AN ALGORITHM FOR GLOBAL OPTIMIZATION USING THE TAYLOR-BERNSTEIN FORM AS INCLUSION FUNCTION

P. S. V. NATARAJ AND K. KOTECHA

Abstract. We investigate the use of higher order inclusion functions in the Moore-Skelboe (MS) algorithm of interval analysis (IA) for unconstrained global optimization. We first propose an improvement of the Taylor-Bernstein (TB) form given in [14] which has the property of higher order convergence. We make the improvement so that the TB form is more effective in practice. We then use the improved TB form as an inclusion function in a prototype MS algorithm and also modify the cut-off test and termination condition in the algorithm. We test and compare on several examples the performances of the proposed algorithm, the MS algorithm, and the MS algorithm with the Taylor model of Berz *et al.* [2] as inclusion function. The results of these (preliminary) tests indicate that the proposed algorithm with the improved TB form as inclusion function is quite effective for low to medium dimension problems studied.

Keywords: Bernstein Polynomials, Global Optimization, Interval Analysis

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 $f : \mathbf{X} \to \Re$ be a $m+1$ times differentiable function for some positive integer m. Let $\bar{f}(\mathbf{X})$ denote the set of all values of f on $\mathbf{X}$. We seek global optimization algorithms that are able to efficiently determine arbitrarily good lower bounds for the minimum of $\bar{f}(\mathbf{X})$.

Many algorithms based on interval analysis (IA) are available to solve this global optimization problem, see for example, [9], [12], [24] and the references cited therein. IA methods are usually based on branch and bound techniques, that is, they start from the 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 the MS algorithm is reliable, it is somewhat slow to converge in 'difficult' problems, when inclusion functions of first and sometimes even second orders are used (inclusion functions are defined in section 2.2 below). Faster convergence could possibly be obtained with higher order inclusion functions, and it is of interest in this work to investigate their effectiveness in some such 'difficult' problems.

Our proposed algorithm for global optimization uses Bernstein polynomials for bounding the range of the polynomial obtained from the Taylor form of the objective function f. The Bernstein form has proven to be useful in bounding ranges of multivariate polynomials over given domains, see for example, the works of Garloff *et al.* [5], [6], [25]. We therefore combine the Bernstein algorithm with the Taylor form and use the resulting so-called Taylor-Bernstein form [14] as an inclusion function form of f in the MS algorithm. The Taylor-Bernstein form is an inclusion function form exhibiting high order convergence, and we therefore expect to obtain faster convergence with this form. The Taylor-Bernstein form also allows us to make the cut-off test and termination condition more effective, and we incorporate these modifications in the proposed algorithm.

Systems and Control Engineering Group, EE Department, IIT Bombay 400076, India. Email: nataraj@ee.iitb.ernet.in.

Department of IT, GH Patel College of Engineering and Technology, Vallabh Vidyanagar, 388120, Gujarat, India. Email: kk@gcet.ac.in.

The rest of this paper is organized as follows. In section 2, we give the essentials of the Bernstein form, Taylor form, and the Taylor-Bernstein form [14]. In section 3, we present a new more effective Taylor-Bernstein form. In section 4, we propose our algorithm for global optimization, with the new Taylor-Bernstein form as an inclusion function and modified cut-off test and termination condition. We can also have the Taylor model of Berz *et al.* as an inclusion function form in the MS algorithm as done, for instance, in the preliminary work in [13]. We call such an algorithm as Algorithm TMS below. In section 5, we test and compare the performance of the proposed algorithm with that of Algorithms TMS and MS on six 'difficult' examples. The concluding remarks of this work are given in section 6.

2. Bernstein, Taylor, and Taylor-Bernstein forms

2.1. **The Bernstein Form.** The Bernstein algorithm 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. The salient features of the Bernstein polynomial 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 exact range.

2.1.1. *Notation and definitions.* In this sub-section, we follow the presentation in [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 $I \leq N$ if $0 \leq i_k \leq n_k$, $k = 1, \ldots, l$. With $I = (i_1, \ldots, i_{r-1}, i_r, \ldots, i_{r+1}, \ldots, i_l)$ we associate index $I_{r,k}$ given by $I_{r,k} = (i_1, \ldots, i_{r-1}, i_r + k, \ldots, i_{r+1}, \ldots, i_l)$ where $0 \leq i_r + k \leq n_r$. Also, we write $\binom{N}{I}$ for $\binom{n_1}{i_1} \ldots \binom{n_l}{i_l}$.

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

$$p(\mathbf{x}) = \sum_{I \in S} a_I \mathbf{x}^I, \quad \mathbf{x} \in \Re^l \tag{2.1}$$

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 $x^I \in \Re^I$. We refer to N as the degree of p. The I^{th}Bernstein polynomial of degree N is defined as

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

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

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

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, we can have an algorithm for finding a patch of Bernstein coefficients.

Algorithm Patch : $B(\mathbf{U}) =$Patch$(\mathbf{X}, a_I)$

Inputs: A box $\mathbf{X}$, a polynomial p as in (2.1) of degree N in l-variables with coefficients a_I.

Outputs: A patch $B(\mathbf{U})$ of Bernstein coefficients of p on $\mathbf{U}$.

BEGIN Algorithm

1. Transform the polynomial p (with coefficients a_I) on $\mathbf{X}$ to a polynomial on $\mathbf{U}$. Denote the coefficients of the latter as a'_I .
2. Find the Bernstein coefficients of p on $\mathbf{U}$ as

$$b_I(\mathbf{U}) = \sum_{J \leq I} \frac{\binom{I}{J}}{\binom{N}{J}} a'_J, \quad I \in S$$

3. Return the patch $B(\mathbf{U}) = (b_I(\mathbf{U}))_{I \in S}$.

END Algorithm

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

Lemma 2.1. [3] *: Let p be a polynomial of degree N. 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.

2.1.2. *Bernstein subdivision process.* 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 r-th component direction ($1 \leq r \leq l$) to produce two further subboxes $\mathbf{D}_A$ and $\mathbf{D}_B$ given by

$$\begin{aligned} \mathbf{D}_A &= \left[\underline{d}_1, \bar{d}_1\right] \times \ldots \times \left[\underline{d}_r, m(d_r)\right] \times \ldots \times \left[\underline{d}_l, \bar{d}_l\right] \\ \mathbf{D}_B &= \left[\underline{d}_1, \bar{d}_1\right] \times \ldots \times \left[m(d_r), \bar{d}_r\right] \times \ldots \times \left[\underline{d}_l, \bar{d}_l\right] \end{aligned}$$

Then, the patches $B(\mathbf{D}_A)$ and $B(\mathbf{D}_B)$ can be obtained from $B(\mathbf{D})$ by executing the following algorithm.

Algorithm Subdivision : $[B(\mathbf{D}_A), B(\mathbf{D}_B), \mathbf{D}_A, \mathbf{D}_B] =$ SD$(\mathbf{D}, B(\mathbf{D}), r)$

Inputs: The box $\mathbf{D} \subseteq \mathbf{U}$, its patch $B(\mathbf{D})$, and a component direction r ($1 \leq r \leq l$) in which $\mathbf{D}$ is to be bisected.

Outputs: Subboxes $\mathbf{D}_A$ and $\mathbf{D}_B$, with respective patches $B(\mathbf{D}_A)$ and $B(\mathbf{D}_B)$

BEGIN Algorithm

1. Bisect $\mathbf{D}$ along the r-th component direction to produce the two subboxes $\mathbf{D}_A$ and $\mathbf{D}_B$.

2. Compute patch $B(\mathbf{D}_A)$ as follows.
   (a) Set : $B^{(0)}(\mathbf{D}) \leftarrow B(\mathbf{D})$
   (b) FOR $k = 1, \ldots, n_r$ DO

$$b_I^{(k)}(\mathbf{D}) = \begin{cases} b_I^{(k-1)}(\mathbf{D}) & : i_r < k \\ \frac{1}{2}\left\{ b_{I_{r,-1}}^{(k-1)}(\mathbf{D}) + b_I^{(k-1)}(\mathbf{D}) \right\} & : i_r \geq k \end{cases}$$

   To obtain the new coefficients, we apply formula given above for $i_j = 0, \ldots, n_j$, $j = 1, \ldots, r-1, r+1, \ldots, l$.
   (c) Set : $B(\mathbf{D}_A) \leftarrow B^{(n_r)}(\mathbf{D})$
3. Find patch $B(\mathbf{D}_B)$ from intermediate values in above step, as follows
   (a) FOR $k = 0$ to n_r DO

$$b_{i_1,\ldots,n_r-k,\ldots,i_l}(\mathbf{D}_B) = b_{i_1,\ldots,n_r,\ldots,i_l}^{(k)}(\mathbf{D})$$

   (b) Set : $B(\mathbf{D}_B) \leftarrow (b_I(\mathbf{D}_B))_{I \in S}$
4. RETURN $\mathbf{D}_A$, $\mathbf{D}_B$, $B(\mathbf{D}_A)$ and $B(\mathbf{D}_B)$

END Algorithm

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

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

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

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

$$S_0 = \{0, n_1\} \times \ldots \times \{0, n_l\}$$

The above vertex condition also holds for any subbox $\mathbf{D} \subseteq \mathbf{U}$, see [17]. Combining the tool of Bernstein subdivision and the vertex condition, we can repeatedly improve the bounds till they are exact, i.e., till the vertex condition is satisfied on every subdivision. This leads to the following algorithm for computing exactly the range of p on $\mathbf{X}$.

Algorithm BernsteinPolynomialBounder : $\bar{p}(\mathbf{X}) = \text{Bounder}(\mathbf{X}, a_I)$

Inputs: A box $\mathbf{X}$, a polynomial p as in (2.1) of degree N in l-variables and having coefficients a_I.

Outputs: The exact range $\bar{p}(\mathbf{X})$.

BEGIN Algorithm

1. (Compute patch $B(\mathbf{U})$) Execute Algorithm Patch

$$B(\mathbf{U}) = \text{Patch}(\mathbf{X}, a_I)$$

2. (Initialize lists) Set $\mathcal{L} \leftarrow \{(\mathbf{U}, B(\mathbf{U}))\}$, $\mathcal{L}^{sol} \leftarrow \{\}$.
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 condition on patch) If $(\mathbf{D}, B(\mathbf{D}))$ satisfies the vertex condition in Lemma 2.2, that is, if $\min B(\mathbf{D})$ resp. $\max B(\mathbf{D})$ occurs at some $I \in S_0$, enter the item in list $\mathcal{L}^{sol}$ and return to previous step.
5. (Subdivide and find new patches) Execute Algorithm Subdivision

$$[B(\mathbf{D}_A), B(\mathbf{D}_B), \mathbf{D}_A, \mathbf{D}_B] = \text{SD}(\mathbf{D}, B(\mathbf{D}), r)$$

   where, r is chosen to vary cyclically[1] from 1 to l.

[1] That is, r varies starting from 1 through l, and then again from 1 through l, and so on. Besides cyclical, other strategies for subdivision exist, and their efficiency investigated in [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 exact polynomial range) Compute the exact range $\bar{p}(\mathbf{X})$ as the minimum to maximum over all the second entries of the items present in list $\mathcal{L}^{sol}$.
8. RETURN $\bar{p}(\mathbf{X})$.

END Algorithm

2.2. **The Taylor form.** In this subsection, we first introduce some further notation as in [23]. Let

$$\lambda = \{\lambda_1, \ldots, \lambda_l\}, \quad |\lambda| = \lambda_1 + ... + \lambda_l, \quad \lambda! = \lambda_1! \ldots \lambda_l!, \quad D^{\lambda} f(x) = \frac{\partial^{\lambda_1 + \ldots + \lambda_l} f(x)}{\partial x_1^{\lambda_1} \ldots \partial x_l^{\lambda_l}} \tag{2.2}$$

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 $\mathbf{X} \in I(\Re^l)$. We call a function $F : I(\mathbf{X}) \to I(\Re)$ an inclusion function for f, if $\overline{f}(\mathbf{Y}) \subseteq F(\mathbf{Y})$ for all $\mathbf{Y} \in I(\mathbf{X})$. An inclusion function F for f is said to converge of order α, if $w(F(\mathbf{Y})) - w(\overline{f}(\mathbf{Y})) \leq L w(\mathbf{Y})^{\alpha}$ 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

$$f(\mathbf{x}) = \underbrace{f(\mathbf{c}) + \sum_{|\lambda|=1}^{m} \frac{D^{\lambda} f(c)}{\lambda!} (\mathbf{x} - \mathbf{c})^{\lambda}}_{p(\mathbf{x})} + \underbrace{\sum_{|\lambda|=m+1} \frac{f^{(\lambda)}(\boldsymbol{\xi})}{\lambda!} (\mathbf{x} - \mathbf{c})^{m+1}}_{r(\mathbf{x})}, \ \mathbf{x} \in \mathbf{X} \tag{2.3}$$

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

Assume an inclusion function of the $(m+1)$-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 [14] :

$$F_{Taylor}(\mathbf{X}) = \bar{p}(\mathbf{X}) + R(\mathbf{X}) \tag{2.4}$$

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 [14] show that the Taylor form has convergence order $m+1$.

Theorem 2.3. [14] *Assume that the Taylor form of order m is as defined above. Then,*

$$\begin{aligned} \overline{f}(\mathbf{X}) &\subseteq F_{Taylor}(\mathbf{X}) \\ w(F_{Taylor}(\mathbf{X})) - w(\overline{f}(\mathbf{X})) &= O(w(\mathbf{X})^{m+1}) \end{aligned} \tag{2.5}$$

2.3. **The Taylor-Bernstein form.** 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 exact range of a multivariate polynomial $\bar{p}(\mathbf{X})$. Lin and Rokne [14] proposed an algorithm that uses Bernstein form to find a (generally nonsharp) enclosure of $\bar{p}(\mathbf{X})$, so that the resulting combined form, which we shall call as the *Taylor-Bernstein* form, still possesses the property of $m+1$ convergence order given by (2.5).

We give below 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 (2.7) below, and that a generally nonsharp enclosure of the exact range of the polynomial part p of Taylor expansion is computed and used.

Algorithm LR [14] : $F_{LR}(\mathbf{X}) = \text{LinRokne}(\mathbf{X}, f, m)$

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

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

1. For the given function f, compute (Taylor) coefficients of p in (2.3) 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 [18], [19].
2. Relate the obtained Taylor coefficients to those of the power form in (2.1), and denote the coefficients in this form as a_I .
3. Compute the l-tuple of indices D given by

$$D = (d_1, \ldots, d_l), \text{ where } d_1, \ldots, d_l \geq [1/w(\mathbf{X})]^{m+1} \tag{2.6}$$

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

$$N' = (n'_1, \ldots, n'_l), \quad \text{where } n'_k = \max\{n_k, d_k\}, \quad k = 1, \ldots, l \tag{2.7}$$

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 : $B(\mathbf{U})$ =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

$$B^* = [\min B(\mathbf{U}), \max B(\mathbf{U})] \tag{2.8}$$

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

$$F_{LR}(\mathbf{X}) = B^* + R(\mathbf{X}) \tag{2.9}$$

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

END Algorithm

Lin and Rokne [14] showed that the Taylor-Bernstein form computed in the above algorithm retains the property of $m+1$ convergence order shown by the Taylor form:

Theorem 2.4. [14] *Let $F_{LR}(\mathbf{X})$ be as computed in Algorithm LR. Then,*

$$\begin{aligned} \overline{f}(\mathbf{X}) &\subseteq F_{LR}(\mathbf{X}) \\ w(F_{LR}(\mathbf{X})) - w(\overline{f}(\mathbf{X})) &= O(w(\mathbf{X})^{m+1}) \end{aligned}$$

3. Proposed Taylor-Bernstein form

As seen from (2.6), D becomes large quite quickly as $w(\mathbf{X})$ becomes smaller, leading to high degrees $N' \gg N$ of the Bernstein polynomials in (2.7). As a consequence, the Bernstein step of Algorithm LR becomes very computationally intensive as the domain intervals shrink in widths.

We therefore propose an algorithm that 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.

We further propose to use in step 1 of our algorithm, the Taylor model technique of Berz *et al.* [2], [15] 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.

The algorithm proposed below computes an enclosure for the range of f on $\mathbf{X}$ using the Taylor form of order m and Bernstein polynomials of degree N. We emphasize that the *exact* 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.

Algorithm TB : $F_{TB}(\mathbf{X}) = TB(\mathbf{X}, f, m)$

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

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

1. For the given function f, compute Taylor coefficients of p in (2.3) 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 (2.1), and denote the coefficients in this form as a_I .
3. Find the exact range $\bar{p}(\mathbf{X})$ on $\mathbf{X}$ using Algorithm BernsteinPolynomialBounder :

$$\bar{p}(\mathbf{X}) = \text{Bounder}(\mathbf{X}, a_I) \tag{3.1}$$

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

$$F_{TB}(\mathbf{X}) = \bar{p}(\mathbf{X}) + R(\mathbf{X}) \tag{3.2}$$

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

END Algorithm

It is trivial to show that the Taylor-Bernstein form computed in the proposed algorithm also has the property of $m+1$ convergence order :

Theorem 3.1. *Let $F_{TB}(\mathbf{X})$ be as computed in Algorithm TB. Then,*

$$\begin{aligned} \overline{f}(\mathbf{X}) &\subseteq F_{TB}(\mathbf{X}) \\ w(F_{TB}(\mathbf{X})) - w(\overline{f}(\mathbf{X})) &= O(w(\mathbf{X})^{m+1}) \end{aligned}$$

Proof. From (2.4) and (3.2), F_{TB} is a Taylor form F_{Taylor}. Now apply Theorem 2.3. ∎

4. Proposed Algorithm

We first outline the well-known MS algorithm of IA (the algorithm below is actually the MS algorithm augmented with the monotonicity test and the cut-off test of Ichida and Fujii [11]. For convenience, we however refer to it as just the MS algorithm).

MS Algorithm for Global Optimization [24]

Inputs: The box $\mathbf{X}$, inclusion functions F and F' (we use the natural interval extension, cf. [19]) for $f : \mathbf{X} \to \Re$ 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[2], 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 (cf. Remark 4.1): discard any box $\mathbf{V}^i$ if $0 \notin F_j^{'}(\mathbf{V}^i)$ for any $j \in \{1, 2, ..., l\}$ and $i = 1, 2$.
8. Calculate $F(\mathbf{V}^1)$ and $F(\mathbf{V}^2)$.

[2]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 Algorithms TMS and TBMS described in the sequel.

9. Set $v^i = \min F\left(\mathbf{V}^i\right)$ for $i = 1, 2$.
10. Update the cut-off value z as

$$z = \min\left\{z, \max F\left(\mathbf{V}^1\right), \max F\left(\mathbf{V}^2\right)\right\}$$

11. Remove $(\mathbf{Y}, y)$ from the list L.
12. Add the pairs $\left(\mathbf{V}^1, v^1\right), \left(\mathbf{V}^2, v^2\right)$ 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 $F(\mathbf{Y})$ is less than ε, then print y and EXIT algorithm.
16. Go to Step 5.

END Algorithm

The first pairs $(\mathbf{Y}, y)$ of the list generated by the Algorithm in each iteration are called as the leading pairs, and the $\mathbf{Y}$ as the leading boxes.

Remark 4.1. *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].

4.1. **Algorithm TMS.** In this algorithm, we simply use the Taylor models of Berz *et al.* [2] as inclusion functions in Algorithm MS. As this involves using **T**aylor models in **M**oore-**S**kelboe algorithm, we call this as Algorithm TMS. Algorithm TMS is not new in the literature, and has been proposed and investigated, for instance, in [13] (and perhaps also by Berz *et al.*).

4.2. **Algorithm TBMS.** Consider a leading box $\mathbf{Y}$ in a given iteration of the MS algorithm, and apply Algorithm TB for finding an enclosure of $\overline{f}(\mathbf{Y})$ using $F_{TB}(\mathbf{Y})$. From (3.2) and Theorem 3.1,

$$\overline{f}(\mathbf{Y}) \subseteq F_{TB}(\mathbf{Y}) = \bar{p}(\mathbf{Y}) + R(\mathbf{Y}) = [\min \bar{p}(\mathbf{Y}), \max \bar{p}(\mathbf{Y})] + [\min R(\mathbf{Y}), \max R(\mathbf{Y})]$$

or

$$\overline{f}(\mathbf{Y}) \subseteq [\min \bar{p}(\mathbf{Y}) + \min R(\mathbf{Y}), \max \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y})] \tag{4.1}$$

Since we obtain the exact range of $\bar{p}(\mathbf{Y})$ in Algorithm TB, we can also construct the so-called inner enclosure of $\overline{f}(\mathbf{Y})$ as

$$[\min \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y}), \max \bar{p}(\mathbf{Y}) + \min R(\mathbf{Y})] \subseteq \overline{f}(\mathbf{Y}) \tag{4.2}$$

From (4.1) and (4.2),

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

which implies

$$\min \bar{p}(\mathbf{Y}) + \min R(\mathbf{Y}) \quad \leq \quad \min \overline{f}(\mathbf{Y}) \quad \leq \quad \min \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y})$$

Therefore, with F_{TB} as an inclusion function, we can redefine the cut-off level in the MS algorithm as $z = \min \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y})$ which is obviously less and hence more effective than the original cut-off level of $\max F(\mathbf{Y}) = \max \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y})$. Further, the error on $\min \overline{f}(\mathbf{Y})$ is seen from the above inequality to be no greater than

$$\{\min \bar{p}(\mathbf{Y}) + \max R(\mathbf{Y})\} - \{\min \bar{p}(\mathbf{Y}) + \min R(\mathbf{Y})\} = \max R(\mathbf{Y}) - \min R(\mathbf{Y}) = w(R(\mathbf{Y}))$$

This means that using F_{TB}, we can redefine the termination condition in MS algorithm based on the width of $R(\mathbf{Y})$, which is smaller and hence more effective than the original one based on $w(F(\mathbf{Y})) = w(\bar{p}(\mathbf{Y})) + w(R(\mathbf{Y}))$.

Based on these ideas, in our second proposed algorithm we make the following modifications to the MS algorithm (note that $\mathbf{Y}$ is the leading box in the current iteration) :

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

Since this global optimization algorithm involves using **T**aylor - **B**ernstein form in **M**oore-**S**kelboe algorithm, we call it as Algorithm TBMS.

Algorithm TBMS

Inputs: The box $\mathbf{X}$, order m of the Taylor form to be used, an inclusion function F' (we use the natural interval extension) for the Jacobian of $f : \mathbf{X} \to \Re$, 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 $F_{TB}(\mathbf{Y})$ using Algorithm TB : $[F_{TB}(\mathbf{Y}), \bar{p}(\mathbf{Y}), R(\mathbf{Y})] = TB(\mathbf{Y}, f, m)$
3. Set $y = \min F_{TB}(\mathbf{Y})$.
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 (cf. Remark 4.1): discard any box $\mathbf{V}^i$ if $0 \notin F_j^{'}(\mathbf{V}^i)$ for any $j \in \{1, 2, ..., l\}$ and $i = 1, 2$.
8. Calculate $F_{TB}(\mathbf{V}^1)$ and $F_{TB}(\mathbf{V}^2)$ using Algorithm TB.
9. Set $v^i = \min F(\mathbf{V}^i)$ for $i = 1, 2$.
10. Update the cut-off value z as
$$z = \min\{z, \min \bar{p}(\mathbf{V}^1) + \max R(\mathbf{V}^1), \min \bar{p}(\mathbf{V}^2) + \max R(\mathbf{V}^2)\}$$
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 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 Algorithm TMS as well as that of Algorithm TBMS follow immediately from the convergence results for inclusion functions of higher order in the MS algorithm, as given by Moore and Ratschek in [20] and Ratschek in [22].

5. Numerical Tests

We test and compare the performances of Algorithms TBMS, TMS, and MS on some examples. We use two values of accuracy, $\varepsilon = 10^{-03}$ and 10^{-05}. We choose the following measures for the tests: number of iterations, computational time, space-complexity (maximum list length), and final list length. For all our 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], [10].

Example 1 Gritton's second problem in Chemical Engineering [13]: The function is

$$\begin{aligned} f(x) \quad = \quad & -371.93625 - 791.2465656 * x + 4044.944143 * x^2 + 978.1375167 * x^3 \\ & -16547.8928 * x^4 + 22140.72827 * x^5 - 9326.549359 * x^6 - 3518.536872 * x^7 \\ & +4782.532296 * x^8 - 1281.47944 * x^9 - 283.4435875 * x^{10} + 202.6270915 * x^{11} \\ & -16.17913459 * x^{12} - 8.88303902 * x^{13} + 1.575580173 * x^{14} + 0.124590848 * x^{15} \\ & -0.03589148622 * x^{16} - 0.0001951095576 * x^{17} + 0.0002274682229 * x^{18} \end{aligned}$$

This is a unidimensional problem, with $l = 1$. We took the initial domain as $\mathbf{X} = [1, 2]$. Algorithm MS is unable to provide a solution, even after 1 hour, and is therefore aborted. The performances of Algorithms TMS and TBMS are given in the below Table[3].

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	50	0.41	37	1	58	0.32	47	2
	10^{-05}	52	0.43	37	1	64	0.34	47	1
4	10^{-03}	7	0.08	6	1	18	0.14	8	2
	10^{-05}	9	0.11	6	1	24	0.18	8	1
6	10^{-03}	3	0.06	2	1	17	0.17	7	2
	10^{-05}	4	0.06	2	1	23	0.21	7	1
8	10^{-03}	2	0.04	2	1	17	0.20	7	2
	10^{-05}	3	0.06	2	1	23	0.26	7	1

Both Algorithms TMS and TBMS are able to find the global minimum fairly quickly, as -0.11811..... It may be noted that Kearfott and Arazyan report in [13] that the software GLOBSOL had some difficulty in tackling this problem.

Example 2 Jennrich and Sampson function [21, problem 6]. The two dimensional function is

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

We take the initial domain as $\mathbf{X} = ([-1, 1], [-1, 1])$. The performances of the various Algorithms are as under.

[3]In the Tables to follow, LL denotes list length. A star entry denotes that the algorithm has been aborted due to the excessive computation time taken (greater than an hour).

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	136	1.75	18	3	354	1.56	32	23
	10^{-05}	141	2.00	18	2	432	2.00	32	24
4	10^{-03}	62	1.20	14	1	349	3.11	32	23
	10^{-05}	65	1.45	14	1	427	3.95	32	24
6	10^{-03}	53	1.66	14	1	349	6.24	32	23
	10^{-05}	55	1.81	14	1	427	7.84	32	24
8	10^{-03}	29	1.66	10	1	349	11.35	32	23
	10^{-05}	31	1.83	10	1	427	14.12	32	24

MS				
Accuracy	Iterations	Time, s	Max. LL	Final LL
10^{-03}	1443	6.71	81	42
10^{-05}	1961	10.20	81	37

The global minima is found in each case as 124.36218.....

Example 3 Bard function [21, problem 8]. The three dimensional function is

$$f(x) = \sum_{i=1}^{15} f_i(x)^2, \quad f_i(x) = y_i - \left(x_1 + \frac{u_i}{v_i x_2 + w_i x_3} \right), u_i = i, v_i = 16 - i, w_i = \min(u_i, v_i)$$

where,

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_i	0.14	0.18	0.22	0.25	0.29	0.32	0.35	0.39	0.37	0.58	0.73	0.96	1.34	2.10	4.39

We take the initial domain as $([-0.25, 0.25], [0.01, 2.5], [0.01, 2.5])$ .The performances of the various Algorithms are as under.

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	406	16.64	74	45	3145	76.13	822	772
	10^{-05}	520	32.13	74	7	$*$	$>$ 3600	$*$	$*$
4	10^{-03}	191	35.00	38	7	3124	86.13	818	772
	10^{-05}	202	60.65	38	1	$*$	$>$ 3600	$*$	$*$
6	10^{-03}	162	67.80	38	2	3123	122.81	818	772
	10^{-05}	165	90.22	38	1	$*$	$>$ 3600		
8	10^{-03}	157	79.90	38	2	3122	181.05	818	772
	10^{-05}	159	92.03	38	1	$*$	$>$ 3600	$*$	$*$

MS				
Accuracy	Iterations	Time, s	Max. LL	Final LL
10^{-03}	6122	466.56	1643	1622
10^{-05}	$*$	$>$ 3600	$*$	$*$

The global minima found using each of the algorithms is $8.21487....E-03$.

Example 4 Multidimensional function of Makino and Berz [16, first example]. The function is

$$f(x) = \frac{4\tan(3x_2)}{3x_1 + x_1\sqrt{\frac{6x_1}{-7(x_1-8)}}} - 120 - 2x_1 - 7x_3(1+2x_2) - \sinh(0.5 + \frac{6x_2}{8x_2+7}) + \frac{(3x_2+13)^2}{3x_3} - 20x_3(2x_3-5) + \frac{5x_1\tanh(0.9x_3)}{\sqrt{5x_2}} - 20x_2\sin(3x_3)$$

This is a three dimensional problem with $l = 3$. We took the initial domain as given in the paper cited, i.e.,

$$\mathbf{X} = ([1.95, 2.05], [0.95, 1.05], [0.95, 1.05])$$

Algorithm MS is unable to provide a solution even after 1 hour, and is aborted. The performances of Algorithms TMS and TBMS are as under.

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	5	0.12	2	1	44	0.17	15	12
	10^{-05}	15	0.30	2	1	64	0.30	15	11
4	10^{-03}	0	0.04	1	1	44	0.47	15	12
	10^{-05}	2	0.07	2	1	64	0.64	15	11
6	10^{-03}	0	0.04	1	1	44	0.95	15	12
	10^{-05}	0	0.05	1	1	64	1.35	15	11
8	10^{-03}	0	0.06	1	1	44	1.75	15	12
	10^{-05}	0	0.08	1	1	64	2.46	15	11

The global minimum is given in the above cited paper as $-2.31166.....$ Algorithms TMS and TBMS are able to find this global minimum successfully and quickly.

Example 5 Brown and Dennis function [21, problem 16]. The function is

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

This is a $4-$dimensional problem. Following [8], we take the initial domain as $\mathbf{X} = ([-10, 0, -100, -20], [100, 15, 0, 0.2])$. Algorithm MS is unable to provide a solution, even after 1 hour, and is aborted. The performances of Algorithms TMS and TBMS are as under.

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	250	35.95	23	1	418	2.53	40	18
	10^{-05}	259	85.18	23	1	476	3.09	45	24
4	10^{-03}	66	7.08	15	1	397	4.38	40	18
	10^{-05}	66	7.46	15	1	455	5.35	45	24
6	10^{-03}	47	29.67	15	1	397	5.37	40	18
	10^{-05}	47	31.23	15	1	455	6.64	45	24
8	10^{-03}	40	113.36	15	1	397	6.67	40	18
	10^{-05}	40	116.57	15	1	455	8.14	45	24

The global minimum is given in the above cited paper as 88860.47976.... Algorithms TMS and TBMS are able to find this global minimum successfully and fairly quickly, for $m \geq 4$.

Example 6 Kowalik and Osborne function [21, problem 15]. The function is

$$f(x) = \sum_{i=1}^{11} f_i(x)^2, \quad f_i(x) = y_i - \frac{x_1(u_i^2 + u_i x_2)}{(u_i^2 + u_i x_3 + x_4)}$$

where,

i	1	2	3	4	5	6	7	8	9	10	11
y_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
u_i	4.0000	2.0000	1.0000	0.5000	0.2500	0.1670	0.1250	0.1000	0.0833	0.0714	0.0625

The global minima of this function over domain $([0.1, 0.2], [0.1, 0.2], [0.1, 0.2], [0.1, 0.2],)$ is $1.02734E-03$. The performances of the various Algorithms are as under.

		TBMS				TMS			
Order, m	Accuracy	Iterations	Time, s	Max. LL	Final LL	Iterations	Time, s	Max. LL	Final LL
2	10^{-03}	33	21.57	5	5	422	2.47	219	219
	10^{-05}	$*$	> 3600	$*$	$*$	$*$	> 3600	$*$	$*$
4	10^{-03}	6	16.49	5	5		5.60	211	211
	10^{-05}	23	571.60	14	1	$*$	> 3600	$*$	$*$
6	10^{-03}	0	7.16	1	1		12.28	211	211
	10^{-05}	8	822.73	6	2	$*$	> 3600	$*$	$*$
8	10^{-03}	0	11.60	1	1		24.31	211	211
	10^{-05}	6	1294.16	5	1	$*$	> 3600	$*$	$*$

MS				
Accuracy	Iterations	Time, s	Max. LL	Final LL
10^{-03}	204	0.25	174	173
10^{-05}	$*$	> 3600	$*$	$*$

5.1. **Discussion.** Based on the results of the above preliminary tests, we make some general observations.

Algorithm MS: takes excessive computation time in all examples, except example 2, and for lower accuracy in examples 3 and 6. Algorithm MS generally requires more time, iterations and list lengths than Algorithms TMS and TBMS.

Algorithm TMS: gives the results in reasonable computational time in all examples for lower accuracy, but needs excessive time in two examples for higher accuracy (examples 3 and 6). The computational time increases as the order m increases. An interesting feature is that the number of iterations remains almost the same for $m \geq 4$, though in a few examples there is a drop in this number when m is increased from 2 to 4. The same holds for the space-complexity and final list length.

Algorithm TBMS: the number of iterations decreases as the order m is increased. A considerable reduction is obtained between $m = 2$ and 4. The maximum list length also decreases considerably between $m = 2$ and 4, but decreases little thereafter. In most examples (examples 2, 4, 5, 6), the computational time first decreases, then increases with m, with the least time required for $m = 4$.

Algorithm TBMS is much faster than Algorithm TMS in all examples, except example 5. The speed-up is about $3 - 4$ times in Examples 1, 2, and 6, and is as high as $10 - 40$ in examples 3 and 4. The speed-up gets better with accuracy. In all examples, Algorithm TBMS requires much smaller list lengths and much lesser number of iterations than Algorithm TMS.

6. Concluding Remarks

In summary, the preliminary tests indicate that Algorithms TMS and TBMS are quite effective compared to Algorithm MS, for lower accuracy problems. For higher accuracy problems, Algorithm TBMS is the most effective one.

The best overall choice, in terms of the number of iterations, space-complexity, and speed seems to be Algorithm TBMS with a medium Taylor order $m = 4$.

Acknowledgments

The authors wish to sincerely thank Drs. Berz, Makino, and Hoefkens for providing the COSY-INFINITY software and extending a lot of help regarding the usage of the software. The authors also wish to thank Dr. Hoefkens for carefully reading the paper and offering various suggestions that improved the presentation of the paper. The second author would also like to thank Dr. N. D. Jotwani of GCET for motivating, encouraging and providing the required facilities for this work.

References

[1] M. Berz and J. Hoefkens. COSY INFINITY Version 8.1 Programming Manual. Technical Report MSUCL-1196, National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, 2001.

[2] M. Berz and G. Hoffstatter. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4:83–97, 1998.

[3] G. T. Cargo and O. Shisha. The Bernstein form of a polynomial. *Jl. of research of NBS*, 70B:79–81, 1966.

[4] T. Csendes and D. Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Jl. numerical analysis*, 34:922–938, 1997.

[5] J. Garloff. The Bernstein algorithm. *Interval Computations*, (2):155–168, 1993.

[6] J. Garloff and A. P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. In *Proc. of the 3rd World Congress of Nonlinear Analysts (WCNA 2000)*, Catania, Sicily, Italy, 2000.

[7] J. Garloff and A. P. Smith. Solution of systems of polynomial equations by using Bernstein expansion. In G. Alefeld, S. Rump, J. Rohn, and T. Yamamoto, editors, *Symbolic Algebraic Methods and Verification Methods*. Springer, Germany, 2001.

[8] D. M. Gay. A trust region approach to linearly constrained optimization. In D. F. Griffiths, editor, *Numerical Analysis, Lecture Notes in Mathematics 1066*. Springer, Berlin, 1984.

[9] E. Hansen. *Global optimization using interval analysis*. Marcel Dekker, 1992.

[10] J. Hoefkens. *Rigorous Numerical Analysis with High-Order Taylor Models*. PhD thesis, Michigan State University, East Lansing, Michigan, USA, 2001. also MSUCL-1217.

[11] K. Ichida and Y. Fujii. An interval arithmetic method for global optimization. *Computing*, 23:85–97, 1979.

[12] R. B. Kearfott. *Rigorous global search: continuous problems*. Dodrecht: Kluwer Academic Publishers, 1996.

[13] R. B. Kearfott and A. Arazyan. Taylor series models in deterministic global optimization. In *Proc. 3rd Int. Conf. and Workshop on Automatic Differentiation*, June 2000.

[14] Q. Lin and J. G. Rokne. Interval approximation of higher order to the ranges of functions. *Computers Math. Applic.*, 31(7):101–109, 1996.

[15] K. Makino and M. Berz. Remainder differential algebras and their applications. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational differentiation: techniques, applications, and tools*, pages 63–75. SIAM, 1996.

[16] K. Makino and M. Berz. Efficient control of the dependency problem based on Taylor model methods. *Reliable Computing*, 5:3–12, 1999.

[17] S. Malan, M. Milanese, M. Taragna, and J. Garloff. B^3 algorithm for robust performance analysis in presence of mixed parametric and dynamic perturbations. In *Proc. of the 31st IEEE CDC*, pages 128–133, 1992.

[18] R. E. Moore. *Interval analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[19] R. E. Moore. *Methods and applications of interval analysis*. SIAM, Philadelphia, 1979.

[20] R. E. Moore and H. Ratschek. Inclusion functions and global optimization II. *Mathematical programming*, 41:341–356, 1988.

[21] J. J. More, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Mathematical Software*, 7(1):17–41, 1981.

[22] H. Ratschek. Inclusion functions and global optimization. *Mathematical Programming*, 33:300–317, 1985.

[23] H. Ratschek and J. Rokne. *Computer methods for the range of functions*. Chichester: Ellis Horwood Limited, 1984.

[24] H. Ratschek and J. Rokne. *New computer methods for global optimization*. New York: Wiley, 1988.

[25] M. Zettler and J. Garloff. Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion. *IEEE Trans. on Automat. Control*, 43(3):425–431, 1998.