RAIRO-Oper. Res. 55 (2021) 3743–3771
https://doi.org/10.1051/ro/2021179

RAIRO Operations Research
www.rairo-ro.org

CONSTRAINED GLOBAL OPTIMIZATION OF MULTIVARIATE POLYNOMIALS USING POLYNOMIAL B-SPLINE FORM AND B-SPLINE CONSISTENCY PRUNE APPROACH

Deepak D. Gawali[1,*], Bhagyesh V. Patil[2], Ahmed Zidna[3] and P. S. V. Nataraj[4]

Abstract. In this paper, we propose basic and improved algorithms based on polynomial B-spline form for constrained global optimization of multivariate polynomial functions. The proposed algorithms are based on a branch-and-bound framework. In improved algorithm we introduce several new ingredients, such as B-spline box consistency and B-spline hull consistency algorithm to prune the search regions and make the search more efficient. The performance of the basic and improved algorithm is tested and compared on set of test problems. The results of the tests show the superiority of the improved algorithm over the basic algorithm in terms of the chosen performance metrics for 7 out-off 11 test problems. We compare optimal value of global minimum obtained using the proposed algorithms with CENSO, GloptiPoly and several state-of-the-art NLP solvers, on set of 11 test problems. The results of the tests show the superiority of the proposed algorithm and CENSO solver (open source solver for global optimization of B-spline constrained problem) in that it always captures the global minimum to the user-specified accuracy.

Mathematics Subject Classification. 90-08.

Received June 7, 2021. Accepted November 30, 2021.

1. Introduction

Generally constrained global optimization of nonlinear programming problems (NLP) is the study of how to find the best (optimum) solution to a problem. The constrained global optimization of NLPs is stated as follows.

$$\min_{x \in \mathbf{x}} f(x) \tag{1.1}$$

$$\text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \ldots, p, \tag{1.2}$$

$$h_j(x) = 0, \quad j = 1, 2, \ldots, q. \tag{1.3}$$

Keywords. Polynomial B-spline, global optimization, polynomial optimization, constrained optimization.

[1] Vidyavardhini's College of Engineering & Technology, Maharashtra, India.

[2] John Deere Technology Centre, Pune, India.

[3] LGIPM, University of Lorraine, Metz, France.

[4] Systems and control Engineering, Indian Institute of Technology Bombay, Maharashtra, India.

*Corresponding author: deepak.gawali@vcet.edu.in

© The authors. Published by EDP Sciences, ROADEF, SMAI 2021

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

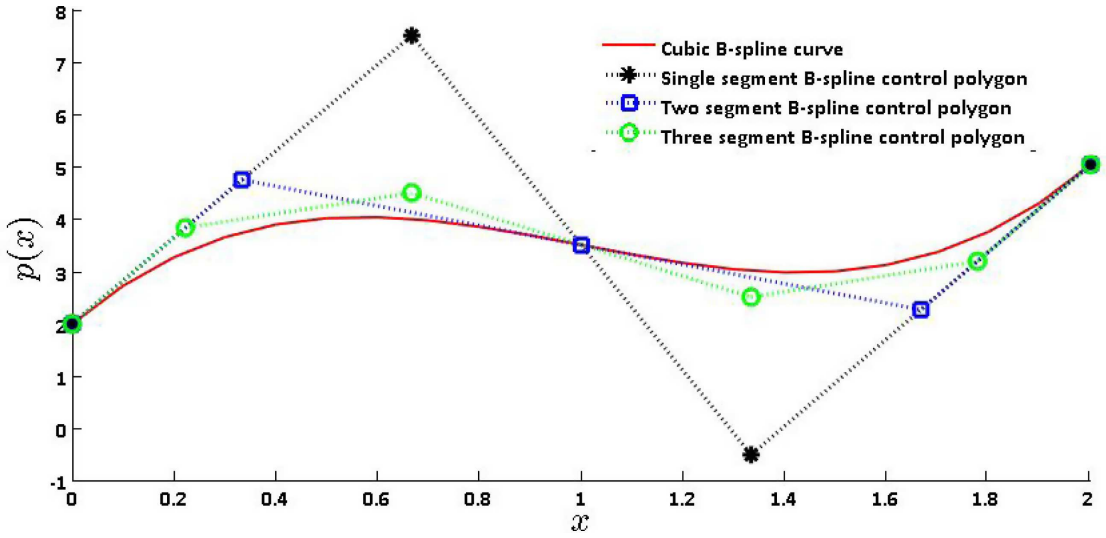

FIGURE 1. Bounds of a univariate polynomial of degree three are improved by increasing the number of B-spline segments.

Branch-and-bound framework is commonly used for solving constrained global optimization problems [13,17]. For instance, several interval methods [14, 18, 19, 35] use this framework to find the global minimum of NLPs. Since interval analysis methods require function evaluation, which leads to a computationally slow algorithm. Compared with these methods the global optimization algorithm of multivariate polynomial using Bernstein form, *e.g.* [26, 27] has the advantage that it avoids function evaluations which might be costly if the degree of the polynomial is high. Global optimization of polynomials using the Bernstein approach needs transformation of the given multivariate polynomial from its power form into its Bernstein form. The minimum and maximum values of the Bernstein coefficients provide lower and upper bounds for the range of polynomial. Generally, this range enclosure (*i.e.* bounds) obtained is overestimated in nature and can be improved by degree elevation. Unfortunately, this process implies the increase of computation time.

In this paper we propose polynomial B-spline as an inclusion function [7,9,11,25]. The minimum and maximum B-spline coefficients provide lower and upper bounds for the range of polynomial. The range enclosure obtained using the polynomial B-spline form can be sharpened by increasing the number of B-spline segments, *i.e.* without degree elevation as shown in Figure 1, which motivates us to use polynomial B-spline form as an inclusion function. In the B-spline approach for unconstrained global optimization [29] a B-spline is used to approximate the objective function with randomly scattered data using the least-square and pseudo-inverse methods. The use of B-spline approach for constrained global optimization is given in [11, 13] and references therein. Strength of the B-spline form, and thus solvers operating on the B-spline form, is the possibility for exact representation of multivariate (piecewise) polynomials and approximate representation of any function by sampling. Whereas we follow the procedure given in Section 2 by [21,22] to obtain the B-spline representation of a multivariate polynomial. This procedure do not require sample points and corresponding function evaluations. To the best of our knowledge, there are few papers in the literature on B-spline constrained global optimization [11, 13].

In this work, we propose B-spline based algorithms for solving non-convex nonlinear multivariate polynomial programming problems, where the objective function f and constraints (g_i & h_j) are limited to be *polynomial* functions. The proposed work extends the Bernstein method in [26, 27] and B-spline method in [11–13] for

constrained NLPs. The extensions are based on tools such as B-spline hull consistency (BsHC) and B-spline box consistency (BsBC) to contract the variable domains. The merits of the proposed approach are: (i) it avoids evaluation of the objective function and constraints; (ii) an initial guess to start optimization is not required, only an initial search box bounding the region of interest; (iii) it guarantees that the global minimum is found to a user-specified accuracy, and (iv) prior knowledge of stationary points is not required. Numerical performance of the proposed basic and improved algorithms are tested on 11 standard benchmark problems taken from [1, 4, 5, 10, 16] with dimensions varying from 2 to 7 and the number of constraints varying from 1 to 11. The optimal value of global minimum obtained with the proposed algorithms for 3 test problems are also compared with CENSO, GloptiPoly and some of the well-known NLP solvers. The rest of the paper is organized as follows. In Section 2, we give the notations and definitions of the B-spline form. In Section 3, we present the *basic* B-spline constrained global optimization and outline range enclosure property, subdivision procedure, the cut-off test, and the *basic* algorithms. In Section 4, we present the B-spline hull, B-spline box consistency techniques, and the *improved* B-spline constrained global optimization. In Section 5, we first compare the performances of the proposed basic and improved algorithms, and then we compare the optimal value of global minimum obtained using proposed algorithm with GloptiPoly [15] and several state-of-the-art NLP solvers. We give the conclusion of the work in Section 6.

2. B-spline form

For $I := [a, b]$ and given positive integers m and k, let $\mathbf{u} := \{x_i\}_{i=-m}^{k+m}$, with mesh length $h := (b-a)/k$, be a uniform grid partition defined by

$$\begin{aligned} x_{-m} &= x_{-m+1} = \cdots = x_0, \\ x_i &= a + ih, \qquad\qquad \text{for } i = 1, \ldots, k, \\ x_{k+1} &= x_{k+2} = \cdots = x_{k+m}. \end{aligned}$$

Then the associated polynomial spline space of degree m is defined by

$$S_m(I, \mathbf{u}) := \big\{ s \in C^{m-1}(I) : s_{|[x_i, x_{i+1}]} \in \mathbb{P}_m, i = 1, \ldots, k-1 \big\},$$

where $\mathbb{P}_m$ is the space of polynomials of degree at most m. It is well known that the set of the classical normalized B-splines $\{N_i^m, i = -m, \ldots, k-1\}$ is a basis for $S_m(I, \mathbf{u})$ that satisfies interesting properties. Among them, for example, each N_i^m is nonnegative on its support and $\{N_i^m\}_{i=-m}^{k-1}$ is a partition of unity. Let I_a^b represent the set of integers between a and b $(a < b)$. Also let $I_a^b = I_{-m}^{k-1} = \{-m, -m+1, \ldots, k-1\}$.

It is well-known (see *e.g.* [3]) that the monomials $x^r, r = 0, \ldots, m$ can be expressed in terms of B-splines through the relations

$$x^t = \sum_{j=-m}^{k-1} \pi_j^t N_j^m(x), \ t = 0, \ldots, m, \quad \text{and} \quad j \in I_a^b, \tag{2.1}$$

where π_j^t are the symmetric polynomials given by

$$\pi_j^t = \frac{\mathrm{Sym}_t(j+1, \ldots, j+m)}{\binom{m}{t}}, \ t = 0, \ldots, m, \tag{2.2}$$

with $\mathrm{Sym}_0(j+1, \ldots, j+m) = 1$, $\pi_j^{(0)} = 1$ and for $t \geq 1$, $\mathrm{Sym}_t(j+1, \ldots, j+m)$ represents the tth elementary symmetric function of $j+1, \ldots, j+m$, *i.e.*,

$$\mathrm{Sym}_t(j+1, \cdots, j+m) = \sum_{v_1, \ldots, v_r} v_1, v_2, \ldots, v_r, \tag{2.3}$$

where $v_1, \ldots, v_r$ are r distinct integers arbitrarily chosen from the array $\{j+1, \cdots, j+m\}$. The number of terms in (2.3) is $\binom{m}{r}$. The B-splines can be computed by the recurrence formula

$$N_i^m(x) = \gamma_{i,m}(x) N_i^{m-1}(x) + (1 - \gamma_{i+1,m}(x)) N_{i+1}^{m-1}(x), \ m \geq 1, \tag{2.4}$$

where

$$\gamma_{i,m}(x) := \begin{cases} \dfrac{x - x_i}{x_{i+m} - x_i}, & \text{if } x_i < x_{i+m}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.5}$$

and

$$N_i^0(x) := \begin{cases} 1, & \text{if } x \in [x_i, x_{i+1}), \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

In order to easily compute bounds for a range of a multivariate polynomial of degree N over an s-dimensional box, one can derive its B-spline representation [21, 22].

2.1. Univariate case

Firstly, we consider a univariate polynomial

$$p(x) := \sum_{t=0}^{n} a_t x^t, \ x \in [a, b], \tag{2.7}$$

to be expressed in terms of the B-spline basis of the space of polynomial splines of degree $m \geq n$ (*i.e.* order $m+1$). By substituting (2.1) into (2.7), we get

$$p(x) = \sum_{t=0}^{n} a_t \sum_{j=-m}^{k-1} \pi_j^{(t)} N_j^m(x) = \sum_{j=-m}^{k-1} \left(\sum_{t=0}^{n} a_t \pi_j^{(t)} \right) N_j^m(x) = \sum_{j=-m}^{k-1} d_j N_j^m(x), \tag{2.8}$$

where

$$d_j := \sum_{t=0}^{n} a_t \pi_j^{(t)}. \tag{2.9}$$

2.2. Multi segment B-splines

Let us consider a polynomial $p(x) = 3.371x^3 - 10.10x^2 + 8.233x + 2$ and $x \in [0, 2]$. Its polynomial B-spline plot with number of segments equal to 1, 2 and 3 are shown in Figure 1. The B-spline with a single segment has four control points, while the one with two segments has five control points, and the one with three segments has six control points. The advantage of B-spline with more number of segments is that we have more control points. This gives a tight range enclosure without having to increase the degree of the B-spline. The drawback of having more segments is the increase in computation time with the number of B-spline coefficients. In our application to global minimization, we find that the B-spline having a number of segment equal to the order of B-spline plus one is a good option.

(1) k equal to order of B-spline ($k = m + 1$):
Let us continue considering same polynomial $p(x)$ as above. Its polynomial B-spline form of degree $m = 3$ and order 4 with the number of segments taken equal to order of B-spline, will consist of seven B-spline coefficients, *i.e.* seven B-spline control points and seven B-spline basis functions. The plot of these seven basis functions is shown in Figure 2. As seen from this figure, one of the B-spline basis function that is N_3^3 lies on the entire domain of x.

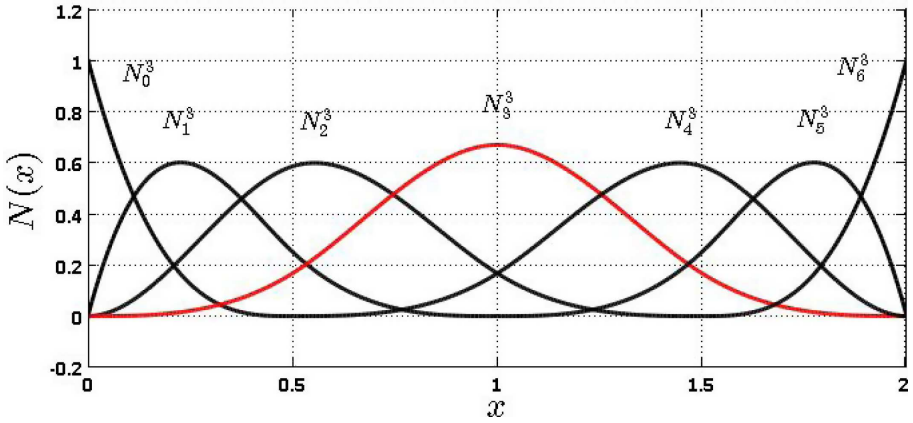

FIGURE 2. Plot of B-spline basis functions for $p(x) = 3.371x^3 - 10.10x^2 + 8.233x + 2;\ x \in [0, 2]$ with number of segments taken equal to order of the B-spline.

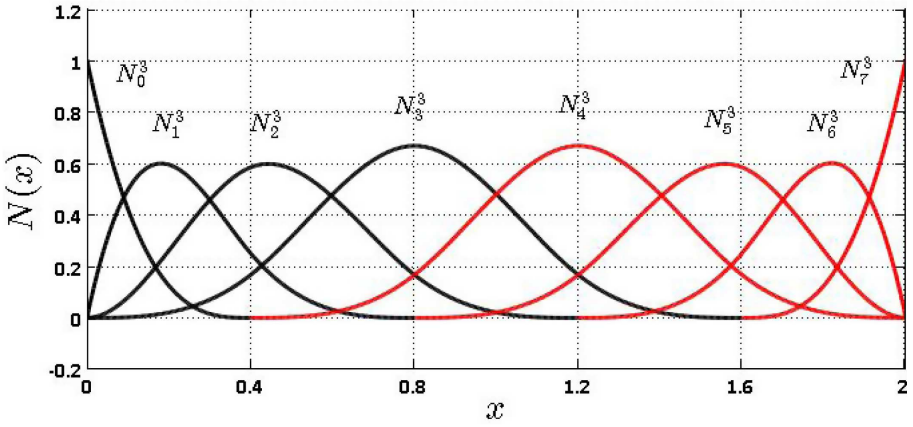

FIGURE 3. Plot of B-spline basis functions for $p(x) = 3.371x^3 - 10.10x^2 + 8.233x + 2;\ x \in [0, 2]$ with number of segments taken equal to order of the B-spline plus one.

(2) k taken equal to order of B-spline plus one ($k = m + 2$):
We continue with the same polynomial $p(x)$ as above and obtain the polynomial B-spline form with the number of segments equal to order plus one. Now, the B-spline has eight B-spline coefficients and eight B-spline basis functions. As shown in Figure 3, half the B-spline basis functions are having the support of lower domain value of x, whereas the other half has the support of upper domain value of x. Because of this symmetry, a B-spline with the number of segments equal to order plus one is a good option for our application of global minimization.

2.3. Multivariate case

Now, we derive the B-spline representation of a given multivariate polynomial

$$p(x_1, x_2, \ldots, x_s) = \sum_{i_1=0}^{n_1} \cdots \sum_{i_s=0}^{n_s} a_{i_1 \ldots i_s} x_1^{i_1} \ldots x_s^{i_s} = \sum_{I \le N} a_I x^I, \tag{2.10}$$

where $I := (i_1, i_2, \ldots, i_s)$, and $N := (n_1, n_2, \ldots, n_s)$. By substituting (2.1) for each x^i, equation (2.10) can be written as

$$\begin{aligned} p(x_1, x_2, \ldots, x_s) &= \sum_{i_1=0}^{n_1} \cdots \sum_{i_s=0}^{n_s} a_{i_1 \ldots i_s} \sum_{j_1=-m_1}^{k_1-1} \pi_{j_1}^{(i_1)} N_{j_1}^{m_1}(x_1) \cdots \sum_{j_s=-m_s}^{k_s-1} \pi_{j_s}^{(i_s)} N_{j_s}^{m_s}(x_s) \\ &= \sum_{j_1=-m_1}^{k_1-1} \cdots \sum_{j_s=-m_s}^{k_s-1} \left(\sum_{i_1=0}^{n_1} \cdots \sum_{i_s=0}^{n_s} a_{i_1 \ldots i_s} \pi_{j_1}^{(i_1)} \ldots \pi_{j_s}^{(i_s)} \right) N_{j_1}^{m_1}(x_1) \ldots N_{j_s}^{m_s}(x_s) \\ &= \sum_{j_1=-m_1}^{k_1-1} \cdots \sum_{j_s=-m_s}^{k_s-1} d_{j_1 \ldots j_s} N_{j_1}^{m_1}(x_1) \ldots N_{j_s}^{m_s}(x_s), \end{aligned} \tag{2.11}$$

so that we have expressed p as

$$p(x) = \sum_{J \le -M}^{K-1} d_J N_J^M(x), \tag{2.12}$$

where $M := \{m_1, \ldots, m_s\}$; $K := \{k_1, \ldots, k_s\}$ and $J := \{j_1, \ldots, j_s\}$ and the B-spline coefficients d_J are given by

$$d_{j_1, \ldots, j_s} := \sum_{i_1=0}^{n_1} \cdots \sum_{i_s=0}^{n_s} a_{i_1 \ldots i_s} \pi_{j_1}^{(i_1)} \ldots . \pi_{j_s}^{(i_s)}. \tag{2.13}$$

The B-spline form of a multivariate polynomial p is defined by (2.11). The partial derivative of a polynomial in a particular direction can be found from the B-spline coefficients of the original polynomial on a box $\mathbf{b} \subseteq \mathbf{x}$, the first partial derivative with respect to x_r of a polynomial $p(x)$ in B-spline form in [34]

$$p_r'(\mathbf{b}) = \frac{n_r}{u_{I+n_r+1} - u_{I+1}} \sum_{I \le N_{r,-1}} \left[d_{I_{r,1}}(\mathbf{b}) - d_I(\mathbf{b}) \right] N_{N_{r,-1},I}(x), 1 \le r \le s, x \in \mathbf{b} \tag{2.14}$$

where u represents knot vector of p. Now, $p_r'(\mathbf{b})$ contains an enclosure of the range of the partial derivative of p on $\mathbf{b}$.

2.4. Example

We consider following example to explain the above ideas.

Example 2.1. Let $p(x, y) = x^2 + y^2 - x - y + 0.34$ and $x, y \in [0.5, 1.5]$. We want to obtain the polynomial B-spline form having two B-spline segments for given power form polynomial.

The matrix form representation of $p(x, y)$ is

$$p(x, y) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} 0.34 & -1 & 1 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \end{bmatrix}.$$

The degree of variable x in the given polynomial is $n_1 = 2$ and that of y is $n_2 = 2$. The degree of B-spline in x direction m_1, can be greater or equal to the degree of x. Similarly, the B-spline in y direction will have a degree m_2 equal to or greater than degree of variable y. In practice, we can therefore take $m_1 = n_1 = 2$ and $m_2 = n_2 = 2$. Therefore, the order, O of B-spline will be $O = m + 1 = 3$ in both the directions. As $k = 2$, the B-spline will have two segments in each direction. As $x, y \in [0.5, 1.5]$, the knot vector for both the variables will be the same.

We have

$$\#\text{ control points} = k + m = 2 + 2 = 4,$$

$$\#\text{ knot elements} = 2 \times n + k + 1 = 2 \times 2 + 2 + 1 = 7.$$

Therefore, the knot vector $\mathrm{u} := \{u_l, l = -m, \ldots m + k\}$ is

$$\mathrm{u} := \{0.5, 0.5, 0.5, 1, 1.5, 1.5, 1.5\}.$$

From (2.11), B-spline representation of $p(x, y)$ can be expressed in matrix form as

$$p(x,y) = \left[N^2_{-2}(x)N^2_{-1}(x)N^2_0(x)N^2_1(x)\right] \begin{bmatrix} d_{-2,-2} & d_{-2,-1} & d_{-2,0} & d_{-2,1} \\ d_{-1,-2} & d_{-1,-1} & d_{-1,0} & d_{-1,1} \\ d_{0,-2} & d_{0,-1} & d_{0,0} & d_{0,1} \\ d_{1,-2} & d_{1,-1} & d_{1,0} & d_{1,1} \end{bmatrix} \begin{bmatrix} N^2_{-2}(y) \\ N^2_{-1}(y) \\ N^2_0(y) \\ N^2_1(y) \end{bmatrix}.$$

From (2.13), we calculate the values of B-spline coefficients as

$$\begin{bmatrix} d_{-2,-2} & d_{-2,-1} & d_{-2,0} & d_{-2,1} \\ d_{-1,-2} & d_{-1,-1} & d_{-1,0} & d_{-1,1} \\ d_{0,-2} & d_{0,-1} & d_{0,0} & d_{0,1} \\ d_{1,-2} & d_{1,-1} & d_{1,0} & d_{1,1} \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} \pi^0_{-2}(y) & \pi^1_{-2}(y) & \pi^2_{-2}(y) \\ \pi^0_{-1}(y) & \pi^1_{-1}(y) & \pi^2_{-1}(y) \\ \pi^0_0(y) & \pi^1_0(y) & \pi^2_0(y) \\ \pi^0_1(y) & \pi^1_1(y) & \pi^2_1(y) \end{bmatrix}}_{Pi \text{ Matrix of } y} * \left\{ \underbrace{\begin{bmatrix} \pi^0_{-2}(x) & \pi^1_{-2}(x) & \pi^2_{-2}(x) \\ \pi^0_{-1}(x) & \pi^1_{-1}(x) & \pi^2_{-1}(x) \\ \pi^0_0(x) & \pi^1_0(x) & \pi^2_0(x) \\ \pi^0_1(x) & \pi^1_1(x) & \pi^2_1(x) \end{bmatrix}}_{Pi \text{ Matrix of } x} * \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \right\}^T.$$

A simple computation leads to the B-spline coefficient matrix

$$\begin{bmatrix} d_{-2,-2} & d_{-2,-1} & d_{-2,0} & d_{-2,1} \\ d_{-1,-2} & d_{-1,-1} & d_{-1,0} & d_{-1,1} \\ d_{0,-2} & d_{0,-1} & d_{0,0} & d_{0,1} \\ d_{1,-2} & d_{1,-1} & d_{1,0} & d_{1,1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.5 & 0.25 \\ 1 & 0.75 & 0.5 \\ 1 & 1.25 & 1.5 \\ 1 & 1.5 & 2.25 \end{bmatrix}}_{Pi \text{ Matrix of } y} * \left\{ \underbrace{\begin{bmatrix} 1 & 0.5 & 0.25 \\ 1 & 0.75 & 0.5 \\ 1 & 1.25 & 1.5 \\ 1 & 1.5 & 2.25 \end{bmatrix}}_{Pi \text{ Matrix of } x} * \begin{bmatrix} 0.34 & -1 & 1 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \right\}^T$$

$$= \begin{bmatrix} -0.16 & -0.16 & 0.34 & 0.84 \\ -0.16 & -0.16 & 0.34 & 0.84 \\ 0.34 & 0.34 & 0.84 & 1.34 \\ 0.84 & 0.84 & 1.34 & 1.84 \end{bmatrix}.$$

3. B-spline constrained global optimization

Let $s \in \mathbb{N}$ be the number of variables and $x = (x_1, x_2, \ldots, x_s) \in \mathbb{R}^s$. A multi-index I is defined as $I = (i_1, i_2, \ldots, i_s) \in (\mathbb{N} \cup \{0\})^s$ and the multi-power x^I is defined as $x^I = (x_1^{i_1}, x_2^{i_2}, \ldots, x_s^{i_s})$. Given a multi-index $N = (n_1, n_2, \ldots, n_s)$ and an index r, we define $N_{r,-l} = (n_1, \ldots.., n_{r-1}, n_r - l, n_{r+1}, \ldots., n_s)$, where $0 \le n_r - l \le n_r$. Inequalities $I \le N$ for multi-indices are meant componentwise, *i.e.* $i_l \le n_l$, $l = 1, 2, \ldots, s$. With

$I = (i_1, \ldots, i_{r-1}, i_r, i_{r+1}, \ldots, i_s)$ we associate the index $I_{r,l}$ given by $I_{r,l} = (i_1, \ldots, i_{r-1},\ i_r + l, i_{r+1}, \ldots, i_s)$, where $0 \leq i_r + l \leq n_r$. A real bounded and closed interval $\mathbf{x}_r$ is defined as

$$\mathbf{x}_r \equiv [\underline{\mathbf{x}_r}, \overline{\mathbf{x}_r}] := [\inf \mathbf{x}_r = \min \mathbf{x}_r, \sup \mathbf{x}_r = \max \mathbf{x}_r] \in \mathbb{IR},$$

where $\mathbb{IR}$ denotes the set of *compact intervals*. Let $\operatorname{wid} \mathbf{x}_r$ denotes the *width* of $\mathbf{x}_r$, that is $\operatorname{wid} \mathbf{x}_r := \overline{\mathbf{x}_r} - \underline{\mathbf{x}_r}$.

Global optimization of polynomials using the polynomial B-spline approach needs transformation of the given multivariate polynomial from its power form into its polynomial B-spline form. Then B-spline coefficients are collected in an array $D(\mathbf{x}) = (d_I(\mathbf{x}))_{I \in S}$, where $S = \{I : I \leq N\}$. This array is called a *patch*. We denote S_0 as a special subset of the index set S comprising indices of the vertices of this array, that is

$$S_0 := \{0, n_1 + k_1 - 1\} \times \{0, n_2 + k_2 - 1\} \times \ldots \times \{0, n_s + k_s - 1\}. \tag{3.1}$$

3.1. Range enclosure property

The following lemma describes the range enclosure property of the B-spline coefficients.

Lemma 3.1 ([21–23, 29]). *Let p be a polynomial of degree N and let $\bar{p}(\mathbf{x})$ denote the range of p on the given domain $\mathbf{x}$. Then, for a patch $D(\mathbf{x})$ of B-spline coefficients it holds*

$$\bar{p}(\mathbf{x}) \subseteq [\min D(\mathbf{x}), \max D(\mathbf{x})].$$

Obtaining the B-spline coefficients of multivariate polynomials by transforming the polynomial from power form to B-spline form provides an enclosure of the range of the multivariate polynomial p on $\mathbf{x}$. Then by Lemma 3.1, the minimum and the maximum values of B-spline coefficient provide lower and upper bounds for the range of polynomial p. This range enclosure will be sharp if and only if $\min(d_I(\mathbf{x}))_{I \in S}$ (respectively $\max(d_I(\mathbf{x}))_{I \in S}$) is attained at the indices of the vertices of the array $D(\mathbf{x})$, as described in Lemma 3.2. This condition is known as the vertex condition.

Based on Bernstein coefficients range enclosure proofs [32], the proof of Lemma 3.1 is given next.

Proof. Let $I_a^b = I_1^{k+m}$ represent the set of integers between a and b $(a < b)$. Also let

$$p(x) = \sum_{j=1}^{m+k} d_j N_j(x), \quad j \in I_a^b,$$

be the B-spline representation for polynomial p and let $\bar{p}$ be its range,

$$\bar{p}([a, b]) \subseteq \left[\min_{1 \leq j \leq m+k} d_j, \max_{1 \leq j \leq m+k} d_j \right],$$

that is

$$\min_{1 \leq j \leq m+k} d_j \leq \sum_j d_j N_j(x) \leq \max_{1 \leq j \leq m+k} d_j,$$

which means

$$\min_{1 \leq j \leq m+k} d_j \leq \min \bar{p} \leq \max \bar{p} \leq \max_{1 \leq j \leq m+k} d_j.$$

□

Lemma 3.2 ([32] Vertex condition). *Let p be a polynomial of degree N and let $\bar{p}(\mathbf{x}) = [a, b]$. Then*

$$a = \min_{0 \leq I \leq N} d_I(\mathbf{x}) \qquad \text{if and only if} \qquad \min_{0 \leq I \leq N} d_I(\mathbf{x}) = \min_{I \in S_0} d_I(\mathbf{x}),$$

and

$$b = \max_{0 \leq I \leq N} d_I(\mathbf{x}) \qquad \text{if and only if} \qquad \max_{0 \leq I \leq N} d_I(\mathbf{x}) = \max_{I \in S_0} d_I(\mathbf{x}).$$

Proof. Let

$$p(x) = \sum_{j=1}^{m+k} d_j N_j(x),$$

be the B-spline representation for polynomial p and let $\bar{p}$ be its range,

$$\bar{p}([0,1]) = [a,b].$$

We first note that

$$p(0) = d_1,$$

and

$$p(1) = d_{m+k}.$$

Suppose now that

$$a = \min_{1\le j\le m+k} d_j,$$

and that $x \in [0,1]$, then if $d_1 = d_2 = \ldots = d_{m+k}$. We have that $p(0) = d_1 = a = d_{m+k} = p(1)$ and the property is valid. If $d_1 = d_2 = \ldots = d_{m+k}$ is not satisfied, then

$$\sum_{j=1}^{m+k} d_j N_j(x) > \min_{1\le j\le m+k} d_j,$$

which means that the minimum cannot occur at an interior point of $[0,1]$ because

$$a = \min_{1\le j\le m+k} d_j,$$

and

$$p(0) = a,$$

hence it must occur at an end point of the interval. Conversely, suppose that

$$\min_{1\le j\le m+k} d_j = \min(d_1, d_{m+k}),$$

and assume that

$$\min_{1\le j\le m+k} d_j = d_1,$$

for simplicity. Then the bound is sharp, *i.e.*

$$a = p(0) = d_1 = \min_{1\le j\le m+k} d_j,$$

the same argument is valid if

$$\min_{1\le j\le m+k} d_j = d_{m+k}.$$

□

Definition 3.3. The vertex condition is said to be met within a given tolerance ϵ, if

$$\min_{S_0} D(\mathbf{x}) - \min D(\mathbf{x}) \le \epsilon \text{ and } \max D(\mathbf{x}) - \max_{S_0} D(\mathbf{x}) \le \epsilon.$$

As said earlier set S_0 comprises of indices of the vertices of the B-spline coefficient array $D(\mathbf{x})$, where $\min_{S_0} D(\mathbf{x})$ gives the minimum value of B-spline coefficient at any vertex point. If the difference between the minimum value of B-spline coefficient at any vertex and the minimum value in B-spline coefficient array is less than ϵ, then vertex condition is said to be met within a given tolerance ϵ.

3.2. B-spline subdivision procedure

The proposed algorithm is based on branch and bound framework of global optimization. Therefore we need to use domain subdivision. Generally, the range enclosure obtained from Lemma 3.1 is over-estimated and can be improved either by subdividing the domain, degree elevation of the B-spline or by increasing the number of B-spline segments. Subdivision is generally more efficient than degree elevation strategy [6] or increasing the number of B-spline segments. Therefore subdivision strategy is preferred over the latter two. A subdivision in the rth direction ($1 \leq r \leq s$) is a bisection perpendicular to this direction. Let

$$\mathbf{x} = \left[\underline{\mathbf{x}_1}, \overline{\mathbf{x}_1}\right] \times \ldots \times \left[\underline{\mathbf{x}_r}, \overline{\mathbf{x}_r}\right] \times \ldots \times \left[\underline{\mathbf{x}_s}, \overline{\mathbf{x}_s}\right], \tag{3.2}$$

be any subbox. Further suppose that $\mathbf{x}$ is bisected along the rth component direction then, two subboxes $\mathbf{x}_A$ and $\mathbf{x}_B$ are generated as

$$\mathbf{x}_A = \left[\underline{\mathbf{x}_1}, \overline{\mathbf{x}_1}\right] \times \ldots \times \left[\underline{\mathbf{x}_r}, m(\mathbf{x}_r)\right] \times \ldots \times \left[\underline{\mathbf{x}_s}, \overline{\mathbf{x}_s}\right], \tag{3.3}$$

$$\mathbf{x}_B = \left[\underline{\mathbf{x}_1}, \overline{\mathbf{x}_1}\right] \times \ldots \times \left[m(\mathbf{x}_r), \overline{\mathbf{x}_r}\right] \times \ldots \times \left[\underline{\mathbf{x}_s}, \overline{\mathbf{x}_s}\right], \tag{3.4}$$

where, $m(\mathbf{x}_r)$ denotes the midpoint of $[\underline{\mathbf{x}_r}, \overline{\mathbf{x}_r}]$.

3.3. The cut-off test

As mentioned earlier, the minimum and maximum B-spline coefficients provide the range enclosure of the function. Let $\tilde{p}$ be the current minimum estimate, and $\{\mathbf{b}, D(\mathbf{b})\}$ be the current item for processing. We denote the minimum over the second entry of item $\{\mathbf{b}, D(\mathbf{b})\}$ as p. If p is greater than $\tilde{p}$, then this item cannot contain global minimum and it can be discarded. If the maximum over the second entry of item $\{\mathbf{b}, D(\mathbf{b})\}$ is lesser than $\tilde{p}$, then the current minimum estimate can be updated, and $\tilde{p}$ takes this maximum value as the new value. Next we present the cut-off test algorithm.

Algorithm 1: Cut-off test.

if $\min D(\mathbf{b}) > \tilde{p}$ **then**
  $\mathcal{L} = \mathcal{L} - \{\mathbf{b}, \mathcal{D}(\mathbf{b})\}$;
else if $\max D(\mathbf{b}) < \tilde{p}$ **then**
  $\tilde{p} = \max D(\mathbf{b})$;
end

3.4. A basic B-spline constrained global optimization algorithm

In this subsection, we present the *basic* B-spline algorithm for constrained global optimization of multivariate nonlinear polynomials. The algorithm is inspired by the one described in [31,33].

This basic algorithm uses the polynomial coefficients of the objective function, the inequality constraints, and the equality constraints. The inputs to the algorithm are the polynomial degrees and the initial search box, while the outputs are the global minimum and global minimizers. The polynomial degree is used to compute the B-spline segment number, as the B-spline is constructed with number of segments equal to order of the B-spline plus one. As equality constraints $h_j(x) = 0$ are difficult to verify on computers with finite precision, the equality constraints $h_j(x) = 0$ in (1.3) are replaced by relaxed constraints $h_j(x) \in [-\epsilon_{\text{zero}}, \epsilon_{\text{zero}}], j = 1, 2, \ldots, q$, where $\epsilon_{\text{zero}} > 0$ is a very small number.

The basic algorithm works as follows. We start the algorithm by computing the B-spline segment vectors K_o, K_{g_i} and K_{h_j}, where $K = [k_1, \ldots, k_s]$, for each variable occurring in the objective, inequality and equality polynomials. We keep it as $order + 1$ for each variable, giving $K = [n_1 + 2, \ldots, n_s + 2]$. Then, we compute the

B-spline coefficients of objective, inequality and equality constraint on the initial search box. We store them in arrays $D_o(\mathbf{x}), D_{g_i}(\mathbf{x})$ and $D_{h_j}(\mathbf{x})$ respectively. We initialize the current minimum estimate $\tilde{p}$ to the maximum B-spline coefficient of the objective function on $\mathbf{x}$. Next, we initialize a flag vector F with each component to zero, a working list $\mathcal{L}$ with the item $\{\mathbf{x}, D_o(\mathbf{x}), D_{g_i}(\mathbf{x}), D_{h_j}(\mathbf{x}), F\}$, and a solution list $\mathcal{L}^{\text{sol}}$ to the empty list.

We then pick the last item from the list $\mathcal{L}$ and delete its entry from $\mathcal{L}$. For this item, we subdivide the box $\mathbf{x}$ along the longest width direction creating two subboxes $\mathbf{b}_1$ and $\mathbf{b}_2$. We compute the B-spline coefficients arrays $\{\mathbf{b}_r, D_o(\mathbf{b}_r), D_{g_i}(\mathbf{b}_r),\ D_{h_j}(\mathbf{b}_r)\}$, $r = 1, 2$ for $\mathbf{b}_1$,$\mathbf{b}_2$ and the B-spline range enclosures $\mathbb{D}_o(\mathbf{b}_r), \mathbb{D}_{g_i}(\mathbf{b}_r)$ and $\mathbb{D}_{h_j}(\mathbf{b}_r)$ of objective, inequality and equality constraint polynomials respectively. We check the feasibility of the inequality and equality constraints for $\mathbf{b}_1$, $\mathbf{b}_2$ using the B-spline coefficients of the constraint polynomials functions by doing the following tests:

- If $\mathbb{D}_{g_i}(\mathbf{b}_r) \leq 0$, $0 \in \mathbb{D}_{h_j}(\mathbf{b}_r)$, $\mathbb{D}_{h_j}(\mathbf{b}_r) \subseteq [-\epsilon_{\text{zero}}, \epsilon_{\text{zero}}]$ for all $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q$, then $\mathbf{b}_r$ is a feasible box.
- If $\mathbb{D}_{g_i}(\mathbf{b}_r) > 0$ for some i, then $\mathbf{b}_r$ is a infeasible box and can be deleted.
- If $0 \notin \mathbb{D}_{h_j}(\mathbf{b}_r)$, $\mathbb{D}_{h_j}(\mathbf{b}_r) \nsubseteq [-\epsilon_{\text{zero}}, \epsilon_{\text{zero}}]$ for some j, then $\mathbf{b}_r$ is a infeasible box and can be deleted.

If $\mathbf{b}_r$ survives these tests and if each component of the flag vector F^r (see Rem. 3.4 below) is equal to unity, we update the current minimum estimate $\tilde{p}$, and add the item $\{\mathbf{b}_r, D_o(\mathbf{b}_r), D_{g_i}(\mathbf{b}_r), D_{h_j}(\mathbf{b}_r), F^r\}$ to the list $\mathcal{L}$, and sort the list in descending order of the minimum of the B-spline coefficients of the objective function. Next, we discard item(s) $\{\mathbf{y}, D_o(\mathbf{y}), D_{g_i}(\mathbf{y}), D_{h_j}(\mathbf{y}), F\}$ from the list $\mathcal{L}$, if $\min D_o(\mathbf{y}) > \tilde{p}$. The last item in the list $\mathcal{L}$ is picked for further processing. If the width of the box and the width of the B-spline range enclosure of the objective polynomial are within the desired accuracy, then we put this item in the solution list $\mathcal{L}^{\text{sol}}$, else we continue the algorithm until the list $\mathcal{L}$ becomes empty.

Remark 3.4. The flag vector F is used to make the algorithm more efficient. Consider, ith inequality constraint is satisfied for $x \in \mathbf{b}$ *i.e.* $g_i(x) \leq 0$ for $x \in \mathbf{b}$. Then there is no need to check again $g_i(x) \leq 0$ for any subbox $\mathbf{b}_0 \subseteq \mathbf{b}$. The same holds true for $h_j(x)$. To handle this information we use flag vector $F = (F_1, \ldots, F_p, F_{p+1}, \ldots, F_{p+q})$, where the components F_f, takes the value 0 or 1, as follows

- $F_f = 1$ if the fth inequality or equality constraint is satisfied for the box.
- $F_f = 0$ if the fth inequality or equality constraint is not yet been verified for the box.

Algorithm 2: Basic $(A_c, \mathrm{N_c}, K_c, \mathbf{x}, \epsilon, \epsilon_{\text{zero}})$.

Input : *Here A_c is a cell structure containing the coefficients array of objective and all the constraints polynomial, N_c is a cell structure, containing degree vector N for objective and all constraints. The elements of degree vector N define the degree of each variable occurring in the objective function and all constraints polynomial, K_c is a cell structure containing vectors corresponding to objective polynomial, K_o and all constraints, i.e. K_{g_i}, K_{h_j}. The elements of this vector define the number of B-spline segments in each variable direction, the initial box $\mathbf{x}$, the tolerance limit ϵ and tolerance parameter ϵ_{zero} to which the equality constraints are to be satisfied.*

Output: *Global minimum $\hat{p}$ and all the global minimizers $z^{(i)}$ in the initial search box $\mathbf{x}$ to the specified tolerance ϵ.*

Begin Algorithm

1 {Compute the B-spline segment numbers};
For each entry of K in K_c, compute $K = N + 2$.;

2 {Compute the B-spline coefficients};
Compute the B-spline coefficients array for objective and constraints polynomial on initial search domain $\mathbf{x}$ *i.e.* $D_o(\mathbf{x})$, $D_{g_i}(\mathbf{x})$ *and* $D_{h_j}(\mathbf{x})$ *respectively. The algorithm in [8] is suggested for the computation.*;

3 {Initialize current minimum estimate}
Initialize the current minimum estimate $\tilde{p} = \max D_o(\mathbf{x})$.

4 {Initialize current minimum estimate}
Initialize the current minimum estimate $\tilde{p} = \max D_o(\mathbf{x})$.

5 {Set flag vector}
Set $F = (F_1, \ldots, F_p, F_{p+1}, \ldots, F_{p+q}) := (0, \ldots, 0)$.

6 {Initialize lists}
$\mathcal{L} \leftarrow \{\mathbf{x}, D_o(\mathbf{x}), D_{g_i}(\mathbf{x}), D_{h_j}(\mathbf{x}), F\}$, $\mathcal{L}^{\text{sol}} \leftarrow \{\}$

7 {Sort the list $\mathcal{L}$}
Sort the list $\mathcal{L}$ *in descending order of* $(\min D_o(\mathbf{x}))$.

8 {Start iteration}
if $\mathcal{L} = \emptyset$ **then**
  go to 13
  else
  pick the last item from $\mathcal{L}$*, denote it as* $\{\mathbf{b}, D_o(\mathbf{b}), D_{g_i}(\mathbf{b}), D_{h_j}(\mathbf{b}), F\}$*, and delete this item entry from* $\mathcal{L}$.
end

9 {Perform cut-off test} {See Section 3.3}
if $\min D_o(\mathbf{b}) > \tilde{p}$ **then**
  Discard the item $\{\mathbf{b}, D_o(\mathbf{b}), D_{g_i}(\mathbf{b}), D_{h_j}(\mathbf{b}), F\}$
  &
  go to 8
else if $\max D_o(\mathbf{b}) < \tilde{p}$ **then**
  $\tilde{p} = \max D_o(\mathbf{b})$
end

10 {Subdivision decision}
if (wid $\mathbf{b}$)**&** $(\max D_o(\mathbf{b}) - \min D_o(\mathbf{b})) < \epsilon$ **then**
  enter the item $\{\mathbf{b}, \min D_0(\mathbf{b})\}$ *to* $\mathcal{L}^{\text{sol}}$ **& go to** 8
  else
  go to 11
end

11 {Generate two sub boxes}

Choose the subdivision direction along the longest direction of $\mathbf{b}$ *and the subdivision point as the midpoint. Subdivide* $\mathbf{b}$ *into two subboxes* $\mathbf{b}_1$ *and* $\mathbf{b}_2$ *such that* $\mathbf{b} = \mathbf{b}_1 \cup \mathbf{b}_2$.

12 **for** $r \leftarrow 1$ **to** 2

(a) {Set flag vector}

Set $F^r = (F_1^r, \ldots, F_p^r, F_{p+1}^r, \ldots, F_{p+q}^r) := F$

for $r \leftarrow 1$ **to** 2

(b) {Compute B-spline coefficients and corresponding B-spline range enclosure for $\mathbf{b}_r$}

Compute the B-spline coefficient arrays of objective and constraints polynomial on box $\mathbf{b}_r$ *and compute corresponding B-spline range enclosure* $\mathbb{D}_o(\mathbf{b}_r), \mathbb{D}_{g_i}(\mathbf{b}_r)$ *and* $\mathbb{D}_{h_j}(\mathbf{b}_r)$ *for objective and constraints polynomial.*

(c) {Set local current minimum estimate}

Set $\tilde{p}_{\text{local}} = \min(\mathbb{D}_o(\mathbf{b}_r))$

(d) **if** $(\tilde{p}_{\text{local}} < \tilde{p})$ **then**

1 **for** $i \leftarrow 1$ **to** p **do**

if $(F_i = 0)$ **&** $(\mathbb{D}_{g_i}(\mathbf{b}_r) \leq 0)$ **then**

$F_i^r = 1$

end

end

2 **for** $j \leftarrow 1$ **to** q **do**

if $(F_{p+j} = 0)$ **&** $\left(\mathbb{D}_{h_j}(\mathbf{b}_r) \subseteq [-\epsilon_{\text{zero}}, \epsilon_{\text{zero}}]\right)$ **then**

$F_{p+j}^r = 1$

end

end

end

(e) **if** $F^r = (1, \ldots, 1)$ **then**

set $\tilde{p} := \min(\tilde{p}, \max(\mathbb{D}_o(\mathbf{b}_r)))$

end

(f) *Enter* $\{\mathbf{b}_r, D_o(\mathbf{b}_r), D_{g_i}(\mathbf{b}_r), D_{h_j}(\mathbf{b}_r), F^r\}$ *into the list* $\mathcal{L}$.

end

13 {Compute the global minimum}

Set the global minimum to the current minimum estimate $\hat{p} = \tilde{p}$.

14 {Compute the global solution}

Find all those items in $\mathcal{L}^{\text{sol}}$ *for which* $\min D_o(\mathbf{b}) = \widehat{p}$. *The first entries of these items are the global minimizer(s)* $\mathbf{z}^{(i)}$.

15 **return** *the global minimum* $\hat{p}$ *and all the global minimizers* $\mathbf{z}^{(i)}$ *found above.*

End Algorithm

3.5. Convergence property of the basic algorithm

The problem of polynomial optimization can be reduced to evaluate the range of the polynomial. On the expansion of the polynomial $p(x)$ into polynomial B-spline form, the range enclosure property of the B-spline form (see Lem. 3.1), *i.e.* the minimum B-spline coefficient, $\min D(\mathbf{x})$ and the maximum B-spline coefficient, $\max D(\mathbf{x})$ provide lower and upper bounds for the range of the polynomial $\overline{p}(x)$, *i.e.* $\overline{p}(\mathbf{x}) \subseteq D(\mathbf{x}) = [\min D(\mathbf{x}), \max D(\mathbf{x})]$. The B-spline range enclosure $D(\mathbf{x})$ is an interval extension of $p(x)$, where an interval extension is defined as

Definition 3.5 (Interval extension and Inclusion monotonicity [2])**.** Let f be a real-valued function of real variables and F be an interval function [2]. Then, F is an interval extension of f if

$$f(x) = F(x) \text{ for all } x \in \mathbb{R}^s \text{ and } f(\mathbf{x}) \subseteq F(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathbb{IR}^s.$$

If F is an interval extension of f, then $F(\mathbf{x})$ bounds the range of f on $\mathbf{x}$. An interval extension F is said to be inclusion monotonic if

$$\mathbf{x}_i \subseteq \mathbf{y}_i,\ i = 1, 2, \ldots, s \Rightarrow F(\mathbf{x}_1, \ldots, \mathbf{x}_s) \subseteq F(\mathbf{y}_1, \ldots, \mathbf{y}_s).$$

Definition 3.6 (Convergence)**.** We say that a sequence of intervals $D\left(\mathbf{b}_1^k\right)$ *converges* to p^*, if both sequences $\left\{D\left(\mathbf{b}_1^k\right)\right\}$ of left endpoints and $\left\{D\left(\mathbf{b}_1^k\right)\right\}$ of right endpoints, converge to p^*.

The main assumptions we need are the contraction properties of the inclusion functions: F, G, and H are the inclusion functions of f,g, and h, we consider the following assumptions:

$$w(F(\mathbf{y})) \to 0 \quad \text{as} \quad w(\mathbf{y}) \to 0 \quad \text{for} \quad \mathbf{y} \in \mathbf{x} \tag{3.5}$$

$$w(G(\mathbf{y})) \to 0, w(H(\mathbf{y})) \to 0 \quad \text{as} \quad w(\mathbf{y}) \to 0 \quad \text{for} \quad \mathbf{y} \in \mathbf{x} \tag{3.6}$$

equation (3.5) implies the continuity of f, and (3.6) implies the continuity of g and h.

Lemma 3.7. *Let $Z_{n1}, \ldots, Z_{nl_n}$ be the boxes which are in L_n that is, the list at the n-th iteration. Let*

$$U_n = \bigcup_{i=1}^{l_n} Z_{ni}.$$

We then have the following theorem,

Theorem 3.8. *If the contraction assumption (3.5) and (3.6) hold, then the sequence (U_n) forms a nested sequence and $\cap_{n=1}^{\infty} U_n = \mathbb{D}$ which means that $U_n \to \mathbb{D}$ if $\mathbb{D} \neq 0$. Where $\mathbb{D}$ is a set of all $x \in \mathbf{x}$ satisfying (1.2) and (1.3) i.e. feasible set of the problem.*

Proof. See [33]. □

Let $Z_{n1}, \ldots, Z_{nl_n}$ be the boxes of list L_n and $U_n = \cup_{i=1}^{l_n} Z_{ni}$ where l_n is the length of L_n.

Let $\tilde{y}_n$ be the current value of $\tilde{y}$ in the list L_n where $\tilde{y} := \min F(\mathbf{x})$ and let f_n be the current value of $\tilde{f}$ in the list L_n where $\tilde{f} := \max F(\tilde{x})$ if a feasible point $\tilde{x}$ is given else $\tilde{f} := \infty$.

The convergence properties of basic constrained algorithm depend on the possibility of applying the midpoint test as often as in order to exhaust all points which are not global minimizers this leads to the following assumption which is mainly of a topological character:

$$(T)\begin{cases} \text{There exists a sequence of points} \\ x_n \text{ lying in the interior of the feasible domain D} \\ \text{and converging to some global minimizers } x^* \in \mathbf{x}^*. \end{cases}$$

The following theorem discusses the convergence properties of basic constrained global optimization algorithm.

Theorem 3.9. *Let basic constrained global optimization algorithm applied to the box $\boldsymbol{x}$, the inclusion functions F, G, and H of f, g, and h, respectively. Let the contraction assumption (3.5) and (3.6) and condition (T) be satisfied. Then the sequence (U_n) is nested and $\cap_{n=1}^{\infty} U_n = \mathbb{D}$ which means that $U_n \to \mathbb{D}$. Furthermore $\tilde{y}_n \to f^*$ with $\tilde{y}_n \leq f^*$ and $f_n \searrow f^*$ as $n \to \infty$.*

Proof. See [33]. □

The similar proof for convergence of algorithm for global optimization of B-spline constrained problems is given in [13].

4. Improved algorithm with B-spline consistency techniques

We can apply the concept of consistency to each constraints of the problem to eliminate subboxes of the given box that cannot contains the solution. Let $q(x) = f(x_1, \ldots, x_n) = 0, x \in \mathbf{x}$ be a constraint that must be satisfied in an optimization problem we use B-spline box and hull consistency to eliminate subboxes of $\mathbf{x}$ that cannot contain a point satisfying $q(x) = f(x_1, \ldots, x_n) = 0$. The B-spline box and hull consistency can be applied for inequalities. Suppose that in place of the equality we have an inequality $q(x) \leq 0$. We can replace this inequality by $q(x) = [-\infty, 0]$ and obtain the equation $q(x) + [0, +\infty] = 0$. B-spline box consistency requires application of interval Newton method, generally it does not perform well when the variable bound is very wide. Next we present the B-spline box consistency (BsBC) and B-spline hull consistency (BsHC) techniques with an examples which show that B-spline hull consistency gives 60% pruning and B-spline box consistency gives 38% pruning in variable bound. These can be viewed as extensions of the ideas in [27] in the context of the polynomial B-spline based approach to global optimization.

4.1. B-spline hull consistency

In this subsection, we present the B-spline hull consistency to reduce the search region. B-spline hull consistency can be viewed as extension of interval hull consistency in the context of the B-spline form. We present next the procedure of interval hull consistency described in [14, 30].

Let us start by considering to apply hull consistency for domain reduction of a variable x_1 using a two variable equality constraint $h(x_1, x_2) = 0$,

$$h(x_1, x_2) = a_{0,0} + a_{1,0}x_1 + a_{0,1}x_2 + a_{1,1}x_1x_2 + \cdots + a_{0,2}x_2^2 + \cdots + a_{n,0}x_1^n + \cdots + a_{o,n}x_2^n = 0. \tag{4.1}$$

The implementation of the hull consistency involves the constraint inversion. To obtain constraint inversion we select term having only one variable with highest degree. Lets consider $a_{n,0}x_1^n$ then constraint inversion is given as,

$$a_{n,0}x_1^n = -a_{0,0} - a_{1,0}x_1 - a_{0,1}x_2 - a_{1,1}x_1x_2 - \cdots - a_{0,2}x_2^2 - \cdots - a_{o,n}x_2^n = h'(x_1, x_2).$$

Then we use B-spline expansion to obtain the range $\mathbf{h}'(\mathbf{x}_1, \mathbf{x}_2)$ of constraint inversion function $h'(x_1, x_2)$. The new value for interval $\mathbf{x}_1$ is estimated in [14] as

$$\mathbf{x}_{1,new} = \left(\frac{\mathbf{h}'(\mathbf{x}_1, \mathbf{x}_2)}{a_{n,0}}\right)^{1/n} \cap \mathbf{x}_1. \tag{4.2}$$

This procedure is repeated to contract the domain of x_2 variable using the same constraint function $h(x_1, x_2) = 0$. This domain box is further contracted in a similar way using the remaining equality constraints. The hull consistency method can also be applied to inequality constraints, we need only replace an inequality of the form $g(x) \leq 0$ by the equation $g(x) = [-\infty, 0]$ in [14]. Then B-spline hull consistency can be applied as before to this equality constraint. We illustrate the B-spline hull consistency method for equality constraint via an example.

Example 4.1. Consider the following equality constraint

$$h(x) = x_1^2 - 3x_1x_2^2 + 2.5x_1^2x_2^2 + 2x_2 - 2x_1x_2 = 0, \tag{4.3}$$

with $\mathbf{x}_1 = [1, 2], \mathbf{x}_2 = [1, 2]$. The B-spline coefficients of this equality constraint are

$$D_h(x) = \begin{bmatrix} 0.5 & 0.3750 & 0.0625 & -0.3125 & -0.75 & -1 \\ 0.75 & 0.6563 & 0.4375 & 0.1875 & -0.0936 & -0.25 \\ 1.6875 & 1.7344 & 1.8984 & 2.1328 & 2.4375 & 2.6250 \\ 3.0625 & 3.3281 & 4.0703 & 5.0234 & 6.1875 & 6.8750 \\ 4.8750 & 5.4375 & 6.9531 & 8.8594 & 11.1563 & 12.5 \\ 6 & 6.75 & 8.75 & 11.25 & 14.25 & 16 \end{bmatrix}.$$

Suppose we choose the term x_1^2 to contract the interval $\mathbf{x}_1$. From (4.3),

$$x_1^2 = 3x_1x_2^2 - 2.5x_1^2x_2^2 - 2x_2 + 2x_1x_2,$$

and the constraint inverse is

$$h_1(x) = 3x_1x_2^2 - 2.5x_1^2x_2^2 - 2x_2 + 2x_1x_2,$$

which is obtained by substituting 0 as coefficient of term x_1^2 in (4.3) and multiplying by -1 to equation (4.3), *i.e.*

$$h(x) = -0x_1^2 + 3x_1x_2^2 - 2.5x_1^2x_2^2 - 2x_2 + 2x_1x_2 = 0.$$

To get a new interval for x_1^2, the interval methods require the evaluation of $h_1(\mathbf{x})$, so computing the B-spline coefficients for $h_1(\mathbf{x})$ we get

$$D_{h_1}(x) = \begin{bmatrix} 0.5 & 0.6250 & 0.9375 & 1.3125 & 1.75 & 2 \\ 0.5 & 0.5938 & 0.8125 & 1.0625 & 1.3438 & 1.5 \\ 0.1875 & 0.1406 & -0.0234 & -0.2578 & -0.5625 & -1.75 \\ -0.4375 & -0.7031 & -1.4453 & -2.3984 & -3.5625 & -4.25 \\ -1.3750 & -1.9375 & -3.4531 & -5.3594 & -7.6563 & -9 \\ -2 & -2.75 & -4.75 & -7.25 & -10.25 & -12 \end{bmatrix},$$

giving the range enclosure for $h_1(\mathbf{x})$ as $\mathbf{h}' = [\min D_{h_1}(\mathbf{x}), \max D_{h_1}(\mathbf{x})] = [-12, 2]$. Thus, a new interval value $\mathbf{x}_1'$ for $\mathbf{x}_1$ is given by

$$(\mathbf{x}_1')^2 = [-12, 2].$$

Since $(\mathbf{x}_1')^2$ must be non-negative

$$\mathbf{x}_1' = \pm[0, 2]^{1/2} = [-1.412, 1.412].$$

The updated value of $\mathbf{x}_1$ is therefore

$$\mathbf{x}_1 = \mathbf{x}_1' \cap \mathbf{x}_1 = [1, 1.412].$$

Next we present B-spline hull consistency (BsHC) algorithm.

Algorithm 3: BsHC $(A_c, N_c, K_c, \mathbf{x}, s_{cv}, p, q)$.

Input : *Here A_c is a cell structure containing the coefficients array of all the constraints, N_c is a cell structure, containing degree vector N for all constraints. Where elements of degree vector N defines the degree of each variable occurring in all constraints polynomial, K_c is a cell structure containing vectors corresponding to all constraints, i.e. K_{g_i}, K_{h_j}. Where elements of this vector define the number of B-spline segments in each variable direction, the number of constraint variables s_{cv}, number of inequality constraints p and number of equality constraints q.*

Output: *A box* $\mathbf{x}$ *that is contracted using B-spline box consistency technique for the given constraints.*

Begin Algorithm

1 *Set* $r = 0$.
for $i \leftarrow 1$ **to** $p + q$ **do**
  for $j \leftarrow 1$ **to** $s_{cv}(i)$ **do**
(a) *Set* $r = r + 1$.
if $r > s_{cv}(i)$ **then**
$r = 1$.
end
(b) {*Formulate the constraint inverse polynomial*}
From the coefficient matrix $A_c(i)$, choose the monomial term in x_r i.e. $a_r x_r^{i_r}$ having the highest degree and substitute zero for its coefficient i.e $a_r = 0$, then multiply the coefficient matrix $A_c(i)$ by -1.
(c) {*Compute* $\mathbf{h}'$}
Compute the B-spline coefficients of the constraint inverse polynomial, and then obtain $\mathbf{h}'$*, as minimum to maximum of these B-spline coefficients. The algorithm in [8] is suggested for the computation.*
if $i < p + 1$ **then**
Compute an interval $w = [-\infty, 0] \cap \left[\min D_{c(i)}(x), \max D_{c(i)}(x)\right]$.
if $w = \emptyset$ **then**
set $\mathbf{x}' = \emptyset$ **& EXIT** *the algorithm.*
else
modify h' *as* $h' = h' + w$.
end
end
if $a_r \neq 0$ **then**
compute $\mathbf{x}'_r = \left(\frac{\mathbf{h}'}{a_r}\right)^{1/i_r} \cap \mathbf{x}_r$
else
EXIT *loop* j.
end

(d) *Update* $\mathbf{x}_r = \mathbf{x}_r \cap \mathbf{x}'_r$.
end
Set $r = 0$.
end
2 **return** $\mathbf{x}' = \mathbf{x}$
End Algorithm

4.2. B-spline box consistency

First of all, we recall the procedure of interval box consistency described in [14,30]. The implementation of interval box consistency involves the application of a one dimensional Newton method, to solve a single equation for a single variable. Let us start by applying box consistency to the following equality constraint

$$h(x) = 0,\ x \in \mathbf{x}. \tag{4.4}$$

We use box consistency to eliminate those subboxes of $\mathbf{x}$ that do not satisfy $h(x) = 0$. If we replace all the variables except rth variable by their interval bounds, we obtain the equation

$$h(x_r) = h(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, x_r, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_s) = 0. \tag{4.5}$$

If $0 \notin h(x_r)$ for x_r in some subinterval $\mathbf{x}'_r$ of $\mathbf{x}_r$, then we do not have consistency for $x_r \in \mathbf{x}'_r$ and the subbox $(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, \mathbf{x}'_r, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_s)$ can be deleted. If a subinterval $\mathbf{x}'_r$ of $\mathbf{x}_r$, cannot be deleted entirely, then it can be contracted by box consistency techniques, using interval Newton operator. Let $\mathbf{x}'_r = [a, b]$ and we seek to bound a_z and b_z in an interval $\mathbf{x}_r$. Contracting the interval bound involves left narrowing and right narrowing operations. The left narrowing operation is done to increase the lower bound a and the right narrowing operation is done to decrease the upper bound b. Consequently the width of the interval reduces.

Left narrowing can be done only when the interval range enclosure value of the constraint function $\mathbf{h}$ over the interval $(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, a, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_s)$ is

$$\mathbf{h}(a) = h(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, a, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_s) \not\ni 0. \tag{4.6}$$

If $0 \notin \mathbf{h}(a)$, then $a < a_z$ so we use a interval Newton method to remove points from the lower end of $\mathbf{x}_r$ that are less than a_z. Thus, the Newton result when expanding about the point a is [14]

$$\mathcal{N}(a, \mathbf{x}_r) = a - \mathbf{h}(a) / \left(\frac{\partial h(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, \mathbf{x}_r, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_s)}{\partial x_r} \right). \tag{4.7}$$

The new contracted interval can be obtained by intersecting the interval values of $\mathbf{x}_r$ and $\mathcal{N}$. That is,

$$\mathbf{x}_{r,new} = \mathbf{x}_r \cap \mathcal{N}(a, \mathbf{x}_r). \tag{4.8}$$

Similarly, the upper bound b can be contracted using the right narrowing operation. This procedure is repeated for $\mathbf{x}_r$, $r = 1, \ldots, s$, using the other equality constraints.

We can also apply box consistency to an inequality constraints, by replacing an inequality of the form $g(x) \leq 0$ by the equation $g(x) = [-\infty, 0]$ in [14], that is rewrite this an another function $h(x)$ given by

$$h(x) = g(x) - [-\infty, 0] = g(x) + [0, \infty] = 0. \tag{4.9}$$

Lets consider to apply box consistency for domain reduction of variable $\mathbf{x}'_r = [a, b]$ using equality constraint (4.9). To apply B-spline Newton contractor for $h(\mathbf{x}_r)$ for end point a, we required interval range enclosure $\mathbf{h}(a)$ of equality constraint (4.9). To obtain $\mathbf{h}(a)$ we first find interval range enclosure $\mathbf{g}(a)$ for $g(x)$ then the interval

range enclosure $\mathbf{h}(a)$ is obtained by considering $\min \mathbf{g}(a)$ and ∞ as lower and upper bound of $\mathbf{h}(a)$ respectively. Thus by applying B-spline Newton contractor for $h(x)$ for end point a, we obtain

$$\mathcal{N}(a, \mathbf{x}_r) = a - (\mathbf{h}(a)/h'_{x_r}) = a - \left([\min \mathbf{g}(a), \infty]/g'_{x_r}\right), \tag{4.10}$$

$$\mathbf{x}_{r,\text{new}} = \mathbf{x}_r \cap \mathcal{N}(a, \mathbf{x}_r) \tag{4.11}$$

where $h'_{x_r} = \left(\frac{\partial h(\mathbf{x}_1,\ldots,\mathbf{x}_{r-1},\mathbf{x}_r,\mathbf{x}_{r+1},\ldots,\mathbf{x}_s)}{\partial x_r}\right)$ and $g'_{x_r} = \left(\frac{\partial g(\mathbf{x}_1,\ldots,\mathbf{x}_{r-1},\mathbf{x}_r,\mathbf{x}_{r+1},\ldots,\mathbf{x}_s)}{\partial x_r}\right)$.

When we apply the B-spline box consistency to a selected variable of a multivariate constraint, then only that variable's domain will be contracted. The domains of the remaining variables will remain unaffected. We apply B-spline box consistency to each variable in turn, to get a contracted box in all variables. Suppose $h(x) = 0$ is the given equality constraint. We compute the B-spline coefficients array $D(\mathbf{x})$ for this constraint, and consider a variable direction, say the first one $\mathbf{x}_1 = [a, b]$. In the B-spline box consistency, we try to increase the value of a and decrease the value of b, thus effectively reducing the width of $\mathbf{x}_1$. The procedure to increase the value of a is listed in the below steps,

(1) Compute interval $h(a)$.
Corresponding to $x_1 = a$, find all B-spline coefficients in $D(\mathbf{x})$. The minimum to maximum of these B-spline coefficients gives an interval $h(a)$.
(2) Check $h(x)$ is feasible or infeasible at end point a.
If $0 \notin h(a)$ *i.e.* $h(a)$ is completely positive interval, means $h(x)$ is infeasible at the end point a else it is feasible.
(3) If $h(x)$ is infeasible at end point a.
Then search starting from a, along $\mathbf{x}_1 = [a, b]$, for the first point at which the constraint becomes just feasible *i.e.* we seek a zero of h. Let us denote this zero as $x_1 = a'$. Clearly $h(x)$ is infeasible over $[a, a')$ and so we can discard it to get a contraction $[a', b]$.
(4) Else (for feasible $h(x)$).
As $0 \in h(a)$, means $h(a)$ is not completely positive interval, then we cannot increase a. We instead switch over to the other end point b and try to decrease it in the same way as we try to increase a.

We illustrate the BsBC method for equality constraints via an example.

Example 4.2. Consider the following equality constraint

$$h(x) = 3x_1^2 - x_2 = 0,$$

with $\mathbf{x}_1 = [0.2, 1]$ and $\mathbf{x}_2 = [\ 0, 1]$. The B-spline coefficients of $h(x)$ are

$$D(x) = \begin{bmatrix} 0.12 & -0.2133 & -0.5467 & -0.88 \\ 0.24 & -0.0933 & -0.4267 & -0.76 \\ 0.72 & 0.3867 & 0.0533 & -0.28 \\ 1.44 & 1.1067 & 0.7733 & 0.44 \\ 2.4 & 2.0667 & 1.7333 & 1.4 \\ 3 & 2.667 & 2.333 & 2 \end{bmatrix}.$$

Note that we are constructing B-spline with *order* + 1 number of segments. Consider the application of BsBC along the first component direction, that is, along x_1. Along the direction x_1, the first row corresponds to $x_1 = a = 0.2$, and the sixth row corresponds to $x_1 = b = 1$. Along the first row, the minimum to maximum values of the B-spline coefficients, are -0.88 and 0.12 giving $\mathbf{h}(a) = [-0.88, 0.12]$. Along the sixth row, the minimum and maximum values of the B-spline coefficients, respectively are 2 and 3 giving $\mathbf{h}(b) = [2, 3]$. Since

$0 \in \mathbf{h}(a)$ the left end point cannot be increased. However, $0 \notin \mathbf{h}(b)$, hence the right end point can be decreased. The partial derivative in the direction x_1, that is, $\mathbf{h}'_{x_1}$ will be as follows

$$\mathbf{h}'_{x_1} = \begin{bmatrix} 1.2 & 2.4 & 3.6 & 4.8 & 6 \\ 1.2 & 2.4 & 3.6 & 4.8 & 6 \\ 1.2 & 2.4 & 3.6 & 4.8 & 6 \\ 1.2 & 2.4 & 3.6 & 4.8 & 6 \end{bmatrix}^T.$$

Hence $\mathbf{h}'_{x_1} = [1.2, 6]$. Now we can perform one iteration of B-spline Newton contractor as

$$\begin{aligned} \mathcal{N}(\mathbf{x}_1) &= b - (\mathbf{h}(b)/\mathbf{h}'_{x_1}) \\ &= 1 - \frac{[2,3]}{[1.2,6]} \\ &= [-1.5, 0.66]. \end{aligned}$$

Therefore, the updated value of $\mathbf{x}_1$ is

$$\begin{aligned} \mathbf{x}'_1 &= \mathcal{N}(\mathbf{x}_1) \cap \mathbf{x}_1 \\ &= [-1.5, 0.66] \cap [0.2, 1] \\ &= [0.2, 0.66]. \end{aligned}$$

Next we present B-spline box consistency (BsBC) algorithm.

Algorithm 4: BsBC $(A_c, N_c, K_c, \mathbf{x}, s_{cv}, p, q)$.

Input : *Here A_c is a cell structure containing the coefficients array of all the constraints, N_c is a cell structure, containing degree vector N for all constraints. Where elements of degree vector N defines the degree of each variable occurring in all constraints polynomial, K_c is a cell structure containing vectors corresponding to all constraints, i.e. K_{g_i}, K_{h_j}. Where elements of this vector define the number of B-spline segments in each variable direction, the number of constraint variables s_{cv}, number of inequality constraints p and number of equality constraints q.*

Output: *A box $\mathbf{x}$ that is contracted using B-spline box consistency technique for the given constraints.*

Begin Algorithm

1 *Set $r = 0$.*
for $i \leftarrow 1$ **to** $p + q$ **do**
  for $j \leftarrow 1$ **to** $s_{cv}(i)$ **do**
    Set $r = r + 1$.
    if $r > s_{cv}(i)$ **then**
      $r = 1$.
    end
(a) {*Compute B-spline coefficient for constraint polynomial*}
    $D_{c(i)}(x)$ using $(A_{c(i)}, N_{c(i)}, K_{c(i)}, \mathbf{x})$. The algorithm in [8] is suggested for the computation.
(b) *Set $a = \inf \mathbf{x}_r$, $b = \sup \mathbf{x}_r$.*
(c) {*Compute domain reduction for $\mathbf{x}_r$*}
    for $l \leftarrow \{a, b\}$ **do**

(d) {*Obtain the B-spline range enclosure,* $\mathbf{h}(l)$}
Obtain $\mathbf{h}(l)$*, as the minimum to maximum of the B-spline coefficient at* $x_r = l$*, occurring along the first row in variable direction* r *of the B-spline coefficient array* $D_{c(i)}(x)$.

(e) {*Obtain the derivative enclosure* h'_{x_r}}
Use equation (2.14) and the B-spline coefficient array $D_{c(i)}(x)$*, to compute the derivative enclosure* h'_{x_r}*, in the direction* x_r.

if $i < p + 1$ **then**
  modify $\mathbf{h}(l)$ *as* $\mathbf{h}(l) = [\min \mathbf{h}(l), \infty]$.
end
if $0 \in \mathbf{h}(l)$ *and* $l == a$ **then**
  we cannot increase a *and try from the right end point* b *of the interval* $\mathbf{x}_r$.
else if $0 \in \mathbf{h}(l)$ *and* $l == b$ **then**
  we cannot decrease b, **go to** sub step (g).
end

(f) *Do one iteration of the univariate B-spline Newton method*

$$\mathcal{N}(\mathbf{x}_r) = l - \left(\mathbf{h}(l)/\mathbf{h}'_{x_r}\right),$$
$$\mathbf{x}'_{r_l} = \mathbf{x}_r \cap \mathcal{N}(\mathbf{x}_r).$$

if $\mathbf{x}'_{r_l} = \emptyset$ **then**
  there is no zero of h *in entire interval* $\mathbf{x}_r$*, and hence the constraint* h *is infeasible over box* $\mathbf{x}$. **Exit** *the algorithm in this case with* $\mathbf{x}' = \emptyset$.
end

end

(g) {*Compute* $\mathbf{x}'_r$ *as follows*}
$\mathbf{x}'_r = \mathbf{x}'_{r_a} \cap \mathbf{x}'_{r_b}$*, if both* $\mathbf{x}'_{r_a}$ *and* $\mathbf{x}'_{r_b}$ *are computed.*
$\mathbf{x}'_r = \mathbf{x}'_{r_a}$ *or* $\mathbf{x}'_{r_b}$*, which ever is computed.*
$\mathbf{x}'_r = \mathbf{x}_r$*, if both* $\mathbf{x}'_{r_a}$ *and* $\mathbf{x}'_{r_b}$ *are not computed.*

(h) *Update* $\mathbf{x}_r = \mathbf{x}_r \cap \mathbf{x}'_r$.

end
Set $r = 0$.

end

3 **return** $\mathbf{x}' = \mathbf{x}$

End Algorithm

4.3. Proposed improved algorithm for constrained global optimization

To apply the proposed B-spline box and B-spline hull consistency algorithms using the basic algorithm (see Sect. 3.4), we modify step 7 of basic algorithm to step *7 given below. We refer to the resulting modified algorithm as improved algorithm.

***7** {Start iteration}
if $\mathcal{L} = \emptyset$ **then**
  go to 13
  else
  pick the last item from $\mathcal{L}$*, denote it as* $\{\mathbf{b}, D_o(\mathbf{b}), D_{g_i}(\mathbf{b}), D_{h_j}(\mathbf{b}), F\}$*, and delete this item entry from* $\mathcal{L}$*.*
end

(a) {*Apply B-spline hull consistency to contract domain box*}
Apply the algorithm BsHC, $\mathbf{b}' = BsHC\,(N_c, A_c, K_c, \mathbf{b}, s_{cv}, p, q)$.

(b) {*Compute B-spline coefficients for new box*}
Compute the B-spline coefficients array of objective and constraint polynomials on the box $\mathbf{b}'$*, respectively as* $D_o(\mathbf{b}'), D_{g_i}(\mathbf{b}')$ *and* $D_{h_j}(\mathbf{b}'), i = 1, \ldots, p, j = 1, \ldots, q$.

(c) {*Apply B-spline box consistency to contract domain box*}
Apply the algorithm BsBC, $\mathbf{b}'' = BsBC\,(N_c, A_c, K_c, \mathbf{b}', s_{cv}, p, q)$

(d) {*Compute B-spline coefficients for new box*}
Compute the B-spline coefficients array of objective and constraint polynomials on the box $\mathbf{b}''$, respectively as $D_o(\mathbf{b}''), D_{g_i}(\mathbf{b}'')$ and $D_{h_j}(\mathbf{b}''), i = 1, \ldots, p, j = 1, \ldots, q$.

---

5. Numerical test

In this section, we present the result and analysis of our tests. The computations are done on a PC Intel i3-370M 2.40 GHz processor, 6 GB RAM, while the algorithms are implemented in MATLAB [24]. An accuracy $\epsilon = 10^{-6}$ is prescribed for computing the global minimum and minimizer(s) in each test problem. For the tests, we select 11 benchmark optimization problems taken from [1, 4, 5, 10, 16] (described in Appendix A). Table 1 reports the global minimum obtained with the proposed algorithms.

5.1. Comparisons between basic and improved proposed algorithms

First, we test and compare the performance of the basic and improved B-spline constrained global optimization algorithms on a set of 11 test problems. The performance metrics are taken as the number of subdivisions and computation time (in seconds) required to compute the global minimum for the problem. These values are reported in Tables 2 and 3 respectively. For these metrics, we give the *percent reduction* computed as

$$\%\ \text{Reduction} = \frac{\text{PMBA} - \text{PMIA}}{\text{PMBA}} \times 100$$

where PMBA is the performance metric with the basic algorithm, and PMIA is the performance metric with the improved algorithm. In Table 2, we compare the number of subdivisions required for obtaining global minima using the basic and improved algorithms. It is found that except problem P2 for all test problems the number of subdivisions required to compute the global minimum reduces in the range of 2.22% to 96%. Whereas *basic* algorithm is unable to solve problem P10.

TABLE 1. Test problems with their domains, dimensions, and the global minimum with the proposed B-spline optimization algorithm.

Ex.	Test Functions	Dim.	Con.	Domain	Global minimum
1	P1	2	1	$[0, 0.5]^2$	0.06857
2	P2	2	2	$[-1, 1]^2$	-4
3	P3	2	2	$[0, 3][0, 4]$	-5.50796
4	P4	2	2	$[0, 1]^2$	0.7418
5	P5	3	3	$[0, 2][0, 10][0, 3]$	-4
6	P6	3	4	$[-5, 5]^3$	0
7	P7	4	7	$[0, 5]^4$	1.089
8	P8	5	1	$[0, 1]^5$	-17
9	P9	6	2	$[0, 1]^5[0, 20]$	-361.5
10	P10	7	11	$[2.6, 3.6][0.7, 0.8][17, 28][7.3, 8.3]$ $[7.3, 8.3][2.9, 3.9][5, 5.5]$	-2994.47
11	STP1	5	1	$[0, 1]^5$	-11

In Table 3, we compare the computation time required for obtaining the global minima using the basic and improved algorithm, it is observed that the improved algorithm computes the results slower than that of the basic algorithm except for the problems number 3, 6, 7 and 9. The use of B-spline box and B-spline hull consistency algorithms found to be very effective in pruning the search domain by discarding those subboxes that cannot contain global minimizers. As shown in Section 4.3 during each iteration improved algorithm requires to compute B-spline coefficients more than once (like steps $^*7(b)$ and $^*7(d)$ in improved algorithm) as compared to basic algorithm. In improved algorithm during each iteration B-spline coefficients are computed after domain pruning by the B-spline box and B-spline hull consistency. Also, in each iteration of B-spline hull consistency algorithm requires to compute B-spline coefficients after each constraint inversion (*cf.* 4.1) to contract the variable bounds. All the variable bounds are contracted using each constraint function individually. This is also evident from the results in the Table 3 for an improved algorithm. We would like to mention that, a more sophisticated implementation of B-spline box and hull consistencies can alleviate the computational bottle neck associated with them. In case of problem P2 *improved* algorithm is unable to minimize the number of subdivision still it is slow because B-spline box consistency (Newton method) is fast when the initial intervals are small. Unfortunately, the running time of algorithm increases linearly with the size of the initial interval [36].

5.2. Comparison with GloptiPoly and other NLP solvers

Next, we compare the optimal value of global minimum found using the proposed B-spline algorithms (*basic* & *improved*) with CENSO, GloptiPoly [16, 20] and BARON, LINDO Global and SCIP NLP solvers. For the current tests, we consider the above set of 11 test functions. The idea is to investigate where same optimal value of global minimum can be found with the considered methods. The CENSO solver implements the algorithm in [13] and is available on Github: `https://github.com/bgrimstad/censo`. The GAMS and GloptiPoly source code for these problems are available at [`https://bit.ly/2YHonrh`].The GAMS interface for BARON, LINDO Global and SCIP is available through the NEOS server [28].

Table 4 report only those test functions from the above set of 11 test functions for which the proposed algorithms provides the optimal value of global minimum compare to CENSO, GloptiPoly and NLP solvers. The bold values in the table indicate the local minimum value. For P8 and STP1 BARON, LINDOGlobal and SCIP are able to capture the local minimum, where CENSO, GloptiPoly and proposed algorithms found the

TABLE 2. Performance comparison of the number of subdivisions required by the basic and improved B-spline algorithms.

Ex.	Test functions	Dim.	Con.	Basic Algo.	Improved Algo.	Subdivision % reduction
1	P1	2	1	4377	2601	40.57
2	P2	2	2	45	44	2.22
3	P3	2	2	155	34	78.06
4	P4	2	2	162	75	53.70
5	P5	3	3	693	287	58.58
6	P6	3	4	6331	461	92.71
7	P7	4	7	1143	45	96.06
8	P8	5	1	211	166	21.32
9	P9	6	2	570	119	79.12
10	P10	7	11	*	106	–
11	STP1	5	1	344	222	35.46

Notes. (⋆)Indicates that the algorithm did not give the result even after one hour and therefore terminated.

TABLE 3. Performance comparison of the computational time taken by the basic and improved B-spline algorithms.

Ex.	Test functions	Dim.	Con.	Basic Algo.	Improved Algo.	Computation time % reduction
1	P1	2	1	114.86	160.96	−40.39
2	P2	2	2	1.45	2.68	−84.82
3	P3	2	2	6.18	3.36	45.63
4	P4	2	2	2.89	3.59	−24.22
5	P5	3	3	20.40	23.90	−17.15
6	P6	3	4	390.13	48.22	87.64
7	P7	4	7	66.52	5.61	91.56
8	P8	5	1	7.19	13.35	−85.67
9	P9	6	2	24.32	13.08	46.21
10	P10	7	11	#	86.75	–
11	STP1	5	1	10.92	28.79	−163.64

Notes. (#)Indicates that the algorithm did not give the result even after one hour and therefore terminated.

TABLE 4. Comparison of optimal value of global minimum obtained using the proposed B-spline algorithm with CENSO, GloptiPoly 3.7 and state-of-the-art NLP solvers.

Sr.	Test function	B-spline algorithm	CENSO	BARON	LINDO Global	SCIP	GloptiPoly (Used relaxation order)
1	P8	−17	−17	**−16.5**	**−16.5**	−17	−17 (RO = 3)
2	P10	−2994.47	−2994.47	**−2994.38**	**−2994.38**	**−2994.38**	**
3	STP1	−11	−11	**−10.60**	**−10.60**	−11	−11 (RO = 3)

Notes. (⋆⋆)Indicates that the solver is unable to solve the test problem. The bold values in the table show the solutions missed by BARON, LINDO Global, and SCIP.

correct global minimum. In P10, BARON, LINDOGlobal and SCIP are able to capture the local minimum and GloptiPoly did not give the result even after an hour and is therefore terminated. In P8 and STP1 (STP1 is a problem constructed by authors based on P8), the relaxation order for GloptiPoly had to be systematically increased to a high order to obtain convergence to the final results.

The proposed algorithms and CENSO find the optimal value of global minimum for all the test problem considered. Where as BARON and LINDO Global are unable to capture optimal value of global minimum in all the 3 test functions even after specifying a large value of the AbsConFeasTol option in GAMS. The SCIP solver is unable to find the optimal value of global minimum for P10 test problem.

For GloptiPoly, the relaxation order need to be systematically increased to a higher order to obtain convergence to the optimal value of global minimum. As the dimension and number of constraints in problem increases, to obtain convergence the relaxation order is gradually increased to a value greater than or equal to the dimension of the problem. For medium dimension problem (like 7 with 11 constraints) GloptiPoly exhausts with memory even with small relaxation order. Whereas other NLP solvers may be able to solve large dimension problems with non-optimal value of global minimum. Here, we want to mention that proposed algorithms are implemented in MATLAB, whereas except GloptiPoly all other NLP solvers mentioned above are implemented in different languages and therefore comparison based on computation time between proposed algorithms and these NLP solvers is not carried out.

6. Conclusion

We presented the basic and improved algorithm for constrained global optimization of multivariate polynomials using polynomial B-spline form. The performance of the basic and improved algorithm are tested and compared on 11 test problems. The test problems had dimensions ranging from 2 to 7 and number of constraints varying from 1 to 11. The results of the test show that improved algorithm is more efficient, in terms of a number of subdivisions with little extra computational time. We also compared the optimal value of global minimum obtained with the proposed algorithms with CENSO, GloptiPoly and some of the well known NLP solvers, on a set of 3 test problems. The result shows the superiority of the proposed algorithm and CENSO solver, in that it captures the global minimum to user specified accuracy. One possible extension of this research work is to investigate the performance of proposed B-spline approach for solving problems with more number of variables. This problem will be addressed in a future work.

Appendix A.

Test problems

We denote the objective functions as $f(x)$ and the initial bounds as $\mathbf{x}_i$, where $i = 1, 2, \ldots, s$.

(1) Test problem P1 [16]:

$$\begin{aligned} &\min\ f(x) = x_1^2 + x_2^2, \\ &\text{s.t.} \\ &\quad -8x_1^3 - 4x_1^2x_2 - 2x_1x_2^2 - 28x_1^2 + x_1x_2 - 3x_2^2 - 22x_1 - 7x_2 + 8 = 0, \end{aligned}$$

where $\mathbf{x}_1 = [0, 0.5], \mathbf{x}_2 = [0, 0.5]$.

(2) Test problem P2 [10]:

$$\begin{aligned} &\min\ f(x) = 10(x_1^2 - x_2)^2 - (x_1 - 1)^2, \\ &\text{s.t.} \\ &\quad 3x_1 + 4x_2 - 25 \le 0, \\ &\quad x_1 - x_1x_2 = 0, \end{aligned}$$

where $\mathbf{x}_1 = [-1, 1], \mathbf{x}_2 = [-1, 1]$.

(3) Test problem P3 [5]:

$$\begin{aligned} &\min\ f(x) = -x_1 - x_2, \\ &\text{s.t.} \\ &\quad x_2 \le 2 + 2x_1^4 - 8x_1^3 + 8x_1^2, \\ &\quad x_2 \le 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 + 36, \end{aligned}$$

where $\mathbf{x}_1 = [0, 3], \mathbf{x}_2 = [0, 4]$.

(4) Test problem P4 [1]:

$$\begin{aligned} &\min\ f(x) = 2x_1 + x_2, \\ &\text{s.t.} \\ &\quad -16x_1x_2 \le 0, \\ &\quad -4x_1^2 - 4x_2^2 \le -1, \end{aligned}$$

where $\mathbf{x}_1 = [0, 1], \mathbf{x}_2 = [0, 1]$.

(5) Test problem P5 [5]:

$$\begin{aligned} &\min\ f(x) = -2x_1 + x_2 - x_3, \\ &\text{s.t.} \\ &\quad \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{y}^T\mathbf{A}\mathbf{x} + \|\mathbf{y}\|^2 - 0.25\|\mathbf{b} - \mathbf{z}\|^2 \ge 0, \\ &\quad x_1 + x_2 + x_3 - 4 \le 0, \\ &\quad 3x_2 + x_3 - 6 \le 0, \end{aligned}$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ -2 & 1 & -1 \end{pmatrix},\ \mathbf{b} = (3, 0, -4)^T,$$

and

$$\mathbf{x}_1 = [0, 2], \mathbf{x}_2 = [0, 10], \mathbf{x}_3 = [0, 3], \mathbf{y} = (1.5, -0.5, -5)^T,\ \mathbf{z} = (0, -1, -6)^T.$$

(6) Test problem P6 [10]:

$$\begin{aligned} &\min\ f(x) = x_3, \\ &\text{s.t.} \\ &\quad 2x_2^2 + 4x_1x_2 - 42x_1 + 4x_1^3 - x_3 \le 14, \\ &\quad -2x_2^2 - 4x_1x_2 + 42x_1 - 4x_1^3 - x_3 \le -14, \\ &\quad 2x_1^2 + 4x_1x_2 - 26x_2 + 4x_2^3 - x_3 \le 22, \\ &\quad -2x_1^2 - 4x_1x_2 + 26x_2 - 4x_2^3 - x_3 \le -22, \end{aligned}$$

where $\mathbf{x}_i = [-5, 5]$, $i = 1, \ldots, 3$.

(7) Test problem P7 [10]:

$$\begin{aligned} &\min\ f(x) = x_4, \\ &\text{s.t.} \end{aligned}$$

$$
\begin{aligned}
&x_1^4x_2^4 - x_1^4 - x_2^4x_3 = 0,\\
&1.4 - 0.25x_4 - x_1 \le 0,\\
&-1.4 - 0.25x_4 + x_1 \le 0,\\
&1.5 - 0.2x_4 - x_2 \le 0,\\
&-1.5 - 0.2x_4 + x_2 \le 0,\\
&0.8 - 0.2x_4 - x_3 \le 0,\\
&-0.8 - 0.2x_4 + x_3 \le 0,
\end{aligned}
$$

where $\mathbf{x}_i = [0,5], i = 1,\ldots,4$.

(8) Test problem P8 [4]:

$$
\begin{aligned}
&\min\ f(x) = \mathbf{c}^T\mathbf{x} - 0.5\mathbf{x}^T\mathbf{Q}\mathbf{x},\\
&\text{s.t.}\\
&\quad 20x_1 + 12x_2 + 11x_3 + 7x_4 + 4x_5 \le 40,
\end{aligned}
$$

where

$$\mathbf{c} = (42, 44, 45, 47, 47.5)^T, \mathbf{Q} = 100\mathbf{I},$$

and $\mathbf{x}_i = [0,1], i = 1,\ldots,5$.

(9) Test problem P9 [5]:

$$
\begin{aligned}
&\min\ f(x) = \mathbf{c}^T\mathbf{x} - 0.5\mathbf{x}^T\mathbf{Q}\mathbf{x} - 10y,\\
&\text{s.t.}\\
&\quad 6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 \le 6.5,\\
&\quad 10x_1 + 10x_3 + x_6 \le 20,
\end{aligned}
$$

where

$$\mathbf{c} = (-10.5, -7.5, -3.5, -2.5, -1.5)^T, \mathbf{Q} = \mathbf{I},$$

and $\mathbf{x}_1 = [0,1], \mathbf{x}_2 = [0,1], \mathbf{x}_3 = [0,1], \mathbf{x}_4 = [0,1], \mathbf{x}_5 = [0,1], \mathbf{x}_6 = [0,20]$.

(10) Test problem P10 [4]:

$$
\begin{aligned}
&\min\ f(x) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right)\\
&\qquad + 7.477\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right),\\
&\text{s.t.}\\
&\quad x_1x_2^2x_3 \ge 27,\\
&\quad x_1x_2^2x_3^2 \ge 397.5,\\
&\quad x_2x_6^4x_3x_4^{-3} \ge 1.93,\\
&\quad x_2x_7^4x_3x_5^{-3} \ge 1.93,\\
&\quad \left\{\left(745x_4x_2^{-1}x_3^{-1}\right)^2 + 16.91 * 10^6\right\}^{0.5}/\left(.1x_6^3\right) \le 1100,\\
&\quad \left\{\left(745x_5x_2^{-1}x_3^{-1}\right)^2 + 157.5 * 10^6\right\}^{0.5}/\left(.1x_7^3\right) \le 850,\\
&\quad x_2x_3 \le 40,\\
&\quad x_1x_2^{-1} \ge 5,
\end{aligned}
$$

$$x_1 x_2^{-1} \leq 12,$$
$$1.5x_6 - x_4 \leq -1.9,$$
$$1.1x_7 - x_5 \leq -1.9,$$

where

$$\mathbf{x}_1 = [2.6, 3.6], \mathbf{x}_2 = [0.7, 0.8], \mathbf{x}_3 = [17, 28], \mathbf{x}_4 = [7.3, 8.3],$$
$$\mathbf{x}_5 = [7.3, 8.3], \mathbf{x}_6 = [2.9, 3.9], \mathbf{x}_7 = [5, 5.5]$$

Special test problem

(1) STP1:

$$\min f(x) = -80x_1^2 - 80x_2^2 - 80x_3^2 - 80x_4^2 - 80x_5^2 + 72x_1 + 74x_2 + 76x_3 + 77x_4 + 77.5x_5,$$
s.t.
$$50x_1 + 42x_2 + 41x_3 + 37x_4 + 34x_5 \leq 90,$$

where $\mathbf{x}_i = [0, 1], i = 1, \ldots, 5.$

References

[1] E.G. Birgin, C. Floudas and J.M. Martínez, Global minimization using an Augmented Lagrangian method with variable lower-level constraints. *Math. Program.* **125** (2010) 139–162.

[2] M.J. Cloud, R.E. Moore and R.B. Kearfott, Introduction to Interval Analysis. Siam, Philadelphia (2009).

[3] C. De Boor, On Calculating with B-splines. *J. Approximation Theory* **6** (1972) 50–62.

[4] C.A. Floudas and P.M. Pardalos, A Collection of Test Problems for Constrained Global Optimization Algorithms. Vol. 455. Springer (1990).

[5] C.A. Floudas, P.M. Pardalos, C. Adjiman, W.R. Esposito, Z.H. Gümüs, S.T. Harding, J.L. Klepeis, C.A. Meyer and C.A. Schweiger, Handbook of Test Problems in Local and Global Optimization. Springer Science & Business Media (2013).

[6] J. Garloff, The Bernstein algorithm. *Interval Comput.* **6** (1993) 154–168.

[7] D.D. Gawali, A. Zidna and P.S.V. Nataraj, Solving nonconvex optimization problems in systems and control: a polynomial B-spline approach. In: Modelling, Computation and Optimization in Information Systems and Management Sciences. Springer (2015) 467–478.

[8] D.D. Gawali, A. Zidna and P.S.V. Nataraj, Algorithms for unconstrained global optimization of nonlinear (polynomial) programming problems: the single and multi-segment polynomial B-spline approach. *Comput. Oper. Res.* **87** (2017) 205–220.

[9] D.D. Gawali, B.V. Patil, A. Zidna and P.S.V. Nataraj, A B-spline global optimization algorithm for optimal power flow problem. In: World Congress on Global Optimization. Springer (2019) 58–67.

[10] Global library. available online at http://www.gamsworld.org/global/globallib.

[11] B. Grimstad, A MIQCP formulation for B-spline constraints. *Optim. Lett.* **12** (2018) 713–725.

[12] B. Grimstad and B.R. Knudsen, Mathematical programming formulations for piecewise polynomial functions. *J. Global Optim.* **77** (2020) 455–486.

[13] B. Grimstad and A. Sandnes, Global optimization with spline constraints: a new branch-and-bound method based on B-splines. *J. Global Optim.* **65** (2016) 401–439.

[14] E. Hansen and G. Walster, Global Optimization Using Interval Analysis, 2nd edition. Revised and Expanded. Vol. 264, Marcel DEKKER, INC. New York (2004).

[15] D. Henrion and J.B. Lasserre, Gloptipoly: global optimization over polynomials with matlab and sedumi. *ACM Trans. Math. Softw. (TOMS)* **29** (2003) 165–194.

[16] D. Henrion and J.B. Lasserre, Solving nonconvex optimization problems. *IEEE Control Syst. Mag.* **24** (2004) 72–83.

[17] R. Horst and P.M. Pardalos, Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht, The Netherlands (1995).

[18] L. Jaulin, Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics. Vol. 1. Springer Science & Business Media (2001).

[19] R.B. Kearfott, Rigorous Global Search: Continuous Problems. Vol. 13. Springer Science & Business Media (2013).

[20] J.B. Lasserre, Global optimization with polynomials and the problem of moments. *SIAM J. Optim.* **11** (2001) 796–817.

[21] Q. Lin and J. Rokne, Methods for bounding the range of a polynomial. *J. Comput. Appl. Math.* **58** (1995) 193–199.

[22] Q. Lin and J. Rokne, Interval approximation of higher order to the ranges of functions. *Comput. Math. App.* **31** (1996) 101–109.

[23] T. Lyche and K. Morken, Spline Methods Draft. Department of Informatics, Centre of Mathematics for Applications, University of Oslo (2008).

[24] Mathworks Inc., MATLAB version 8.0.0.783 (R 2012 b), Inc. Natick, Massachusetts, United States (2012).
[25] D. Michel, H. Mraoui, D. Sbibih and A. Zidna, Computing the range of values of real functions using B-spline form. *Appl. Math. Comput.* **233** (2014) 85–102.
[26] P.S.V. Nataraj and M. Arounassalame, An algorithm for constrained global optimization of multivariate polynomials using the Bernstein form and John optimality conditions. *Opsearch* **46** (2009) 133–152.
[27] P.S.V. Nataraj and M. Arounassalame, Constrained global optimization of multivariate polynomials using Bernstein branch and prune algorithm. *J. Global Optim.* **49** (2011) 185–212.
[28] NEOS Server for optimization. http://www.neos-server.org/neos/solvers/ (2018).
[29] S. Park, Approximate branch-and-bound global optimization using B-spline hypervolumes. *Adv. Eng. Softw.* **45** (2012) 11–20.
[30] B.V. Patil, *Global optimization of polynomial mixed-integer nonlinear problems using the Bernstein form.* Ph.D. thesis, Indian Institute of Technology, Bombay (2012).
[31] B.V. Patil, P.S.V. Nataraj and S. Bhartiya, Global optimization of mixed-integer nonlinear (polynomial) programming problems: the bernstein polynomial approach. *Computing* **94** (2012) 325–343.
[32] H. Ratschek and J. Rokne, Computer Methods for the Range of Functions. Ellis Horwood Limited, Chichester, England (1984).
[33] H. Ratschek and J. Rokne, New Computer Methods for Global Optimization. Ellis Horwood Limited, Chichester, England (1988).
[34] C.K. Shene, CS3621 Introduction to computing with geometry notes. http://www.cs.mtu.edu/shene/COURSES/cs3621/NOTES/ (2014).
[35] R. Vaidyanathan and M. El-Halwagi, Global optimization of nonconvex nonlinear programs via interval analysis. *Comput. Chem. Eng.* **18** (1994) 889–897.
[36] P. Van Hentenryck, D. McAllester and D. Kapur, Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal.* **34** (1997) 797–827.

Subscribe to Open (S2O)
A fair and sustainable open access model

This journal is currently published in open access under a Subscribe-to-Open model (S2O). S2O is a transformative model that aims to move subscription journals to open access. Open access is the free, immediate, online availability of research articles combined with the rights to use these articles fully in the digital environment. We are thankful to our subscribers and sponsors for making it possible to publish this journal in open access, free of charge for authors.

Please help to maintain this journal in open access!

Check that your library subscribes to the journal, or make a personal donation to the S2O programme, by contacting subscribers@edpsciences.org

More information, including a list of sponsors and a financial transparency report, available at: https://www.edpsciences.org/en/maths-s2o-programme