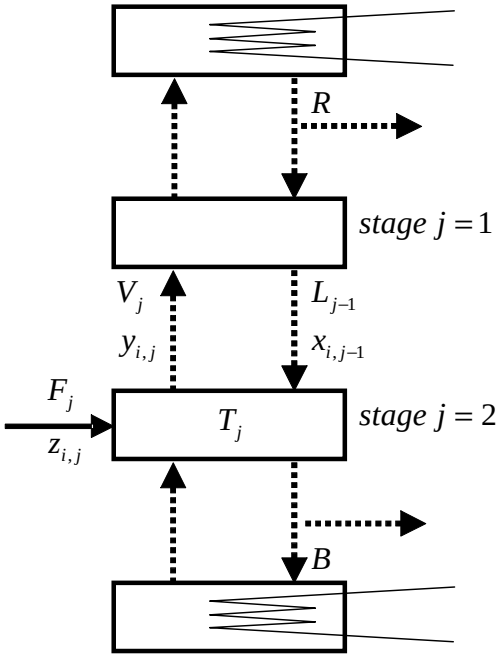

Fig. 3b.