INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS
Int. J. Circ. Theor. Appl., **26**, 81–92 (1998)

AN EFFICIENT INTERVAL METHOD FOR GLOBAL ANALYSIS OF NON-LINEAR RESISTIVE CIRCUITS

L. V. KOLEV*
Department of Automatica, Technical University of Sofia, 1756 Sofia, Bulgaria

SUMMARY

A new method for finding the set of all operating points of non-linear resistive circuits is suggested. It takes into account the fact that the circuit equations are in the so-called separable form where any equation $f_i(x)$ is the sum of terms $f_{ij}(x_j)$, each term depending on a single variable. The method is based on the introduction of an approximation of every real non-linear function $f_{ij}(x_j)$ by an appropriate linear interval function, i.e. by a real linear function having an additive interval term. The parameters of the linear interval approximations are dynamically updated at each iteration of the computational process.

Numerical experiments show that the computational efficiency of the new method is vastly superior to that of other known methods for global analysis, especially for circuits of large size. © 1998 John Wiley & Sons, Ltd.

KEY WORDS: non-linear resistive circuits; global analysis; interval method

1. INTRODUCTION

It is well known that the problem of global analysis of non-linear resistive circuits (locating all dc operating points of the circuit investigated) consists of finding all the real solutions of a corresponding system of non-linear algebraic equations. More specifically, the following problem hereafter referred to as the GA problem has been considered in numerous publications.

The GA problem. Let $f: X^{(0)} \subset \mathbb{R}^n \to \mathbb{R}^n$ be the function describing the dc operation of the circuit studied where $X^{(0)}$ is a (large enough) 'box' in $\mathbb{R}^n$. Find the set $S(f, X^{(0)}) = \{x^{(1)}, \ldots, x^{(p)}\}$ of all real solutions x^s, $s = 1, \ldots, p$ of the system

$$f(x) = 0 \tag{1}$$

which are contained in $X^{(0)}$, i.e. when

$$x \in X^{(0)} = (X_1^{(0)}, \ldots, X_n^{(0)}) \tag{2a}$$

where each component $X_i^{(0)}$ of $X^{(0)}$ is an interval

$$X_i = [\underline{x}_i, \bar{x}_i] \tag{2b}$$

Typically, $f(x)$ is in the well-known hybrid representation form

$$f(x) = \varphi(x) - Hx - s \tag{3a}$$

with

$$\varphi_i(x) = \varphi_i(x_i), \quad i = 1, \ldots, n \tag{3b}$$

*Correspondence to: L. V. Kolev, Department of Automatica, Technical University of Sofia, 1756 Sofia, Bulgaria.

CCC 0098–9886/98/010081–12$17.50

Received 3 January 1997
Revised 26 June 1997

which describes a large class of non-linear dc electronic circuits[1] or models the equilibrium states of circular neural networks.[2] If the circuit investigated contains bipolar and MOS transistors, the non-linear functions in (3b) take on the form[3]

$$\varphi_i(x) = \begin{cases} \varphi_{i1}(x_i) - \varphi_{i2}(x_{i+1}) & \text{if } i \text{ is odd} \\ -\varphi_{i1}(x_{i-1}) - \varphi_{i2}(x_i) & \text{if } i \text{ is even} \end{cases}$$

The functions $\varphi_i(x_i)$ (or $\varphi_{i1}(x_i)$ and $\varphi_{i2}(x_{i+1})$) model physical two-terminal non-linear elements and are, therefore, originally smooth enough functions (at least continuously differentiable). In this case the GA problem will be referred to as the CDF problem. In order to simplify the original CDF problem, the functions $\varphi_i(x_i)$ are often approximated by piecewise-linear functions and then the ensuing GA problem will be referred to as the PWL problem. The solution of the GA problem presents enormous difficulties, especially for larger n.

Presently, there exist several interval analysis methods[4-10] that are capable of infallibly solving the CDF problem within prescribed accuracy. Experimental evidence has however shown that their computational efficiency only permits the analysis of circuits of moderate dimensionality n. Indeed, all interval methods known to date involve recursive splitting of the initial box $X^{(0)}$ into subboxes X^ν and locating the solution available in X^ν or establishing its absence in X^ν. Unfortunately, the number of X^ν and hence the computational effort needed to solve the GA problem tends to grow exponentially with the dimensionality of the problem.

An improved interval method has been recently suggested in Reference 8. It is based on the use of the so-called interval slopes[9] (instead of interval derivatives used in the previous methods). Although this method seems to be the best in the group of interval methods, its applicability for circuits of higher dimension still remains questionable.

In this paper, a new highly efficient interval method for solving the GA problem (both in its CDF or PWL form) will be suggested. It is based on a dynamically updated interval approximation of the non-linear functions which exploits to a fuller extent the fact that the circuit equations (3) and (4) are of separable form. A function $f_i(x)$ on n variables is called separable if it has the following form

$$f_i(x) = \sum_{j=1}^{n} f_{ij}(x_j) \tag{5}$$

Clearly, functions (3) and (4) are separable.

The separability property has already been employed with the view to improving the numerical efficiency of a continuation method for tracing solution curves in Reference 11 and of non-interval methods for non-linear analysis of resistive circuits in References 12 and 13.

The method herein suggested is derived in Section 2. Two algorithms implementing the new method are presented in the next section. The first one is based on vector form operations. The second algorithm is an improvement obtained by the use of componentwise computations. Finally, in Section 4 the numerical efficiency of this method is compared with that of other global analysis methods.

2. BASIC RESULTS

The new method is based on some theoretical results to be presented in this section.

Let X denote any subbox X^ν generated during the computational process by splitting the initial box X^0. First, a new interval approximation of a component $f_{ij}(x_j)$ in X_j will be suggested. No restrictions on the functions $f_{ij}(x_j)$ are imposed except for the requirement that they be continuous. The above approximation is illustrated graphically in Figure 1 for the case of continuously differentiable (CD) functions.

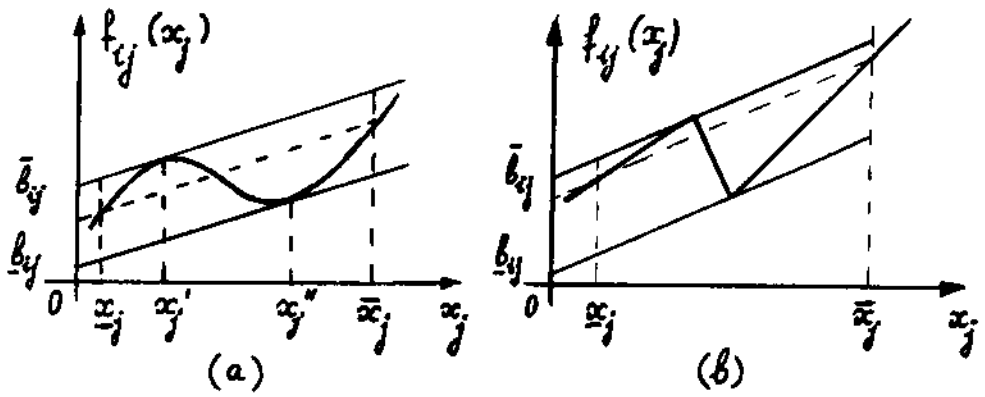

Figure 1. Geometrical illustration of the linear interval approximation of $f_{ij}(x_j)$ in $X_j = [\underline{x}_j, \bar{x}_j]$: (a) for the case of a continuously differentiable function, and (b) for the case of piecewise-linear functions

Unlike the previous methods where the functions $f_{ij}(x_j)$ are approximated using interval derivatives[4–7] or interval slopes[8,9] the new approximation of $f_{ij}(x_j)$ in X_j is chosen in the following form:

$$L_{ij}(x_j) = b_{ij}^{\mathrm{I}} + a_{ij}x_j, \quad x_j \in X_j \tag{6}$$

where $b_{ij}^{\mathrm{I}} = [\underline{b}_{ij}, \bar{b}_{ij}]$ is an interval while a_{ij} is a real number. Both b_{ij}^{I} and a_{ij} are to be determined such that the following inclusion property should hold:

$$f_{ij}(x_j) \in b_{ij}^{\mathrm{I}} + a_{ij}x_j, \quad x_j \in X_j \tag{7}$$

A simple and efficient procedure for finding a_{ij}, $\underline{b}_{ij}$ and $\bar{b}_{ij}$ is suggested here for the case of CD functions. It is motivated by elementary geometrical considerations and can be readily modified for the case of PWL functions.

Procedure 1. First, compute

$$\underline{f}_{ij} = f_{ij}(\underline{x}_{ij}), \qquad \bar{f}_{ij} = f_{ij}(\bar{x}_j) \tag{8}$$

Then a_{ij} is defined as the slope

$$a_{ij} = (\bar{f}_{ij} - \underline{f}_{ij})/(\bar{x}_j - \underline{x}_j) \tag{9}$$

Afterwards, the following equation is solved for x_j:

$$\frac{\mathrm{d}}{\mathrm{d}x_j}(f_{ij}) = f'_{ij}(x_j) = a_{ij} \tag{10}$$

In the general case (10) will have several solutions. They all can be located infallibly using an interval version of the Newton method (the so-called extended interval Newton algorithm, Reference 5, Section 1.4.1). Among them, two solutions denoted x' and x'' are to be chosen such that the following conditions should be satisfied. Let

$$l_1(x_j) = \bar{b}_{ij} + a_{ij}x_j \tag{11}$$

be a straight line passing through the point $(x'_j, f_{ij}(x'_j))$ and having slope a_{ij} such that

$$f_{ij}(x_j) \leqslant l_1(x_j), \quad x_j \in X_j \tag{12}$$

Similarly, let the straight line

$$l_2(x_j) = \underline{b}_{ij} + a_{ij}x_j \tag{13}$$

passing through the point $(x_j'', f_{ij}(x_j''))$ have the property

$$f_{ij}(x_j) \geqslant l_2(x_j), \quad x_j \in X_j \tag{14}$$

Now it is easily seen that

$$\underline{b}_{ij} = f_{ij}(x_j'') - a_{ij}x_j'', \qquad \bar{b}_{ij} = f_{ij}(x_j') - a_{ij}x_j' \tag{15a, b}$$

In the special case where $f_{ij}(x_j)$ is either convex or concave in X_j, equation (10) has a unique solution $\tilde{x}$. In this case formulae (15) are to be modified as follows. If $f_{ij}(x_j)$ is convex then let $x_j' = \underline{x}_j$ and $x_j'' = \tilde{x}_j$; similarly, in the case of a concave function $f_{ij}(x_j)$ let $x_j' = \tilde{x}_j$ and $x_j'' = \underline{x}_j$.

The above procedure remains, essentially, the same in the case of PWL functions $f_{ij}(x_j)$. The only difference is that the points x_j' and x_j'' are now found directly on the PWL function by elementary geometrical consideration.

Using the above procedure a real matrix

$$A = \{a_{ij}\}$$

and an interval vector $b^{\mathrm{I}} = (b_1^{\mathrm{I}}, \ldots, b_n^{\mathrm{I}})$ are formed with

$$b_i^{\mathrm{I}} = \sum_{j=1}^{n} b_{ij}^{\mathrm{I}} \tag{16}$$

On account of (5), (7), (12), (14) and (16)

$$f_i(x) \in \sum_{j=1}^{n} a_{ij}x_j + b_i^{\mathrm{I}}, \quad x \in X, \quad i = 1, \ldots, n \tag{17}$$

or in vector form

$$f(x) \in Ax + b^{\mathrm{I}}, \quad x \in X \tag{18}$$

If y is a solution of (1) in X, then $f(y) = 0$ and by (18)

$$0 \in Ay + b^{\mathrm{I}}, \quad y \in X \tag{19}$$

Now, we can state the main result of the section.

Theorem 1. All the solutions y to

$$f(x) = 0 \tag{20}$$

contained in X are also contained in the solution set $S(X)$ of the system

$$Ax + b = 0, \quad b \in b^{\mathrm{I}} \tag{21}$$

where b is any real vector contained in b^{I}.

The proof of the theorem is given in the appendix.

Since b^{I} is an interval vector the set $S(X)$ is a convex polyhedron. Indeed, (21) is in fact a system of n linear equalities and $2n$ two-sided linear inequalities.

We are ready to state another basic result.

Theorem 2. All the solutions y to (20) *in X are also contained in the intersection*

$$P(X) = S(X) \cap X \tag{22}$$

(See the appendix for the proof of the theorem.)

Corollary 1. If P(X) is empty, i.e. if

$$S(X) \cap X = \emptyset \tag{23}$$

the system (20) *has no solution in X.*

Let $H(P, X)$ denote the interval hull of $P(X)$, that is the smallest interval vector (box) containing $P(X)$. Consider the following iteration procedure:

$$X^{(k+1)} = H(P(X^{(k)})) \cap X^{(k)}, \quad k \geqslant 0 \tag{24}$$

Similarly to other methods[4–8], procedure (24) may be used for designing a method for finding all real solutions to (1) in $X^{(0)}$. Such an approach would, however, be rather costly since $2n$ linear programming problems are to be solved at each iteration to determine $H(P(X^{(k)}))$. Therefore, a simpler and, presumably, more efficient procedure is suggested here.

Let $H(S, X)$ denote the interval hull of $S(X)$. Furthermore, let $B = -b^{I}$ with components

$$B_i = [\underline{B}_i, \bar{B}_i] = [-\bar{b}_i, \ -\underline{b}_i] \tag{25}$$

If the matrix A is not singular, $H(S, X)$ is given by the formula

$$H(S, X) = A^{-1}B \tag{26}$$

Indeed, $H(S, X)$ is by definition the interval solution of the following linear interval equation:

$$Ax = b, \quad b \in B \tag{27}$$

Since A is a real non-singular matrix formula (26) is valid. (In the case where A happens to be singular, the current box X is split along its widest side into two boxes and Procedure 1 is applied to each box separately.)

Let $C = A^{-1}$ and $Y = H(S,X)$. It follows from (26) that the components $Y_i = [\underline{y}_i, \bar{y}_i]$ of y are given by the formulae

$$\underline{y}_i = \sum_{j=1}^{n} \underline{y}_{ij} \tag{28}$$

with

$$\underline{y}_{ij} = \begin{cases} C_{ij}\underline{B}_i & \text{if } C_{ij} \geqslant 0 \\ C_{ij}B_i & \text{if } C_{ij} < 0 \end{cases} \tag{29}$$

$$\bar{y}_i = \sum_{j=1}^{n} \bar{y}_{ij} \tag{30}$$

with

$$\bar{y}_{ij} = \begin{cases} C_{ij}\bar{B}_i & \text{if } C_{ij} \geqslant 0 \\ C_{ij}\underline{B}_i & \text{if } C_{ij} < 0 \end{cases} \tag{31}$$

Now, the following interative procedure based on $H(S, X)$ is suggested here.

Procedure 2. Let $X^{(k)}$ be the current box generated by the present method. Using Procedure 1 determine $C^{(k)}$ and $B^{(k)}$ corresponding to $X^{(k)}$. By formulae (28)–(31) compute $Y^{(k)}$. The iterative procedure is then

defined as follows:

$$X^{(k+1)} = Y^{(k)} \cap X^{(k)}, \quad k \geqslant 0 \tag{32}$$

The procedure may result in three outcomes.

A. The sequence $X^{(k+1)}$ converges to a fixed interval (box) X^*. In practice, the procedure is stopped whenever the reduction in the volume of the current box $X^{(k+1)}$ as compared to that of the preceding box $X^{(k)}$ is smaller then a constant ε_1. In this case $X^{(k+1)}$ is split along its widest side into two boxes X^L and X^R (left and right). The right box is stored into a list L for further processing. The left box is remained $X^{(0)}$ and the iterative procedure (28)–(32) is resumed.

B. At some K

$$Y^{(k)} \cap X^{(k)} = \emptyset \tag{33}$$

Since $Y = H(S, X) \subseteq S(X)$ is follows from Corollary 1 that system (20) has no solution in $X^{(k)}$ if (33) becomes valid. In this case $X^{(k)}$ is discarded from further consideration.

C. The sequence $X^{(k+1)}$ converges to a solution $x^{(s)}$ as k increases. Actually, the iterations are stopped whenever the width of $X^{(k+1)}$ becomes smaller than a constant ε_2 (accuracy with respect to x). Now x^s is approximated by the centre x^c of $X^{(k+1)}$ and x^c is substituted in (1). If

$$\text{tol} = \max[|f_i(x^c)|, \quad i = 1, \ldots, n] > \varepsilon_3 \tag{34}$$

(ε_3 is the accuracy of x^c with respect to the system of equations) then the iterations are resumed; otherwise, x^c is accepted as a solution to (1).

If outcomes B and C occur then a new box (the first box stored in L) is retrieved from L, renamed $X^{(0)}$ and Procedure 2 is again applied. It can be shown (similarly to References 4–8) that this process of generating, storing the retrieving subboxes will terminate in a finite number of iterations locating all the solutions to (1) within prescribed accuracy or establishing the absence of solutions to (1) in $X^{(0)}$.

Since the present method is capable of solving both the CDF and PWL versions of the GA problem, henceforth only the former version will be considered.

3. NUMERICAL IMPLEMENTATION

Based on the theoretical results from the previous section a basic algorithm of the new method has been elaborated.

Algorithm 1

Step 0. (Initialization). Let $X = X^{(0)}$, $v = 0$ (v is the number of iterations), $n_s = 0$ (number of solutions in $X^{(0)}$), $l = 0$ (number of boxes stored in L). Assign values of l_L (length of the list L), ε_1, ε_2 and ε_3.

Step 1. Call Procedure 1 to compute the real matrix A and the interval vector B corresponding to X.

Step 2. Compute $C = A^{-1}$.

Step 3. Call Procedure 2.

Step 4a. If Procedure 2 terminates in outcome B, proceed to step 5.

Step 4b. If it stops in outcome C print the result in real and interval form, put $n_s = n_s + 1$ and go to step 5.

Step 4c. If outcome A occurs, then split the current box into X_L and X_R. Store X_R into L and put $l = l + 1$. Rename X_L as X and go back to step 1.

Step 5. If $l = 0$, terminate and print n_s. If $n_s = 0$ then (1) has no solution in $X^{(0)}$, otherwise n_s indicates the number of solutions located in Step 4b.

If $l > 0$ then proceed to next step.

Step 6. Retrieve the first element of L (the first box stored in L), rename it X and put $l = l - 1$. Resume the computation process from step 1.

Algorithm 1 is based on vector operations. Its convergence can be improved if component operations are introduced. Thus whenever a reduction of a component occurs, this will be used immediately for reducing (if possible) the remaining components $X_j^{(k+1)}$, $j = i + 1, \ldots, n$. Additionally, a more effective exclusion rule than (33) will be employed.

To introduce this componentwise algorithm we need to modify Procedures 1 and 2 to Procedures 3 and 4, respectively.

Procedure 3. In this procedure, all the computations in Procedure 1 are carried out for a fixed j (that is for a fixed column) and a given X_j. Thus, on exit from Procedure 3 we have the jth column A_j of A and an interval vector b_j^1 whose components are b_{ij}^1, $i = 1, \ldots, n$. For reasons to become clear later, it is expedient to introduce two real matrices B^l and B^u and to store $\underline{b}_{ij}$ and $\bar{b}_{ij}$ in the jth row of B^l (lower end-points) and B^u (upper end-points), respectively.

To introduce Procedure 4, formula (32) is written in componentwise form

$$X_i^{(k+1)} = Y_i^{(k)} \cap X_i^{(k)} \tag{35}$$

Now, according to the idea to get the most out of the reduction of $X_i^{(k+1)}$ we check whether

$$X_i^{(k+1)} \subset X_i^{(k)} \tag{36}$$

If (36) holds (in practice if the reduction is greater than some threshold ε_4) Procedure 3 is immediately called to recomputed the corresponding ith column A_i of A, and the ith columns B_i^l and B_i^u of B^l and B^u, respectively. Let these updated vectors are denoted $\tilde{A}_i$, $\tilde{B}_i^l$ and $\tilde{B}_i^u$. Similarly let $\tilde{A}^{(k)}$, $\tilde{C}^{(k)}$ and $\tilde{B}^{(k)}$ designate the corresponding updated matrices $A^{(k)}$, $C^{(k)}$ and interval vector $B^{(k)}$. Now, the lower and upper end-points of $\tilde{B}^{(k)}$ are easily evaluated:

$$\underline{\tilde{B}}^{(k)} = \underline{B}^{(k)} - B_i^l + \tilde{B}_i^l \tag{37a}$$

$$\bar{\tilde{B}}^{(k)} = \bar{B}^{(k)} - B_i^u + \tilde{B}_i^u \tag{37b}$$

It is seen from (28)–(31) that to compute the next component $Y_{i+1}^{(k)}$ we need the updated row $\tilde{C}_{i+1}^{(k)}$ corresponding to the updated matrix $\tilde{A}^{(k)}$. The latter one is equal to matrix $A^{(k)}$ except for its ith column $A_i^{(k)}$ that is replaced by the updated column $\tilde{A}_i$. Thus, $C_{i+1}^{(k)}$ can be found by updating the matrix $C^{(k)} = (A^{(k)})^{-1}$ when the ith column $A_i^{(k)}$ of $A^{(k)}$ is modified by the increment vector

$$\Delta = \tilde{A}_i - A_i^{(k)} \tag{38}$$

Then the updated matrix $\tilde{C}^{(k)}$ can be found inexpensively (requiring roughly n^2 multiplications) by

$$\tilde{C}^{(k)} = C^{(k)} - \frac{1}{\alpha} C^{(k)} D_i C^{(k)} \tag{39}$$

with

$$\alpha = 1 + (C^{(k)} D_i)_{ii}$$

where D_i is a matrix whose all columns are zero except for the ith column which is Δ.

Having determined $\tilde{C}_{i+1}^{(k)}$, finally $Y_{i+1}^{(k)}$ is evaluated by formulae (28)–(31) where $\tilde{C}_{i+1}^{(k)}$ and $\tilde{B}^{(k)}$ are used.

Now, we are in a position to present Procedure 4.

Procedure 4. For a given X_i corresponding to a fixed index i call Procedure 3 to yield the new vector $\tilde{A}_i$, $\tilde{B}_i^l$ and $\tilde{B}_i^u$. Compute the interval vector $\tilde{B}^{(k)}$ by (37). Determine the real vector $\tilde{C}_{i+1}^{(k)}$ as the $(i + 1)$th row of $\tilde{C}^{(k)}$ computed by (39). Finally, evaluate $Y_{i+1}^{(k)}$ by (28)–(31).

The componentwise algorithm incorporates an improved exclusion rule (as compared to (33)) which is based on (23). From the definition of $S(X)$ it is easily seen that condition (23) is fulfilled for the ith co-ordinate if X does not intersect the 'slice' defined by

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant \bar{B}_i, \qquad \underline{B}_i \leqslant \sum_{j=1}^{n} a_{ij}x_j \tag{40a, b}$$

Now, define

$$\underline{z}_i = \sum_{j=1}^{n} \underline{z}_{ij} \tag{41a}$$

where

$$\underline{z}_{ij} = \begin{cases} a_{ij}\underline{x}_j & \text{if } a_{ij} \geqslant 0 \\ a_{ij}\bar{x}_j & \text{if } a_{ij} < 0 \end{cases} \tag{41b}$$

and

$$\bar{z}_i = \sum_{j=1}^{n} \bar{z}_{ij} \tag{42a}$$

$$\bar{z}_{ij} = \begin{cases} a_{ij}\bar{x}_j & \text{if } a_{ij} \geqslant 0 \\ a_{ij}\underline{x}_j & \text{if } a_{ij} < 0 \end{cases} \tag{42b}$$

Hence, the new exclusion rule states that (23) is valid if

$$\underline{z}_i > \bar{B}_i \quad \text{or} \quad \bar{z}_i < \underline{B}_i \tag{43a, b}$$

The componentwise algorithm is thus based on the following procedure.

Procedure 5

Step 0. Let $i = 0$.
Step 1. Put $i = i + 1$. If $i > n$ go to Step 4, else proceed to next step.
Step 2. If the exclusion rule (43) holds, then go to Step 5, otherwise go to next step.
Step 3. Cell Procedure 4 to yield Y_i. If

$$Y_i \in X_i$$

then call Procedure 3 and go to Step 1, otherwise go to Step 1.
Step 4 If the condition for splitting is fulfilled, terminate with outcome A, otherwise go to next step.
Step 5. The Procedure is terminated with outcome B.
Step 6. If (34) holds, go to Step 1, otherwise terminate with outcome C.

Now we are in a position to present the componentwise algorithm.

Algorithm 2

Steps 0–2. Same as in Algorithm 1.
Step 3. Call procedure 5.
Step 4a. If Procedure 5 terminates in outcome B, proceed to step 5.
Steps 4b–6 are the same as in Algorithm 1.

The above two algorithms become impractical for large-size circuits because of the inversion of $A^{(k)}$. Algorithm 2 can, however, be readily adapted to encompass the case of large n. Indeed, by formulae (28)–(31),

after the vector B has been evaluated, we only need the ith row C_i of $C = A^{-1}$ to determine Y_i. Thus, C_i can be found by solving the linear system

$$(A)^{\mathrm{T}} C_i = e_i \tag{44}$$

where the symbol T means transpose and e_i is the ith column of the identity $(n \times n)$ matrix E. Indeed

$$(CA)^{\mathrm{T}} = (A)^{\mathrm{T}} . (C)^{\mathrm{T}} = E \tag{45}$$

and (44) follows from (45).

For large-scale circuits, A is a rather sparse matrix and sparse matrix methods should be employed for solving (44) in this case. Additional improvement is possible if one takes into account that according to (38) A is perturbed at each iteration j by only a column vector Δ_j. This perturbation can be written equivalently in the form of dyad, i.e.

$$\tilde{A} = A + \Delta_j \gamma^{\mathrm{T}} \tag{46}$$

where γ is a column vector whose elements are zero except for its jth element which is equal to one. Then a very efficient sparse matrix method[14] which exploits the above dyad form perturbation can be made use of.

Remark 1. To avoid cluttering up the presentation of the present method with technical details, nothing has been as yet said about the so-called clustering phenomenon characteristic of any interval method for solving systems of non-linear equations. If clustering occurs this means that on exit the method generates around each small box X^S containing a solution x^s a certain number of boxed $X^{S_1}, X^{S_2}, \ldots, X^{S_k}$ of the same width of X^S. However, unlike the box X^S, the remaining boxes X^{S_i} do not contain the solution x^s. Clustering appears, essentially, because the exclusion rule used is not capable of deleting small enough boxes around a solution. The clustering effect grows stronger as the problem dimension n increases. It has been observed with all the known interval methods for global dc analysis. For one and the same problem, the clustering effect decreases if the method used has better convergence towards a solution. As will be shown in the next section, the clustering effect of the present method, if clustering ever appears, is much less pronounced in comparison with other known methods.

In the previous methods[7,8,10] clustering was detected 'manually' by inspection of the solutions. In the present method, simple procedure for detecting and deleting clustering has been incorporated in Algorithm 1 whenever Procedure 2 terminates in outcome C.

Remark 2. Finally, a remark regarding the interval arithmetic implementation of the present interval method is to be made. Any interval method is to be implemented using appropriate interval arithmetic to ensure strict enclosures of the numerical results obtained. The present method for solving (1) requires outward rounding of all elementary arithmetic operations involved to guarantee that the intervals X^S obtained in Procedure 2, outcome C will cover the exact solutions x^s.

Outward rounding also ensures that no solution is ever lost in the process of reducing the size of the current box X or when discarding altogether the whole box X. Thus, the use of interval arithmetic makes the present method self-validating, guaranteering that all the operating points of the circuit investigated will be infallibly located within the desired accuracy. It should, however, be borne in mind that interval arithmetic operations are 5–6 times more expensive than the traditional floating-point arithmetic and require dedicated software packages.

It should also be mentioned that the numerical validation of this paper's method for global non-linear analysis can be implemented alternatively by a stochastic method[15] without appealing to interval computations at all. However, unlike the interval implementation which yield interval results covering the exact solutions with mathematical certainty the latter method only provides numerical results which have a certain number of true significant bits within prescribed probability.

4. ILLUSTRATIVE EXAMPLES

The numerical performance of the present method has been tested on several systems of equations of the form (3) with n ranging from $n = 3$ to $n = 10$, using Algorithm 1. At this stage, all calculations are implemented using traditional floating-point arithmetic operations. In all the cases, the CDF problem is solved. To illustrate the improved numerical efficiency of the new method two examples will be given.

Example 1

The electric circuit studied contains a transistor and a diode (Example 6.2 in Reference 5). The description (3) is in this case given by three equations with

$$\varphi_1(x_1) = 10^{-9}(e^{38x_1} - 1)$$
$$\varphi_2(x_2) = 1{\cdot}98 \times 10^{-9}(e^{38x_2} - 1)$$
$$\varphi_3(x_3) = 10^{-9}(e^{38x_3} - 1)$$

$$H = \begin{bmatrix} -0{\cdot}6689 & 1{\cdot}6722 & -0{\cdot}6689 \\ -0{\cdot}6622 & -1{\cdot}3445 & -0{\cdot}6622 \\ -1 & -1 & -4 \end{bmatrix}$$

$$S = (8{\cdot}0276, \ -4{\cdot}0535, \ 6)^{\mathrm{T}}$$

The problem is to find all the operating points of the circuit (the set $S(f, X^{(0)})$); within the interval region

$$X^{(0)} = ([0, 1], [-5, 0], [0, 1])$$

with $\varepsilon_2 = 0{\cdot}01$. The number of iterations needed by the present method to locate the only solution within $X^{(0)}$ with the desired accuracy is $N_i = 7$.

To compare the efficiency of the new method (denoted as M6) with that of other interval methods (methods M2–M5 from R.5), data about N_i are given in Table I.

Table I

Method	M2	M3	M4	M5	M6
N_i	92	18	43	18	7

It should also be mentioned that no clustering effect has been observed with this problem.

The superiority of the present method over the other known interval methods is much more pronounced for problems of higher dimension, as illustrated in the following example.

Example 2

In this example, a circuit containing 10 tunnel diodes and studied in Reference 8 and 13 has a description (3) of the form

$$\varphi_i(x_i) = 2{\cdot}5x_i^3 - 10{\cdot}5x_i^2 - 10{\cdot}5x_i^2 + 11{\cdot}8x_i, \quad i = 1, \ldots, 10$$
$$H = \{h_{ij}\} \text{ with } h_{ij} = -1$$
$$S = (-1, \ -2, \ -3, \ \ldots, \ -10)$$

Table II

Method	M1	M2	M3
N_i	524143	116522	146

The interval $X^{(0)}$ is defined by

$$x_i \in [-1, 4], \quad i = 1, \ldots, n.$$

The accuracy ε_2 has been chosen to be 10^{-4}.

Two interval methods were applied in Reference 8 to solve the CDF problem considered. The first method denoted here as M1 is based on the use of interval derivatives slopes. The second method designated as M2 is an improved version which uses interval slopes. This paper's method denoted as M3 has also solved the problem considered and has found within the same accuracy $\varepsilon_2 = 10^{-4}$ all the nine solutions contained in $X^{(0)}$. However, the data in Table II concerning the number of iterations required to solve the GA problem for the circuit studied reveal that the present method is vastly superiorer to the best interval methods known as regards computer time.

On account of its fast convergence rate the new method has also improved characteristics as regards memory volume requirements. Indeed, the maximum number of boxes n_m stored during computation reached the value of 6 for method M3 while n_m was manifold higher for M1 and M2. Also, the clustering effect is now by far less pronounced. Thus, Algorithm 1 (without the procedure for deleting the clustering mentioned in Remark 1) produced only a total of 13 solutions and it was very easy to locate the 4 duplicating boxes. In contrast, among the two previous methods, even the better method M2 generated decades of clustering boxes, thus requiring much bigger memory volume.

5. CONCLUSIONS

In this paper the problem of finding the set of all operating points of non-linear resistive circuits whose equations are written in the separable form (5) is considered. The only assumption about the characteristics of the non-linear elements involved is that they are modelled by continuous functions $f_{ij}(x_j)$. Thus, both the CDF and PWL problems are encompassed by this paper's problem statement.

A new interval method for solving the problem considered has been suggested which exploits to a much higher extent than other known methods the separability property of the circuit equations. It is based on a suitable approximation of the functions $f_{ij}(x_j)$ by linear interval functions L_{ij} defined by (6). Thus, at each iteration of the method a single linear interval system (27) with a real matrix A is solved. This distinguishes advantageously the present method from the other known interval methods where a much more complex linear interval system having an interval matrix and a real right-hand side vector is solved.

Two algorithms for the numerical implementation of the method have been proposed. While Algorithm 1 is based on vector operations, Algorithm 2 is implemented using componentwise operations thus ensuring a better rate of convergence.

The experimental data reveal that as regards computer time and memory volume requirements the present method exceeds considerably the other known methods for solving the global analysis problem considered. Its superiority becomes much more pronounced as the dimension n of the circuit equations system increases.

The present method can be extended to systems of algebraic equations of arbitrary form. This can be done by transforming the original system into an equivalent separable form system using a method recently suggested in Reference 16. Since the system thus obtained is, generally, much larger than the original system such an approach should appeal to the sparse matrix version of Algorithm 2 herein suggested to ensure its numerical efficiency. Work is presently in progress to implement this scheme.

ACKNOWLEDGEMENTS

This work was supported by the Bulgarian Research Foundation under Contract No. TN-554/96. The author is greatful to one of the reviewers for his valuable comments.

APPENDIX

Proof of Theorem 1

If $y \in X$ is a solution to (20) in X, then $f(y) = 0$ and by the definition of A and b^I the inclusion (19) holds on account of (18). Therefore, the following equality:

$$Ax + b^I = 0 \tag{47}$$

defines a set of points $S(X)$ which contains every solution y to (23) available in X. The interval equations (47) is a short form of representing the system of equations:

$$Ax + b = 0, \quad b \in b^I \tag{48}$$

Hence all solutions y to (20) are in the solution set of (48) which completes the proof.

Proof of Theorem 2

The proof follows directly from obvious set-theoretical considerations. Indeed, as shown by Theorem 1 the solutions y to (20) are in $S(X)$. But at the same time they are in X. Hence all solutions y are also in the intersection $S(X) \cap X$.

REFERENCES

1. L. O. Chua and P. M. Lin, *Computer Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
2. L. O. Chua and L. Yang, 'Cellular neural networks: theory and applications', *IEEE Trans. Circuits and Systems*, **CAS-35**, 1257–1290 (1988).
3. M. Tadeusiewicz and K. Glovienka, 'An algorithm for finding all operating points of electronic circuits containing intricate models of transistors', *Proc. ECCTD'95 Eur. Conf. Circuit Theory & Design*, 1995, pp. 127–130
4. L. V. Kolev, 'Finding all solutions of non-linear resistive circuit equations via interval analysis', *Int. J. Circ. Theor. Appl.*, **12**, 175–178 (1984).
5. L. V. Kolev, *Interval Methods for Circuit Analysis*, World Scientific, Singapore, 1993.
6. L. V. Kolev and V. M. Mladenov, 'An interval method for finding all operating points of non-linear resistive circuits', *Int. J. Circ. Theor. Appl.*, **13**, 257–267 (1990).
7. L. V. Kolev and V. M. Mladenov, 'An interval method for global non-linear dc circuits analysis', *Int. J. Circ. Theor. Appl.*, **22**, 233–241 (1994)
8. L. V. Kolev and V. M. Mladenov, 'Use of interval slopes in implementing as interval method for global non-linear DC circuits analysis', *Int. J. Circ. Theor. Appl.* **25**, 37–42 (1997).
9. L. V. Kolev, 'Use of interval slopes for the irrational parts of factorable functions', *Reliable Computing* **3**, 83–93 (1996).
10. V. M. Mladenov and L. V. Kolev, 'Interval methods for solving cellular neural networks (CNNs) equations', *The 3rd. Int. Conf. on Electronics, Circuits and Systems (ICECS'96)*, Rodos, Greece, 1996.
11. L. V. Kolev, 'Separability property of the non-linear equations for resistive circuits', *Electricity*, **2**, 70–73 (1986) (in Russian).
12. K. Yamamura and K. Horiuchi, 'A globally and quadratically convergent algorithm for solving non-linear resistive networks', *IEEE Trans. Computer-Aided Design*, **9**, 487–499 (1990).
13. K. Yamamura, 'Finding all solutions of piecewise-linear resistive circuits using simple sign tests', *IEEE Trans. Circuits Systems I*, **40**, 546–551 (1993).
14. T. Fujisawa, E. S. Kuh and T. Ohtsuki, 'A sparse matrix method for analysis of piecewise-linear resistive networks', *IEEE Trans. Circuit Theory*, **CT-19**, 571–584 (1972).
15. M. Pichat and J. Vignes, *Ingenierie du Controle de la Precision des Calculs sur Ordinateur*, Technip, Paris, 1993.
16. K. Yamamura, 'An algorithm for representing functions of many variables by superpositions of functions of one variable and addition', *IEEE Trans. Circuits System I*, **43**, 338–340 (1996)