The required proposed algorithms are given here without detailed explanation.

1 Algorithm for B-spline coefficient computation

1.1 Algorithm for knot vector generation

Algorithm Knotvector_ generation:

Knotvector $(1:(2\times n)+k+1)$ = Knotvector_ generation $(n, k, \mathbf{x}_r)$

Inputs: Degree n of the polynomial, number of segments k and domain $\mathbf{x}_r := [\underline{\mathbf{x}_r}, \overline{\mathbf{x}_r}]$.

Output: The knot vector as an output.

BEGIN Algorithm

```
1. Set $a = \inf(\mathbf{x}_r)$, $b = \sup(\mathbf{x}_r)$
2. Knot_vector$(1,1)$=$a$
3. For $i$= $1 : k-1$
   (a) Knot_vector $(1, i+1)$ =$a + (i \times (b-a)/k)$
4. End (for $i$-loop)
5. Knot_vector$(1, k+1)$=$b$
6. Knot_vector $(1, (2\times n)+k+1) = [Knot\_vector(1,1)^n, Knot\_vector, Knot\_vector(1,k+1)^n]$
7. RETURN Knotvector=Knot_vector
```

END Algorithm

1.2 Algorithm for Pi matrix

Algorithm Pi_ matrix:

Pi $(1:n+k, 1:(n+1)) = Pi\_\ matrix(Knotvector, k, n)$

Inputs: Degree n of the polynomial, segment number k and Knotvector.

Output: The matrix Pi.

BEGIN Algorithm

```
1. For $j = 1 : (n+k)$
   (a) {Compute knotpart} as follows
       knotpart = Knotvector$(1, j+1 : j+n)$
   (b) For $d = 1 : n+1$
        i. {Compute symmetric polynomial value}
           $\sigma\_$ value = Symmetric_ polynomial_ value $(knotpart, d)$
       ii. $Pi'(j,d) = \sigma\_\ value / \binom{n}{d}$
      iii. End (for $d$-loop)
   (c) End (for $j$-loop)
2. RETURN Pi = $Pi'$
```

END Algorithm

1.3 Algorithm for Symmetric_ polynomial_ value

Algorithm Symmetric_ polynomial_ value:

σ_ value = Symmetric_ polynomial_ value(Knotpart, d)

Inputs: knotpart and degree d of symmetric polynomial.

Output: The value of symmetric polynomial σ_ value.

```
BEGIN Algorithm

  1. Set $A$ = knotpart
  2. If $A = \emptyset$ and $d = 1$
     $\sigma' = 1$
  3. Else $\sigma' = \begin{pmatrix} A \\ d \end{pmatrix}$
  4. $\sigma' = \sum\{\prod(\sigma'(1 : end, :))\}$
  5. RETURN $\sigma$_ value = $\sigma'$

END Algorithm
```

1.4 Algorithm for B-spline_ coefficient_ matrix

Algorithm B-spline_ coefficient_ matrix:

$D(x)$ = B-spline_ coefficient_ matrix $(A, \mathbf{N}, K, \mathbf{x})$
Inputs: Matrix A of coefficients of polynomial in power form, degree $\mathbf{N}$ of polynomial, segment number K, and the s - dimensional domain box $\mathbf{x}$.
Output: The B-spline coefficient matrix $D(x)$.

```
BEGIN Algorithm

  1. For $i$ = 1:$s$
     (a) {Compute Knotvector}
         $Knotvector_i(1 : (2 \times n_i) + k_i + 1)$ = Knotvector_ generation $(n_i, k_i, \mathbf{x}_i)$
     (b) {Compute Pi matrix}
         $Pi_i(1 : n_i + k_i, 1 : (n_i + 1)) = Pi\_ \; matrix(Knotvector_i, k_i, n_i)$
  2. End (for $i$-loop)
  3. For $j$ = 1:$s$
     (a) $A = Pi_j(1 : n + k, 1 : n + 1) \times A(0 : n, 0 : (n + 1)^{s-1} - 1)$
     (b) Transpose $A$
     (c) Reshape $A$ to the required matrix shape.
  4. End (for $j$-loop)
  5. RETURN $D(x) = A$

END Algorithm
```

2 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.

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 $h_j(x) = 0$ are replaced by relaxed constraints $h_j(x) \in [-\epsilon_{zero}, \epsilon_{zero}], j = 1, 2, \ldots, q$, where $\epsilon_{zero} > 0$ is a very small number.

The basic algorithm works as follows. We start the algorithm by computing the B-spline segments for each variable occurring in the objective, inequality and equality polynomials. That is, as K_o, K_{g_i} and K_{h_j}, where $K = [k_1, \ldots, k_s]$. 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 polynomials 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 $\mathcal{L}^{sol}$ 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 $\mathbb{D}_{g_i}(\mathbf{b}_r) \leq 0$,$0 \in \mathbb{D}_{h_j}(\mathbf{b}_r)$,$\mathbb{D}_{h_j}(\mathbf{b}_r) \subseteq [-\epsilon_{zero}, \epsilon_{zero}]$ 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 $0 \notin \mathbb{D}_{h_j}(\mathbf{b}_r)$,$\mathbb{D}_{h_j}(\mathbf{b}_r) \nsubseteq [-\epsilon_{zero}, \epsilon_{zero}]$ 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 Remark 1 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 $\mathcal{L}^{sol}$, else we continue the algorithm until the list $\mathcal{L}$ becomes empty.

Remark 1. The flag vector F is used to make the algorithm more efficient. Consider, i^{th} 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 f^{th} inequality or equality constraint is satisfied for the box.
- $F_f = 0$ if the f^{th} inequality or equality constraint is not yet been verified for the box.

Algorithm : Basic Algorithm for constrained global optimization

$[\hat{p}, z^{(i)}] = \text{Basic } (A_c, \mathbf{N_c}, K_c, \mathbf{x}, \epsilon, \epsilon_{zero})$

Inputs: Here A_c is a cell structure containing the coefficients array of objective and all the constraints polynomial, $\mathbf{N}_c$ is a cell structure, containing degree vector $\mathbf{N}$ for objective and all constraints. Where elements of degree vector $\mathbf{N}$ defines the degree of each variable occurring in objective 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}. Where elements of this vector define the number of B-spline segments in each variable direction, the initial box $\mathbf{x} \in \mathbb{IR}^s$, the tolerance limit ϵ and tolerance parameter ϵ_{zero} to which the equality constraints are to be satisfied.

Outputs: 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 $K = \mathbf{N} + 2$.

```
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.
3. {Initialize current minimum estimate}
   Initialize the current minimum estimate $\tilde{p} = \max D_o(\mathbf{x})$.
4. {Set flag vector}
   Set $F = (F_1, \dots, F_p, F_{p+1}, \dots, F_{p+q}) := (0, \dots, 0)$
5. {Initialize lists}
   Set $\mathcal{L} \leftarrow \{\mathbf{x}, D_o(\mathbf{x}), D_{g_i}(\mathbf{x}), D_{h_j}(\mathbf{x}), F\}$, $\mathcal{L}^{sol} \leftarrow \{\}$.
6. {Sort the list $\mathcal{L}$ }
   Sort the list $\mathcal{L}$ in descending order of $(\min D_o(\mathbf{x}))$.
7. {Start iteration}
   If $\mathcal{L}$ is empty go to step 12. Otherwise 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 $\mathcal{L}$.
8. {Perform cut-off test}
   Discard the item $\{\mathbf{y}, D_o(\mathbf{y}), D_{g_i}(\mathbf{y}), D_{h_j}(\mathbf{y}), F\}$ if $\min D_o(\mathbf{y}) > \tilde{p}$ and return to step 7.
9. {Subdivision decision}
   If
   $$(\text{wid } \mathbf{b}) \ \& \ (\max D_o(\mathbf{b}) - \min D_o(\mathbf{b})) < \epsilon$$
   then add the item $\{\mathbf{b}, \min\ D_0(\mathbf{b})\}$ to $\mathcal{L}^{sol}$ and go to step 7. Else go to step 10.
10. {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 $\mathbf{b} = \mathbf{b}_1 \cup \mathbf{b}_2$ .
11. For $r = 1, 2$
   (a) {Set flag vector}
       Set $F^r = (F_1^r, \dots, F_p^r, F_{p+1}^r, \dots, F_{p+q}^r) := F$
   (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 $\tilde{p}_{local} = \min(\mathbb{D}_o(\mathbf{b}_r))$.
   (d) If $\tilde{p}_{local} > \tilde{p}$ go to sub step i.
   (e) for $i = 1, \dots, p$ if $F_i = 0$ then
        i. If $\mathbb{D}_{g_i}(\mathbf{b}_r) > 0$ then go to sub step f.
       ii. If $\mathbb{D}_{g_i}(\mathbf{b}_r) \leq 0$ then set $F_i^r = 1$.
   (f) for $j = 1, \dots, q$ if $F_{p+j} = 0$ then
        i. If $0 \notin \mathbb{D}_{h_j}(\mathbf{b}_r)$ then go to sub step i.
       ii. If $\mathbb{D}_{h_j}(\mathbf{b}_r) \subseteq [-\epsilon_{zero}, \epsilon_{zero}]$ then set $F_{p+j}^r = 1$
   (g) If $F^r = (1, \dots, 1)$ then set $\tilde{p} := \min(\tilde{p}, \max(\mathbb{D}_o(\mathbf{b}_r)))$.
   (h) 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 $\mathcal{L}$.
   (i) end (of $r$-loop)
12. {Compute the global minimum}
   Set the global minimum to the current minimum estimate, $\hat{p} = \tilde{p}$.
13. {Compute the global solution}
   Find all those items in $\mathcal{L}^{sol}$ for which $\min D_o(\mathbf{b}) = \widehat{p}$. The first entries of these items are the global minimizer(s) $\mathbf{z}^{(i)}$.
14. Return the global minimum $\hat{p}$ and all the global minimizers $\mathbf{z}^{(i)}$ found above.
   END Algorithm
```