Proceedings of the 10th WSEAS International Conference on CIRCUITS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp173-178)

[Invited Paper]

An Efficient Algorithm for Finding All DC Solutions of Nonlinear Circuits

KIYOTAKA YAMAMURA, KOKI SUDA, and WATARU KUROKI
Department of Electrical, Electronic, and Communication Engineering
Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551
JAPAN

Abstract: - An efficient algorithm is proposed for finding all DC solutions of nonlinear (not piecewise-linear) circuits with mathematical certainty. This algorithm is based on interval analysis, the LP test using the dual simplex method, the contraction method, and a special technique which makes the algorithm not require large memory space and not require copying tableaus. By numerical examples, it is shown that the proposed algorithm could find all solutions of a system of 2 000 nonlinear circuit equations in acceptable computation time.

Key–Words: - Circuit simulation, DC analysis, nonlinear circuit, finding all solutions

1 Introduction

Finding all DC solutions of nonlinear circuits is an important problem in circuit simulation. Many algorithms have been proposed for finding all DC solutions of piecewise-linear (PWL) circuits which are obtained by PWL approximation of nonlinear functions [1]–[12]. However, no algorithm has been used in practical circuit simulation because this problem is essentially very difficult. Namely, if the number of PWL resistors is n and if each PWL function consists of K segments, then the total number of linear regions (denoted by L) on which we seek for solutions is K^n. Hence, the computation time of the algorithms generally grows exponentially with n.

Recently, several algorithms succeeded in finding all DC solutions of PWL circuits with $n \geq 100$ in acceptable computation time [8]–[12]. That is, in 1996, only small circuits with $n \leq 14$ could be solved [7], but in 1998, a problem where $n = 100$ and $L = 10^{100}$ was solved for the first time [8]. In 2000, a problem where $n = 200$ and $L = 10^{200}$ was solved [10], and in 2002, a problem where $n = 300$ and $L = 10^{300}$ was solved [11]. In 2003, a problem where $n = 500$ and $L = 10^{500}$ was solved in practical computation time [12]. Thus, the study of algorithms for finding all DC solutions of PWL circuits has been remarkably developed, if we do not consider the errors of PWL approximation.

However, PWL approximation often changes the number of solutions. In the numerical experiments of the above papers, PWL functions consisting of ten segments ($K = 10$) are used, but the PWL approximation with such a small K often vanish some solutions which actually exists as shown later. In order to design a circuit with high reliability, it is necessary to find all solutions of the original nonlinear circuits. This problem reduces to finding all solutions of systems of nonlinear equations, which is a much more difficult problem than the PWL case.

As a computational method to find all solutions of nonlinear equations, interval analysis based techniques are well-known [13], and various algorithms based on interval computation have been developed [14]–[16]. Using the interval algorithms, all solutions of nonlinear equations contained in a given box[1] can be found with mathematical certainty. However, the computation time of the interval algorithms generally grows exponentially with the number of variables n. Therefore, in order to apply the interval algorithms to practical circuits, it is necessary to improve the computational efficiency from various viewpoints. Recently, all solutions of systems of 200 nonlinear equations could be obtained in practical computation time [16]. At the present time, the algorithm proposed in [16] is one of the most efficient algorithms for finding all solutions of systems of nonlinear equations.

In this paper, we extend the algorithms in [16],

[1] An n-dimensional rectangular region with the sides parallel to the coordinate axes will be called a box.

and propose an efficient algorithm for finding all DC solutions of nonlinear circuits.

2 Basic Algorithm

In this section, we first summarize the basic procedures of interval algorithms [13].

Consider a nonlinear resistive circuit described by a system of n nonlinear equations [1]:

$$f(x) \stackrel{\triangle}{=} Pg(x) + Qx - r = 0 \tag{1}$$

where $x = (x_1, x_2, \cdots, x_n)^T \in R^n$ is a variable vector, $r = (r_1, r_2, \cdots, r_n)^T \in R^n$ is a constant vector, P and Q are $n \times n$ constant matrices, and $g(x) = [g_1(x_1), g_2(x_2), \cdots, g_n(x_n)]^T$ is a nonlinear function with component functions $g_i(x_i) : R^1 \to R^1$ $(i = 1, 2, \cdots, n)$. In this paper, we discuss the problem of finding all solutions of (1) contained in an initial region D which is given by an n-dimensional box.

An n-dimensional interval vector with components $[a_i, b_i]$ $(i = 1, 2, \cdots, n)$ is denoted by

$$X = ([a_1, b_1], [a_2, b_2], \cdots, [a_n, b_n])^T. \tag{2}$$

Geometrically, X is an n-dimensional box.

In interval algorithms, the following procedure is performed recursively, beginning with the initial box $X = D$. At each level, we analyze the box X. If there is no solution of (1) in X, then we exclude it from further consideration. If there is a unique solution of (1) in X, then we compute it by some iterative method. In the field of interval analysis, computationally verifiable sufficient conditions for nonexistence, existence and uniqueness of a solution in X have been developed. If these conditions are not satisfied and neither existence nor nonexistence of a solution in X can be proved, then bisect X in some appropriately chosen coordinate direction to form two new boxes; we then continue the above procedure with one of these boxes, and put the other one on a stack for later consideration. Thus, we can find all solutions of (1) contained in $D \subset R^n$ with mathematical certainty.

However, the computation time of the interval algorithms tends to grow exponentially with n. One of the difficulties of these algorithms is that the number of boxes to be analyzed is extremely large for large scale systems. Therefore, it is necessary to develop a powerful test for nonexistence of a solution in a given box so that we can exclude many boxes containing no solution at an early stage of the algorithm.

Next, we summarize the powerful nonexistence test proposed in [9],[14], and [16].

In this test, the nonexistence of a solution to (1) is checked as follows. We first calculate the interval extensions [2] of $g_i(x_i)$ $(i = 1, 2, \cdots, n)$ over $X = ([a_1, b_1], \cdots, [a_n, b_n])^T$. Let the interval extension of $g_i(x_i)$ over $[a_i, b_i]$ be $[c_i, d_i]$. Then, we introduce auxiliary variables y_i $(i = 1, 2, \cdots, n)$ and put $y_i = g_i(x_i)$. If $a_i \le x_i \le b_i$, then $c_i \le y_i \le d_i$.

Now we replace each nonlinear function $g_i(x_i)$ in (1) by the auxiliary variable y_i and the linear inequality $c_i \le y_i \le d_i$, and consider the linear programming (LP) problem:

$$\begin{aligned} &\max \text{ (arbitrary constant)} \\ &\text{subject to} \\ &\quad Py + Qx - r = 0 \\ &\quad a_i \le x_i \le b_i, \qquad i = 1, 2, \cdots, n \\ &\quad c_i \le y_i \le d_i, \qquad i = 1, 2, \cdots, n \end{aligned} \tag{3}$$

where $y = (y_1, y_2, \cdots, y_n)^T \in R^n$. Then, we apply the simplex method to (3).

Evidently, all solutions of (3) which exist in X satisfy the constraints in (3) if we put $y_i = g_i(x_i)$. Namely, the feasible region of the LP problem (3) is a convex polyhedron containing all solutions of (1) in X. Hence, if the feasible region is empty, then we can conclude that there is no solution of (1) in X.

The emptiness or nonemptiness of the feasible region of (3) can be checked by the simplex method. If the simplex method terminates with the information that the feasible region is empty, then there is no solution of (1) in X. This test is called the LP test.

By introducing the LP test to the interval algorithms (such as the Krawczyk-Moore algorithm [13]), all solutions of (1) can be found very efficiently. In [14], this algorithm solves a system of nonlinear equations with $n = 60$ in practical computation time, although the original Krawczyk-Moore algorithm can solve the system only for $n \le 12$.

In [16], it is shown that the LP test can be performed with a few pivotings (often no pivoting) per box by using the dual simplex method from the second box. Using this technique, the LP test becomes not only powerful but also efficient. In [16], this improved LP test is introduced to the Krawczyk-Moore algorithm, which succeeded in finding all solutions to systems of nonlinear equations with $n = 200$.

[2] The interval extension of $g_i(x_i)$ can be calculated by replacing the variable x_i with the interval $[a_i, b_i]$ and by replacing the arithmetic operations with the corresponding interval operations [13]. It is known that the interval extension contains the range of $g_i(x_i)$ over $[a_i, b_i]$.

3 Proposed Algorithm

Although the LP test proposed in [16] requires only a few pivotings per box, it has to be performed on many boxes, especially when K is large. In order to decrease the number of boxes on which the LP test is performed, we introduce the contraction methods proposed in [15] which contract a box X to a smaller box $\bar{X}$ containing the same solutions [3].

In the proposed algorithm, we use the algorithm in [16] as the base and perform the contraction method each time after the LP test is performed on a box X. Then, we bisect the reduced box $\bar{X}$ and repeat the same procedure on the sub-boxes.

However, there is one more problem in the LP test algorithm when it is applied to large scale problems. Namely, since the algorithm has the structure of a binary tree, it requires very large memory space. In other words, tableaus of the dual simplex method have to be copied and reserved at each node of the binary tree. Moreover, in our past numerical experiments, the time needed for copying tableaus occupied a large part of the total computation time of the algorithm in [16]. In order to overcome this problem, we propose a technique for making the algorithm not require large memory space and not require copying tableaus.

When we apply the simplex method to (3), we first apply the variable transformation $\bar{x}_i = x_i - a_i$ and $\bar{y}_i = y_i - c_i$, and introduce the slack variables $\bar{\lambda}_i$ and $\bar{\mu}_i$ $(i = 1, 2, \cdots, n)$ so that the LP problem is transformed into a standard form:

$$
\begin{aligned}
&\max \text{ (arbitrary constant)}\\
&\text{subject to}\\
&\quad P\bar{y} + Q\bar{x} - \bar{r} = 0\\
&\quad \bar{x}_i + \bar{\lambda}_i = b_i - a_i, \qquad i = 1, 2, \cdots, n\\
&\quad \bar{y}_i + \bar{\mu}_i = d_i - c_i, \qquad i = 1, 2, \cdots, n\\
&\quad \bar{x}_i \geq 0,\ \bar{y}_i \geq 0, \qquad i = 1, 2, \cdots, n\\
&\quad \bar{\lambda}_i \geq 0,\ \bar{\mu}_i \geq 0, \qquad i = 1, 2, \cdots, n
\end{aligned}
\tag{4}
$$

where $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)^T$, $\bar{y} = (\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_n)^T$, and $\bar{r} = r - P(c_1, c_2, \cdots, c_n)^T - Q(a_1, a_2, \cdots, a_n)^T$. Then, we construct the initial tableau.

We explain the proposed idea using Figs. 1 and 2. Consider that we have performed the LP test on a box X in Fig. 1 and have obtained an optimal tableau for (4). Here, the term *optimal* implies that the optimality condition is satisfied in the *auxiliary objective function row* because the simplex method for (3) consists of Phase I only. Then, consider that we next perform the LP test on a box X'. Let $[c_i, d_i]$ and $[c'_i, d'_i]$ be the ranges of $g_i(x_i)$ over $[a_i, b_i]$ and $[a'_i, b'_i]$, respectively, as shown in Fig. 2. In the LP test for X', we similarly introduce auxiliary variables y_i $(i = 1, 2, \cdots, n)$ and consider the LP problem:

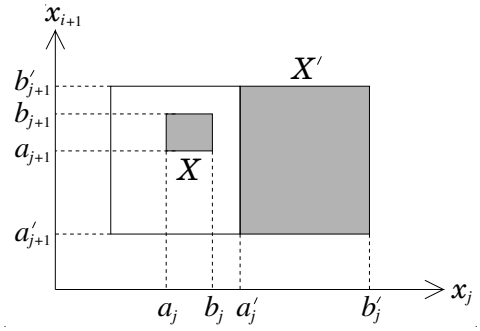

Figure 1: Assume that the LP test is performed on X' after X.

$$
\begin{aligned}
&\max \text{ (arbitrary constant)}\\
&\text{subject to}\\
&\quad Py + Qx - r = 0\\
&\quad a'_i \leq x_i \leq b'_i, \qquad i = 1, 2, \cdots, n\\
&\quad c'_i \leq y_i \leq d'_i, \qquad i = 1, 2, \cdots, n.
\end{aligned}
\tag{5}
$$

Applying the variable transformation $\tilde{x}_i = x_i - a'_i$ and $\tilde{y}_i = y_i - c'_i$, and introducing the slack variables, (5) is transformed into a standard form:

$$
\begin{aligned}
&\max \text{ (arbitrary constant)}\\
&\text{subject to}\\
&\quad P\tilde{y} + Q\tilde{x} - \tilde{r} = 0\\
&\quad \tilde{x}_i + \tilde{\lambda}_i = b'_i - a'_i, \qquad i = 1, 2, \cdots, n\\
&\quad \tilde{y}_i + \tilde{\mu}_i = d'_i - c'_i, \qquad i = 1, 2, \cdots, n\\
&\quad \tilde{x}_i \geq 0,\ \tilde{y}_i \geq 0, \qquad i = 1, 2, \cdots, n\\
&\quad \tilde{\lambda}_i \geq 0,\ \tilde{\mu}_i \geq 0, \qquad i = 1, 2, \cdots, n.
\end{aligned}
\tag{6}
$$

However, from $\bar{x}_i = x_i - a_i$ and $\tilde{x}_i = x_i - a'_i$, $\bar{x} = \tilde{x}_i + (a'_i - a_i)$ $(i = 1, 2, \cdots, n)$ hold. Then, from $\bar{x}_i + \bar{\lambda}_i = b_i - a_i$ in (4) and $\tilde{x}_i + \tilde{\lambda}_i = b'_i - a'_i$ in (6), $\bar{\lambda}_i = \tilde{\lambda}_i + (b_i - b'_i)$ $(i = 1, 2, \cdots, n)$ hold. Similarly, $\bar{y}_i = \tilde{y}_i + (c'_i - c_i)$ and $\bar{\mu}_i = \tilde{\mu}_i + (d_i - d'_i)$ $(i = 1, 2, \cdots, n)$ hold. Substituting these relations to the previous optimal tableau for (4), the optimal tableau for (6) is easily obtained, which differs from the previous tableau only in the constant column.

Of course, this tableau may not be feasible (i.e., all elements in the constant column may not be non-

[3] In [15], an algorithm for finding all DC solutions of nonlinear circuits is proposed using the contraction method. However, the effectiveness of this algorithm to large scale problems is not clear because in the numerical experiments of [15], the algorithm was applied only to small circuits with $n \leq 10$.

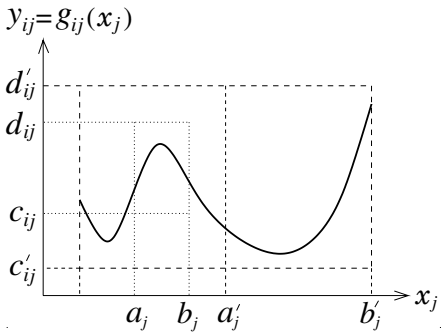

Figure 2: Illustration of the interval extensions of $g_i(x_i)$ over $[a_i, b_i]$ and $[a'_i, b'_i]$.

negative), but always dual feasible because the optimality condition is satisfied. Hence, starting from this tableau, we can perform the dual simplex method and check the existence of the feasible region of (5). Thus, the LP test using the dual simplex method can be performed without copying (reserving) the tableau at each node.

In most cases, this dual simplex method requires only a few pivotings. It often requires no pivoting; namely, if the dual feasible tableau is feasible (i.e., all elements in the constant column are non-negative) or the tableau indicates that the feasible region is empty, then the dual simplex method terminates with no pivoting. Hence, the average number of pivotings per box becomes very small.

This technique also improves the computational efficiency substantially, because as stated before, the time needed for copying tableaus occupies a large part of the total computation time in the original algorithm.

4 NUMERICAL EXAMPLES

We introduced the proposed techniques to the well-known Krawczyk-Moore algorithm [13] and implemented the new algorithm using the programming language C on a Sun Blade 2000 (UltraSPARC-III Cu 1.2GHz). In this section, we show some numerical examples.

Example 1: Consider a system of n nonlinear equations:

$$g(x_i) + x_1 + x_2 + \cdots + x_n - i = 0, \quad i = 1, 2, \cdots, n$$
$$\text{where} \quad g(x) = 2.5x^3 - 10.5x^2 + 11.8x$$

which describes a nonlinear resistive circuit containing n tunnel diodes [7]–[12],[14],[16]. The initial region is $D = ([-10, 10], \cdots, [-10, 10])^T$. Note that the conventional Krawczyk-Moore algorithm could solve this system for $n = 12$ in about three hours, and for $n = 14$ in about 44 hours in [14].

Table 1: Comparison of computation time.

n	S	Ref.[14] T (s)	Ref.[16] T (s)	Proposed T (s)
100	9	71 306	1 060	2
200	13	∞	26 748	21
300	11	∞	∞	65
400	9	∞	∞	124
500	13	∞	∞	333
600	11	∞	∞	543
700	9	∞	∞	632
800	11	∞	∞	1 788
900	19	∞	∞	3 704
1 000	17	∞	∞	5 706
1 100	9	∞	∞	4 387
1 200	9	∞	∞	6 377
1 300	21	∞	∞	22 123
1 400	9	∞	∞	10 741
1 500	13	∞	∞	22 501
1 600	23	∞	∞	58 157
1 700	11	∞	∞	33 638
1 800	9	∞	∞	30 745
1 900	9	∞	∞	41 906
2 000	9	∞	∞	48 805
2 100	11	∞	∞	77 179
2 200	23	∞	∞	173 785
2 300	15	∞	∞	169 243
2 400	9	∞	∞	85 771
2 500	9	∞	∞	136 934

Table 1 compares the computation time of the algorithm in [14], that in [16], and the proposed algorithm, where S denotes the number of solutions obtained by the algorithms, T (s) denotes the computation time, and ∞ denotes that it could not be computed in practical computation time (in this paper, less than 2 days) or memory overflow occured. As seen from the table, the proposed algorithm could solve this system for $n = 1000$ in about 1.5 h, and for $n = 2000$ in about 13 h. This is the first numerical example where all solutions of a system of 2 000 nonlinear circuit equations are found in practical computation time.

Fig. 3 illustrates the growth of the computation time when n increases. It is seen that the computation time of the proposed algorithm grows exponentially, but not very explosively. It is also seen that the

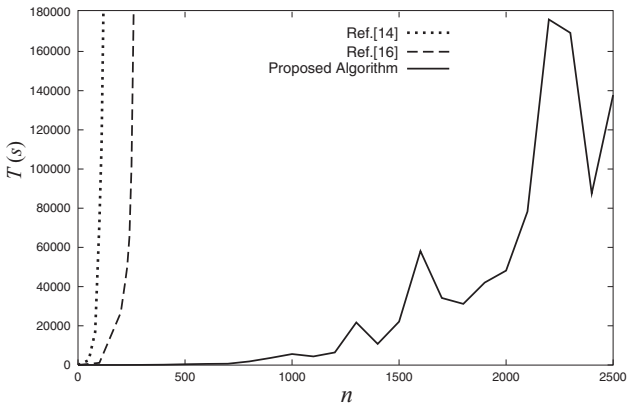

Figure 3: Computation time of the algorithms.

Table 2: Result in Ref. [12].

n	L	S	T (s)
100	10^{100}	9	319
200	10^{200}	9	2 620
300	10^{300}	11	2 758
400	10^{400}	3	8 305
500	10^{500}	3	21 364

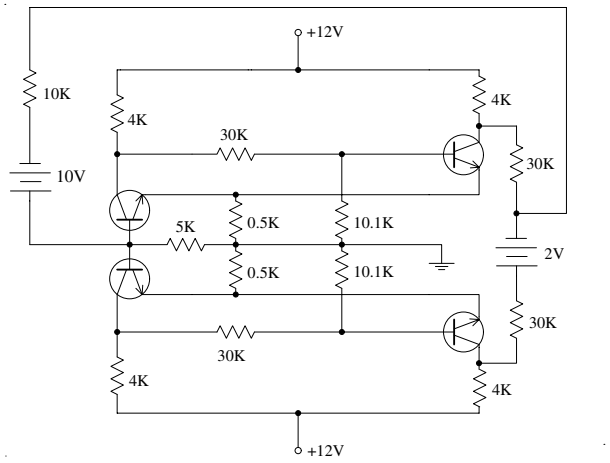

Figure 4: Transistor circuit 1.

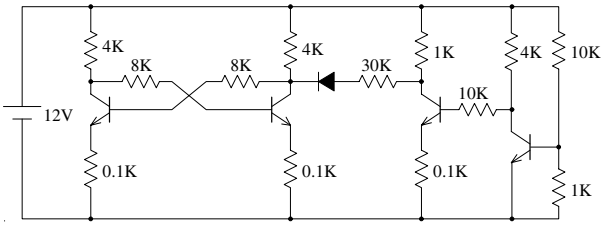

Figure 5: Transistor circuit 2.

computation time depends largely on the number of solutions.

Table 2 shows the result of computation described in [12] where the algorithm proposed in [12] was applied to the PWL versions of the n-tunnel diodes circuits on a 450MHz computer. In Table 2, $L = 10^n$ denotes the total number of linear regions. Namely, the nonlinear function representing the characteristic of the tunnel diodes is approximated by a PWL function with ten segments. As seen from Tables 1 and 2, PWL approximation often changes the number of solutions. Thus, finding all solutions of the original nonlinear equations is important to design a circuit with high reliability.

We also compare the memory space needed in the algorithms. When $n = 200$, the algorithm in [16] needed 334204 Kbyte memory, but the proposed algorithm needed only 7546 Kbyte memory. We also note that, when $n = 1000$, the proposed algorithm needed 158632 Kbyte memory, and when $n = 2000$, it needed 627864 Kbyte memory, although the algorithm in [16] could not solve this system for $n > 290$ on a 1G RAM computer.

Example 2: We next solved the transistor circuits shown in Figs. 4–7 [1]–[12]. Then, we found 9, 3, 11, and one solution(s) in 0.08, 0.06, 1.55, and 1.09 s, respectively. It is seen that all solutions were found in little computation time.

5 CONCLUSION

In this paper, an efficient algorithm has been proposed for finding all DC solutions of nonlinear circuits. It has been shown that the proposed algorithm does not require large memory space and does not require copying tableaus, which improves the computational efficiency substantially. It has also been shown that the proposed algorithm could find all solutions of a system of 2 000 nonlinear circuit equations in practical computation time for the first time. Since the proposed algorithm does not use PWL approximation, it is useful to design a multistate circuit with high reliability.

Acknowledgements: The authors wish to thank Mr. Akinori Machida of Chuo University for his help in the numerical experiments. This work was supported in part by the 21st Cenury Center of Excellence Program "Research on Security and Reliability in Elec-

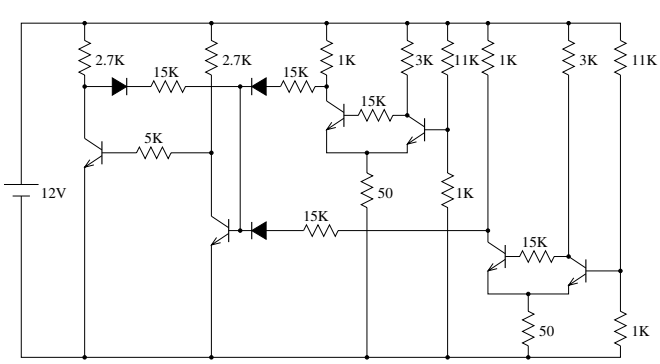

Figure 6: Transistor circuit 3.

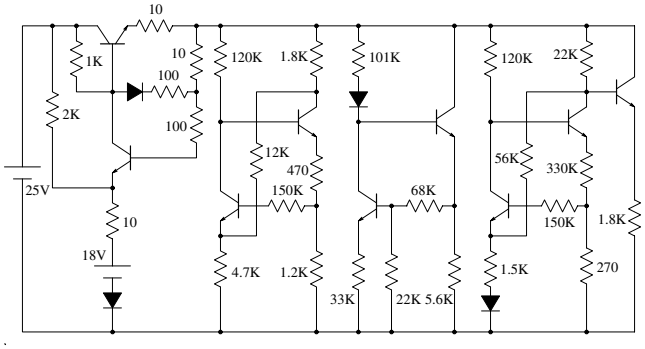

Figure 7: Transistor circuit 4.

tronic Society" of the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References:

[1] L. O. Chua and R. L. P. Ying, Finding all solutions of piecewise-linear circuits, *Int. J. Circuit Theory Appl.*, Vol.10, No.3, 1982, pp. 201–229.

[2] Q. Huang and R. Liu, A simple algorithm for finding all solutions of piecewise-linear networks, *IEEE Trans. Circuits Syst.*, Vol.36, No.4, 1989, pp. 600–609.

[3] T. Nishi, An efficient method to find all solutions of piecewise-linear resistive circuits, *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, 1989, pp. 2052–2055.

[4] K. Yamamura and M. Ochiai, An efficient algorithm for finding all solutions of piecewise-linear resistive circuits, *IEEE Trans. Circuits Syst. I*, Vol.39, No.3, 1992, pp. 213–221.

[5] S. Pastore and A. Premoli, Polyhedral elements: A new algorithm for capturing all the equilibrium points of piecewise-linear circuits, *IEEE Trans. Circuits Syst. I*, Vol.40, No. 2, 1993, pp. 124–132.

[6] K. Yamamura, Finding all solutions of piecewise-linear resistive circuits using simple sign tests, *IEEE Trans. Circuits Syst. I*, Vol.40, No.8, 1993, pp. 546–551.

[7] K. Yamamura and M. Mishina, An algorithm for finding all solutions of piecewise-linear resistive circuits, *Int. J. Circuit Theory Appl.*, Vol.24, No.2, 1996, pp. 223–231.

[8] K. Yamamura and T. Ohshima, Finding all solutions of piecewise-linear resistive circuits using linear programming, *IEEE Trans. Circuits Syst. I*, Vol.45, No.4, 1998, pp. 434–445.

[9] K. Yamamura and K. Yomogita, Finding all solutions of piecewise-linear resistive circuits using an LP test, *IEEE Trans. Circuits Syst. I*, Vol.47, No.7, 2000, pp. 1115–1120.

[10] K. Yamamura and S. Tanaka, Performance evaluation of the LP test algorithm for finding all solutions of piecewise-linear resistive circuits, *Int. J. Circuit Theory Appl.*, Vol.28, No.5, 2000, pp. 501–506.

[11] K. Yamamura and S. Tanaka, Finding all solutions of piecewise-linear resistive circuits using the dual simplex method, *Int. J. Circuit Theory Appl.*, Vol.30, No.6, 2002, pp. 567–586.

[12] K. Yamamura and R. Kaneko, Finding all solutions of piecewise-linear resistive circuits using the simplex method, *IEEE Trans. Circuits Syst. I*, Vol.50, No.1, 2003, pp. 160–165.

[13] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, Philadelphia, 1979.

[14] K. Yamamura, H. Kawata, and A. Tokue, Interval solution of nonlinear equations using linear programming, *BIT—Numerical Mathematics—*, Vol.38, No.1, 1998, pp. 186–199.

[15] L. V. Kolev, An efficient interval method for global analysis of non-linear resistive circuits, *Int. J. Circuit Theory Appl.*, Vol.26, No.1, 1998, pp. 81–92.

[16] K. Yamamura and T. Fujioka, Finding all solutions of systems of nonlinear equations using the dual simplex method, *Proc. Int. Symp. Nonlinear Theory and its Applications*, Zao, Japan, 2001, pp. 219–222.