JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 6, ISSUE 07, 2019

An Interval Newton Methods for bounding zeros of Polynomial Systems using B-spline expansion

Deepak Gawali
Systems & Control Engineering Department, Indian Institute Technology, Bombay
ddgawali2002@gmail.com

Abstract:

The purpose of this paper is to study a technique of finding the zeros of a nonlinear polynomial equations. Interval method can be used to obtain rigorous bounds on a roots in a given box. The proposed algorithms for obtaining the roots of the polynomial system is based on the following technique:

1) transformation of the original nonlinear algebraic equations into polynomial B-spline form; 2) includes a pruning step using B-spline Newton operator.

We compare the performance of our proposed B-spline Newton operator with the interval Newton operator using two numerical examples. The results of the tests show the superiority of the proposed algorithm, in terms of selected performance metrics.

Keywords: *Nonlinear polynomial equations systems, Polynomial B-spline form, Interval analysis, Interval Newton operator.*

I. INTRODUCTION

Today, systems of polynomial equations arise in robotics, coding theory, optimization, mathematical biology, computer vision, game theory, statistics, machine learning, control theory and numerous other areas. A system of polynomial equations given by

$$f(x) = 0, \tag{1}$$

where $f = (f_1, f_2, \cdots, f_n)$, and each f_i is a s dimensional polynomial of independent variables $x = (x_1, x_2, \cdots, x_s)$. The zeros of (1) can be obtained by several methods including continuation methods [1] and elimination theory [2], [3].

In [4],[5], the authors proposed several root finding algorithms for the solving systems of nonlinear polynomial equations. In [6],[7],[8],[9] the authors use interval methods for bounding zeros of systems of nonlinear polynomial equations. The approach of interval computation guaranteed an interval that contains all zeros of the system of polynomial equations can be assured using branch and bound strategy. Generally, interval branch and bound methods are time consuming because they requires evaluation of the polynomial functions during each iteration.

Narrowing operators like Hansen-Sengupta, Newton, and Krawczyk can be introduced for pruning the search space. The interval enclosures for these narrowing operators requires evaluation of polynomial function derivatives during each iteration. Finding polynomial function derivatives using interval methods is often time-consuming. Again, in [10],[11] the authors combine Krawczyk contractor and domain subdivision for bounding zeros of systems of nonlinear polynomial equations in B-spline and Bernstein form respectively.

We present an algorithm based on B-spline Newton operator for bounding zeros of systems of polynomial equations. The B-spline coefficient computation algorithm in [12],[13],[14] used for unconstrained optimization problems. The proposed algorithm combine the advantages of the B-spline Newton algorithm, and the B-spline coefficient computation algorithm to find the zeros of system of polynomial equations.

We use B-spline expansion approach to obtain estimate for the range of polynomial in power form. On expanding the polynomial in power form into polynomial B-spline form the minimum and the maximum value of B-spline coefficients provides the bound on the range of polynomial in power form. To obtain tight bounds on the range enclosure we increase number of segments of B-spline as shown in Figure. 1.

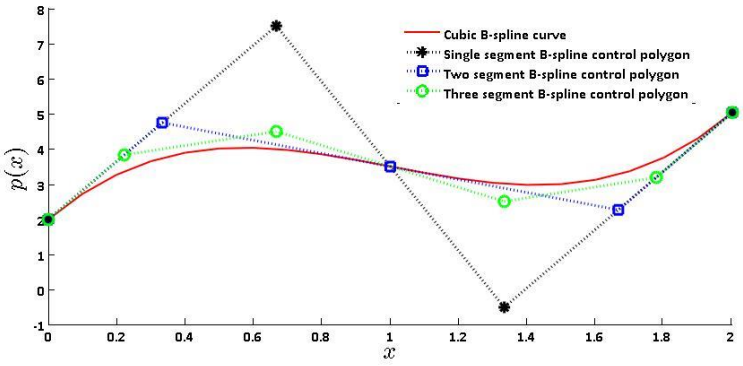

Figure 1: Improvement in the range enclosure of univariate polynomial by increasing the number of segments of B-spline.

The computational complexity of B-spline coefficients computation as given in [14] is $O((m+k)^s m)$. Therefore, to minimize the computation time a B-spline with single segments is a best option for bounding zeros of systems of polynomial equations.

This paper is organized as follows. Section 2, gives an overview of B-spline expansion and domain subdivision approach. Section 3, we present the B-spline Newton operator and propose an algorithm for bounding zeros of polynomial systems. In section 4, describes the results of numerical test. Section 5 concludes the paper.

II. BACKGROUND: POLYNOMIAL B-SPLINE FORM

Firstly, we present brief review of B-spline form, which is used as inclusion function to bound the range of multivariate polynomial in power from. The B-spline form is then used as basis of main zero finding algorithm in section 3.

We follow the procedure given in [7],[6] for B-spline expansion. Let $\varphi(t_1,\cdots t_l)$ be a multivariate polynomial in l real variables with highest degree $(m_1+\cdots m_l)$, (2).

$$\varphi(t_1,\cdots t_l)=\sum_{s_1=0}^{m_1}\cdots\sum_{s_l=0}^{m_l}a_{s_1\cdots s_l}t_1^{s_1}\cdots t_l^{s_l}. \tag{2}$$

2.1 Univariate polynomial

Lets consider univariate polynomial case first, (3)

$$\varphi(t)=\sum_{s=0}^{m}a_s t^s,\ t\in[p,q], \tag{3}$$

for degree d (i.e. order d+1) B-spline expansion where $d\geq m,$ on compact interval I=[p,q]. We use $\psi_d(I,\mathbf{u})$ to represent the space of splines of degree d on the uniform grid partition known as *Periodic* or *Closed* knot vector, $\mathbf{u}$:

$$\mathbf{u}:=\{t_0<t_1<\cdots<t_{k-1}<t_k\}, \tag{4}$$

Where $t_i:=p+iy,\ 0\leq i\leq k,\ k$ denotes B-spline segments and $y:=(q-p)/k.$

Let $\mathbf{P}_d$ reflects the space of degree d splines. We then denote the space of degree d splines with C^{d-1} continuous on $[p,q]$ and defined on $\mathbf{u}$ as

$$\psi_d(I,\mathrm{u}):=\{\psi\in C^{d-1}(I):\psi|[t_i,t_{i+1}]\in \mathrm{P}_d, i=0,\cdots,k-1\}. \tag{5}$$

Since $\psi_d(I,\mathrm{u})$ is $(k+d)$ dimension linear space [8]. Therefore to construct basis of splines supported locally for $\psi_d(I,\mathrm{u}),$ we use few extra knots $t_{-d}\leq\cdots\leq t_{-1}\leq p$ and $q\leq t_{k+1}\leq\cdots\leq t_{k+d}$ at the ends in knot vector. These types of knot vectors are known as *Open* or *Clamped* knot vectors, (6). Since knot vector $\mathbf{u}$ is uniform grid partition, we choose $t_i:=p+iy$ for

$i \in \{-d,\cdots,-1\} \cup \{k+1,\cdots,k+d\}$,

$$\mathbf{u} := \{t_{-d} \le \cdots \le t_{-1} \le p = t_0 < t_1 < \cdots < t_{k-1} < q = t_k \le t_{k+1} \le \cdots \le t_{k+d}\}. \quad (6)$$

The B-spline basis $\{B_i^d(t)\}_{i=1}^{k-1}$ of $\psi_d(I,\mathrm{u})$ is defined in terms of divided differences:

$$B_i^d(t) := (t_{i+d} - t_i)[t_i, t_{i+1}, \cdots, t_{i+d+1}](.-t)_+^d, \quad (7)$$

where $(.)_+^d$ represent the truncated power of degree d. This can be easily proven that

$$B_i^d(t) := \Omega_d\left(\frac{t-a}{h} - i\right), -d \le i \le k-1, \quad (8)$$

where

$$\Omega_d(t) := \frac{1}{d!}\sum_{i=0}^{d+1}(-1)^i \binom{d+1}{l}(t-l)_+^d, \quad (9)$$

$B_i^d(t) := (t_{i+d} - t_i)[t_i, t_{i+1}, \cdots, t_{i+d+1}](.-t)_+^d$, is the polynomial B-spline of the degree d. The B-spline basis can be computed by a recursive relationship that is known as *Cox-deBoor* recursion formula

$$B_i^d(t) := \gamma_{i,d}(t) B_i^{d-1}(t) + (1 - \gamma_{i+1,d}(t)) B_{i+1}^{d-1}(t), d \ge 1, \quad (10)$$

where

$$\gamma_{i,d}(t) = \begin{cases} \dfrac{t-t_i}{t_{i+d}-t_i}, & \text{if } t_i \le t_{i+d}, \\ 0, & \text{otherwise}, \end{cases} \quad (11)$$

and

$$B_i^0(t) := \begin{cases} 1, & \text{if } t \in [t_i, t_{i+1}), \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

The set of spline basis $\{B_i^d(t)\}_{i=1}^{k-1}$ satisfies following interesting properties:

1. Each $B_i^d(t)$ is positive on its support $[t_i, t_{i+d+1}]$.
2. Set of spline basis $\{B_i^d(t)\}_{i=1}^{k-1}$ exhibits a partition of unity, i.e. $\sum_{i=1}^{k-1} B_i^d(t) = 1$.

The power basis functions $\left\{t^r\right\}_{r=0}^{m}$ in power form polynomial (3) can be represented in term of B-spline using following relation

$$t^s := \sum_{v=-d}^{k-1} \pi_v^{(s)} B_v^d(t), s = 0,\cdots,d, \tag{13}$$

and the symmetric polynomial $\pi_v^{(s)}$ defined as

$$\pi_v^{(s)} := \frac{\mathrm{Sym}_s\left(v+1,\cdots,v+d\right)}{k^s \binom{d}{s}},\ s = 0,\cdots,d. \tag{14}$$

Then by substituting (13) in (3) we get B-spline extension of power form polynomial (3) as follows:

$$\varphi(t) := \sum_{s=0}^{m} a_s \sum_{v=-d}^{k-1} \pi_v^{(s)} B_v^d(t) = \sum_{v=-d}^{k-1} \left[\sum_{s=0}^{m} a_s \pi_v^{(s)}\right] B_v^d(t) = \sum_{v=-d}^{k-1} d_v B_v^d(t), \tag{15}$$

where

$$d_v := \sum_{s=0}^{m} a_s \pi_v^{(s)}. \tag{16}$$

2.2 Multivariate polynomial case

Lets consider next multivariate power form polynomial (17) for B-spline expansion

$$\varphi(t_1,\cdots t_l) := \sum_{s_1=0}^{k_1} \cdots \sum_{s_l=0}^{k_l} a_{s_1\cdots s_l} t_1^{s_1} \cdots t_l^{s_l} = \sum_{\mathbf{s}\le\mathbf{k}} a_{\mathbf{s}} t^{\mathbf{k}}, \tag{17}$$

where $\mathbf{s} := (s_1,\cdots,s_l)$ and $\mathbf{k} := (k_1,\cdots,k_l)$. By substituting (13) for each t^s, (17) can be written as

$$\begin{aligned}
\varphi(t_1,t_2,...,t_l) &= \sum_{s_1=0}^{m_1} ... \sum_{s_s=0}^{m_l} a_{s_1\ldots s_l} \sum_{v_1=-d_1}^{k_1-1} \pi_{v_1}^{(s_1)} B_{v_1}^{d_1}(t_1) ... \sum_{v_l=-d_l}^{k_l-1} \pi_{v_l}^{(s_l)} B_{v_l}^{d_l}(t_l), \\
&= \sum_{v_1=-d_1}^{k_1-1} ... \sum_{v_l=-d_l}^{k_s-1} \left(\sum_{s_1=0}^{m_1} ... \sum_{s_l=0}^{m_l} a_{s_1\ldots s_l} \pi_{v_1}^{(s_1)} ... \pi_{v_l}^{(s_l)} \right) B_{v_1}^{d_1}(t_1) ... B_{v_l}^{d_l}(t_l), \\
&= \sum_{v_1=-d_1}^{k_1-1} ... \sum_{v_l=-d_l}^{k_l-1} d_{v_1\ldots v_l} B_{v_1}^{d_1}(t_1) ... B_{v_l}^{d_l}(t_l),
\end{aligned} \tag{18}$$

we can write (18) as

$$\varphi(t) := \sum_{\mathrm{v} \le \mathrm{k}} d_{\mathrm{v}} B_{\mathrm{v}}^{\mathrm{k}}(t). \tag{19}$$

where $\mathrm{v} := (v_1, \cdots, v_l)$ and d_{v} is B-spline coefficient given as

$$d_{v_1 \ldots v_l} = \sum_{s_1=0}^{m_1} \ldots \sum_{s_l=0}^{m_l} a_{s_1 \ldots s_l} \pi_{v_1}^{(s_1)} \ldots \pi_{v_l}^{(s_l)}. \tag{20}$$

The B-spline expansion of (17) is given by (18). The derivative of polynomial can be found in a particular direction using the values of d_{v} i.e. B-spline coefficients of original polynomial for $\mathbf{y} \subseteq I$, the derivative of a polynomial $\varphi(t)$ with respect to t_r in polynomial B-spline form is (21),

$$\varphi_r'(\mathbf{y}) = \frac{m_r}{\mathbf{u}_{\mathbf{s}+m_r+1} - \mathbf{u}_{\mathbf{s}+1}} \times \sum_{I \le \mathbf{m}_{r,-1}} \left[d_{\mathbf{s}_{r,1}}(\mathbf{y}) - d_{\mathbf{s}}(\mathbf{y}) \right] B_{\mathbf{m}_{r,-1},\mathbf{s}}(t), 1 \le r \le l, t \in \mathbf{y}, \tag{21}$$

where $\mathbf{u}$ is a knot vector. The partial derivative $\varphi_r'(\mathbf{y})$ now includes range enclosure for derivative of φ on $\mathbf{y}$. Lin and Rokne proposed (14) for symmetric polynomial and used closed or periodic knot vector (4). Due to change in knot vector from (4) to (6) we propose new form of (14) as follows,

$$\pi_v^{(s)} := \frac{\mathrm{Sym}_s(v+1, \cdots, v+d)}{\binom{d}{s}}. \tag{22}$$

2.3 B-spline range enclosure property

$$\varphi(t) := \sum_{i=1}^{m} d_i B_i^d(t), t \in \mathbf{y}. \tag{23}$$

Let (23) be a B-spline expansion of polynomial $q(t)$ in power form and $\bar{q}(\mathbf{y})$ denotes the range of the power form polynomial on subbox $\mathbf{y}$. The B-spline coefficients are collected in an array $D(\mathbf{y}) := (d_i(\mathbf{y}))_{i \in \Re}$ where $\Re := \{1, \cdots, m\}$. Then for $D(\mathbf{y})$ it holds

$$\bar{q}(\mathbf{y}) \subseteq D(\mathbf{y}) = [\min D(\mathbf{y}), \max D(\mathbf{y})]. \tag{24}$$

The range of the minimum and the maximum value of B-spline coefficients of

multivariate polynomial B-spline expansion provides an range enclosure of the multivariate polynomial q on $\mathbf{y}$.

2.4 Subdivision procedure

We can improve the range enclosure obtained by B-spline expansion using subdivision of subbox $\mathbf{y}$. Let

$$\mathbf{y} := \left[\underline{\mathbf{y}}_1, \overline{\mathbf{y}}_1\right] \times \cdots \times \left[\underline{\mathbf{y}}_r, \overline{\mathbf{y}}_r\right] \times \cdots \times \left[\underline{\mathbf{y}}_l, \overline{\mathbf{y}}_l\right],$$

represent the box to be subdivided in the r th direction $(1 \le r \le l)$. Then two subboxes $\mathbf{y}_\mathbf{A}$ and $\mathbf{y}_\mathbf{B}$ are generated as follows

$$\mathbf{y}_\mathbf{A} := \left[\underline{\mathbf{y}}_1, \overline{\mathbf{y}}_1\right] \times \cdots \times \left[\underline{\mathbf{y}}_r, m(\mathbf{y}_r)\right] \times \cdots \times \left[\underline{\mathbf{y}}_l, \overline{\mathbf{y}}_l\right],$$
$$\mathbf{y}_\mathbf{B} := \left[\underline{\mathbf{y}}_1, \overline{\mathbf{y}}_1\right] \times \cdots \times \left[m(\mathbf{y}_r), \overline{\mathbf{y}}_r\right] \times \cdots \times \left[\underline{\mathbf{y}}_l, \overline{\mathbf{y}}_l\right],$$

where $m(\mathbf{y}_r)$ is a midpoint of $[\underline{\mathbf{y}}_r, \overline{\mathbf{y}}_r]$.

III. B-SPLINE NEWTON OPERATOR

In general problem of computing all zeros of a system of nonlinear polynomial equations within some finite domain can be formulated based on the computation of the range of nonlinear functions over some interval. In order to prune the search space for solution, some form of interval contractors such as Hansen-Sengupta, interval Newton, Krawczyk, etc. need to be used to contract the search bounds. The interval Newton operator is given in [19] as

$$\mathbf{N}\left(\mathbf{p}, \mathbf{y}, \check{y}\right) = \check{y} - \frac{p(\check{y})}{\mathbf{p}'(\mathbf{y})}. \tag{25}$$

Let $p : \mathbf{y} = [\underline{y}, \overline{y}] \to \square$ be a continuously differentiable multivariate polynomial on $\mathbf{y}$, let that there exists $y^* \in \mathbf{y}$ such that $p\left(y^*\right) = 0,$ and suppose that $\check{y} \in \mathbf{y}$. Then, since the mean value theorem implies

$$0 = p\left(y^*\right) = p\left(\check{y}\right) + p'\left(\xi\right)\left(y^* - \check{y}\right),$$

therefore $y^* = \check{y} - \dfrac{p\left(\check{y}\right)}{p'(\xi)}$ for some $\xi \in \mathbf{y}$. If $\mathbf{p}'(\mathbf{y})$ is any interval extension of the derivative of p over $\mathbf{y}$, then

$$y^* \in \check{y} - \frac{p\left(\check{y}\right)}{\mathbf{p}'(\mathbf{y})}, \quad \check{y} \in \mathbf{y}. \tag{26}$$

Because of (26), any solution of $p(y) = 0$ that are in $\mathbf{y}$ must also be in $\mathrm{N} = \left(\mathbf{p}, \mathbf{y}, \check{y}\right)$ and therefore (26) is the basis of the *univariate* Newton method (25).

The *univariate* Newton method (25) can be extended as a *Multivariate* Newton method which execute an iteration equation similar to equation (25).

Suppose now that $y \in \square^s$ and $f(y) \in \square^n$ (continuously differentiable nonlinear) polynomial equations in s unknowns, and let that $\check{y} \in \square^S$. Then a basic formula for multivariate Newton method is

$$\mathbf{N}\left(f, \mathbf{y}, \check{y}\right) = \check{y} + \mathbf{w}, \tag{27}$$

where $\mathbf{w}$ is a vector of interval bounding all zeros w of system $Aw = -f\left(\check{y}\right)$, as $A \in \mathbf{f}'(\mathbf{y})$, such that $\mathbf{f}'(\mathbf{y})$ is the Jacobi matrix f interval extension over $\mathbf{y}$. Therefore obtaining the interval vector $\mathbf{w}$ bounding the solution set to the interval linear system in (27) is an important step in multivariate interval method,

$$\mathbf{f}'(\mathbf{y})\mathbf{w} = -f\left(\check{y}\right).$$

From (25) and (27), the interval vector $\mathbf{w}$ is given by

$$-\frac{f\left(\check{y}\right)}{\mathbf{f}'(\mathbf{y})} = \mathbf{w}.$$

Thus the interval linear system form of multivariate Newton method is given as

$$\mathbf{f}'(\mathbf{y})\times\mathbf{w} = -f\left(\overset{\vee}{y}\right), \tag{28}$$

It is necessary to precondition the system (28) by a point matrix $Y \in \square^{n\times n}$ given by the inverse of the midpoint matrix of an interval extension of the Jacobi matrix $\mathbf{f}'(\mathbf{y})$, i.e. $Y = \left\{\text{mid}\,\mathbf{f}'(\mathbf{y})\right\}^{-1}$.

$$A\times \mathrm{w} = B, \tag{29}$$

where $A = Y\times\mathbf{f}'(\mathbf{y})$ and $B = -Y\times f\left(\overset{\vee}{y}\right)$.

Then to compute sharper bounds on $\mathbf{w}$ the interval Gauss-Seidel method [5] or the interval hull method [20] can be used to solve the system (29). The components of $\mathbf{N}\left(\mathbf{f},\mathbf{y},\overset{\vee}{y}\right)$ is given by (27). Then the intersection $\mathbf{y}\cap\mathbf{N}\left(\mathbf{f},\mathbf{y},\overset{\vee}{y}\right)$ results in contracted domain of $\mathbf{y}$.

Interval Newton method requires repeated evaluation of the (polynomial) function at $\overset{\vee}{y}\in\mathbf{y}$ to compute $f(\overset{\vee}{y})$, which can be time consuming operation. Moreover, interval computations are used for finding $\mathbf{f}'(\mathbf{y})$ to compute the precondition matrix Y, which apart from time consuming , often give quite pessimistic results.

The B-spline Newton method can alleviate some of these difficulties. In this method, it is quite simple and straightforward to compute $f(\overset{\vee}{y})$, if we choose $\overset{\vee}{y}$ to be any vertex point of $\mathbf{y}$, then $f(\overset{\vee}{y})$ is given directly by the B-spline coefficient value at $\overset{\vee}{y}$. This obviates the need to evaluate the system of polynomial at $\overset{\vee}{y}$ as done in the interval Newton method. In proposed method the B-spline coefficients of the first partial derivatives are simply obtained as the differences of coefficients of the original polynomial f (21).

We now present the algorithm for bounding zeros of polynomial systems similar to [21],

---

Algorithm 3.1: Algorithm for bounding zeros of polynomial systems

---

Input : Here A_c is a cell structure containing the coefficients array a_I of the polynomials in the power form. N_c is a cell structure, containing degree vector, N_I which contains degree of each variable in polynomial function. Initial bound $\mathbf{x}$ of each

variable and tolerance limit δ.

Output : The zero(s) of f in $\mathbf{x}$ or $\{\varnothing\}$ as no solution exists in $\mathbf{x}$.

Begin Algorithm

1 {Compute the B-spline coefficients}

Compute the B-spline coefficients $D_i(\mathbf{x})$ of given n polynomials on the initial box $\mathbf{x}$, where $i = 1, 2, \cdots, n.$ (Use algorithms given in [12])

2 {Initialize iteration number}

Set $k = 0,\ \mathbf{x}^{(0)} = \mathbf{x}.$

3 {Compute $f(\check{x})$ }

Choose $\check{x} = \text{mid}(\mathbf{x}^{(k)})$ and obtain the value of $f(\check{x})$ directly from the B-spline coefficient value at the vertex of $\text{mid}(\mathbf{x}^{(k)})$.

4 {Compute $\mathbf{f}'(\mathbf{x})$ }

Use the B-spline coefficients of f on $\mathbf{x}^{(k)}$, to compute the B-spline coefficients of all the first partial derivatives of f on $\mathbf{x}^{(k)}$ via (21). From the minimum and maximum B-spline coefficients of the first derivative, construct their range enclosure interval, and form the interval Jacobian matrix $\mathbf{f}'(\mathbf{x})$.

5 {Compute the precondition matrix Y }

Compute the preconditioning matrix Y as

$$Y = \left\{\text{mid}\,\mathbf{f}'\left(\mathbf{x}^k\right)\right\}^{-1}.$$

6 {Solve linear interval system and update solution}

Solve the linear interval system

$$Y \times \mathbf{f}'(\mathbf{x}) \times \mathbf{v} = -Y \times f\left(\check{x}\right),$$

	and obtain $\mathbf{N}\left(\mathbf{f}, \mathbf{x}^{(k)}, \overset{\vee}{x}\right)$ to update the solution as $\mathbf{x}^{(k)} = \mathbf{x}^{(k)} \cap \mathbf{N}\left(\mathbf{f}, \mathbf{x}^{(k)}, \overset{\vee}{x}\right).$
7	{Return $\{\varnothing\}$ } If $\mathbf{x}^{(k)} = 0,$ then return $\{\varnothing\}$ as solution and *exit* algorithm.
8	{Termination} If $\mathbf{x}^{(k)} < \grave{o},$ then return $\mathbf{x}^{(k+1)}$ as solution and *exit* algorithm.
9	Set $k = k+1$ and **go to step 3.**

End Algorithm

IV. NUMERICAL RESULTS

We consider the two problems from [22] to test and compare the performance of B-spline Newton operator (BNO) over the interval Newton operator (INO). An tolerance limit of $\grave{o} = 10^{-06}$ is prescribed for computing the set of roots in each test problem. The number of iterations and computational time (in seconds) are taken as the performance metrics. Our MATLAB source code implementation of interval Newton operator using INTLAB [23] solver is made available at [bit.ly/2UMNl7e] for all two test problems

We proposed a constrained global optimization algorithm to solve the problem of domain of attraction in control system using polynomial B-spline form as an inclusion function to bound the range of nonlinear multivariate polynomial function. The algorithm does not need any linearization or relaxation techniques and solves the problem to specified accuracy.

Example 1: This example is taken from [22]. This is a problem with 4 variables. The polynomial systems is given by

$$\begin{aligned}
&x_1 + x_2 + x_3 + x_4 + 1 = 0,\\
&x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4 = 0,\\
&x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4 + x_1x_4 = 0,\\
&x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4 + x_3x_4x_1 + x_1x_2x_4 = 0.
\end{aligned}$$

and the bounds on the variables are

$$x_1 = [0.95, 1.05],\ x_2 = [0.95, 1.05],\ x_3 = [-2.65, -2.6],\ x_4 = [-0.4, -0.37].$$

From Table 2 we observe that the existing interval Newton operator require 4 iterations to bound the roots of the polynomial systems with the accuracy of $\grave{o} = 10^{-06}$. The proposed B-spline Newton operator algorithm computes the result in 5 iterations within the same accuracy. The computational time required for the proposed B-spline Newton operator is 1.7108 seconds, whereas the interval Newton operator method requires computational time 2.235 seconds. The results of algorithm are tabulated in Table 1.

Table 1: Roots of Example 1.

Roots	
x_1	1
x_2	1
x_3	-2.6180
x_4	-0.3819

Table 2: Comparison of performance between BNO and INO.

	Number of Iterations	**Computation Time (Sec.)**
BNO	5	1.71
INO	4	2.23

Example 2: This example is taken from [22]. This is a problem with 5 variables. The polynomial systems is given by

$$\begin{aligned}
&x_1 + x_2 + x_3 + x_4 + x_5 + 1 = 0,\\
&x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5 = 0,\\
&x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5 + x_1x_5 = 0,\\
&x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5 + x_1x_4x_5 + x_1x_2x_5 = 0.\\
&x_1x_2x_3x_4 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_1x_2x_4x_5 + x_1x_2x_3x_5 = 0.
\end{aligned}$$

and the bounds on the variables are

$$x_1 = [0.95, 1.05],\ x_2 = [-3.75, -3.70],\ x_3 = [-0.28, -0.25],\ x_4 = [0.95, 1.01],\ x_5 = [0.95, 1.01].$$

From Table 3 we observe that the existing interval Newton operator require 4 iterations to bound the roots of the polynomial systems with the accuracy of $\grave{o}=10^{-06}$. The proposed B-spline Newton operator algorithm computes the result in 7 iterations within the same accuracy. The computational time required for the proposed B-spline Newton operator is 1.23 seconds, whereas the interval Newton operator method requires computational time 1.44 seconds. The results of algorithm are tabulated in Table 3.

Table 3: Roots of Example 2.

Roots	
x_1	1
x_2	-3.7320
x_3	-0.2679
x_4	1
x_5	1

Table 4: Comparison of performance between BNO and INO.

	Number of Iterations	**Computation Time (Sec.)**
BNO	7	1.23
INO	4	1.44

V. CONCLUSION

In this paper we presented an algorithm for contracting the search domain using B-spline Newton operator. The computational examples demonstrate that the algorithm suggested quite effectively solves the polynomial system in terms of computational time because B-spline approach avoids evaluation of the polynomial functions but requires more number of iterations due to over estimation in range enclosure of the first partial derivatives of the original polynomial.

REFERENCES

[1] Morgan A. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. SIAM; 2009.

[2] Kolev L V. An improved method for global solution of non-linear systems. *Reliab Comput*. 1999;5(2):103-111.
[3] Kolev L. An interval method for global nonlinear analysis. *IEEE Trans Circuits Syst I Fundam Theory Appl*. 2000;47(5):675-683.
[4] Jäger C, Ratz D, iyegyer K, Rats L. A combined method for enclosing all solutions of nonlinear systems of polynomial equations. *Reliab Comput*. 1995;1(1):41-64. doi:10.1007/BF02390521
[5] Kearfott RB. *Rigorous Global Search: Continuous Problems.* Vol 13. Springer Science & Business Media, Berlin; 2013.
[6] Hammer R, Hocks M, Kulisch U, Ratz D. *Numerical Toolbox for Verified Computing I: Basic Numerical Problems Theory, Algorithms, and Pascal-XSC Programs*. Vol 21. Springer Science & Business Media; 2012.
[7] Hansen E, Walster G. *Global Optimization Using Interval Analysis: Revised and Expanded*. Vol 264. (2, ed.). Marcel Dekker, New York; 2004.
[8] Moore RE. *Methods and Applications of Interval Analysis.* SIAM, U.S.A.; 1979.
[9] Nataraj PS V, Sondur S. Construction of bode envelopes using REP based range finding algorithms. *Int J Autom Comput*. 2011;8(1):112-121.
[10] Arounassalame M. Analysis of Nonlinear Electrical Circuits Using Bernstein Polynomials. *Int J Autom Comput*. 2012;9(1):81-86.
[11] Michel D, Zidna A. Interval-Krawczyk Approach for Solving Nonlinear Equations Systems in B-spline Form. In: *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer; 2015:455-465.
[12] Gawali DD, Zidna A, Nataraj PSV. Algorithms for unconstrained global optimization of nonlinear (polynomial) programming problems: The single and multi-segment polynomial B-spline approach. *Comput Oper Res*. 2017;87. doi:10.1016/j.cor.2017.02.013
[13] Gawali D., Zidna A., Nataraj P.S.V.. *Solving Nonconvex Optimization Problems in Systems and Control: A Polynomial B-Spline Approach*. Vol 359.; 2015. doi:10.1007/978-3-319-18161-5_40
[14] Gawali D. D., Patil B. V., Zidna A, Nataraj P. S. V.. *A B-Spline Global Optimization Algorithm for Optimal Power Flow Problem*. Vol 991.; 2020. doi:10.1007/978-3-030-21803-4_6
[15] Lin Q, Rokne JG. Methods for bounding the range of a polynomial. *J Comput Appl Math*. 1995;58:193-199.
[16] Lin Q, Rokne JG. Interval approximation of higher order to the ranges of functions. *Comput Math with Appl*. 1996;31(7):101-109.
[17] DeVore RA, Lorentz GG. *Constructive Approximation*. Vol 303. Springer Science & Business Media, Berlin; 1993.
[18] Kearfott RB. Encyclopedia of Optimization. In: Springer US; 2009:1763-1766.
[19] Neumaier A. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. *Reliab Comput*. 1999;5(2):131-136.
[20] Nataraj PSV, Arounassalame M. An interval Newton method based on the Bernstein form for bounding the zeros of polynomial systems. *Reliab Comput*. 2011;15(2):185-212.
[21] Verschelde J. The PHC Pack, the Database of Polynomial Systems. Published online 2001.
[22] Rump SM. INTLAB-interval laboratory. In: *Developments in Reliable Computing*. Springer; 1999:77-104.