An Interval Newton Method Based on the Bernstein Form for Bounding the Zeros of Polynomial Systems

P. S. V. Nataraj
Systems and Control Engineering, CRNTS Building, Indian Institute of Technology Bombay, Mumbai-400 076, India
nataraj@sc.iitb.ac.in

M. Arounassalame
Systems and Control Engineering, CRNTS Building, Indian Institute of Technology Bombay, Mumbai-400 076, India*
aroun@sc.iitb.ac.in

Abstract

Interval Newton methods are widely used to find reliable enclosures for the roots of the polynomial systems. But the computation process needs the evaluation of an enclosure for the derivative of the polynomial and evaluation of the polynomial at a particular point. Again, the subdivisions (if any are needed) will require repeated evaluation of the function values. We propose an alternative approach using Bernstein coefficients to find the value of the Newton operator. This Bernstein Newton operator doesn't require evaluation of the function at any particular point. Instead the function value is obtained directly from the vertex points. Again the computation of the derivative is very simple using the Bernstein coefficient approach. In addition to this the range enclosures obtained using Bernstein coefficients are much sharper than the range enclosures obtained using many other interval forms. We compare the performance of the proposed Bernstein Newton Contractor with that of Interval Newton contractor using numerical examples, proving the superiority of the proposed approach.

Keywords: Bernstein Newton contractor, Bernstein polynomials, Interval Newton, subdivision
AMS subject classifications: 65H10, 65G20, 65G30, 34A34

*Submitted: January 20, 2009; Accepted: January 16, 2010.

1 Introduction

Engineering applications such as robotics, global optimization, chemical process, etc. require finding all the isolated solutions to polynomial systems. Polynomials are popular in curve and surface representations and in many critical problems arising in computer aided geometric design. In general these problems are reduced to finding zeros of a polynomial system of equations

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

where $f = (f_1, f_2, ...f_n)$, and each f_i is a polynomial of l independent variables $\mathbf{x} = (x_1, x_2, ..., x_l)$. Several root-finding algorithms are proposed in literature [7, 10] to find the solutions to system of polynomial equations. The solutions can be obtained either by directly solving the set of nonlinear polynomial equations or by solving the set of simpler functions obtained by decomposing the nonlinear polynomial equations. The nonlinear polynomial systems of equations can be directly solved using interval analysis [6, 14, 5]. In general, guaranteed interval enclosures to all the zeros of the polynomial systems can be obtained using interval branch and bound methods. Interval methods often require repeated evaluation of the polynomial functions, which is a time consuming operation. Pruning operators such as interval Newton can be introduced to reduce the number of iterations. But the evaluation of interval enclosures for these operators requires derivatives. Finding derivatives of polynomial systems using interval methods is also a time consuming process. In this paper we propose an algorithm based on the Bernstein form combined with the Newton operator to find the enclosures for the roots of systems of polynomial equations.

Any multivariate power form of a polynomial can be represented equivalently by the Bernstein form of the polynomial [1]. The coefficients of the Bernstein polynomial are called Bernstein coefficients. The minimum and maximum Bernstein coefficients enclose the range of the polynomial. In general the range enclosure obtained by Bernstein form is much sharper than the range enclosures obtained using other interval forms [22, 23]. The subdivision of the intervals and the evaluation of the values of the polynomial systems over the subdivided intervals are avoided by using the Bernstein coefficient approach. The Bernstein coefficients of the subdivided intervals can be derived directly from the Bernstein coefficients of the original interval.

A basic Bernstein form of Branch and Bound (BBB) type of algorithm was already used in [11, 12]. This type of algorithm can be used for the computation of interval enclosures for the solution of polynomial systems. But this method requires more subdivisions, making the algorithm inefficient in terms of computation time and number of iterations. The algorithm can be made more efficient by introduction of pruning steps using the proposed Bernstein Newton operator. Since we are proposing a method based on Bernstein approach to evaluate the value of Newton operator, we can avoid the complications in the interval derivative, as the derivative in Bernstein domain is much simpler and straightforward. This algorithm is more efficient than the BBB algorithm as it reduces the number of iterations considerably. We further modified this algorithm to avoid the evaluation of Bernstein coefficients after the pruning step. We use Bernstein coefficient contraction to replace this evaluation step, so as to avoid this slow computation procedure.

The remaining sections in this paper are organized as follows. In section 2 we give a brief introduction to the Bernstein expansion of multivariate power form polynomials and the subdivision procedure. In section 3, we explain the algorithm for the proposed Bernstein Newton approach. In section 4, we propose the Bernstein coefficient

contraction procedure which avoids the reevaluation of Bernstein coefficients over the new domain. In section 5, we compare the performance of the proposed and existing methods on a few numerical examples. Finally, conclusions of the work are given in section 6.

2 Background

2.1 The Bernstein form

Following the notations given in [2, 15, 17], let $l \in \mathbb{N}$ be the number of variables and $x = (x_1, x_2, ..., x_l) \in \mathbb{R}^l$. A multi-index I is defined as $I = (i_1, i_2, ..., i_l) \in \mathbb{N}^l$ and multi-power x^I is defined as $x^I = (x_1^{i1}, x_2^{i2}, ..., x_l^{il})$. A multi-index N is defined as $N = (n_1, n_2, ..., n_l)$. Inequalities $I \leq N$ for multi-indices are meant component-wise, where $0 \leq i_k$, $k = 1, 2, ..., l$. With $I = (i_1, ..., i_{r-1}, i_r, i_{r+1}, ..., i_l)$ we associate the index $I_{r,k}$ given by $I_{r,k} = (i_1, ..., i_{r-1}, i_r + k, i_{r+1}, ..., i_l)$, where $0 \leq i_r + k \leq n_r$. Also we write (N/I) for $(n_1/i_1, n_2/i_2, ..., n_l/i_l)$. Let $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ be a real interval, where $\underline{\mathbf{x}}$=inf $\mathbf{x}$ and $\overline{\mathbf{x}}$=sup $\mathbf{x}$. The width of the interval $\mathbf{x}$ is represented as wid $\mathbf{x} = \overline{\mathbf{x}} - \underline{\mathbf{x}}$

We can write an l-variate polynomial p in the form

$$p(x) = \sum_{I \leq N} a_I x^I, x \in \mathbb{R}^l, \tag{2}$$

with N as the degree of p. We can expand a given multivariate polynomial into a Bernstein polynomial to obtain bounds for its range over an l-dimensional box $\mathbf{x}$=($\mathbf{x}_1\mathbf{x}_2...\mathbf{x}_l$). Without loss of generality, we consider the unit box $\mathbf{u} = [0, 1]^l$ since any nonempty box $\mathbf{x}$ of $\mathbb{R}^l$ can be mapped affinely onto this box. The I^{th} Bernstein polynomial of degree N is defined as

$$B_I^N(x) = B_{i_1}^{n_1}(x_1)...B_{i_l}^{n_l}(x_l), x \in \mathbb{R}^l, \tag{3}$$

where for $i_j = 0, 1, ..., n_j$, $j = 1, 2, ..l$

$$B_{i_j}^{n_j}(x_j) = \begin{pmatrix} n_j \\ i_j \end{pmatrix} x_j^{i_j}(1 - x_j)^{n_j - i_j} \tag{4}$$

The Bernstein coefficients $b_I(\mathbf{u})$ of p over the unit box $\mathbf{u}$ are given by

$$b_I(\mathbf{u}) = \sum_{J \leq I} \frac{\begin{pmatrix} I \\ J \end{pmatrix}}{\begin{pmatrix} N \\ J \end{pmatrix}} a_J, I \leq N. \tag{5}$$

Thus the Bernstein form of a multivariate polynomial p is defined by

$$p(x) = \sum_{I \leq N} b_I(\mathbf{u}) B_I^N(x) \tag{6}$$

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

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

Lemma 1 [16, 17] *Let p be a polynomial of degree N, and let $\overline{p}(x)$ denote the range of p on the given domain $\boldsymbol{x}$. Then, the following property holds for a patch $B(\boldsymbol{u})$ of Bernstein coefficients: $\overline{p}(x) \subseteq [minB(\boldsymbol{u}), maxB(\boldsymbol{u})]$.*

The range enclosure of the multivariate polynomial p on the domain $\mathbf{x}$ can be found by transforming the polynomial from power form to Bernstein form. Then, by Lemma 1, the coefficients of the expansion in the Bernstein form provide lower and upper bounds for the range.

The following lemma describes the vertex property of the Bernstein coefficients.

Lemma 2 (Vertex property lemma) [3] *The upper bound or lower bound is sharp if and only if $min(b_I(\boldsymbol{u}))_{I\in S}$ (resp., $max(b_I(\boldsymbol{u}))_{I\in S}$) is attained at the indices of the vertices of the array $B(\boldsymbol{u})$. This condition is known as the vertex property.*

Remark 1 *The value of the Bernstein coefficients present at the vertex points represent the true value of the function at these points.*

The partial derivative of a polynomial in a particular direction can be found using the Bernstein coefficients of the original polynomial using the following relation [21, 25]. On a box $\mathbf{d} \subseteq \mathbf{x}$, the first partial derivative with respect to x_r of a polynomial $p(x)$ in Bernstein form is

$$p'_r(\mathbf{d}) = \frac{n_r}{\text{wid } \mathbf{d}} \sum_{I \leq N_{r,-1}} [b_{I_{r,1}}(\mathbf{d}) - b_I(\mathbf{d})] B_{N_{r,-1},I}(x),\ 1 \leq r \leq l, \quad x \in \mathbf{d} \tag{7}$$

Now, $p'_r(\mathbf{d})$ contains an enclosure of the range of the partial derivative of p on $\mathbf{d}$. Similar formulae exist for the higher partial derivatives.

The Bernstein coefficients of a multivariate polynomial p over a box can be efficiently computed using the *matrix* method of Ray [21]. Ray uses a two-dimensional array or matrix representation for computing the coefficients of a $n-$dimensional polynomial on a general box. We use this method for the computation of Bernstein coefficients in the sequel.

2.2 Subdivision procedure

The range enclosure obtained using Bernstein coefficients can be further improved either by degree elevation of the Bernstein polynomial or by subdivision. The subdivision strategy is generally more efficient than the degree elevation strategy [2, 25] and is therefore preferred. A subdivision in the r^{th} direction ($1 \leq r \leq l$) is a bisection perpendicular to this direction. Let

$$\mathbf{x} = [\underline{x_1}, \overline{x_1}] \times ... \times [\underline{x_r}, \overline{x_r}] \times ... \times [\underline{x_l}, \overline{x_l}]$$

be any subbox and suppose the patch $B(\mathbf{x})$ has already been computed. Further suppose that $\mathbf{x}$ is bisected along the r^{th} component direction. Then, two subboxes $\mathbf{x}_A$ and $\mathbf{x}_B$ are generated by this bisection or subdivision as

$$\begin{aligned} \mathbf{x}_A &= [\underline{x_1}, \overline{x_1}] \times ... \times [\underline{x_r}, m(\mathbf{x}_r)] \times ... \times [\underline{x_l}, \overline{x_l}], \\ \mathbf{x}_B &= [\underline{x_1}, \overline{x_1}] \times ... \times [m(\mathbf{x}_r), \overline{x_r}] \times ... \times [\underline{x_l}, \overline{x_l}] \end{aligned}$$

where, $m(\mathbf{x}_r)$ denotes the midpoint of $[\underline{x_r}, \overline{x_r}]$. Starting with $B^0(\mathbf{x}) = B(\mathbf{x})$ we set for $k = 1, 2, ..n_r$,

$$b_I^{(k)}(\mathbf{x}) = \left\{ \begin{array}{c} b_I^{(k-1)}(\mathbf{x}) : i_r \leq k \\ (1-\lambda) b_{I_{r,-1}}^{(k-1)}(\mathbf{x}) + \lambda b_I^{(k-1)}(\mathbf{x}) : k \leq i_r \end{array} \right\} \tag{8}$$

where λ is the subdivision parameter. To obtain the new coefficients, the above formula is applied for $i_j = 0, ..., n_j, j = 1, ..., r-1, r+1, ..., l$. Then

$$B(\mathbf{x}_A) = B^{n_r}(\mathbf{x}) \tag{9}$$

The Bernstein coefficients $B(\mathbf{x}_B)$ on the neighboring subbox $\mathbf{x}_B$ are the intermediate values of this computation since for $k = 0, 1, ..., n_r$ the following relation holds [25]

$$b_{i_1,...,n_r-k,...,i_l}(\mathbf{x}_B) = b_{i_1,...,n_r,...,i_l}(\mathbf{x}_A) \tag{10}$$

By the above subdivision procedure, the explicit transformation of the subboxes generated by the subdivisions back to original box is avoided.

3 Bernstein Newton operator

The proposed Bernstein Newton method is based on the following:

- If we choose $x^{(k)}$ to be any vertex point of $\mathbf{x}^{(k)}$, then by Remark 1, $f\left(x^{(k)}\right)$ is given directly by the Bernstein coefficient value at $x^{(k)}$. This obviates the need to evaluate the system of polynomials at $x^{(k)}$ as done in the interval Newton method.
- From the Bernstein coefficients of f on $\mathbf{x}^{(k)}$, we can compute the Bernstein coefficients of each first partial derivative by just finding their difference, cf. (7). The range enclosure of any partial derivative is simply the minimum to maximum over the respective set of Bernstein coefficients. Using these range enclosures, we form the interval Jacobian matrix $\mathbf{J}\left(x^{(k)}, \mathbf{x}^{(k)}\right)$. Thus, there is no need for automatic differentiation or interval computations to find the interval partial derivatives, as done presently in the interval Newton method.
- At each iteration, the resulting linear interval system can be solved using the interval Gauss-Seidel method [9] or the interval hull method [18], just as done presently in the interval Newton method.
- The Bernstein coefficients on the contracted box $\mathbf{x}^{(k+1)}$ are needed in the next iteration for further work. Since straightforward evaluation of the Bernstein coefficients on $\mathbf{x}^{(k+1)}$ can be time consuming, we instead use the Bernstein coefficient contraction (BCC) method in section 4 to obtain the Bernstein coefficients for the contracted box. In this procedure, the coefficients for $\mathbf{x}^{(k+1)}$ are derived from those already available for $\mathbf{x}^{(k)}$ (but are not freshly computed).

We now present the algorithm.

Algorithm: The Bernstein Newton method

Inputs: Degree N of the variables $x = (x_1, x_2, ..., x_l)$ occurring in the polynomials $f = (f_1, f_2, ..., f_l)$, the coefficients a_I of these polynomials in the power form, the $l-$dimensional initial domain box $\mathbf{x}$, and the tolerance parameter ϵ.

Outputs: Either the message 'no solution exists in $\mathbf{x}$', or the zero(s) of $f = (f_1, f_2, ..., f_l)$ in $\mathbf{x}$.

BEGIN Algorithm

1. {Compute the Bernstein coefficients} Using the *matrix* method of Ray [21], compute the Bernstein coefficients $B(\mathbf{x})$ of f on the initial box $\mathbf{x}$.
2. {Initialize iteration number} Set $k = 0$, $\mathbf{x}^{(0)} = \mathbf{x}$.
3. {Compute f at a vertex point} Choose $x^{(k)}$ to be any vertex point of $\mathbf{x}^{(k)}$, and obtain the value of $f\left(x^{(k)}\right)$ directly from the Bernstein coefficient value at $x^{(k)}$.
4. {Compute $\mathbf{J}$} From the Bernstein coefficients of f on $\mathbf{x}^{(k)}$, compute the Bernstein coefficients of all the first partial derivatives of f on $\mathbf{x}^{(k)}$via (7). From the minimum and maximum Bernstein coefficients of the first derivatives, construct their range enclosing intervals, and form the interval Jacobian matrix $\mathbf{J}\left(x^{(k)}, \mathbf{x}^{(k)}\right)$.
5. {Compute the preconditioner} Compute the preconditioning matrix R as
$$R = \left\{\text{mid } \mathbf{J}\left(x^{(k)}, \mathbf{x}^{(k)}\right)\right\}^{-1}$$
6. {Solve linear interval system} Solve the linear interval system
$$-Rf\left(x^{(k)}\right) = R\mathbf{J}\left(x^{(k)}, \mathbf{x}^{(k)}\right)\left[\mathbf{N}\left(x^{(k)}, \mathbf{x}^{(k)}\right) - x^{(k)}\right]$$
and obtain $\mathbf{N}\left(x^{(k)}, \mathbf{x}^{(k)}\right)$. The interval Gauss-Seidel method [9] or the interval hull method [18] can be used to solve the system. Update the solution
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \bigcap \mathbf{N}\left(x^{(k)}, \mathbf{x}^{(k)}\right)$$
7. (Check for non-existence) If $\mathbf{x}^{(k+1)} = \emptyset$, then print 'no solution exists in $\mathbf{x}$' and EXIT algorithm.
8. {Termination} If wid $\mathbf{x}^{(k+1)} < \epsilon$, print 'solution: ', $\mathbf{x}^{(k+1)}$ and EXIT algorithm.
9. {Find Bernstein coefficients for the new box} Apply the Bernstein Coefficient Contraction procedure given in section 4
$$B(\mathbf{x}^{(k+1)}) = \text{BCC}(B(\mathbf{x}^{(k)}), \mathbf{x}^{(k)}, \mathbf{x}^{(k+1)})$$
and obtain the Bernstein coefficients on the contracted box $\mathbf{x}^{(k+1)}$.
10. Set $k = k + 1$ and return to Step 3.

END Algorithm

If division by zero occurs in Step 6 of the above algorithm, then it can be dealt with using extended interval arithmetic, see [6].

The presence of multiple zeros in $\mathbf{x}$ can be handled in this manner, but requires maintaining a working list of boxes, etc. We refer the reader to [4].

4 Bernstein Coefficient Contraction

The Bernstein Newton algorithm contracts an input box $\mathbf{d}$ into a smaller box $\mathbf{d}_{new}$. The Bernstein coefficients on the new box $\mathbf{d}_{new}$ then need to be computed for further work. Straightforward evaluation of the Bernstein coefficients over the new box will, however, be time consuming. We instead propose to use a new so-called Bernstein coefficient contraction to obtain the Bernstein coefficients for the contracted box $\mathbf{d}_{new}$. The coefficients for the contracted box $\mathbf{d}_{new}$ are derived from those already available for $\mathbf{d}$.

This procedure uses the idea in section 2.2. To explain the idea, consider any component direction, say the first. In this direction, the interval components are $\mathbf{d}_1 = [\underline{\mathbf{d}_1}, \overline{\mathbf{d}_1}]$ and $\mathbf{d}_{1new} = [\underline{\mathbf{d}_{1new}}, \overline{\mathbf{d}_{1new}}]$, with $\mathbf{d}_{1new} \subseteq \mathbf{d}_1$.

First, obtain the Bernstein coefficients on the subinterval $[\underline{\mathbf{d}_1}, \overline{\mathbf{d}_{1new}}]$ from those on the interval $\mathbf{d}_1 = [\underline{\mathbf{d}_1}, \overline{\mathbf{d}_1}]$. This can be done using (8) and (9) where $\lambda = \lambda_1$ is given by

$$\lambda_1 = \frac{(\overline{\mathbf{d}_{1new}} - \underline{\mathbf{d}_1})}{(\overline{\mathbf{d}_1} - \underline{\mathbf{d}_1})} \tag{11}$$

Next, similarly obtain the Bernstein coefficients on the subinterval $[\underline{\mathbf{d}_1}, \underline{\mathbf{d}_{1new}}]$ from those on the interval $[\underline{\mathbf{d}_1}, \overline{\mathbf{d}_{1new}}]$, where $\lambda = \lambda_2$ is given by

$$\lambda_2 = \frac{(\underline{\mathbf{d}_{1new}} - \underline{\mathbf{d}_1})}{(\overline{\underline{\mathbf{d}_{1new}}} - \underline{\mathbf{d}_1})} \tag{12}$$

Lastly, obtain the required Bernstein coefficients on subinterval $[\underline{\mathbf{d}_{1new}}, \overline{\mathbf{d}_{1new}}]$ as intermediate values of the computation via (10), as $[\underline{\mathbf{d}_{1new}}, \overline{\mathbf{d}_{1new}}]$ is the neighboring subinterval to $[\underline{\mathbf{d}_1}, \underline{\mathbf{d}_{1new}}]$ when $[\underline{\mathbf{d}_1}, \overline{\mathbf{d}_{1new}}]$ is subdivided. Repeat this procedure in all other component directions to obtain the Bernstein coefficients $B(\mathbf{d}_{new})$.

We present an algorithm for this so-called Bernstein coefficient contraction.

Algorithm: Bernstein Coefficient Contraction (BCC)
$B(\mathbf{d}_{new}) = \mathbf{BCC}(B(\mathbf{d}), \mathbf{d}, \mathbf{d}_{new})$

Inputs: Bernstein coefficients $B(\mathbf{d})$ for the given box $\mathbf{d}$, given box $\mathbf{d}$, and the contracted box $\mathbf{d}_{new} \subseteq \mathbf{d}$.
Outputs: Bernstein coefficients $B(\mathbf{d}_{new})$ for the contracted box $\mathbf{d}_{new}$.

BEGIN Algorithm:

1. Choose the first component direction, i.e., consider $\mathbf{d}_1$.
2. Compute the values of λ_1 and λ_2 using equations (11) and (12).
3. Using $\lambda = \lambda_1$ in (8) and (9), obtain the value of Bernstein coefficients on $[\underline{\mathbf{d}_1}, \overline{\mathbf{d}_{1new}}]$ from the Bernstein coefficients on $\mathbf{d}_1$.
4. Using $\lambda = \lambda_2$ in (8), (9) and (10), obtain the Bernstein coefficients on subinterval $[\underline{\mathbf{d}_{1new}}, \overline{\mathbf{d}_{1new}}]$ from the Bernstein coefficients found in the above step.
5. Repeat the above steps for all the remaining component directions.
6. Return the Bernstein coefficients got at the end of the above step as $B(\mathbf{d}_{new})$.

END Algorithm

5 Examples

In this section we present two examples which illustrate the superiority of the proposed Bernstein Newton operator over the interval Newton operator. We used INTLAB [20], a MATLAB [13] package to perform interval related operations. All computations are done on a desktop PC Pentium IV, 3 GHz with 512 MB RAM.

5.1 Example 1

This example is taken from [19, 24]. This is a problem with 4 variables. The polynomial system is given by

$$1 + x_1 + x_2 + x_3 + x_4 = 0$$
$$x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4 = 0$$
$$x_1x_2 + x_1x_2x_3 + x_2x_3x_4 + x_3x_4 + x_4x_1 = 0$$
$$x_1x_2x_3 + x_1x_2x_3x_4 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 = 0$$

and the bounds on the variables are

$$\mathbf{x}_1 = [0.95, 1.05], \mathbf{x}_2 = [0.95, 1.05], \mathbf{x}_3 = [-2.65, -2.6], \mathbf{x}_4 = [-0.4, -0.37].$$

We perform three consequential iterations to contract the domain using Bernstein Newton contractor (BNC) and Interval Newton contractor (INC). We perform the iterations until we obtain an accuracy of 10^{-10}. The results are tabulated in Table 1.

Table 1: Comparison Convergence of solution using Interval Newton contractor and Bernstein Newton contractor

Iter	Bounds (Interval Newton)	Width (Interval Newton)	Bounds (Bernstein Newton)	Width (Bernstein Newton)
1	[0.9568225842, 1.0433575799]	0.0865349957	[0.9967561714, 1.0032653277]	0.0065091563
	[0.9500000000, 1.0500000000]	0.0000000001	[0.9959739153, 1.0040438373]	0.0080699220
	[-2.6243772238, -2.6000000000]	0.0243772238	[-2.6213883739, -2.6124901398]	0.0088982341
	[-0.3846435421, -0.3700000000]	0.0146435421	[-0.3829125022, -0.3804434631]	0.0024690391
2	[0.9888255033, 1.0109474627]	0.0221219594	[0.9999842682, 1.0000154365]	3.11682999e-05
	[0.9863125650, 1.0136169675]	0.0273044025	[0.9999802551, 1.0000194310]	3.91759000e-05
	[-2.6243772238, -2.6122128287]	0.0121643951	[-2.6180607485, -2.6180075112]	5.32372999e-05
	[-0.3846435421, -0.3797684476]	0.0048750945	[-0.3819734874, -0.3819590099]	1.44774999e-05
3	[0.9999564046, 1.0000499831]	9.35785000e-05	[0.9999999999, 1.0000000000]	<1e-10
	[0.9999446065, 1.0000611056]	1.16499100e-04	[0.9999999999, 1.0000000000]	<1e-10
	[-2.6181056946, -2.6179507960]	1.54898599e-04	[-2.6180339888, -2.6180339887]	<1e-10
	[-0.3819894111, -0.3819396989]	4.97121999e-05	[-0.3819660112, -0.3819660112]	<1e-10
4	[0.9999999961, 1.0000000037]	7.60000007e-09		
	[0.9999999952, 1.0000000046]	9.40000011e-09		
	[-2.6180339951, -2.6180339823]	1.28000001e-08		
	[-0.3819660131, -0.3819660093]	3.79999998e-09		
5	[0.9999999999, 1.0000000000]	<1e-10		
	[1.0000000000, 1.0000000000]	<1e-10		
	[-2.6180339887, -2.6180339887]	<1e-10		
	[-0.3819660112, -0.3819660112]	<1e-10		

From Table 1 we observe that the existing interval Newton operator requires 5 iterations to bound the roots of the polynomial systems with the accuracy of 10^{-10}. The proposed Bernstein Newton operator algorithm computes the result in 3 iterations within the same accuracy. The computational time required for the proposed Bernstein Newton method is 0.186 seconds, whereas the interval Newton operator method requires a computational time 0.235 seconds.

5.2 Example 2

This example is taken from [7]. This is a problem in 3 variables. The system of polynomial equations is

$$5x_1^9 - 6x_1^5x_2^2 + x_1x_2^4 + 2x_1x_3 = 0$$
$$-2x_1^6x_2 + 2x_1^2x_2^3 + 2x_2x_3 = 0$$
$$x_1^2 + x_2^2 - 0.265625 = 0$$

and the initial bound is $\mathbf{x} = ([0.45, 0.5], [0.2, 0.24], [0, 0.03])$. We perform a few iterations to contract the domain using Bernstein Newton contractor and Interval Newton contractor. We perform the iterations until we obtain an accuracy of 10^{-8}. The results are tabulated in Table 2.

Table 2: Comparison Convergence of solution using Interval Newton contractor and Bernstein Newton contractor

Iter	Bounds (Interval Newton)	Width (Interval Newton)	Bounds (Bernstein Newton)	Width (Bernstein Newton)
1	[0.44999999, 0.49054379]	0.04054380	[0.44999999, 0.47706957]	0.027069580
	[0.20000000, 0.24000000]	0.04000000	[0.20000000, 0.24000000]	0.040000000
	[0.00000000, 0.00912466]	0.00912466	[0.00000000, 0.00413040]	0.004130400
2	[0.45294658, 0.47072578]	0.01777920	[0.46553462, 0.46952652]	0.003991900
	[0.20000000, 0.23705003]	0.03705003	[0.21314384, 0.22152842]	0.008384580
	[0.00000000, 0.00185627]	0.00185627	[0.00000000, 0.00024456]	2.445600000e-04
3	[0.46548509, 0.46991059]	0.00442550	[0.46694462, 0.46701765]	7.302999999e-05
	[0.21231032, 0.22291622]	0.01060590	[0.21799997, 0.21814033]	1.403600000e-04
	[0.00000000, 0.00041399]	4.13990000e-04	[0.00000000, 0.00000450]	4.500000000e-06
4	[0.46690292, 0.46705687]	1.53949999e-04	[0.46698001, 0.46698001]	<1e-08
	[0.21792489, 0.21822869]	3.03799999e-04	[0.21807033, 0.21807033]	<1e-08
	[0.00000000, 0.00001413]	1.41300000e-05	[0.00000000, 0.00000000]	<1e-08
5	[0.46697998, 0.46698003]	5.000000002e-08		
	[0.21807028, 0.21807038]	1.000000000e-07		
	[0.00000000, 0.00000000]	<1e-08		
6	[0.46698001, 0.46698001]	<1e-08		
	[0.21807033, 0.21807033]	<1e-08		
	[0.00000000, 0.00000000]	<1e-08		

¿From Table 2 we observe that the existing interval Newton operator requires 6 iterations to bound the roots of the polynomial systems with the accuracy of 10^{-08}. The proposed Bernstein Newton operator algorithm computes the result in 4 iterations within the same accuracy. The computational time required for the proposed Bernstein Newton method is 0.171 seconds, whereas the interval Newton operator method requires a computational time 0.296 seconds.

6 Conclusions

In this paper we presented an algorithm for contracting the search domain using Bernstein Newton contractor. We used Bernstein approach to find the enclosures for the derivatives of the polynomial functions. The evaluation of the values of the functions is avoided by directly obtaining the values of the functions at corner points. We used Bernstein coefficients contraction to avoid the repeated computation of Bernstein coefficients of polynomial functions. We presented two examples to show the superiority of the Bernstein Newton contractor. It is found in these examples that the proposed

algorithm encloses the roots of the polynomial systems with the desired accuracy in smaller number of iterations as compared to the interval Newton contractor

References

[1] J. Garloff, Convergent bounds for range of multivariate polynomials, *Interval Mathematics* (K. Nickel, Ed.), *Lecture Notes in Computer Science*, Springer, Berlin, vol. 212, pp. 37–56, 1985.

[2] J. Garloff, The Bernstein algorithm, *Interval Computations*, vol. 2, pp. 154–168, 1993.

[3] J. Garloff, Solving strict polynomial inequalities by Bernstein expansion, *The Use of Symbolic Methods in Control System Analysis and Design* (N. Munro, Ed.), IEE Contr. Engg. Ser., Springer, Berlin, vol. 56, pp. 339–352, 1999.

[4] R. Hammer, M. Hocks, U. Kulisch and D. Ratz, *Rigorous Global Search: Continuous Problems,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[5] R. Hammer, M. Hocks, U. Kulisch and D. Ratz, *Numerical Toolbox for Verified Computing I*, Springer-Verlag, 1991.

[6] E. Hansen and G. W. Walster, *Global Optimization using Interval Analysis,* Marcel Dekker, Inc., New York, 2004.

[7] C. Jager and D. Ratz, A combined method for enclosing all solutions of non-linear systems of polynomial equations, *Reliable Computing*, vol. 1, pp. 41–64, 1995.

[8] L. Jaulin, M. Kieffer and O. Didrit, *Applied Interval Analysis,* Springer, London, 2001.

[9] R. B. Kearfott, *Rigorous Global Search: Continuous Problems,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[10] L. Kolev, An interval method for global nonlinear analysis, *IEEE Tran. Circuits and Systems-I*, vol. 45, no. 5, pp. 675–683, 2000.

[11] S. Malan, M. Milanese, M. Tarangna and J. Garloff, B^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic perturbations, *Proc. 31st Conf. on Decision and Control.*, Tucson, Arizona, pp. 128–133, 1992.

[12] S. Malan, M. Milanese, M. Tarangna and J. Garloff, Robust analysis and design of control system using interval arithmetic, *Automatica*, vol. 33, no. 7, pp. 1363–1372, 1997.

[13] M. R. Guide, MATLAB version 6.1, The Mathworks Inc., Natick, MA, 2001.

[14] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[15] P. S. V. Nataraj and M. Arounassalame, A new subdivision algorithm for Bernstein polynomial approach to global optimization, *International Journal of Automation and Computing*, vol. 4, no. 4, pp. 342–352, 2007.

[16] P. S. V. Nataraj and K. Kotecha, Global optimization with higher order inclusion function forms part 1: A combined Taylor-Bernstein form, *Reliable Computing*, vol. 10, no. 1, pp. 27–44, 2004.

[17] P. S. V. Nataraj and K. Kotecha, An improved interval global optimization algorithm using higher order inclusion function forms, *Journal of Global Optimization*, vol. 32, no. 1, pp. 35–63, 2005.

[18] A. Neumaier, A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations, *Reliable Computing,* vol. 5, pp. 131–136, 1999.

[19] A. K. Prakash, Vectorized interval analysis algorithms and their applications, Ph. D. dissertation, IIT Bombay, Mumbai, India, 2003.

[20] S. M. Rump, *INTLAB-Interval Laboratory,* Developments in reliable computing, Kluwer Academic Publishers, pp. 77-104, 1999.

[21] S. Shaswathi Ray, A new approach to range computation of polynomials using the Bernstein form, Ph. D. dissertation, IIT Bombay, Mumbai, India, 2006.

[22] A. P. Smith, Fast construction of constant bound functions for sparse polynomials, *Journal of Global Optimization*, vol. 43, no. 2-3, pp. 445-458, 2009.

[23] V. Stahl, Interval methods for bounding the range of polynomials and solving systems of nonlinear equations, Ph. D. dissertation, University of Linz, Linz, 2006.

[24] J. Verschelde, The PHC pack, the database of polynomial systems, *Technical report*, University of Illinois, Mathematics Department, Chicago, 2001.

[25] M. Zettler and J. Garloff, Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion, *IEEE Trans. Autom. Contr.*, vol. 43, no. 5, pp. 425–431, 1998.