Journal of Global Optimization (2005) 32: 35–63
DOI 10.1007/s10898-004-5906-2

An Improved Interval Global Optimization Algorithm Using Higher-order Inclusion Function Forms

P.S.V. NATARAJ and K. KOTECHA
Systems and Control Engineering Group, Room 114, ACRE Building, Indian Institute of Technology, Bombay 400076, India (e-mail: Nataraj@ee.iitb.ac.in)

(Received 8 September 2003; accepted in revised form 13 March 2004)

Abstract. We propose an improved algorithm for unconstrained global optimization in the framework of the Moore–Skelboe algorithm of interval analysis (H. Ratschek and J. Rokne, New computer methods for global optimization, Wiley, New York, 1988). The proposed algorithm is an improvement over the one recently proposed in P.S.V. Nataraj and K. Kotecha, (J. Global Optimization, 24 (2002) 417). A novel and powerful feature of the proposed algorithm is that it uses a variety of inclusion function forms for the objective function – the simple natural inclusion, the Taylor model (M. Berz and G. Hoffstatter, Reliable Computing, 4 (1998) 83), and the combined Taylor–Bernstein form (P.S.V. Nataraj and K. Kotecha, Reliable Computing, in press). Several improvements are also proposed for the combined Taylor–Bernstein form. The performance of the proposed algorithm is numerically tested and compared with those of existing algorithms on 11 benchmark examples. The results of the tests show the proposed algorithm to be overall considerably superior to the rest, in terms of the various performance metrics chosen for comparison.

1. Introduction

Let $\Re$ be the set of reals, $\mathbf{X} \subseteq \Re^l$ be a right parallelepiped parallel to the axes (also called as a box), and let $\bar{f}(\mathbf{X})$ denote the set of all values of an arbitrary function $f : \mathbf{X} \to \Re$ on $\mathbf{X}$. Let $I(\mathbf{X})$ be the set of all boxes contained in $\mathbf{X}$. Let the width of an interval $\mathbf{X}$ be defined as $w(\mathbf{X}) := \max \mathbf{X} - \min \mathbf{X}$ if $\mathbf{X} \in I(\Re)$, and as $w(\mathbf{X}) := \max\{w(\mathbf{X}_1), \ldots, w(\mathbf{X}_l)\}$, if $\mathbf{X} \in I(\Re^l)$.

DEFINITION 1.1 [23]. A function $F : I(\mathbf{X}) \to I(\Re)$ is said to be an inclusion function for f, if

$$\bar{f}(\mathbf{Y}) \subseteq F(\mathbf{Y}) \quad \text{for all } \mathbf{Y} \in I(\mathbf{X}).$$

DEFINITION 1.2 [23]. An inclusion function F for f is said to have convergence order α, if

$$w(F(\mathbf{Y})) - w(\bar{f}(\mathbf{Y})) \leqslant L w(\mathbf{Y})^{\alpha} \quad \text{for all } \mathbf{Y} \in I(\mathbf{X}),$$

where L and α are some positive constants.

We consider the global optimization problem of determining arbitrarily good lower bounds for the minimum of $\bar{f}(\mathbf{X})$. Many algorithms based on interval analysis (IA) are available to solve the global optimization problem (see, e.g. [8, 11, 24] and the references cited therein). IA methods are usually based on branch and bound techniques, i.e. they start from an initial box $\mathbf{X}$, subdivide $\mathbf{X}$ and store the subboxes in a list, discarding subboxes which are guaranteed not to contain a global minimizer until the desired accuracy in terms of the width of the intervals in the list is achieved. A basic branch and bound algorithm of IA is the so-called Moore–Skelboe (MS) algorithm [24]. Although MS algorithm is reliable, it is usually somewhat slow to converge in 'difficult' problems, when inclusion functions of first and sometimes even second orders are used for the objective function f. Faster convergence could possibly be obtained with inclusion functions of order $\alpha > 2$ (which we shall refer to as *higher*-order inclusion functions).

Recently, an interval global optimization algorithm TBMS based on a higher-order inclusion function was proposed [20]. Algorithm TBMS uses the improved Taylor–Bernstein (TB) form F_{TB} as a higher-order inclusion function. F_{TB} is constructed using Bernstein polynomials for bounding the range of the polynomial obtained from the Taylor form of f. In order to make it more effective in practice, F_{TB} is constructed differently from that of Lin and Rokne's (LR) TB form F_{LR} [13]. Performance comparisons between algorithms MS (based on the natural inclusion function), TMS (based on the Taylor model [2] as inclusion function), and TBMS show TBMS algorithm to be the most effective one in terms of iterations, space-complexity, and speed.

On the other hand, in typical interval global optimization applications, the algorithm starts with a large initial domain, and reduces it eventually to small enough solution boxes through domain splitting or subdivision techniques. However, in such cases it may become difficult to compute F_{LR} and F_{TB} over the entire range of domain widths, because of large memory and/or time requirements. An appropriate combination of F_{LR} and F_{TB}, called the *combined* TB form F_{CTB}, can be useful in these situations [21]. Numerical tests reported in [21] showed the combined form F_{CTB} to be more effective than either F_{LR} or F_{TB} over the entire range of domain widths. These tests also revealed another finding: the simple natural inclusion form can sometimes produce much tighter range enclosures than the sophisticated Taylor and TB forms, even for small domain widths [21].

Motivated by these recent findings, in this paper we present an improved interval global algorithm using higher-order inclusion function forms. The proposed algorithm improves upon the existing TBMS algorithm [20] in the following ways:

- It uses a *variety* of inclusion function forms for computing the range enclosure of objective function *f*:
  - the simple natural inclusion form [17],
  - the Taylor model [2], and
  - the combined TB form [21] (the form is actually modified and used to produce possibly tighter enclosures of the function minimum).
- It uses more effective cut-off test and termination conditions in the branch and bound part of the algorithm.
- It uses a new polynomial bounder based on the Bernstein form, with the following improvements:
  - A simplified vertex condition check, to find only the minimum over a given Bernstein patch.
  - A monotonicity test for discarding Bernstein patches where surely no global minimizer can lie.
  - A cut-off test for discarding Bernstein patches where surely no global minimizer can lie.
  - An improved direction-selection strategy for subdivision of Bernstein patches.

The performance of the proposed algorithm is compared with those of algorithms MS, TMS, and TBMS on 11 benchmark examples.

The rest of this paper is organized as follows. In Sections 2 and 3, we give the essentials of the various inclusion forms and different interval optimization algorithms considered in this work. In Section 4, we detail the various algorithmic improvements, leading to the proposed algorithm in Section 5. In Section 6, we numerically test and compare the performance of the proposed algorithm with those of algorithms MS, TMS, and TBMS, and discuss the obtained test results. We conclude the work in Section 7.

2. Taylor–Bernstein Inclusion Function Forms

2.1. THE BERNSTEIN FORM

The Bernstein form has established itself as an important tool for finding bounds on the range of multivariate polynomials (see, for instance, [7, 25] and the references cited therein). An introduction to the Bernstein form is given in the book [23]. The salient features of the Bernstein form approach are:

(1) The computation of the bounds conveys the information about the sharpness of these bounds.
(2) The approach avoids functional evaluations which might be costly if the degree of the polynomial is high.

(3) When bisecting a box and applying the Bernstein form to one of the two subboxes to get an enclosure for the range over this subbox, we obtain without any extra cost an enclosure for the range over the other subbox.
(4) For sufficiently small boxes the Bernstein form gives the range.

In this section, we follow the notation used in Ref. [7]. Let l be the number of variables and $\mathbf{x} = (x_1, \ldots, x_l) \in \Re^l$. A multi-index I is an ordered l-tuple of non-negative integers $I = (i_1, \ldots, i_l)$. For two given multi-indices I, N, we write $I \leqslant N$ if $0 \leq i_k \leq n_k$, $k = 1, \ldots, l$. With $I = (i_1, \ldots, i_{r-1}, i_r, i_{r+1}, \ldots, i_l)$ we associate index $I_{r,k}$ given by $I_{r,k} = (i_1, \ldots, i_{r-1}, i_r + k, i_{r+1}, \ldots, i_l)$ where $0 \leq i_r + k \leq n_r$. Also, we write

$$\binom{N}{I} \quad \text{for} \quad \binom{n_1}{i_1} \cdots \binom{n_l}{i_l}.$$

We can expand a given multivariate polynomial into Bernstein polynomials to obtain bounds for its range over an l-dimensional box $\mathbf{X}$. Without loss of generality, consider the unit box $\mathbf{U} = [0,1]^l$ since any nonempty box $\mathbf{X}$ of $\Re^l$ can be mapped affinely onto this box.

Let $p(\mathbf{x})$ be a multivariate polynomial in l-variables with real coefficients. Denote by $N = (n_1, \ldots, n_l)$ the tuple of maximum degrees so that n_k is the maximum degree of x_k in $p(\mathbf{x})$ for $k = 1, \ldots, l$. Denote by $S = \{I : I \leq N\}$ the set containing all the tuples from $\Re^l$ which are 'smaller than or equal to' the tuple N of maximum degrees. Then, we can write an arbitrary l-variate polynomial p in the form

$$p(\mathbf{x}) = \sum_{I \in S} a_I \mathbf{x}^I, \quad \mathbf{x} \in \Re^l, \tag{1}$$

where for $\mathbf{x} = (x_1, \ldots, x_l) \in \Re^l$ we set $\mathbf{x}^I = x_1^{i_1} x_2^{i_2} \ldots x_l^{i_l}$, where $a_I \in \Re$ represents the corresponding coefficient to each $x^I \in \Re^l$. We refer to N as the degree of p. The Ith Bernstein polynomial of degree N is defined as

$$B_I^N(\mathbf{x}) = B_{i_1}^{n_1}(x_1) \cdots B_{i_l}^{n_l}(x_l), \quad \mathbf{x} \in \Re^l,$$

where for $i_j = 0, \ldots, n_j$, $j = 1, \ldots, l$

$$B_{i_j}^{n_j}(x_j) = \binom{n_j}{i_j} x_j^{i_j} (1 - x_j)^{n_j - i_j}.$$

The Bernstein coefficients $b_I(\mathbf{U})$ of p over the unit box $\mathbf{U}$ are given by

$$b_I(\mathbf{U}) = \sum_{J \leqslant I} \frac{\binom{I}{J}}{\binom{N}{J}} a_J, \quad I \in S.$$

Thus, the Bernstein form of a multivariate polynomial p is defined by

$$p(\mathbf{x}) = \sum_{I \in S} b_I(\mathbf{U}) B_I^N(\mathbf{x}).$$

The Bernstein coefficients are collected in an array $B(\mathbf{U}) = (b_I(\mathbf{U}))_{I \in S}$, called a *patch*. Based on the above, an algorithm for finding a patch of Bernstein coefficients is given in Ref. [20].

The following result describes the range enclosure property of the Bernstein coefficients.

LEMMA 2.1 [3]. *Let p be a polynomial of degree N, and let $\bar{p}(\mathbf{X})$ denote the range of p on the given domain $\mathbf{X}$. Then, the following property holds for a patch $B(\mathbf{U})$ of Bernstein coefficients*:

$$\bar{p}(\mathbf{X}) \subseteq [\min\ B(\mathbf{U}), \max\ B(\mathbf{U})].$$

We can find an enclosure of the range of the multivariate polynomial p on $\mathbf{X}$ by transforming the polynomial into Bernstein form. Then, by Lemma 2.1, the coefficients of the expansion in the Bernstein form provide lower and upper bounds for the range. The obtained range enclosure can be further improved either by degree elevation of the Bernstein polynomial or by subdivision. The subdivision strategy is generally more efficient than the degree elevation strategy [5] and is therefore preferred.

Let $\mathbf{D}$ be any subbox of $\mathbf{U}$ generated by bisection, and suppose the patch $B(\mathbf{D})$ has been already computed. Further suppose $\mathbf{D}$ is bisected along the rth component direction $(1 \leqslant r \leqslant l)$ to produce two further subboxes $\mathbf{D}_A$ and $\mathbf{D}_B$ given by

$$\mathbf{D}_A = [\underline{d}_1, \bar{d}_1] \times \cdots \times [\underline{d}_r, m(d_r)] \times \cdots \times [\underline{d}_l, \bar{d}_l],$$
$$\mathbf{D}_B = [\underline{d}_1, \bar{d}_1] \times \cdots \times [m(d_r), \bar{d}_r] \times \cdots \times [\underline{d}_l, \bar{d}_l].$$

Then, the patches $B(\mathbf{D}_A)$ and $B(\mathbf{D}_B)$ can be obtained from $B(\mathbf{D})$ by executing Algorithm Subdivision given in Ref. [20].

The following result gives a condition called the *vertex condition*, which can be used to verify if the enclosure given by the Bernstein coefficients is the range.

LEMMA 2.2 [3]. *Let p be a polynomial of degree N. Let $B(\mathbf{U})$ be a patch on $\mathbf{U}$. Then*,

$$\bar{p}(\mathbf{U}) = [\min\ B(\mathbf{U}), \max\ B(\mathbf{U})]$$
$$\Leftrightarrow \min\ B(\mathbf{U}) \textit{ resp. } \max\ B(\mathbf{U}) \textit{ occurs at some } I \in S_0,$$

where S_0 is a special subset of the index set S defined by

$$S_0 = \{0, n_1\} \times \cdots \times \{0, n_l\}.$$

The above vertex condition also holds for any subbox $\mathbf{D} \subseteq \mathbf{U}$ (see [15]).

The vertex condition is said to be satisfied within a given machine precision μ, if

$$\min_{S_0} B(\mathbf{U}) - \min B(\mathbf{U}) \leq \mu \quad \text{and} \quad \max B(\mathbf{U}) - \max_{S_0} B(\mathbf{U}) \leq \mu.$$

By repeated subdivision, the bounds for the range of the given polynomial over a box can be improved until they are accurate to the given machine precision, i.e., till for every subdivision the vertex condition is satisfied within machine precision. The proof of this statement readily follows from the results in Ref. [15]. Note that the application of the vertex condition often considerably speeds up the process over the one involving just subdivision, because any subdivision on which the vertex condition is satisfied within machine precision need not be further subdivided. Further speedups can be obtained by using additional pruning criteria (cf. [15]).

This leads to the following algorithm for computing the range[1] $\bar{p}(\mathbf{X})$ of polynomial p on $\mathbf{X}$ [20].

2.2.1. *Bounder Algorithm.* $\bar{p}(\mathbf{X}) = \text{Bounder}(\mathbf{X}, a_I)$

Inputs: A box $\mathbf{X}$, a polynomial p as in (1) of degree N in l-variables and having coefficients a_I.
Outputs: The range $\bar{p}(\mathbf{X})$.
BEGIN Algorithm

```
(1) (Compute patch $B(\mathbf{U})$) Execute Algorithm Patch (cf. [20])
        $B(\mathbf{U}) = \text{Patch}(\mathbf{X}, a_I)$.
(2) (Initialize lists) Set $\mathcal{L} \leftarrow \{(\mathbf{U}, B(\mathbf{U}))\}$, $\mathcal{L}^{\text{sol}} \leftarrow \{\}$.
(3) (Select item for processing) If $\mathcal{L}$ is empty, go to step 7. Otherwise, pick
    the first item from $\mathcal{L}$, denote it as $(\mathbf{D}, B(\mathbf{D}))$, and delete the item entry
    from $\mathcal{L}$.
(4) (Check vertex is met within $\mu$ on patch) If $(\mathbf{D}, B(\mathbf{D}))$ satisfies the vertex
    condition within given machine precision $\mu$, enter the item in list $\mathcal{L}^{\text{sol}}$
    and return to previous step.
(5) (Subdivide and find new patches) Execute Algorithm Subdivision (cf.
    [20])
        $[B(\mathbf{D}_A), B(\mathbf{D}_B), \mathbf{D}_A, \mathbf{D}_B] = \text{SD}(\mathbf{D}, B(\mathbf{D}), r)$,
    where, $r$ is chosen to vary cyclically[2] from 1 to $l$.
```

[1]Henceforth, unless otherwise specified, when we refer to the range $\bar{p}(X)$ of the polynomial p on X, we mean the computed range - i.e., an enclosure of the polynomial range i.e. accurate to the given machine precision.

[2]That is, r varies starting from 1 to l, and then again from 1 to l, and so on. Besides cyclical, other strategies for subdivision exist, and their efficiency investigated in Ref. [6].

(6) (Add new entries to list) Enter the new items $(\mathbf{D}_A, B(\mathbf{D}_A))$ and $(\mathbf{D}_B, B(\mathbf{D}_B))$ at end of list $\mathcal{L}$, and return to step 3.
(7) (Compute the polynomial range) Compute the range $\bar{p}(\mathbf{X})$ as the minimum to maximum over all the second entries of the items present in list $\mathcal{L}^{\text{sol}}$.
(8) RETURN $\bar{p}(\mathbf{X})$.
END Algorithm

2.2. THE TAYLOR FORM

In this section, we first introduce some further notation as in Ref. [23]. Let

$$\lambda = \{\lambda_1, \ldots, \lambda_l\}, \quad |\lambda| = \lambda_1 + \cdots + \lambda_l, \quad \lambda! = \lambda_1! \cdots \lambda_l!,$$

$$D^\lambda f(x) = \frac{\partial^{\lambda_1 + \cdots + \lambda_l} f(x)}{\partial x_1^{\lambda_1} \cdots \partial x_l^{\lambda_l}}. \tag{2}$$

Let $I(\mathbf{X})$ be the set of all boxes contained in $\mathbf{X}$. Let the width of an interval $\mathbf{X}$ be defined as $w(\mathbf{X}) = \max \mathbf{X} - \min \mathbf{X}$ if $\mathbf{X} \in I(\Re)$, and as $w(\mathbf{X}) = \max\{w(\mathbf{X}_1), \ldots, w(\mathbf{X}_l)\}$, if $\mathbf{X} \in I(\Re^l)$. Let the midpoint of an interval $\mathbf{X}$ be defined as $m(\mathbf{X}) = (\min \mathbf{X} + \max \mathbf{X})/2$ if $\mathbf{X} \in I(\Re)$, and as $m(\mathbf{X}) = \{m(\mathbf{X}_1), \ldots, m(\mathbf{X}_l)\}$, if $\mathbf{X} \in I(\Re^l)$. Let $\bar{f}(\mathbf{X})$ denote the range of f on $\mathbf{X}$. A function F: $I(\mathbf{X}) \to I(\Re)$ is an inclusion function for f, if $\bar{f}(\mathbf{Y}) \subseteq F(\mathbf{Y})$ for all $\mathbf{Y} \in I(\mathbf{X})$. An inclusion function F for f is said to have convergence order α, if $w(F(\mathbf{Y})) - w(\bar{f}(\mathbf{Y})) \leq Lw(\mathbf{Y})^\alpha$ for all $\mathbf{Y} \in I(\mathbf{X})$, where L and α are positive constants.

Let $f: \mathbf{X} \to \Re$ be a function that is $m+1$ times differentiable on $\mathbf{X}$. Then, the Taylor expansion of f of order m is given as

$$f(\mathbf{x}) = \underbrace{f(\mathbf{c}) + \sum_{|\lambda|=1}^{m} \frac{D^\lambda f(c)}{\lambda!} (\mathbf{x} - \mathbf{c})^\lambda}_{p(\mathbf{x})} + \underbrace{\sum_{|\lambda|=m+1} \frac{f^{(\lambda)}(\xi)}{\lambda!} (\mathbf{x} - \mathbf{c})^{m+1}}_{r(\mathbf{x})}, \quad \mathbf{x} \in \mathbf{X}, \tag{3}$$

where $\mathbf{c} = m(\mathbf{X})$ and $\boldsymbol{\xi} \in \mathbf{X}$. We call $p(\mathbf{x})$ the polynomial part and $r(\mathbf{x})$ the remainder part of the Taylor expansion.

Assume an inclusion function of $(m+1)$th derivative of f exists and is bounded, and furthermore that it has the isotonicity property [23]. Then, the corresponding Taylor form of order m, denoted by F_{Taylor}, can be expressed as [13]:

$$F_{\text{Taylor}}(\mathbf{X}) = \bar{p}(\mathbf{X}) + R(\mathbf{X}) \tag{4}$$

where $\bar{p}(\mathbf{X})$ is the (exact) range of the polynomial part $p(\mathbf{x})$ on $\mathbf{X}$, and $R(\mathbf{X})$ is any inclusion for the range of the remainder part $r(\mathbf{x})$ on $\mathbf{X}$. Lin and Rokne [13] show that the Taylor form has convergence order $m+1$ (the proof as given in Ref. [13] is with $R(\mathbf{X})$ as the natural inclusion function; however, the proof also holds with $R(\mathbf{X})$ as any inclusion function).

REMARK 2.1. For obtaining the required convergence order, it is not necessary to compute the range of the polynomial part of the Taylor form (over a given box) exactly – it is sufficient to compute an enclosure of the same to an accuracy of $v = \mathrm{O}(w(\mathbf{X})^{m+1})$. This latter accuracy can in turn be set to the given machine precision, as rounding errors dominate for accuracies less than the machine precision, i.e., for $v < \mu$.

2.3. THE TAYLOR–BERNSTEIN FORM OF LIN–ROKNE

The Taylor form provides an enclosure for the range of f over $\mathbf{X}$ with convergence order $m+1$. However, it requires the computation of the range of a multivariate polynomial $\bar{p}(\mathbf{X})$. Lin and Rokne [13] proposed an algorithm that uses Bernstein form to find a (generally non-sharp) enclosure of $\bar{p}(\mathbf{X})$, so that the resulting TB form still possesses the property of $m+1$ convergence order.

Below we give the Lin and Rokne algorithm for finding an enclosure of the range of f on $\mathbf{X}$. Note that this algorithm uses the Taylor form of order m and Bernstein polynomials of sufficiently high degree N' given by (6), and that a generally non-sharp enclosure of the range of the polynomial part p of Taylor expansion is computed and used.

2.3.1. *LR Algorithm* [13]. $F_{\mathrm{LR}}(\mathbf{X}) = \mathrm{LR}(\mathbf{X}, f, m)$

Inputs: The box $\mathbf{X}$, an expression for the function f, and the order m of Taylor form to be used.

Output: An enclosure $F_{\mathrm{LR}}(\mathbf{X})$ of the range of f on $\mathbf{X}$.

(1) For the given function f, compute the coefficients of p in (3) and also the remainder interval $R(\mathbf{X})$. This may be done automatically on a computer equipped with interval arithmetic using Moore's recursive technique for Taylor coefficients computation (see [16, 17]).

(2) Relate the obtained Taylor coefficients to those of the power form in (1), and denote the coefficients in this form as a_I.

(3) Compute the l-tuple of indices D given by

$$D = (d_1, \ldots, d_l), \quad d_1, \ldots, d_l \geq \left[\frac{1}{w(\mathbf{X})}\right]^{m+1}, \tag{5}$$

and then the l-tuple of indices N' given by

$$N' = (n'_1, \ldots, n'_l), \quad n'_k = \max\{n_k, d_k\}, \quad k = 1, \ldots, l, \tag{6}$$

and construct $S' = \{I : I \leq N'\}$.

(4) Find a patch $B(\mathbf{U})$ of Bernstein coefficients of p on $\mathbf{U}$ by executing Algorithm Patch: $B(\mathbf{U}) = \mathrm{Patch}(\mathbf{X}, a_I)$ with S' used in place of S in this algorithm. Then, compute an enclosure for the range of $\bar{p}(\mathbf{X})$ as

$$B^* = [\min\ B(\mathbf{U}), \max\ B(\mathbf{U})]. \tag{7}$$

(5) Compute an enclosure for the range of f over $\mathbf{X}$ as

$$F_{\mathrm{LR}}(\mathbf{X}) = B^* + R(\mathbf{X}). \tag{8}$$

(6) RETURN $F_{\mathrm{LR}}(\mathbf{X})$.
END Algorithm

2.4. IMPROVED TAYLOR–BERNSTEIN FORM

As seen from (5), D becomes large quite quickly as $w(\mathbf{X})$ becomes smaller, leading to high degrees $N' \gg N$ of the Bernstein polynomials in (6). As a consequence, the Bernstein step of LR algorithm becomes very computationally intensive as the domain intervals shrink in widths.

Algorithm TB in Ref. [20] uses a different Bernstein step based on Bernstein polynomials of degree N (note that N is the minimum degree of Bernstein polynomials we can possibly use) and is equipped with the tools of subdivision and vertex condition checks. Further, in step (1) of this algorithm, the Taylor model technique of Berz *et al.* [2, 14] is used for computing the Taylor coefficients in parallel with the remainder interval. Berz *et al.* have shown that the Taylor model technique is more computationally efficient and gives tighter results than a direct implementation of Moore's recursive techniques.

Algorithm TB computes an enclosure for the range of f on $\mathbf{X}$ using the Taylor form of order m and Bernstein polynomials of degree N. The range of polynomial part of Taylor expansion is computed in this algorithm using Bernstein subdivision, and a vertex condition check is done on every subdivision.

2.4.1. *TB Algorithm.* $F_{\mathrm{TB}}(\mathbf{X}) = TB(\mathbf{X}, f, m)$

Inputs: The box $\mathbf{X}$, an expression for the function f, and the order m of Taylor form to be used.
Output: An enclosure $F_{\mathrm{TB}}(\mathbf{X})$ of the range of f on $\mathbf{X}$.
(1) For the given function f, compute Taylor coefficients of p in (3) in parallel with the remainder interval $R(\mathbf{X})$ using the Taylor model technique of Berz *et al.* [2].
(2) Relate the obtained Taylor coefficients to those of the power form in (1), and denote the coefficients in this form as a_I.
(3) Find the range $\bar{p}(\mathbf{X})$ on $\mathbf{X}$ using Algorithm Bounder:

$$\bar{p}(\mathbf{X}) = \text{Bounder}(\mathbf{X}, a_I). \tag{9}$$

(4) Using $R(\mathbf{X})$ obtained in step (1) and $\bar{p}(\mathbf{X})$ obtained in step (3), compute an enclosure for the range of f over $\mathbf{X}$ as

$$F_{\text{TB}}(\mathbf{X}) = \bar{p}(\mathbf{X}) + R(\mathbf{X}). \tag{10}$$

(5) RETURN $F_{\text{TB}}(\mathbf{X})$.
END Algorithm

It is trivial to show that the TB form computed in the proposed algorithm also has the property of $m+1$ convergence order.

2.5. COMBINED TAYLOR–BERNSTEIN FORM

Typically, the F_{TB} form requires excessive subdivisions for 'large' $w(\mathbf{X})$, whereas the F_{LR} form requires excessively high degrees of Bernstein form for 'small' $w(\mathbf{X})$. It may be advantageous to have a new inclusion form that switches between these two forms depending on the domain widths, i.e., behave as F_{LR} for 'large' domain widths, and as F_{TB} for 'small' domain widths.

Let D be as in (5) and recall that N is the tuple of maximum degrees of $\mathbf{x}$ in $p(\mathbf{x})$ given by (1). The basic idea of the combined form is as follows. From (5) and (6),

- for 'large' $w(\mathbf{X})$, $D \ll N$, so $N' = N$. Therefore, for such domain widths, it would be simpler and more efficient to use F_{LR} based on a Bernstein form of degree N, rather than F_{TB} that involves successive subdivisions till the vertex property is satisfied on every subdivision.
- for 'small' $w(\mathbf{X})$, $D \gg N$, so $N' \gg N$. Therefore, for such domain widths, it would be more efficient to use F_{TB} based on Bernstein form of degree N, rather than F_{LR} that involves Bernstein form of high to very high degree.

Therefore, if $N \geq D$, we invoke LR algorithm given in Section 2.3 except that S is used in place of S'. That is, we compute a non-sharp enclosure of the exact range of the polynomial part of Taylor expansion using Bernstein polynomials of degree N. Otherwise, we invoke TB algorithm given in Section 2.4, i.e., we compute the exact range of polynomial part of Taylor expansion using subdivision and a vertex condition check on every subdivision. The combined algorithm is called as CTB algorithm and the resulting form as the combined TB form, denoted F_{CTB}.

As F_{CTB} uses either of the existing TB forms F_{LR} or F_{TB} to enclose the function range for any given domain width, and since F_{LR} and F_{TB} have the $(m+1)$th convergence order property, it follows that F_{CTB} also has the $(m+1)$th convergence order property.

Algorithm CTB in [21] computes such a combined TB form.

2.5.1. *CTB Algorithm.* $[F_{\mathrm{CTB}}(\mathbf{X}), \bar{p}(\mathbf{X}), B^*, R(\mathbf{X}), i_f] = \mathrm{CTB}(\mathbf{X}, f, m)$

Inputs: The box $\mathbf{X}$, an expression for the function f, and the order m of Taylor form to be used.
Output: Enclosure $F_{\mathrm{CTB}}(\mathbf{X})$ of the range of f on $\mathbf{X}$, the range $\bar{p}(\mathbf{X})$ of the polynomial part of the Taylor form of f, an enclosure B^* of the same, an enclosure $R(\mathbf{X})$ of the remainder part of the Taylor form, and a flag i_f that takes the value zero (resp. unity) depending on whether F_{LR} (resp. F_{TB}) form is used in the algorithm.
Note: Depending on whether F_{LR} (resp. F_{TB}) is used, the quantity $\bar{p}(\mathbf{X})$ (resp. B^*) is set to the empty interval.

```
(1)  For the given function $f$, compute Taylor coefficients of $p$ in (3) in parallel with the remainder interval $R(\mathbf{X})$, using the Taylor model technique of Berz et al. [2].
(2)  Relate the obtained Taylor coefficients to those of the power form in
(1), and denote the coefficients in this form as $a_I$.
(3)  Compute the $l$-tuple of indices $D$ given by
     $D = (d_1, \ldots, d_l), \quad d_1, \ldots, d_l \geq \left[\frac{1}{w(\mathbf{X})}\right]^{m+1}.$
     If $N \geqslant D$ then go to the following step, else go to step (8).
(4)  Set flag $i_f = 0$ and $\bar{p}(\mathbf{X})$ to the empty interval.
(5)  Find a patch $B(\mathbf{U})$ of Bernstein coefficients of $p$ on $\mathbf{U}$ by executing Algorithm Patch in [20]:
     $B(\mathbf{U}) = \mathrm{Patch}(\mathbf{X}, a_I),$
     then compute an enclosure $B^*$ for the range of $\bar{p}(\mathbf{X})$ as
     $B^* = [\min\ B(\mathbf{U}), \max\ B(\mathbf{U})].$
(6)  Compute an enclosure for the range of $f$ over $\mathbf{X}$ as
     $F_{\mathrm{CTB}}(\mathbf{X}) = B^* + R(\mathbf{X}).$
(7)  Go to step (11).
(8)  Set flag $i_f = 1$ and $B^*$ to the empty interval.
(9)  Compute the range $\bar{p}(\mathbf{X})$ using Algorithm Bounder:
     $\bar{p}(\mathbf{X}) = \mathrm{Bounder}(\mathbf{X}, a_I).$
(10) Using $R(\mathbf{X})$ obtained in step (1) and $\bar{p}(\mathbf{X})$ obtained in above step, compute an enclosure for the range of $f$ over $\mathbf{X}$ as
     $F_{\mathrm{CTB}}(\mathbf{X}) = \bar{p}(\mathbf{X}) + R(\mathbf{X}).$
(11) RETURN $F_{\mathrm{CTB}}(\mathbf{X})$, $\bar{p}(\mathbf{X})$, $B^*$, $R(\mathbf{X})$, $i_f$ and EXIT.
END Algorithm
```

3. Interval Global Optimization using Taylor–Bernstein Inclusion Functions

3.1. MS ALGORITHM

We first outline the well-known Moore–Skelboe (MS) algorithm of interval analysis. Actually, the algorithm is the MS algorithm augmented with the monotonicity test and the cut-off test of Ichida and Fujii [10]. However, for convenience we refer to it as just the MS algorithm.

3.1.1. *MS Algorithm for Global Optimization [24]*

Inputs: The box $\mathbf{X}$, natural inclusion functions [17] F and F' for the function f and its Jacobian, respectively, and an accuracy parameter ε.
Output: A lower bound, of accuracy ε, on the global minimum of f over $\mathbf{X}$. This lower bound is output as the value of variable y in the last but one step below.

```
BEGIN Algorithm
(1) Set $\mathbf{Y} = \mathbf{X}$.
(2) Calculate $F(\mathbf{Y})$.
(3) Set $y = \min\ F(\mathbf{Y})$.
(4) Initialize the list $L = ((\mathbf{Y}, y))$ and the cut-off value $z = \max\ F(\mathbf{Y})$.
(5) Choose a coordinate direction $k$ parallel to which $\mathbf{Y}$ has an edge of
    maximum length,[3] i.e., choose $k$ as
        $k = \{i : w(\mathbf{Y}) = w(\mathbf{Y}_i)\}$.
(6) Bisect $\mathbf{Y}$ in direction $k$ getting boxes $\mathbf{V}^1$ and $\mathbf{V}^2$ such that $\mathbf{Y} = \mathbf{V}^1 \bigcup \mathbf{V}^2$.
(7) Monotonicity test (see Remark 3.1): discard any box $\mathbf{V}^i$ if $0 \notin F'_j(\mathbf{V}^i)$ for
    any $j \in \{1, 2, \dots, l\}$ and $i = 1, 2$.
(8) Calculate $F(\mathbf{V}^1)$ and $F(\mathbf{V}^2)$.
(9) Set $v^i = \min\ F(\mathbf{V}^i)$ for $i = 1, 2$.
(10) Update the cut-off value $z$ as
        $z = \min\{z, \max\ F(\mathbf{V}^1), \max\ F(\mathbf{V}^2)\}$.
(11) Remove $(\mathbf{Y}, y)$ from the list $L$.
(12) Add the pairs $(\mathbf{V}^1, v^1), (\mathbf{V}^2, v^2)$ to the list $L$ such that the second mem-
     bers of all pairs of the list do not decrease.
(13) Cut-off test: discard from the list all pairs whose second members are
     greater than $z$.
(14) Denote the first pair of the list by $(\mathbf{Y}, y)$.
(15) If the width of $F(\mathbf{Y})$ is less than $\varepsilon$, then print $y$ and EXIT algorithm.
```

[3]For other bisection strategies that have often been found more efficient (see, for instance [4]). The same remark also holds for the bisection step in TMS and TBMS algorithms described in the sequel.

(16) Go to step (5).
END Algorithm

The first pair $(\mathbf{Y}, y)$ of the list in each algorithmic iteration is called the leading pair, and $\mathbf{Y}$ the leading box.

REMARK 3.1. In the monotonicity test if $0 \notin F'_j(\mathbf{V}^i)$ then the interior of $\mathbf{V}^i$ cannot contain a global minimizer. The edge of $\mathbf{V}^i$ still can contain global minimizer if that part of the edge which has the smallest function values is also part of the edge of $\mathbf{X}$. Otherwise, no global minimizer lies in $\mathbf{V}^i$. For details, see [24].

3.2. TAYLOR MOORE – SKELBOE (TMS) ALGORITHM

In this algorithm, we simply use the Taylor model of Berz *et al.* [2] as an inclusion function form for the objective function f in MS algorithm. As this involves using **T**aylor model in the Moore–Skelboe algorithm, we call it as TMS algorithm. TMS algorithm is not new in the literature, and has been proposed and investigated, for instance, in Ref. [12].

3.3. TAYLOR–BERNSTEIN MOORE–SKELBOE (TBMS) ALGORITHM

TBMS algorithm in Ref. [20] involves the following modifications to MS algorithm:

(1) The TB form F_{TB} is used as an inclusion function form for f. Using this form, an enclosure of the range of f over a given box can be obtained using TB algorithm.
(2) The cut-off value is now defined as $z = \min\ \bar{p}(\mathbf{Y}) + \max\ R(\mathbf{Y})$.
(3) The termination criterion is modified, based on the width of the remainder interval $R(\mathbf{Y})$.

3.3.1. *TBMS Algorithm [20]*

Inputs: The box $\mathbf{X}$, order m of the Taylor form to be used, natural inclusion function F' for the Jacobian of the function f, and an accuracy parameter ε.
Output: A lower bound, of accuracy ε, on the global minimum of f over $\mathbf{X}$. This lower bound is output as the value of variable y in the last but one step below.

```
BEGIN Algorithm
(1) Set $\mathbf{Y} = \mathbf{X}$.
(2) Calculate $F_{\mathrm{TB}}(\mathbf{Y})$ using TB algorithm: $[F_{\mathrm{TB}}(\mathbf{Y}), \bar{p}(\mathbf{Y}), R(\mathbf{Y})] = TB(\mathbf{Y}, f, m)$
(3) Set $y = \min\ F_{\mathrm{TB}}(\mathbf{Y})$.
(4) Initialize the list $L = ((\mathbf{Y}, y))$ and the cut-off value $z$ as
        $z = \min\ \bar{p}(\mathbf{Y}) + \max\ R(\mathbf{Y})$.
```

(5) Choose a coordinate direction k parallel to which $\mathbf{Y}$ has an edge of maximum length, i.e., choose k as

$$k = \{i : w(\mathbf{Y}) = w(\mathbf{Y}_i)\}.$$

(6) Bisect $\mathbf{Y}$ in direction k getting boxes $\mathbf{V}^1$ and $\mathbf{V}^2$ such that $\mathbf{Y} = \mathbf{V}^1 \bigcup \mathbf{V}^2$.
(7) Monotonicity test (see Remark 3.1): discard any box $\mathbf{V}^i$ if $0 \notin F'_j(\mathbf{V}^i)$ for any $j \in \{1, 2, \ldots, l\}$ and $i = 1, 2$.
(8) Calculate $F_{\mathrm{TB}}(\mathbf{V}^1)$ and $F_{\mathrm{TB}}(\mathbf{V}^2)$ using TB algorithm.
(9) Set $v^i = \min\ F(\mathbf{V}^i)$ for $i = 1, 2$.
(10) Update the cut-off value z as

$$z = \min\left\{z, \min\ \bar{p}(\mathbf{V}^1) + \max\ R(\mathbf{V}^1), \min\ \bar{p}(\mathbf{V}^2) + \max\ R(\mathbf{V}^2)\right\}.$$

(11) Remove $(\mathbf{Y}, y)$ from the list L.
(12) Add the pairs $(\mathbf{V}^1, v^1), (\mathbf{V}^2, v^2)$ to the list L such that the second members of all pairs of the list do not decrease.
(13) Cut-off test: discard from the list all pairs whose second members are greater than z.
(14) Denote the first pair of the list by $(\mathbf{Y}, y)$.
(15) If the width of $R(\mathbf{Y})$ is less than ε, then print y and EXIT algorithm.
(16) Go to step (5).
END Algorithm

The convergence properties of TMS algorithm as well as that of TBMS algorithm follow immediately from the convergence results for inclusion functions of higher-order in the MS algorithm, as given by Moore and Ratschek [18] and Ratschek [22].

4. Proposed Improvements

Bounder algorithm in Section 2.1 computes the range $\bar{p}(\mathbf{X})$ of the polynomial part p on $\mathbf{X}$. For global optimization problems, where the computation of min $\bar{f}(\mathbf{X})$ is of interest, this algorithm can be tailored and improved as given below. We call the resulting improved algorithm as NewBounder algorithm.

A study of the optimization TBMS algorithm presented in Section 3.3 reveals that the quantity max $\bar{p}(\mathbf{Y})$, where $\mathbf{Y}$ is the current leading box, is never used in the algorithm and is therefore not of interest. Note that the quantity max $\bar{p}(\mathbf{Y})$ required in step (8) of TBMS algorithm is actually found in step (4) of Bounder algorithm, through the application of the vertex condition to max $B(\mathbf{D})$.

Since max $\bar{p}(\mathbf{Y})$ is not of interest in TBMS algorithm, in step (4) of Bounder algorithm we may avoid applying the vertex condition to max $B(\mathbf{D})$ and instead apply it only to min $B(\mathbf{D})$. With this modification, a new NewBounder algorithm arises from Bounder algorithm. It computes an enclosure $P(\mathbf{X})$ of the range $\bar{p}(\mathbf{X})$, with $P(\mathbf{X})$ such that

$$\min\ P(\mathbf{X}) = \min\ \bar{p}(\mathbf{X}), \quad \max\ P(\mathbf{X}) \geq \min\ \bar{p}(\mathbf{X}). \tag{11}$$

We can also incorporate additional improvements into NewBounder algorithm:

- A monotonicity test similar to that used in the MS algorithm, for discarding boxes where surely no global minimizer of p lies.
- A cut-off test, similar to that used in the MS algorithm, for discarding boxes where surely no global minimizer of p lies.
- An improved strategy for selection of subdivision direction of boxes.

We discuss below each of these improvements and then present NewBounder algorithm.

4.1. MONOTONICITY TEST FOR BERNSTEIN PATCHES

On a box $\mathbf{D} \subseteq \mathbf{U}$, the partial derivative with respect to x_r of a polynomial $p(\mathbf{x})$ in Bernstein form is [7]

$$\frac{\partial p}{\partial x_r}(\mathbf{x}) = n_r \sum_{I \leqslant N_{r,-1}} [b_{I_{r,1}}(\mathbf{D}) - b_I(\mathbf{D})] B_{N_{r,-1},I}(\mathbf{x}), \quad 1 \leqslant r \leqslant l, \quad \mathbf{x} \in \mathbf{D}. \tag{12}$$

REMARK 4.1. Let $P'_r(\mathbf{D})$ denote an enclosure of the range of the above partial derivative on $\mathbf{D}$. In the monotonicity test if $0 \notin P'_r(\mathbf{D})$ then the interior of $\mathbf{D}$ cannot contain a global minimizer of p on $\mathbf{U}$. The edge of $\mathbf{D}$ can still contain global minimizer if that part of the edge which has the smallest polynomial value is also part of $\mathbf{U}$. Otherwise, no global minimizer of p lies in $\mathbf{D}$, and $\mathbf{D}$ can be discarded.

The enclosure $P'_r(\mathbf{D})$ can be found by evaluating the natural interval inclusion of the right-hand side of the expression in (12). However, in some cases, the evaluation can be avoided.

REMARK 4.2. From the fact that the Bernstein polynomials $B_{N_{r,-1},I}$ are always non-negative, it is easy to see from (12) that if all $[b_{I_{r,1}}(\mathbf{D}) - b_I(\mathbf{D})]$ are positive (resp. negative), then $P'_r(\mathbf{D}) > 0$ (resp. $P'_r(\mathbf{D}) < 0 \Rightarrow p$) is monotonic with respect to direction r on box $\mathbf{D} \Rightarrow$ the interior of $\mathbf{D}$ cannot contain global minimizer of p.

4.2. DIRECTION SELECTION FOR BERNSTEIN PATCHES

In step (5) of Bounder algorithm in Section 2.1, the direction in which the boxes are subdivided is varied cyclically from 1 to l. A more efficient strategy for selection of the subdivision direction could result in considerably fewer boxes being created and significant overall speed up of this algorithm.

Zettler and Garloff [25] suggest selection of the subdivision direction as the one along which the maximum absolute value of the partial derivatives of p occurs. Applying the triangle inequality and properties of Bernstein polynomials to (12), the authors show that the quantity

$$\max_{xsD}\left|\frac{\partial p}{\partial x_r}(\mathbf{x})\right|$$

can be estimated as

$$\widetilde{I}_r = \max_{I \leqslant N_{r,-1}} |b_{I_{r,1}}(\mathbf{D}) - b_I(\mathbf{D})|.$$

The direction selected for subdivision r_0 is such that

$$\widetilde{I}_{r_0} = \max_{1 \leqslant r \leqslant l} \widetilde{I}_r. \tag{13}$$

A similar strategy can be given based on the second partial derivatives. In NewBounder algorithm, we use the above direction-selection strategy based on the first partial derivatives.

4.3. CUT-OFF TEST FOR BERNSTEIN PATCHES

The list $\mathcal{L}$ in Bounder algorithm consists of pairs $(\mathbf{D}, B(\mathbf{D}))$. Suppose we arrange this list at every iteration such that the minimums of the second members, i.e., min $B(\mathbf{D})$, of all pairs of the list do not decrease.

Now, consider the leading box $\mathbf{D}$ of the list $\mathcal{L}$ at any given iteration of the algorithm. If min $B(\mathbf{D})$ satisfies the vertex condition, then by the range enclosure property of Bernstein coefficients, min $\bar{p}(\mathbf{D}) =$ min $B(\mathbf{D})$. As min $\bar{p}(\mathbf{D}) \geqslant$ min $\bar{p}(\mathbf{X})$, we may discard all boxes $\mathbf{D}'$ in the list $\mathcal{L}$ for which min $B(\mathbf{D}') >$ min $B(\mathbf{D})$.

Suppose instead that min $B(\mathbf{D})$ does not satisfy the vertex condition. Then, $\mathbf{D}$ is subdivided into two subboxes $\mathbf{D}_A$, $\mathbf{D}_B$ and the patches $B(\mathbf{D}_A)$, $B(\mathbf{D}_B)$ computed. By the range enclosure property of Bernstein coefficients given in Section 2.1,

$$\bar{p}(\mathbf{D}_A) \subseteq [\min\ B(\mathbf{D}_A), \max\ B(\mathbf{D}_A)], \quad \bar{p}(\mathbf{D}_B) \subseteq [\min\ B(\mathbf{D}_B), \max\ B(\mathbf{D}_B)].$$

So, if min $B(\mathbf{D}_B) >$ max $B(\mathbf{D}_A)$ then the box $\mathbf{D}_B$ can be discarded in the search for the global minimum. In fact, we may also discard all other boxes $\mathbf{D}'$ in the list $\mathcal{L}$ for which min $B(\mathbf{D}') >$ max $B(\mathbf{D}_A)$.

4.4. ALGORITHM FOR BOUNDING POLYNOMIAL RANGE

We are now ready to present the improved NewBounder algorithm. This algorithm is specially meant for global optimization problems where the primary interest is in obtaining sharp values for min $\bar{p}(\mathbf{X})$ whereas the quantity max $\bar{p}(\mathbf{X})$ can be overestimated.

4.4.1. *NewBounder Algorithm.* $P(\mathbf{X}) = \text{NewBounder}(\mathbf{X}, a_I)$

Inputs: A box $\mathbf{X}$, a polynomial p as in (1) of degree N in l-variables and having coefficients a_I.
Output: An enclosure $P(\mathbf{X})$ of the range $\bar{p}(\mathbf{X})$, where $P(\mathbf{X})$ is as in (11).

```
BEGIN Algorithm
(1) (Compute patch $B(\mathbf{U})$) Execute Patch algorithm
        $B(\mathbf{U}) = \text{Patch}(\mathbf{X}, a_I)$.
(2) (Initialize lists) Set $\mathcal{L} \leftarrow \{(\mathbf{U}, B(\mathbf{U}))\}$, $\mathcal{L}^{\text{sol}} \leftarrow \{\}$. Set cut-off value $z' = \max\ B(\mathbf{U})$.
(3) (Select item for processing) If $\mathcal{L}$ is empty, go to step (11). Otherwise, pick the first item from $\mathcal{L}$, denote it as $(\mathbf{D}, B(\mathbf{D}))$, and delete the item entry from $\mathcal{L}$.
(4) (Check if vertex condition for the min on patch is met within $\mu$) If $(\mathbf{D}, B(\mathbf{D}))$ is such that $\min B(\mathbf{D})$ satisfies the vertex condition within the given machine precision $\mu$ then
    (a) Update the cut-off value as $z' = \min\{z', \min\ B(\mathbf{D})\}$.
    (b) Enter the item in list $\mathcal{L}^{\text{sol}}$ and return to previous step.
(5) (Subdivide and find new patches) Execute Subdivision algorithm
        $[B(\mathbf{D}_A), B(\mathbf{D}_B), \mathbf{D}_A, \mathbf{D}_B] = \text{SD}(\mathbf{D}, B(\mathbf{D}), r_0)$,
    where, $r_0$ is chosen as in (13).
(6) (Monotonicity test, see Remarks 4.1 and 4.2): discard box $\mathbf{D}_A$ if $0 \notin P'_r(\mathbf{D}_A)$ for any $r \in \{1, 2, \ldots, l\}$. Do likewise for box $\mathbf{D}_B$.
(7) Update the cut-off value as $z' = \min\{z', \max\ B(\mathbf{D}_A), \max\ B(\mathbf{D}_B)\}$.
(8) (Add new entries to list) Enter the new items $(\mathbf{D}_A, B(\mathbf{D}_A))$ and $(\mathbf{D}_B, B(\mathbf{D}_B))$ to the list $\mathcal{L}$ such that minimums of the second members, i.e., $\min\ B(\mathbf{D})$, of all pairs of the list do not decrease.
(9) Cut-off test: discard from the list all pairs whose minimums of the second members are greater than $z'$.
(10) Return to step (3).
(11) Compute an enclosure $P(\mathbf{X})$ of the range $\bar{p}(\mathbf{X})$ as the minimum to maximum over all the second entries of the items present in list $\mathcal{L}^{\text{sol}}$.
(12) RETURN $P(\mathbf{X})$.
END Algorithm
```

4.5. A TIGHTER ENCLOSURE OF THE FUNCTION MINIMUM

In the proposed algorithm for optimization given below, we are interested in computing an enclosure that is as tight as possible for the global minimum of the objective function f. This quantity is given by min $\bar{f}(\mathbf{Y})$, where $\mathbf{Y}$ is the leading box at any given iteration of the algorithm. Suppose we compute the combined TB form using CTB algorithm and obtain an enclosure of $\bar{f}(\mathbf{Y})$. If $w(\mathbf{Y})$ happens to be small enough, then the improved

TB form $F_{\mathrm{TB}}(\mathbf{Y})$ is in turn invoked in CTB algorithm. In this case, we can use the interval

$$[\min\ \bar{p}(\mathbf{Y}) + \min\ R(\mathbf{Y}), \min\ \bar{p}(\mathbf{Y}) + \max\ R(\mathbf{Y})]$$

instead of the interval $F_{\mathrm{CTB}}(\mathbf{Y})$ to get a tighter enclosure of $\min \bar{f}(\mathbf{Y})$. MIN_CTB algorithm encapsulates this idea.

4.5.1. *MIN_CTB Algorithm.* $F_{\mathrm{min,CTB}}(\mathbf{X}) = \mathrm{MIN\_CTB}(\mathbf{X}, f, m)$

Inputs: The box $\mathbf{X}$, an expression for the function f, and the order m of Taylor form to be used.
Output: An enclosure $F_{\mathrm{min,CTB}}(\mathbf{X})$ for $\min \bar{f}(\mathbf{X})$.

```
BEGIN Algorithm
(1) Call CTB algorithm:
        $[F_{\mathrm{CTB}}(\mathbf{X}), \bar{p}(\mathbf{X}), B^*, R(\mathbf{X}), i_f] = \mathrm{CTB}(\mathbf{X}, f, m)$.
(2) If $i_f = 0$ set
        $F_{\mathrm{min,CTB}}(\mathbf{X}) = F_{\mathrm{CTB}}(\mathbf{X})$
    else set
        $F_{\mathrm{min,CTB}}(\mathbf{X}) = [\min\ \bar{p}(\mathbf{X}) + \min\ R(\mathbf{X}), \min\ \bar{p}(\mathbf{X}) + \max\ R(\mathbf{X})]$.
(3) RETURN $F_{\mathrm{min,CTB}}(\mathbf{X})$ and EXIT.
END Algorithm
```

5. A New Optimization Algorithm

We next propose an optimization algorithm based on the following ideas. As before, let $\mathbf{Y}$ be the leading box at any given iteration of the MS algorithm. Then,

(1) Since the computation of $F_{\mathrm{NIE}}(\mathbf{Y})$ is relatively inexpensive, and since $F_{\mathrm{NIE}}(\mathbf{Y})$ sometimes gives sharper enclosures than the sophisticated TB forms even for small domains, we always compute $F_{\mathrm{NIE}}(\mathbf{Y})$.
(2) If $w(R(\mathbf{Y})) > w(F_{\mathrm{NIE}}(\mathbf{Y}))$, then $F_{\mathrm{NIE}}(\mathbf{Y})$ gives a sharper enclosure of the range than the TB forms. Since the effort to bound the polynomial range $\bar{p}(\mathbf{Y})$ may not be worthwhile in these cases, we do not use the TB forms and instead use $F_{\mathrm{NIE}}(\mathbf{Y})$.
(3) If the Taylor model technique of Berz *et al.* [2, 14] is used for computing $R(\mathbf{Y})$ needed in the above step, then we concurrently also obtain the Taylor model $F_{\mathrm{TM}}(\mathbf{Y})$. Then, as anyway the cost of computing $F_{\mathrm{TM}}(\mathbf{Y})$ is incurred in the Taylor model technique, instead of using only $F_{\mathrm{NIE}}(\mathbf{Y})$ we can use $F_{\mathrm{NIE}}(\mathbf{Y}) \bigcap F_{\mathrm{TM}}(\mathbf{Y})$.
(4) If $w(R(\mathbf{Y})) \leqslant w(F_{\mathrm{NIE}}(\mathbf{Y}))$, we also use the combined TB form and get an enclosure of the global minimum $\min\ \bar{f}(\mathbf{Y})$ using $F_{\mathrm{min,CTB}}(\mathbf{Y})$. We then intersect the result with $F_{\mathrm{NIE}}(\mathbf{Y}) \bigcap F_{\mathrm{TM}}(\mathbf{Y})$ to obtain a (hopefully) sharper enclosure $F(\mathbf{Y})$ of the global minimum $\min\ \bar{f}(\mathbf{Y})$.

(5) A lower bound on the global minimum min $\bar{f}(\mathbf{Y})$ is obtained as $y = \min\ F(\mathbf{Y})$.
(6) The global minimum min $\bar{f}(\mathbf{Y})$ cannot exceed max $F(\mathbf{Y})$. Hence, the cut-off value z in the MS algorithm can be updated accordingly.
(7) Thus, the global minimum min $\bar{f}(\mathbf{Y})$ is bounded by [min $F(\mathbf{Y})$, max $F(\mathbf{Y})$]. The maximum possible error in computing the global minimum is therefore given by $w(F(\mathbf{Y}))$.
(8) This leads to the termination condition for the algorithm as $w(F(\mathbf{Y})) < \varepsilon$.

We can now present our algorithm for global optimization. Since our global optimization algorithm involves the **C**ombined **T**aylor–**B**ernstein form in **M**oore–**S**kelboe type algorithm, we call it as CTBMS algorithm.

4.5.2. *CTBMS Algorithm*

Inputs: The box $\mathbf{X}$, order m of the Taylor form to be used, natural inclusion function F_{NIE} for the function $f : \mathbf{X} \rightarrow \Re$, an inclusion function F' for the Jacobian of f, and an accuracy parameter ε.
Output: A lower bound, of accuracy ε, on the global minimum of f over $\mathbf{X}$. This lower bound is output as the value of variable y in the last but one step below.

```
BEGIN Algorithm
(1) Set $\mathbf{Y} = \mathbf{X}$.
(2) Calculate $F_{\mathrm{NIE}}(\mathbf{Y})$ and $F_{\mathrm{TM}}(\mathbf{Y}) = p(\mathbf{Y}) + R(\mathbf{Y})$.
    (a) If $w(R(\mathbf{Y})) > w(F_{\mathrm{NIE}}(\mathbf{Y}))$, set
        $F(\mathbf{Y}) = F_{\mathrm{NIE}}(\mathbf{Y}) \bigcap F_{\mathrm{TM}}(\mathbf{Y})$,
    and go to the following step, else compute $F_{\mathrm{min,CTB}}(\mathbf{Y})$ using MIN_CTB algorithm
        $F_{\mathrm{min,CTB}}(\mathbf{Y}) = \mathrm{MIN\_CTB}(\mathbf{Y}, f, m)$,
    and set
        $F(\mathbf{Y}) = F_{\mathrm{NIE}}(\mathbf{Y}) \bigcap F_{\mathrm{TM}}(\mathbf{Y}) \bigcap F_{\mathrm{min,CTB}}(\mathbf{Y})$.
(3) Set $y = \min\ F(\mathbf{Y})$.
(4) Initialize the list $L = ((\mathbf{Y}, y))$ and the cut-off value $z = \max\ F(\mathbf{Y})$.
(5) Choose a coordinate direction $k$ parallel to which $\mathbf{Y}$ has an edge of maximum length, i.e., choose $k$ as
        $k = \{i : w(\mathbf{Y}) = w(\mathbf{Y}_i)\}$.
(6) Bisect $\mathbf{Y}$ in direction $k$ getting boxes $\mathbf{V}^1$ and $\mathbf{V}^2$ such that $\mathbf{Y} = \mathbf{V}^1 \bigcup \mathbf{V}^2$.
(7) Monotonicity test (see Remark 3.1): discard any box $\mathbf{V}^i$ if $0 \notin F'_j(\mathbf{V}^i)$ for any $j \in \{1, 2, \ldots, l\}$ and $i = 1, 2$.
(8) For $i = 1, 2$ do the following: Calculate $F_{\mathrm{NIE}}(\mathbf{V}^i)$ and $F_{\mathrm{TM}}(\mathbf{V}^i) = p(\mathbf{V}^i) + R(\mathbf{V}^i)$.
```

```
    (a) If $w(R(\mathbf{V}^i)) > w(F_{\mathrm{NIE}}(\mathbf{V}^i))$, set
        $F(\mathbf{V}^i) = F_{\mathrm{NIE}}(\mathbf{V}^i) \bigcap F_{\mathrm{TM}}(\mathbf{V}^i)$
    else compute $F_{\min,\mathrm{CTB}}(\mathbf{V}^i)$ using MIN_CTB algorithm:
        $F_{\min,\mathrm{CTB}}(\mathbf{V}^i) = \mathrm{MIN\_CTB}(\mathbf{V}^i, f, m)$,
    and set
        $F(\mathbf{V}^i) = F_{\mathrm{NIE}}(\mathbf{V}^i) \bigcap F_{\mathrm{TM}}(\mathbf{V}^i) \bigcap F_{\min,\mathrm{CTB}}(\mathbf{V}^i)$.
(9) Set $v^i = \min\ F(\mathbf{V}^i)$ for $i = 1, 2$.
(10) Update the cut-off value $z$ as
        $z = \min\{z, \max\ F(\mathbf{V}^1), \max\ F(\mathbf{V}^2)\}$.
(11) Remove $(\mathbf{Y}, y)$ from the list $L$.
(12) Add the pairs $(\mathbf{V}^1, v^1), (\mathbf{V}^2, v^2)$ to the list $L$ such that the second members of all pairs of the list do not decrease.
(13) Cut-off test: discard from the list all pairs whose second members are greater than $z$.
(14) Denote the first pair of the list by $(\mathbf{Y}, y)$.
(15) If $w(F(\mathbf{Y})) < \varepsilon$ then print $y$ and EXIT algorithm.
(16) Go to step (5).
END Algorithm
```

From [20, Section 4.2], it is straightforward to prove that y is a lower bound on the global minimum of f over $\mathbf{X}$. Further, the convergence properties of CTBMS algorithm follows immediately from the convergence results for inclusion functions of higher-order in the MS algorithm, as given by Moore and Ratschek in [18] and Ratschek [22].

6. Numerical Tests

We test and compare the performances of CTBMS, TBMS, TMS, and MS algorithms on 11 benchmark examples. We set the accuracy $\varepsilon = 1\mathrm{e} - 05$ and the Taylor order $m = 4$. For all computations, we use a PC/Pentium III 800 MHz 256 MB RAM machine with a FORTRAN 90 compiler, and version 8.1 of the COSY-INFINITY package of Berz *et al.* [1, 9].

To compare the performances of the various algorithms, we use the following performance metrics:

- Number of algorithmic iterations;
- Computational time (seconds);
- Maximum list length;
- Final list length.

The examples are as given below.

EXAMPLE 6.1. Jennrich and Sampson function [19, Problem 6]. The 2-dim function is

$$f(x) = \sum_{i=1}^{10} f_i(x)^2, \quad f_i(x) = 2 + 2i - (\exp(ix_1) + \exp(ix_2)).$$

We take the initial domain as $\mathbf{X} = ([-1, 1]^2)$.

EXAMPLE 6.2. Bard function [19, Problem 8]. The 3-dim function is

$$f(x) = \sum_{i=1}^{15} f_i(x)^2, \quad f_i(x) = y_i - \left(x_1 + \frac{u_i}{v_i x_2 + w_i x_3}\right),$$

$$u_i = i, \quad v_i = 16 - i, \quad w_i = \min(u_i, v_i),$$

where, the values of y_i for $i = 1, \ldots, 15$ are given in the cited paper. We take the initial domain as $\mathbf{X} = ([-0.25, 0.25], [0.01, 2.5]^2)$.

EXAMPLE 6.3. Box 3-dim function [19, Problem 12]. The function is

$$f(x) = \sum_{i=1}^{10} f_i(x)^2, \quad f_i(x) = \exp(-t_i x_1) - \exp(-t_i x_2)$$
$$- x_3[\exp(-t_i) - \exp(-10t_i)], \quad t_i = \frac{i}{10}.$$

We take the initial domain as $\mathbf{X} = ([-20, 20], [1, 20]^2)$.

EXAMPLE 6.4. Brown and Dennis function [19, Problem 16]. The 4-dim function is

$$f(x) = \sum_{i=1}^{20} f_i(x)^2, \quad f_i(x) = (x_1 + t_i x_2 - \exp(t_i))^2$$
$$+ (x_3 + x_4 \sin(t_i) - \cos(t_i))^2, \quad t_i = \frac{i}{5}.$$

We take the initial domain as $\mathbf{X} = ([-10, 0, -100, -20], [100, 15, 0, 0.2])$.

EXAMPLE 6.5. Variably dimensioned function [19, Problem 25]. The 2-dim function is

$$f(x) = \sum_{i=1}^{4} f_i(x)^2, \quad f_1(x) = x_1 - 1, \quad f_2(x) = x_2 - 1,$$

$$f_3(x) = \sum_{j=1}^{2} j(x_j - 1), \quad f_4(x) = \left(\sum_{j=1}^{2} j(x_j - 1)\right)^2.$$

We take the initial domain as $\mathbf{X} = ([-1.5, 1.5]^2)$.

EXAMPLE 6.6. Linear – rank 1 with zero columns and rows [19, Problem 34]. The 2-dim function is

$$f(x) = \sum_{i=1}^{4} f_i(x)^2, \quad f_1(x) = -1, \quad f_2(x) = (2x_1 + 3x_2) - 1,$$
$$f_3(x) = 2(2x_1 + 3x_2) - 1, \quad f_4(x) = -1.$$

We take the initial domain as $\mathbf{X} = ([-10, 10]^2)$.

EXAMPLE 6.7. Linear function – full rank [19, Problem 32]. The 4-dim function is

$$f(x) = \sum_{i=1}^{4} f_i(x)^2, \quad f_i(x) = x_i - \frac{1}{2}\left(\sum_{j=1}^{4} x_j\right) - 1.$$

We take the initial domain as $\mathbf{X} = ([-1, 1]^4)$.

EXAMPLE 6.8. Extended Rosenbrock function [19, Problem 21]. The 2-dim function is

$$f(x) = \sum_{i=1}^{2} f_i(x)^2, \quad f_1(x) = 10(x_2 - x_1^2), \quad f_2(x) = 1 - x_1.$$

We take the initial domain as $\mathbf{X} = ([-12, 12]^2)$.

EXAMPLE 6.9. Discrete boundary value function [19, Problem 28]. The 2-dim function is

$$f(x) = \sum_{i=1}^{2} f_i(x)^2, \quad f_i(x) = 2x_i - x_{i-1} - x_{i+1} + \frac{h^2(x_i + t_i + 1)^3}{2},$$
$$h = \frac{1}{3}, t_i = ih, \quad x_0 = x_3 = 0.$$

We take the initial domain as $\mathbf{X} = ([-5, 5]^2)$.

Table 1. Domains used, dimensions and the global minimum over the given domain

Ex.	Test function	Dim	Domain	Global minimum
6.1	Jennrich and Sampson	2	$[-1,1]^2$	124.36217...
6.2	Bard	3	$[-0.25,0.25]\ [0.01,2.5]^2$	8.213...E−3
6.3	Box 3-dim	3	$[-20,20][1,20]^2$	0.00000...
6.4	Brown and Dennis	4	$[-10,100][0,15][-100,0]\ [-20,0.2]$	88860.47976...
6.5	Variably dim.	2	$[-1.5,1.5]^2$	0.00000...
6.6	Linear–rank1	2	$[-10,10]^2$	2.19999...
6.7	Linear–full rank	4	$[-1,1]^4$	0.00000...
6.8	Extended Rosenbrock	2	$[-12,12]^2$	0.00000...
6.9	Discrete boundary	2	$[-5,5]^2$	0.00000...
6.10	Brown almost–linear	4	$[-2.5,2.5]^4$	0.00000...
6.11	Chebyquad	4	$[-2,2]^4$	0.00000...

EXAMPLE 6.10. Brown almost – linear function [19, Problem 27]. The 4-dim function is

$$f(x) = \sum_{i=1}^{4} f_i(x)^2, \quad f_i(x) = x_i + \sum_{j=1}^{4} x_j - 5,$$

$$i = 1, \ldots, 3, \quad f_4(x) = \left(\prod_{j=1}^{4} x_j \right) - 1.$$

We take the initial domain as $\mathbf{X} = ([-2.5, 2.5]^4)$.

Table 2. Comparison of number of iterations required by various algorithms

Ex.	Test Function	Dim	Iterations	MS	TMS	TBMS	CTBMS
6.1	Jennrich	2	Number	1961	427	65	60
	and		Ratio	–	4.59	30.17	32.68
	Sampson		% Reduction	–	78.23	96.69	96.94
6.2	Bard	3	Number	*	*	202	46
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.3	Box 3-dim	3	Number	1208	*	451	105
			Ratio	–	–	2.68	11.5
			% Reduction	–	–	62.67	91.31
6.4	Brown	4	Number	*	455	66	3
	And		Ratio	–	–	–	–
	Dennis		% Reduction	–	–	–	–
6.5	Variably	2	Number	23	151	3	5
	Dimensioned		Ratio	–	0.15	7.67	4.6
			% Reduction	–	−556.52	86.96	78.26
6.6	Linear	2	Number	*	*	133	134
	– rank 1		Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.7	Linear	4	Number	4757	*	166	122
	– full rank		Ratio	–	–	28.66	38.99
			% Reduction	–	–	96.51	97.44
6.8	Extended	2	Number	86	2906	82	24
	Rosenbrock		Ratio	–	0.03	1.05	3.58
			% Reduction	–	−327.91	4.65	72.09
6.9	Discrete	2	Number	69	133	20	19
	Boundary		Ratio	–	0.52	3.45	3.63
	value		% Reduction	–	−92.75	71.01	72.46
6.10	Brown	4	Number	*	*	669	331
	–almost linear		Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.11	Chebyquad	4	Number	*	*	*	1448
			Ratio	–	–	–	–
			% Reduction	–	-	–	–

EXAMPLE 6.11. Chebyquad function [19, Problem 35]. The 4-dim function is

$$f(x) = \sum_{i=1}^{4} f_i(x)^2, \quad f_i(x) = \frac{1}{4}\sum_{j=1}^{4} T_i(x_j) - \int_0^1 T_i(x)\,\mathrm{d}x,$$

where T_i is the ith Chebyshev polynomial shifted to the interval $[0, 1]$. Hence,

$$\int_0^1 T_i(x)\,\mathrm{d}x = \begin{cases} 0 & \text{for } i \text{ odd,} \\ \frac{-1}{(i^2-1)} & \text{for } i \text{ even.} \end{cases}$$

Table 3. Comparison of computation time required by various algorithms

Ex.	Test Function	Dim	Time	MS	TMS	TBMS	CTBMS
6.1	Jennrich and Sampson	2	Number	10.1	3.9	1.47	1.2
			Ratio	–	2.59	6.87	8.42
			% Reduction	–	61.39	85.45	88.12
6.2	Bard	3	Number	> 1 h	> 1 h	60.1	48.2
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.3	Box 3-dim	3	Number	11.1	*	28.9	4.08
			Ratio	–	–	0.38	2.72
			% Reduction	–	–	−160.36	63.24
6.4	Brown and Dennis	4	Number	> 1 h	5.31	7.35	2.88
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.5	Variably dimensioned	2	Number	7E−2	1.9E−2	5E−2	2E−2
			Ratio	–	3.68	1.4	3.5
			% Reduction	–	72.86	28.57	71.43
6.6	Linear – rank 1	2	Number	> 10 h	> 10 h	3559.9	3011.9
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.7	Linear – full rank	4	Number	287.3	*	59.2	46.0
			Ratio	–	–	4.85	6.25
			% Reduction	–		79.39	83.99
6.8	Extended Rosenbrock	2	Number	2E−2	10.76	1.0	0.37
			Ratio	–	0.002	0.02	0.05
			% Reduction	–	−5E4	−4900	−1750
6.9	Discrete Boundary value	2	Number	3E−2	2E−1	3E−1	8E−2
			Ratio	–	0.15	0.1	0.38
			% Reduction	–	−566.67	−900.1	−166.67
6.10	Brown almost linear	4	Number	> 10 h	> 10 h	4914.0	3112.72
			Ratio	–	–	–	–
			\% Reduction	–	–	–	–
6.11	Chebyquad	4	Number	> 10 h	> 10 h	> 10 h	2963.6
			Ratio	–	–	–	–
			\% Reduction	–	–	–	–

We take the initial domain as $\mathbf{X} = ([-2, 2]^4)$.

6.1. RESULTS AND DISCUSSION

Table 1 lists the global minimum obtained using CTBMS algorithm in each example, whereas Tables 2–5 give the obtained results in terms of the chosen performance metrics.[4] For each metric, the values of ratio and the percent reduction are computed as

Table 4. Comparison of maximum list length required by various algorithms

Ex.	Test Function	Dim	Max. list length	MS	TMS	TBMS	CTBMS
6.1	Jennrich and Sampson	2	Number	81	32	14	13
			Ratio	–	2.53	5.79	6.23
			% Reduction	–	60.49	82.72	83.95
6.2	Bard	3	Number	*	*	38	15
			Ratio	–	–	–	–
			% Reduction	–	-	–	–
6.3	Box 3-dim	3	Number	295	*	94	42
			Ratio	–	–	3.14	7.02
			% Reduction	–	–	68.14	85.76
6.4	Brown and Dennis	4	Number	*	44	15	2
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.5	Variably Dimensioned	2	Number	7	38	1	5
			Ratio	–	0.18	7.0	1.4
			% Reduction	–	−442.86	85.71	28.57
6.6	Linear – rank 1	2	Number	*	*	53	53
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.7	Linear – full rank	4	Number	1547	*	82	63
			Ratio	–	–	18.87	24.56
			% Reduction	–	–	94.70	95.93
6.8	Extended Rosenbrock	2	Number	20	340	43	12
			Ratio	–	0.06	0.47	1.67
			% Reduction	–	−1600	−115	40
6.9	Discrete Boundary Value	2	Number	12	11	7	5
			Ratio	–	1.1	1.71	2.4
			% Reduction	–	8.33	41.67	58.34
6.10	Brown Almost Linear	4	Number	*	*	370	60
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.11	Chebyquad	4	Number	*	*	*	305
			Ratio	–	–	–	–
			% Reduction	–	–	–	–

[4]A starred entry in the tables indicates that a solution was not obtained with the corresponding algorithm for the prescribed accuracy, due to excessive time and/or memory requirements.

Table 5. Comparison of final list length required by various algorithms

Ex.	Test Function	Dim	Final list length	MS	TMS	TBMS	CTBMS
6.1	Jennrich and Sampson	2	Number	37	24	1	1
			Ratio	–	1.54	37	37
			% Reduction	–	35.13	97.30	97.30
6.2	Bard	3	Number	*	*	1	3
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.3	Box 3-dim	3	Number	254	*	1	15
			Ratio	–	–	254	16.9
			% Reduction	–	–	99.60	94.09
6.4	Brown and Dennis	4	Number	*	24	1	1
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.5	Variably dimensioned	2	Number	5	6	1	1
			Ratio	–	0.83	5	5
			% Reduction	–	−20	80	80
6.6	Linear – rank 1	2	Number	*	*	11	11
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.7	Linear – full rank	4	Number	1144	*	1	1
			Ratio	–	–	1144	1144
			% Reduction	–		99.91	99.91
6.8	Extended Rosenbrock	2	Number	15	200	1	1
			Ratio	–	0.08	15	15
			% Reduction	–	−12.33	93.33	93.33
6.9	Discrete boundary value	2	Number	11	8	1	1
			Ratio	–	1.38	11	11
			% Reduction	–	27.27	90.90	90.90
6.10	Brown almost linear	4	Number	*	*	2	7
			Ratio	–	–	–	–
			% Reduction	–	–	–	–
6.11	Chebyquad	4	Number	*	*	*	1
			Ratio	–	–	–	–
			% Reduction	–	–	–	–

$$\text{Ratio} = \frac{\text{Perf. metric with basic algorithm}}{\text{Perf. metric with proposed algorithm}},$$

$$\%\ \text{reduction} = \frac{\text{Perf. metric with basic algorithm} - \text{Perf. metric with proposed algorithm}}{\text{Perf. metric with basic algorithm}} \times 100.$$

Table 6 gives the averages of the ratio and percent reduction over all the test examples, while Table 7 gives the ranking of the CTBMS algorithm for the various performance metrics (a higher rank is assigned to the algorithm with lesser performance metric value).

At the outset, we note that for the considered domains and accuracy, the proposed CTBMS algorithm is able to solve all the test examples, whereas MS, TMS, and TBMS algorithms are able to solve only 54.45, 45.45, and 90.90% of the test examples, respectively. Table 6 shows that

Table 6. Minimum, mean, and maximum of ratios and reductions, with respect to MS algorithm

		Ratio			% Reduction		
Perf. metric	Alg.	Min.	Mean	Max.	Min.	Mean	Max.
Iterations	TMS	0.03	1.32	4.59	−556.52	−224.74	78.23
	TBMS	1.05	12.28	30.17	4.65	69.76	96.69
	CTBMS	3.58	15.83	38.99	72.09	84.75	97.44
Computa-tional time	TMS	0.002	1.61	3.68	−5E4	−1E4	72.86
	TBMS	0.02	2.27	6.87	−4900	−961.18	85.45
	CTBMS	0.05	3.55	6.25	−1750	−268.32	88.12
Maximum list length	TMS	0.06	0.97	2.53	−1600	−493.51	60.49
	TBMS	0.47	6.16	18.87	−115	42.99	94.70
	CTBMS	1.4	7.21	24.56	28.57	65.43	95.93
Final list length	TMS	0.08	0.96	1.54	−20	7.52	35.13
	TBMS	5	244.34	1144	80	93.51	99.91
	CTBMS	5	204.82	1144	80	92.59	99.91

Table 7. Rankings obtained by proposed CTBMS algorithm

	Number of problems			
Performance metric	1st Rank	2nd Rank	3rd Rank	4th Rank
Iterations	9	2	0	0
Computational time	8	3	0	0
Maximum list length	10	1	0	0
Final list length	8	3	0	0

for a majority of the test problems, CTBMS algorithm gives large reductions in maximum list length, number of iterations, and final list length, while it gives improvement in computation time for more than half the test problems. Table 7 shows that the proposed algorithm is able to achieve the 1st rank in 81.81% of the test examples for the iterations metric, in 72.72% of the test examples for the computational time metric, in 90.90% of the test examples for the maximum list length metric, and in 72.72% of the test examples for the final list length metric. Overall, the proposed algorithm is found to be the most efficient one for every performance metric.

7. Conclusion

We presented a novel algorithm for global optimization that combines the sophisticated TB forms, Taylor model, and the simple natural inclusion function. The performance of the proposed algorithm was tested and compared with those of existing MS algorithms on a collection of 11 benchmark problems. Overall, the proposed algorithm was found to be the most efficient one for every performance metric.

References

1. Berz, M. and Hoefkens, J. (2001), COSY INFINITY Version 8.1 Programming Manual. *Technical Report MSUCL-1196* National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI.
2. Berz, M. and Hoffstatter, G. (1998), Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing* 4, 83–97.
3. Cargo, G.T. and Shisha, O. (1966), The Bernstein form of a polynomial. *Journal of Research NBS* 70B, 79–81.
4. Csendes, T. and Ratz, D. (1997), Subdivision direction selection in interval methods for global optimization. *SIAM Journal of Numerical Analysis*, 34, 922–938.
5. Garloff, J. (1993), The Bernstein algorithm. *Interval Computations* 2, 155–168.
6. Garloff, J., and Smith, A.P. (2001), Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Nonlinear Analysis* 47, 167–178.
7. Garloff, J. and Smith, A.P. (2001), Solution of systems of polynomial equations by using Bernstein expansion. In: Alefeld, G., Rump, S., Rohn, J. and Yamamoto, T. (eds.), *Symbolic Algebraic Methods and Verification Methods*. Springer, Germany.
8. Hansen, E. (1992), *Global Optimization using Interval Analysis*. Marcel Dekker.
9. Hoefkens, J. (2001), *Rigorous Numerical Analysis with High-order Taylor Models*. PhD Thesis. Michigan State University, East Lansing, MI. also MSUCL-1217.
10. Ichida, K. and Fujii, Y. (1979), An interval arithmetic method for global optimization. *Computing* 23, 85–97.
11. Kearfott, R.B. (1996), *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht.
12. Kearfott, R.B. and Arazyan, A. (2000), Taylor series models in deterministic global optimization. In: *Proc. 3rd Int. Conf. and Workshop on Automatic Differentiation*, France.
13. Lin, Q. and J.Rokne, G. (1996), Interval approximation of higher-order to the ranges of functions. *Computers Mathematical Applications* 31(7), 101–109.
14. Makino, K. and Berz, M. (1996), Remainder differential algebras and their applications. In: Berz, M., Bischof, C., Corliss, G. and Griewank, A. (eds.), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, pp. 63–75.
15. Malan, S., Milanese, M., Taragna, M., and Garloff, J. (1992), B^3 algorithm for robust performance analysis in presence of mixed parametric and dynamic perturbations. In: *Proc. of the 31st IEEE CDC*, pp. 128–133.
16. Moore, R.E. (1966), *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ.
17. Moore, R.E. (1979), *Methods and Applications of Interval Analysis*. SIAM, Philadelphia.
18. Moore, R.E. and Ratschek, H. (1988), Inclusion functions and global optimization II. *Mathematical Programming* 41, 341–356.
19. More, J.J., Garbow, B.S. and Hillstrom, K.E. (1981), Testing unconstrained optimization software. *ACM Trans. Mathematical Software* 7(1), 17–41.
20. Nataraj, P.S.V. and Kotecha, K. (2002), An algorithm for global optimization using the Taylor–Bernstein form as inclusion function. *Journal of Global Optimization* 24, 417–436.
21. Nataraj, P.S.V. and Kotecha, K. (in press), Global optimization with higher-order inclusion function forms – Part 1: A combined Taylor–Bernstein form. *Reliable Computing*, in press.

22. Ratschek, H. (1985), Inclusion functions and global optimization. *Mathematical Programming* 33, 300–317.
23. Ratschek, H. and Rokne, J. (1984), *Computer Methods for the Range of Functions*. Ellis Horwood Limited, Chichester.
24. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*. Wiley, New York.
25. Zettler, M. and Garloff, J. (1998), Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion. *IEEE Transaction of Automatic Control* 43(3), 425–431.