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

L. V. KOGEV*  
Department of Automatics, Technical University of Sofia, 1788 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 not only the circuit equations but also the power supplies from where any a priori lower bounds of the variables are known. A suitable global method is applied to each non-linear function $f(x)$ by an appropriate interval iteration scheme, in the form of a non-linear function having an additive interval term. The boundaries of the 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 (as is investigated) consists of finding all the real solutions of a corresponding system of non-linear algebraic equations. More specifically, the following problem has been referred to as the GA problem and has been considered in numerous publications.

The GA problem, $(x | f(x) = 0)$, is the function describing the dc operation of the circuit studied, where $X = [x_1, \ldots, x_n]$ is a vector containing all the circuit's parameters. The set of solutions $x$ of $f(x) = 0$, $x \in X^n$, are the solutions of all real solutions $x$, $x \in X^n$, to the system $f(x) = 0$.

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

$$f(x) = x_1(x_1 - \epsilon)$$

with

$$\epsilon = \epsilon(x_1, \ldots, x_n)$$

where $x_1(x_1 - \epsilon)$ is the interval function.

---

*Correspondence to: L. V. Kogev, Department of Automatics, Technical University of Sofia, 1788 Sofia, Bulgaria.
which describes a large class of non-linear de-electronic circuits or models the equilibrium state of similar neural networks. If the circuit investigated contains bipolar and MOS transistors, the nonlinear functions in (5) take on the form

\[ f(x) = \begin{cases} \phi(x) - \phi(x,0) & \text{if } x < 0 \\ \phi(x) - \phi(x,1) & \text{if } x \geq 0 \end{cases} \]

The functions \( \phi(x) \) or \( \phi(x,0) \) model physical two-terminal non-linear elements and are, therefore, generally 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 \( \phi(x) \) are often approximated by piecewise linear functions and then the resulting GA problem will be referred to as the PWL problem. The solution of the GA problem presents numerical difficulties, especially for larger \( n \).

Recently, there exist several interval analysis methods that are capable of exactly 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. Indeed, all interval methods known to date involve recourse splitting of the initial box \( X^0 \) into sub-boxes \( X^j \) and locating the solution available in \( X^j \) or concluding its absence in \( X^j \). Unfortunately, the number of \( X^j \) 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. If based on the use of the so-called interval loops (instead of interval determinants used in 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 is 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 separate form.

A function \( f(x) \) on \( x \) variables is said to be separate if it has the following form

\[ f(x) = \sum_{i=1}^{n} f_i(x) \]

where, \( f_i(x) \) is a function of \( x_i \) alone. Clearly, functions (1) and (4) are separable.

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

The method here 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 computational iteration. 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 \( f \) denote any system. \( f \) generated during the computational process by splitting the initial box \( X^0 \).

First, a new interval approximation of a component \( f_i(x) \) in \( X \), will be suggested. No restrictions are made on the functions \( f_i(x) \) as imposed except for the requirement that they be continuous. The above approximation is illustrated graphically in Figure 1 for the case of continuously differentiable (C2) functions.
Unlike the previous methods where the functions $f(x)$ are approximated using interval derivatives or interval slopes, the new approximation of $f(x)$ in $X$ is chosen as the following form:

$$ f(x) = s_{j} = a_{j}x_{j}, \quad x_{j} \in X_{j} $$

where $X_{j} = [b_{j}, c_{j}]$ is an interval whose $a_{j}$ is a real number. Both $b_{j}$ and $a_{j}$ are to be determined such that the following inclusion property should hold:

$$ f(x) \in [b_{j}, c_{j}] = a_{j}x_{j}, \quad x_{j} \in X_{j} $$

A simple and efficient procedure for finding $a_{j}$, $b_{j}$, and $c_{j}$ is regressed here for the case of first functions. It is motivated by elementary geometrical considerations and can be readily modified for the case of PWL functions.

**Procedure 1:** First, compute

$$ f_{j} = f(x_{j}), \quad s_{j} = a_{j} $$

Then, $a_{j}$ is defined as the slope

$$ a_{j} = (c_{j} - b_{j})/s_{j} = s_{j} $$

Afterwards, the following equation is solved for $s_{j}$

$$ s_{j} \leftarrow f_{j}(x) = a_{j} $$

In the general case, there will have several solutions. They all may be located reliably using an interval version of the Newton method of the so-called extended interval Newton algorithm, Reference 5, Sections (4.1). Among them, two solutions denoted $s'$ and $s''$ are to be chosen such that the following conditions should be satisfied:

$$ f(x) \in [s', s''], \quad x_{j} \in X_{j} $$

Let $s_{j}(x) = s_{j} = a_{j}x_{j}$ be a straight line passing through the point $(x_{j}', f(x_{j}'))$ having slope $a_{j}$ such that

$$ f(x) \in [s_{j}(x), x_{j} \in X_{j}] $$

© 1999 John Wiley & Sons, Ltd.
Similarly, let the straight line
\[ l(x) = b_2 + a_2 x \]  
(13)

passing through the point \((x_2', y_2')\) have the property
\[ f(x) > l(x), \quad x \in X_1 \]  
(14)

Now it is easy seen that
\[ b_2 + l(x) = a_2 y_2' \quad b_2 + l(x) = a_2 y_2' \]  
(15a, b)

In the special case when \(l(x)\) is either convex or concave in \(X_1\), equation (15) has a unique solution \(x\). In this case formula (13) can be modified as follows. If \(f(x)\) is convex then the \(x_2 = x_2\) and \(x_2' = x_2\), similarly, in the case of a concave function \(f(x)\) \(x_2 = x_2\) and \(x_2' = x_2\).

The above procedure remains, essentially, the same in the case of PWL functions \(f(x)\). The only difference is that the points \(x_2\) and \(x_2'\) are now found directly on the PWL function by elementary geometrical considerations.

Using the above procedure a real matrix
\[ A = (a_{ij}) \]

and an interval vector \(b = (b_1, \ldots, b_n)\) are formed with
\[ b = \sum_{j=1}^{n} b_j \]  
(16)

On account of (1), (4) and (16)
\[ f(x) = \sum_{j=1}^{n} a_{ij} x_i + b_i, \quad x \in X, \quad i = 1, \ldots, n \]  
(17)

or in vector form
\[ f(x) = A \mathbf{x} + b, \quad x \in X \]  
(18)

If \(y\) is a solution of (1) in \(X\), then \(f(y) = 0\) and by (18)
\[ 0 = A y + b, \quad y \in X \]  
(19)

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

**Theorem 1.** All the solutions \(y\) to
\[ f(x) = 0 \]  
(20)

contained in \(X\) are also contained in the solution set \(S(E)\) of the system
\[ Ax + b = 0, \quad b \in \mathbb{R}^n \]  
(21)

where \(b\) is any real vector contained in \(b\).

The proof of the theorem is given in the appendix.

Since \(S(E)\) is an interval vector set the set \(S(E)\) is a convex polyhedron. Indeed, (21) is in fact a system of a linear equations and 2n two-sided linear inequalities.

We are ready to state another basic result.
Theorem 2. All the solutions \( y \) to (29) in \( X \) are also contained in the intersection
\[
P(\Omega) = 2(\Omega) \cap X
\] (22)
(See the appendix for the proof of the theorem.)

Corollary 1. If \( P(\Omega) \) is empty, i.e., if
\[
3(\Omega) \cap X = \emptyset
\] (23)
then the system (29) has no solution in \( X \).

Let \( H(P, \Omega) \) denote the interval hull of \( P(\Omega) \), that is the smallest (interval) vector (box) containing \( P(\Omega) \). Consider the following iterative procedure:

\[
X^{k+1} = H(P(\Omega))^{k+1} \cap X \quad k \geq 0
\] (24)

Similarly to other methods, the procedure (24) may be used for designing a method for finding all real solutions of (1) in \( X \). Such an approach would, however, be rather costly since \( 2^n \) iterations are required to determine \( H(P(\Omega))^{k+1} \).

Therefore, a simpler and, presumably, more efficient procedure is suggested here.

Let \( H(\Omega, X) \) denote the interval hull of \( 3(\Omega) \). Furthermore, let \( B = -N \) with components
\[
B_{ij} = \begin{cases} -B_{ii} & \text{if } C_{ii} \geq 0 \\ -B_{ii} & \text{if } C_{ii} < 0 \end{cases}
\] (25)

If the matrix \( A \) is not singular, \( H(\Omega, X) \) is given by the formula
\[
H(\Omega, X) = A^{-1}B
\] (26)
Indeed, \( H(\Omega, X) \) is by definition the interval solution of the following linear interval equation:

\[
A^T \begin{bmatrix} x \\ 1 \end{bmatrix} = B^T \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\] (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 I is applied to each box separately.

Let \( C = A^T \) and \( Y = H(\Omega, X) \). It follows from (26) that the components \( Y_{ii} = [L_{ii}, U_{ii}] \) of \( Y \) are given by the formula
\[
Y_{ii} = \frac{1}{C_{ii}} \sum_{j=1}^{n} \frac{L_{ij}}{C_{ij}}
\] (28)

with
\[
L_{ij} = \begin{cases} \frac{L_{ij}}{C_{ij}} & \text{if } C_{ij} > 0 \\ \frac{U_{ij}}{C_{ij}} & \text{if } C_{ij} < 0 \end{cases}
\] (29)

and
\[
U_{ij} = \begin{cases} \frac{U_{ij}}{C_{ij}} & \text{if } C_{ij} > 0 \\ \frac{L_{ij}}{C_{ij}} & \text{if } C_{ij} < 0 \end{cases}
\] (30)

Now, the following iterative procedure based on (26), (31) is suggested here.

Procedure 2. Let \( X^0 \) be the current box generated by the present method. Using Procedure 1 determine \( C^0 \) and \( B^0 \) corresponding to \( X^0 \). By formula (26) and (31) compute \( Y^0 \). The iterative procedure is then
defined as follows:

\[ X^{(n+1)} = F(X^{(n)}), \quad n \geq 0 \]  \hspace{1cm} (32)

The procedure may result in three outcomes:

A. The sequence \( X^{(n+1)} \) converges to a fixed interval box \( X^* \). In practice, the procedure is stopped whenever the reduction in the volume of the current box \( X^{(n+1)} \) as compared to that of the preceding box \( X^{(n)} \) is smaller than a constant \( c_1 \). In this case \( X^{(n+1)} \) is split along its widest side into two boxes \( X^* \) and \( X^* \) and each is tested. The wider box is then the outer box \( X^* \) and the iterative procedure (30)-(32) is repeated.

B. At some \( n \)

\[ F(X^{(n)}) = X^{(n)} \]  \hspace{1cm} (33)

Since \( Y = H(X) \subset \mathbb{R}^D \) is followed by Corollary 1 that system (28) has no solution in \( X^{(n)} \) if (33) becomes valid. In this case \( X^{(n)} \) is discarded further consideration.

C. The sequence \( X^{(n+1)} \) converges to a solution \( X^* \) of \( X \). Actually, the iterations are stopped whenever the width of \( X^{(n+1)} \) becomes smaller than a constant \( c_2 \mbox{ (accuracy with respect to } \\ X^* \mbox{ )}. Now \( X^* \) is approximated by the unique \( X^* \) of \( X^{(n)} \) and \( X^* \) is substituted in (3). If

\[ \max \{ (\| x \|), \quad i = 1, \ldots, k \} > c_2 \]  \hspace{1cm} (34)

\( c_2 \) is the accuracy of \( X^* \) with respect to the system of equations) then the iterations are resumed; otherwise, \( X^* \) is accepted as a solution to (3).

If \( c_2 > c_3 \) then a new box (the first box started in \( L_1 \)) is removed from \( L \), restored \( X^{(n)} \) and Procedure 2 is again applied. It can be shown (similarly to References 4-8) that this process of generating, storing the receiving intervals will terminate in a finite number of iterations keeping all the solutions to (3) within prescribed accuracy or establishing the absence of solutions to (3) in \( X^{(n)} \).

Since the present method is capable of solving both the CFO and PRU 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 \( X = X_0; \quad i = 0; \quad \| x \| \) is the number of iterations \( \| x \| \) is the output in \( X^{(i)} \). \( i = 0 \) (number of boxes stored in \( L \)). Assign values of \( l_0 \) (length of the box \( L_1 \)) and \( c_1 \) and \( c_2 \).

Step 1. Call Procedure 1 to compute the real matrix \( a \) and the interval vector \( b \) corresponding to \( X \).

Step 2. Compute \( C = a \).

Step 3. Call Procedure 2.

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

Step 5. If \( C \) is a constant, compute \( x_b \) and terminate the outermost loop, return to step 1.

Step 6. Compute the outermost \( x_b \), then put the current box into \( X_1 \) and \( X_1 \) into \( L_1 \) and push \( i = i + 1 \).

Return \( X_1 \), to \( X \) and go back to step 1.

Step 7. IF \( i = 0 \), terminate and print \( x_b \). IF \( i = 0 \) then (3) has no solution in \( X^{(n)} \); otherwise, \( x_b \) indicates the number of solutions found in \( L_1 \). Return to step 1.
Step 8. Retrace the first element of L (the first box added to L) to trace a tie to i, return i = 1. Reverse the computation process from step 3.

Algorithm 1 is based on vector operations. Its consequences can be improved if component operations are introduced. Thus whereas a reduction of a component occurs, this will be used immediately (if reducing if possible) the remaining components A, B, . . . , e. 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 Xj. Thus, we omit from Procedure 1 we have the jth column A, of A and an (initial) vector B whose components are B0, 1, . . ., e. For reasons to become clear later, it is expedient to introduce two new matrices B and B' and to store B0 and B0' in the jth row of B (lower end-points) and B' (upper end-points), respectively.

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

\[ X^{i+1} = X^i + \Phi X^i \]  \hspace{1cm} (35)

Now, according to the idea to get the most out of the reduction of \( X^{i+1} \) we check whether

\[ X^{i+1} = X^i \]  \hspace{1cm} (36)

If (36) holds in practice, if the reduction is greater than some threshold (Procedure 3) is immediately called to recompute the corresponding ith column \( A_i \) of A and the ith columns \( B_i \) and \( B'_i \) of B and B', respectively. Let \( \Delta_i, B_i \), and \( B'_i \) denote \( B_i \), \( B'_i \) for \( A_i \), \( B_i \), and \( B'_i \) for \( A_i \). Similarly to (35) \( \Phi_B \) and \( \Phi_{B'} \) designate the corresponding updated matrices \( A_i, B_i \), and \( B'_i \) respectively.

Now, the lower and upper end-points of \( B_i \) are easily evaluated:

\[ B_i = B_{i-1} - \Phi_{B_i} \]  \hspace{1cm} (37a)

\[ B'_i = B'_{i-1} + \Phi_{B'_i} \]  \hspace{1cm} (37b)

It is seen from (37a)-(37b) that to compute the most component \( \Phi_{B_i} \) and \( \Phi_{B'_i} \), we need the updated row \( C_i \), corresponding to the updated matrix \( B_i \). The lower end-point of \( B_i \) is equal to matrix \( A_i \) except for its ith column, \( \Phi_{B_i} \), that is replaced by the updated column \( \Delta_i \). Thus, \( C_i \), can be found by updating the matrix \( C_i \) = \( (\Phi_{B_i})^{-1} \) when the ith column \( A_i \) of \( A_i \) is modified by the increment vector \( \Delta_i \), \( \Delta_i = \Delta_i \)  \hspace{1cm} (38)

Then the updated matrix \( C_i \) can be found in order to compute the updated matrix \( \Phi_{B_i} \) by multiplying

\[ C_i = C_{i-1} - \frac{1}{z} C_{i-1} Q_i C_{i-1} \]  \hspace{1cm} (38)

with

\[ z = 1 + |\Phi_{B_i}| \]

where \( Q_i \) is a matrix whose all columns are zero except for the ith column which is \( \Delta_i \).

Having determined \( C_i \), finally \( \Phi_{B_i} \), is evaluated by forward/back (36) (37) when \( C_i \), and \( B_i \) are used. Now, we are in a position to proceed Procedure 4.

Procedure 4. For a given \( \Delta_i \), corresponding to a fixed j (call Procedure 3 to yield the new vector \( \Delta_i, B_i \), and \( B'_i \)), compute the interval vector \( B^i \) by (37). Determine the real vector \( C^i \), for the j = (ith row of \( C^i \) ) component by (39). Finally, evaluate \( \Phi_{B_i} \) by (36) (37).

© 1989 John Wiley & Sons Ltd.

(Original Page Date: 20/01/92 15:00)
The compositewise algorithm incorporates an improved exclusive rule (as compared to (32)) which is based on (23a). From the definitions of 2.4 it is easily seen that condition (23a) is fulfilled for the rh-coordinates if it does not intersect the 'bar' defined by

\[ \sum_{j=1}^{n} a_j y_j \leq R. \]  

Now, define

\[ L = \sum_{j=1}^{n} x_j \]  

where

\[ x_j = \begin{cases} a_j y_j & \text{if } a_j \geq 0 \\ a_j y_j & \text{if } a_j < 0 \end{cases} \]  

and

\[ L = \sum_{j=1}^{n} x_j \]  

\[ x_j = \begin{cases} a_j y_j & \text{if } a_j \geq 0 \\ a_j y_j & \text{if } a_j < 0 \end{cases} \]  

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

\[ L \geq R \quad \text{or} \quad L < R. \]  

The compositewise algorithm is thus based on the following procedure.

**Procedure 3**

Step 1. Let \( i = 0 \).

Step 2. If \( i > 0 \) go to Step 4, else proceed to next step.

Step 3. If the exclusion rule (43) holds, then go to Step 5, otherwise go to next step.

Step 5. Call Procedure 4 to yield \( Y_i \). If \( Y_i = x_i \), then call Procedure 3 and go to Step 1, otherwise go to next step.

Step 6. The Procedure is terminated with outcome B.

Step 7. If (34) holds, go to Step 5, otherwise terminate with outcome C.

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

**Algorithm 2**

Steps 1-2. Same as in Algorithm 1.

Step 3. Call Procedure 3.

Step 4a. If Procedure 3 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-rh circuits because of the iteration of \( \rho \). Algorithm 2 can, however, be readily adapted to encompass the case of large \( \rho \) (in fact, by formular (35)–(37)).
GLOBAL ANALYSIS OF NON-LINEAR RESISTIVE CIRCUITS

The vector \( \mathbf{v} \) has been evaluated, we only need the 4th row of \( \mathbf{C} = \mathbf{D}^{-1} \) to determine \( \mathbf{y}_c \). Thus, \( \mathbf{C} \) can be found by solving the linear system

\[
\mathbf{C} \mathbf{y}_c = \mathbf{v}
\]

(46)

where the symbol \( \mathbf{T} \) means transpose and \( \mathbf{1} \) is the 4th column of the identity (or \( \mathbf{1} \times \mathbf{4} \)) matrix \( \mathbf{E} \). Indeed

\[
\mathbf{C} = \mathbf{L} \mathbf{T} \mathbf{E}^{-1}
\]

(47)

and (46) follows from (45).

For large-scale systems, \( \mathbf{A} \) is a rather sparse matrix and sparse matrix methods should be employed for solving \( \mathbf{Ax} = \mathbf{b} \) in this case. Additional improvement is possible if one takes into account that according to (45)

\[
\mathbf{A} \mathbf{x} = \mathbf{b}
\]

(45)

\( \mathbf{A} \mathbf{x} = \mathbf{b} \), \( \mathbf{x} \) is perturbed at each iteration by only a column vector \( \Delta \mathbf{x} \). The perturbations can be written equivalent in the form of a 4-dim. vec.

\[
\mathbf{x} = \mathbf{A}^{-1} \mathbf{b} + \Delta \mathbf{x}
\]

(48)

where \( \Delta \mathbf{x} \) is a column vector whose elements are zero except for its 4th element which is equal to one. Then a very efficient sparse matrix method [5] which exploits the above fact to perform perturbations can be made use of.

Remark 1. To avoid clustering up the generation of the present method with network details, nothing has been said about the non-linear clustering phenomenon characteristic of any interval method for solving systems of nonlinear equations. If clustering occurs this means that one or the method generates towards each output a certain number of bounded \( \mathbf{x}_n \) solution of the same width of \( \mathbf{x}_n \). However, unlike the box \( \mathbf{x}_n \), the remaining boxes do not contain the solution \( \mathbf{x}_n \). Clustering appears, essentially, because the exclusion rule used is not capable of excluding small enough boxes around a solution. The clustering effect grows stronger as the problem dimension increases. It has been observed with all the known interval methods for global optimization. For one and the same problem, the clustering effect depends on the method used and 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, clustering was detected 'manually' by inspection of the solutions. In the present method, simple procedure for detecting and diagnosing clustering has been incorporated in Algorithms 1 and 2 in Procedure 2, described in section C.

Remark 2. Finally, a remark regarding the interval arithmetic implementation of the present interval method is due. Any interval method is to be implemented using appropriate interval arithmetic to ensure strict enclosure of the numerical results obtained. The present method for solving (1) requires interval rounding of all elementary arithmetic operations involved to guarantee that the intervals \( \mathbf{x}_n \) obtained in Procedure 2, described in section C, will cover the exact solutions \( \mathbf{x}_n \).

Interval rounding also ensures that no solution is ever lost in the process of reducing the size of the current box \( \mathbf{x}_n \) or when discarding another box \( \mathbf{x}_n \). Thus, the set of interval arithmetic makes the present method self-validating, guaranteeing 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 5-10 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 nonlinear analysis can be implemented alternatively by a stochastic method [6] without appealing to interval computations at all. Unlike the interval implementation which yields interval results covering the exact solutions with mathematical certainty, the stochastic 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 a ranging from $n = 3$ to $n = 10$, using Algorithm 1. At this stage, all calculations are implemented using traditional finite-point arithmetic operations; but all the same, 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 9). The description (3) is in the case given by three equations with

$\phi(x_i) = 10^{-5} x_i^{3n} - 1$

$\phi(x_i) = 10^{9} x_i^{3n} - 1$

$\phi(x_i) = 10^{-5} x_i^{3n} - 1$

The problem is to find all the operating points of the circuit (the set $\mathbf{X}^{**}$) within the interval region $\mathbf{X}^{**} = [0, 1], [-5, 5], [0, 1]$

The number of iterations needed by the present method to locate the only solution within $\mathbf{X}^{**}$ with the desired accuracy is $N_1 = 7$.

**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

$\phi(x_i) = 9 x_i^{2} - 10 x_i^{2} + 11 x_i, \quad i = 1, \ldots, 10$

$H = 0$ with $\lambda_i = -1$

$S = \{-1, -2, -3, \ldots, -10\}$

It should also be mentioned that no damping 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

$\phi(x_i) = 9 x_i^{2} - 10 x_i^{2} + 11 x_i, \quad i = 1, \ldots, 10$

$H = 0$ with $\lambda_i = -1$

$S = \{-1, -2, -3, \ldots, -10\}$

It should also be mentioned that no damping 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.
The interval $A_i^k$ is defined by

$$A_i^k = [L_i, U_i], \quad i = 1, \ldots, n$$

The accuracy $\varepsilon_i$ has been chosen to be $10^{-6}$.

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 (8). The second method designated as M2 is an improved version which uses interval slopes. This paper's method denoted as M2 has also solved the problem considered and has found within the same accuracy $\varepsilon_i = 10^{-6}$ all the data solutions contained in $A_i^k$. Moreover, the data in Table II concerning the number of iterations required to solve the CDF problems for the circuit model reveal that the present method is vastly superior to the best interval methods known to us at present.

On account of its fast convergence rate the new method has also improved characteristics as regards memory volume requirements. Indeed, the maximum number of terms $n_i$ stored during computation reached the value of 2 for both methods M1 and M2. Also, the clustering effect is now by far less pronounced. Thus, Algorithm 1 (without the procedure for detecting the clustering mentioned in Reference 1) produced only a total of 13 solutions and it was very easy to locate the clustering boxes. In contrast, among the two previous methods, even the better method M2 possessed dozens of clustering boxes, thus requiring much larger memory volumes.

5. CONCLUSIONS

In this paper the problem of finding the set of all operating points of non-linear passive circuits whose equations are written in the separable form (15) is considered. The only assumption about the characteristics of the non-linear elements involved is that they are modeled by continuous functions $f(x)$. Thus, both the CDF and PWL problems are accompanied by the 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 function $f(x)$ by linear interval functions $f(x)$ defined by (15). Thus, at each iteration of the method a single linear interval system (77) with a large matrix is solved. This dramatically diminishes the present method from the other known interval methods when 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 ranks considerably better than the other known methods for solving the global analysis problem considered. Its superiority becomes much more pronounced as the dimension $m$ of the circuit equations system increases.

The present method can be extended to systems of algebraic equations of arbitrary form. This may be done by transforming the original system into an augmented separable form system using a method recently suggested in Reference 14. Thus the system thus obtained is, generally, much larger than the original system such an approach should appeal to the space matrix version of Algorithm 2 before suggested to ensure its numerical efficiency. Work is presently in progress to implement this scheme.
This work was supported by the Bulgarian Science Foundation under Contract No TN-5149. The author is grateful to one of the reviewers for his valuable comments.

APPENDIX

Proof of Theorem 1

If x ∈ S is a solution to (20) in X then $f(x) = 0$ by the definition of $A$ and if $D(x)$ holds on account of (40). Therefore, the following equality:

$$Ax = b^0$$

defines a set of points $S_{E(x)}$ which contains every solution to (22) available in X. The interval equation (22) in a short form of representing the system of equations:

$$Ax + b = 0, \quad b \in S$$

Hence all solutions to (22) are in the set of solutions of equation 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 $y \in S_{E(x)}$ are in $S_{E(x)}$ but at the same time they are in $X$. Hence all solutions $y$ are also in the intersection

$S_{E(x)} \cap X$.

REFERENCES