J Glob Optim (2009) 45:403–426
DOI 10.1007/s10898-008-9382-y

An efficient algorithm for range computation of polynomials using the Bernstein form

Shashwati Ray · P. S. V. Nataraj

Received: 19 January 2008 / Accepted: 22 November 2008 / Published online: 10 December 2008

Abstract We present a novel optimization algorithm for computing the ranges of multivariate polynomials using the Bernstein polynomial approach. The proposed algorithm incorporates four accelerating devices, namely the *cut-off* test, the *simplified vertex* test, the *monotonicity* test, and the *concavity* test, and also possess many new features, such as, the generalized matrix method for Bernstein coefficient computation, a new subdivision direction selection rule and a new subdivision point selection rule. The features and capabilities of the proposed algorithm are compared with those of other optimization techniques: interval global optimization, the filled function method, a global optimization method for imprecise problems, and a hybrid approach combining simulated annealing, tabu search and a descent method. The superiority of the proposed method over the latter methods is illustrated by numerical experiments and qualitative comparisons.

Keywords Bernstein polynomials · Global optimization · Interval analysis · Polynomial optimization · Range computation

1 Introduction

Polynomial optimization problems arise in the mathematical modelling of several real world applications. Knowledge of the range of a polynomial in several variables on a multidimensional rectangle is relevant for numerous investigations and applications in numerical and functional analysis, combinatorial optimization, and finite geometry.

To find the globally optimal solutions of polynomial programs, several existing methods use linearization techniques to approximate polynomial terms [19]. Since linearizations are only approximations, accuracy (i.e., sharpness) is less important than efficiency [13]. A few algorithms for global optimizations of multivariate polynomial functions use some

S. Ray · P. S. V. Nataraj (✉)
Systems and Control Engineering Group, ACRE Building, Indian Institute of Technology, Bombay 400 076, India
e-mail: nataraj@sc.iitb.ac.in

relaxation techniques (such as linear matrix inequalities), which are further modifications of the linearization techniques. These techniques involve sum of squares and semidefinite programming [15,17]. The accuracy of the solutions depends heavily on the relaxation order. When the relaxation order increases, the number of relaxed variables increases, hence the overall computational time increases very quickly.

In contrast, interval arithmetic based optimization algorithms [14,18] are based on branch and bound methods. They always obtain guaranteed bounds, but tend to be slow [13,34]. Exact algebraic techniques for solving polynomial programming problems find all the critical points, and then identify the smallest value of the polynomial at any critical point. Such techniques include Gröbner bases, eigenvalue method, resultants and discriminants and numerical homotopy methods [3,24], but these may be computationally expensive and the number of critical points could be infinite.

Another method for computing the range of a polynomial is obtained through the use of Bernstein forms; these are intimately connected to Bernstein polynomials [20]. Range analysis using the Bernstein form does not require function evaluations. The method relies on the simple idea that if a polynomial is written in the Bernstein basis [20] over a box, the range of the *multivariate* polynomial is bounded by the values of the minimum and maximum Bernstein coefficients [1,4,5]. The key feature of the Bernstein approach to range computations is that bounds on the global optima are *guaranteed*. The approach does not require any initial guess for starting the optimization, but only an initial search box bounding the domain of interest.

Polynomial optimization using the Bernstein approach needs transformation of the given multivariate polynomial from its power form into its Bernstein form, and subsequently computation of the Bernstein coefficients. The range enclosure obtained using the Bernstein coefficients can be sharpened by subdividing the domain box at an appropriate point along a suitable direction. Irrelevant boxes where the global optimizers do not lie, can be efficiently discarded using the so-called *acceleration* devices, thus further avoiding unnecessary subdivisions. Thus, if efficient methods for Bernstein coefficient computation, efficient rules for subdivision, and powerful acceleration devices become available, then these could be used for the development of a new algorithm for range computation that incorporates all these tools.

Garloff et al. proposed affine and convex relaxations in a branch and bound framework. They construct affine lower bound functions for multivariate polynomials based on Bernstein expansion [8,9,29]. These lower bound functions make the relaxed problem a linear programming problem. An advantage is that the bound functions can be constructed with a guarantee even in the presence of rounding errors.

Motivated by the above works using the Bernstein form, we first propose certain devices, namely the *cut-off* test, the *simplified vertex* test, the *monotonicity* test, and the *concavity* test to delete certain boxes where the global optimizers surely cannot lie. Although, the ideas of *cut-off* test, *monotonicity* test and *concavity* test are borrowed from the interval analysis literature, for instance, [14,22], the accelerating devices introduced are new in the context of computing polynomial ranges with the Bernstein approach. Similar ideas of *cut-off* test and *monotonicity* test have also been used in [23] to find the global minima of a function. We then develop a global optimization algorithm based on the Bernstein approach for efficient determination of the ranges and the optimizers of multivariate polynomials on general box-like domains. Finally, we conduct numerical experiments to test and compare the performance of the proposed algorithm with those of some other recent approaches, on several standard polynomial problems of different dimensions.

The rest of the paper is organized as follows. In Sect. 2, we give the notations and definitions of the Bernstein polynomials. In Sect. 3, we give the associated algorithms needed for polynomial range finding using the Bernstein approach. Here we also give a detailed description of each of the proposed accelerating device. In Sect. 4, we present the proposed algorithm for polynomial range finding based on the Bernstein approach using these associated algorithms. In Sect. 5, we first numerically test and compare the performance of the proposed algorithm with that of the interval global optimization package GlobSol [18], and then discuss and compare the qualitative features using their numerical results for three other classes of global optimization techniques vis-a-vis those of the proposed algorithm. In Sect. 6, we give the conclusions of the present work.

2 Bernstein form

Following the notations in [7], let $l \in \mathbb{N}$ be the number of variables and $x = (x_1, x_2, \ldots, x_l) \in \mathbb{R}^l$. Define a multi-power x as $x^I = (x_1^{i_1}, x_2^{i_2}, \ldots, x_l^{i_l})$ and a multi-index of maximum degrees N as $N = (n_1, n_2, \ldots, n_l)$ and associate the index $N_{r,-k} = (n_1, \ldots, n_{r-1}, n_r - k, n_{r+1}, \ldots, n_l)$, where $0 \le n_r - k \le n_r$. Further, define a multi-index I as $I = (i_1, i_2, \ldots, i_l) \in \mathbb{N}^l$ and associate the 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 $0 \le i_r + k \le n_r$. Inequalities $I \le N$ for multi-indices are meant component-wise, where $0 \le i_k \le n_k, k = 1, 2, \ldots, l$. Also, write $\binom{N}{I}$ for $\binom{n_1}{i_1}, \ldots, \binom{n_l}{i_l}$.

Let $\mathbf{x} = [\underline{x}, \overline{x}], \overline{x} \ge \underline{x}$ be a real interval, where $\underline{x} = \inf \mathbf{x}$ is the infimum, and $\overline{x} = \sup \mathbf{x}$ is the supremum of the interval $\mathbf{x}$. The width of the interval $\mathbf{x}$ is defined as $\operatorname{wid} \mathbf{x} = \overline{x} - \underline{x}$. For an l-dimensional interval vector or box $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l)$, the width of $\mathbf{x}$ is $\operatorname{wid} \mathbf{x} = (\operatorname{wid} \mathbf{x}_1, \operatorname{wid} \mathbf{x}_2, \ldots, \operatorname{wid} \mathbf{x}_l)$.

An l-variate polynomial p of degree N is written in the power form as

$$p(x) = \sum_{I \le N} a_I x^I, a_I \in \mathbb{R}, x = (x_1, x_2, \ldots, x_l) \in \mathbb{R}^l. \tag{1}$$

We can expand the multivariate polynomial in (1) into Bernstein polynomials over an l-dimensional box $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_l})$. Without loss of generality, we consider the unit box $\mathbf{u} = [0, 1]^l$, since any nonempty box $\mathbf{x}$ of $\mathbb{R}^l$ can be affinely mapped onto $\mathbf{u}$.

The transformation of a polynomial from its power form (1) into its Bernstein form results in

$$p(x) = \sum_{I \le N} b_I(\mathbf{u}) B_{N,I}(x), \quad x \in \mathbf{u}. \tag{2}$$

The coefficients $b_I(\mathbf{u})$ are called the Bernstein coefficients of p over $\mathbf{u}$, and $B_{N,I}(x)$ is called the Ith Bernstein polynomial of degree N defined as

$$B_{N,I}(x) = B_{i_1}^{n_1}(x_1) B_{i_2}^{n_2}(x_2) \ldots B_{i_l}^{n_l}(x_l)$$

where

$$B_{i_j}^{n_j}(x_j) = \binom{n_j}{i_j} x_j^{i_j} (1 - x_j)^{n_j - i_j}, i_j = 0, \ldots, n_j, j = 1, \ldots, l.$$

Each set of coefficients (a_I or b_I) in (1) and (2) can be computed from the other as [2]:

$$a_I = \sum_{J \le I} (-1)^{I-J} \binom{N}{I} \binom{I}{J} b_J$$

$$b_I(\mathbf{u}) = \sum_{J \le I} \frac{\binom{I}{J}}{\binom{N}{J}} a_J, \quad I \le N. \tag{3}$$

The Bernstein coefficients are collected in an array $B(\mathbf{u}) = (b_I(\mathbf{u}))_{I \in S}$, where $S = \{I : I \le N\}$. This array is called a *Bernstein patch*.

Let $\overline{p}(\mathbf{x})$ denote the range of polynomial p on $\mathbf{x}$. Then, by the *range enclosing property* of the Bernstein coefficients [5],

$$\overline{p}(\mathbf{u}) \subseteq [\min B(\mathbf{u}), \max B(\mathbf{u})] \tag{4}$$

The enclosure interval on the right is called the *Bernstein range enclosure*. We can also formulate this property as the *convex hull property* [6] which states

$$conv\,\{(x, p(x)) : x \in \mathbf{u}\} \subseteq conv\,\{(I/N, b_I(\mathbf{u})) : I \in S\} \tag{5}$$

where *conv* denotes the *convex hull* and $(I/N, b_I(\mathbf{u}))$ are the *control points* of p. Thus, the convex hull property states that the graph of p over $\mathbf{u}$ is contained in the convex hull of the control points.

Let S_0 be a special subset of the index set S comprising those indices of the vertices of the array $B(\mathbf{u})$, i.e., let

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

Then, the lower (resp., upper) bound of the Bernstein range enclosure (4) is sharp if and only if $\min b_I(\mathbf{u})_{I \in S}$ (resp., $\max b_I(\mathbf{u})_{I \in S}$) is attained at a Bernstein coefficient $b_I(\mathbf{u})$ with $I \in S_0$. This condition is known as the *vertex condition* [5]. The vertex condition holds also for any subbox $\mathbf{d} \subseteq \mathbf{u}$ [21]. Further, the vertex condition is said to be met *within a given tolerance* ε, if

$$\min_{S_0} B(\mathbf{u}) - \min B(\mathbf{u}) \le \varepsilon \text{ and } \max B(\mathbf{u}) - \max_{S_0} B(\mathbf{u}) \le \varepsilon \tag{6}$$

If we want to tighten the Bernstein range enclosure in (4) when the vertex property does not hold, we subdivide the domain box $\mathbf{x}$ into smaller subboxes, and apply the Bernstein expansion to the polynomial p in (1) on the resulting subboxes. By repeated subdivisions, the Bernstein range enclosure of the given polynomial over a box can be sharpened until they are accurate to the given tolerance.

Let $\mathbf{d}$ be a subbox of $\mathbf{u}$. In the Bernstein form, the first partial derivative of the polynomial p in (1) with respect to $x_r (1 \le r \le l)$ is given by [7]

$$p'_r(x) = \frac{\partial p}{\partial x_r}(x) = n_r \sum_{I \le N_{r,-1}} \left[b_{I_{r,1}}(\mathbf{d}) - b_I(\mathbf{d}) \right] B_{N_{r,-1},I}(x), \quad x \in \mathbf{d} \tag{7}$$

Thus, the Bernstein coefficients b'_I of the first partial derivative of p with respect to x_r can be obtained simply by forming the differences of its successive Bernstein coefficients. Define

$$b'_I(\mathbf{d}) := n_r \left(b_{I_{r,1}}(\mathbf{d}) - b_I(\mathbf{d}) \right) \tag{8}$$

Then,

$$p'_r(x) = \sum_{I \leq N_{r,-1}} b'_I(\mathbf{d}) B_{N_{r,-1},I}(x), \quad x \in \mathbf{d} \tag{9}$$

The array $B'_r(\mathbf{d})$ is used to denote the Bernstein coefficients $\left(b'_I(\mathbf{d})\right)_{I \leq N_{r,-1}}$.

The second partial derivative of p with respect to x_r is given by

$$\begin{aligned}\frac{\partial^2 p}{\partial x_r^2}(x) &= n_r(n_r-1) \sum_{I \leq N_{r,-2}} \left[b_{I_{r,2}}(\mathbf{d}) - 2b_{I_{r,1}}(\mathbf{d}) + b_I(\mathbf{d})\right] B_{N_{r,-2},I}(x), \quad x \in \mathbf{d} \\ &= (n_r-1) \sum_{I \leq N_{r,-2}} \left[b'_{I_{r,1}}(\mathbf{d}) - b'_I(\mathbf{d})\right] B_{N_{r,-2},I}(x), \quad x \in \mathbf{d} \end{aligned} \tag{10}$$

Thus, the Bernstein coefficients b''_I of the second partial derivative of p with respect to x_r can be obtained simply by forming the differences of the successive Bernstein coefficients of the derivative of the polynomial p. Define

$$b''_I(\mathbf{d}) := (n_r - 1)\left(b'_{I_{r,1}}(\mathbf{d}) - b'_I(\mathbf{d})\right) \tag{11}$$

Then,

$$p''_r(x) = \sum_{I \leq N_{r,-2}} b''_I(\mathbf{d}) B_{N_{r,-2},I}(x), \quad x \in \mathbf{d} \tag{12}$$

We use the array $B''_r(\mathbf{d})$ to denote the Bernstein coefficients $\left(b''_I(\mathbf{d})\right)_{I \leq N_{r,-2}}$.

3 Associated algorithms for polynomial range finding

3.1 Bernstein coefficients computation

In the approach called the Matrix method for Bernstein coefficients computation [27], the given polynomial coefficients (irrespective of the dimensionality), are always arranged in the form of a *matrix* instead of a multidimensional array. The size of this matrix depends on the number of variables of the polynomial, and the maximum power of each variable in the polynomial. The computation of Bernstein coefficients then proceeds using only matrix operations such as inverse, multiplication, transpose and reshape. Moreover, the transformation of the polynomial coefficients from a general box-like domain to unit box domain is an inherent part of this method.

The algorithm for computing the Bernstein coefficients based on the Matrix method [27] has many associated algorithms. To save space here, we describe only the call to the algorithm, along with the inputs and the outputs. Further details of the algorithm are available in [27].

Algorithm Bernstein_Matrix: $B(\mathbf{u}) =$ Bernstein_Matrix $(N, l, \mathbf{x}, A)$

Inputs: Degree N of the polynomial p, number of variables l, domain box $\mathbf{x}$ with l components, and coefficient matrix A whose elements are the polynomial coefficients a_I.

Output: A patch $B(\mathbf{u})$ of Bernstein coefficients of p. Note that B is output in the form of a matrix.

3.2 Subdivision direction selection [27]

Suppose we have a list $\mathcal{L}$ of items $(\mathbf{d}, B(\mathbf{d}))$, with the domain box $\mathbf{d}$ and the Bernstein patch $B(\mathbf{d})$. We define a *solution box* as a box for which the vertex condition is satisfied within the specified tolerance ε [cf. (6)]. Let $\mathcal{L}^{sol}$ be a solution list comprising items having solution boxes and their Bernstein patches. Then, we define the *current range estimate* $\widehat{p} = [\inf \widehat{p}, \sup \widehat{p}]$ as the minimum and maximum of all the Bernstein patches in $\mathcal{L}^{sol}$.

In order to select a direction for efficient subdivision, we first select the box having the smallest (or the largest) Bernstein coefficient from the current list $\mathcal{L}$ of working boxes.

An algorithm for this box selection rule is as follows.

Algorithm Select_box: $\mathbf{d}^* =$ Select_box $(\mathcal{L}, \mathcal{L}^{sol}, \widehat{p})$

Inputs: Working list $\mathcal{L}$, solution list $\mathcal{L}^{sol}$, and current range estimate $\widehat{p}$.

Outputs: The box $\mathbf{d}^*$ selected for subdivision.

BEGIN Algorithm

1. From the working list $\mathcal{L}$, choose $\mathbf{d}_1$ and $\mathbf{d}_2$ as those boxes for which the minimum and the maximum values of the Bernstein coefficients are attained, respectively.
2. If $\mathcal{L}^{sol} \neq \emptyset$ then

    1. {Compute distances}
    $dist1 = \inf \widehat{p} - \min B(\mathbf{d}_1)$
    $dist2 = \max B(\mathbf{d}_2) - \sup \widehat{p}$.
    2. {Compare}
    if $dist1 > dist2$ then $\mathbf{d}^* \leftarrow \mathbf{d}_1$, else $\mathbf{d}^* \leftarrow \mathbf{d}_2$.

    else $\mathbf{d}^* \leftarrow \mathbf{d}_1$
3. {Return}
    return $\mathbf{d}^*$.
    END Algorithm

Once we have selected the box $\mathbf{d}^*$, we bisect it in every component direction and find the hull of the Bernstein range enclosure over (both) the resulting subboxes. We then select the direction of subdivision as the one that gives the tightest hull, i.e., which provides the tightest Bernstein range enclosure over both the subboxes. This direction is also taken to be the subdivision direction for all other boxes that are present in $\mathcal{L}$. The philosophy behind the proposed rule is that "the subdivision direction for the *selected box* of the current list $\mathcal{L}$ is likely the suitable subdivision direction for *all other* boxes currently in $\mathcal{L}$".

Based on this rule, we can have an algorithm for selecting the direction of subdivision.

Algorithm Direction_selection_rule_R1: $k =$ Direction_selection_rule_R1 $(\mathcal{L}, \mathcal{L}^{sol}, \widehat{p}, l)$

Inputs: Working list $\mathcal{L}$, solution list $\mathcal{L}^{sol}$, current range estimate $\widehat{p}$, number of variables l.

Outputs: The component direction k selected for subdivision.

BEGIN Algorithm

1. {Selection of box}
    $\mathbf{d}^* =$ Select_box $(\mathcal{L}, \mathcal{L}^{sol}, \widehat{p})$
2. {Iterate to find the hull of Bernstein range enclosure in each component direction}
    for $r = 1$ to l do

    1. Bisect $\mathbf{d}^*$ along coordinate direction r such that
    $\mathbf{d}^* = \mathbf{d}^*_A \cup \mathbf{d}^*_B$.

2. Compute the Bernstein patches $B(\mathbf{d}_A^*)$, $B(\mathbf{d}_B^*)$.
3. Determine

$$d(r) := -\operatorname{wid}\left[\min\left\{\min B(\mathbf{d}_A^*), \min B(\mathbf{d}_B^*)\right\}, \max\left\{\max B(\mathbf{d}_A^*), \max B(\mathbf{d}_B^*)\right\}\right]$$

3. {Find the subdivision direction}
$k := \min\left\{j : j \in \{1, 2, \ldots, l\} \text{ and } d(j) = \max_{r=1}^{l} d(r)\right\}$.
4. {Return}
return k.

END Algorithm

3.3 Subdivision point selection [27]

After choosing a suitable subdivision direction, we next select an appropriate point for subdivision. We propose to subdivide the box at a point close to where the partial derivative of the polynomial becomes equal to zero. The location of this point is estimated using the Bernstein form of the derivative polynomial. If there are many such points, we choose the one where the second partial derivative is maximum. By doing so, we expect the polynomial to thereby become monotonic (in this direction), at least over one of the resulting subboxes. This may lead to satisfaction of the vertex condition on the subbox within the given tolerance. If this turns out to be the case, then the box need not be further subdivided.

Based on this idea we have an algorithm for selecting the subdivision point.

Algorithm Subdivision_point_selection: λ_r = Subdivision_point_selection $(B(\mathbf{d}), r, N_r)$

Inputs: Bernstein coefficients $B(\mathbf{d})$ of the box $\mathbf{d}$ to be subdivided, the selected subdivision direction r, and the multi-index of maximum degrees N of each variable of the polynomial p.

Outputs: Subdivision point λ_r in the rth component direction.

BEGIN Algorithm

1. {Compute Bernstein coefficients $b'_I(\mathbf{d})$ of $p'_r(x)$}
$b'_I(\mathbf{d}) = n_r\left(b_{I_{r,1}}(\mathbf{d}) - b_I(\mathbf{d})\right)$.
2. {Compute differences of the successive Bernstein coefficients $b'_I(\mathbf{d})$ wherever the control polygon of the derivative polynomial changes sign}
Set $\lambda_r := 0.5$
for $I = 0$ to $N_{r,-1}$ do

   1. if $b'_I(\mathbf{d})b'_{I_{r,1}}(\mathbf{d}) < 0$ then
      1. Form set S_c.

$$S_c := \left\{\left\{\begin{pmatrix} I/(N_{r,-1}) \\ b'_I(\mathbf{d}) \end{pmatrix}, \begin{pmatrix} I_{r,1}/(N_{r,-1}) \\ b'_{I_{r,1}}(\mathbf{d}) \end{pmatrix}\right\} : b'_I(\mathbf{d})b'_{I_{r,1}}(\mathbf{d}) < 0\right\}$$

      2. {Check from S_c}
      if $\left|b'_I(\mathbf{d})\right| \neq \left|b'_{I_{r,1}}(\mathbf{d})\right|$ then compute $b''_I(\mathbf{d}) = (n_r - 1)\left(b'_{I_{r,1}}(\mathbf{d}) - b'_I(\mathbf{d})\right)$

3. {From S_c find the location of maximum value of $b''_I(\mathbf{d})$}
Choose that element from S_c for which $\left|b''_I(\mathbf{d})\right|$ is maximum.

4. {Use this element from S_c to compute the subdivision point λ_r in direction r}

$$\lambda_r = \frac{\frac{I}{N_{r,-1}} b'_{I_{r,1}}(\mathbf{d}) - \frac{I_{r,1}}{N_{r,-1}} b'_I(\mathbf{d})}{b'_{I_{r,1}}(\mathbf{d}) - b'_I(\mathbf{d})}, \lambda_r \in [0, 1]$$

5. {Return}
return λ_r.

END Algorithm.

3.4 Accelerating devices

To efficiently discard certain boxes where the global optimizers are sure not to lie, we use certain accelerating devices namely, the cut-off test, the simplified vertex test, the monotonicity test and the concavity test.

3.4.1 The cut-off test

This test is introduced to eliminate certain boxes where the optimizers surely cannot lie, thereby avoiding unnecessary subdivisions. If $\widehat{p}$ is the current range estimate, then we can discard from $\mathcal{L}$ all those entries $(\mathbf{d}, B(\mathbf{d}))$ for which

$$\inf \widehat{p} \leq \min B(\mathbf{d}) \text{ and } \max B(\mathbf{d}) \leq \sup \widehat{p},$$

since these entries do not lead to improvements in the current range estimate $\widehat{p}$.

An algorithm for performing the *cut-off* test is as follows:
Algorithm Cut_off_test: $\mathcal{L} =$ Cut_off_test $(\mathcal{L}, \widehat{p})$
Inputs: Current range estimate $\widehat{p}$, the list $\mathcal{L}$.
Outputs: A pruned list $\mathcal{L}$.
BEGIN Algorithm

1. {Execute for all boxes in the list $\mathcal{L}$}
For each item $(\mathbf{d}, B(\mathbf{d}))$ in $\mathcal{L}$, do the following: if $\inf \widehat{p} \leq \min B(\mathbf{d})$ and $\sup \widehat{p} \geq \max B(\mathbf{d})$ then discard the item $(\mathbf{d}, B(\mathbf{d}))$ from $\mathcal{L}$.
2. {Return}
Output the pruned list $\mathcal{L}$.

END Algorithm

3.4.2 The simplified vertex test

Sometimes, a patch $B(\mathbf{d})$ does not give any improvement in the range enclosure even after repeated subdivisions of $\mathbf{d}$. A *simplified vertex* test is proposed to avoid such subdivisions. Consider an item $(\mathbf{d}, B(\mathbf{d}))$ of the list $\mathcal{L}$ at a given iteration of the algorithm. Suppose that the vertex condition within ε in (6) is satisfied only for $\min B(\mathbf{d})$ but not for $\max B(\mathbf{d})$, i.e., suppose

$$\min_{S_0} B(\mathbf{d}) - \min B(\mathbf{d}) \leq \varepsilon \text{ but } \max B(\mathbf{d}) - \max_{S_0} B(\mathbf{d}) > \varepsilon$$

In such cases, the item would normally continue to be processed in Algorithm Range, as the vertex condition within ε is not fully satisfied for this item. However, further suppose

that for this item we also have max $B(\mathbf{d}) \leq \sup \widehat{p}$. Then, even if the item were to be further processed, it would not lead to any improvement in updating the current range estimate $\widehat{p}$. This is because, over the box $\mathbf{d}$, the minimum of the range within ε has already been obtained as min $B(\mathbf{d})$, while the maximum of the range over this box (an upper bound of which is max $B(\mathbf{d})$) would not lead to an increase in updating the sup $\widehat{p}$, as max $B(\mathbf{d}) \leq \sup \widehat{p}$. Therefore, the item can be removed from list $\mathcal{L}$, and instead deposited in the solution list $\mathcal{L}^{sol}$, where min $B(\mathbf{d})$ would be later used for updating $\widehat{p}$.

A similar logic can be used for the situation where the vertex condition within ε is satisfied for max $B(\mathbf{d})$ but not for min $B(\mathbf{d})$, and where we also have min $B(\mathbf{d}) \geq \inf \widehat{p}$.

Based on this idea, we propose an algorithm for the *simplified vertex* test.

Algorithm Simplified_Vertex_test: $[\mathcal{L}^{sol}, \mathcal{L}] =$ Simplified_Vertex_test $(\widehat{p}, \mathcal{L}, \mathcal{L}^{sol}, \varepsilon)$

Inputs: Current range estimate $\widehat{p}$, the working list $\mathcal{L}$, solution list $\mathcal{L}^{sol}$, and the tolerance ε to which the Bernstein range enclosure is to be found.

Outputs: An updated solution list $\mathcal{L}^{sol}$ and a pruned working list $\mathcal{L}$.

BEGIN Algorithm

1. {Check for all items in the list}
   Do for each item $(\mathbf{d}, B(\mathbf{d}))$ in $\mathcal{L}$

   1. {Check 'vertex condition within ε' for min $B(\mathbf{d})$ and compare max $B(\mathbf{d})$ with current range estimate}
      If min $B(\mathbf{d})$ satisfies 'vertex condition within ε', and if max $B(\mathbf{d}) \leq \sup \widehat{p}$, then delete the item $(\mathbf{d}, B(\mathbf{d}))$ from $\mathcal{L}$ and deposit it in $\mathcal{L}^{sol}$.
   2. {Check 'vertex condition within ε' for max $B(\mathbf{d})$ and compare min $B(\mathbf{d})$ with current range estimate}
      If max $B(\mathbf{d})$ satisfies 'vertex condition within ε', and if min $B(\mathbf{d}) \geq \inf \widehat{p}$, then delete the item $(\mathbf{d}, B(\mathbf{d}))$ from $\mathcal{L}$ and deposit it in $\mathcal{L}^{sol}$.

2. {Return}
   Return the updated solution list $\mathcal{L}^{sol}$ and the pruned working list $\mathcal{L}$.

END Algorithm

3.4.3 The monotonicity test

Another device to avoid unnecessary subdivisions and accelerate the range finding algorithm, is the one based on the monotonicity test for the Bernstein patches. The monotonicity test checks whether p is strictly monotone in an entire subbox $\mathbf{d} \subset \mathbf{u}$, in which case the interior of $\mathbf{d}$ cannot contain a global minimizer or maximizer. Therefore, if

$$0 \notin \overline{p'_r}(\mathbf{d}) \text{ for some } r = 1, \ldots, l$$

then p is strictly monotone over $\mathbf{d}$ with respect to the rth coordinate, and so the interior of $\mathbf{d}$ cannot contain a global extremum point of p. The boundary of $\mathbf{d}$ can still contain a global extremum point of p, if that part of boundary of $\mathbf{d}$ containing extremal polynomial value is also part of the boundary of the original domain box $\mathbf{u}$.

The Bernstein polynomials $B_{N_{r,-1},I}(x)$ in (9) are always non-negative on the interval they are defined. So, if inf $B'_r(\mathbf{d}) > 0$ resp. sup $B'_r(\mathbf{d}) < 0$ then p is monotonically increasing resp. decreasing with respect to direction r on the box $\mathbf{d}$. In either case, the interior of $\mathbf{d}$ cannot contain a global extremum point of p.

If the boundary of $\mathbf{d}$ in any component direction, say rth direction, is also a part of the boundary of $\mathbf{u}$, then $\mathbf{d}$ cannot be discarded if p is monotonic in this direction. In such a case, p can be converted into two $l-1$ variate polynomials by 'freezing' the polynomial at its endpoints in the rth direction. To evaluate the Bernstein coefficients of these two reduced dimension polynomials, we can use the following lemma.

Lemma 1 [35] *Let p be an l-variate polynomial and let $B(\mathbf{d})$ be the patch of its Bernstein coefficients on $\mathbf{d}$. Then, the Bernstein coefficients of p on the m-dimensional faces of $\mathbf{d}$ are just the coefficients on the respective m-dimensional faces of the patch $B(\mathbf{d})$, $0 \leq m \leq l-1$.*

According to the above lemma, the Bernstein coefficients of the two $(l-1)$ dimensional polynomials are given by the Bernstein coefficients (of the l-dimensional polynomial) at the respective faces of $\mathbf{d}$ in the rth component direction.

Therefore, if p is monotonic with respect to some rth direction on the box $\mathbf{d}$, and $\mathbf{d}$ has no interval endpoints in common with the domain box $\mathbf{u}$, then $\mathbf{d}$ can be discarded. But, if $\mathbf{d}$ has an endpoint in common with $\mathbf{u}$ only in the rth component direction, then $\mathbf{d}$ can be discarded only if p is monotonic in any of the remaining component directions, else $\mathbf{d}$ is retained. Further, if p is monotonic in the rth component direction, there is no need to further subdivide $\mathbf{d}$ in this direction, as it would not give any further improvements on the *current range estimate* $\widehat{p}$. Instead, p can be converted to two polynomials of lesser dimension and the above lemma applies. In this way, some unnecessary subdivisions are avoided, leading to the acceleration of the algorithm.

3.4.4 The concavity test

The *concavity* test (or rather the non-convexity test) checks whether the polynomial p is *not* convex on a subbox $\mathbf{d} \subset \mathbf{u}$, in which case $\mathbf{d}$ cannot contain a global minimizer in its interior [12]. Moreover, a global minimizer can only lie on the boundary of $\mathbf{d}$ if that part of the boundary which contains the minimum polynomial value is also a part of the boundary of $\mathbf{u}$.

Consider the computation for the minimum of p. If the polynomial is *convex* in some subbox $\mathbf{d}$ then its Hessian matrix H must be positive semidefinite. A necessary condition for this is that all diagonal elements of the Hessian matrix are non-negative. Therefore, if

$$\overline{H_{rr}}(\mathbf{d}) < 0 \text{ for some } r = 1, \ldots, l$$

then p cannot be *convex* over $\mathbf{d}$, so $\mathbf{d}$ cannot contain a minimizer and $\mathbf{d}$ can be deleted. Analogous logic holds for computing the maximum of p.

In the Bernstein form, the diagonal elements of the Hessian are given by (12). Since the Bernstein polynomials $B_{N_{r,-2},I}(x)$ in (12) are always positive, $\sup B''_r(\mathbf{d}) < 0 \Rightarrow \overline{H_{rr}}(\mathbf{d}) < 0$, i.e., if all $b''_I(\mathbf{d})$ in (12) are negative, then the minimizer of the polynomial cannot lie in $\mathbf{d}$. Similarly, the maximizer of the polynomial cannot lie in $\mathbf{d}$, if all $b''_I(\mathbf{d})$ in (12) are positive, as $\inf B''_r(\mathbf{d}) > 0 \Rightarrow \underline{H_{rr}} > 0$.

We use these tests to discard those boxes lying in the interior of $\mathbf{u}$, where the minimizer and the maximizer are sure not to lie. If $\mathbf{d}$ cannot contain a minimizer, then we can discard it only if we are sure it would not contain a maximizer. Therefore, we can discard $\mathbf{d}$ from $\mathcal{L}$ if, for any r, $\sup B''_r(\mathbf{d}) < 0$ and $\max B(\mathbf{d}) \leq \sup \widehat{p}$. Similarly, if $\mathbf{d}$ cannot contain a maximizer, then we can discard it only if we are sure it would not contain a minimizer. Hence, we can discard $\mathbf{d}$ from $\mathcal{L}$ if, for any r, $\inf B''_r(\mathbf{d}) > 0$ and $\min B(\mathbf{d}) \geq \inf \widehat{p}$.

4 The proposed algorithm Range_Matrix

In the proposed algorithm, the Bernstein coefficients are evaluated only once and stored as a *matrix* instead of the conventional multidimensional array. All the processing in the algorithm (like subdivision, checking vertex property) are carried out on *matrices* rather than on *multidimensional arrays*. Polynomials of any dimension can be more easily handled by this matrix method. The matrices are created from multidimensional arrays (as shown in Fig. 1). The size of the matrix depends on the number of variables of the polynomial, and the maximum power of each variable in the polynomial.

For example, consider a polynomial in three variables x_1, x_2, x_3 in the power form on $\mathbf{u}$

$$p(x) = 2 + 4x_1 + 5x_1^2 - x_2 + 2x_1x_2 + x_1x_3 - x_2x_3 + 6x_1^2x_2x_3 + 2x_3^2 - x_1x_3^2 + x_1^2x_2x_3^2$$

where, $n_1 = 2, n_2 = 1$ and $n_3 = 2$. The coefficients of this polynomial are input to the proposed algorithm as a *matrix* A arranged as

$$A = \begin{pmatrix} a_{000} & a_{010} & a_{001} & a_{011} & a_{002} & a_{012} \\ a_{100} & a_{110} & a_{101} & a_{111} & a_{102} & a_{112} \\ a_{200} & a_{210} & a_{201} & a_{211} & a_{202} & a_{212} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & -1 & 2 & 0 \\ 4 & 2 & 1 & 0 & -1 & 0 \\ 5 & 0 & 0 & 6 & 0 & 1 \end{pmatrix}$$

Similarly, the Bernstein coefficients are computed and stored in the proposed algorithm as a 3×6 *matrix* (instead of a three-dimensional array).

We now describe the details of the proposed algorithm. We first compute the Bernstein coefficients using Algorithm Bernstein_Matrix (see Sect. 3.1). We next initialize a list $\mathcal{L}$ consisting of item $(\mathbf{u}, B(\mathbf{u}))$ with the domain box $\mathbf{u}$ and the Bernstein patch $B(\mathbf{u})$. We also initialize a solution list $\mathcal{L}^{sol}$ to the empty list. From the list $\mathcal{L}$, we then pick each item $(\mathbf{d}, B(\mathbf{d}))$, check for vertex condition satisfaction, and perform the *simplified vertex* test within the specified tolerance ε. The successful items are removed from $\mathcal{L}$ and deposited in the solution list $\mathcal{L}^{sol}$. The *current range estimate* $\widehat{p}$ is then updated as the interval spanned by the minimum and maximum of all the Bernstein patches in $\mathcal{L}^{sol}$. Subsequently, we perform the *cut-off* test, the *monotonicity* test and the *concavity* test to delete irrelevant boxes in the list $\mathcal{L}$ which would not contribute to updating the current range estimate $\widehat{p}$. For the pruned list $\mathcal{L}$, we then select the subdivision direction k for subdivision, based on the direction selection rule R1 (see Sect. 3.2). In the next step, we pick each item from $\mathcal{L}$, delete its entry from $\mathcal{L}$ and find the subdivision point λ_k using the subdivision point rule (see Sect. 3.3). Then, we subdivide the box $\mathbf{d}$ into subboxes $\mathbf{d}_A$ and $\mathbf{d}_B$ and compute the Bernstein patches $B(\mathbf{d}_A)$ and $B(\mathbf{d}_B)$ using the relations given in [7]. The Bernstein coefficients $B(\mathbf{d}_B)$ on the neighboring subbox $\mathbf{d}_B$ are obtained as intermediate values of the computation of $B(\mathbf{d}_A)$. We add the new items $(\mathbf{d}_A, B(\mathbf{d}_A))$ and $(\mathbf{d}_B, B(\mathbf{d}_B))$ to the list $\mathcal{L}$. We continue the entire process till $\mathcal{L}$ becomes

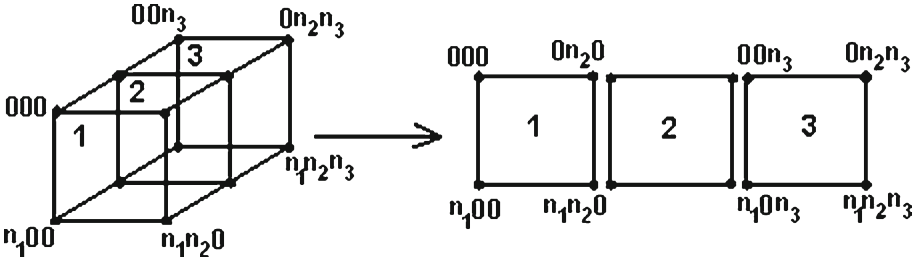

Fig. 1 Two-dimensional or matrix representation of a three-dimensional coefficient array, showing the vertices

empty and there are no more boxes left to be processed. Finally, the desired range enclosure is computed as the *current range estimate* $\widehat{p}$ and the algorithm terminates.

We next present the proposed algorithm for computing the ranges of multivariate polynomials using the Bernstein polynomial approach.

Algorithm Range_Matrix: $\widehat{p} =$ Range_Matrix $(N, A, \mathbf{x}, \varepsilon)$

Inputs: Multi-index N of maximum degrees of each variable of the polynomial, coefficient matrix A whose elements are the polynomial coefficients $a_I \in \mathbb{R}$, the initial box $\mathbf{x} \in \mathbb{IR}^l$, and the tolerance ε to which the Bernstein range enclosure is to be found.

Outputs: An enclosure $\widehat{p}$ of the range of the range of polynomial on $\mathbf{x}$, computed to the specified tolerance ε.

BEGIN algorithm

1. {Compute the Bernstein coefficients}
   $B(\mathbf{u}) =$ Bernstein_Matrix $(N, l, \mathbf{x}, A)$
2. {Initialize lists}
   $\mathcal{L} \leftarrow \{(\mathbf{u}, B(\mathbf{u}))\}$, $\mathcal{L}^{sol} \leftarrow \{\}$.
3. {Start iteration}
   If $\mathcal{L}$ is empty go to step 12.
4. {Check for each box whether vertex condition is met within ε}
   For each item $(\mathbf{d}, B(\mathbf{d}))$ in $\mathcal{L}$, if $(\mathbf{d}, B(\mathbf{d}))$ satisfies *vertex condition within* ε as given by (6), then enter the item in list $\mathcal{L}^{sol}$ and delete its entry from $\mathcal{L}$.
5. {Compute current range estimate}
   Compute $\widehat{p}$ as the interval spanned by the minimum and maximum of the second entries of all the items present in $\mathcal{L}^{sol}$ (i.e., over all the Bernstein patches present in the $\mathcal{L}^{sol}$).
6. {Perform simplified vertex test}
   $[\mathcal{L}^{sol}, \mathcal{L}] =$ Simplified_vertex_test $(\widehat{p}, \mathcal{L}, \mathcal{L}^{sol}, \varepsilon)$.
7. {Perform cut off test}
   $\mathcal{L} =$ Cut_off_test $(\mathcal{L}, \widehat{p})$.
8. {Perform monotonicity test}
   For each item in $\mathcal{L}$, perform the monotonicity test to obtain a pruned list $\mathcal{L}$.
9. {Perform concavity test}
   For each item in $\mathcal{L}$, perform the concavity test to obtain a pruned list $\mathcal{L}$.
10. {Choose a component direction k for subdivision}
    $k =$ Direction_selection_rule_R1 $(\mathcal{L}, \mathcal{L}^{sol}, \widehat{p}, l)$
11. {For each box, find the subdivision point in kth direction and subdivide}
    For $i = 1$ to length $(\mathcal{L})$ do

    a. pick the i^{th} item $(\mathbf{d}, B(\mathbf{d}))$ from $\mathcal{L}$ and delete its entry from $\mathcal{L}$
    b. $\lambda_k =$ Subdivision_point_selection $(B(\mathbf{d}), k, N)$
    c. Subdivide $\mathbf{d}$ at λ_k in component direction k. This generates two subboxes $\mathbf{d}_A$ and $\mathbf{d}_B$ such that $\mathbf{d} = \mathbf{d}_A \cup \mathbf{d}_B$.
    d. Compute $B(\mathbf{d}_A)$ and $B(\mathbf{d}_B)$ (cf. [7]). Enter these new items $(\mathbf{d}_A, B(\mathbf{d}_A))$ and $(\mathbf{d}_B, B(\mathbf{d}_B))$ at the end of list $\mathcal{L}$.

    Go to step 3.
12. {Return}
    Return the current range estimate $\widehat{p}$ as the desired range enclosure of the polynomial p on $\mathbf{x}$.

END Algorithm

5 Comparison with some existing optimization methods

In this section, we compare and discuss the capabilities of the proposed Bernstein polynomial approach with those of some other approaches such as interval global optimization, filled function, combinatorial topology and the hybrid approach. We begin by comparing with the interval global optimization approach.

5.1 Comparison with interval global optimization

Among the most reliable deterministic global optimization methods are interval branch and bound techniques [14,18,22,26]. GlobSol [18] is a well-known self-contained interval global optimization package based on FORTRAN 90 for the solution of constrained and unconstrained global optimization problems with rigor. The special features of GlobSol are that it is based on *interval* branch-and-bound methods, and combines powerful tools such as automatic differentiation, constraint propagation, and specialized interval techniques such as the Interval Newton method [22]. Constraint propagation and the Interval Newton method are used to narrow down the width of the domain box, and an effective point method is used to find approximate feasible points. Also incorporated are a special augmented system mode for least squares problems and an overestimation-reducing "peeling" process for bound-constraints.

GlobSol compiles and runs the user-provided FORTRAN 90 code to produce an internal representation or code list for the objective function, the constraints, and their gradients. The optimization code interprets the derivative code list at run time to perform floating point and interval evaluations of the objective function, gradient, and the Hessian matrix. A separate data file is defined where the user supplies the initial search limits whose coordinates are considered as the bound constraints, along with an initial guess for a global optimizer, if available. The GlobSol configuration file is used to control printing, set tolerances, and select the various algorithm components, such as the choice of preconditioner, choice of the Interval Newton method, form of problem, etc. Performance statistics, both in a report form and as inputs to spreadsheets, are available as outputs of the package.

We test and compare the performances of GlobSol with that of the proposed Algorithm Range_Matrix on 18 polynomial problems, by computing their range enclosures on given domains, to the specified tolerance ε. These 18 problems are taken from [31] and are described in Appendix A. All the code is developed in Forte FORTRAN 95 [30], and all computations are performed on a Sun 440 MHz Ultra Sparc 10 Workstation with 2 GB RAM. All rounding errors are accounted for by using interval arithmetic support provided in the compiler. All the numerical results are obtained with $\varepsilon = 10^{-15}$ for problems of dimension lesser than 6, and with $\varepsilon = 10^{-10}$ for problems of higher dimensions. For computational purposes, we set a specified time limit of 5 h.

Table 1 reports the range enclosures obtained for the 18 problems with the proposed algorithm and GlobSol, along with their abbreviated names and dimensions. In all the problems, the proposed algorithm gives more *accurate* results than GlobSol. However, in one of the problems, GlobSol was unable to compute the range in the time limit of 5 h as indicated by a '$*$' in this table.

For comparing both the methods, we choose the performance metrics as computation time (in seconds) and the total number of boxes processed. For these metrics, we give the values of *ratio* as

Table 1 Polynomial range enclosures obtained with the proposed algorithm, and GlobSol

Ex	Test function	Dim	Range obtained with	
			Range_Matrix	GlobSol
1	L.V. 3	3	[−9.3500000000,14.8000000000]	[−9.3499999999,14.7999999999]
2	R.D. 3	3	[−36.7126906800,10.4056040300]	[−36.7126906799,10.4056040300]
3	L.V. 4	4	[−20.8000000000,22.8000000000]	[−20.7999999999,22.7999999999]
4	Cap 4	4	[−3.1800966258,4.4852773332]	[−3.1800966258,4.4852773332]
5	Wrig 5	5	[−30.25,40.0]	[−30.2499999999,40.0]
6	Reim 5	5	[−5.0,5.0]	[−4.9999999999,4.9999999999]
7	Hun 5	5	[−1436.515078155,161.120543283]	[−1436.5150781550,−161.1205432831]
8	Cyc 5	5	[−30000.0,50000.0]	[−30000.0,∗]
9	Mag 6	6	[−0.25,280.0]	[−0.2499999999,279.9999999999]
10	C. D. 6	6	[−270397.4000000,270202.600000]	[−270397.399999999,270202.599999999]
11	But 6	6	[−1.4393333333,0.2190000000]	[−1.4393333333,0.2189999999]
12	Hair 6	6	[−1875.25,−48.25]	[−1875.2499999999,−48.2500000000]
13	Reim 6	6	[−937501.0000000,937499.000000]	[−937500.999999997,937498.999999997]
14	Mag 7	7	[−0.25,330.0]	[−0.2499999999,329.9999999999]
15	Cyc 7	7	[−5.0,7.0]	[−4.9999999999,6.9999999999]
16	Heart 8	8	[−1.3677500000,1.74345327935]	[−1.3677546999,1.7434485793]
17	Viras 8	8	[−29.0,21.0]	[−28.9999999999,20.9999999999]
18	Cyc 8	8	[−8.0,8.0]	[−7.9999999999,7.9999999999]

$$\frac{\text{Performance metric with basic algorithm}}{\text{Performance metric with proposed algorithm}} \tag{13}$$

and the *percent reduction* computed as

$$= \frac{\text{Performance metric with basic} - \text{Performance metric with proposed algorithm}}{\text{Performance metric with basic algorithm}} \times 100 \tag{14}$$

where the basic algorithm is taken as GlobSol and the proposed algorithm is Algorithm Range_Matrix.

In Table 2, we report the computational times taken by GlobSol and Range_Matrix, respectively in columns 4 and 6. In column 8 of the table we give the ratio of the computational times taken by GlobSol to that of Range_Matrix, whereas in column 10 we give the percent reduction in the computation time of Range_Matrix over that of GlobSol. In the same table, in columns 5 and 7, respectively, we report the number of boxes processed by GlobSol and the proposed algorithm. In columns 9 and 11, we give the ratio and the percent reduction in the number of boxes processed by GlobSol with that of the proposed method Range_Matrix. In the table, '∗' entries indicate that the range could not be computed within the time limit, and '−' entries indicate that no further analysis could be done as a consequence of this.

From Table 2, we observe that concerning the computational time, the proposed algorithm gives reduction in 17 problems, whereas GlobSol gives a reduction in only one problem. Wherever there are improvements, the ratio varies from 1.609 to 239560 and the average ratio is as high as 28282. Similarly, the percent reduction varies from 37.86% to 100% in the

Table 2 Comparison of computational times (in seconds) taken and number of boxes processed by Globsol and by the proposed algorithm

Ex	Test function	Dim	GlobSol		Range_Matrix		Ratio		Percent reduction	
			Time	Boxes	Time	Boxes	Time	Boxes	Time	Boxes
1	L. V. 3	3	0.0500	16	0.0030	3	16.667	5.33	94.00	81.25
2	R. D. 3	3	0.0100	3	0.0027	3	3.704	1.00	73.00	0.00
3	L. V. 4	4	0.3500	108	0.0056	3	62.500	36.00	98.40	97.22
4	Cap 4	4	0.3700	78	0.0680	201	5.441	0.39	81.62	−157.69
5	Wrig 5	5	0.0100	3	0.0032	3	3.125	1.00	68.00	0.00
6	Reim 5	5	7.8300	1844	4.7706	63	1.641	29.27	39.07	96.58
7	Hun 5	5	2.0300	135	2.1078	75	0.963	1.80	−3.83	44.44
8	Cyc 5	5	*	*	0.0010	1	–	–	–	–
9	Mag 6	6	2.1700	498	0.3910	127	5.550	3.92	81.98	74.50
10	C. D. 6	6	2.2400	90	0.0009	1	2392.1	90	99.96	98.89
11	But 6	6	1.1300	125	0.0129	7	87.597	17.86	98.86	94.40
12	Hair 6	6	0.0200	2	0.0020	1	10.249	2	90.24	50.00
13	Reim 6	6	1.2700	128	0.7175	1	1.770	128	43.50	99.22
14	Mag 7	7	5.0500	1459	3.1380	255	1.609	5.72	37.86	82.52
15	Cyc 7	7	108.750	17686	0.0010	1	101160	17686	100.00	99.99
16	Heart 8	8	12.350	438	0.7337	27	16.83	16.22	94.06	93.84
17	Viras 8	8	222.760	21548	0.0016	1	137470	21548	100.00	100.00
18	Cyc 8	8	386.600	40000	0.0016	1	239560	40000	100.00	100.00

17 problems. The average percent reduction (over all the problems) in computational time with the proposed algorithm is 76.28%.

We also observe that concerning the total number of boxes processed, the proposed algorithm gives reduction in 17 problems, whereas GlobSol gives reduction in only one problem. Where there is an improvement, the ratio varies from 1.00 to 40000 and the average ratio is as high as 4680.74. Similarly, the percent reduction varies from 0% to 100% in the 17 problems. The average percent reduction (over all the problems) in the number of boxes with the proposed algorithm is 62.07%.

Thus, we can conclude that the proposed algorithm is overall considerably more efficient than GlobSol in computing the ranges of these 18 polynomial problems.

5.2 Comparison with other approaches

In this subsection, we slightly modify the proposed algorithm Range_Matrix described in the previous section, to find only the global *minimum* value for polynomial programming problems on a closed bounded domain, and as a by-product identify the global minimum points. Using the computational results for seven standard test problems taken from the available global optimization literature and described in Appendix B, we bring forth the qualitative features of the proposed approach as compared to the approaches of Zhang et al. [36], Vrahatis et al. [33] and Salhi and Queen [28].

We first give a brief outline of the latter approaches, referring to the respective algorithms as ZNLT, VST and SQ. We then discuss and compare the features and capabilities of these methods on some test problems.

5.2.1 The filled function approach

A popular global optimization approach called the filled function approach is based on the sequential improvement of local optima [16,25]. This approach uses gradient-type methods coupled with certain auxiliary functions to move from one local minimizer to another better one. The filled function approach finds global minima of multidimensional non-convex functions. The idea behind the approach is to construct a 'filled function' $P(x)$ to move away from the current local minimizer x_1^* of the original objective function $f(x)$, and find a better minimizer x_2^* of $f(x)$ such that $f(x_2^*) < f(x_1^*)$, or to determine that the current local minimizer x_1^* is already a global minimizer of $f(x)$. Ge and Qin [10,11] define a filled function of $f(x)$ at a local minimizer x_1^* of $f(x)$ as a function $P(x)$ that (1) has the maximizer as x_1^*; (2) has no minimizer or saddle point present in any higher basin of $f(x)$; (3) has a minimizer present along the direction $x - x_1^*$.

Zhang et al. [36] propose a new two-parameter (ρ and μ) filled function at a local minimizer x_1^*, for minimizing the objective function $f(x)$. First, a local minimization function is used to find the local minimizers of the objective function. Then, starting at an initial point in the neighborhood of the current local minimizer, the proposed filled function is minimized along the search directions in order to reach a point such that the function value is smaller than the value at the previous local minimizer. Then, this point is used as an initial point in a local search to find a next better local minimizer. The parameters ρ and μ are selected small enough to locate the next better minimizer, else their values are reduced successively. The algorithm is repeated in a two-phase iterative fashion until a global solution is identified (no better local solution can be found). The algorithm, referred to as Algorithm ZNLT below gives a much improved performance in finding a global minimum solution.

Algorithm ZNLT is examined by Zhang et al. on two problems namely Goldstein and Price and Six hump camel functions listed in Appendix B. Apart from the search domain, the inputs required are the values of the parameters ρ and μ needed to build the filled function, a set of m initial points to start the search and minimize the filled function, and the tolerance ε.

From a study of Algorithm ZNLT and the results, we observe that the success of the algorithm depends upon choice of the initial points and a successful initial point would identify only one global minima. Selection of parameters ρ and μ also affect the success rate of locating another better local minima. Nevertheless, the success rate can be increased by increasing the density of initial points. Unfortunately, the growth in number of initial points is exponential with respect to the dimension to the problem. Moreover, the function value has to be evaluated at every new minimizer and compared with that of the previous one. In this way, the computational burden of Algorithm ZNLT could become very high.

In contrast, the proposed Algorithm Range_Matrix does not need any initial point to start with, no other local minimization function is used to locate the minimizers, and no function evaluations are required. The proposed algorithm could also quite easily and successfully extract all the global minimizers in both these problems.

5.2.2 The combinatorial topology approach

Several global optimization techniques consist of a local and a global component. The local component is usually a traditional gradient descent local optimization technique, while the global component is used to search globally the search space, in a complete and exhaustive fashion. In many applications, there are imprecise values for the input data as well as for the function values. Information about the function is obtained in the form of an approximation to the true function value, contaminated by a small amount of noise.

Vrahatis et al. [32,33] propose an algorithm applicable to problems with imprecise function and gradient values. The algorithm makes use of the interval Newton method to roughly isolate the stationary points of the function. A characterization criterion is based upon the gradient values of the function is used, in order to characterize the isolated stationary points as minima, maxima, or saddle points. The characterization criterion implements topological degree theory. The localized minima are computed by applying a real-valued generalized bisection method (which requires only the signs of the gradient values to be correct). This bisection method is a global convergent method and can be applied to a box of arbitrary size that includes a stationary point. The algorithm chooses those points characterized as minima and computes all of them within a given accuracy to obtain the global minimum.

We refer to this algorithm as Algorithm VST. Algorithm VST has been examined by Vrahatis et al. on three problems namely, Himmelblau, Extended Kearfott and Quadratic functions listed in Appendix B.

We note that the success of Algorithm VST depends upon the isolation of *all* the stationary points, which could be very large. To isolate a stationary point, many interval function calls may have to be made. Further, to characterize, localize, and compute the minima, too many real function evaluations may have to be performed including some redundant function evaluations.

Algorithm Range_Matrix found successfully the results for these three problems, too. In comparison, Algorithm Range_Matrix does not need to look for the stationary points as these are generated during the computation of the minimum of the polynomial. Moreover, no function evaluations are required.

5.2.3 The hybrid approach

Salhi and Queen [28] investigated the problem of optimizing possibly non-differentiable functions having several minima. They propose a hybrid approach that combines simulated annealing (SA), tabu search (TS), and a descent method based on a Simplex method. SA and TS are heuristic methods that examine the potential moves from a single starting solution. While SA is used to find the good regions where local minima might exist, the descent method speedily moves the current solution to its local minima, and TS prevents returning to previously visited solutions (which can be local minima already found). When deciding the next move, the descent method relies only on the function values at the vertices of the simplex rather than on the curvature of the function. Hence, the method may fail if a local minimum happens to be in a small region surrounded by areas where the function has values smaller than at the current local minimum. So, these regions are made tabu, i.e., restrict those moves that belong to regions that have already been investigated. Tabu regions are defined as balls, whose radii can vary dynamically and whose centers may be either the local minimum found or the solution obtained by the SA which led to such a local minimum. This technique not only produces the first global minimum, but also detects other global minima along with those local minima which are nearly as good as the global ones.

We refer to this algorithm as Algorithm SQ. Algorithm SQ is been examined by Salhi and Queen on five problems namely, Goldstein and Price function, Rosenbrock functions (2 and 4 variables) and Zakharov functions (2 and 5 variables) listed in Appendix B. To obtain some statistical results, the authors carry out 10 independent trials for each function. Each trial corresponds to a new starting point of the method, which is chosen randomly. The frequency of occurrence of the best local minima may not be always 100%. Thus, there is no guarantee that the global minimum is always found.

Algorithm Range_Matrix also found the results of these five problems successfully. The advantages that Algorithm Range_Matrix has are that no initial points are needed, and no functions evaluations are required. Moreover, the proposed algorithm is guaranteed to find the global minimum, and all the minimizers are extracted with no extra work.

6 Conclusions

We proposed an algorithm for polynomial range finding based on the Bernstein polynomial approach. Our algorithm incorporates many new features, such as, the generalized matrix method for Bernstein coefficient computation, a new subdivision direction selection rule, a new subdivision point selection rule, and four accelerating devices. We compared the efficiency of the proposed algorithm with that of interval global optimization on 18 polynomial problems, on the basis of the computation time and the number of boxes processed. We found the proposed algorithm to be more efficient in all performance metrics. We also illustrated qualitatively the superior features of the proposed algorithm over those of three other optimization techniques: the filled function method, a global optimization method for imprecise problems, and a hybrid approach combining simulated annealing, tabu search and a descent method.

Acknowledgements The authors thank Professor Jürgen Garloff of University of Applied Sciences in Constance, Germany for his comments and suggestions on the work reported in this paper.

Appendix A

Description of test problems

In the following, we list the polynomials p, the domain boxes $\mathbf{x}$, the abbreviated and full names, and the dimensionality of the problems used in our tests. The problems are arranged in the order of increasing dimensionality. All these test problems are from Verschelde's PHC pack [31].

1 **L. V. 3**: A neural network modeled by an adaptive Lotka-Volterra system, $l = 3$

$$p(x_1, x_2, x_3) = x_1 {x_2}^2 + x_1 x_3^2 - 1.1 x_1 + 1$$
$$\mathbf{x}_i = [-1.5, 2], \quad i = 1, 2, 3$$

2 **R. D. 3**: A 3-dimensional reaction diffusion problem, $l = 3$

$$p(x_1, x_2, x_3) = x_1 - 2x_2 + x_3 + .835634534 x_2 (1 - x_2)$$
$$\mathbf{x}_i = [-5, 5], \quad i = 1, 2, 3$$

3 **L. V. 4**: A neural network modeled by an adaptive Lotka-Volterra system, $l = 4$

$$p(x_1, x_2, x_3, x_4) = x_1 x_2^2 + x_1 x_3^2 + x_1 x_4^2 - 1.1 x_1 + 1$$
$$\mathbf{x}_i = [-2, 2], \quad i = 1, \ldots, 4$$

4 **Cap 4**: Caprasse's system: $l = 4$

$$p(x_1, x_2, x_3, x_4) = -x_1x_3^3 + 4x_2x_3^2x_4 + 4x_1x_3x_4^2 + 2x_2x_4^3 + 4x_1x_3 + 4x_3^2 - 10x_2x_4 - 10x_4^2 + 2$$
$$\mathbf{x}_i = [-.5, .5], \quad i = 1, \ldots, 4$$

5 **Wrig 5**: System of A. H. Wright, $l = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = x_5^2 + x_1 + x_2 + x_3 + x_4 - x_5 - 10$$
$$\mathbf{x}_i = [-5, 5], \quad i = 1, \ldots, 5$$

6 **Reim 5**: The 5-dimensional system of Reimer, $l = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = -1 + 2x_1^6 - 2x_2^6 + 2x_3^6 - 2x_4^6 + 2x_5^6$$
$$\mathbf{x}_i = [-1, 1], \quad i = 1, \ldots, 5$$

7 **Hun 5:** The 5-dimensional Hunecke, $l = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = x_2^6x_3 + x_2x_3^6 + x_1^2x_2^4x_5 - 3x_1x_2^2x_3^2x_4x_5 + x_3^4x_4^2x_5 - x_1^3x_3x_4x_5^2 - x_1x_2x_4^3x_5^2 + x_2x_3x_5^5$$
$$\mathbf{x_1} = [0, 1], \mathbf{x_2} = [2, 3], \mathbf{x_3} = [-2, -1], \mathbf{x_4} = [1, 3], \mathbf{x_5} = [-2, -1]$$

8 **Cyc 5**: The cyclic 5-roots problem, $l = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + x_2x_3x_4x_5$$
$$\mathbf{x}_i = [-10, 10], \quad i = 1, \ldots, 5$$

9 **Mag 6**: A problem of magnetism in physics, $l = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2 + x_6^2 - x_6$$
$$\mathbf{x}_i = [-5, 5], \quad i = 1, \ldots, 6$$

10 **C. D. 6**: Camera displacement between two positions, scaled first frame, $l = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = -6.8x_1x_4 - 3.2x_1x_5 + 1.3x_1x_6 + 5.1x_1 - 3.2x_3x_4 - 4.8x_2x_5 - 0.7x_2x_6 - 7.1x_2 + 1.3x_3x_4 - 0.7x_3x_5 + 9.0x_3x_6 - x_3 + 5.1x_4 - 7.1x_5 - x_6 + 2.6$$
$$\mathbf{x}_i = [-100, 100], \quad i = 1, \ldots, 6$$

11 **But 6**: Butcher's problem, $l = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = x_6x_2^2 + x_5x_3^2 - x_1x_4^2 + x_4^3 + x_4^2 - 1/3x_1 + 4/3x_4$$
$$\mathbf{x_1} = [-1, 0], \mathbf{x_2} = [-.1, .9], \mathbf{x_3} = [-.1, .5], \mathbf{x_4} = [-1, -.1], \mathbf{x_5} = [-.1, -.05], \mathbf{x_6} = [-.1, -.03]$$

12 **Hair 6**: Hairer, $l = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = x_3^3x_4 + x_2^3x_5 + x_1^3x_6 - .25$$
$$\mathbf{x}_i = [2, 5], i = 1, 2, 3, \mathbf{x}_i = [-5, -2], \quad i = 4, 5, 6$$

13 **Reim 6**: The 6-dimensional system of Reimer, $l = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = -1 + 2x_1^7 - 2x_2^7 + 2x_3^7 - 2x_4^7 + 2x_5^7 - 2x_6^7$$
$$\mathbf{x}_i = [-5, 5], \quad i = 1, \ldots, 6$$

14 **Mag 7**: Katsura 6, a problem of magnetism in physics, $l = 7$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) &= x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2 + 2x_6^2 + 2x_7^2 - x_1 \\ \mathbf{x}_i &= i = 1, \ldots, 7 \end{aligned}$$

15 **Cyc 7**: The cyclic 7-roots problem, $l = 7$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) &= x_1x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_1x_3x_4x_5x_6x_7 \\ &\quad + x_1x_2x_4x_5x_6x_7 + x_1x_2x_3x_5x_6x_7 + x_1x_2x_3x_4x_6x_7 \\ &\quad + x_1x_2x_3x_4x_5x_7 \\ \mathbf{x}_i &= [-1, 1], i = 1, \ldots, 7 \end{aligned}$$

16 **Heart 8**: Heart-dipole problem, $l = 8$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) &= x_1x_6^3 - 3x_1x_6x_7^2 + x_3x_7^3 - 3x_3x_7x_6^2 + x_2x_5^3 \\ &\quad - 3x_2x_5x_8^2 + x_4x_8^3 - 3x_4x_8x_5^2 + 0.9563453 \end{aligned}$$

$$\begin{aligned} &\mathbf{x_1} = [-.1, .4], \mathbf{x_2} = [.4, 1], \mathbf{x_3} = [-.7, -.4], \mathbf{x_4} = [-.7, .4], \mathbf{x_5} = [.1, .2], \\ &\mathbf{x_6} = [-.1, .2], \mathbf{x_7} = [-.3, 1.1], \mathbf{x_8} = [-1.1, -.3] \end{aligned}$$

17 **Viras 8**: the construction of Virasoro algebras, $l = 8$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) &= -2x_1x_4 + 2x_1x_7 - 2x_2x_5 + 2x_2x_7 - 2x_3x_6 + 2x_3x_7 \\ &\quad + 2x_4x_7 + 2x_5x_7 + 8x_6x_7 - 6x_6x_8 + 8x_7^2 \\ &\quad + 6x_7x_8 - x_7 \\ \mathbf{x}_i &= [-1, 1], \quad i = 1, \ldots, 8 \end{aligned}$$

18 **Cyc 8** : The cyclic 8-roots problem, $l = 8$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) &= x_1x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_8 + x_1x_3x_4x_5x_6x_7x_8 \\ &\quad + x_1x_2x_4x_5x_6x_7x_8 + x_1x_2x_3x_5x_6x_7x_8 \\ &\quad + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5x_7x_8 \\ &\quad + x_1x_2x_3x_4x_5x_6x_8 \\ \mathbf{x}_i &= [-1, 1], \quad i = 1, \ldots, 8 \end{aligned}$$

Appendix B

Additional test problems

In this appendix, we describe the additional polynomial test problems considered in Sect. 5.2.

- **Problem 1: Goldstein price** (2 variables) [36]

$$p(x) = g(x)h(x)$$

where,

$$\begin{aligned} g(x) &= 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \\ h(x) &= 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \end{aligned}$$

The initial search box is

$$\mathbf{x}_1 = [-2, 2], \quad \mathbf{x}_2 = [-2, 2]$$

Global minimum of the function is

$$f(x^*) = 3.0$$

and the global minimizer is

$$x^* = (0, -1)$$

- **Problem 2: Six hump back camel function** (2 variables) [36]

$$p(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

The initial search box is

$$\mathbf{x}_1 = [-3, 3], \quad \mathbf{x}_2 = [-3, 3]$$

Global minimum of the function is

$$f(x^*) = -1.031628453489616$$

and the global minimizers are

$$x_1^* = (0.089842005044578136, -0.71265634021124658)$$
$$x_2^* = (-0.089842005045197792, 0.71265634021270497)$$

- **Problem 3: Himmelblau function** (2 variables) [33]

$$p(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

The initial search box is

$$\mathbf{x}_1 = [-5, 5], \mathbf{x}_2 = [-5, 5]$$

Global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizers are

$$x_1^* = (-2.8044215066473197, 3.131309722462987)$$
$$x_2^* = (-3.779310248573142, -3.283185853881085)$$
$$x_3^* = (3.584428342000785, -1.8481265521645198)$$
$$x_4^* = (3.0, 2.0)$$

- **Problem 4: Extended Kearfott function** (n variables) [33]

$$p(x) = \sum_{i=1}^{n-1} (x_i^2 - x_{i+1})^2 + (x_n^2 - x_1)^2$$

The initial search box is

$$\mathbf{x} = [-2, 2]^n$$

For $n = 4$, the global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizers are

$$x_1^* = (0.0, 0.0, 0.0, 0.0)$$
$$x_2^* = (1.0, 1.0, 1.0, 1.0)$$

- **Problem 5: Quadratic function** (n variables) [33]

$$p(x) = x_1^2 + x_2^2 + \cdots + x_n^2 - r$$

The initial search box is

$$\mathbf{x} = [-99.99, 100]^n$$

For $n = 8$ and $r = -2$, the global minimum of the function is

$$f(x^*) = -2.0$$

and the global minimizer is

$$x^* = (0, 0, 0, 0, 0, 0, 0, 0)$$

- **Problem 6: Rosenbrock function** (n variables) [28]

$$p(x) = \sum_{i=1}^{n-1} \left[100(x_i - x_{i+1})^2 + (x_i - 1)^2\right]$$

The initial search box is

$$\mathbf{x} = [-5, 10]^n$$

For $n = 2$, the global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizer is

$$x^* = (1.0, 1.0)$$

For $n = 4$, the global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizer is

$$x^* = (1.0, 1.0.1.0, 1.0)$$

- **Problem 7: Zakharov function** (n variables) [28]

$$p(x) = \left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$$

The initial search box is

$$\mathbf{x} = [-5, 10]^n$$

For $n = 2$, the global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizer is

$$x^* = (0, 0)$$

For $n = 5$, the global minimum of the function is

$$f(x^*) = 0.0$$

and the global minimizer is

$$x^* = (0, 0, 0, 0, 0)$$

References

1. Berchtold, J., Bowyer, A.: Robust arithmetic for multivariate Bernstein-form polynomials. Comput. Aided Geom. Des. **32**, 681–689 (2000)
2. Berchtold, J., Voiculescu, I., Bowyer, A.: Multivariate Bernstein form polynomials. Technical Report 31/98, School of Mechanical Engineering (1998)
3. Cox, D., Little, J., O'Shea, D.: Using algebraic geometry. In: Graduate Texts in Mathematics, vol. 185. Springer-Verlag, New York (1998)
4. Farouki, R.T., Rajan, V.T.: On the numerical condition of polynomials in Bernstein form. Comput. Aided Geom. Des. **4**, 191–216 (1987)
5. Garloff, J.: The Bernstein algorithm. Interval Comput. **2**, 154–168 (1993)
6. Garloff, J.: The Bernstein expansion and its applications. J. Am. Romanian Acad. **25–27**, 80–85 (2003)
7. Garloff, J., Smith, A.P. : Solution of systems of polynomial equations by using Bernstein expansion. In: Alefeld, G., Rohn, J., Rump, S., Yamamoto, T. (eds.) Symbolic Algebraic Methods and Verification Methods, pp. 87–97. Springer, New York (2001)
8. Garloff, J., Smith, A.P. : A comparison of methods for the computation of affine lower bound functions for polynomials. In: Jermann, C., Neumaier, A., Sam, D. (eds.) Global Optimization and Constraint Satisfaction: 2nd International Workshop, COCOS 2003, Lecture Notes in Computer Science, pp. 71–85. Springer, Berlin (2005)
9. Garloff, J., Jansson, C., Smith, A.P.: Lower bound functions for polynomials. J. Comput. Appl. Math. **157**(1), 207–225 (2003)
10. Ge, R.P., Qin, Y.F.: A class of filled functions for finding global minimizers of a function of several variables. J. Optim. Theory Appl. **54**(2), 241–252 (1987)
11. Ge, R., Qin, Y.: The globally convexized filled functions for global optimization. Appl. Math. Comput. **35**, 131–158 (1990)
12. Hammer, R., Hocks, M., Kulisch, U., Ratz, D.: Numerical Toolbox for Verified Computing I. Springer Verlag, Heidelberg, New York (1993)
13. Hansen, E.R.: Nonlinear equations and optimization. Comput. Math. Appl. **25**(10/11), 125–145 (1993)
14. Hansen, E.R., Walster, G.W.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (2004)
15. Henrion, D., Lasserre, J.B.: Gloptipoly: global optimization over polynomials with Matlab and SeDuMi. ACM Trans. Math. Soft. **29**, 165–194 (2003)
16. Horst, R., Pardalos, P.M.: Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht (1995)
17. Jibetean, D., Laurent, M.: Semidefinite approximations for global unconstrained polynomial optimization. SIAM J. Optim. **16**(2), 490–514 (2005)
18. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer Academic Publishers, Dordrecht (1996)
19. Li, H.L., Chang, C.T.: An approximate approach of global optimization for polynomial programming problems. Eur. J. Oper. Res. **107**, 625–632 (1998)
20. Lorenz, C.G.: Bernstein Polynomials. University of Toronto Press, Toronto (1953)

21. Malan, S., Milanese, M., Taragna, M., Garloff, J.: B3 algorithm for robust performance analysis in presence of mixed parametric and dynamic perturbations. In: Proceedings of the 31st Conference on Decision and Control, pp. 128–133. Tucson, Arizona (1992)
22. Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
23. Nataraj, P.S.V., Kotecha, K.: An improved interval global optimization algorithm using higher order inclusion function forms. J. Global Optim. **32**(1), 35–63 (2005)
24. Parrilo, P.A., Sturmfels, B.: Minimizing polynomial functions. In: Basu, S., Gonzales-Vega, L. (eds.) Algorithmic and Quantitative Real Algebraic Geometry, vol. 60 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science (2003)
25. Pinter, J.D .: Global optimization: software, test problems, and applications. In: Pardalos, P.M., Romeijn, H.E. (eds.) Handbook of Global Optimization, vol. 2, pp. 515–569. Kluwer Academic Publishers, London (2002)
26. Ratschek, H., Rokne, J.: Computer Methods for the Range of Functions. Ellis Horwood, New York (Chichester) (1984)
27. Ray, S.: A new appraoch to range computation of polynomials using the Bernstein form. PhD Thesis, System and Control Engineering, Indian Institute of Technology, Bombay, India (2007)
28. Salhi, S., Queen, N.M.: A hybrid algorithm for detecting global and local minima when optimizing functions with many minima. Eur. J. Oper. Res. **155**, 51–67 (2004)
29. Smith, A.P: Fast construction of constant bound functions for sparse polynomials. J. Global Optim. July (2007, published online)
30. Sun Microsystems, Palo Alto, CA, USA. Forte FORTRAN 95 User Manual (2001)
31. Verschelde, J.: The PHC pack, the database of polynomial systems. Technical Report, University of Illinois, Mathematics Department, Chicago, USA (2001)
32. Vrahatis, M.N.: A generalized bisection method for large and imprecise problems. In: Alefeld, G., Frommer, A., Lang, B. (eds.) Scientifc Computing and Validated Numerics, pp. 186–192. Akademie Verlag, Berlin (1996)
33. Vrahatis, M.N., Sotiropoulos, D.G., Triantafyllou, E.C. : Global optimization for imprecise problems. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds.) Developments in Global Optimization, pp. 37–54. Kluwer, The Netherlands (1997)
34. Wolfe, M.A.: Interval methods for global optimization. Appl. Math. Comput. **75**(2–3), 179–206 (1996)
35. Zettler, M., Garloff, J.: Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion. IEEE Trans. Automatic Control **43**(3), 425–431 (1998)
36. Zhang, L.S., Ng, C.K., Li, D., Tian, W.W.: A new filled function method for global optimization. J. Global Optim. **28**, 17–43 (2004)