Worst-Case Tolerance Analysis of Linear DC and AC Electric Circuits

Ludomir Kulsar, Senior Member, IEEE

1. Introduction

Worst-case tolerance analysis of linear circuits is a well-established area of circuit theory. The traditional approach to handling this problem is to use the Monte Carlo method. Starting with the pioneer works (1) and (2), an alternative approach based on the application of interval analysis and the concept of interval analysis  has been in existence for several decades. The methods utilizing the latter approach are known as interval methods (3-15), and the bibliography therein. Because of their high reliability, the interest in these methods has lately considerably increased.

Most worst-case tolerance analysis problems for linear circuits can be formulated in the following ways:

1. In explicit form as corresponding global optimization problems (2).
2. In implicit form as a system of linear equations (16).
3. In implicit form as a system of linear equations with interval coefficients (17-21).

This paper falls into the latter group of formulations. All known methods pertaining to this group are based, in one way or another, on the exact or approximate solution of a system of linear equations whose elements are either independent variables (in the case of direct circuits) or interval coefficients (in the case of circuits with independent variables). The requirement for independence of the elements implies degenerate situations in the form of the system (22).
II. FORMULATION OF THE T3 PROBLEM

In this paper, an arbitrary T3 problem can be formulated as the following manner. First, a real linear algebraic system of equations is set up

\[ A\beta = \theta \]  

(1a)

where \( A \) is a positive-definite matrix, \( \beta \) and \( \theta \) are \( n \times 1 \) vectors and \( A \) is a \( n \times n \) matrix, respectively. The elements of \( A_{ij} \) and \( \theta_i \) are, in general, nonlinear functions of \( n \) parameters

\[ n_{i}(\lambda) = \sum_{k=1}^{m} \lambda_k \]  

(1b)

\( \lambda \) and the parameters \( n_i \) are real valued within some prescribed intervals, i.e.,

\[ n_k = n_{k_0}, \quad k = 1, \ldots, m \]  

(1c)

Assumptions (1) and (2) are assumed to be satisfied.

At this point, the following assumptions are needed.

Assumption 1: Each element \( A_{ij} \) of \( A \) is assumed to be an odd function of \( \lambda_i \) and \( \lambda_j \), respectively.

Assumption 2: The system is assumed to be linearly independent.

The solution set of the problem (1a) is the set

\[ \beta = (\gamma_1, \gamma_2, \ldots, \gamma_n) \]  

(2)

The interval solutions of \( \beta \) will be denoted \( \beta^l \) and \( \beta^u \) will be denoted exact interval solutions to problem (1a,1b,2c). Any other interval \( \beta \) with \( \beta^l \leq \beta \leq \beta^u \) will be referred to as an interval solution to (1a,1b,2c). Finally, an interval solution \( \beta \) with the property \( \beta^l \leq \beta \leq \beta^u \) will be referred to as an exact solution to (1a,1b,2c).

The solution set of the problem (1a)-(2) is the set of all possible \( \beta^l \) and \( \beta^u \) for the T3 problem in (1a,1b,2c).

The problems formulation (2) and (1) is the same as the one in Section 3.1. However, the solution set of (2) is a subset of the solution set of (1).

III. OUTER SOLUTION

In this section, a method for determining the outer solutions of the T3 problem is described. The solution set of (2) is the set of all possible \( \beta^l \) and \( \beta^u \) for the T3 problem in (1a,1b,2c).

The solution set of (2) is a subset of the solution set of (1).

The solution set of (2) is the set of all possible \( \beta^l \) and \( \beta^u \) for the T3 problem in (1a,1b,2c).
where $A_{r} \in \mathbb{R}^{n \times n}$ and $(\mathbf{v}, \mathbf{w})$ is a real $n \times 1$ vector. The form (8) can be interpreted in an approximate way using the algorithm of [10]. It has the inclusion property

$$f(x) \in L_{f}(x), \quad x \in x.$$  \hspace{2cm} (9)

Secondly, consider the product

$$c_{x} \in \mathbb{R}^{n \times k} \quad x \in x,$$

where $a_{x} \in \mathbb{R}^{n \times m}$ and $b_{x} \in \mathbb{R}^{m \times k}$ are the respective matrices and nulls (9)

$$c_{x} \subseteq \mathbb{R}^{n \times k}$$

where $a_{x} \in \mathbb{R}^{n \times m}$ and $b_{x} \in \mathbb{R}^{m \times k}$ are the respective matrices and nulls (14)

$$c_{x} \subseteq \mathbb{R}^{n \times k}$$

In accordance with (8), the corresponding linear interval forms of (16) and (18) are

$$f_{a}(x) = \sum_{i=1}^{n} a_{i} x_{i} + a_{0}, \quad x \in x$$

where

$$f_{a}(x) = \sum_{i=1}^{n} a_{i} x_{i} + a_{0}, \quad x \in x$$

and have the inclusion property

$$f_{a}(x) \in L_{f}(x), \quad x \in x.$$  \hspace{2cm} (16)

The approach here adhered to determining an answer relative to (16) and (18) and the method for solving linear system equations in (17). For this purpose, (16)-(18) is written in the form

$$f(x, y) = A_{f}(x) - B_{f}(y) = 0, \quad x, y \in x.$$  \hspace{2cm} (14)

Let $\mathbf{v}$ be the interval vector whose components are defined through (18). Constating

$$f_{v}(x) = A_{v}(x) - B_{v}(y) = 0, \quad x \in x.$$  \hspace{2cm} (15)

where $A_{v}$ and $B_{v}$ are $n \times n$ and $n \times m$ matrices. As a special case, the inclusion property

$$f_{v}(x) \in L_{f}(x), \quad x \in x.$$  \hspace{2cm} (16)

Note: We shall obtain explicit expressions for $A_{f}$ and $B_{f}$ and $\mathbf{v}$ and $\mathbf{w}$ with (15). With this, we first introduce the interval matrices $f_{a}$, for the interval matrices $A_{a}$ and $B_{a}$, for the interval matrices $\mathbf{v}$ and $\mathbf{w}$. Note: for the interval matrices $A_{a}$ and $B_{a}$, for the interval matrices $\mathbf{v}$ and $\mathbf{w}$.

$$L_{f} + D_{f} + C_{f} \leq 0, \quad y \in y$$

where $L_{f} + D_{f} + C_{f} \leq 0, \quad y \in y$.

Now consider the linear interval system related to (18)

$$L_{f} + D_{f} + C_{f} \leq 0, \quad y \in y$$

where $L_{f} + D_{f} + C_{f} \leq 0, \quad y \in y$.

So far, we have assumed that $\mathbf{v}$ is a known interval vector. Now, we proceed to determining $\mathbf{v}$ as an exact solution to

$$L_{v} + D_{v} + C_{v} \leq 0, \quad y \in y.$$  \hspace{2cm} (17)

where $L_{v}$ and $D_{v}$ are defined by (15) and (18), respectively.
We have seen to determine the solution of $\mathbf{y}$. On account of (27a) (27b) and (22)

$$\mathbf{y} = \mathbf{y}_0 + \int_0^t \mathbf{f}(\mathbf{y}(\tau)) d\tau.$$  

Let by simplicity of notation

$$x = e^\mathbf{y}, \quad y = e^\mathbf{y} x$$  

Taking into account (27a), a reasonable choice for $\mathbf{y}$ is to determine $x$ as the solution $\mathbf{y}^*$ of the equation

$$\mathbf{y} = \mathbf{y}_0 + \int_0^t \mathbf{f}(\mathbf{y}(\tau)) d\tau.$$  

or equivalently

$$\int_0^t \mathbf{f}(\mathbf{y}(\tau)) d\tau = 0.$$  

The main result of this section is the following theorem.

**Theorem 1:** Assume the solution $\mathbf{y}^*$ to system (28) is positive. Then

1. the linear vector

$$\mathbf{x} = \mathbf{x}^* \mathbf{y}^*$$  

where

$$\mathbf{y}^* = [y_1, \ldots, y_n]^T$$  

is an interior solution to (14) (15).

We refer $\mathbf{y}^*$ as a monomial for many $p \in \mathbf{y}$.

The proof of the above theorem is based on the following theorem which states that if $\mathbf{y}$ is a real solution of (28), then $\mathbf{y}^*$ is also a solution of (28). Using (30), we can determine (28) as positive, then the solution is obtained from (28). If the other hand, system (28) does not have a positive solution, the method for not applicable.

If the $\mathbf{y}^*$ problem is to find an outer solution $\mathbf{y}$ (14) (15) and (22), $\mathbf{y}$ can be computed as follows:

$$\mathbf{y}_0 = \mathbf{y}(x)$$  

where $\mathbf{y}_0$ is the range of $\mathbf{y}$ in $\mathbf{y}$. The above method is obtained by applying the outer approximation to the exact solution of (28) problem considered to be referred to as method III.

**IV. OUTER SOLUTIONS**

In this section, a simple method for determining an outer solution $\mathbf{y}_0$ is proposed in (15) (16) and (22) will be presented. In fact, we construct really and all components $\mathbf{y}_0$ of $\mathbf{y}_0$. As in the previous sections, the method with the above approach to determining the components $\mathbf{x}_k$ of the outer solution $\mathbf{y}_0$ of problem (14) (15) and (22).

The solution is based on a local approximation technique of (28) and the following theorem which states that the outer problem for $\mathbf{y}_0$ is also a solution of (28). Using (31) and (32), we can determine (28) as positive, then the solution is obtained from (28). If the other hand, system (28) does not have a positive solution, the method for not applicable.

If the $\mathbf{y}_0$ problem is to find an outer solution $\mathbf{y}_0$ (14) (15) and (22), $\mathbf{y}_0$ can be computed as follows:

$$\mathbf{y}_0 = \mathbf{y}(x)$$  

where $\mathbf{y}_0$ is the range of $\mathbf{y}$ in $\mathbf{y}$. The above method is obtained by applying the outer approximation to the exact solution of (28) problem considered to be referred to as method III.
and $\delta^p$ is the solution of the real system

$$\begin{cases} \delta^p_i & = \delta_i, \quad i = 1, \ldots, m. \end{cases} \tag{17b}$$

Now, we determine new values $\delta^p_i$, $i = 1, \ldots, m$, using the following formulas

$$\delta^p_i = \begin{cases} \delta_i, & \text{if } 0 \geq \delta_i \geq 0, \quad i = 1, \ldots, m, \quad \text{(38)} \\
\delta_i, & \text{if } 0 < \delta_i, \quad i = 1, \ldots, m, \end{cases}$$

and from the vector $\delta^p = (\delta^p_1, \ldots, \delta^p_m)$. Then we solve the system

$$\mathbf{A}^p \mathbf{x} = \mathbf{b}^p$$

to find the vector $\mathbf{x}$. If

$$\mathbf{x}^* < \mathbf{x}^p,$$ \hspace{1cm} (60)

$\mathbf{x}^*$ is retained, $\delta^p$, and the procedure is repeated. Otherwise, the procedure is terminated and the final bound $\mathbf{x}^* \in \mathbf{x}^p$ is given by the corresponding components $\delta^p_i$.

A similar procedure is valid for determining the upper endpoint $\mathbf{x}^*$ of $\mathbf{x}^p$.

Procedure 2: For a fixed $i$, let $p = p^i$ and repeat the computations (26a) and (26b) of Procedure 1. Now subtract the known vector $\delta_i \delta_i$ with components $\delta_i$ in the constraint

$$\delta_i = \gamma^i, \quad \text{if } 0 \geq 0 \geq \gamma^i, \quad i = 1, \ldots, m. \tag{41}$$

Then, the corresponding system (25) is solved and a new vector $\delta^i$ is found, if

$$\delta^i > \delta^p_i$$ \hspace{1cm} (42)

$\delta^i$ is retained, $\delta^i$ and $\delta^p$ are retained, and the procedure is repeated from the step where the procedure is stopped and the new bound $\delta^i$ is given by the corresponding component $\delta^i$. It is seen that Procedures 1 and 2 implement a strategy which is based on a logical operation scheme. If the actual development of $\delta^p$ is similar to the procedure, the procedure will stop. In fact, the exact solution $\mathbf{x}^p$ is obtained in the general case. Procedures 1 and 2 will only provide upper bounds on $\mathbf{x}^p$.

V. Exact Solution

In this section, the exact solution is the Ta problem (1a)-(1d) and (2), or problem (3a)-(3d) and (4b) will be solved. For simplicity of the presentation, first, a method for computing the exact solution $\mathbf{x}^p$ of the simple Ta problem (1a)-(1d) will be presented. The method is applicable only if the distance $d(A_0, A_1)$ is guaranteed to be nonnegative. Therefore, the solution can be obtained in the following way.

Let $\mathbf{x}^p$ be the exact solution of (1a)-(1d) computed by the method presented in Section III. Initially, let $d(A_0, A_1)$ denote the distance of the remaining quantities for $p \in p$. Then, we can write $d(A_0, A_1)$ as follows:

$$d(A_0, A_1) = (A_0 - A_1)^2 \tag{44}$$

and obviously

$$\frac{\partial d}{\partial p}(\mathbf{x}) = (A_0 - A_1) \quad \text{in } p \in p. \tag{45}$$

Hence, the derivative considered is guaranteed to be nonnegative in $p \neq 0$. For the special case of linear functions $\delta_0(x)$ and $\delta_1(p)$, the equation $\delta_0(x)$ and $\delta_1(p)$ can be solved.

As in the previous section, $\mathbf{x}_0$ and $\mathbf{x}_1$ can be computed as at some iteration $p$ of the following system:

$$\mathbf{A}^p \mathbf{x} = \mathbf{b}^p, \quad p \in p. \tag{46}$$

To get a priori interval vector $\mathbf{x}^p$, system (46) will be solved by the method (III) from Section III. Finally, $\mathbf{x}_1$ is obtained as $\mathbf{x}^p$.

Using (46), we determine the estimates $D_0$. Now we make the following assumption:

Assumption 2: We assume that each interval $D_i$, $i = 1, \ldots, m$, satisfies the condition

$$D_i > 0$$ \hspace{1cm} (74)

or the condition

$$D_i < 0 \tag{77}$$

On account of inclusion (46) the fulfillment of Assumption 2 guarantees that $\delta_0$ is monotonous with respect to $\delta_1$ parameter $p$.

Now, we define two vectors $\mathbf{z}^p$ and $\mathbf{z}^q$ as follows

$$\mathbf{z}^p_i = \begin{cases} \delta_i, & \text{if } D_i > 0, \quad i = 1, \ldots, m, \quad \text{(46a)} \\
\delta_i, & \text{if } D_i < 0, \quad i = 1, \ldots, m. \quad \text{(46b)} \end{cases}$$

The exact solution $\mathbf{x}^p$ of systems (1a)-(1d) can be found using the following theorem.

Theorem 1: If Assumption 2 holds, then the $i$th component $\mathbf{x}^p_i$ of the exact solution $\mathbf{x}^p$ is determined as follows:

1) $\mathbf{x}^p_i$ is equal to the $i$th component of the solution of

$$\mathbf{A}^p \mathbf{x} = \mathbf{b}^p \tag{49}$$

where the vector $\mathbf{p}^a$ and $\mathbf{p}^b$ are determined according to (28a) and (28b).
The theorem follows directly from the above considerations about the eigenvalues of \( A(x) \).

On the basis of the above proof, we have the following procedure for determining any eigenvalue \( \lambda_i \) of the exact solution \( \mathbf{z} \). Before initiating the procedure, however, we solve system (48) to get as an initial solution \( \mathbf{z}_0 \).

Procedure 2: For a given \( k \), solve system (48) using method M. In this step, we find the interval vector \( \mathbf{z}_0 \). By (48), compute \( \mathbf{z}_0 \), \( i = 1, \ldots, n \). Check conditions (27). If all of them are satisfied, determine the initial interval vectors \( \mathbf{z}_0^{(l)} \) and \( \mathbf{z}_0^{(r)} \) along (48) (48b). Finally, solve system (48a) to get the lower and upper points \( z_i^{(l)} \) and the upper and lower points \( z_i^{(r)} \) of the interval components \( z_i \) of the exact solution to system (48a) (48b).

In some cases, it is possible to determine \( z_i^{(l)} \) and \( z_i^{(r)} \) from all conditions (43) (44) (47) (48) (48b) are satisfied, that is, if Procedure 3 is not applicable. Indeed, by (48), for a fixed \( k_0 \), the set of those indices \( I \) for which either (43a) (44a) holds while \( z_i^{(l)} \) denote the set of indices \( I \) for which (43b) (44b) is satisfied. Using (48a) (48b) in which \( k = k_0 \), we can determine those components of the vectors \( \mathbf{z}_0^{(l)} \) and \( \mathbf{z}_0^{(r)} \) which are guaranteed to take on real and real values. Thus, each vector can be partitioned into two parts as follows:

\[
\mathbf{z}_0 = \left[ \begin{array}{c} \mathbf{z}_0^{(l)} \\ \mathbf{z}_0^{(r)} \end{array} \right]
\]

where only the components \( z_i^{(l)} \), \( i \in I_2 \), are found to be on these values while the corresponding components \( z_i^{(r)} \), \( i \in I_1 \), form the interval vector \( \mathbf{z}_0^{(r)} \). Let the indices \( I_1 \) and \( I_2 \) be the \( m_1 \) and \( m_2 \), respectively, of the components \( \mathbf{z}_0^{(l)} \) and \( \mathbf{z}_0^{(r)} \) of the interval vector \( \mathbf{z}_0 \). Now, we can apply method M by using (48a) (48b) in which \( k = k_0 \), \( I = I_2 \), \( z_i^{(l)} = 0 \) and \( z_i^{(r)} = 0 \). We then proceed as in Procedure 3 to get \( \mathbf{z}_0^{(l)} \) and \( \mathbf{z}_0^{(r)} \).

Procedure 3: For a fixed \( k \), the next step is to solve the following modified system of the form (48a) (48b)

\[
\begin{align*}
A' (x) & \mathbf{z} = \mathbf{b}, \\
\mathbf{z} & = \left[ \begin{array}{c} \mathbf{z}^{(l)} \\ \mathbf{z}^{(r)} \end{array} \right], \\
\mathbf{b} & = \left[ \begin{array}{c} \mathbf{b}^{(l)} \\ \mathbf{b}^{(r)} \end{array} \right].
\end{align*}
\]

Also, find the corresponding interval solution \( \mathbf{z}_0^{(l)} \) of the modified system (54a) (54b) to get the corresponding interval vector \( \mathbf{z}_0^{(l)} \). Thus, we can compute by (48) the elements \( \mathbf{z}_0^{(l)} \), \( i \in I_2 \), of the corresponding components \( \mathbf{z}_0^{(l)} \) and \( \mathbf{z}_0^{(r)} \). Finally, \( \mathbf{z}_0^{(r)} \) is computed as the unique component of the solution of

\[
\begin{align*}
A' (x) & \mathbf{z} = \mathbf{b}, \\
\mathbf{z} & = \left[ \begin{array}{c} \mathbf{z}^{(l)} \\ \mathbf{z}^{(r)} \end{array} \right], \\
\mathbf{b} & = \left[ \begin{array}{c} \mathbf{b}^{(l)} \\ \mathbf{b}^{(r)} \end{array} \right].
\end{align*}
\]

We now present a procedure for determining the upper and lower points \( \mathbf{z}_0^{(l)} \) of \( \mathbf{z}_0^{(r)} \).

Procedure 4: Now we solve system (54c) (54d) in which \( \mathbf{z}^{(l)} \) is replaced with \( \mathbf{z}^{(r)} \). Thus, we compute the elements \( \mathbf{z}_0^{(l)} \), corresponding to \( \mathbf{z}_0^{(r)} \). Once again, we assume that the eigenvalue condition (43a) (44a) is satisfied. Using (48a) (48b) we find the exact vector \( \mathbf{z}_0^{(l)} \) whose components are \( z_i^{(l)} \), \( i \in I_2 \). Finally, \( z_i^{(r)} \) is computed as the unique component of the solution of

\[
\begin{align*}
A' (x) & \mathbf{z} = \mathbf{b}, \\
\mathbf{z} & = \left[ \begin{array}{c} \mathbf{z}^{(l)} \\ \mathbf{z}^{(r)} \end{array} \right], \\
\mathbf{b} & = \left[ \begin{array}{c} \mathbf{b}^{(l)} \\ \mathbf{b}^{(r)} \end{array} \right].
\end{align*}
\]

The above approach to computing \( \mathbf{z}_0^{(l)} \) will be referred to as method MD.

Method M can be used only if both Procedures 3 and 4 are applicable. However, the method is applicable if both methods are applicable and if the corresponding solutions of (48a) (48b) are not feasible for all components of \( \mathbf{z}_0^{(l)} \), a new approach can be made to determine \( \lambda_i \). With this in mind, we treat \( \mathbf{z}_0^{(l)} \) as a new interval parameter vector. We then proceed for the new interval parameter vector \( \mathbf{z}_0^{(l)} \), and apply once again method MD.

The new computational scheme will also be referred to as method MD.

Method M is also applicable in the general case of nonlinear equation (2). Thus, for instance, we have again to operate to formula (43c) when, however, all initial conditions are to be replaced with their initial counterparts. Alternatively, we employ Procedure 4 and 5, which have been modified accordingly.

VI. NUMERICAL EXAMPLE

The new methods will be illustrated with the following example. The linear AC circuit considered (1) is shown in Fig. 1.

![Fig. 1. Linear AC Circuit](image)

The elements (real part) values of the interval element parameters are

\[
\begin{align*}
R_1 & = 10.0 \Omega, \\
L_1 & = 0, \\
C_1 & = 10^{-3} F, \\
\mathbf{C} & = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},
\end{align*}
\]

while the initial condition is

\[
\mathbf{A} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\]

The problem is to solve circuit (1) using the modified numerical method. The numerical results obtained are shown in MATLAB. The obtained results are compared to the results obtained by the classical numerical method.
A. Open Solutions

Unlike the algebraic-iterative and complex calculation, the result obtained by the iterative formula depends on the system describing the problem at hand. Thus, the initial value of V will be the same for all the systems. Whether the systems of equations used can be used by the formula of simple equation form will be shown in the next section. However, it is known that the formula for solving a system of equations is employed. For the purpose of comparison these types of systems of equations will be considered.

Example 27. This is a system of equations of the type used in Chap. 3 (Chap. 3). It consists of 2x = 2x + 2y = 1000 equations where X and Y are the number of unknown variables and branch currents, respectively.

\[
A \Delta b = b
\]

and the iterative solution is only in the first 14 positions along the main diagonal of the coefficient matrix A. The initial conditions are obtained as shown:

\[
x_1 = x_{1,0}, x_2 = x_{2,0}, \ldots, x_n = x_{n,0}
\]

For the circuit shown by Fig. 34, where all \( x_i \) is an even integer and

\[
x_i = x_{i,0} + 1, 0 \leq i \leq 20
\]

where \( x_i \) is the unknown variable and \( A_i = [R_i, 1] \) is a symmetric interval

\[
y_i = X_i, 0 \leq i \leq 20
\]

with

\[
X_i = \frac{1}{2} [R_i + 1], R_i = a_i + b_i, 0 \leq i \leq 20
\]

The real part V of the output voltage is represented by the component x = 0 of the unit vector Y.

Example 28. This is the solved hybrid system of equations (11)-(15). It has the same structure as system 2, i.e., the real part of the complex variables are given by (18), but now system (55a) has been extended to involve \( n = 2n \) unknowns. For the circuit shown in Fig. 35,

\[
\begin{align*}
p_1 &= p_{1,0}, p_2 &= p_{2,0}, p_3 &= p_{3,0}, \ldots, p_n &= p_{n,0} \\
p_{n+1} &= p_{n+1,0}, p_{n+2} &= p_{n+2,0}, \ldots, p_{2n} &= p_{2n,0}
\end{align*}
\]

and \( Q_i = [R_i, B_i] \). Now, V is given by \( x_0 \).

System 32. In this case, system (55a) is set up using modal analysis and involves \( n = 2n \) equations. Now

\[
\begin{align*}
q_1 &= q_{1,0}, q_2 &= q_{2,0}, \ldots, q_n &= q_{n,0} \\
q_{n+1} &= q_{n+1,0}, q_{n+2} &= q_{n+2,0}, \ldots, q_{2n} &= q_{2n,0}
\end{align*}
\]

where \( q_{n+1} = 0, 0 \leq i \leq n \). For the circuit investigated

\[
q_i = Q_i, 0 \leq i \leq 2n
\]

and

\[
q_{n+1} = 0, 0 \leq i \leq 2n
\]

The output variable V is given by \( x_0 \).

We first compute the results for the system of equations obtained by the present method and also by the present method and the result for the two methods of branch currents. Although a better version of Huer’s method has been used in this case, the authors [12] and [13],

\[
\text{Table 4}
\]

\[
\text{The table of data for system 32 and 33}
\]

B. Some Solutions

The same solution V for Y is obtained by Procedure 1 and Procedure 2. Table 4 presents data for the values of V, the total number of iterations, N, and the number of steps corresponding to the number of iterations of the number of equations used in the solution. The system of equations used in the solution is practically the same:

\[
\text{Table 5}
\]

\[
\text{The comparison of the results obtained by the two methods}
\]

Table 4 shows that the present method is superior to the method by Huer’s method. Indeed, the results of the computer V obtained by the present method are shown as compared to those corresponding to Huer’s method. These results confirm the theoretical prediction and also the computer V obtained by the present method. The present method requires the computer in the same computing time V.

In Table 5, we present data on the maxima solution V for two values of Y obtained by the present method of 32 and 33. It is seen that for both values of Y the computer in the same computing time. However, the computer V obtained by the present method is superior to the computer V obtained by Huer’s method. Indeed, the results of the computer V obtained by the present method are shown as compared to those corresponding to Huer’s method. These results confirm the theoretical prediction and also the computer V obtained by the present method. The present method requires the computer in the same computing time V.
The inner and outer solutions obtained by methods M2 and M1, respectively, provide right two-valued branches on each real point $p^*$ and $p^*$ of the exact solution $V^*$. Table V lists data for three branches corresponding to $l = 0.5$, $l = 1$ and $l = 1.5$ obtained by methods M1 and M2 on the inner and outer branches Monte Carlo method (MC) and Hunter's method (H), respectively.

C. Exact Solution

Using method M3 (see Section IV, 3), the exact solution $V^*$ has been obtained for all systems of equations when $l = 0.5$, $l = 1$ and $l = 1.5$. Table VI lists results for the approximations $V^*$, the total values of $p^*$ and $p^*$ corresponding to the summation of both $p^*$ and $p^*$ and the required computing time $t$.

It is observed that $t = 0$, i.e., the method provided the best result for the entire range of $l$ considered, including $l = 1$. Method M3 is comparable with method M2 in the sense that for $l = 0.5$, none of the monotonicity conditions (4) and (5) is violated. However, if $l = 1$, the following result has been obtained (Table VII).

In this case, method M1 provides the best solution only for the exact inner solution $V^*$. It cannot converge to the upper and lower parts of $V^*$, since after 4 iterations conditions (4) and (5) are not satisfied for $l = 1$ (Table VIII). However, in general terms, the method provides a good solution based on $V^*$.

D. Analysis of the Numerical Results

The comparison analysis of the numerical results obtained can be summarized as follows. The convergence of the outer solution depends on the type of approximations used. According to this criterion, the best results were obtained when system $S_2$ was employed. This is supported by a valid conclusion for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$


TABLE VII

<table>
<thead>
<tr>
<th>System</th>
<th>$y$</th>
<th>$x$</th>
<th>$S_1$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>0.5</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE VIII

<table>
<thead>
<tr>
<th>System</th>
<th>$y$</th>
<th>$x$</th>
<th>$S_1$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>0.5</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$. However, the method provides a good solution based on $V^*$. The table also shows that the exact solution is obtained for low- and moderate-sized circuits. However, for large $l$-TA problems, the exact solution $V^*$ will be preferable. Further numerical experiments are needed to make a decision on this point. The exact solution is obtained with least computational effort when system $S_2$ is used. It should, however, be verified if this conclusion remains valid for large-scale circuits.

It should also be noted that the use of speed-up circuits, the overall efficiency of the present approach seems to be superior to that of the Monte Carlo method. It is suggested that the method be extended to other systems. It is not essential that all experiments be done on the inner solution as in fact equal to the exact solution, i.e., $y = x$.
Modeling of Nonlinear Electrical Networks: A Computation Algorithm Approach
A. Kajzl, K. Schächer, H. Grabner

An Improved Algorithm for the Analysis of MOS Transistor Circuits Having Multiple DC Solutions
M. Tchudakowicz, S. Hafner

Matrix Representation and Parametrizations of Lossless and Reciprocal Multipoles
H. Lenard, P. Pauly

Solving of Equations of Nonlinear Circuits and Control Systems Using Differential Transformations
E.D. Grodon, O.V. Stokach

Communications

OTDM: An Old Idea Solved New Problems
K. D. Karrer, M. Szewcz, R. Kiehl, B. Fichtel

Combining a Time-Domain Pilot Estimation Scheme with an Adaptive Antenna Array for an OTDM System
R. Suzuki

Combined Linear and Nonlinear Multi-User Detection for Coded OTDM-COMA
V. Khan

Single Carrier Transmission with Frequency Domain Equalization: An Overview
A. Kapul, H. Wilscring, M. Hemm, R. Hoppe

Spectral and Systems

A Wave Theory of Long LMS Adaptive Filters
R.J. Scholtens

Advanced Spectrum Estimation Methods for Signal Analysis in Electric Engineering
T. Lobos, Z. Czerniak, J. Mazri, H.-J. Kapp

Velocity Estimation from Radar Bearing Measurements
M. Pfeifer, A. Steiner, H. Hoepfner, B. Voigt

A General Clustering Algorithm Through Linear and Non-Linear Methods
J. Shep, G. Johansson, D.O. Rode

Solving Linear Transport Problems by Using Adequately Modified Hessbly Neural Network
I. Kratoch, B. Ronan, D. Karlo

Communications

SMR optimization for the TD-COMA downlink with COMA code pooling and adaptive transmit antennas
P.K. Basu, Y. Lu, M. Weiser

Signal Processing for Transmitter Position Estimation in SIMA Systems
V. Poch, V. Chichkazhov, P. Russian

Witt Functions for Signal Processing and Code Division Multiplexing
P. Pfitzner

Computer Modeling of Signal Interaction in Optical Communication Systems
S. Zeigner, L. Sumehroz, D. Welnke

New Implementation of network elements for quantum information processing with linear ion traps
K. Panihe, W. Walms
A general interval method for tolerance analysis

Lubomir Kolev, Ivan Nenov

1. INTRODUCTION

INTRODUCTION methods for solving various types of circuit tolerance analysis problems have been in evidence for over twenty years [15]. The major part of this investigation is related to the interval method problem for linear circuits. The linear tolerance problem in multiparametric setting is considered in [7, 11, 13, 15, 17] and addresses the multiparametric tolerance for nonlinear circuits. The proposed methods, however, differ substantially from one another depending on the class of circuits analyzed (linear or nonlinear), type of problem to be solved (deterministic or probabilistic settings) and the type of problem formulation used (as a global constrained optimization problem or in the form of an interval linear or nonlinear systems of equations).

In this paper, a general interval method for solving any of the known tolerance analysis problems for both linear and nonlinear circuits is suggested. It is based on a modification and generalization of a method proposed by the authors in a previous paper. A tolerance problem was analyzed, now the system of equations describing the tolerance problem considered is in rather a general form

\[ f(x, p) = 0 \]  \tag{1a}

where \( f \) is a \( n \)-dimensional vector function, \( x \) is a \( m \)-dimensional input parameter vector, \( p \) is a \( n \)-dimensional parameter vector and \( x \) is the corresponding interval vector (box). It is assumed that a pair \( (x, p) \) corresponding to the nominal solution is known such that

\[ f(x, p) = 0 \]  \tag{1b}

The solution on \( F_x(p) \) of (1) is the set

\[ S_x(p) = \{x, f(x, p) = 0, p \in P \} \]  \tag{2a}

where \( S_x(p) \) will be denoted \( S_x^0 \); any other interval set such as \( S_x^{\pm} \) will be referred to as an interval (vector) based on \( S_x(p) \). In the present paper, the tolerance problem considered is reduced to finding a bound on \( S_x(p) \). A method for computing a is suggested which is based on an alternative linear interval enclosure of non-linear functions in a given box [10, 12, 13].

To simplify presentation, it is hereafter assumed that \( n = k \), \( k \) is a number of equations equals number of input variation.

2. PROBLEM DEFINITION

It is known that a continuous function \( g(x_1, \ldots, x_n) \) can be enclosed in a box by the following affine linear interval function

\[ L_x(t) = \sum_{i=1}^{n} a_i x_i + b \]  \tag{3}

where \( a_i \) are real numbers and \( b \) is an interval (vector) having the property

\[ g(x) \in L_x(t), x \in \mathbb{R}^n \]  \tag{4a}

Special form of linear function \( L_x(t) \) are valid when \( g \) is an \( n \)-dimensional function. Now

\[ L_x(t) = At + b, x \in \mathbb{R}^n \]  \tag{5}

where \( a \) is a real matrix and \( b \) is an interval vector and for the new notation, property (6) is also valid. Computational procedures for determining \( g \) and \( b \) are suggested in [10].

On analysis of (3), the linear interval enclosure of \( L_x(t) \) in the box \( x = [x_p, \ldots, x_p] \) will be

\[ L_x(t) = At + b, x \in [x_p, \ldots, x_p] \]  \tag{6}
In this section, a method for determining an upper bound \( r \) on the solution set \( S_r(p) \) of (1) is presented. It consists of two stages: during the first stage, a "good" starting box \( A_0 \) is determined, the second stage aims at improving \( r \) by making it narrower.

**Stage 1:** Let \( p \) be the center of \( p \). First, the maximal rotation \( A = A(p) \) is formed by solving \( f(x, p) = 0 \). Next, a narrow box \( A_0 \) of width \( v_0 \) centered at \( A \) is introduced and, simultaneously, system (1) is relaxed to \( A \) by the linear interval form:

\[
\Delta_y(x) = A^T \Delta_y + A^T b = \Delta_y \quad \text{for} \quad x = A^T x_0 + b
\]

where \( \Delta_y = A^T p + b \).

The first stage can be implemented in two different ways using the following two procedures.

**Procedure 1:** It is initiated by setting \( x = x \) and going back to (7).

**Procedure 2:** It starts as Procedure 1 by computing \( x \) using (8). In this part, \( x \) is centered at \( A \) and the new \( x \) is formed by the rotation:

\[
x_r = A^T x + b
\]

Next, the new interval (as in the previous procedure) from (7) with \( x = x_r \).

It is assumed that Procedure 1 (Procedure 2) converges to a symmetric interval vector (box)

\[
x = \overline{a} - \underline{a}
\]

where \( x = x \) and \( x = \Delta_y \).

**Stage 2:** After the box \( A_0 \) has been determined by (9) we present to the second stage of the present method. Now we try to reduce \( r \) using the following procedure.

**Procedure 3:** Let \( x = x \) and construct the corresponding bound approximation of \( f(x, p) = 0 \), using (8) and (9). Thus, the corresponding box \( \Delta_y \) is made by the intersection \( x = A^T x + b \).

As before, the iterative process is started by picking \( A = A_0 = \overline{a} - \underline{a} \) and going back to (7). It is terminated when the distance between two successive boxes becomes smaller than an accuracy \( \varepsilon \). The corresponding accuracy box contains the values of the corresponding tolerance parameters described by (1).

**The distance used in the stopping criterion in Procedure 1 (1) has been chosen as**

\[
d = \max \left( \frac{1}{\| A \|}, \left| \frac{\Delta_y}{\| A \|} \right| \right)
\]

where \( \varepsilon \) stands for the width.

The second stage of the present method poses a computationally more complex task. Indeed, for \( r = \overline{a} - \underline{a} \) the best obtained at the 1-1 iteration of Procedure 3 if the condition

\[
\| A \| (\Delta_y) \leq 10^{-6}(\Delta_y)
\]

is satisfied (or its dual is satisfied) for some \( k = 1, \ldots, 26 \), then

\[
b_k = \left( I_r \right) \Delta_y
\]

where \( I_r \) is the identity matrix.

**Numerical Experiments**

In this section, two numerical experiments illustrating the applicability and efficiency of the present method are given. The algorithms have been tested using algorithms A1 and A2. The algorithms were programmed using the Algol-60 language and run on a mainframe computer. The linear interval procedure of the present method is implemented by a subroutine that performs the extensive search in [9].

**Example 1:** The system of equations:

\[
x_1 + 3x_2 + 4x_3 = 5
\]

\[
x_1 - x_2 + 2x_3 = 1
\]

was solved by the method described in this paper. The solution of the corresponding tolerance problem associated with the interval (4) is calculated with an accuracy of 0.001. This example is computational and may be interesting for the pure mathematician as well as for the applied mathematician who wishes to see that the present method is efficient.
is $x = (x_1, x_2, x_3)$ and the parameter vector is $p = (p_1, p_2, p_3)$. We chose $\theta$ as the sum of $\alpha$ given in (13b). The corresponding nominal solution $\bar{x}$ was found with accuracy $\varepsilon = 10^{-6}$ using a non-linear equation solver implementing the dogleg method (1) and (13).

$$\bar{x} = (0.556, -2.354, -0.468)$$  \hspace{1cm} (14)

This result in (14) was, however, given only up to four decimal places.

Application of algorithms A1 and A2 with $\alpha_0 = 0.5$ yielded the following results, respectively, for the interpolation of the tolerance problem considered:

$$x = (0.556, 0.5682), \quad (0.5635, -3.3553)$$  \hspace{1cm} (15)

$$x = (0.2482, 0.5685, -3.8910, -2.1154).$$  \hspace{1cm} (16)

For algorithm A1, $\alpha = 0.5$ and the difference of (15) was observed at $u = 1.1$ of Function 3. For algorithm A2 we chose $\alpha_0 = 0.5$ and between $x_2 = 0$ interval was calculated at $x_2 = 2$. Thus, both branches (15) and (16) are supposed to contain the solution set of (13), where

$$x = (x_1, x_2, x_3, x_4, x_5)$$  \hspace{1cm} (17a)

and a simple characteristic

$$x = y_0 = \alpha_0 - 1$$  \hspace{1cm} (17b)

with

$$x = y_0 = x_0 = y_0$$  \hspace{1cm} (17c)

All the remaining 5 parameter values $x_1, x_2, x_3, x_4, x_5$ determine the remaining 5 parameters within the interval [28].

The model of the tolerance problem (13) consists of the two equations (13a) and (13b) and the node variables $y_1, y_2, \ldots, y_6$. Again, we receive the well-defined tolerance analysis problem associated with this modified context.

A numerical difficulty arises in finding the nominal solution $\bar{x}$ for the example. In the well-known numerical problems caused by the non-uniform node variables, it is important to avoid such problems by selecting values of $y_1$ of 0.5, 0.5, 0.5, 0.5, 0.5, 0.5. Then, for the new nominal solution $\bar{x}$. The solution $\bar{x}$ was calculated. The values $\bar{x}$ were chosen so that $x_2$ is equal to $x_1$ and $x_1$ and node variables (13) have values between $\bar{x}$ and $\bar{x}$ A unique solution is.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>Stage 1</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

The numerical accuracy corresponding to the two algorithms are shown in Table 1. It is seen that algorithm A2 requires less iterations as compared to algorithm A1.

The same example was solved in [5] by an algorithm similar in structure to algorithm A2; however, as mentioned in the previous section, each branch of both the first and second stage of the algorithm is associated with the solution of a corresponding linear interval system and requires more time than algorithm A2. The following result was obtained:

$$x = (0.2582, 0.3376, 0.3237, 0.3237)$$  \hspace{1cm} (17d)

which is to be expected that the result (17) is more conservative as compared to (15) and (16) and at the same time takes more iterations to be reached. The number of iterations on the first stage and 2 iterations on the second stage.

Example 2. This example is a modification of Example 2.2 in [5]. The lower side of the obtained tolerance problem is shown in Fig. 3.2. (17) Literature elements having respectively 9-tuple characteristics.

$$x = 10^{0.5}, \ 2.5 \times 10^{0.5}, \ 11.8$$  \hspace{1cm} (18a)
IV. CONCLUSION

An iterative method for solving various classes of nonlinear analysis problems (linear and nonlinear, deterministic and stochastic) has been suggested. The method is iterative in nature and requires the original nonlinear problem to be rewritten as a set of linear interval equations. It is based on a modified iterative linear interval equations (2). If the unprocessed nonlinear equations (2a) is interpreted, the method guarantees that the obtained solution is a real value in some interval, i.e., the inclusion (12b) is fulfilled.

A computer program implementing the method has been developed in a C environment. The numerical results obtained so far (including those not reported here) are rather encouraging.

REFERENCES

Cheap and Tight Bounds on the Solution Set of Perturbed Systems of Nonlinear Equations

LUBOMIR V. KOLFV and IVO P. NENOV
Facility of Automatics, Technical University of Sofia, 178 Sofia, Bulgaria.
E-mail: kolfov@infranet.bg, ivo@infranet.bg

(Received: 9 July 2001; accepted: 28 November 2001)

Abstract. This paper presents an iterative method for computing an outer interval bound on the solution set of parameter-dependent systems of nonlinear equations for the case where the parameters take on their values within preset intervals. The method is based on a recently suggested alternative linear interval enclosure of factorized nonlinear functions in a given box. It comprises two stages: during the first stage, a relatively narrow starting box is determined using an appropriate inflation technique while the second stage aims at reducing the width of the starting box.

Two algorithms implementing the method have been programmed in a C++ environment. Numerical examples serve to indicate that the second algorithm is rather efficient computation-wise.

The method is well-illustrated: the fulfillment of a simple inclusion test checked during the second stage ensures that the interval bound thus found is an outer approximation to the solution set of the perturbed system investigated.

1. Introduction

The paper addresses the well-known problem of bounding the solution set of perturbed (i.e. parameter-dependent) systems of nonlinear equations (e.g. [1]-[3], [10]-[12]). More specifically, let the system considered be

\[ f(x, p) = 0, \]
\[ p \in \mathbb{P}, \]

where \( f : U \subseteq \mathbb{R}^n \times \mathbb{R}^m \), \( D \subseteq \mathbb{R}^m \), and \( E \subseteq \mathbb{R}^m \) are closed and connected sets with \( D \times E \subseteq U \), and \( p \) is an \( m \)-dimensional interval vector in \( E \). (For simplicity of notation, following [4], [18], throughout the paper interval quantities will be denoted by bold face letters while ordinary font letters will stand for real non-interval quantities.)

It is assumed that a pair \((x^0, p^0) \in U\) is known such that \( f(x^0, p^0) = 0 \) with \( p^0 \in \mathbb{P} \); \( p^0 \) is usually the center of \( p \). The solution set \( S_f(p) \) of (1.1) is the set

\[ S_f(p) := \{ x : f(x, p) = 0, p \in \mathbb{P} \}. \]

The interval hull of \( S_f(p) \) will be denoted \( x^* \); any other interval \( x \) satisfying \( x^* \subseteq x \) will be referred to as an interval (outer) bound on \( S_f(p) \). The width of \( x \) (or \( x^* \)) serves as a measure for the sensitivity of the solution \( x(p) \) when \( p \) varies around \( p^0 \) in \( p \).
A method for determining $\mathbf{x}^*$ is suggested in [3]. It reduces to globally solving $2n$ constrained optimization problems. As it is rather time-consuming, its applicability is limited to systems of low size $n$. Most often, a tight interval bound $\mathbf{x}$ is sought (e.g. [10]-[12]). In [23], [3], [10]-[12] use is made of either an interval extension of the Jacobian to compute $\mathbf{v}$ or $\mathbf{x}$ ($\mathbf{x}^* = \mathbf{x}$) or an interval slope matrix.

In the present paper, we suggest a new approach to tackling the problem of finding a bound $\mathbf{x}$ on $S$. It is based on an alternative linear interval enclosure of factorable non-linear functions in a given box [5]-[7].

The paper is organized as follows. Section 2 presents the basic approach adopted and the main results thereby obtained. The new method for computing the bound $\mathbf{x}$ is presented in Section 3. Two numerical examples illustrating the applicability of the method suggested are given in Section 4. The paper ends up with final remarks in Section 5.

2. Main Results

Let $g : \mathbf{z} \in \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous factorable function. It is known [7] that $g$ can be enclosed by the following affine linear interval function

$$L_g(z) = \sum_{j=1}^{n} a_j z_j + b$$

(2.1)

where $a_j$ are real numbers and $b$ is an interval having the property

$$g(z) \in L_g(z), \quad z \in \mathbf{z}.$$  

(2.2)

Similar formulae are valid in the case where $g : \mathbf{z} \in \mathbb{R}^n \rightarrow \mathbb{R}^m$. Now

$$L_g(z) = Az + b, \quad z \in \mathbf{z}.$$  

(2.3)

where $A$ is a real matrix and $b$ is an interval vector; for the new notation, property (2.2) is also valid. Constructive procedures for determining $A$ and $b$ are presented in [3]-[7].

Referring back to system (1.1), let $\mathbf{x}$ be a box large enough to contain $S(p)$, $\mathbf{x}^*$ and the bound $\mathbf{x}$ associated with a given $p$. The main result of the section is formulated in the following theorem.

**THEOREM 2.1.** Let $\mathbf{x}^* \subset \mathbf{x}$ and $\mathbf{z} = \mathbf{x}^*$. Furthermore, let

$$L_z(x, p) = A^T x + A^T p + b, \quad x \in \mathbf{x}^*, \quad p \in \mathbf{p}.$$  

(2.4)

be the linear interval enclosure of (1.1) in $z = [x, p]$. Also, let $S_z(p)$ denote the solution set of the linear interval system $A^T x + A^T p + b = 0, \quad p \in \mathbf{p}$.  

(2.5)
Then

\[ S_f(p) \subset S_{\bar{f}}(p). \quad (2.6) \]

Proof. Denote (2.4) equivalently as

\[ L_z(x) = Ax + b, \quad z = \lambda. \quad (2.7) \]

where \( z = (x, p). \) On account of the inclusion property (2.2)

\[ f(z) = Az + b, \quad \forall z \in \lambda. \quad (2.8) \]

If \( y \in \lambda \) is a zero of (1.1), then \( f(y) = 0. \) Hence from (2.8)

\[ 0 = Ay + b. \quad (2.9) \]

Let \( b = [\bar{b}, \bar{b}] \). The inclusion (2.9) can be written as

\[ 0 \leq Ay + \bar{b} \quad (2.10a) \]

and

\[ 0 \geq Ay + \bar{b} \quad (2.10b) \]

or equivalently

\[ 0 = Ay + b, \quad b \in \bar{b}. \quad (2.11) \]

Returning back to the components \( A^* \) and \( A^p \) of \( A \)

\[ A^*x + A^p + b = 0, \quad p \in \bar{p}, \quad b \in \bar{b}. \quad (2.12) \]

So, if \( x \in S_f(p) \), then there exists a pair \( (x, p) \) satisfying (2.12). But (2.12) defines the solution set \( S_f(p) \) of (2.5). Hence \( x \in S_f(p) \) implies \( x \in S_{\bar{f}}(p) \) which completes the proof.

COROLLARY 2.1. The solution set \( S_f(p) \) of (1.1) is also contained in the intersection

\[ S_f(p) = S_f(p) \cap x^* . \quad (2.13) \]

Let \( h_i \) denote the interval half of \( S_f(p) \). Then \( h_i \) is a bound on the solution set \( S_f(p) \).

We can find a slightly wider bound than \( h_i \) in the following way. Rewrite (2.12) in the form

\[ A^*x + b^p = 0, \quad b^p = b^p. \quad (2.14a) \]
where

\[ b' = A'p + b. \quad (2.14b) \]

Using the same argument as in Theorem 2.1 and Corollary 2.1, we have the following results.

**Theorem 2.2.** Let \( x^* \subseteq x' \subseteq x'' \) and let \( S_1(b') \) denote the solution set of (2.14). Then

\[ S_1(p) \subseteq S_1(b'). \quad (2.15) \]

**Corollary 2.2.** The solution set \( S_1(p) \) of (1.1) is also contained in the intersection

\[ S_1(p) = S_1(b') \cap x'. \quad (2.16) \]

Let \( h_2 \) be the interval hull of \( S_{2'} \); then \( h_2 \) is another bound on the solution set of (1.1). It follows from elementary set-inclusion considerations that

\[ h_2 \geq h_1. \quad (2.17) \]

It is easily seen that \( h_1 \) or \( h_2 \) can be determined by solving 3n linear programming problems associated with (2.12) or (2.14), respectively. Such an approach, however, appears to be rather costly for larger \( n \). Therefore, a slightly wider but by far less expensive bound \( h_3 \) will be suggested now. It is based on the following theorem.

**Theorem 2.3.** Let \( x^* \subseteq x' \subseteq x'' \) and

\[ h_3 = -(A' - 1)^{-1}b'. \quad (2.18) \]

Then

\[ S_1(p) \subseteq h_3 \quad (2.19) \]

and

\[ h_1 \leq h_2 \leq h_3. \quad (2.20) \]

**Corollary 2.3.** The solution set \( S_1(p) \) of (1.1) is also contained in the intersection \( h_3 \cap x' \).

The proof of the above theorem and corollary follows directly from Theorem 2.2 and Corollary 2.2.

Unlike \( h_1 \) and \( h_2 \), the bound \( x = h_3 \) is determined in a comparatively much cheaper manner by just one single inversion of the real matrix \( A' \) and a subsequent multiplication by an interval vector.
Remark 2.1. The bound $h_1$ can be improved if rather than using (2.18) $h_1$ is computed as follows

$$h_1 = -Cb - [CA^t]p,$$

(2.21)

where $C$ is the inverse of $A^t$. The validity of (2.21) follows from (2.12) if (2.12) is premultiplied by matrix $C$. Moreover, it is easily seen that

$$h_1 \leq h_1.$$

(2.22)

Indeed, from (2.14) and (2.18)

$$h_1 = -Cb - C[A^t]p.$$

(2.23)

Comparison of (2.21) with (2.23) and application of the subadditivity property lead to (2.22). It should however be borne in mind that formulae (2.18), (2.14b) require a lesser volume of computation than (2.21) and may turn out to be a better choice for large-size problems.

Henceforth, to simplify presentation, only the cruder bound $h_1$ will be used.

3. The New Method

In this section, we present a method for determining a bound $x$ on the solution set $S_x(p)$ of (1.1). It consists of two stages: during the first stage, a "good" starting box $x^0$ is determined; the second stage is based on Theorem 2.3 and aims at improving $x^0$ by making it narrower.

From a computational efficiency point of view, the selection of a good starting box for the second stage of the present method is of great importance. Indeed, if $x^0$ is chosen too large, the second stage will take too many iterations to converge; conversely, if $x^0$ is not large enough, it might not contain the outer solution $x = x_1$ as required by Theorem 2.3.

We start by presenting the first stage of the new method. This stage can be implemented in two different ways using the following two procedures.

PROCEDURE 3.1. We choose $pl = p^0$ (cf. the centre of $p$) and determine $x^0$ as the corresponding solution of (1.1).

Now a narrow box $x^0$ of small width $\varepsilon_0$ centered at $p^0$ is introduced and (1.1) is enclosed by the linear interval form (2.1) in $x^0 = (x^0, p)$. I.e. we determine

$$L(x^0, p) = [A^0 x + A^0 p + b_0],$$

$$x \in x^0, \quad p \in p.$$

(3.1a)

It is to be stressed that (3.1) is an enclosure of (1.1) only in $x^0$. However, (3.1a) will be used as a linear approximation of (1.1) in a larger box $x^1 = (x^1, p)$. The component $x^1$ of $x^1$ is determined in the following way. First, based on Theorem 2.3 we compute
\[ x^t = -(A^t)^{-1}b^t. \] (3.2a)

where
\[ b^t = A^t_i p + b_i. \] (3.2b)

Now the iterative procedure is started by putting \( x^0 = x^1 \) and going back to (3.1).

**PROCEDURE 3.2.** It is similar in structure to the previous procedure. The only difference lies in the way the component \( x^t \) is determined at each iteration. We start as in Procedure 3.1 by computing \( x^t \) using (3.2). At this point \( x^t \) is renamed \( x^t' \) and the new \( x^t \) is found by taking the union
\[ x^{t+1} = x^t' \cap x^t. \] (3.2c)

Next we let \( x^0 = x^1 \) and the iterations continue from (3.1) as in the previous procedure.

At this point, we need the following assumption.

**ASSUMPTION 3.1.** For a given box \( p \) Procedure 3.1 (Procedure 3.2) is convergent to a stationary interval vector \( x^t \) having the property
\[ x^t \subseteq x^t'. \] (3.3)

This assumption seems to be fulfilled most often in practice for relatively small boxes \( p \) and under reasonable requirements (such as given, e.g., in [33], [30], and [12]) on the non-linear function \( f \) in (1.1). The inclusion (3.3) is expected because of the fact that at each iteration \( k \) before convergence the current approximation \( L_k(x^t, p) \) of \( f(x, p) \) becomes better and the box \( x^k \) becomes larger than \( L_k(x^{t-1}, p) \) and \( x^{t-1} \), respectively.

In practice, Procedure 3.1 (Procedure 3.2) is terminated whenever the distance between two successive iterations \( x^{t+1} \) and \( x^{t-1} \) becomes smaller than an accuracy \( e_t \). This approximate stationary box denoted as \( x^t \) may be smaller than the stationary box \( x^t' \). To facilitate inclusion (3.3), we inflate \( x^t \), i.e., we let
\[ x^t = x^t + (1 + e_t)(-R, R), \] (3.4)

where \( R \) is the radius of \( x^t \) and \( e_t \geq 0 \).

After the box \( x^t \) has been determined by (3.4) we proceed to the second stage of the present method. Now we try to reduce \( x^t \) using the following procedure.

**PROCEDURE 3.3.** We let \( x^0 = x^t \) and construct the corresponding linear approximation of \( f(x, p) \) in \( (x^t, p) \) using (3.1). By (3.2a) and (3.2b) we find the corresponding box \( x^t' \). Next, a new box \( x^t'' \) is introduced by the intersection
\[ x^{t+1} = x^t' \cap x^t. \] (3.5)

Now we let \( x^{t+1} = x^t \) and the iterative process continues from (3.1). It terminates when the distance between two successive boxes becomes smaller than an accuracy \( e_t \).
The distance used in the stopping criterion for Procedures 3.1 to 3.3 is computed as the maximum among the absolute values of the differences between the widths of the corresponding components.

It is seen that the method suggested above can be implemented as:

a) Algorithm A1 which is based on Procedures 3.1 and 3.3;

b) Algorithm A2 which uses Procedures 3.2 and 3.3.

Experimental evidence seems to indicate that Algorithm A2 requires less iterations than Algorithm A1 to solve the perturbed problem considered.

The second stage of the present method permits to computationally test the validity of inclusion (3.3) in Assumption 3.1. More precisely, we have the following result.

THEOREM 3.1. Let \( \mathbf{x}^* \) be determined by Procedure 3.1 or Procedure 3.2 using (3.4). Let \( \mathcal{K}^{(i)} \) be the box obtained at the k-th iteration of Procedure 3.3 with \( \mathbf{x}^0 = \mathbf{x}^* \). If the condition

\[
\mathcal{K}^{(i)} \subseteq \text{int}(\mathcal{K}^*) \tag{3.6}
\]

is fulfilled for some \( k \geq 1 \), then the second stage of the method validates the inclusion (3.3).

Proof. On account of Corollary 2.3 the solution set \( S(p) \) as well as its interval hull \( \mathcal{K}^* \) cannot have points lying outside the intersection \( \mathcal{K} \cap \mathcal{K}^* \). Thus, \( \mathcal{K}^* \) cannot have points outside \( \mathcal{K}^{(i)} \cap \mathcal{K}^* \). Now assume that (3.6) holds for some \( k \). In this case, \( \mathcal{K}^{(i)} \) lies strictly within \( \mathcal{K}^* \) and is therefore enclosed by a "ring" (formed by the difference \( \mathcal{K}^* \setminus \mathcal{K}^{(i)} \)) which does not contain points belonging to \( \mathcal{K}^* \). On the other hand, \( \mathcal{K}^{(i)} \) is bounded by \( \mathcal{K}^* \) by construction, i.e.

\[
\mathcal{K}^* = \mathcal{K}^{(i)} \tag{3.7}
\]

Finally, on account of (3.7), the validity of (3.6) implies the inclusion (3.3)

\[
\mathcal{K}^* \subseteq \mathcal{K}^{(i)}
\]

which concludes the proof.

Remark 3.1. We can reduce the overestimation of the bound \( \mathbf{x}^* \) obtained by the present method appealing to the well-known technique of partitioning the parameter box \( p \) into a given number \( N \) of subboxes \( p^{(i)} \). We then apply the method to each subbox \( p^{(i)} \) to get a corresponding bound \( \mathcal{K}^{(i)} \). The box \( \mathcal{K}^* \) bounding the solution set of the original problem is now obtained as the interval hull of the union of all boxes \( \mathcal{K}^{(i)} \).

Obviously, such an approach is only applicable to problems where the dimension \( n \) of the parameter vector \( p \) is small.

4. Numerical Examples

In this section we give two examples illustrating the applicability of the method suggested. The examples have been solved by both Algorithms A1 and A2 with
$a_i = c_i = c_1 = 10^{-4}$. The algorithms were programmed using the algorithmic language C++. The linear interval enclosures (2.3) were generated automatically by a procedure that implements the approach suggested in [71].

**EXAMPLE 4.1.** The system of equations is

\[
\begin{align*}
(e - x_1) / p_1 - x_1 &= 0, \\
(1 - x_1) / p_2 - x_1 &= 0, \\
x_2 - x_1^2 / (1 + x_1) &= 0,
\end{align*}
\]

(4.1)

where $e$ is a constant. In this example $x = (x_1, x_2, x_3)$ and $p = (p_1, p_2)$. We chose $e = 3.25$ and $p_1 = 2000, p_2 = 1000$. The corresponding point solution $x^*$ is

\[x^* = (1.083, 0.5399, 0.001083).\]

(4.2)

The parameter vector $p$ was chosen to be

\[p = ([1980, 2200], [900, 1100]).\]

(4.3)

For this simple example, the interval hull $x^*$ of the solution set of (4.1), (4.3) can be easily computed to be (approximately)

\[x^* = ([0.9435, 1.2327], [0.4709, 0.6031], [0.00098, 0.001211]).\]

(4.4)

Application of Algorithms A1 and A2 with $e_2 = 0.05$ led to the following bound on (4.4):

\[x = ([0.9129, 1.254], [0.4546, 0.6182], [0.0009618, 0.001216]).\]

(4.5)

It is seen that the box (4.5) is an outer approximation of the solution set (4.4) of the perturbed system (4.1), (4.3).

The satisfaction of the inclusion (3.6) ensuring the validity of (4.5) was achieved for both algorithms at the first iteration of the second stage, i.e., for $k = 1$ of Procedure 3.3.

Table 1 lists the number of iterations $N_i$ needed to terminate stages 1 and 2 of the respective algorithms as well as the total number of iterations for each algorithm. It is seen that Algorithm A2 requires less iterations as compared to Algorithm A1.
Table 2.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.</td>
<td>27</td>
<td>8</td>
<td>35</td>
</tr>
<tr>
<td>Algorithm</td>
<td>13</td>
<td>7</td>
<td>20</td>
</tr>
</tbody>
</table>

EXAMPLE 4.2: In this example the perturbed system is

\[ 10^{-7}(e^{0.001x_1} - 1) + p_1x_1 - 1.6722x_2 + 0.0689x_3 - 8.0267 = 0, \]

\[ 1.9x + 0.10^{-9}e^{0.001x_1} - 1) + 0.6622x_1 + px_2 + 0.0622x_3 + 4.0535 = 0, \]

\[ 10^{-7}(e^{0.001x_1} - 1) + x_1 + x_2 + px_3 - 6 = 0. \]

\[ p = (p_1, p_2, p_3) = (0.6020, 0.7358, [1.21, 1.4801, [3.6, 4.4]) \]

(4.6a) (4.6b)

and models an electric circuit containing a transistor, a diode and two resistors [9].

Application of Algorithms A1 and A2 yielded the following results, respectively:

\[ x = ([0.5402, 0.5682], [-3.8926, -3.1153], [0.3483, 0.5387]), \]

\[ x = ([0.5402, 0.5683], [-3.8903, -3.1179], [0.3473, 0.5351]). \]

(4.7) (4.8)

For Algorithm A1, \( \epsilon_2 = 0.05 \) and the fulfillment of (3.6) was achieved at \( k = 1 \) of Procedure 3.3, For Algorithm A2 we chose \( \epsilon_2 = 0 \) and nevertheless (3.6) was satisfied already at \( k = 2 \) of Procedure 3.3. Thus, both bounds (4.7) and (4.8) are guaranteed to contain the solution set (4.6).

The numbers of iterations corresponding to the two algorithms are listed in Table 2. Once again, as in Example 4.1 Algorithm A2 outperforms Algorithm A1.

Example 4.2 was solved in [9] by an algorithm similar in structure to algorithm A2 (however, in [9] each iteration of both the first and second-stage of the algorithm requires the solution of a corresponding linear interval system) and the following bound was obtained

\[ x = ([0.5103, 0.5778], [-4.3520, -2.6756], [0.3483, 0.5898]). \]

(4.9)

It is worth noting that the bound (4.9) is more conservative as compared to (4.7) and (4.8) and at the same time took more iterations to be reached: total number of iterations 166 (85 iterations for the first stage and 81 iterations for second stage).

5. Conclusion

A method for tackling the problem of bounding the solution set of a parameter-dependent non-linear systems of equations (1.1) by an interval box \( x \) has been proposed. The method is based on a recently suggested linear interval enclosure
The theoretical basis of the method is provided in Section 2: Theorems 2.1 to 2.3. The method proper is presented in Section 3 where two two-staged algorithms are suggested. According to Theorem 3.1, their second stage (Procedure 3.3) involves the computational verification of the validity of the algorithms.

The new method is implemented as a computer program written in C++. Numerical evidence seems to indicate that it provides cheap and tight bounds on the solution set of the perturbed non-linear systems investigated. These bounds are, however, not rigorous since the present implementation of the method does not account for round-off errors. It is the intention of the authors to develop an algorithm and a computer program which will implement the method with complete computational rigor, thus providing infallible outer bounds on the perturbed solution set.

Acknowledgements

The authors are thankful to the referees for their remarks and suggestions.

References

WORST-CASE TOLERANCE ANALYSIS OF NON-LINEAR CIRCUITS USING AN INTERVAL METHOD

Lubomir V. Kolev and Valeri M. Mladenov
Faculty of Automation, Technical University of Sofia, 1166 Sofia, BULGARIA
Tel. +359 2 9842000, e-mail: valeri@univ.sofia.bg

Abstract

In this paper, a new iterative interval method applicable to both linear and non-linear circuit networks of non-linear elements in series is proposed. Results being more general in diffusion than other known methods in the way the components interconnection at each iteration is set up and solved approximately. Another distinction is the fact that new iterative linear interval problem associated with the non-linear characteristic for a current whose non-linear interval characteristics are linearized around the corresponding nominal dc operating point of the original non-linear circuit studied.

1 Introduction

Several methods have been put forward for solving the worst-case tolerance analysis problems for linear circuits [1]-[3]. The known methods are based on the following two basic approaches: (i) determining the range of a non-linear function defining the output variable of the interval parameters [1], [2], [3] and (ii) solving the corresponding system of linear equations having interval coefficients [1], [2], [3]. In both approaches, the non-linear functions are approximated by a series of linear functions. A new method based on interval calculus and on the principle of convex hulls for the non-linear functions is proposed. The method involves the setting up and solving of a linear system of equations. The method is applied to the worst-case tolerance analysis for non-linear circuits. The method is compared with the existing methods and it is shown that the proposed method is more efficient.

2 Problem statement

To simplify the presentation, only the circuit shown in Fig. 1 is considered. The circuit contains only independent current-controlled non-linear elements. Furthermore, it is assumed that the linear interval characteristics of the non-linear elements are known exactly. In this case, the circuit can be linearized by the linear method, and the resulting linear system can be solved using traditional linear analysis techniques.

Without loss of generality, it is assumed that only one non-linear element is present. The non-linear element is characterized by a set of linear interval characteristics, and the independent voltage source is present at the input of the circuit.

The circuit can be expressed in the form of a set of linear interval equations:

\[ y = A x + b \]

where \( y \) is the output vector, \( x \) is the input vector, \( A \) is the linear interval matrix, and \( b \) is the linear interval vector.

The solution of the interval linear system is obtained by solving a set of interval linear equations for all possible values of the input vector. The solution of the interval linear system is characterized by the fact that the interval linear equations are solved simultaneously for all possible values of the input vector.

The solution of the interval linear system is obtained by solving a set of interval linear equations for all possible values of the input vector. The solution of the interval linear system is characterized by the fact that the interval linear equations are solved simultaneously for all possible values of the input vector.
\( \mathbf{S} = \mathbf{S}(\mathbf{A}, \mathbf{B}) \) (1.1)

The above formulation can be extended to more general discrete-time circuits as well as to considering various linear time-invariant problems in nonlinear circuits.

3 The New Method

3.1 Algorithm of the Method

Once again, for brevity, only the linear version problem will be considered.

Let \( \mathbf{A} \) and \( \mathbf{B} \) denote the discrete-time linear system matrices. Using some standard methods for the linear version (2), the corresponding solution schema (1) can be found. Note that \( \mathbf{S}(\mathbf{A}, \mathbf{B}) \) is evaluated exactly.

\[
\mathbf{S}(\mathbf{A}, \mathbf{B}) = \mathbf{C}^{-1} \mathbf{B} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

The diagonal matrices \( \mathbf{C} \) with non-zero entries \( c_{ii} \) and zeros \( c_{ij}, i \neq j \) are formed. At this stage, the following linear discrete-time linear system problem is solved:

\[
\mathbf{C} \mathbf{x} = \mathbf{A} \mathbf{x} + \mathbf{B} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

Let \( \mathbf{X}^0 \) denote an approximate initial solution of (3). Having the property to ensure the result, an initial solution \( \mathbf{X}^0 \) is selected. At this point, each of (4) is represented as a linear equation of the following form (3):

\[
\mathbf{C} \mathbf{b} = \mathbf{a} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

Let the approximate solution of (7) be \( \mathbf{X}^0 \). At this stage, we have the system:

\[
\mathbf{S}(\mathbf{A}, \mathbf{B}) = \mathbf{C}^{-1} \mathbf{B} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

The above formulation can be extended to more general discrete-time circuits as well as to considering various linear time-invariant problems in nonlinear circuits.

3.2 Solving Linear Time-Invariant Problems

Since the original method exploits the discrete-time representation of systems (7), we use all efficient approaches. Each equation (7) is solved in each iteration. In [6], the corresponding linear time-invariant problem is solved approximately using an expanded version of Hessen's method [7]. The same method could be used for solving (7). A simple method is, however, suggested here.

Equation (7) is rewritten in the form

\[
\mathbf{C} \mathbf{x} = \mathbf{A} \mathbf{x} + \mathbf{b} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

where \( \mathbf{C}, \mathbf{A}, \mathbf{b} \) are the corresponding matrices. On account of (1.1), it is seen that 3 vectors \( \mathbf{C} \) diagonal matrices. The following systems are then introduced:

\[
\mathbf{C} \mathbf{x} = \mathbf{A} \mathbf{x} + \mathbf{b} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

where \( \mathbf{C} = \mathbf{C} \mathbf{C} \mathbf{C} \) is a real number (the slope of \( \mathbf{C} \) with \( \mathbf{X}^0 \) being \( \mathbf{X}^0 \) in a corresponding interval). By introducing the diagonal matrices \( \mathbf{C} \mathbf{x} \) whose entries are \( c_{ii} \), of \( \mathbf{C} \mathbf{x} \), \( a_{ij} \), \( b_{ij} \), \( c_{ij} \), and \( \mathbf{x} \), the following linear discrete-time linear system problem is solved and solved:

\[
\mathbf{C} \mathbf{x} = \mathbf{A} \mathbf{x} + \mathbf{b} \quad \text{for } \mathbf{C} \text{ diagonal matrices.}
\]

Let the approximate solution of (7) be \( \mathbf{X}^0 \). At this stage, we have the system:
4 A numerical example

To demonstrate the applicability of the new method to electrical circuit analysis, consider the circuit of Figure 1. The circuit parameters are given by:

\[ R_1 = 10 \Omega \quad \text{and} \quad R_2 = 20 \Omega \]

Using the new method, the following results are obtained:

\[ \begin{bmatrix} s & -2 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = 0 \]

The solution of the diagonal elements of matrix A was chosen to be 205. Using the algorithm described in section 3.1 and the simple method for solving the linear interval vector problem from section 3.2, the following interval vector \( \mathbf{x}^* \) is obtained:

\[ \mathbf{x}^* = \begin{bmatrix} 0.000 \pm 0.07 \end{bmatrix} \]

5 Conclusion

The proposed new numerical method is applicable to both discrete and continuous time-domain analysis of linear circuits. The method is based on the direct method of solving the linear system of circuit equations, which yields a more accurate solution than other numerical techniques used in standard circuit analysis. The proposed method is efficient and can be used for solving large and complex circuits. The accuracy of the method is demonstrated through the numerical examples provided.
X. International Symposium on Theoretical Electrical Engineering
September 6 - 9, 1999
Magdeburg, Germany

Conference Proceedings

International Steering Committee

International Steering Committee

T. Hanyu, Japan
A. Kovács, Hungary
I. Kose, Germany
E. E. Kriuzis, Greece
W. Marta, Germany
D. Mayer, Czech Republic
I. O. Mayergoyz, USA
K. Matiello, Poland
B. D. Rokaj, Yugoslavia
K. R. Richter, Austria
M. Rizzoli, Italy
H. Roman, Romania
A. Satava, Italy
B. Sipera, Poland
L. Slavik, Slovakia
J. K. Syrjäniemi, UK
M. Telcs, Hungary, Poland
C. W. Trowbridge, UK
H. Uekermann, Germany

Hosted by Otto-von-Guericke University
Institute for Electronics, Signal Processing and Communications

Sponsored by

IEEE DFG VDE ITG
A NEW INTERVAL METHOD FOR GLOBAL OPTIMIZATION

L. V. Kalev
Technical University of Sofia, Faculty of Automation, Bulgaria, Sofia 1756.
Email: Kalev@netcom.upsa

Abstract
Interval methods are intrinsic methods capable of solving the general nonlinear programming problem globally, providing reliable bounds both on the optimum (upper) and the corresponding optimal conditions. However, new, comprehensive, completely reliable, global optimization algorithms that do not have exponential numerical efficiency or provide only partial information about the solution (in the sense of correct bounds on the optimum) are not yet available. In this paper, a new interval method is suggested which seems to have improved numerical efficiency and is based on a new concept of intervalization of the nonlinear function model.

1. Introduction

Solving interval optimization problems using various standard optimization methods has been a major concern in the past two decades [7, 8].

In this paper, a new interval method is suggested for solving nonlinear function optimization problems. To solve this problem, we propose the following nonlinear function optimization problem:

\[ \min_{x \in R^n} f(x) \]

where \( x \) is a real-valued vector and \( f(x) \) is a given function, \( x \) is a real-valued vector, and \( f(x) \) is assumed to be continuous and differentiable.

In this problem, each function model is represented by two inequality constraints.

Let \( f(x) \) and \( g(x) \) be two continuous, differentiable functions. The interval method for solving (1) is based on the following interval linearization of \( f(x) \):

\[ L(x) = f(x) + \frac{\partial f(x)}{\partial x} \Delta x \]

where \( x \) is the vector of \( x \) and \( \frac{\partial f(x)}{\partial x} \) is either the partial derivative of \( f(x) \) or the corresponding interval value of the partial derivative of \( f(x) \) in the form

\[ f(x) = f(x) + \frac{\partial f(x)}{\partial x} \Delta x \]

where \( x \) is a real-valued vector and \( \Delta x \) is an interval. The use of (2) in the computational scheme of the new global optimization method leads to improved performance as compared to the other known methods, since it presents a better (tightest) enclosure of the original nonlinear function. Another advantage of the interval method is the fact that it is applicable to a wide range of functions that are not continuous or non-differentiable.

2. New interval linearization of a nonlinear function

Two algorithms for determining the new interval linearization will be presented in this section.

2.1. First algorithm

Let \( f(x) \) be a mathematical function \( f : D \subseteq R^n \rightarrow R \). The transformation of a mathematical function \( f(x) \), using the new nonlinear interval form (3), can be obtained by following the approach suggested in [9-10]. If \( f(x) \) is in separate form, \( x_1 \) and \( x_2 \) can be determined by a procedure given in [10]. For an arbitrary function \( f(x) \) which is continuous or non-differentiable, the procedure outlined by the approach suggested in [7, 9, 10] is transformed into a system of equations of the form:

\[ A x = b \]

where \( A \) is a \( m \)-dimensional matrix and \( b \) is a \( m \)-dimensional vector.

Thus, a system of nonlinear interval equations is generated. Finally, the system of equations is transformed into an interval form (3) corresponding to the interval function.

Then, the algorithm used above consists of the following steps:

Step 1. Transformation to separate form.

A function for which to be not separable form

\[ x = f(x) + \frac{\partial f(x)}{\partial x} \Delta x \]
none of these terms may be missing. The transformation of an arbitrary function into a set of functions of the univariate form can be done by the approach suggested in [4]. This possibility will be illustrated by the following example.

Example 1. Let

\[ f(x) = 3 + 2x^2 + 3x + 1 + x^3 \]  

with \( x \in \mathbb{R} \) and \( X = \{1, 2, 3\} \) \((5a)\).

The problem is to convert this into a set of univariate functions. With this in mind, we introduce two auxiliary variables \( x_1 \) and \( x_2 \) to get

\[ f(x) = 3 + 2x_1^2 + 3x_2 + 1 + x_1^3 \]  

with \( x_1, x_2 \in \mathbb{R} \) \((5b)\).

Now all expressions are in one or two variables. We need to choose the corresponding ranges:

\[ X_1 = \{1, 2\} \]  

and \[ X_2 = \{1, 2\} \] \((5c)\).

Step 2. Transforming the auxiliary variables:
The above example shows that the general case \( f(x) \) can be transformed by the introduction of a given number of auxiliary variables into a set of univariate functions of corresponding form.

\[ f(x) = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \]  

with \( x_1, x_2, \ldots, x_n \in \mathbb{R} \) \((5d)\).

With this in mind, each function \( f(x) \) can be replaced by a corresponding linear function \( \hat{f}(x) \) through the following transformation:

\[ \hat{f}(x) = \sum_{i=1}^{n} a_i x_i \]  

where \( a_i \in \mathbb{R} \) \((5e)\).

In the general case:

\[ f(x) = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j x_j \]  

with \( a_i, b_j \in \mathbb{R} \) \((5f)\).

Finally, \( f(x) \) is given by (5f) has been replaced in the form

\[ f(x) = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j x_j \]  

by the linear expression

\[ \hat{f}(x) = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j x_j \]  

with \( a_i, a_j, b_i, b_j \in \mathbb{R} \) \((5g)\).

2.2. Second algorithm

Now, the real advantage of \( f(x) \) is that \( f(x) \) is a linear functions \((5h)\), i.e., components of the linear combination \( a_i x_i + b_j x_j \) can be solved separately (e.g., see [2]). However, to simplify the presentation, the linear model structure \((5i)\) will be less determined for the class of univariate functions. Later on, the approach adopted for polynomial functions will be extended to arbitrary univariate functions.

A. Polynomial Functions

In the construction of an algorithm we shall suggest for determining the linear model structure \((5i)\), for the special case where \( f(x) \) is a polynomial function. This algorithm is based on the form of a polynomial interval \((5j)\). We shall consider a slightly different polynomial representation of the following form:

\[ f(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \]  

where \( c_i, d_j \in \mathbb{R} \) \((5k)\).

Let \( X = \{X_1, X_2, \ldots, X_n\} \)

\[ X_1 = \{1, \ldots, n\} \]  

and \( X_2 = \{1, \ldots, m\} \) \((5l)\).

In the general case:

\[ f(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \]  

with \( c_i, d_j \in \mathbb{R} \) \((5m)\).

Let \( \hat{f}(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \) \((5n)\).

We shall determine the components of \( \hat{f}(x) \) in the polynomial interval \((5o)\). We define the polynomial interval \((5p)\):

\[ f(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \]  

with \( c_i, d_j \in \mathbb{R} \) \((5q)\).

The problem is to determine \( \hat{f}(x) \) in the polynomial interval \((5r)\).

Finally, \( f(x) \) is given by (5r) has been replaced in the form

\[ f(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \]  

by the linear expression

\[ \hat{f}(x) = \sum_{i=0}^{n} c_i x_i + \sum_{j=1}^{m} d_j x_j \]  

with \( c_i, d_j \in \mathbb{R} \) \((5s)\).
Multiplication: The product \( E \) of two vectors \( A \) and \( B \) is another vector \( C \) defined by

\[
C = A \times B = \sum_{i=1}^{3} A_i B_i n_i
\]

where \( n_i \) are the unit vectors in the coordinate system.

(17a)

(17b)

(17c)

(17d)

(17e)

(17f)

(17g)

(17h)

(17i)

(17j)

(17k)

(17l)

(17m)

(17n)

(17o)

(17p)

(17q)

(17r)

(17s)

(17t)

(17u)

(17v)

(17w)

(17x)

(17y)

(17z)

(18a)

(18b)

(18c)

(18d)

(18e)

(18f)

(18g)

(18h)

(18i)

(18j)

(18k)

(18l)

(18m)

(18n)

(18o)

(18p)

(18q)

(18r)

(18s)

(18t)

(18u)

(18v)

(18w)

(18x)

(18y)

(18z)

(19a)

(19b)

(19c)

(19d)

(19e)

(19f)

(19g)

(19h)

(19i)

(19j)

(19k)

(19l)

(19m)

(19n)

(19o)

(19p)

(19q)

(19r)

(19s)

(19t)

(19u)

(19v)

(19w)

(19x)

(19y)

(19z)

(20a)

(20b)

(20c)

(20d)

(20e)

(20f)

(20g)

(20h)

(20i)

(20j)

(20k)

(20l)

(20m)

(20n)

(20o)

(20p)

(20q)

(20r)

(20s)

(20t)

(20u)

(20v)

(20w)

(20x)

(20y)

(20z)

(21a)

(21b)

(21c)

(21d)

(21e)

(21f)

(21g)

(21h)

(21i)

(21j)

(21k)

(21l)

(21m)

(21n)

(21o)

(21p)

(21q)

(21r)

(21s)

(21t)

(21u)

(21v)

(21w)

(21x)

(21y)

(21z)

(22a)

(22b)

(22c)

(22d)

(22e)

(22f)

(22g)

(22h)

(22i)

(22j)

(22k)

(22l)

(22m)

(22n)

(22o)

(22p)

(22q)

(22r)

(22s)

(22t)

(22u)

(22v)

(22w)

(22x)

(22y)

(22z)

B. Fumiceable functions

The approach suggested in Section 2.2.1 will be extended to arbitrary fumices, then in function

\( f: \mathbb{C} \to \mathbb{C} \) is a fumiceable function if and only if it can be represented by an expression \( f(x, y) = z \) which takes the form \( f(x, y) = z \) in function.

For the case of a function \( f(x, y) \), the list of admissible expressions is given in (2). An approach to finding non-differentiable functions is considered in (2).

(24a)

(24b)

(24c)

(24d)

(24e)

(24f)

(24g)

(24h)

(24i)

(24j)

(24k)

(24l)

(24m)

(24n)

(24o)

(24p)

(24q)

(24r)

(24s)

(24t)

(24u)

(24v)

(24w)

(24x)

(24y)

(24z)
\[ F_n = \sum_{i=0}^{n} a_i x_i \]  

where coefficients \( a_i \) and address sum \( b \) can be determined in a recursive way using only linear operations \((O(n))\) (addition, multiplication and subtracting of two \( O(n) \) intervals) and \((O(n))\) multiplication of \( O(n) \) interval by scalar.

Example 9.2. Find the interval enclosure for \( f(x) = \exp(x) \cdot x \), \( x_0=0 \), \( n=1, 2, 3 \).

The function \( f(x) \) can be defined by the corresponding difference \( f(x) = \exp(x) \cdot x \) following sequence:

\[ f(x) = f(x_0) + f(1)(x-x_0) + f(2)(x-x_0)^2/2 + \ldots \]

Applying the above algorithm we first have to compute the estimates for \((0,1,2,3)\) difference:

\[ F_0 = 1, F_1 = e, F_2 = e + 1, F_3 = e + 2 \]

Using Procedure 1 form [1] we have:

\[ \Phi_x(0,1) = 1, \Phi_x(1,2) = e, \Phi_x(2,3) = e + 1 \]

In a similar way, we find the estimates for \((0,1,2,3)\):

\[ \Phi_y(0,1) = 1, \Phi_y(1,2) = e, \Phi_y(2,3) = e + 1 \]

Here \( \Phi_x(0,1) = 1, \Phi_y(0,1) = 1 \) is the factorial polynomial for \((0,1)\)

\( \Phi_x(1,2) = e, \Phi_y(1,2) = e \) is the factorial polynomial for \((1,2)\)

\( \Phi_x(2,3) = e + 1, \Phi_y(2,3) = e + 1 \) is the factorial polynomial for \((2,3)\)

The equations \((11)\) and \((12)\) are then implemented in the form of two generalized norms \( F_0 \) and \( F_1 \). Finally, the estimates \( F_2(x) \) \((x)\) given by the product \( F_0(x)F_1(x)F_2(x) \)

\[ F_2(x) = \exp(x) \cdot x \]

Using \((9)\), we have obtained:

\[ E_0 = [0.125, 0.2] \rightarrow [e, e + 1] \]

The same example was solved in [9] using first- and second-order interval derivatives, and shown.

Procedure 3: Let \( F_0 \) be a closure upper bound of \( x_0(t) \) \( x_0 \), where \( x_0 \in \mathbb{R}^n \)

\[ x_0(t) = F_0(t) \]

3. The new method

In order to develop the following procedure to be carried out at each mention The internal estimates used in every procedure are implemented using the same linear format.

Procedure 1: Let \( F_0 \) be a closure lower bound of \( x(0) \), where \( x \in \mathbb{R}^n \)

\[ x(0) = F_0 \]

4. Numerical example

We consider the following optimization problem:

\[ \min \{ x \in \mathbb{R}^n : g(x) \leq 0, h(x) = 0 \} \]

where \( g(x) = x^2 - 2 \), \( h(x) = x - 1 \)

5. Conclusion

The solution to the problem is obtained.

\[ x^* = 1, g(x^*) = 1 \]

Thus, we conclude the following optimization problem (5).

\[ x^* = 1, g(x^*) = 1, h(x^*) = 0 \]
5. Conclusion

In this paper, the general nonlinear programming problem
is analyzed. A new method for solving interval functions with
unknown parameters is presented. It is based on an
approximation of the function by a polynomial of a
suitable degree, which is in turn used to estimate the
parameters of the function. The presented method
is capable of accurately estimating the parameters
with any desired accuracy.

4. Acknowledgments

This work was supported by the Bulgarian Research
Agency under Contract No 97-54997.

5. References

[1] [1] Moos, M. Methods and algorithms of interval
An Interval Method for Global Nonlinear Analysis

Lubonic Kolar

Abstract—This paper presents an interval method for solving the set of interval equations contained in the problem of finding the solutions to a system of nonlinear equations. The method is based on the following concepts: (i) intervalization of the nonlinear equations, (ii) intervalization of the interval equations, and (iii) intervalization of the interval solutions. The method is applied to a system of interval equations, and the results are compared with those obtained by other methods.

1. Introduction

The problem of finding the solutions of a system of interval equations is a fundamental problem in scientific and engineering applications. The method presented in this paper is based on the following concepts:

(a) intervalization of the nonlinear equations,
(b) intervalization of the interval equations,
(c) intervalization of the interval solutions.

The method is applied to a system of interval equations, and the results are compared with those obtained by other methods.

The resulting interval method is compared with the methods of interval analysis and interval derivatives. The results show that the interval method is more accurate than the methods of interval analysis and interval derivatives.

The interval method presented in this paper is based on the following concepts:

(a) intervalization of the nonlinear equations,
(b) intervalization of the interval equations,
(c) intervalization of the interval solutions.

The method is applied to a system of interval equations, and the results are compared with those obtained by other methods.

The resulting interval method is compared with the methods of interval analysis and interval derivatives. The results show that the interval method is more accurate than the methods of interval analysis and interval derivatives.
approach. First, by adding a certain number of auxiliary variables, the original general form system (3) is transformed into a larger system of the special form

$$f(x) = \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} f_i(x_i$$

$$= x, x' = 1, \ldots, x' = n, x = 1, \ldots, n$$

defining a $n$-dimensional box, i.e., when

$$f(x) = x^n$$

The next few sentences discuss the transformation of the general form system (3) to the system of separable form (10). It is noted that the system remains separable if the separable form is also separable. The separability of the computational scheme of the general form is discussed in Section II. The overall efficiency of the resulting system is evaluated in Section V.

The paper is organized as follows. Section II presents the transformation of the general form system (3) to the system of separable form (10). In Section III, the previous method is extended for the case of separable form on system (3). The improvements in computational scheme of the extended method are presented in Section IV. The overall efficiency of the resulting method is evaluated by several examples. Concluding remarks are given in Section V.

II. TRANSFORMATION TO SEPARABLE FORM

In this section, it will be shown how a system of general form (3) can be transformed into a system of separable form (10). More specifically, the system's components $f$ are assumed to be composed of four distinct operations: i.e., 1. Basic operations (such as multiplication and addition), 2. Elementary operations (such as polynomial functions), 3. And the general operation (a).

The transformation of (3) into (10) includes two stages: 1) transforming (1a) into (10), and 2) transforming (1b) into (10).

A. Transforming the System of Equations

The approach taken is to transform the general form system (3a) into the separable form (10a). The idea of decomposing a nonlinear function into a set of simpler functions has already been suggested in the literature (e.g., [21]), and [22]). This previous approach is a generalization of the algorithm from [24] which takes into consideration the separability of the nonlinear function.

Let $f_1$ and $f_2$ be subfunctions of $f$ depending on at least one variable. Consider the following four cases:

$$f = f_1 + f_2$$

$$f = f_1 f_2$$

If both $f_1$ and $f_2$ depend on only one and the same variable, then $f$ is obviously separable in all the four cases.

If $f_1$ depends on only one single variable and $f_2$ depends on another single variable, then only (10) is separable. Nevertheless, this case can be easily transformed into separable form. Consider the product (10). By introducing two auxiliary variables $x_i$ is transformed into separable form as follows:

$$f = f_1 f_2$$

$$f = f_1 f_2$$

$$f = f_1 f_2$$

The case can be easily reduced to the second by replacing $x_1 = f_1$ with $x_2 = f_1$. It should be noted that this is only possible if the original functions $f_1, f_2$ for all values of its argument. For the last case, the transformation to separable form is as follows:

$$f = f_1 f_2$$

$$f = f_1 f_2$$

$$f = f_1 f_2$$

where the product $f_1 f_2$ must be transformed using (10). It should be mentioned that (10) is valid for all values of the argument. If both $f_1$ and $f_2$ depend on more than one variable we first introduce auxiliary variables and then apply the above approach. For instance, case (2) is transformed using (10). Now, by representing $f_1$ and $f_2$ in separable form, $f$ can be put into separable form.

In order to make $f_1$ and $f_2$ separable, we perform the above transformations to $f_1$ and $f_2$, regarding them as if they were parts of a larger system. The process can be implemented on a computer program [24] for the more complex case of transformation to separable form.

To illustrate the above approach, we shall consider an example.

Example 1: Consider the system

$$f_1 = x_1 x_2 + x_3 x_4 + x_5 x_6 = 0$$

$$f_2 = x_7 x_8 x_9 = 0$$

$$f_3 = x_1 x_2 + x_3 = 0$$

where $x = [x_1, x_2, \ldots, x_n]$ belongs to the initial box $B = [0, 1]$ with components

$$x_1 = -1, \quad x_2 = 1, \quad x_3 = -1, \quad x_4 = 1, \quad x_5 = 0, \quad x_6 = 1.$$
On account of (16) and (17), we get
\[ 1 - r_x - r_z + 2(x_2 - x_z) = 0 \]
\[ 2x_2 - x_z = 0 \]
\[ r_x = 0 \]
\[ r_z = 0 \]
\[ x_2 = x_z = 0. \]  
(12a)

Equation (12a) is now in separable form. However, while the original system (11) has four equations of four variables, the separable system (12a) has two equations of two variables. For this reason, (12a) will be referred to as the augmented system.

So far, we have transformed (11) to the separable form version (12a). If we complete the solution of Example 1, we need also to solve the remaining two-dimensional (12a) box. \( \mathbb{V}^0 \) in a corresponding two-dimensional 1-D box, (10).

II. Formulate the Boundary Box

We now consider the transformation of \( \mathbb{V}^0 \) in Example 1. We do so by replacing the transformed box by the augmented system of separable form.

\[ x_0 = x \]
\[ x_0 = x \]
\[ x_0 = x \]
\[ x_0 = x \]  
(12a)

The transformed system is in separable form.

III. SOLVING SEPARABLE FORM SYSTEMS

In this section, an internal method for solving the separable form system (12a) will be presented. It is an extension of a previous method (13) developed for the special case of separable systems. More specifically, the idea is to reduce the separable form (12a) by an appropriate internal function (13) to a form representing models of the separable systems. The transformed system (12a) is then replaced by the corresponding two-dimensional 1-D box.

A. Enclosure for Separable Functions

To motivate completeness, let us first write the separable form system (12a) in a more familiar form.

Enclosure for \( l(x) \): Let \( X = (x_0, x_1, x_2, x_3) \) \( X \subseteq \mathbb{V}^0 \). The linear interval enclosure approximation of \( l(x) \) in \( X \) suggested earlier (14) is in the form
\[ l(x) \geq l \]  
(18)

where \( B_0 \) is an interval, where \( \nu \) is a real number with \( \nu \), and \( c \cdot \nu \) is the nucleus of \( \nu \), and \( c \) is a constant such that the following inclusion property holds:
\[ l(x) \in B_0 + c \nu \cdot \mathbb{V}^0 \]  
(19)

It is an account of (21) that \( l(x) \) is called a linear interval enclosure approximation of \( l(x) \). A procedure for determining \( \nu \) and \( B_0 \) which will be called Procedure 1, has been suggested in (13).
Remark 1. In the original paper (13), no restrictions on the functions \( f(x) \) are imposed except for the requirement that they be continuous. The Procedure 1 denotes the procedure for determin- ing the envelope (22) in applicable for the case of continuous differentiable (C2) functions and parabolic linear (P2) functions. If not, however, be shown using simple geometric considerations as in (13) that the linear integral envelope (22) can also be constructed, even in the case of discontinuous functions having bounded discontinuities.

2. Enclosure for \( a, y \).

To simplify notation, we shall consider the product

\[
x \in X, \quad y \in Y
\]

where \( x \) and \( y \) are intervals rather than the product space \( X \times Y \).

If \( x = x_0 + \Delta x \) and \( y = y_0 + \Delta y \), then

\[
x = x_0 + \Delta x_0 + \Delta x + \Delta y_0 + \Delta y
\]

(22)

When \( x \) and \( y \) are in \( X \) and \( Y \), the extreme values of \( \Delta x_0, \Delta y_0 \) and \( \Delta x, \Delta y \), respectively, are the intervals of \( X \) and \( Y \). Let \( B_r, B_y \) be intervals of \( X \) and \( Y \). Let \( B_r = (B_r)^{-} \cup (B_r)^{+} \) and \( B_y = (B_y)^{-} \cup (B_y)^{+} \).

Thus, the product \( x \) has been enclosed by a linear interval expression since

\[
x = x_0 + \Delta x_0 + \Delta x + \Delta y_0 + \Delta y
\]

(23)

where \( x_0, \Delta x_0 \), and \( \Delta y_0 \) are real numbers while \( \Delta x, \Delta y \) are intervals of \( X \) and \( Y \).

Now an interval.

Returning to the original form (22), a variation of the 2nd order in \( C(\alpha, \beta) \) in (20), it is easily verified that the corresponding linear interval

\[
x = x_0 + \Delta x_0 + \Delta x + \Delta y_0 + \Delta y
\]

(24)

with \( x_0, \Delta x_0 \), and \( \Delta y_0 \) are real numbers while \( \Delta x, \Delta y \) are intervals of \( X \) and \( Y \).

The proof of the paper's theorem is made to that of Theorem 1 in (15) and will therefore be omitted.

Remark 2. It should be noted that \( A \) and \( V \) depend on the box \( X \). i.e., (15). It should also be noted that \( A \) and \( V \) need not be

\[
A = b = 0, \quad b \in B(\alpha, \beta)
\]

(25)

where \( B(\alpha, \beta) \) is any interval contained in \( B \).

Now an interval.

To simplify notation, here and hereafter the shorter form \( A \) and \( V \) is used.

B. The Generalized Method

Using the asymptotic representation (26) and the inclusion (21) and (25), it is seen that the following inclusion is valid

\[
x = x_0 + \Delta x_0 + \Delta x + \Delta y_0 + \Delta y
\]

(26)

where

\[
x = x_0 + \Delta x_0 + \Delta x + \Delta y_0 + \Delta y
\]

(27)

Now an interval.

A matrix

\[
A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}
\]

is introduced and a \( n \times n \) dimensional interval vector

\[
V = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}
\]

is formed. Then, (27a) can be put in vector form

\[
f(\mathbf{x}) \in \mathcal{A} \mathbf{u} + \mathcal{V}, \quad \mathbf{x} \in X.
\]

(28)

If \( y \) is a solution of (25), then \( f(x) = 0 \) and by (28)

\[
0 \leq \mathcal{A} \mathbf{u} + \mathcal{V}, \quad \mathbf{x} \in X.
\]

(29)

Now we can state the main result of the section.

Theorem 2. All the functions \( y \) to

\[
f(\mathbf{x}) = 0, \quad \mathbf{x} \in X
\]

(30)

with \( f(x) \) given by (28), which are contained in the \( n \)-dimensional box \( X \) are also contained in the solution set \( \mathcal{S}(\mathbf{u}, \mathcal{V}) \) of the system

\[
A \mathbf{u} + \mathbf{V} = 0, \quad \mathbf{u} \in \mathcal{U}(X).
\]

(31)

where \( \mathcal{U}(X) \) is any interval.

To simplify notation, here and hereafter the shorter form \( \mathcal{A} \) and \( \mathcal{V} \) is used.

Formula (1), Theorem 1 of this paper is an almost verbatim replica of Theorem 1 of the earlier paper [15] where the separable form equations were considered. In reality, this paper's result is more general since it covers the case of equations in separable form. The most important distinction is the inclusion (21) and, more specifically, the expression (25) and (27a) for the slopes \( a_{ii} \) and the intervals \( I_{ii} \), respectively.

Based on Theorem 1, the method of (15) is readily extended to cover the separable case. However, the only difference is that now the square real matrix \( A \) is a (2 \times 2) interval matrix in (25) and the intervals \( I_{ii} \) in (26) are \( (n \times n) \) dimensional while their counterparts in (11)
Theorem 2. Let \( F: \mathbb{R}^n \rightarrow \mathbb{R}^n \) be a C^2 function in the domain \( D \) and \( x^* \in D \). Introduce the operator
\[
\mathcal{K}(x) = -\nabla f(x) - \nabla g(x),
\]
where \( f(x) \) and \( g(x) \) are defined as in Procedures 1 and 2, respectively. Then, if \( x \) is a minimum of \( \mathcal{K}(x) \), the 2nd order necessary condition is satisfied.

Example 1. The problem is to find all real solutions to the system (10) of polynomial equations. The system was solved using the Newton-Raphson method by [11]. The number of iterations needed to locate a solution within an accuracy \( \epsilon \) was \( N = 50 \). The method was also solved using the method described in [12], [13], [14]. Although the number of iterations needed to locate a solution within an accuracy \( \epsilon \) was \( N = 50 \), the corresponding minimum was found in all cases. The global analysis problem described in [15] was used to find the minimum of the test function.

Example 2. The problem is to find the following voltage transfer function:
\[
V(x) = \frac{1}{0.16x^2 + 2.471}
\]
by means of a global optimization method [11, 12, 13, 14]. It is desired to determine the components of the voltage vector \( V(x) \), \( V_1(x) \), \( V_2(x) \), \( V_3(x) \), and \( V_4(x) \).
It can be checked that these values are solutions of the following design equations system:

\[ \begin{align*}
    x_{12} &= 2.474 \\
    x_{23} &= 0.196 \\
    x_{11} + x_{12} + x_{13} &= 4.605 \\
    x_{21} + x_{22} + x_{23} &= 6.866 \\
    x_{12} + x_{22} + x_{32} &= 5.029 \\
    x_{13} + x_{23} + x_{33} &= 0.327
\end{align*} \]

(53)

where \( x_{ij} = f_{ij} \), \( i, j = 1, 2, 3 \), and \( x_{11} = 1/3 \). The result box \( \lambda'' \) is given by

\[ \lambda'' = [0.60, 1.246] \quad \lambda''^{(2)} = [0.60, 1.246] \quad \lambda''^{(3)} = [0.60, 1.246] \]

The derived solution accuracy \( \varepsilon \) was chosen to be \( 10^{-6} \). Using Algorithm 1, three solutions

\[ \begin{align*}
    r_1 &= 1.121 \pm 0.008, 1.586 \pm 0.122, 1.556 \pm 0.021 \\
    r_2 &= 1.121 \pm 0.008, 1.702 \pm 0.019, 1.781 \pm 1.087
\end{align*} \]

(54)

of \( \varepsilon = 0.008 \) have been found within the shown accuracy. The number of iterations \( N \) required in (91) is 2. It is worthwhile noting that Knowles’s method requires 179 iterations to find all the three solutions (54) at the same accuracy.

IV. IMPLEMENTATIONS

As shown in Section III.B, to Example 2, a section 1 of the generalized method reduces, essentially, to setting up and solving the following \( n = n \) linear system

\[ \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} \]

(55)

where in Example 2, \( A \) and \( b \) are given in (51). In this section, two modifications will be introduced into the computational scheme of Algorithm 1. The first is associated with the elimination of all auxiliary variables from the linear system (56). Thus, (55) is transformed to a system

\[ \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} \]

(57)

of reduced \( n = n \) size. The second modification consists of applying the control propagation approach (e.g., (56)) to the updated system (57). These modifications result in a considerable improvement of the numerical efficiency of Algorithm 1.

A. Reformulating the Auxiliary Variables

The computational efforts needed to solve (57) can be substantially reduced (for large \( n \) and \( n \) if the auxiliary variables are eliminated from system (57)). This possibility will be shown by way of Example 2. Indeed, from (50)

\[ \begin{align*}
    x_{12} &= x_{11} + x_{12} + x_{13} = 4 \\
    x_{23} &= x_{21} + x_{22} + x_{23} = 6 \\
    x_{13} &= x_{11} + x_{12} + x_{13} = 8
\end{align*} \]

(58)

Substituting (58) into the third and fourth equation of (56), we get

\[ \begin{align*}
    a_1 x_1 + a_2 x_2 + a_3 x_3 &= 10 \\
    a_1 x_1 + a_2 x_2 + a_3 x_3 &= 10 \\
    a_1 x_1 + a_2 x_2 + a_3 x_3 &= 10 \]

(59)

where

\[ \begin{align*}
    a_1 &= x_{11} - x_{12} - x_{13} \\
    a_2 &= x_{21} - x_{22} - x_{23} \\
    a_3 &= x_{31} - x_{32} - x_{33}
\end{align*} \]

(60)

It is seen that the new system (60) has only four equations of four variables, while the augmented linear system (57) has six equations of six variables.

In the general case, by elimination of the auxiliary variables, the linear interval system of augmented dimension \( n = n \) is reduced to system (60) of dimension \( n \). Solving (60) is a much easier problem than solving (57), especially for larger \( n \) and \( n \).

It should be emphasized that the elimination of the auxiliary variables is carried out in each iteration, i.e., for each current box \( X \). Therefore, we have to compute, at each iteration, bounds on the auxiliary variables. This can be done using (17) i.e., by evaluating inner and outer bounds \( f_{a,1}(X) \) and \( f_{a,1}(X) \) for all \( i \). The range \( f_{a,1}(X) \) of \( f_{a,1}(X) \) when \( f_{a,1}(X) \) is the ith component of the vector function \( f_{a,1}(X) \). In general, the interval solution \( f_{a,1}(X) = f_{a,1}(X) \) of \( f_{a,1}(X) = X \) can be computed using some of the methods available in interval analysis [10, 11, 12]. However, Procedure 1 allows an alternative simpler approach in evaluating \( f_{a,1}(X) \). Indeed, \( f_{a,1}(X) \) are univariate functions. Therefore

\[ f_{a,1}(X) = \sum_{i=1}^{n} f_{a,1,i}(X) \]

(61)

where

\[ f_{a,1,i}(X) = \min \{ f_{a,1,i}(f_{a,1}, X) \}
\]

(62)

Here

\[ f_{a,1,i}(X) = \max \{ f_{a,1,i}(f_{a,1}, X) \}
\]

(63)

\[ f_{a,1,i}(X) = \sum_{i=1}^{n} f_{a,1,i}(X)
\]

(64)

\[ f_{a,1}(X) = \sum_{i=1}^{n} f_{a,1,i}(X)
\]

(65)

Thus, the interval expression \( f_{a,1}(X) \) can be computed in the form

\[ f_{a,1}(X) = \sum_{i=1}^{n} f_{a,1,i}(X)
\]

(66)

where \( f_{a,1,i}(X) \) is the interval expression of the range

of \( f_{a,1,i}(X) \) at \( X \).
The algorithm of the present method which implements the elimination of the independent variables will be referred to as ALGORITHM 1.

Example 1. The considered example is the well-known network problem on a finite and a more realistic, the network problem on a finite and an infinite network, respectively, with a linear equation:

\[ \begin{align*}
    x_1 + x_2 &= 3 \\
    2x_1 - x_2 &= 3
\end{align*} \]

The simultaneous solution is:

\[ \begin{align*}
    x_1 &= 1 \\
    x_2 &= 2
\end{align*} \]  

(1)

The last equation of this is not an independent term. Hence, however, by each time interim value with this by introducing an auxiliary variable:

\[ x_3 = x_1 - x_2 \]  

(2)

The corresponding two simultaneous equations are:

\[ \begin{align*}
    x_1 + x_2 + x_3 &= 3 \\
    2x_1 - x_2 + x_3 &= 3
\end{align*} \]

(3)

The initial \( x_n \) was chosen to have equal components:

\[ x_n = \left[ \begin{array}{c}
    1 \\
    1 \\
    1
\end{array} \right] \]

(4)

The example considered was solved using Algorithm 1, which solves the simultaneous equations (5) and Algorithm 2, which solves the system of linear equations. Both algorithms have local stability, the two solutions \( (x) \) with \( x_n \) are.

(5)

In the corresponding experiment, Algorithm 1 with solution (5) and Algorithm 2 with solution (2) were used in the two solution systems, respectively.

The example considered was solved using Algorithm 1, which solves the simultaneous equations (5) and Algorithm 2, which solves the system of linear equations. Both algorithms have local stability, the two solutions \( (x) \) with \( x_n \) are.

For computation, the example given in the same example has been solved using Algorithm 1, which solves the simultaneous equations (5) and Algorithm 2, which solves the system of linear equations. Both algorithms have local stability, the two solutions \( (x) \) with \( x_n \) are.

(6)

\[ Y_n = X - \sum_{i=0}^{x_n} a_i X_i \]  

(7)

It is seen that this stage implements the known interval Gauss-Seidel scheme (12)-(14). Now, if (14) is, however, a simple version since at least one of the computed intervals contains the correct solution, then the number of intervals needed to converge may be, in general, more than the number of intervals needed to converge.

Stage II. Now one iteration of Procedure 3 is applied to the last, which is referred to from Stage I.

**B. Use of Convex Programming**

The convexity property is related to the application of this technique, as well as to the so-called superposition of the solution.

(8)

The problem is to find an interval solution to (5), that is, an interval vector \( \mathbf{x} \) which contains the solution of (6).

\[ \mathbf{X} = \left[ \begin{array}{c}
    \mathbf{x}_1 \\
    \mathbf{x}_2 \\
    \vdots \\
    \mathbf{x}_n
\end{array} \right] \]

(9)

Similarly, it is seen that each component \( \mathbf{x}_i \) of (10) is an interval, with each interval vector \( \mathbf{x}_i \) can be determined by solving two linear programming problems:

\[ \mathbf{y}_1 = \min \mathbf{c} \mathbf{x} \]  

(10)

\[ \mathbf{y}_2 = \max \mathbf{c} \mathbf{x} \]

(11)

(12)

(13)

(14)

(15)

(16)

(17)

(18)

(19)

(20)

(21)

(22)

(23)

(24)

(25)

(26)

(27)

(28)
**Algorithm 4A.** This is the extended version of the previous algorithm, in which the first stage is modified as follows.

\[
Y_i = \frac{1}{h} \left( \sum_{k=1}^{n} \frac{x_k}{x_k} \right)
\]

Stage 1. (The same as in algorithm 4A.)

Stage 2. The algorithm is repeated on all elements of the set of the real roots, and the internal vector \( Y \) is computed at the start of the current iteration and stored until the iterations have been performed during the iteration. A linear approximation is used in the computation of \( i \) and \( j \) is implemented in the next algorithm.

Algorithm 5. The algorithm is used to compute \( i \) in the following manner. Initially \( i = 1 \), compute the first two of \( X \) and if the first two of \( X \) are computed, then the second of \( X \) is computed. The second of \( X \) is computed, and the second of \( X \) is computed, and the second of \( X \) is computed.

Stage 3. The algorithm is repeated on all elements of the set of the real roots, and the internal vector \( Y \) is computed at the start of the current iteration and stored until the iterations have been performed during the iteration. A linear approximation is used in the computation of \( i \) and \( j \) is implemented in the next algorithm.

**Example 3:** The system to be solved is:

\[
\begin{align*}
&x_1(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_2(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_3(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_4(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_5(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_6(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_7(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_8(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
&x_9(t) = 2x_1(t) + 3x_2(t) + 4x_3(t) \\
\end{align*}
\]

The initial guess \( X(0) \) has the following component:

\[
X^{(0)} = X^{(0)} = \begin{bmatrix} 2.0 & 0.0 \end{bmatrix}, \quad i = 2, \ldots, 9.
\]

The GNA problem considered has 9 solutions:

\[
X^{(0)} = \begin{bmatrix} -0.3756 & 0.2375 \end{bmatrix}, \quad i = 0, 1, 2, \ldots, 9.
\]

The GNA problem considered has 9 solutions:

\[
X^{(0)} = \begin{bmatrix} -0.3756 & 0.2375 \end{bmatrix}, \quad i = 0, 1, 2, \ldots, 9.
\]

The GNA problem considered has 9 solutions:

\[
X^{(0)} = \begin{bmatrix} -0.3756 & 0.2375 \end{bmatrix}, \quad i = 0, 1, 2, \ldots, 9.
\]

The GNA problem considered has 9 solutions:

\[
X^{(0)} = \begin{bmatrix} -0.3756 & 0.2375 \end{bmatrix}, \quad i = 0, 1, 2, \ldots, 9.
\]
(67) is solved in a most efficient manner using the potential of propagation approach as implemented in Algorithm 3.

Experiments (7a) show that as regards computer time and memory values, requirements, the present method surpasses the other known iterative methods for solving the ODE problem considered.

These results hint on a possibility for further improvement of the numerical efficiency of the new method by incorporating some scheme ideas and techniques from other subroutines (49) in order to increase the solution of the original nonlinear system (11) as the linear system (67).

APENDIX
Proof of Theorem 2 Let (a) and (b) hold. Then (2b) can be transformed into the fixed point format

\[ \mathbf{C} \pi(x) = \pi(x) \]

where \( \mathbf{C} = a \mathbf{N} - b \mathbf{I} \) and \( \mathbf{N} \) is the given matrix. We assume

\[ \mathbf{C} \in \mathbf{L}(\mathbf{R}^n, \mathbf{R}^n) \]

where:\n
\[ \mathbf{L}(\mathbf{R}^n, \mathbf{R}^n) \]

is the set of all linear maps from \( \mathbf{R}^n \) to \( \mathbf{R}^n \).

Obviously, for any \( x \in \mathbf{R}^n \),

\[ \mathbf{C} \pi(x) = \pi(x) \]

Thus, the fixed point theorem (2b) is satisfied. Therefore, by the Banach fixed point theorem, there is a unique fixed point \( \pi(x) \) which is the solution of (2b).

ACKNOWLEDGMENT
The authors express their sincere thanks to the valuable reviewers for their valuable comments and suggestions.

REFERENCES
AN INTERVAL METHOD FOR GLOBAL INEQUALITY-CONSTRAINT OPTIMIZATION PROBLEMS

L. Kolev and D. Penev
Faculty of Automation
Technical University of Sofia, Sofia 1756, BULGARIA
e-mail: kolev@fa.tu-sofia.bg

ABSTRACT

An interval method is suggested for solving the global inequality-constraint optimization problems of the following type: minimize a given objective function subject to a box and linear inequality-constraint constraints. The method is based on a new interval approach to solving the inequality-constraint constraints, which is a combination of slack and slack and difference constraint methods. The set of these techniques in the comprehensive solution of the global problem enables to lead to improved performance in comparison to some special interval methods of the same class.

1. INTRODUCTION

Interval methods (see e.g., [1]-[4]) are applied in solving mathematical problems (see e.g., [5]-[7]). In this paper, a new interval method is suggested for global optimization of the following type: minimize the objective function

\[ f(x) \]

subject to the constraints

\[ g_i(x) \leq 0, \quad i = 1, \ldots, m \]

\[ l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n \]

where \( f(x) \) is a real-valued function and \( g_i(x) \) is a given linear or nonlinear function. The objective function is subject to the constraints, which are in the form of linear or nonlinear functions.

There exist several methods for solving problems of this type (see e.g., [1]-[4], [11]-[13]). However, interval methods are not so popular in solving problems of this type. The reason is that they are not as efficient as other methods, such as the Newton-Raphson method or the gradient descent method.

2. LINEAR PROGRAMMING APPROACH

In this section, we introduce the interval programming shift problem to be incorporated into the comprehensive solution of the linear programming problem.

\[ \min f(x) \]

subject to

\[ Ax \leq b \]

where \( f(x) \) is a real-valued function on \( x \in \mathbb{R}^n \) and \( A \in \mathbb{R}^{m \times n} \). The objective function is subject to the constraints, which are in the form of linear functions.
Procedure 7.
It is assumed that the input to the approach of [?] are the linear constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and the objective function $f = \mathbf{c}^T \mathbf{x}$.

Step 1. Form the corresponding LP and solve $d = \arg \min \{ f \}$.

```
\text{Minimize} \quad f = \mathbf{c}^T \mathbf{x} \\
\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
\mathbf{x} \geq 0
```

Step 2. Solve the following LP problem:

```
\text{Maximize} \quad d = \sum_{i=1}^{m} c_i x_i \\
\text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m \\
\mathbf{x} \geq 0
```

and the corresponding dual $d = \arg \max \{ d \}$.

```
\min \{
\sum_{i=1}^{m} c_i \xi_i \\
\text{subject to} \quad \sum_{i=1}^{m} a_{ij} \eta_i \geq b_j, \quad j = 1, \ldots, m \\
\xi_i, \eta_i \geq 0
\}
```

To obtain the optimum $d$, the optimum values of $\xi_i$ and $\eta_i$ can be obtained.

```
\xi_i = \frac{b_j - \sum_{i=1}^{m} a_{ij} \eta_i}{c_i}, \quad i = 1, \ldots, m \\
\eta_i = \frac{b_j - \sum_{i=1}^{m} a_{ij} \xi_i}{c_i}, \quad i = 1, \ldots, m
```

where $f$ is the value of the objective function at the optimum LP solution.

```
f = d
```

```
f = \sum_{i=1}^{m} c_i x_i
```

3. CONSTRAINT PROPAGATION

This method has already been applied in the context of solving linear programming problems using the simplex method. The following criteria are used for constraint propagation:

- If the constraints are not saturated, then the constraint is retained.
- If the constraints are saturated, then the constraint is relaxed.

```
\begin{align}
& \frac{\sum_{i=1}^{m} c_i x_i}{c_i} = \max \{ \frac{b_j - \sum_{i=1}^{m} a_{ij} \eta_i}{c_i}, \frac{b_j - \sum_{i=1}^{m} a_{ij} \xi_i}{c_i} \}, \\
& \mathbf{x} \geq 0, \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}
\end{align}
```

The dual LP problem is solved for each constraint $c_i$.

```
\begin{align}
& \text{Minimize} \quad \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

```
\begin{align}
& \text{Maximize} \quad d = \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \geq b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

```
\begin{align}
& \text{Minimize} \quad \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

```
\begin{align}
& \text{Maximize} \quad d = \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \geq b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

```
\begin{align}
& \text{Minimize} \quad \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i = b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

```
\begin{align}
& \text{Maximize} \quad d = \sum_{i=1}^{m} c_i x_i \\
& \text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i = b_j, \quad j = 1, \ldots, m \\
& \mathbf{x} \geq 0
\end{align}
```

Each inequality constraint is treated as a separate case.
4. ALGORITHM OF THE METHOD

The algorithm is described in Algorithm 1.

Algorithm 1: Algorithm Description

5. NUMERICAL EXAMPLES

Case 1: Example

Case 2: Example

Case 3: Example
Table 1

<table>
<thead>
<tr>
<th>S</th>
<th>T</th>
<th>G</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The above table is an example of a table used in the text to present data.

6. SUMMARY

A new exact method for globally solving non-linear convex quadratic problems is presented. This method has the same degree of complexity as other known methods, but for global optimization, it is not the best choice. For problems with linear constraints, the new method is less effective than other known methods.

The new method is a combination of the global optimization method and the local optimization method. The global optimization method is used to find the global minimum of the objective function, and the local optimization method is used to refine the solution obtained by the global optimization method.

The results of the new method are compared with the results obtained by other methods. The new method is more effective than the other methods in finding the global minimum of the objective function.

7. REFERENCES


