Global Optimization Toolbox using Bernstein Form

Indian Institute of Technology, Bombay

2009-2010

INDEX

Global Optimization

Global Optimization

Authors: Prof. P. S. V. Nataraj, Mr. Dhiraj Magare, Mr. Bhagyesh Patil

Optimization Primer

1 What is Optimization?

Optimization is the study of how to find the best (optimum) solution to a problem.

2 Objective Function

The objective function, or cost function is the mathematic representation of the problem to be solved. The function should be formulated such that its values vary with a parameter (or parameters). The goal is to find the value of the parameter at which the function obtains an extreme (i.e. lowest or highest) value. Typically, if the highest value is the goal, the function is multiplied by -1 so that this new function now obtains its minimum where the old function obtains its maximum. This is done so that algorithms can be standardized, always looking for the minimum of the function provided. Because of this, the problem is generally expressed

$$\min_x f(x)$$

where f is the objective function.

Generally, it is very difficult to find the solutions analytically. Therefore, most optimization methods are iterative, meaning they start at an initial point, and "move" around the parameter space looking for the solution. There are many ways to do this.

3 Global VS Local Minima

Figure 1 illustrates a function f defined over a two-dimensional space X = (X1,X2). A point x^* is called a global minimizer if over the entire possible range of parameters, provides the lowest function value. This is generally extremely hard to find, and will be discussed in further detail.

$$f(x^*) \leq f(x)$$

A local minimizer is a point x^* which is not necessarily lower than ALL other points, but it is lower than all of the points surrounding it. There are very "mathy" ways to describe these neighboring points.

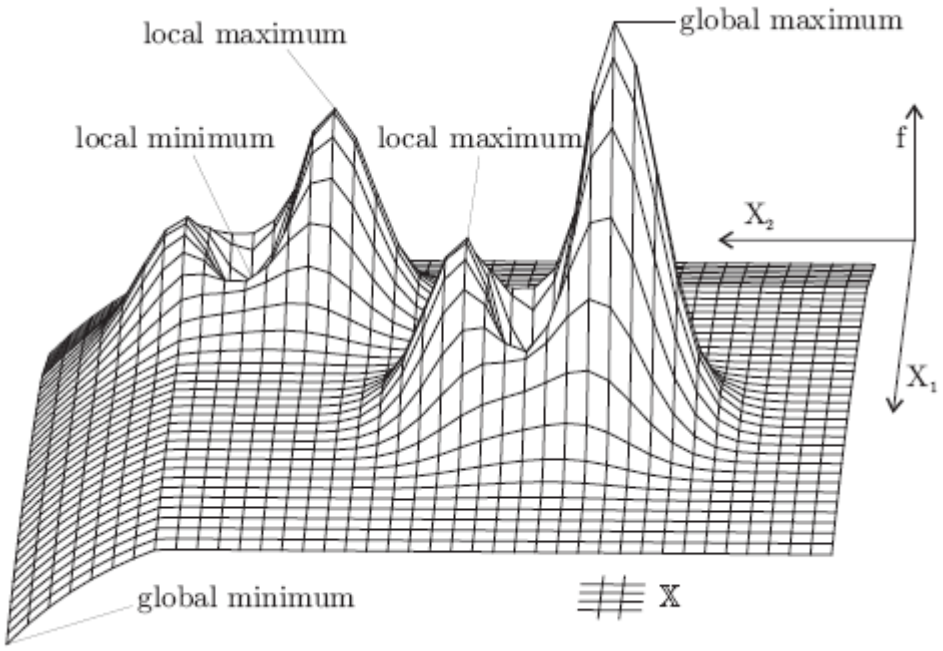


Figure 1 : Global and local optima of a two-dimensional function

4 Unconstrained Optimization

Optimization problems for which the parameter is allowed to take any value are called “unconstrained” problems.

4.1 Optimality Conditions

From basic calculus, we know that the minimum of a function occurs when the derivative is 0. Since we frequently deal with more than a single dimensional problem, we require that the gradient be zero.

4.1.1 First Order Necessary Conditions

A necessary condition is a condition which must be true at a solution point, but knowing that it is true is not enough to ensure the point is a solution. They are called “first order” because they look at only the first derivative of the objective function.

$$\nabla f(x^*) = 0$$

4.1.2 Second Order Necessary Conditions

$$\nabla^2 f(x^*) \geq 0$$

That is, the Hessian of the objective function is positive semi-definite. When, $\nabla^2 f(x^*) = 0$ the point could be a saddle point of the function instead of a minimum, which is the reason this condition is not sufficient to show that a point is the minimum.

4.1.3 Second Order Sufficient Conditions

A sufficient condition is one which if it is true for a point, the point is definitely a solution.

$$\nabla^2 f(x^*) > 0$$

As described above, the only difference between this sufficient condition and the second order necessary condition is that the Hessian is not allowed to be zero.

4.2 Line Search Methods

A general class of algorithm which, at each iteration, chooses a direction from the current point and looks for a lower function value along that direction. The length of the step to take in that direction is extremely important.

4.3 Steepest Descent Direction

The negative gradient is an obvious choice for the direction which will yield a lower objective function value. This method is very simple in that it does not require calculating the second derivative. However, it can be very slow.

4.4 Newton Direction

$$p_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k$$

This is troublesome because if the Hessian is not positive definite, the Newton direction may not be defined. However, if it does work, it has a fast rate of convergence.

4.5 Quasi-Newton Method

In place of the true Hessian, they use an approximation to the hessian which is updated at each iteration. This approximation is found by looking at the difference in the gradient, which is exactly a discrete second derivative. The search direction is found by simply replacing the Hessian with the Hessian approximation.

$$p_k = -B_k^{-1} \nabla f_k$$

Optimization Strategy

Most global optimization strategies fall into one of the two categories - Exact methods and Heuristic methods.
Deterministic algorithms belonging to the first group of exact methods, provide a rigorous guarantee for finding at least one or all global solutions. However, for larger dimensional models, and for more complicated model functions, the associated computational burden can easily become excessive.
Stochastic algorithms, which come under the second category of heuristic methods, as a rule do not provide a strict convergence guarantee.

Main focus of this toolbox is on global optimization for polynomial problems. Polynomial optimization problems arise in the mathematical modelling of several real world applications. Knowledge of the range of a polynomial in several variables on a multidimensional rectangle is relevant for numerous investigations and applications in numerical and functional analysis, combinatorial optimization, and finite geometry. In many applications powerful multidimensional polynomial representations are crucial to many aspects of object modelling, For instance, in control and communication applications,

simple and efficient polynomial evaluation is important for real-time implementation of nonlinear controllers.

Exact algebraic techniques for solving polynomial programming problems find all the critical points, and then identify the smallest value of the polynomial at any critical point. Such techniques include Grobner bases, eigenvalue method, resultants and discriminants and numerical homotopy methods , but these may be computationally expensive and the number of critical points could be infinite.

To find the globally optimal solutions of polynomial programs, several existing methods use linearization techniques to approximate polynomial terms . Since linearizations are only approximations, accuracy (i.e., sharpness) is less important than efficiency. Few algorithms for global optimizations of multivariate polynomial functions use some relaxation techniques (such as linear matrix inequalities), which are further modifications of the linearization techniques. These techniques involve sum of squares and semidefinite programming. The accuracy of the solutions depends heavily on the relaxation order (the minimal relaxation order can never be less than twice the degree of the polynomial), and hence have to be invoked iteratively with increasing orders until the global optimum is reached. When the relaxation order increases, the number of relaxed variables increases, hence the overall computational time increases very quickly.

Another method for computing the range of a polynomial is obtained through the use of Bernstein forms; these are intimately connected to Bernstein polynomials. Range analysis using the Bernstein form is based on the Bernstein convex hull property. The method relies on the simple idea that if a polynomial is written in the Bernstein basis over a box, the range of the polynomial is bounded by the values of the minimum and maximum Bernstein coefficients. The Bernstein form also allows bounding the range of a multivariate polynomial over a box.

Optimization Approaches

A. Deterministic

The most successful are:

- Interval Optimization / Interval Algebra.
- Branch and bound methods
- Methods based on real algebraic geometry

B. Stochastic, thermodynamics

Several Monte-Carlo-based algorithms exist:

- Simulated annealing
- Direct Monte-Carlo sampling
- Basin hopping technique (aka Monte-Carlo with minimization)
- Stochastic tunneling
- Parallel tempering
- Continuation methods

C. Heuristics and metaheuristics

Other approaches include heuristic strategies to search the search space in a (more or less) intelligent way, including:

- Evolutionary algorithms (e.g., genetic algorithms and evolution strategies)
- Swarm-based optimization algorithms (e.g., particle swarm optimization and ant colony optimization)
- Memetic algorithms, combining global and local search strategies
- Reactive search optimization (i.e. integration of sub-symbolic machine learning techniques into search heuristics)
- Differential evolution
- Graduated optimization

D. Response surface methodology based approaches

- IOSO Indirect Optimization based on Self-Organization

Links for global optimization

- http://www.mat.univie.ac.at/~neum/glopt.html
- http://www.it-weise.de/projects/book.pdf (Free e-book by Thomas Weise)

Applications of Global optimization

Typical examples of global optimization applications include:

- Protein structure prediction (minimize the energy/free energy function)
- Computational phylogenetics (e.g., minimize the number of character transformations in the tree)
- Traveling salesman problem and circuit design (minimize the path length)
- Chemical engineering (e.g., analyzing the Gibbs free energy)
- Safety verification, safety engineering (e.g., of mechanical structures, buildings)
- Worst case analysis
- Mathematical problems (e.g., the Kepler conjecture)
- The starting point of several molecular dynamics simulations consists of an initial optimization of the energy of the system to be simulated.
- Spin glasses
- Calibration of radio propagation models

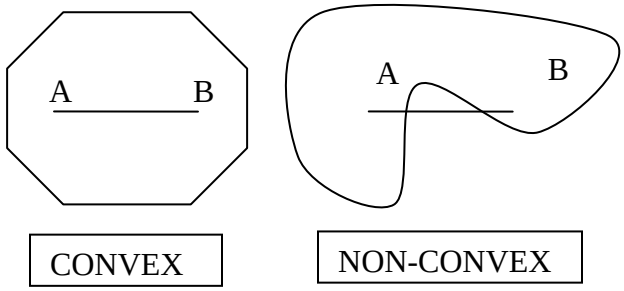
Optimization Toolbox using Bernstein Form

Bernstein Polynomial approach

If a polynomial is written in the Bernstein basis over a box, then the range of the polynomial is bounded by the values of the minimum and maximum Bernstein coefficients. Capturing the beautiful properties of the Bernstein polynomials, tight inclusions for the range of the polynomial on the given domain can be obtained.

The key feature of the Bernstein approach to range computations is that bounds on the global optima are guaranteed. The approach does not require any initial guess for starting the optimization, but only an initial search box bounding the domain of interest. Moreover, if multiple solutions are present, then all solutions are guaranteed to be obtained. Without any prior knowledge of stationary points, the global optimum can be found to the prescribed accuracy, subject to the machine precision. The approach has an added advantage that it requires neither resetting of any ‘tuning parameter’ values nor any ‘relaxations’.

- Convex / Non – Convex Problems



- Constrained / Unconstrained Problems

a) Constrained Problems :

I) Equality constrained Problems :

$$\min_x \{ f(x) \} \text{ subject to } g(x) = 0$$

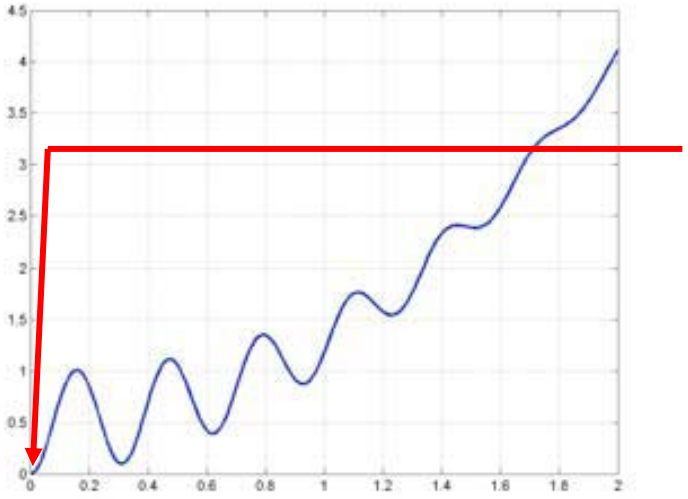
where $f(.)$ is the scalar-valued objective function and $g(.)$ is vector-valued on constraint function

II) Inequality constrained Problems:

$$\min_x \{ f(x) \} \text{ subject to } g(x) = 0 \text{ and } h(x) \le 0$$

where $f(.)$ is the scalar-valued objective function and $g(.)$ is vector-valued on equality constraint function; $h(.)$ is the inequality constraint function

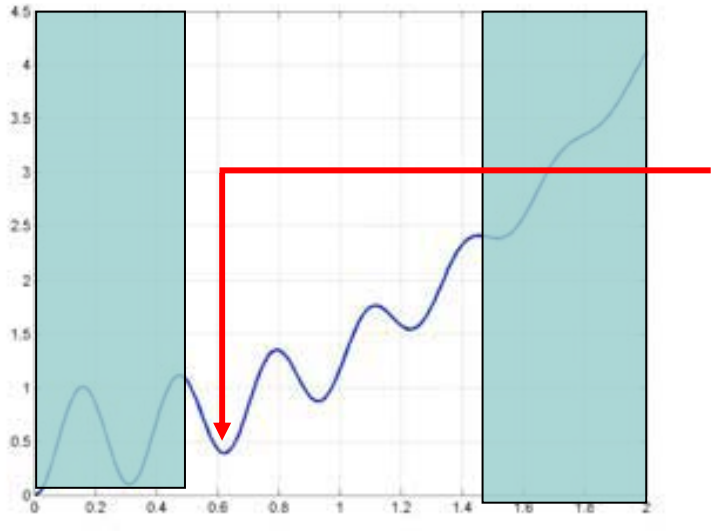
Example: find the optimum of the following function within the range $[0, +\infty)$



Minimum is zero

b) Unconstrained Problems :

Example: find the optimum of the following function within the range [0.5 , 1.5]



- Univariate / Multivariate Problems

Objective function involving one variable may be called univariate.

Objective function involving more than one variable may be called multivariate.

Bernstein form

Consider the n[th] degree polynomial p in a single variable in power form $x \in U = [0,1]$

$$p(x) = \sum_{i=0}^{n} a_i x^i \qquad a_i \in \mathbb{R} \;,\; n \in \mathbb{N}$$

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

$$p(x) = \sum_{j=0}^{k} b_j^k B_j^k(x), \qquad \forall\; x \in [0,1] \text{ and } k \geq n$$

where $B_j^k(x)$ are the Bernstein basis polynomials of degree k and b_j^k are the Bernstein coefficients

$$b_j^k = \sum_{i=0}^{j} a_i \frac{\binom{j}{i}}{\binom{k}{i}}$$

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

Properties of Bernstein Coefficients

1. Let $\overline{p}(u)$ denote the range of polynomial p on u. Then, by the range enclosing property of the Bernstein coefficients

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

2. Convex hull property :

Let $l \in \mathrm{N}$, be the number of variables and $x = (x_1, x_2, ..., x_l) \in \square^l$. A multi-index I is defined as $I = (i_1, i_2, ..., i_l) \in \mathrm{N}^l$ and and multi-power x is defined as $x^I = (x_1^{i_1}, x_2^{i_2}, ..., x_l^{i_l})$. A multi-index of maximum degrees N is defined as $N = (n_1, n_2, ..., n_l)$ and I / N for $(i_1 / n_1, i_2 / n_2, ..., i_l / n_l)$. For the multivariate case, the control points are $(I / N, b_I(u)) : I \in S$ .The convex hull property is,

$$conv\{(x, p(x)) : x \in u\} \subseteq conv\{(I / N, b_I(u)) : I \in S_0\}$$

where $S_0 = \{0, n_1\} \times \{0, n_2\} \times ... \times \{0, n_l\}$ $l \in S$ and $conv$ P denotes the convex hull of P, i.e. the smallest convex set containing the set P. Thus, the convex hull property states that the range $\overline{p}(u)$ is contained in the convex hull of the control points.

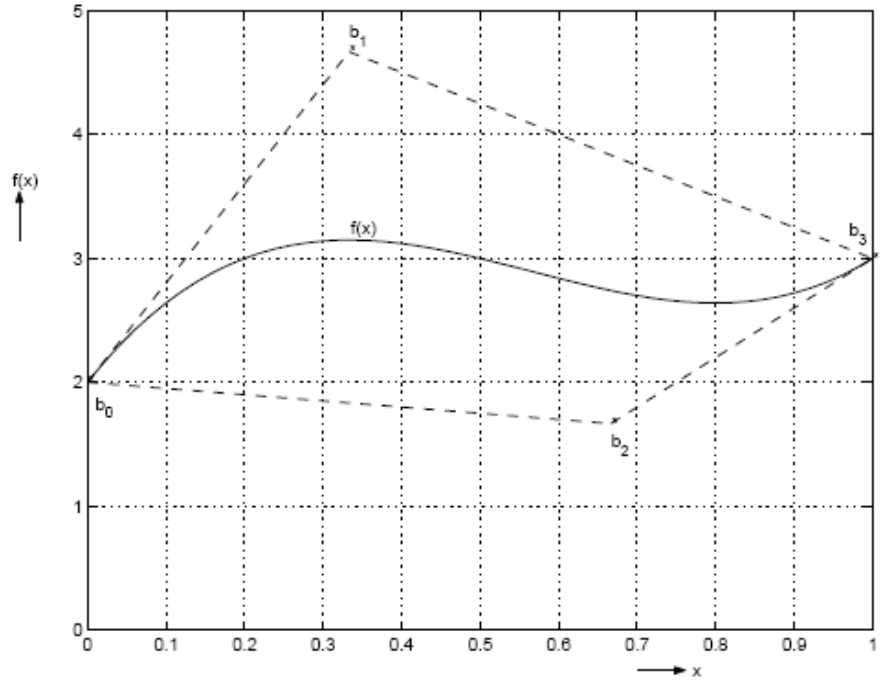


Figure : The polynomial function, its Bernstein coefficients, and the convex hull

Illustration

Example 1: To illustrate the Bernstein approach, consider the simple polynomial

$$p(x) = x(1-x)$$

whose range, [0, $\frac{1}{4}$]

➔ In this example, comparing with $a_2x^2 + a_1x + a_0$ where power n and coefficient are , $n = 2, a_0 = 0, a_1 = 1, a_2 = -1$

Using Bernstein coefficient formula $b_j^k = \sum_{i=0}^{j} a_i \frac{\binom{j}{i}}{\binom{k}{i}}$ Here, k (max. polynomial power)=2.

i & j = 0,1,2 (i.e. up to k).

gives, $b_0^2 = 0$, $b_1^2 = \frac{1}{2} or 0.5$, $b_2^2 = 0$

For Example

Bernstein Coefficient for following polynomials are:

Example 1: x^2
Ans:
b_0^2 =---------- , b_1^2 =---------, b_2^2 =----------

Example 2: $x^3 + x^2 - x$
Ans:
b_0^3 =---------- , b_1^3 =---------, b_2^3 =----------, b_3^3 =----------

Example 3: $x^4 + x$
Ans:
b_0^4 =---------- , b_1^4 =---------, b_2^4 =----------, b_3^4 =----------, b_4^4 =----------

Example 4: $x^3 - 2x^2 - x + 12$
Ans:
b_0^3 =---------- , b_1^4 =---------, b_2^3 =----------, b_3^3 =----------

Example 5: $1 - x + x^2 + 2x^3 + 3x^4$
Ans:
b_0^4 =---------- , b_1^4 =---------, b_2^4 =----------, b_3^4 =----------, b_4^4 =----------

Example 6: $-1 - x + 5x^2 - 3x^3$
Ans:
b_0^3 =---------- , b_1^3 =---------, b_2^3 =----------, b_3^3 =----------

Vertex Property of Bernstein Form or Vertex Test

- Remarkable feature:
  Bernstein form provides us with a criterion to indicate if calculated estimation is range or not.

- Cargo and Shisha (1966), give such a criterion based on the vertex property.
- Lemma 1 (Range lemma) (Cargo and Shisha, 1966): The range is bounded by the Bernstein coefficients as:

$$\overline{p}([0,1]) \subseteq \left[\min_j b_j^k, \max_j b_j^k \right]$$

- Lemma 2 (Vertex lemma)

$$\overline{p}([0,1]) = \left[\min_j b_j^k, \max_j b_j^k \right]$$

  If and only if,

$$\min_j b_j^k = \min\{b_0^k, b_k^k\}$$

  and

$$\max_j b_j^k = \max\{b_0^k, b_k^k\}$$

- Vertex lemma also holds for any subinterval of [0,1].

Illustration

Consider again Example 1: $p(x) = x(1-x)$

For k=2, Bernstein Coefficients are $b_0^2 = 0$, $b_1^2 = \frac{1}{2} or 0.5$, $b_2^2 = 0$

Range lemma implies,

$$\overline{p}([0,1]) \subseteq \left[0, \frac{1}{2}\right]$$

➔Check if above enclosure is the range itself or not.

Now, apply Vertex lemma

Minimum Bernstein coefficient is b_0^2 and b_2^2 - occurs at vertices $j \in \{0,2\}$.

Maximum Bernstein coefficient is b_1^2 - occurs at vertex j =1.

Vertex lemma is satisfied for the minimum.

Vertex lemma is not satisfied for the maximum as, $\max_j b_j^k \neq \max\{b_0^2, b_2^2\}$.

So, by vertex lemma, above enclosure is not the range.

Vertex Test Algorithm:

1. Calculate Bmin from current B (Bernstein patch).
2. Calculate minimum vertices of B. Bvermin.
3. Calculate vertices of B. Bvertex.
4. Check whether Bmin = = Bvermin ?
    1. Then, update stored minimum, zcap = min(Bmin,Bvermin)
       // initial zcap = max(B) taken as worst case.
    2. Check Bmin = = Bvertex. ?
    3. Store current box in solution list, LXsol.
5. If no, go for box subdivision.

Subdivision of Bernstein Form

Vertex Condition not satisfied then apply Subdivide test – J. Garloff (1993).

Let $D=[\underline{d},\overline{d}]\subseteq U$ and assume we have already the Bernstein coefficients on D. Suppose D is bisected to produce two sub boxes D_A and D_B given by,

$$D_A=[\underline{d},m(D)]\ ;\ D_B=[m(D),\overline{d}]\ .$$

Then, the Bernstein coefficients on the sub boxes D_A and D_B can be obtained from those on D, by executing the following algorithm.

Subdivision Algorithm:

Inputs: The box, $D\subseteq U$ and its Bernstein Coefficients $\{\overline{b}_j^k\}$.

Outputs: Sub boxes D_A and D_B, and their Bernstein Coefficients $\{\tilde{b}_j^k\}$ and $\{\hat{b}_j^k\}$.

STRAT

1. Bisect D to produce the two sub boxes D_A and D_B.
2. Compute the Bernstein Coefficients on sub box D_A as follows:
   a. Set: $b_j^0\leftarrow\overline{b}_j^k$, for $j=0,...,k$
   b. FOR $i=1,...,k$ DO

$$b_j^i=\begin{cases}b_j^{i-1} & for\ \ j<i\\ \frac{1}{2}\left\{b_{j-1}^{i-1}+b_j^{i-1}\right\} & for\ \ j\geq i\end{cases}$$

   To obtain the new coefficients, apply the formula given above for $j=0,...,k$.

   c. Find the Bernstein coefficients on sub box D_A as

$$\tilde{b}_j^k=b_j^k,\ \text{for}\ \ j=0,...,k\ .$$

3. Find Bernstein Coefficients on sub box D_B from intermediate values in above step, as follows

$$\hat{b}_j^k=b_k^j\ \ \text{for}\ \ j=0,...,k\ .$$

4. RETURN D_A, D_B and the associated Bernstein Coefficients $\{\tilde{b}_j^k\}$ and $\{\hat{b}_j^k\}$.

Illustration

Algorithm Subdivision for Example 1.

For k =4, we have already the Bernstein coefficients $\bar{b}_j^k$ for the interval D= [0,1].
With these as the inputs to Algorithm subdivision, the results are:
Bisect D to produce the two sub boxes D_A = [1,0.5] and D_B =[0.5,1].

The Bernstein coefficients on the sub box D_A are $\left(0, \frac{1}{8}, \frac{10}{48}, \frac{1}{4}, \frac{1}{4}\right)$

The Bernstein coefficients on the sub box D_B are $\left(0, \frac{1}{8}, \frac{10}{48}, \frac{1}{4}, \frac{1}{4}\right)$

It is coincidental here that Bernstein coefficients for both the sub boxes are the same.

Hence, by range lemma

$$p(D_A) \subseteq \left[0, \frac{1}{4}\right]$$

$$p(D_B) \subseteq \left[0, \frac{1}{4}\right]$$

And by vertex lemma,

$$p(D_A) = \left[0, \frac{1}{4}\right]$$

$$p(D_B) = \left[0, \frac{1}{4}\right]$$

In this example, using just *one* subdivision and application of the vertex lemma to the sub boxes we have been able to obtain the range of the given polynomial.

Cut-off Test

At any subdivision level, check if range in each new patch is already include in actual range stored (patches for which vertex condition is satisfied).
Reject patch, if YES.

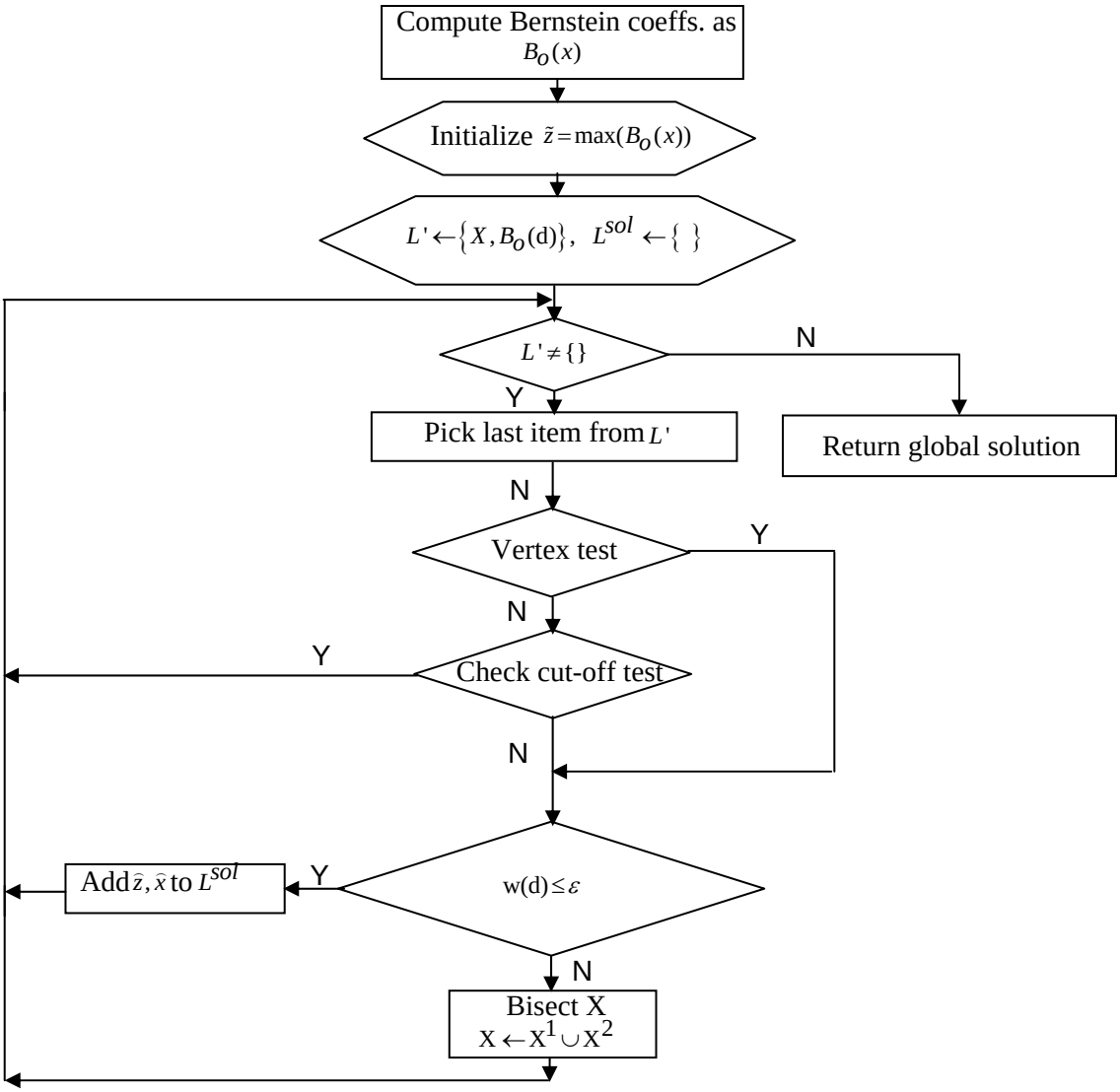
Cut-off test Algorithm:

1. Calculate minimum of B: Bmin
2. Use latest updated zcap
3. Check IF Bmin > zcap, THEN
   delete or discard current box. END
4. Update zcap=min(zcap,Bvermin)

Global Optimization Algorithm

Inputs: Degree N of all variables, polynomial coefficient matrices A for the objective function and specified box X.

Output: Global minimum zcap to the specified tolerance $\in$ and all global minimizers $X^{(i)}$



Univariate and Unconstrained Examples with their Global minimum and minimizer

Example 1. $-x^4-3x^3+x^2+2x-1 \quad x=[-1,1]$

Use Tolerance on minimum (zcap) and minimizer (bound/box): 0.000001

Ans: **Type on Scilab command window:**

```
x=-1:0.1:1;
f1=-x.^4-3*x.^3+x.^2+2.*x-1; // Use dot operator for calculation
plot(x,f1)
```

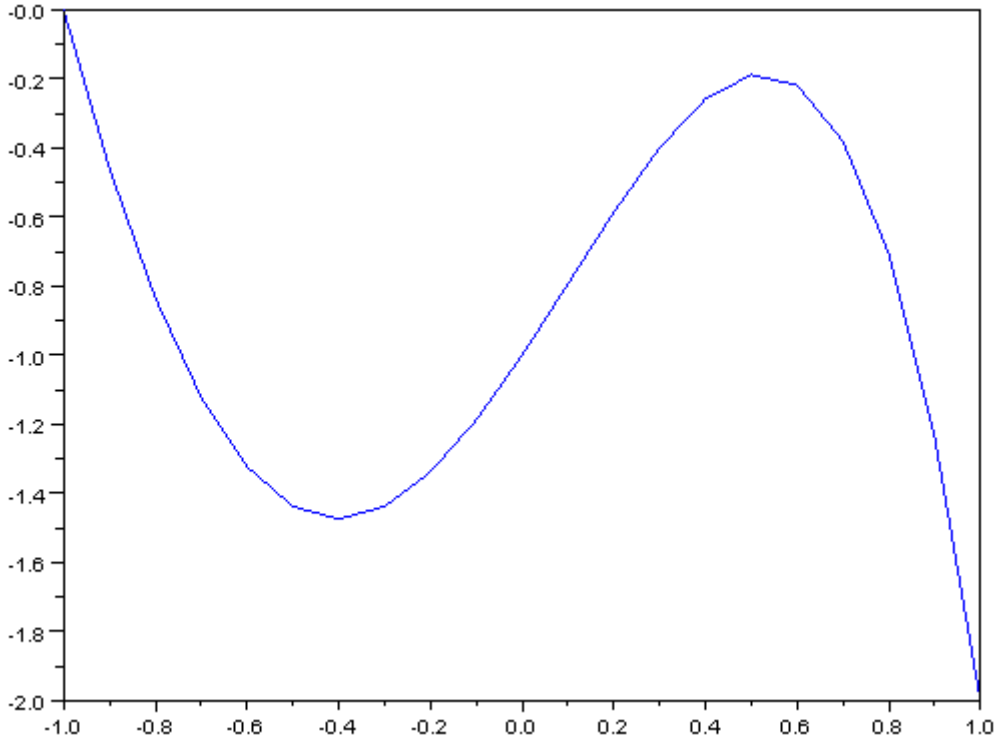


Figure: Example 1

Minimum, zcap = -2

Minimizer, x= [1,1]

Example 2. $10x^3 - 17x^2 + 8x + 2 \quad x = [0,1]$

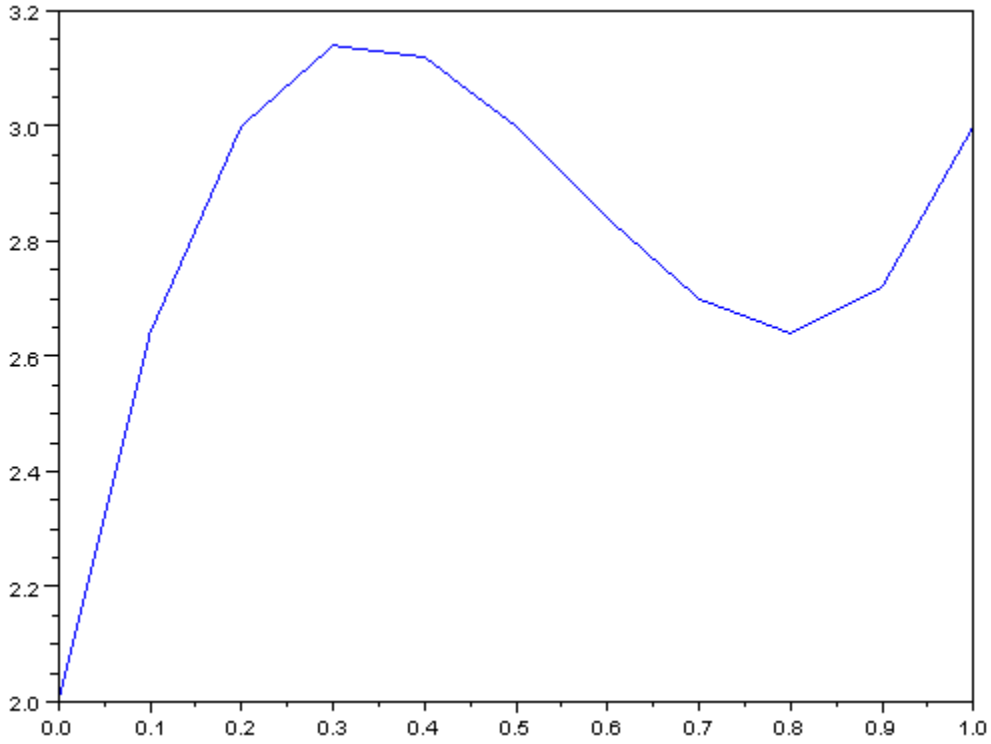


Figure: Example 2

Ans:
Minimum, zcap = 2
Minimizer, x=[0,0]

Example 3. $2.3x^6 + x^5 + x^3 - 0.4x + 0.5 \quad x = [-0.8, 0.6]$

Ans:
Minimum, zcap = 0.4106
Minimizer, x=[0.3184,0.31834]

Example 4. $x^5 + 0.2x^4 + 0.5x^3 + 0.7x^2 - x \qquad x = [0,1]$

Ans:
Minimum, zcap = -0.2407
Minimizer, x=[0.4043,0.4043]

Example 5. $-x^5 + 0.2x^4 + 0.5x^3 + 0.7x^2 - x \quad x = [-1,1]$

Ans:
Minimum, zcap = -0.6
Minimizer, x=[1,1]

Bivariate and Multivariate Examples with their Global Minimum and minimizer

1. Camel back: The six hump camel back function
f(x,*y*) = 4*x^2-2.1*x^4+1/3*x^6+x*y-4*y^2+4*y^4
where, **x,y** = [-3, 3],
zcap =
- 1.0316285
Lsol =
Lsol(1)
- 0.0898418 - 0.0898418
0.7126565 0.7126565

Lsol(2)
0.0898418 0.0898418
- 0.7126565 - 0.7126565

2. Booth: The function defined by
f(*x,y*) = *74-38*x+5*x^2-34*y+8*x*y+5*y^2*

x,y=[-10, 10]
zcap =
5.116D-13
Lsol =
3.0004883 3.0004883
0.9997559 0.9997559

3) Heart dipole: A heart dipole problem (Eight Variables)
f(p,q,r,s,t,u,v,w) = -p*u^3+3*p*u*v^2-r*v^3+3*r*v*u^2-q*t^3+3*q*t*w^2-s*w^3+3*s*w*t^2-0.9563453

p= [-0.1, 0.4], *q* = [0.4, 1], *r* = [-0.7,-0.4], *s* = [-0.7, 0.4],
t = [0.1, 0.2], *u* = [-0.1, 0.2], *v* = [-0.3, 1.1], *w* = [-1.1,-0.3]

zcap =
- 1.7434486
Lsol =
0.4 0.4
0.4 0.4
- 0.7 - 0.7
- 0.7 - 0.7
0.1 0.1
- 0.0789063 - 0.0789063
- 0.3 - 0.3
- 1.1 - 1.1

References

1. BOOK: C. G. Lorenz. Bernstein polynomials. University of Toronto Press, Toronto, 1953.
2. Shashwati Ray, P.S.V. Nataraj. An efficient algorithm for range computation of polynomials using the Bernstein form. *Journal of Global Optimization,* 45: 403-426,2009
3. G. T. Cargo and O. Shisha. The Bernstein form of a polynomial. *Jl. of research of NBS,*70B:79-81,1966
4. S. Ray, A new approach to range computation of polynomial problems using the Bernstein form. *PhD thesis,* Systems and Control Engineering, IIT Bombay, India, 2007.