Experiments Using Interval Analysis for Solving a Circuit Design Problem

H. RATSCHEK*
Mathematisches Institut der Universität Düsseldorf

and

J. ROKNE
Department of Computer Science, The University of Calgary

(Received: 16 April 1992; accepted: 3 March 1993)

Abstract. An already classical attempt at solving a circuit design problem leads to a system of 9 nonlinear equations in 9 variables. The sensitivity of the problem to small perturbations is extraordinarily high. Since 1974 several investigations have been made into this problem and they hint at one solution in the restricted domain of the nonnegative reals. The investigations did not give error estimates nor did they present conclusive evidence that the solution found is the only one in the domain of the nonnegative reals. Our paper reports on experimental computations which used various kinds of interval analytic methods while also sometimes reflecting on Wright–Cutteridge's philosophy and theses. The computations resulted in a guarantee that in the domain of consideration, that is, the interval [0, 10] for each of the 9 variables, exactly one solution did exist, which was near the solution known up to now. Finally, our solution could be localized within a parallelpiped with edge lengths between 10^{-6} and $3.2 \cdot 10^{-4}$.

Key words. Global optimization, least squares problems, global zero search, electrical circuits, transistor modeling, interval computations, branch-and-bound.

1. Introduction

Based on the fundamental work of Ebers and Moll [4] a bipolar transistor was modeled by an electrical circuit ([2, 3]) leading to the following set of equations

$$\begin{aligned}
&\alpha_k(x) = 0\,, \quad k = 1, \ldots, 4 \\
&\beta_k(x) = 0\,, \quad k = 1, \ldots, 4 \\
&\gamma(x) = 0
\end{aligned} \tag{1}$$

in the variable $x \in R^9$ where

$$\begin{aligned}
\alpha_k(x) = (1 - x_1x_2)x_3\{\mathrm{e}^{[x_5(g_{1k} - g_{3k}x_7 10^{-3} - g_{5k}x_8 10^{-3})]} - 1\} \\
-g_{5k} + g_{4k}x_2\,, \quad k = 1, \ldots, 4\,,
\end{aligned}$$

*Thanks are due to the Natural Sciences and Engineering Research Council of Canada for supporting this paper.

Journal of Global Optimization **3**: 501–518, 1993.

$$\begin{aligned}\beta_k(x) &= (1 - x_1x_2)x_4\{e^{[x_6(g_{1k} - g_{2k} - g_{3k}x_7 10^{-3} + g_{4k}x_9 10^{-3})]} - 1\} \\ &\quad -g_{5k}x_1 + g_{4k}\,, \quad k = 1, \ldots, 4\,, \\ \gamma(x) &= x_1x_3 - x_2x_4\,.\end{aligned} \tag{2}$$

The numerical constants were given by ($k = 1, \ldots, 4$):

g_{1k}	0.485	0.752	0.869	0.982
g_{2k}	0.369	1.254	0.703	1.455
g_{3k}	5.2095	10.0677	22.9274	20.2153 .
g_{4k}	23.3037	101.779	111.461	191.267
g_{5k}	28.5132	111.8467	134.3884	211.4823

These were obtained as a result of laboratory experiments. For the purpose of this paper, however, the system (1) is assumed to be a test problem to be globally solved. The presence of the exponential terms with large constants in (2) results in a very large variation in function and derivative values over moderate input values and makes the system (1) extremely hard to solve. The problem therefore appealed to several researchers who tried a variety of approaches for overcoming the large difficulties encountered.

In 1974, Cutteridge [2] combined local damped Newton–Raphson steps with the conjugate gradient method and a second-order gradient-descent method with eigenvalue determination where the two latter methods were applied to the least squares problem as to minimize

$$f = \gamma^2 + \sum_{k=1}^{4} (\alpha_k^2 + \beta_k^2) \tag{3}$$

which corresponded to (1). The approximate solution

$$x_A = (0.9 \quad 0.45 \quad 1.0 \quad 2.0 \quad 8.0 \quad 8.0 \quad 5.0 \quad 1.0 \quad 2.0)^T \tag{4}$$

was found and it was claimed that it was accurate to four figures. Cutteridge emphasized that only the sophisticated combination of the three methods had led to a positive result i.e., it did not suffice to only use the first two approaches mentioned above which was the state of the art for methods for solving such problems in 1974.

In 1978 Price [18] applied a controlled random search (CRS) procedure to the least squares problem (3). For a first run, the initial search domain was $0 < x_k < 10$ for $k = 1, \ldots, 9$. This first run resulted in 200 points that were clustered around

$$x^{(1)} = (0.9 \quad 1.0 \quad 3.7 \quad 3.9 \quad 7.1 \quad 8.1 \quad 0.4 \quad 1.9 \quad 2.8)^T \tag{5}$$

where $f(x^{(1)}) = 22.6$. The second run took an initial search domain having the same volume as the first, but centered on $x^{(1)}$. This resulted in a point $x^{(2)}$ with a negative component, $x_7^{(2)} = -1.75$ and a value $f(x^{(2)}) = 3.8 \times 10^{-2}$. This re-start strategy was applied several times till after the sixth run no significant improvement was obtained. Thus, an optimum point $x^{(6)}$ was found which had two negative components,

$$x_7^{(6)} = -8.049 \quad \text{and} \quad x_9^{(6)} = -0.4278$$

where the function value was $f(x^{(6)}) = 3.9 \times 10^{-4}$. Price mentioned that

> *It is not known how many other global minima the function possesses, but one occurs close to the point x_A.*

He furthermore stated that it would appear that the sensitivity of the function is very much greater in the region of the other minima.

In 1980 Dimmer and Cutteridge [3] made an attempt to solve (1) directly with an elaborate Gauss–Newton variant that used second derivatives. Examinations of the residuals (3), were incorporated in order to choose suitable correction vectors for the Gauss–Newton steps. The area of successful starting points was – compared with the previous methods – rather small and was obtained by perturbing x_A within a maximum distance of 1.2 w.r.t. each component. This area is, however, larger than the areas tried with Hartley's method, Marquardt's method, and a quasi-Newton method, cf. [3]. The only solution that could be found was again x_A.

The approaches mentioned above only provide some *evidence* for the fact that x_A is the only solution with positive components. There could be a number of solutions with positive components to the equations (1) and (3) and no explicit error estimate of x_A has been available so far. The aim of this paper is therefore to show that

> *A solution close to x_A is the only global solution of the equations (1) in the domain $[0, 10.0]^9$*

and to provide a *guaranteed error estimate* of this solution. Tools of interval arithmetic, interval analysis and branch-and-bound principles turned out to be an appropriate means for reaching this goal.

After the above short problem history we should mention the purpose of our paper. That is, we intend primarily to report on the steps and techniques used to attain our goal and, secondly

(i) to emphasize the global aspect of the problem, i.e. we have shown with strictly mathematical methods that a unique solution exists in the domain $[0, 10]^9$,

(ii) to give an approximate value for the solution with guaranteed error-bounds,

(iii) to hint at the discrepancy between theoretical knowledge and practical skills. In this matter we were stimulated by the opinion of Wright and

Cutteridge [25] that "mathematical progress has been concentrated away from applications in favor of studies on theoretical convergence properties . . .".

(iv) to hint at the differences between local and global approaches where for the special problem we are reporting we got almost no help from the literature,

(v) observing at which state of the computation gradient information was helpful. Here we were again stimulated by Wright and Cutteridge's [25] discussion about using gradient information or not.

(vi) stating that in the competition between "solving the zero finding problem directly" versus "solving it via a least-squares approach" the latter was found to be inferior.

(vii) emphasizing the role of the brain where we were stimulated by Moore's [16] beautiful sentence that "a computer can be much more powerful than otherwise if the brain of a mathematician is allowed as one of the 'peripherals' " and by Wright and Cutteridge [25] again who said that "whilst experimental methods are being used it is felt that online working is the most convenient medium for this type of work". This led to a type of interactive optimization [we thank one of the anonymous referees for suggesting this term], which will play a central role in Section 5.

Our strategy for attaining our computational goal was to try available deterministic methods which kept the global target of not losing any solution in the domain $[0, 10]^9$ then proving existence and uniqueness of a solution after the solution was localized to a domain of reasonably small width.

Such methods were

(i) interval arithmetical optimization techniques applied to the least-squares version,

(ii) Newton-like and topological techniques for solving (1),

(iii) subdivision methods for controlling the search for solutions over the global domain piece by piece.

Since it turned out that – in contrast to local methods – the treatment of the least-squares approach was inferior to treating (1) directly and since first and second order methods could be applied only in a neighborhood of the solution with radius 10^{-2} our main tools were subdivision techniques. The crucial step was to deemphasize automated subdivision since this process could be improved by concepts like a most promising component function or a most promising bisection direction.

Our strategies were far from being perfect. They were however successful, despite enormous costs. The costs were so high that the computation for this problem is certainly not competitive, but at this time, we know no other method which has been applied to this circuit design problem and which has led to the same guaranteed result of locating exactly one solution in this huge domain, completed with a reliable error estimate.

The outline of the paper is as follows. Section 2 provides supplementary interval tools which together with Ratschek and Voller [24] covers the interval arithmetic background of this paper. In Section 3 two deterministic ways of solving the problem are compared, i.e., the direct zero search vs. the optimization approach with the related least squares problem. In order to speed up the interval computations it is necessary to replace global techniques with local ones as early in the computation as possible. Section 4 explains why this replacement only succeeded towards the end of the computations. Section 5 reports on the proper experimental investigations with subdividing and slicing of the domain and scaling and weighting of the functions. One of the main experiences was that the local scaling techniques failed when they were applied to the global problem. Section 6 gives an overview of the progress and of the main stages of the computations.

2. Interval Tools

The tools we used to obtain a truly global perspective on solving (1) and (3) are based on interval analysis, especially on methods for solving global optimization and global zero search problems (see, for example, the monographs [1, 22, 17, 5]). For a first survey of the area and for further references we refer to [23, 24]. General aspects of branch-and-bound principles can be found in [7]. Although most of the required interval arithmetic background for the paper can be found in [24] we here emphasize a few points which are beyond the scope of that paper.

For the particular problem dealt with here the interval arithmetic was realized on a network of Sparc 1 computers. These adhere to the IEEE standards for floating point arithmetic [8, 9]. These standards allow in particular for 4 rounding modes which can be controlled by software. One of these modes, truncation, is used to efficiently implement machine interval arithmetic via outward rounding of the interval boundaries. Machine interval arithmetic is understood to be any inclusion isotone implementation of interval arithmetic on a computer. Outward rounding means to replace intervals A that are not representable on the computer by larger intervals A_M having machine numbers as boundaries. Hence rounding errors, approximation errors, etc., can be kept under control automatically. Some of the computations were also executed on Atari computers using PASCAL-SC [14].

Let I be the set of real compact intervals. The functional χ: $I \rightarrow [-1, 1]$ is defined as (cf. [20])

$$\chi([a, b]) = \begin{cases} a/b\,, & \text{if } |a| \leq |b|, \quad b \neq 0\,, \\ b/a\,, & \text{if } |a| \geq |b|\,, \quad a \neq 0\,, \\ -1\,, & \text{else.} \end{cases}$$

χ can be interpreted as a measure for the symmetry of an interval w.r.t. the zero point. The extremum cases are $\chi(A) = 1$ saying that $A \neq 0$ (as point interval) is completely asymmetric, and $\chi(A) = -1$ saying that A is completely symmetric.

The range of a function f over a box $X \in I^m$ is denoted by $\Box f(X)$ and the natural interval extension of f over X by $f(X)$. The width of an interval or box X is denoted by $w(X)$.

It is generally believed that computing derivatives is expensive. Using automatic differentiation, however, time and cost savings can be achieved, cf. [15, 19]. In the case of the functions we had to deal with, it was, however, cheapest and best to compute the partial derivatives or their inclusions analytically, since they could be constructed from expressions the values of which were already known from the evaluation of the functions themselves.

The interval Newton method and its refinement, the Hansen and Greenberg [6] realization were the main tools in pursuing our target. Therefore we give a short overview of the m-dimensional interval Newton algorithm.

Consider the problem of finding an x such that $\phi(x) = 0$ for a given function ϕ: $X \to R^m$, $X \in I^m$ and assume that both $\phi(x)$ and the Jacobian matrix $j(x)$ have inclusion functions $\Phi(Y)$ and $J(Y)$ respectively. Then the basic interval Newton method is defined by the following algorithm:

THE INTERVAL NEWTON ALGORITHM

1. *Set* $X^{(0)} := X$.
2. *For* $n = 0, 1, 2, \ldots$
   (*a*) *choose* $x^{(n)} \in X^{(n)}$,
   (*b*) *determine a box* $Z^{(n+1)}$ *enclosing the solution* $Y^{(n+1)}$ *of the linear interval equation with respect to* Y

$$J(X^{(n)})(x^{(n)} - Y) = \Phi(X^{(n)}) , \qquad (6)$$

   that is, $Y^{(n+1)}$ *is the set of all vectors* $y \in R^m$ *for which a real matrix* $A \in J(X^{(n)})$ *and a real vector* $b \in \Phi(X^{(n)})$ *exists such that* $A(x^{(n)} - y) = b$ *holds* (*Note:* $Y^{(n+1)}$ *need not be a box*),
   (*c*) *set* $X^{(n+1)} := Z^{(n+1)} \cap X^{(n)}$.

The following general properties are useful for understanding the application of the algorithm:

1. If a zero, ξ, of ϕ exists in X then $\xi \in X^{(n)}$ for all n.
   This means that no zero is ever lost! This implies:
2. If $X^{(n)}$ is empty for some n then ϕ has no zeros in X.
3. If $Z^{(n+1)}$ is obtained by Gauss–Seidel or by Gauss elimination, then
   (a) if $Z^{(n+1)} \subseteq X^{(n)}$ for some n then ϕ has a zero in $X^{(n)} \subseteq X$,

(b) if $Z^{(n+1)} \subseteq int\, X^{(n)}$ for some n then ϕ has a unique zero in X (*int* means the topological interior).

4. Under certain reasonable conditions one obtains

$$w(X^{(n+1)}) \leqslant \alpha(w(X^{(n)}))^2$$

for some constant $\alpha \geqslant 0$.

In the following we also denote $Z^{(n+1)}$ by $N(X^{(n)})$ and $N(X^0)$ by $N(X)$. These are the Newton iterates prior to executing the intersection (c).

Interval Newton methods are distinguished by the particular choice of the superbox $Z^{(n+1)}$. If the choice does not matter or if it is not specified then the methods are still discussed under the label interval Newton methods. If the choice is specified then the methods are labelled with the particular way of choosing the superbox. The Hansen–Greenberg realization, for instance, contains a preconditioning and a sophisticated sequence of Gauss–Seidel and Gauss eliminations steps for solving (6), see [6, 22].

Both the interval Newton method and the Hansen–Greenberg realization were implemented for solving the problem (1) respectively (3). The interval Newton method was mainly used to prove that a solution close to the solution x_A was the only solution to the equation system in $[0.0, 10.0]^9$ whereas the Hansen–Greenberg realization has better numerical properties and was able to provide sharper bounds for the final solution.

3. Optimization Problem versus Zero Search

It was clear from the beginning and from the experiences reported in the literature that no local method would lead to a guaranteed solution of the problem. Nevertheless, stimulated by the positive results of Cutteridge [2] we first tried a treatment of the problem in a least squares setting. The advantages of this approach seemed to be exclusively restricted to the tricky combination of the local damped Newton–Raphson steps, the conjugate gradient method and the gradient-descent method as mentioned in Section 1. It even turned out that the first phase of our computations were without success because of the use of the least squares approach to the optimization problem! The reason became meanwhile evident: If we remind that the monotonicity test is the main tool for the global optimization access, this results mainly in checking that a box Y of the interior of the domain $X_0 = [0, 10]^9$ does not contain a global minimizer if

$$0 \notin F^{(j)}(Y) \quad \text{for one } j = 1, \dots, 9$$

where $F^{(j)}$ is an inclusion function for $(\partial f)/(\partial x_j)$. The main tool for the global zero search is in proving that boxes Y do not contain a zero of the function

$$g=(\alpha_1 \ldots \alpha_4 \quad \beta_1 \ldots \beta_4 \quad \gamma) \tag{7}$$

via the so-called *midpoint* or *Moore's test*, that is simply

$$0 \notin g_j(Y) \quad \text{for one } j=1,\ldots,9 .$$

Now, f and g are connected by $f=\Sigma_{i=1}^{9} g_i^2$ which implies

$$\frac{\partial}{\partial x_j} f=(g_1 \cdots g_n) \cdot \left(\frac{\partial}{\partial x_j} g_1 \cdots \frac{\partial}{\partial x_j} g_9\right) .$$

Hence, if 0 is not contained in the range of $(\partial f)/(\partial x_j)$ over Y, then $0 \notin \Box g_i(Y)$ for some i. This means logically, that if the monotonicity test can be applied successfully to f (so that Y can be discarded from further processing) then the same success can be obtained using Moore's test w.r.t. g. Always! (The converse holds under certain regularity conditions of the Jacobian matrix of g.)

Further, the computational costs of evaluating is $(\partial f)/(\partial x_j)$ essentially higher than the ones of g_i, which gives another reason for preferring the approach with g. Finally, the excess-widths of $(\partial f(y))/(\partial x_j)$ will be larger than the excess-widths of the g_i's since the partial derivatives of f are arithmetical combinations of the g_i's and their partial derivatives. This means that the monotonicity test had less chance of being successful than the Moore's test.

When we discovered this we continued the computations with the original zero search problem of g over $X_0=[0,10]^9$.

4. Local Approach versus Global Approach

By the local approach we generally mean the application of the interval Newton algorithm to a box $X \subset X_0$, getting a sequence of nested boxes, $(X^{(n)})$, $n=0, 1, \ldots$. We can use the algorithm directly for the zero search of g in X or for the search for stationary points of f.

The local approach to a box $X \subset X_0$ is successful if

(i) $N(X^{(n)})$ is contained in the interior of $X^{(n)}$ for some n. (In this case a unique zero or stationary point, x^*, lies in X if some mild assumptions for $Z^{(n+1)}$ hold, and the nested sequence converges to x^*.)

(ii) $X^{(n)}=\emptyset$ for some n (which means that X contains no solution and can be discarded from further processing).

The local approach to a box is not successful if either (i) or (ii) cannot be reached after a certain number of iterations. This can be made precise when we use the common termination criteria for interval Newton algorithms. We will not elaborate further on this here.

By the global approach to X we mean the subdivision of X iteratively into subboxes Y_k keeping track of them till it can be decided whether Y_k contains a solution (zero or stationary point) or not. The decisions can follow from Moore's test or monotonicity test or from a successful local approach to Y_k. Since the numerical costs of the global approach, which has a branching structure, increase exponentially w.r.t. the dimension of a problem one will try to replace it by the local approach whenever possible.

Hence an effective way to solve global problems like ours is to start with the global approach till the subboxes are sufficiently small so that the local approach can be applied successfully. In the case of our circuit design problem the analysis of the problem was very disappointing: The Jacobian matrix of g (and even more the Hessian of f) was so sensitive that the local approach was only successful when the boxes Y_k had an edge length not larger than 10^{-2}.

Let us keep in mind that the cube X_0 contains 10^{27} cubes of width 10^{-2}. Since the only aid in eliminating larger boxes was finally Moore's test, it was extremely important to develop techniques for an effective subdivision where the aim was to subdivide in such a manner that larger boxes were thrown out as soon as possible by Moore's test. This is reported in the next section.

5. Subdivision Strategies

We start with a simple example to demonstrate the importance of subdivision techniques. Assume we want to prove that the function $h(x) = x(2 - x) - 5/3$ has no zero in the interval $V = [0, 1]$. Considering the natural interval extension $h(V) = [0, 1][1, 2] - 5/3 = [-5/3, 1/3]$ is not helpful since $0 \in h(V)$. But if V is bisected into two subintervals, $V_1 = [0, 1/2]$ and $V_2 = [1/2, 1]$ then the natural interval extensions yield $h(V_1) = [-5/3, -2/3]$ and $h(V_2) = [-7/6, -1/6]$. Since $0 \notin h(V_1)$ and $0 \notin h(V_2)$ neither V_1 nor V_2 contain a zero of h due to the inclusion isotony. Hence $V = V_1 \cup V_2$ contains no zero.

We learn from this example that the (guaranteed) verification that a box contains no zero will lead to a partition of the box into subparts, say Y_k, $k = 1, 2, \ldots$, which might be proven not to contain a zero. The mathematical background of this technique is the phenomenon of the excess width tending to zero if the box width is tending to zero, provided the functions are reasonable and provided that reasonable inclusion functions are chosen. Hence, a subdivision strategy will subdivide the area step by step and check whether such a new subpart contains no zero. If so it can be rejected. Otherwise it will be stored for further subdivisions. A good subdivision strategy will try to discard large subparts as early as possible to keep the numerical costs low. Such a strategy can also be interpreted as a *branch-and-bound* method. We had to deal with several subdivision varieties. *Slicing* means to cut off a thin piece (called *slice*) from the current box *at the edge* normal to a coordinate direction. *Subslicing* means to subdivide a slice into several pieces (*subslices*) of equal size normal to between one and three coordinate directions with the intention of getting a larger number of subparts, usually 10 to 10000. *Bisecting* or *binarily subdividing* means to cut any part of the box into two pieces of equal size normal to a coordinate direction.

Our strategy was

(i) first, to find a slice from the current box that had a reasonable chance of being rejected by (ii) and (iii),
(ii) second, to find a reasonable subslicing (one to three subslice directions, a number of parts in each of these directions)
(iii) to apply a simple bisection procedure, cf. the Slice Elimination Algorithm, to each of the subslices gained by (ii) for a final processing.

Initially, the domain was $X_0 = [0, 10]^9$. Then the solution and verification process was to cut off one slice after the other and to process it using (ii) and (iii) in order to show that no zero was contained in it and thus reject it. Hence, at each phase of the solution process a current box was maintained and it was guaranteed that no solution occurred outside this box in X_0. When a slice had been rejected the current box was shrunk to reflect the elimination of the slice.

A slice of thickness s in the direction i was a slice, Y, with the i-th coordinate axis as cutting direction and $w(Y_i) = s$. The typical thickness of a slice in the cutting direction was 0.1.

The slicing was done until the current box, that contained the solution, had been shrunk so far that the faster local methods like interval Newton variants could be applied.

Let us discuss some details. As the computations progressed one slice was eliminated at a time. In difficult cases the slices had to be dealt with as a set of subslices. These were then eliminated one at a time until the original slice had been eliminated. The slicing process was always done manually whereas the subslicing process was at various times done both manually and automatically.

It was *suspected* that there was no other solution except a solution close to x_A, however, as pointed out earlier, the aim was to prove this assertion.

Each subslice, or in a few favorable cases the slice itself, was treated by a comparatively simple binary subdivision program as described by [10, 11, 22, 13] and others, as described below. This program was used for both the zero finding procedure and the global optimization procedure. This program included routines to either evaluate an inclusion function G of g in (7) or inclusion functions F and $F^{(i)}$ of f in (3) and its partial derivatives.

The outline of this program for the version for (1) is given by the

SLICE ELIMINATION ALGORITHM.

1. Set $Y := X$, the subslice.
2. Initialize list $\mathcal{L} := (Y)$.
3. Choose a coordinate direction ν (depending on various criteria as discussed later).
4. Bisect Y normal to direction ν, getting boxes V_1, V_2 such that $Y = V_1 \cup V_2$.

```
 5. Remove $Y$ from the list $\mathcal{L}$.
 6. For $j = 1, 2$
      (a) For $i = 1, 2, \dots, 9$
            $\alpha$. Calculate $G_i(V_j)$.
            $\beta$. If $G_i(V_j) \not\ni 0$ then go to (c).
      (b) Enter $V_j$ onto the list at the end of the list.
      (c) end (of $j$-loop).
 7. If list is empty then terminate with output:
      • Subslice has no global minimizer.
 8. Denote the last item of the list by $Y$.
 9. Go to 3.
10. End.
```

For the global optimization version the step 6. (a) was replaced by

(a) *If* $0 \notin F(V_j)$ *or* $0 \notin F^{(i)}(V_j)$ *for some i then go to* (*c*).

In the program step 8 the last item of the list is designated for further bisection. This means that a box was subdivided repeatedly until it was shown via 6(a)β. that the current final list box could have no zero in it (or, equivalently, no minimizer in the global optimization version). Then the previous box on the list was considered and so on until the list was exhausted. The strategy of subdividing the last box on the list first resulted in lists that only grew to moderate lengths with between 20 and 40 boxes when the computations were successful.

In step 6(a)β. a box is eliminated if any one component does not contain zero. This elimination step was done by the Moore-test respectively, the monotonicity test or local methods.

In this step it is important to be able to compute the inclusion G of g or F of f as accurately as possible. This was done via the natural interval extensions. Attempts to use mean value forms or other centered forms (see for example [21]) were not successful mainly due to the nature of the system (1), respectively (3), since the computation of the natural interval extension of this system resulted in inclusions that were already close to the range. This meant that the sophisticated techniques described in [21] were not able to improve on the computations.

The choice of bisection direction in step 3 of the basic program is crucial to the performance of the elimination process. For instance the Moore–Skelboe algorithm, as described in [23], bisects the box normal to the direction of maximum box length. This strategy was also tried and the computation time tended to be excessive.

A large number of experimental programs were tested for the elimination of slices. Early on it became clear that a strategy had to be developed for the choice of the next bisection direction in step 3 of the algorithm. Consider for example eliminating domains that cannot contain a zero of the function

$$h(x) = x_1 + 10^{10}x_2 - 1, \quad x \in X = [0, 10]^2$$

using the Slice Elimination Algorithm. If $X = (X_1, X_2)$ it is clear that subdividing X_1 will have negligible effect on the range of values obtained from the natural interval extension whereas subdividing X_2 will eventually result in the elimination of a subbox.

The basic premise was that the coordinate direction of largest width should be taken for the bisection. This was then modified by the use of *weighting factors* in a variety of ways. The different methods for selecting weighting factors were

1. Simple subdivision with no weighting factors where the selection of direction was only depending on the direction of maximum box width. This was briefly discussed above.
2. Bisection with selection of coordinate direction only dependent on edge widths and fixed weighting factors chosen by experience. This approach was used initially, but it failed for slices closer to x_A.
3. Bisection with selection of coordinate direction dependent on edge widths and dynamically calculated weighting factors derived from various parameters such as the Jacobian matrix. Variations of this technique were used extensively to eliminate a large number of slices, see also [12].

The final successful subdivision program for discarding slices without solution that was arrived at at the very end of the whole project consisted of the following steps:

1. Choose the right slice X by hand. (From experience we knew that, at many stages of the project a few slices existed which were easier to treat than the others.)
2. Compute $J_g(X) = (J_{ij})_{i,j=1}^{9}$, that is a natural interval extension of the Jacobian matrix of g over X, or use the Jacobian matrix of a superbox of X.
3. Select so-called *prechosen weighting factors*, λ_j, $j = 1, \ldots, 9$, for the edge lengths of the boxes occurring in the Slice Elimination Algorithm by hand. These factors will influence the bisection direction in Step 3 of the algorithm and remain fixed during the treatment of X. The selection was done by experience mainly by considering the edge lengths of X, the entries J_{ij} and the course of the computational success at adjacent slices.
4. When trying to create a formula for the weighting factors to eliminate the non-programmable experience part, cf. point 3, we first established the concept of a *most promising component function*, say g_i. This component was expected to respond to Moore's test before the other, g_j, $j \neq i$, did. Then the bisection directions for the algorithm could primarily be adapted to the needs of g_i. The most promising function g_i, was deemed to be that component g_j, $j = 1, \ldots 9$ which maximized

$$w[g_j(X)]/(\min\{|\min g_j(X)|, |\max g_j(X)|\} + \epsilon) \tag{8}$$

where ϵ was a small positive constant included to avoid division by zero. The entries $|J_{ij}|$, $j = 1, \ldots, 9$ were then considered as *preliminary weighting factors*.

5. Choose constants $\mathcal{U}$, $\mathcal{L}$, $\mathcal{T}$ (upper, lower, threshold) to scale the preliminary weighting factors as explained below. The *scaled weighting factors* are denoted by $\mathcal{F}_j$, $j = 1, \ldots, 9$.
6. The *final weighting factors* for the boxes Y occurring in the alg. were the numbers $\lambda_j \mathcal{F}_j$, $j = 1, \ldots, 9$. The *weighted widths* for such boxes were defined as

$$ww(Y_j) = \lambda_j \mathcal{F}_j w(Y_j),\ j = 1, \ldots, 9\,.$$

The index ν in step 3 of the alg. was then chosen so that ν maximized $ww(Y_j)$ subject to $j = 1, \ldots, 9$.

7. In order to make the state of the current working area such as a subslice or a subbox of the alg. (in contrast to the current slice) more relevant we frequently split up the computation of the different weighting factors. The prechosen factors were derived from the Jacobian matrix over the slice X, cf. point 2, and remained fixed during the whole elimination procedure for this slice. The same held for the constants $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{T}$. The preliminary and the scaled weighting factors were, however, computed either for each subslice or sometimes for each box occurring in the algorithm. This updating procedure diminished the number of iterations essentially but, especially in the second mentioned case, the numerical costs for computing $J_g(X)$ for each box Y were too high.

Let us supplement a few details of these points: Formula (8) expressed our thought that it was reasonable to choose that component which had one end-point of $g_i(X)$ closest to zero relative to the size of $g_i(X)$. In this manner it could be argued that after a series of bisections the most promising component function would have a good chance to reject a great portion of the subboxes generated by the algorithm.

In point 5 the Jacobian entries were scaled depending on the parameters $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{T}$ in the following manner: Suppose the index of the most promising component function was i and $\mathcal{B} = \max_{ij=1,\ldots,9}|J_{ij}|$. Let $c_j = \chi(J_{ij})$. Then the *scaled weighting factors* were defined as

$$\mathcal{F}_j = |J_{ij}|(1 + c_j)\mathcal{U}/\mathcal{B},\ j = 1, \ldots, 9\,.$$

Values of $\mathcal{F}_j < \mathcal{T}$ were however excluded and replaced by $\mathcal{L}$. The parameter $\mathcal{U}$ was set to 200 by experience. This scaling guaranteed that the coordinate directions with absolutely large Jacobian entries would be favored first. A parameter value $\mathcal{U} = 200$ corresponded roughly to 7 or 8 bisections in one direction if the values of $\mathcal{L}$ and $\mathcal{T}$ were about 1. The parameters $\mathcal{L}$ and $\mathcal{T}$

ensured that if the choice of the most promising component function was wrong then all was not lost. The reason is that the less favored directions will also be considered after 7 to 8 bisections (also depending on the edge widths).

We also had to struggle with the observation that the algorithm was very sensitive to changes in the parameters $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{T}$. Typically, with one choice of parameters a given slice would be eliminated in 50,000 iterations whereas a slightly different choice would require 10,000,000 iterations for the same slice and in extreme cases the iterations did not terminate within a reasonable time.

An experiment was made varying the parameter $\mathcal{U}$ for the choice of $\mathcal{L}$ and $\mathcal{T}$ as described by the following table

	$\mathcal{L}$	$\mathcal{T}$
A	1.0	1.0
B	1.0	1.5
C	1.0	2.0
D	5.0	1.0

With the box $[5.0, 10.0] \times [0.0, 10.0]^8$ and the input parameters given in the table the algorithm gave the results plotted in Figure 1 which showed the number of bisections needed till the box had been eliminated.

In difficult cases where there was no guidance as to a choice of parameters and bisection directions we bisected a sample box in each direction and compared the

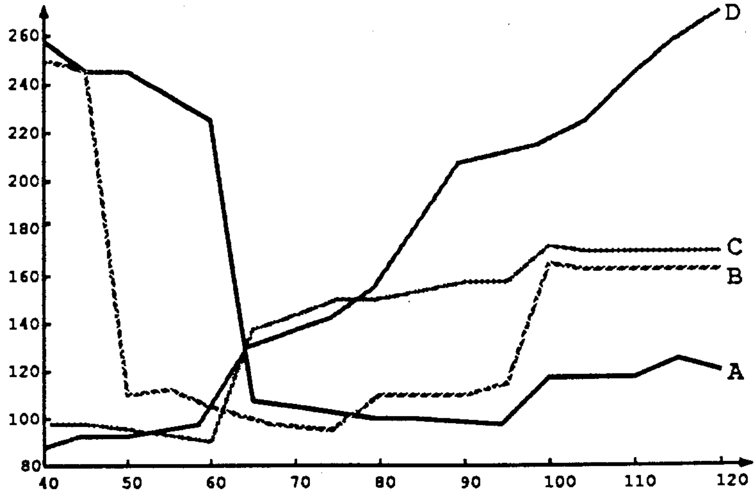

Fig. 1. Plot of number of iterations depending on $\mathcal{U}$.

18 box halves w.r.t. largest range reduction. This gave us an idea of which bisection directions could lead to success and installed the prechosen weighting factors and $\mathcal{U}$, $\mathcal{L}$, $\mathcal{T}$ accordingly.

Other unsuccessful cases could be dealt with by varying the formulas for picking out the most promising component function. Such formulas replacing (8) were, for instance,

$$2/w_j - 1/a_j \text{ } \mathit{if} \text{ } a_j,\ w_j \neq 0, \quad \mathit{or} \quad 2/\sqrt{w_j} - \sqrt{w_j}/a_j \text{ } \mathit{if} \text{ } a_j,\ w_j \neq 0\,,$$

with $w_j = w[g_j(X)]$ and $a_j = \max\{|u|\colon u \in g_i(X)\}$.

Even though our subdivision strategy was very cumbersome, it had the advantage of working which finally led to success.

6. The Course of the Computations

In this section we describe the global order of the numerical steps of the whole problem solving enriched by some further computational details.

A. Variable elimination. In order to reduce the dimension of the problem, Cutteridge [2] used the ninth equation,

$$\gamma(x) = x_1x_3 - x_2x_4 = 0$$

to eliminate the variable x_4 so that only eight equations in eight variables were to be solved. A direct translation of such a substitution into interval arithmetic would not be wise because of the increase of width of the eliminated interval variable. Hence we kept nine equations and nine-variables. But, whenever a subslice or subbox Y in Slice Elimination Algorithm was going to be used, parts of Y_i, $i = 1, \ldots, 4$ were removed that could not satisfy the equation $\gamma = 0$. Moore's test or further bisections were then applied to Y' instead of Y where

$$\begin{aligned}
&Y_j' = Y_j \quad \text{for} \quad j = 5, \ldots, 9\,,\\
&Y_4' = (Y_1Y_3/Y_2) \cap Y_4\,,\\
&Y_3' = (Y_2Y_4'/Y_1) \cap Y_3\,,\\
&Y_2' = (Y_1Y_3'/Y_4') \cap Y_2\,,\\
&Y_1' = (Y_2'Y_4'/Y_3') \cap Y_1\,.
\end{aligned}$$

These equations occur if variable eliminations based on $x_1x_3 - x_2x_4 = 0$ would have been executed. That is, no variable is really eliminated but those box shrinkings are considered which would have been caused by the eliminations.

This procedure had to be modified when division through zero intervals occurred. We drop the details. If any of the intersections were empty then the box or subslice could be eliminated since it could not contain a zero.

B. Existence and uniqueness test. In order to localize the suspected zero and to prove that no second zero is available in X_0, we had to make a lot of experiments with boxes around or near X_A applying the interval Newton algorithm in order to find out boxes in which just one zero exists. To be consistent with the existing interval arithmetical theory it was necessary to use some specific methods for determining the interval hull Z_{n+1} of the system (6), for instance, the interval Gauss algorithm (cf. [1]). We finally were successful with the box

$$X_I = \begin{pmatrix} 0.8990229964 & 0.9009769558 \\ 0.4490229785 & 0.4509769976 \\ 0.9990229606 & 1.000979900 \\ 1.999019980 & 2.000979900 \\ 7.999019622 & 8.000979423 \\ 7.999019622 & 8.000979423 \\ 4.999019622 & 5.000979900 \\ 0.9990229606 & 1.000979900 \\ 1.999019980 & 2.000979900 \end{pmatrix}$$

and got

$$N(X_I) = \begin{pmatrix} 0.899999499320 & 0.900000214576 \\ 0.449942409992 & 0.450029253959 \\ 0.999914228916 & 1.000097632408 \\ 1.999960184097 & 2.000184535980 \\ 7.999773979187 & 8.000171661376 \\ 7.999220371246 & 8.000111579895 \\ 4.999865055084 & 5.000213146209 \\ 0.999968469142 & 1.000006437301 \\ 1.999950766563 & 2.000164270401 \end{pmatrix}.$$

Since $N(X_I)$ is contained in the interior of X_I existence as well as uniqueness of a solution in X_I was proven. Hence our efforts were concentrated to prove that no further zero existed in X_0.

C. Rejection of $X_0 \backslash X_I$. We wanted to show that $X_0 \backslash X_I$ contained no zeros of g. Then, together with *B.*, it was proven that X_0 contains exactly one zero of g, and that it lied in $N(X_I)$.

After having finished *B.* and having condemned the least-squares approach, we proceeded as follows:

1. Set $X := X_0$.
2. Choose an edge slice Y of X with $Y \backslash X_I \neq \emptyset$ for getting processed.
3. Choose between the global approach (slice elimination algorithm with Moore's test) or the local approach (interval Newton method and Hansen–Greenberg variant) or combinations for the elimination of Y.
4. In case of the global approach, choose the subslicing and bisecting mode for Y (Section 5).
5. If within a prescribed amount of steps it can be proven that Y contains no zero, then

(a) Set $X := X \backslash int\, Y$, where $int\, Y$ denotes the interior of Y,
(b) If $X \backslash X_I \neq \emptyset$ then go to 2
else STOP (since the aim is reached).

6. Go to 4. or 3. or 2. modifying or rejecting the former parameters or choices (that means, try it again!).

When we finally reached the STOP of step 5, the box X_0 was shrunk to

$$X^* = \begin{pmatrix} 0.899999 & 0.900000 \\ 0.449962 & 0.450004 \\ 0.999955 & 1.00005 \\ 2.00001 & 2.00009 \\ 7.99987 & 8.00008 \\ 7.99953 & 7.99985 \\ 4.99994 & 5.00011 \\ 0.999978 & 0.999997 \\ 2.00000 & 2.00010 \end{pmatrix} \subset X_I \,. \tag{9}$$

Hence it was proven that the unique zero of g in X_0 lies in the box

$$X^* \cap N(X_I) \,.$$

D. Computational costs. One word has to be said about the number of function evaluations (f.e.) we needed. There is only one comment: There were too many f.e., in order to be proud of reaching the goal and to be really successful, since it was in the order of billions. Typically the bisection limit, often reached, was 5×10^6 for each slice. However, it was an experimental computation where number of f.e. is a dubious concept: If a computation had to be executed, we started with a certain strategy and needed, say n_1 f.e. Then we changed the parameter, observed from the beginning that the number would be too high, and stopped with n_2 f.e. The third time we needed $n_3 \ll n_1$ f.e. The three tests were helpful for getting good parameters for the processing of the next box. But, which is the real number of f.e. to be reported in this part of the computation? Is it n_1 or n_3, or $n_1 + n_3$ or $n_1 + n_2 + n_3$? Each of these answers had its right, also n_3, since everybody could repeat the computation with n_3 f.e. since the necessary parameters are then already known.

7. Conclusion

In this paper a numerically very difficult system of equations arising from a circuit modeling problem has been solved in the global domain $[0, 10]^9$. Success was achieved using a combination of interval methods, subdivision and branch-and-bound strategies where the choice of bisection direction was made based on a scaling strategy developed as the computations progressed. Although the computation was successful it was extremely costly in terms of machine cycles. It is therefore recommended that such methods for solving unstable systems should be further investigated with the aim of reducing the computations. These methods would be based on improved selection criteria for the subdivision process.

References

1. Alefeld, G. and Herzberger, J. (1983), *Introduction to Interval Computations*, Academic Press, New York.
2. Cutteridge, O. P. D. (1974), Powerful 2-part program for solution of nonlinear simultaneous equations, *Electronics Letters* **10**, 182–184.
3. Dimmer, P. R. and Cutteridge, O. P. D. (1980), Second derivative Gauss–Newton-based method for solving nonlinear simultaneous equations, *IEE Proc.* **127**, 278–283
4. Ebers, J. J. and Moll, J. L. (1954), Large-scale behavior of junction transistors, *IEE Proc.* **42**, 1761–1772.
5. Hansen, E. R. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.
6. Hansen, E. R. and Greenberg, R. I. (1983), An interval Newton method, *Applied Math. and Comp.* **12**, 89–98.
7. Horst, R. and Tuy, H. (1990), *Global Optimization*, Springer-Verlag, Berlin.
8. IEEE (1985), IEEE standard for binary floating-point arithmetic. IEEE Standard 754–1985, IEEE, New York.
9. IEEE (1987), IEEE standard for radix-independent floating-point arithmetic. IEEE Standard 854–1987, IEEE, New York.
10. Kearfott, B. (1979), An efficient degree-computation method for a generalized method of bisection, *Numerische Mathematik* **32**, 109–127.
11. Kearfott, B. (1987), Some tests of generalized bisection, *ACM Trans. Math. Software* **13**, 197–220.
12. Kearfott, B. (1987), Abstract generalized bisection and a cost bound, *Math. of Comput.* **49**, 187–202.
13. Kearfott, B. (1990), INTBIS, a portable interval Newton/bisection package, *ACM. Trans. Math. Software* **16**, 152–157.
14. Kulisch, U. (ed.) (1986), *PASCAL-SC Manual and System Disks*, Wiley-Teubner Series in Computer Science, Stuttgart.
15. Moore, R. E. (1979), *Methods and Applications of Interval Analysis*, SIAM, Philadelphia.
16. Moore, R. E. (1990), Interval tools for computer aided proofs in analysis, in *Computer Aided Proofs in Analysis*, ed. K. R. Meyer and D. S. Schmidt, IMA Series, vol. 28, Springer-Verlag, Berlin.
17. Neumaier, A. (1990), *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge.
18. Price, W. L. (1978), A controlled random search procedure for global optimization, in *Towards Global Optimization* 2, ed. L. C. W. Dixon and G. P. Szegö, North-Holland, Amsterdam, pp. 71–84.
19. Rall, L. B. (1981), *Automatic Differentiation*, Springer-Verlag, Berlin.
20. Ratschek, H. (1975), Nichtnumerische Aspekte der Intervallarithmetik, in *Interval Mathematics*, ed. by K. Nickel, Springer-Verlag, Berlin, pp. 48–74.
21. Ratschek, H. and Rokne, J. (1984), *Computer Methods for the Range of Functions*, Ellis Horwood, Chichester.
22. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester.
23. Ratschek, H. and Rokne, J. (1991), Interval tools for global optimization, *Computers and Mathematics with Applications* **21**, 41–50.
24. Ratschek, H. and Voller, R. (1991), What can interval analysis do for global optimization? *Journal of Global Optimization* **9**, 111–130.
25. Wright, D. J. and Cutteridge, O. P. D. (1976), Applied optimization and circuit design, *Computer Aided Design* **8**, 70–76.