Application of a subdivision algorithm for solving nonlinear algebraic systems

Fernanda de Castilhos Corazza, José Vladimir de Oliveira and Marcos Lúcio Corazza*

*Departamento de Engenharia de Alimentos, Universidade Regional Integrada, Av. Sete de Setembro, 1621, 99700-000, Erechim, Rio Grande do Sul, Brazil. *Author for correspondence. E-mail: mlcorazza@uricer.edu.br*

ABSTRACT. A subdivision algorithm is presented and applied to solving commonly found chemical engineering problems described by nonlinear algebraic systems. For this purpose, a web-based library available in the literature was used as the main source to select a wide class of one- and multidimensional problems, comprising phase and chemical equilibrium, conversion in tubular and continuous stirred tank reactors, material and energy balances, etc. The problems are classified according to the literature as low, intermediate and of high degree of numerical difficulty based on specific characteristics, like discontinuities in the functions, multiple solutions with the occurrence of false and unfeasible roots, and the presence of null derivative values. It is shown that the algorithm is efficient and robust, even for multidimensional problems of high numerical difficulty, allowing to find simultaneously all the feasible roots of nonlinear algebraic systems, naturally excluding false and unfeasible solutions, with a relatively low CPU time. These features make the algorithm an interesting alternative to deal with chemical engineering problems in contrast to some methods currently in the literature.

Key words: numerical analysis, nonlinear equations, multiple solutions, subdivision algorithm, simulation, chemical engineering processes.

RESUMO. Aplicação de um algoritmo de subdivisão para solução de sistemas de equações algébricas não-lineares. Um algoritmo de subdivisão é apresentado e aplicado à solução de problemas descritos por sistemas de equações algébricas não-lineares comumente encontrados na engenharia química. Uma biblioteca disponível, na literatura, foi utilizada como fonte principal para a seleção dos problemas a serem resolvidos com uma ou várias dimensões, compreendendo problemas de equilíbrio químico e de fases, conversão em reatores tubulares e contínuos, balanços material e energético, entre outros. Os problemas foram classificados pela literatura com grau de dificuldade numérica baixa, intermediária e alta, com base em características específicas como a existência de descontinuidades nas funções, múltiplas soluções com raízes falsas. O algoritmo mostrou-se eficiente e robusto, mesmo para problemas multidimensionais de alta dificuldade numérica, permitindo encontrar simultaneamente todas as raízes corretas (fisicamente possíveis) dos sistemas algébricos não-lineares, naturalmente excluindo soluções falsas com um tempo de CPU relativamente baixo. Estas características fazem deste algoritmo uma alternativa interessante para solucionar os problemas da engenharia química em contraste com alguns métodos atualmente disponíveis na literatura.

Palavras-chave: análise numérica, equações não lineares, múltiplas soluções, algoritmo de subdivisão, simulação, processos de engenharia química.

Introduction

The development of efficient iterative methods and strategies for obtaining all solutions of complex problems in chemical engineering processes is of unquestionable relevance for both academia and industry circles. In particular, chemical engineers are often interested in solving systems of nonlinear algebraic equations with high-dimensional order, involving a variety of numerical difficulties existing in this wide and fascinating (chemical engineering) science field. Despite some significant advances reached in the last years, there are still many challenges to overcome, such as the development of robust and easy-implementation algorithms able to simultaneous find all solutions of nonlinear algebraic systems. These remarkable challenges in this area are still greater when transcendental terms are encountered in these equations (Gritton *et al.,* 2001). The complex nonlinear nature of several problems in chemical engineering is commonly

described by phenomenological or empirical models represented by nonlinear algebraic equations. Consequently, the development of methods for solving this kind of mathematical problem is fundamental towards running accessible chemical engineering processes, and also for the proposition and establishment of new technologies, mainly those referring to process simulation, analysis, synthesis and optimization (Gritton *et al*., 2001; Shacham *et al*., 2002).

According to the literature, methods for determining zeros of a single nonlinear equation $f(x) = 0$ or of a system of nonlinear equations $F(x) = 0$, can be classified into three classes:

i) local methods. These methods have as main feature the need for an initial guess sufficiently close to the intended root, which is in fact its major limitation. The initial guess accuracy necessary for successful implementation of these methods varies with the function non-linearity degree (Shacham *et al.*, 2002). These methods nevertheless present the excellent property of quadratic convergence. To find all roots of nonlinear systems, several distinct initial guesses are generally required, and to each initial guess there is only one solution associated. The Newton (or Newton-Raphson) and quasi-Newton are the local methods most commonly employed (Press *et al*., 1992; Pernice and Walker, 1998; Kreyszig, 1999; Shacham *et al*., 2002).

ii) global methods. These methods are characterized by offering assured convergence, independently of how close the initial guesses are to the roots. Besides, they can be adapted to search multiple roots or solutions from a unique initial guess (Shacham *et al*., 2002). Within the class of global methods, one might call attention to the Homotopy Continuation Method (Davidenko, 1953; Keller, 1978; Wayburn and Seader, 1987; Kuno and Seader, 1988; Seader *et al*., 1990; Seider *et al.*, 1991) and the Improved Memory Method (Shacham, 1989; 1990). In brief, the Homotopy method is capable of finding the roots of a nonlinear equation, or of a system of nonlinear algebraic equations arising from combination of two functions: a function for which a zero is known or readily obtained, say $G(x)$, and other functions whose zeros (roots) are sought ($F(x)$). The roots of the equation or system of equations are obtained by tracking a path from known solutions of a given simpler arbitrary equation ($G(x)$) until finding the root of the $F(x)$ equation. In this method, multiple solutions can be found (Wayburn and Seader, 1987; Kuno and Seader, 1988; Seader *et al*., 1990; Seider *et al.*, 1991; Gritton, 2001). The Improved Memory Method (IMM), presented by Shacham (Shacham, 1989; 1990) can be extremely efficient and robust, provided the user has a very good idea of the approximate value of the solution. Compared with the IMM, the Homotopy Continuation Method is computationally hard; nevertheless a seeking interval is not necessary.

iii) interval methods. These methods are able to locate all roots from specified variable intervals. However, their implementation is hard, needing specific operators for any arithmetic operations, which are known as interval operators (Kerfott and Novoa, 1990; Kerfott *et al*., 1994; Hua *et al*., 1996; 1998; Gang Xu, 2001). Moreover, the analytic Jacobian must be known and implemented, which can be seen as an additional difficulty for its utilization, since not always are the equations differentiable or its derivation may be very onerous (Kolev, 1998).

In a general sense, one can notice that the alternatives mentioned above present numerical limitations or implementation difficulties. These facts have prompted permanent investigations dedicated to the development of new robust algorithms, and the improvements of existing ones.

Recently, Smiley and Chun (2001) presented a subdivision algorithm that allows one to locate simultaneously all roots of systems of nonlinear algebraic equations, from a given interval for variables. This characteristic allows classifying it as a global method. Another significant characteristic of the proposed method is that only conventional arithmetic operations are needed, making its implementation and application effortless. According to these authors, some of main features of the algorithm are its simplicity, robustness and reliability.

One of the first algorithms presented in the literature that makes use of a schematic subdivision was the Weyl algorithm (Weyl, 1921), which was applied to determine the zero of a single polynomial equation. The use of subdivision algorithms has been reported in other contexts, such as the generation of curves and surfaces in geometric design problems (Gregory, 1991). According to Smiley and Chun (2001), a subdivision algorithm was also used by Dellnitz and Hohmann (1997) to determine the attractor of a dynamic system. The reader may be interested in reading the report of Pan (1997), who presented a short history and recent advances on solving polynomial equations.

To our knowledge, application of subdivision algorithms in typical problems of chemical engineering has not been found in the specific technical literature. In this context, considering the potential of the method

and the need for reliable and robust mathematical tools to solve nonlinear problems, the aim of this work is to report the application of a subdivision algorithm to some commonly found chemical engineering problems, such as multiple steady state determination, reactor conversion and chemical and phase equilibrium calculations.

Material and methods

In general, a subdivision algorithm consists in starting from given intervals for the system variables, to apply a subdivision procedure for these intervals, generating congruent subintervals, and follow an evaluation test to check the existence of solutions in these subintervals. If the subinterval succeeds in the test, it is retained and continues in the procedure; otherwise it is discarded. After a finite number of subdivisions, a conventional method of local convergence (e.g., Newton-Raphson) can be used to find the roots. This may be done using the midpoint of the final subintervals, kept as initial guess for the conventional method of local convergence. In this way, depending on the number of subdivisions applied, the solution (or solutions) is confined in the retained intervals, which generally provides assured and efficient convergence of the local method employed.

According to Smiley and Chun (2001), the basic idea of the subdivision algorithm is: given an interval for each variable of the system ("rectangle") $R \in \Box^d$, find all values of $\mathbf{x}^* = \{\mathbf{x} \in \Box^d : \mathbf{F}(\mathbf{x}) = 0\}$, by successive subdivision in R; where d is the dimension of the problem and **F**(x) is the system of nonlinear algebraic equations.

In order to proceed with the subdivision of R, a partition sequence is necessary. A simple procedure to division of R to congruent sub-rectangles can be obtained by dividing R into two equal parts in each coordinate. Thus, for a rectangle set in $\Box^d$ there will be 2^d new generated rectangles or sub-rectangles. One can then name R of "parent" rectangle and the congruent sub-rectangles (from the division of R) of "children" rectangles. Then, for each "i" subdivision, R_{ij} children sub-rectangles are obtained, where "j" is the number of generated sub-rectangles. For each R_{ij} obtained sub-rectangle, a test for verifying the existence of roots of the system is applied. If the R_{ij} sub-rectangle passes the test, it is retained and will be a new parent rectangle, which will be in turn subjected to a new subdivision at $i = i + 1$.

A simple selection criterion for the sub-rectangles retaining can be the use of the minimum value of the Euclidean norm of F(x), $\|\mathbf{F}(x)\|$. For each value of $i \geq 1$ one can define:

$$\text{if } \min_{x \in R} \|\mathbf{F}(x)\| \leq 2^{-i}, \tag{1}$$

R_{ij} is retained; otherwise it is discarded.

However, the requirement of finding the global minimum of $\|\mathbf{F}(x)\|$ for each tested rectangle may be unviable, mainly for multidimensional systems. Other selection criterion can then be defined as:

$$\text{if } \left\|\mathbf{F}(\mathbf{x}_{ij})\right\| \leq 2^{-i} + \rho_j L \tag{2}$$

R_{ij} is retained; otherwise it is discarded.

Here, ρ_j is the half largest side of the rectangle "j", defined by $\rho_j = \left\|a_k - b_k\right\|_{\infty, j}$, $k = 1,..,d$ and $\|\mathbf{F}(x_{ij})\|$ is the Euclidean norm of F(x) in the midpoint (x_{ij}) set in R_{ij} rectangle. This selection criterion requires the knowledge of L, the Lipschitz's constant, which is defined for each tested problem.

A third selection criterion can be thought as:

$$\text{if } \left\|\mathbf{F}(\mathbf{x}_{ij})\right\| \leq 2^{-i} + \tau_{ij} \tag{3}$$

R_{ij} is retained; otherwise it is discarded.

In this equation,

$$\tau_{ij} = \frac{1}{2}\sum_{k=1}^{d}\left(\max_{\mathbf{y} \in R_{ij}}\left\|\frac{\partial \mathbf{F}(\mathbf{y})}{\partial y_k}\right\|_2\right)|b_k - a_k| \tag{4}$$

where **y** is a vector of random points in R_{ij}, and a_k and b_k are, respectively, the lower and upper bounds of the R_{ij} rectangle in the "k" dimension (or coordinate).

According to Smiley and Chun (2001) the selection criterion of Equation (3) was the one that presented the best results concerning the acceleration of the rectangles exclusion process. For this reason, this was the selection criterion adopted throughout this work.

For obtaining the coordinates of each "child" rectangle, the following scheme (or formulation) is proposed in this work:

$$\begin{cases} a_k = A_k + \boldsymbol{\alpha}_{m,d} v_k \\ b_k = a_k + v_k \end{cases} \qquad m = 1,...,2^d;\ k = 1,...,d \tag{5}$$

and,

$$v_k = \frac{1}{2}(B_k - A_k) \qquad k = 1,...,d \tag{6}$$

where $\alpha_{m,d}$ is a matrix of 0 (zero) and 1 (one) elements

combined in such way that, at each "*m*" iteration, "children" rectangles coordinates are obtained by Equation (5); A_k and B_k are, respectively, the lower and upper bounds of the considered "parent" rectangles for each "*k*" variable; v_k is the midpoint coordinate of the "parent" rectangle for "*k*" variable, a_k and b_k are the lower and upper coordinates for the new "child" rectangle in "*k*" variable.

Furthermore, the utilization of an appropriate coverage strategy can be significant for accelerating the rectangles' exclusion, mainly for complex problems. The coverage can be made after a finite number of subdivisions; the retained rectangles are revaluated and those not presenting common coordinates are defined as new rectangles.

In the present work, only rectangles with one adjacent side, e.g., rectangles that have the same coordinates, in at least one dimension, are used to define a new rectangle. After the coverage, the new rectangles are tested employing the selection criterion. In Figure 1, the coverage strategy is illustrated. Figure 1(a) shows the retained rectangles after the *i-th* subdivision, and Figure 1(b) depicts the redefined rectangles after the coverage.

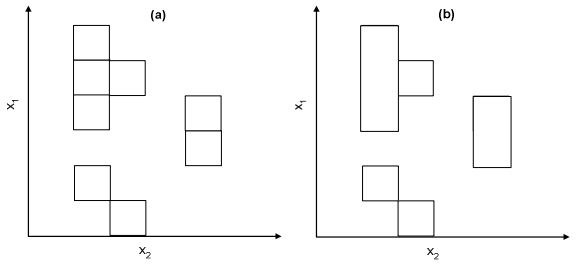

Figure 1. (a) Rectangle saved after "*i*" subdivisions; **(b)** Rectangles presented in **(a)** after the coverage.

In the present work, a computational code of the subdivision algorithm, with the denomination of SubDivNL, was implemented in Fortran 90 language. The basic version of this code is described in the algorithm presented in below. In this algorithm, *iRan* is the sample number for each R_{ij} rectangle, *iMax* is the maximum number of subdivisions, *iCov* is the maximum number of coverage and *M* is the number of retained rectangles for each subdivision "*i*".

i) Given d, A_k e B_k (k=1,..,d), iRan, iMax, F(x), iCov
ii) Set $M = 1$ (initialize with a "parent" rectangle)
iii) For l=1,...,iCov do:
  a. For $i=1,...,iMax$ do:
    b.i. Set $m = 0$
    b.ii. For $j=1,...,M*2^d$ do:
      1. Apply Eqs. (5) and (6) to obtain the coordinate for each R_{ij} "child" rectangle
      2. Get *nRan* random samples into R_{ij}
      3. Evaluate $\left\|\mathbf{F}\left(\mathbf{x}_{ij}\right)\right\|$ for each random number into R_{ij}
      4. Evaluate τ_{ij} (Eq. 4)
      5. If $\left\|\mathbf{F}\left(\mathbf{x}_{ij}\right)\right\| \le 2^{-i} + \tau_{ij}$ (criterion 3) then
        a. Retain the R_{ij} rectangle
        b. Set $m = m + 1$
      6. Else discard it
    b.iii. Set $M = m$
  b. Evaluate the adjacent rectangles (coverage)
  c. Reset *M* (*M* = number of new rectangles)
iv) Use the Newton method to find $\mathbf{x}^* = \left\{\mathbf{x} \in \Re^d : \mathbf{F}\left(\mathbf{x}\right) = 0\right\}$, applying the midpoint of each R_{ij} as initial guess
v) Check the solutions (roots) found
vi) Show all solutions (roots)

In order to illustrate the robustness and reliability of the proposed algorithm in finding all real solutions (roots) of nonlinear algebraic equations, some case studies are presented. Initially, with the aim to illustrate the procedure, a generic problem is considered. Afterwards, authentic, chemical engineering problems are investigated.

Results and discussion

For the solution of the following illustrative problem and also for the subsequent applications, a Personal Computer was used, with a Pentium IV processor, 2.66 GHz and 512 MB of RAM memory. For all of the considered problems, the midpoint of each selected final rectangle was used as the initial guess for Newton's method (subroutine mnewt presented by Press *et al*., 1992). Besides, it may be important to emphasize that throughout this work, the adopted criterion for coverage was to add rectangles with adjacent sides to form a new rectangle. Also, a scheme for variable sampling as a function of the number of subdivisions (*i*) is proposed in the case of high (greater than two) dimensional problems, resulting in significant reduction in CPU time.

Illustrative problem

Let us consider the following system of nonlinear algebraic equations (two-dimensional problem):

$$f_1(w_1, w_2) = w_1^2 w_2 + \sin w_1$$
$$f_2(w_1, w_2) = \ln w_2 + w_1 w_2 \quad (7)$$

In order to locate the roots of this problem, the following intervals for the variables w_1 [-10.0, 10.0] and variable w_2 [1.0, 20.0] were taken. The maximum number of subdivisions was $iMax = 5$, and the coverage number $iCov = 2$.

From the retained rectangles after $i = 5$ for $iCov = 2$, and using the midpoint in each final rectangle as the initial guess, Newton's method was then employed to find the roots of the problem. The distinct solutions found (roots), with four decimals of significance are presented in Table 1 along with the roots obtained with the application of software Maple 9.5 (Waterloo Inc.®). Notice that, for the latter case different and multiple initial guesses were needed to find the solutions of the problem.

Table 1. Roots of the illustrative problem applying the SubDivNL algorithm and Maple 9.5 software.

Roots Number	SubDivNL algorithm	Maple 9.5	
	Roots Values (w_1, w_2)	Initial Guess (w_1, w_2)	Roots Values (w_1, w_2)
Root 1	0.224811×10^{-11}, 1.0000	1. (1.0, 2.0) 2. (5.0, 20.0) 3. (10.0, 30.0)	0.0000, 1.0000
Root 2	-36.7786×10^{-2}, 265.8089×10^{-2}	4. (-5.0, 2.0) 5. (10.0, -30.0) 6. (-10.0, -30.0)	-36.7786×10^{-2}, 265.8089×10^{-2}

From these results one can see that the subdivision algorithm was efficient in finding simultaneously all the roots (solutions) for the problem considered. This is an important characteristic of the method, an advantage relatively to local convergence methods. The CPU processing time for solving the illustrative problem with the present algorithm using the PC configuration described before was only 0.10 s.

Application to chemical engineering problems

Shacham *et al.* (2002) has presented a pool of systems of nonlinear algebraic equations, typical problems of chemical engineering science, with distinct degree of difficulty, from one to multidimensional systems, which are quite useful to test algorithm performance and numerical methods intended to solve nonlinear algebraic equations. In this work, the mentioned web-based library was used as a reference to compare the results from the application of the subdivision algorithm SubDivNL. Details about the problems considered in this work can be found in the original reference (Shacham *et al*., 2002). In the present work, the variables units and constants, as well as the whole nomenclature for each problem were kept and used as presented in the original reference.

One-dimensional problems

In this section, some one-dimensional problems are solved. A brief description of each one, as well as the parameters employed in the algorithm SubDivNL are presented. The results found for the one-dimensional problems obtained from this work are shown in Table 2, along with a comparison with literature. This table presents the roots found by the SubDivNL algorithm and the roots values and intervals used in the literature.

Table 2. Results for one-dimensional problems.

Roots Number	SubDivNL algorithm	Literature (Shacham *et al*., 2002)	
	Roots Values	Interval	Roots Values
P1 Problem: variable (P)			
Root 1	25.7439	Xmin = 1.0	25.7440
Root 2	78.9054	Xmax= 100.0	78.9054
P2 Problem: variable (X)			
Root 1	0.058654	Xmin = 0.0	0.058655
Root 2	0.600323	Xmax = 1.0	0.600323
P3 Problem: variable (X)			
Root 1	0.999984	Xmin = 0.75 Xmax = 1.25	0.999252

P1 Problem: pressure drop in a converging-diverging nozzle

The P1 problem deals with the pressure drop in a converging-diverging nozzle. The model for this problem comprises an implicit equation in pressure (P). According to Shacham *et al*. (2002), this problem presents a low degree of difficulty and a discontinuity for negative values of P, resulting in a constraint to this variable ($P \geq 0$). Another feature of this function is the existence of a stationary (maximum) point, i.e., null derivative, hindering the convergence of any method that uses derivative values. The following values were used for the SubDivNL algorithm: the interval of the unknown variable (P) was [0.0, 100.0], the subdivision number $i = 14$, and the coverage was set equal to zero ($iCov = 0$). The subdivision algorithm found the two roots for this problem with a CPU time of 0.05 s.

P2 Problem: chemical equilibrium I

Problem P2 refers to chemical equilibrium, with an implicit equation in the conversion (X). Therefore, this variable is restricted to [0.0, 1.0] interval. This is the seek interval utilized in the SubDivNL algorithm. Also, a subdivision number $i = 14$ without covering ($iCov = 0$) was adopted. This problem was classified by Shacham *et al*. (2002) as of intermediate difficulty level. The function is undefined for X = 0.26667, which makes difficult the convergence of methods based on derivative values. Again, as it can be verified in Table 2, this kind of behavior did not obstruct the subdivision algorithm to find all the roots with a low CPU time, equal to 0.05 s.

P3 Problem: equilibrium conversion in a tubular reactor

This problem consists in the conversion evaluation in a tubular reactor. The interval used for the conversion (X), which is the independent variable, was [0.5, 1.5], together with i = 10 and $iCov$ = 3. The required CPU time for solving the problem was 0.36 s. It may be pertinent to mention that, for this problem, the function values are very low (in the order of 10^{-8}) in the investigated interval, and there is a discontinuity point close to the root, making its determination difficult. Nevertheless, the SubDivNL algorithm did not encounter severe difficulties in solving this problem and found its two roots simultaneously, in spite of being classified with high level of numerical difficulty by the literature (Shacham *et al*., 2002). This fact may have been the cause of the increase in the computational time in relation to the problems presented previously.

For all one-dimensional problems considered in this section, the SubDivNL algorithm showed to be efficient in determining all the roots with low CPU time, independent on the difficulty degree.

Two-dimensional problems

Table 3 presents the results obtained in solving three investigated two-dimensional problems. Similarly to Table 2, results from the literature (Shacham *et al*., 2002) and those obtained with the application of the SubDivNL algorithm are compared. It can be observed from this table that literature points out the use of different initial guesses to find the roots.

Table 3. Results for two-dimensional problems.

Roots Number	SubDivNL algorithm		Literature (Shacham *et al.*, 2002)	
	Roots Values		Initial Guess	Roots Values
P4 Problem: variables (X1, X2)				
Root 1	0.600323, -3.5799		1. (0.0, 1.0) 2. (0.5, 0.1)	[a] 0.600323, -3.5799
Root 2	0.586546×10^{-1}, 0.867438		3. (0.0, 1.0) 4. (0.5, 0.1)	0.586546×10^{-1}, 0.867438
P5 Problem: variables (X, T)				
Root 1	0.963868, 346.1637		1. (1.0, 400.0) 2. (0.0, 300.0) 3. (0.5, 320.0) 4. (0.0, 350.0)	0.963868, 346.1636
P6 Problem: variables (X1, X2)				
Root 1	0.757396, 0.021302		1. (0.9, 0.5) 2. (0.5, 0.5) 3. (0.4, 0.005)	0.757396, 0.021302
Root 2	-	-	4. (0.6, 0.1)	[a] 1.098984, -0.149492
P7 Problem: variables (A, B)				
Root 1	0.758077, 1.124904		1. (0.5, 0.5) 2. (1.0, 1.0) 3. (5.0, 5.0)	0.758077, 1.124903
Root 2	-	-	4. (8.0, 2.0)	[b] -4.2476×10^{-7}, 8.9296×10^{-7}

[a]Unfeasible solution. For problem P6, most methods do not converge from the 1st initial guess and will converge to the unfeasible solution after the 4th initial guess.
[b]False solution.

P4 Problem: chemical equilibrium II

The P4 problem is composed by two implicit equations in X_1 and X_2, the conversions of components 1 and 2, respectively. In the determination of the roots with the SubDivNL algorithm, 6 subdivisions were used (i = 6) and just one coverage ($iCov$ = 1). For the purpose of illustration, we first assumed for the SubDivNL the interval for both variables as [-4.0, 4.0]. The SubDivNL algorithm found the two roots starting from this interval, with a low CPU time, equal to 0.21 s. Conversely, for the results obtained by the literature, several initial estimates were necessary for the determination of these two roots. According to Shacham *et al.* (2002), this problem can be classified as of intermediate level. Note that the first and second initial guesses converge to the first root, and the third and fourth guesses converge to the second root.

For this problem, since the variables are constrained into [0.0, 1.0] interval, the first root reported in the literature is in fact an unfeasible solution. With the present algorithm (SubDivNL), such root can be avoided by simply restricting the interval of the variables (X1, X2) to [0.0, 1.0]. Starting from this interval, only the second (feasible) root is found (X1 = 0.058655, X2 = 0.867438).

P5 Problem: steady state material and energy balances on a reactor

The equations of the P5 problem refer to the material and energy balances in a reactor in steady state, resulting in two nonlinear equations in the variables conversion (X) and temperature (T). For this problem, only one root was found by the SubDivNL algorithm, as can be verified in Table 3. The subdivision algorithm used the seek intervals of [0.0, 1.0], [100.0, 800.0], with no coverage ($iCov$ = 0) and ten subdivisions (i = 10). The CPU time for the solution of this problem with the SubDivNL algorithm was 0.40 s, which is low, mainly considering the numerical difficulty for convergence of the local method to the root.

The cited literature classifies this problem as of high numerical difficulty level and the authors use four initial guesses for the determination of the root. In addition, these authors report that, due to the occurrence of singular points, most of the numerical methods available in the literature do not converge starting from the second and third initial guesses.

P6 Problem: conversion in a chemical reactor

Problem P6 deals with the conversion in a reactor and is classified, according to the literature (Shacham *et al*., 2002), as of high degree of numerical difficulty. The problem is composed by

two nonlinear equations with variables X_1 and X_2, which are constrained into [0.0, 1.0] interval. The parameters used in the roots search (solutions) were: subdivision number i = 5, coverage number *iCov* = 2 and variables intervals for X_1 and X_2 of [0.1, 0.9], [0.001, 1.0], respectively. The results presented in the literature for this case show the same root, but with three different initial guesses. One should also note from Table 3 that, the convergence from the fourth guess leads to an unfeasible solution, not found by the SubDivNL algorithm, since that value is out of the interval used.

This example calls attention to an important characteristic of the SubDivNL algorithm, i.e., the method prevents convergence for solutions that are out of the specified interval, which means to respect the restrictions for the variables involved.

P7 Problem: van Laar equation coefficients from azeotropic data (ethanol + n-heptane system)

Problem P7 refers to the calculation of the coefficients of van Laar's equation, parameters A and B, from an experimental azeotropic point for the system ethanol + n-heptane. The parameters used in the SubDivNL algorithm were in the interval of [0.0, 2.0] for both variables, number of subdivisions i = 6 and number of coverage *iCov* = 1. With this specification, the algorithm found only one root, with a CPU time of 0.2 s.

According to the literature, this problem can be classified as of high degree of numerical difficulty, as it presents a discontinuity, the system of equations is not defined at A=0 and B=0, and the solution is restricted to A, B > 0. For this case, the literature points out the determination of two roots, the first one obtained from three different initial guesses giving the same root found by the SubDivNL algorithm (see Table 3). The other root reported in the literature, not found by the SubDivNL algorithm, is a false solution because it does not obey the restriction of the variables. This problem illustrates again the significant feature of the SubDivNL algorithm in keeping the solutions inside (restricted to) the specified interval.

For the two-dimensional problems presented, the SubDivNL algorithm was able to find simultaneously all the roots, from only one seek interval, without presenting convergence problems. This means that common difficulties arising from the application of conventional algorithms reported in the literature, such as the need for using several initial guesses to find different roots and convergence problems, are easily overcome by the SubDivNL algorithm. Moreover, the algorithm did not allow the convergence to unfeasible and false solutions. This is a relevant feature of the SubDivNL algorithm, particularly to chemical engineering applications, where often not all the roots of nonlinear algebraic systems are true solutions of the engineering problem.

P8 Problem: phase stability test applied to binary liquid-liquid equilibrium

Liquid-liquid equilibrium calculations can be considered a class of problems of high degree of numerical difficulty because multiple solutions (roots) can be generally found (Baker *et al*., 1982; Corazza *et al*., 2004; Rangaiah, 2001; Gecegormez and Demirel, 2005).

For the stability test, we have employed the criterion of the Gibbs surface tangent plane distance function (*TDP*):

$$TPD(\mathbf{y}) = \sum_{k}^{nc} y_k \left\{ \ln(\gamma_k y_k) - \ln(\gamma_k^* z_k) \right\} \quad (8)$$

where, γ_k and γ_k^* are the activity coefficients of component "k" for new tried and tested phase, respectively, y_k and z_k are the compositions of new (tried) and tested phase, respectively, and *nc* stands for number of components. As the solutions for Equation (8) are stationary points, the system can be solved by differentiating with respect to y_k to obtain the following system of non-linear algebraic equations:

$$\left\{ \ln(\gamma_k \gamma_k) - \ln(z_k \gamma_k^*) \right\} - \left\{ \ln(y_{nc} \gamma_{nc}) - \ln(z_{nc} \gamma_{nc}^*) \right\} = 0 \; ; \; k = 1, \ldots, nc-1 \quad (9)$$
$$\sum_{k}^{nc} y_k = 1$$

In order to test the subdivision algorithm for liquid-liquid phase stability analysis, the binary system water(1)+butanenitrile(2) was selected from the literature (Gecegormez and Demirel, 2005), and the NRTL model was used to describe the real liquid solution. The system was chosen due to its non easy numerical solution. The parameters of the NRTL model were taken from (Letcher and Naicker, 2001). The SubDivNL algorithm was employed using the interval of [0.0, 1.0], for composition of tested liquid phases (**y**), with six subdivisions (i = 6), and no coverage (*iCov* = 0). Table 4 presents the stability test calculations at some distinct global compositions specified, where one can see that the same solutions (roots) were found by the SubDivNL algorithm compared to the interval analysis method, but with a notably faster CPU (Computer Process Unit) time.

Table 4. Results for Problem P8: stability test analysis applied to liquid-liquid equilibrium of water(1)+butanenitrile(2) at 298.15 K.

Input (z_1, z_2)	SubDivNL			IN/GB* (Gecegormez and Demirel, 2005)		
	Roots (y_1, y_2)	TPD (y)	CPU (s)	Roots (y_1, y_2)	TPD (y)	CPU (s)
(0.15, 0.85)	(0.1500, 0.8500)	1.0000×10^{-9}	0.08	(0.1500, 0.8500)	0.0000×10^{-0}	26
	(0.5810, 0.4190)	4.2080×10^{-2}		(0.5810, 0.4190)	4.2080×10^{-2}	
	(0.9980, 2.0467×10^{-3})	-0.2230×10^{-0}		(0.9979, 2.0467×10^{-3})	-0.2230×10^{-0}	
(0.20, 0.80)	(0.2000, 0.8000)	1.0000×10^{-8}	0.10	(0.2000, 0.8000)	0.0000×10^{-0}	24
	(0.4835, 0.5165)	1.0652×10^{-2}		(0.4835, 0.5165)	1.0652×10^{-2}	
	(0.9981, 1.8634×10^{-3})	-0.2942×10^{-0}		(0.9981, 1.8634×10^{-3})	-0.2942×10^{-0}	
(0.25, 0.75)	(0.2500, 0.7500)	1.0000×10^{-9}	0.10	(0.2500, 0.7500)	0.0000×10^{-0}	25
	(0.4034, 0.5966)	1.6082×10^{-3}		(0.4034, 0.5966)	1.6082×10^{-3}	
	(0.9982, 1.7843×10^{-3})	-0.3253×10^{-0}		(0.9982, 1.7843×10^{-3})	-0.3253×10^{-0}	
(0.30, 0.70)	(0.3000, 0.7000)	1.0000×10^{-9}	0.10	(0.3000, 0.7000)	0.0000×10^{-0}	27
	(0.3407, 0.6593)	2.9594×10^{-5}		(0.3407, 0.6593)	2.9584×10^{-5}	
	(0.9982, 1.7564×10^{-3})	-0.3359×10^{-0}		(0.9982, 1.7564×10^{-3})	-0.3359×10^{-0}	
(0.40, 0.60)	(0.2524, 0.7476)	-1.4308×10^{-3}	0.09	(0.2524, 0.7476)	-1.4309×10^{-3}	26
	(0.4000, 0.6000)	1.0000×10^{-8}		(0.4000, 0.6000)	0.0000×10^{-0}	
	(0.9982, 1.7820×10^{-3})	-0.3276×10^{-0}		(0.9982, 1.7821×10^{-3})	-0.3276×10^{-0}	
(0.45, 0.55)	(0.2195, 0.7805)	-5.5825×10^{-3}	0.09	(0.2195, 0.7805)	-5.5825×10^{-3}	25
	(0.4500, 0.5500)	1.0000×10^{-8}		(0.4500, 0.5500)	0.0000×10^{-0}	
	(0.9982, 1.8241×10^{-3})	-0.3153×10^{-0}		(0.9982, 1.8242×10^{-3})	-0.3153×10^{-0}	
(0.50, 0.50)	(0.1909, 0.8091)	-1.4010×10^{-2}	0.09	(0.1909, 0.8091)	-1.4010×10^{-2}	25
	(0.5000, 0.5000)	1.0000×10^{-9}		(0.5000, 0.5000)	0.0000×10^{-0}	
	(0.9981, 1.8865×10^{-3})	-0.2991×10^{-0}		(0.9981, 1.8865×10^{-3})	-0.2991×10^{-0}	

* **I**nterval Analysis Method.

Three-dimensional problems

Table 5 presents the results obtained with SubDivNL algorithm and those available in the literature (Shacham *et al.*, 2002) for the three-dimensional problems considered here. As one could expect, solution of these problems with the SubDivNL algorithm required longer CPU time compared to two-dimensional examples, due probably to the greater dimension of the problems and the consequent greater coverage number adopted to root seeking.

P9 Problem: steady state operation of a CSTR reactor

The equations of the first three-dimensional problem describe the steady state operation of an adiabatic CSTR. Problem P9 takes into account three equations with concentrations of A (C_A) and B (C_B), and the temperature (T) as variables. The intervals used by the SubDivNL algorithm were [0.0, 4.0], [0.0, 4.0], and [300.0, 800.0], respectively. In addition, six subdivisions (i = 6) and no coverage ($iCov$ = 0) were adopted. The algorithm spent 3.09 s for the determination of the five roots, a more CPU time-consuming compared to previous problems. This is a consequence of the larger dimension of this problem, and also due to the greater number of rectangles retained during the execution of SubDivNL algorithm required to assure simultaneous determination of the five roots.

The CPU time can increase rapidly with the increase of dimension of the problem as well as with the number of roots. Nevertheless, as mentioned by Smiley and Chun (2001), for small rectangles (after some exclusion) a greater number of sampling can be avoided. Then, in this work, we propose the scheme shown in Figure 2 to sample the random number in each children rectangle.

Table 5. Results for three-dimensional problems.

Roots Number	SubDivNL algorithm	Literature (Shacham *et al.*, 2002)	
	Roots Values	Initial Guess	Roots Values
P9 Problem: variables (C_A, C_B, T)			
Root 1	2.7873, 0.212680, 310.2126	(3.0, 0.0, 300)	2.7873, 0.212679, 310.2126
Root 2	2.3805, 0.619577, 333.4925	(3.0, 0.0, 350)	2.3804, 0.619577, 333.4925
Root 3	0.126396, 2.8499, 462.5692	(3.0, 0.0, 400)	0.126397, 2.8499, 462.5692
Root 4	0.380121×10^{-02}, 1.7136, 594.0274	(3.0, 0.0, 600)	0.380106×10^{-02}, 1.7136, 594.0274
Root 5	0.379757×10^{-03}, 0.688892, 691.6242	(3.0, 0.0, 700)	0.379757×10^{-04}, 0.688892, 691.6242
P10 Problem: variables (T, X, T_j)			
Root 1	573.8548, 0.521391, 537.2534	[a] -	537.8548, 0.521391, 537.2534
Root 2	590.3498, 0.330187, 585.7298	[a] -	590.3498, 0.330187, 585.7298
Root 3	671.2278, 0.354669×10^{-01}, 660.4162	[a] -	671.2783, 0.0354195, 660.4629
P11 Problem: variables (x_1, y_1, α)			
Root 1	0.226975×10^{-01}, 0.977303, 0.532268	(0.9, 0.4, 0.6) (0.5,. 0.1, 0.9)	0.226975×10^{-01}, 0.977303, 0.532268
Root 2	-	- (0.5, 0.0, 1.0)	[b] $-7.979065 \times 10^{-13}$, 2.545161×10^{-10}, -9.143888×10^{09}
Root 3	-	- (0.9, 0.5, 0.5)	[c] 0.686757, 0.313243, 1.470821

[a]Not presented in the original reference.
[b]False solution. [c]Unfeasible solution.

After this modification (variable sampling as a function of subdivision level *i*), the CPU time required in Problem P9 was decreased to 1.98 s. Thus, the variable sampling as proposed in Figure 2 can help reducing significantly the computational time for more complex problems (higher dimensional problems and greater number of roots).

```
For iRan = 5 do
    If (d ≤ 2) then
        nRan = d x iRan
    If (d > 2) then
        If (i ≤ 3) then
            nRan = d x iRan
        If (3 < i ≤ 5) then
            nRan = d – 1
        If (i > 5) then                nRan = 1
```

Figure 2. Proposed scheme for sampling *nRan* random numbers inside the rectangles.

The roots found by the literature and by the SubDivNL algorithm are shown in Table 5, where one should noted that for determining all the roots it was necessary to run the algorithm five times arriving from distinct initial guess, while the SubDivNL algorithm found all the roots from only one interval and just one execution.

P10 Problem: multiple steady states in a cooled exothermic CSTR reactor

This problem considers the determination of multiple steady state conditions in a cooled exothermic CSTR reactor. The variables are temperature of the reaction medium (T), conversion (X) and temperature of the cooled jacket (T_j). The SubDivNL algorithm was employed using the intervals of [300.0, 700.0], [0.0, 1.0], and [300.0, 700.0] for T, X and T_j, respectively, with six subdivisions (i = 6), and no coverage ($iCov$ = 0). Three roots were found in this work with a CPU time of 5.23 s. This problem is classified by Shacham *et al*. (2002) as of low difficulty degree. For this problem, the initial guesses used by the literature are not provided in the above-mentioned reference.

P11 Problem: two-phase flash of a nonideal binary mixture (isobutanol-water)

This problem refers to a flash of a nonideal binary mixture. The variables are the mole fractions of one of the components in the vapor (y_1) and liquid (x_1) phases and the fraction of one of the phases (α); these variables are of course constrained in the [0.0, 1.0] interval. The parameters used in the SubDivNL algorithm were: five subdivisions (i = 5), three coverage ($iCov$ = 3) and variables intervals of [$0.1x10^{-05}$, 1], [$0.1x10^{-05}$, 1] and [0.0, 1.0], respectively, for y_1, x_1 and α. The SubDivNL algorithm found one root with a CPU time of 2.77 s. In Table 5 the three roots reported by the literature for this problem are presented: one of them is the same determined by the SubDivNL algorithm and the other ones are false or unfeasible solutions.

P12 Problem: phase stability test applied to ternary liquid-liquid equilibrium

The formulation of phase stability test for a ternary liquid-liquid system is the same presented previously. In order to test the subdivision algorithm, the ternary system water(1)+critic acid(2)+butan-2-ol(3) was chosen (Letcher and Naicker, 2001; Gecegormez and Demirel, 2005), with the NRTL parameters taken from Litomen *et al*. (2001).

The SubDivNL algorithm was employed using the interval of [0.0, 1.0], for composition of tested liquid phases (y), with six subdivisions (i = 6), and no coverage ($iCov$ = 0). From Table 6, some stability test calculations, at distinct global compositions specified are presented.

From the results shown in Table 6, it can be verified that the subdivision algorithm can be used to perform liquid-liquid stability test calculations, and it arises as an interesting alternative for phase equilibrium calculations.

In general, one can see that the present algorithm was able to find all the multiple solutions for the tested three-dimensional problems without the need for specifying initial guesses. In contrast to the results available in the literature, unfeasible and false solutions were avoided when the SubDivNL was employed. Moreover, the SubDivNL algorithm was capable of finding all of the feasible solutions with only one execution, whereas the methods available in the literature require in most cases several initial guesses and executions for the determination of all roots.

Table 6. Results for Problem P12: stability test analysis applied to liquid-liquid equilibrium of water(1) +critic acid(2)+butan-2-ol(3) at 298.15 K.

Input (z1, z2, z3)	SubDivNL			IN/GB[#] (Gecegormez and Demirel, 2005)	
	Roots (y_1,y_2, y_3)	TPD(y)	CPU (s)	Roots (y_1,y_2, y_3)	TPD(y)
(0.20, 0.05, 0.75)	(0.1752, 2.7558×10^{-2}, 0.7972)	-2.3698×10^{-4}	15.2	(0.1752, 2.7558×10^{-2}, 0.7972)	-2.3699×10^{-4}
	(0.2000, 0.0500, 0.7500)	1.0000×10^{-8}		(0.2000, 0.0500, 0.7500)	0.0000 × 10–0
	(0.2530, 0.1748, 0.5722)	-5.9447×10^{-3}		(0.2530, 0.1748, 0.5722)	-5.9447×10^{-3}
(0.15, 0.10, 0.75)	(8.3523×10^{-2}, 7.0909×10^{-3}, 0.9094)	-2.1738×10^{-2}	15.2	(8.3523×10^{-2}, 7.0909×10^{-3}, 0.9094)	-2.1739×10^{-2}
	(0.1500, 0.1000, 0.7500)	1.0000×10^{-8}		(0.1500, 0.1000, 0.7500)	0.0000 × 10–0
	(0.1544, 0.2908, 0.5548)	-1.7323×10^{-2}		(0.1544, 0.2908, 0.5548)	-1.7323×10^{-2}
(0.20, 0.10, 0.70)	(0.1257, 1.1877×10^{-2}, 0.8624)	-1.0321×10^{-2}	17.1	(0.1257, 1.1877×10^{-2}, 0.8624)	-1.0321×10^{-2}
	(0.2000, 0.1000, 0.7000)	1.0000×10^{-9}		(0.2000, 0.1000, 0.7000)	0.0000 × 10–0
	(0.2177, 0.1838, 0.5985)	-1.8376×10^{-3}		(0.2177, 0.1838, 0.5985)	-1.8376×10^{-3}
(0.05, 0.15, 0.80)	(1.8972×10^{-2}, 2.1628×10^{-3}, 0.9789)	-9.0052×10^{-2}	18,0	(1.8972×10^{-2}, 2.1628×10^{-3}, 0.9789)	-9.0052×10^{-2}
	(3.6280×10^{-2}, 0.3989, 0.5649)	-3.6271×10^{-2}		(3.6280×10–2, 0.3989, 0.5649)	-3.6271×10^{-2}
	(0.0500, 0.1500, 0.8000)	1.0000×10^{-9}		(0.0500, 0.1500, 0.8000)	0.0000 × 10–0
(0.10, 0.20, 0.70)	(4.1828×10^{-2}, 2.8004×10^{-3}, 09554)	-8.2501×10^{-2}	14,3	(4.1828×10^{-2}, 2.8004×10^{-3}, 0.9554)	-8.2501×10^{-2}
	(9.6537×10–2, 0.2567, 0.6467)	-4.2772×10^{-4}		(9.6537×10–2, 0.2567, 0.6467)	-4.2773×10^{-4}
	(0.1000, 0.2000, 0.7000)	1.0000×10–9		(0.1000, 0.2000, 0.7000)	0.0000 × 10–0
(0.20, 0.20, 0.60)	(0.1095, 9.3104×10^{-3}, 0.8812)	-1.4729×10^{-2}	15,6	(0.1095, 9.3104×10^{-3}, 0.8812)	-1.4729×10^{-2}
	(0.1878, 0.1126, 0.6996)	1.9549×10^{-3}		(0.1878, 0.1126, 0.6996)	1.9549×10^{-3}
	(0.2000, 0.2000, 0.6000)	1.0000×10^{-8}		(0.2000, 0.2000, 0.6000)	0.0000 × 10–0

[#]For this system the computational time is not presented in the literature.

Four-dimensional problem

A four dimensional problem was solved by SubDivNL algorithm. These results obtained here and those from the literature are presented in Table 7.

Table 7. Results for the four-dimensional problem.

Roots Number	SubDivNL algorithm	Literature (Shacham *et al.*, 2002)	
	Roots Values	Initial Guess	Roots Values
P13 Problem: variables (C_A, C_B, C_C, T)			
Root 1	2.3804, 0.619577, 0.164268×10^{-05}, 333.4925	(3.0, 0.0, 0.0, 350.0)	2.3804, 0.619577, 0.164268×10^{-05}, 333.4925
Root 2	2.7873, 0.212680, 0.646861×10^{-07}, 310.2126	(3.0, 0.0, 0.0, 300.0)	2.787320, 0.212680, 0.646861×10^{-08}, 310.2126
Root 3	0.1263970, 2.849909, 0.236946×10^{-01}, 462.56916	(3.0, 0.0, 0.0, 400.0)	0.126397, 2.849909, 0.236946×10^{-01}, 462.5692
Root 4	0.380106×10^{-02}, 1.7136, 1.2826, 594.0274	(0.0, 2.0, 1.0, 600.0)	0.380106×10^{-02}, 1.7136, 1.2826, 594.0274
Root 5	0.379754×10^{-03}, 0.688892, 2.3107, 691.6242	(3.0, 0.0, 0.0, 700.0)	0.379757×10^{-03}, 0.688892, 2.3107, 691.6242

P13 Problem: steady state operation of an adiabatic CSTR reactor

The problem deals with the determination of multiple steady state conditions of an adiabatic CSTR reactor. The intervals for the independent variables are [0.0, 3.0], [0.0, 3.0], [0.0, 3.0] and [200.0, 700.0] for the concentration of components A (C_A), B (C_B), and C (C_C), and reactor temperature (T), respectively. The parameters adopted for the SubDivNL algorithm were two coverage ($iCov$ = 2) and three subdivisions (i = 3). The SubDivNL algorithm found five solutions from the intervals defined, spending 13.47s for the determination of all roots. This is a relative difficult problem for obtaining the roots due to the presence of multiplicity. The literature also reports five solutions for this problem, but requesting five different initial guesses to accomplish it.

Conclusion

From the results obtained in this work, one can see that the subdivision algorithm, as it was implemented, showed to be robust and efficient in the resolution of typical problems of chemical engineering described by nonlinear algebraic systems. In the investigated problems, all feasible solutions could be found since false or unfeasible solutions were naturally excluded through the specification of the variables intervals. Another important characteristic of the algorithm is that it seems to be capable of finding all of the solutions for problems that present strong discontinuities, singular points and regions where the functions of the system are undefined. The SubDivNL algorithm presents characteristics that allow it to be classified as a global method, since in all problems of this study it did not present any convergence problems. It may be convenient to emphasize its easy implementation, without the necessity of specific operators.

Additionally, the subdivision algorithm presented an excellent performance with regard to CPU time for multidimensional problems. The CPU time was observed to increase with increasing dimension of

the problem, number of subdivisions and coverage. The employed criterion of exclusion of sub-rectangles showed to be efficient and viable in the resolution of such problems. Investigation on how the CPU time is affected by the criterion of rectangles exclusion is not within the scope of this work and is underway within our working group.

Considering that the present tool may be of interest to researchers and to those committed to chemical engineering education, a basic version of the SubDivNL algorithm can be freely obtained upon contacting the authors.

Acknowledgements

The authors thank CAPES, CNPq and FAPERGS for the financial support for this work.

References

BAKER, L.E. *et al.* Gibbs energy analysis of phase equilibria. *Soc. Petrol. Engrs. J.*, Richardson, v. 22, p. 731-742, 1982.

CORAZZA, M.L. *et al.* Robust strategy for SLV equilibrium calculations at high pressures. *Fluid Phase Equilibr.*, Amsterdam, v. 221, p. 113-126, 2004.

DAVIDENKO, D. On a new method of numerically integrating a system of nonlinear equations. *Dokl. Akad. Nauk,* Moscow, v. 88, p. 601-602, 1953.

DELLNITZ, M.; HOHMANN, A. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math*., Amsterdam, v. 75, p. 293-317, 1997

GANG XU, B.S. *Reliable phase stability and equilibrium calculation using interval analysis for equations of state models.* 2001. Thesis (Ph.D in Chemical Engineering)-University of Notre Dame, Notre Dame, 2001.

GECEGORMEZ, H.; DEMIREL, Y. Phase stability analysis using interval newton method with NRTL model. *Fluid Phase Equilibr.*, Amsterdam, v. 237, p. 48-58, 2005.

GREGORY, J.A. An introduction to bivariate uniform subdivision: numerical analysis. *In*: GRIFFITHS, D.F.; WATSON, G.A. (Ed.). *Pitman research notes in mathematics*. New York: Longman, 1991. p. 103-117.

GRITTON, K.S. *et al.* Global homotopy continuation procedures for seeking all roots of a nonlinear equation. *Comput. Chem. Eng.*, Amsterdam, v. 25, p. 1003-1019, 2001.

HUA, J.Z. *et al.* Reliable prediction of phase stability using an interval-Newton method. *Fluid Phase Equilibr.*, Amsterdam, v. 116, p. 52-59, 1996.

HUA, J.Z. *et al.* Reliable computation of phase stability using interval analysis: cubic equation of state models. *Comput. Chem. Eng.*, Amsterdam, v. 22, p. 1207-1214, 1998.

KELLER, H.B. Global homotopies and Newton methods. *In*: BOOR, C. *et al. Recent advances in numerical analysis*. New York: Academic Press, 1978.

KERFOTT, R.B. *et al*. Algorithm 737: Intlib, a portable Fortran 77 interval standard function library. *ACM Trans. Math. Softw.*, New York, v. 20, p. 447-459, 1994.

KERFOTT, R.B.; NOVOA, M. Algorithm 681: INTBIS, a portable interval Newton/bisection package. *ACM Trans. Math. Softw.*, New York, v. 16, p. 152-157, 1990.

KOLEV, L.V. A new method for global solution of systems of non-linear equations. *Reliable Comput.*, Berlin, v. 45, p. 125-146, 1998.

KREYSZIG, E. *Advanced engineering mathematics*. 8th ed. New York: John Wiley, 1999.

KUNO, M.; SEADER, J.D. Computing all real solutions to systems of nonlinear equations with fixed-point homotopy. *Ind. Eng. Chem. Res.*, Washington, D.C., v. 27, p. 1320-1329, 1998.

LETCHER, T.M.; NAICKER, P.K. Liquid-liquid equilibria for mixtures of water + an Alkanol + a Nitrile compound at 298.15 K. *J. Chem. Eng. Data*, Washington, D.C., v. 46, p. 1436-1441, 2001.

LINTOMEN, L. *et al*. Liquid-liquid equilibrium of the water + Citric Acid + short chain alcohol + tricaprylin system at 298.15 K. *J. Chem. Eng. Data*, Washington, D.C., v. 46, p. 546–550, 2001.

PAN, V.Y. Solving a polynomial equation: some history and recent progress. *SIAM Rev.*, Philadelphia, v. 39, p. 187-220, 1997.

PERNICE, M.; WALKER, H.F. Nitsol: a Newton iterative solver for nonlinear systems. *SIAM J. Sci. Comput.*, Philadelphia, v. 19, p. 302-318, 1998.

PRESS, W.H. *et al*. *Numerical recipes in Fortran: the art of scientific computing.* 2nd ed. Cambridge: Cambridge University Press, 1992.

RANGAIAH¸ G.P. Evaluation of genetic algorithms and simulated annealing for phase equilibrium and stability problems. *Fluid Phase Equilibr.*, Amsterdam, v. 187-188, p. 83-109, 2001.

SEADER, J.D. *et al*. Mapped continuation methods for computing all solutions to general systems of nonlinear equations. *Comput. Chem. Eng.*, Amsterdam, v. 14, p. 71-85, 1990.

SEIDER, W.D. *et al.* Nonlinear analysis in process design. *AIChE J.,* New York, v. 37, p. 1-38, 1991.

SHACHAM, M. An improved memory method for the solution of a nonlinear equation. *Chem. Eng. Sci.*, Amsterdam, v. 44, p. 1495-1501, 1989.

SHACHAM, M. A variable order method for solution of a nonlinear algebraic equation. *Comput. Chem. Eng.*, Amsterdam, v. 14, p. 621-629, 1990.

SHACHAM, M. *et al.* A web-based library for testing performance of numerical software for solving nonlinear algebraic equations. *Comput. Chem. Eng.*, Amsterdam, v. 26, p. 547-554, 2002.

SMILEY, M.W.; CHUN, C. An algorithm for finding all solutions of a nonlinear system. *J. Comput. Appl. Math.*,

Amsterdam, v. 137, p. 293-315, 2001.

WAYBURN, T.L.; SEADER, J.D. Homotopy continuation method for computer-aided process design. *Comput. Chem. Eng.,* Amsterdam, v. 11, p. 7-25, 1987.

WEYL, H. Randbermerkungen zu hauptproblemen der mathematik, ii, fundamentalsatz der algebra and grundlagen der matehmatik. *Math. Z.,* Heidelberg, v. 20, p. 131-151, 1921.

Received on August 26, 2006.
Accepted on August 28, 2007.