Frontiers In Global Optimization, pp. 1-2
C. A. Floudas and P. M. Pardalos, Editors
©2003 Kluwer Academic Publishers

An Improved Method for the Computation of Affine Lower Bound Functions for Polynomials

J. Garloff
University of Applied Sciences / FH Konstanz
Department of Computer Science
Postfach 100543
D-78405 Konstanz
Germany
garloff@fh-konstanz.de

A. P. Smith
University of Applied Sciences / FH Konstanz
Institute for Applied Research
Postfach 100543
D-78405 Konstanz
Germany
smith@fh-konstanz.de

Abstract

This paper addresses the problem of finding tight affine lower bound functions for multivariate polynomials. Such underestimating functions are needed if global optimization problems involving polynomials are solved with a branch and bound method. These bound functions are constructed by using the expansion of the given polynomial into Bernstein polynomials. In contrast to our previous method which requires in the general case the solution of a linear programming problem, we propose here a method which requires only the solution of a system of linear equations together with a sequence of back substitutions and the computation of slopes. An error bound exhibiting quadratic convergence in the univariate case and some numerical examples are presented.

Keywords: Bernstein polynomials, control points, convex hull, bound functions, complexity, branch and bound, global optimization.

1 Introduction

Finding a convex lower bound function for a given function is of paramount importance in global optimization when a branch and bound approach is used. Of special interest are convex envelopes, i.e., uniformly best underestimating convex functions, cf. [3, 9, 12].

Because of their simplicity and ease of computation, constant and affine lower bound functions are especially useful. Constant bound functions are thoroughly used when interval computation techniques are applied to global optimization, cf. [7, 8, 11]. However, when using constant bound functions, all information about the shape of the given function is lost. A compromise between convex envelopes, which require in the general case much computational effort, and constant lower bound functions are affine lower bound functions. In [5] we concentrate on such bound functions for multivariate polynomials. These bound functions are constructed from the coefficients of the expansion of the given polynomial into Bernstein polynomials. For properties of Bernstein polynomials the reader is referred to [2, 4, 10, 13]. In the univariate case the computational work for constructing such bound functions is negligible, but in the multivariate case a linear programming problem has to be solved. In the branch and bound framework it may happen that one has to solve subproblems on numerous subboxes of the starting region, so that for higher dimensions solving the linear programming problems becomes a computational burden.

In this paper we present a method for constructing such affine lower bound functions for polynomials which requires the computation of slopes, the solution of a system of linear equations, and a sequence of back substitutions. This method in general requires fewer arithmetic operations and has lower complexity than our previous approach in [5].

The organisation of the paper is as follows: In the next section we recall some basic definitions and properties of Bernstein polynomials. Affine lower bound functions based on Bernstein expansion together with an error bound are presented in Section 3. The results are illustrated by numerical examples in Section 4. Finally, some conclusions are given.

2 Bernstein polynomials and notation

We define multiindices $i = (i_1, \ldots, i_n)^T$ as vectors, where the n components are nonnegative integers. The vectors 0 and 1 denote the multiindices with all components equal to 0 or 1, respectively, which should not cause ambiguity. Comparisons are used entrywise. Also the arithmetic operators on multiindices are defined componentwise such that $i \odot l := (i_1 \odot l_1, \ldots, i_n \odot l_n)^T$, for $\odot = +, -, \times$, and $/$ (with $l > 0$). For $x \in \mathbf{R}^n$ its multipowers are

$$x^i := \prod_{\mu=1}^{n} x_\mu^{i_\mu}. \tag{1}$$

Multipowers of multiindices are not required here; instead we shall write $i^0, \ldots, i^n$ for a sequence of $n+1$ multiindices. For the sum we use the notation

$$\sum_{i=0}^{l} := \sum_{i_1=0}^{l_1} \cdots \sum_{i_n=0}^{l_n} . \tag{2}$$

A multivariate polynomial p of degree $l = (l_1, \ldots, l_n)^T$ can be represented as

$$p(x) = \sum_{i=0}^{l} a_i x^i \quad \text{with} \quad a_i \in \mathbf{R}, 0 \leq i \leq l, \text{ and } a_l \neq 0. \tag{3}$$

The ith Bernstein polynomial of degree l is

$$B_i(x) := \binom{l}{i} x^i (1-x)^{l-i}, \tag{4}$$

where the generalized binomial coefficient is defined by $\binom{l}{i} := \prod_{\mu=1}^{n} \binom{l_\mu}{i_\mu}$, and x is contained in the unit box $I = [0,1]^n$. It is well-known that the Bernstein polynomials form a basis in the space of multivariate polynomials, and each polynomial in the form (3) can be represented in its Bernstein form over I

$$p(x) = \sum_{i=0}^{l} b_i B_i(x), \tag{5}$$

where the *Bernstein coefficients* b_i are given by

$$b_i = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} a_j \quad \text{for} \quad 0 \leq i \leq l. \tag{6}$$

A fundamental property for our approach is the *convex hull property*

$$\left\{ \begin{pmatrix} x \\ p(x) \end{pmatrix} : x \in I \right\} \subseteq conv \left\{ \begin{pmatrix} i/l \\ b_i \end{pmatrix} : 0 \leq i \leq l \right\}, \tag{7}$$

where the convex hull is denoted by *conv*. The points $\binom{i/l}{b_i}$ are called *control points* of p. The enclosure (7) yields the inequalities

$$\min\{b_i : 0 \leq i \leq l\} \leq p(x) \leq \max\{b_i : 0 \leq i \leq l\} \tag{8}$$

for all $x \in I$. For ease of presentation we shall sometimes simply use b_i to denote the control point associated with the Bernstein coefficient b_i, where the context should make this unambiguous. Exponentiation on control points, Bernstein coefficients, or vectors is also not required here; therefore $b^0, \ldots, b^n$ is a sequence of $n+1$ control points or Bernstein coefficients (with $b^j = b_{ij}$), and $u^1, \ldots, u^n$ is a sequence of n vectors.

3 Affine lower bound functions

In this section we show how affine lower bound functions for multivariate polynomials based on Bernstein expansion can be constructed by only solving a system of linear equations together with a sequence of back substitutions. The simplest type of affine lower bound function is a constant one. The left-hand side of (8) implies that the constant function provided by the minimum Bernstein coefficient

$$c_0(x) = b_{i^0} = \min\{b_i : 0 \leq i \leq l\} \tag{9}$$

is an affine lower bound function for the polynomial p given by (3) over the unit box I. The following construction aims to find hyperplanes passing through the control point b^0 (associated with the minimum Bernstein coefficient b_{i^0}) which approximate from below the lower part of the convex hull of the control points increasingly well. In addition to b^0, we will designate n additional control points $b^1, \ldots, b^n$. Starting with c_0, we will construct from these control points a sequence of affine lower bound functions $c_1, \ldots, c_n$. We end up with c_n, a hyperplane which passes through a lower facet of the convex hull spanned by the control points $b^0, \ldots, b^n$. In the course of this construction, we will generate a set of linearly independent vectors $\{u^1, \ldots, u^n\}$ and we will compute slopes from b^0 to b^j in direction u^j. Also, w^j will denote the vector connecting b^0 and b^j.

3.1 Algorithm

First Iteration:

$$\text{Let } u^1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Compute slopes g_i^1 from the control point b_i to b^0 in direction u^1:

$$g_i^1 = \frac{b_i - b^0}{\frac{i_1}{l_1} - \frac{i_1^0}{l_1}} \quad \text{for all } i \text{ with } i_1 \neq i_1^0.$$

Let i^1 be a multiindex with smallest absolute value of associated slope g_i^1. Designate the control point $b^1 = \left(\frac{i^1}{l}, b_{i^1}\right)^T$, the slope $\alpha_1 = g_{i^1}^1$, and the vector $w^1 = \frac{i^1 - i^0}{l}$. Define the lower bound function

$$c_1(x) = b^0 + \alpha_1 u^1 \cdot \left(x - \frac{i^0}{l}\right).$$

jth Iteration, $j = 2, \ldots, n$:

$$\text{Let } \tilde{u}^j = \begin{pmatrix} \beta_1^j \\ \vdots \\ \beta_{j-1}^j \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\text{such that } \tilde{u}^j \cdot w^k = 0, \ k = 1, \ldots, j-1. \tag{10}$$

Normalize this vector thusly:

$$u^j = \frac{\tilde{u}^j}{\|\tilde{u}^j\|}. \tag{11}$$

Compute slopes g_i^j from the control point b_i to b^0 in direction u^j:

$$g_i^j = \frac{b_i - c_{j-1}(\frac{i}{l})}{\frac{i-i^0}{l} \cdot u^j} \quad \text{for all } i, \text{ except where } \frac{i-i^0}{l} \cdot u^j = 0. \tag{12}$$

Let i^j be a multiindex with smallest absolute value of associated slope g_i^j. Designate the control point $b^j = \left(\frac{i^j}{l}, b_{ij}\right)^T$, the slope $\alpha_j = g_{ij}^j$, and the vector $w^j = \frac{i^j - i^0}{l}$. Define the lower bound function

$$c_j(x) = c_{j-1}(x) + \alpha_j u^j \cdot \left(x - \frac{i^0}{l}\right). \tag{13}$$

Remark: Solving (10) for the coefficients $\beta_1^j, \ldots, \beta_{j-1}^j$ requires the solution of a system of $j-1$ linear equations in $j-1$ unknowns. This system has a unique solution due to the linear independence amongst the vectors $w^1, \ldots, w^n$, given in the following theorem.

Theorem 3.1 *Let $w^{j,k}$ be the vectors in $\mathbf{R}^j$ given by taking the first j components of w^k, for $k = 1, \ldots, j$. Then the vectors $w^{j,1}, \ldots, w^{j,j}$ are linearly independent, for $j = 1, \ldots, n$.*

Proof (by induction):
Define vectors $u^{j,k}$ analogously by taking the first j components of u^k. The result holds for $j = 1$. Assume that $w^{j-1,1}, \ldots, w^{j-1,j-1}$ are linearly independent. By adding an extra component, we have that $w^{j,1}, \ldots, w^{j,j-1}$ are linearly independent. By (10), noting that only the first j components of u^j are nonzero, we have that $u^{j,j} \cdot w^{j,k} = u^j \cdot w^k = 0$, for $k = 1, \ldots, j-1$, i.e. $u^{j,j}$ is in the orthogonal complement of $w^{j,1}, \ldots, w^{j,j-1}$. Similarly by (12), we have that $u^{j,j} \cdot w^{j,j} = u^j \cdot w^j \neq 0$, i.e. $w^{j,j}$ is not orthogonal to $u^{j,j}$. Therefore $w^{j,j}$ is not in the subspace spanned by $w^{j,1}, \ldots, w^{j,j-1}$. $\square$

For the n iterations of the above algorithm, the solution of such a sequence of systems of linear equations would normally require $\frac{1}{6}n^4 + O(n^3)$ arithmetic operations. However we can take advantage of the fact that, in the jth iteration, the vectors $w^1, \ldots, w^{j-1}$ are unchanged from the previous iteration. The solution of these systems can then be formulated as Gaussian elimination applied rowwise to the single $(n-1) \times (n-1)$ matrix whose rows consist of the vectors $w^{n-1,1}, \ldots, w^{n-1,n-1}$ and right-hand side $-(w_n^1, \ldots, w_n^{n-1})^T$. In addition, a sequence of back-substitution steps has to be performed. Then altogether only $n^3 + O(n^2)$ arithmetic operations are required.

Let

$$L = \sqrt[n]{\prod_{i=1}^{n}(l_i + 1)}.$$

There are then L^n Bernstein coefficients, so that the computation of the slopes g_i^j (12) in all iterations requires at most $n^2 L^n + L^n O(n)$ arithmetic operations. This new approach therefore requires less computational effort in general than the method in [5], which is based on the solution of a linear programming problem with upto $L^n - 1$ constraints. [1]

[1] In our computations, cf. Sect. 4, we have chosen exactly $L^n - 1$ constraints.

Theorem 3.2 *With the notation of the above algorithm, it holds for all $j = 0, \ldots, n$ that*

$$c_j\left(\frac{i^k}{l}\right) = b^k, \quad \textit{for } k = 0, \ldots, j.$$

Proof (by induction):
By (9), we already have that $c_0\left(\frac{i^0}{l}\right) = b^0$. Assume that

$$c_{j-1}\left(\frac{i^k}{l}\right) = b^k, \quad \text{for } k = 0, \ldots, j-1.$$

Then we have for $k = 0$ by (13) and the induction hypothesis that

$$\begin{aligned} c_j\left(\frac{i^0}{l}\right) &= c_{j-1}\left(\frac{i^0}{l}\right) + \alpha_j u^j \cdot \left(\frac{i^0}{l} - \frac{i^0}{l}\right) \\ &= b^0. \end{aligned}$$

If $k = 1, \ldots, j-1$, we can conclude using (10)

$$\begin{aligned} c_j\left(\frac{i^k}{l}\right) &= c_{j-1}\left(\frac{i^k}{l}\right) + \alpha_j u^j \cdot \left(\frac{i^k}{l} - \frac{i^0}{l}\right) \\ &= b^k + \alpha_j u^j \cdot w^k \\ &= b^k. \end{aligned}$$

Finally, if $k = j$, we apply (12) to obtain

$$\begin{aligned} c_j\left(\frac{i^j}{l}\right) &= c_{j-1}\left(\frac{i^j}{l}\right) + \frac{b^j - c_{j-1}(\frac{i^j}{l})}{\frac{ij-i^0}{l} \cdot u^j} u^j \cdot \left(\frac{i^j}{l} - \frac{i^0}{l}\right) \\ &= c_{j-1}\left(\frac{i^j}{l}\right) + b^j - c_{j-1}\left(\frac{i^j}{l}\right) \\ &= b^j. \qquad \square \end{aligned}$$

In particular, we have that

$$c_n\left(\frac{i^k}{l}\right) = b^k, \ k = 0, \ldots, n, \tag{14}$$

which means that c_n passes through all $n+1$ control points $b^0, \ldots, b^n$. Since c_n is by construction a lower bound function, $b^0, \ldots, b^n$ must therefore span a lower facet of the convex hull of all control points.

As in [5], we obtain a pointwise error bound for the underestimating function c_n.

Theorem 3.3 *Let $\{b_i\}_{i=0}^l$ denote the Bernstein coefficients of an n-variate polynomial of degree l. Then the affine lower bound function c_n satisfies the a posteriori error bound*

$$0 \le p(x) - c_n(x) \le \max\left\{ b_i - c_n\left(\frac{i}{l}\right) \ : \ 0 \le i \le l \right\}, \ x \in I.$$

In the univariate case, this error bound specifies to the following bound which exhibits quadratic convergence with respect to the width of the intervals, see [5].

Corollary 3.1 *Suppose $n = 1$ and that the assumptions of Theorem 3.3 hold, then the affine lower bound function c_n satisfies the error bound*

$$0 \le p(x) - c_n(x) \le \max \left\{ \left(\frac{b_i - b^0}{\frac{i}{l} - \frac{i^0}{l}} - \frac{b^1 - b^0}{\frac{i^1}{l} - \frac{i^0}{l}} \right) \left(\frac{i}{l} - \frac{i^0}{l} \right) \; : \; 0 \le i \le l, \; i \ne i^0 \right\}, \; x \in I.$$

From [5] we already know that affine polynomials coincide with their affine lower bound functions constructed therein. We present here for the bound functions provided by the above algorithm a proof which is shorter than that for the more general case considered in [5].

Theorem 3.4 *Let $p(x) = a_0 + a_1 x_1 + \ldots + a_n x_n$. Then the lower bound function c_n coincides with p on I.*

Proof:
If p is affine, then $l = 1$ and $b_i = p(i)$, $0 \le i \le 1$, and we can conclude from (14)

$$c_n(i^k) = b^k = p(i^k), \; k = 0, \ldots, n.$$

Since the vectors $w^1, \ldots, w^n$ are linearly independent, the statement follows. □

Theorem 3.4 suggests that almost affine polynomials should be approximated rather well by the affine lower bound function c_n. This is confirmed by our numerical experiences.

Due to rounding errors, inaccuracies may be introduced into the calculation of the Bernstein coefficients and the lower bound functions. Especially it may happen that the computed lower bound function value is greater than the corresponding original function value. This may lead to erroneous results in applications. Functions which are guaranteed lower bound functions in the presence of rounding errors can be obtained along the suggestions made in [5].

In [6] we introduce a lower bound function for univariate polynomials which is composed of two affine lower bound functions. The extension to the multivariate case is as follows: In each step, compute slopes as before, but select α_j^- as the greatest negative g_i^j value, and α_j^+ as the smallest positive g_i^j value. From each previous lower bound function c_{j-1}, generate two new lower bound functions, using α_j^- and α_j^+. Instead of a sequence of functions, we now obtain after n iterations upto 2^n lower bound functions due to the binary tree structure.

4 Examples

With this method, we compute lower bound functions for a number of multivariate polynomials (3) in n variables with degree $l = (D, \ldots, D)^T$ and k non-zero terms. The non-zero coefficients are randomly generated with $a_i \in [-1, 1]$. These bound functions are compared to the bound functions of our previous method [5], which utilizes the linear programming

Table 1. Results for random polynomials

Method				Constant Bound Function		New Bound Function		Previous Bound Function [5]	
n	D	k	$(D+1)^n$	time (s)	δ	time (s)	δ	time (s)	δ
2	2	5	9	0.000040	1.414	0.000069	0.981	0.00020	0.976
2	6	10	49	0.00013	1.989	0.00031	1.677	0.0025	1.695
2	10	20	121	0.00039	2.867	0.00074	2.511	0.023	2.543
4	2	20	81	0.00037	3.459	0.0012	2.797	0.0082	2.847
4	4	50	625	0.0024	5.678	0.0093	5.045	2.82	5.056
6	2	20	729	0.0011	4.043	0.016	3.353	4.48	3.403
8	2	50	6561	0.0093	6.941	0.24	6.291	greater than	
10	2	50	59049	0.091	7.143	3.43	6.503	1 minute	

solver `LP_SOLVE` [1], and to constant bound functions defined by the minimum Bernstein coefficient b^0, cf. (9). Table 1 lists the results for different values of n, D, and k; $(D+1)^n$ is the number of Bernstein coefficients. In each case 100 random polynomials were generated and the mean computation time and error are given. The time required for the computation of the Bernstein coefficients is included; this is equal to the time for the constant bound functions. An upper bound on the discrepancy between the polynomial and its lower bound function over I is computed according to Theorem 3.3 as

$$\delta = \max_i \left\{ b_i - c_n \left(\frac{i}{l} \right) \right\}.$$

The results were produced with `C++` on a 2.4 GHz PC. The mean δ values for the two non-constant bound functions are very similar, with our new method exhibiting a slight improvement in all but the first case. However for an individual polynomial, one method may deliver a significantly superior bound function to the other, with the results only frequently identical in the $n = 2$ case. For $n \leq 4$ the computation time for our new bound functions is of the same order of magnitude as for the constant bound functions, and is much improved over the previous approach. Before one could typically compute bound functions in less than a second only for $n \leq 4$; for the new method this can done for $n \leq 8$.

5 Conclusions

We have presented a new method for the computation of affine lower bound functions for multivariate polynomials based on Bernstein expansion, for which a general construction requiring the solution of a linear programming problem was given in [5]. By our new method affine bound functions can be computed much more cheaply, and it may therefore be of greater practical use. Indeed one may compute upto 2^n of these new bound functions for a single given polynomial which jointly bound the convex hull of the control points much more closely than a single bound function from our previous approach, in less time.

It is worth noting that in the current version of our algorithm the choice of direction vectors u^j (11) is rather arbitrary. However our numerical experience suggests that this may influence the resultant bound function (i.e. which lower facet of the convex hull of the

control points is emulated). A future modification to the algorithm may therefore use a simple heuristic function to choose these vectors in an alternative direction such that a more suitable facet of the lower convex hull is designated. With the orthogonality requirement (10), there are $n - j$ degrees of freedom in this selection.

Acknowledgement

The authors gratefully acknowledge support from the Ministry of Education and Research of the Federal Republic of Germany under contract no. 1707001. They also thank Dr. Christian Jansson for his careful reading of the manuscript and his helpful comments.

References

[1] Berkelaar M., `LP_SOLVE`: Linear Programming Code. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/

[2] Cargo G. T. and Shisha O. (1966), "The Bernstein form of a polynomial," *J. Res. Nat. Bur. Standards Vol. 70B*, 79–81.

[3] Floudas C. A. (2000), "Deterministic Global Optimization: Theory, Methods, and Applications," *Series Nonconvex Optimization and its Applications Vol. 37*, Kluwer Acad. Publ., Dordrecht, Boston, London.

[4] Garloff J. (1986), "Convergent bounds for the range of multivariate polynomials," *Interval Mathematics 1985,* K. Nickel, editor, *Lecture Notes in Computer Science Vol. 212*, Springer, Berlin, 37–56.

[5] Garloff J., Jansson C. and Smith A. P. (2003), "Lower bound functions for polynomials," *J. Computational and Applied Mathematics*, to appear.

[6] Garloff J., Jansson C. and Smith A. P. (2003), "Inclusion isotonicity of convex-concave extensions for polynomials based on Bernstein expansion," *Computing*, to appear.

[7] Hansen, E. R. (1992), "Global Optimization Using Interval Analysis," Marcel Dekker, Inc., New York.

[8] Kearfott R. B. (1996), "Rigorous Global Search: Continuous Problems," *Series Nonconvex Optimization and its Applications Vol. 13*, Kluwer Acad. Publ., Dordrecht, Boston, London.

[9] Liberti L. and Pantelides C. C. (2002), "Convex envelopes of monomials of odd degree," *J. Global Optimization Vol. 25*, 157–168.

[10] Prautzsch H., Boehm W. and Paluszny M. (2002), "Bézier and B-Spline Techniques," Springer, Berlin, Heidelberg.

[11] Ratschek H. and Rokne J. (1988), "New Computer Methods for Global Optimization," Ellis Horwood Ltd., Chichester.

[12] Tawarmalani M. and Sahinidis N. V. (2002), "Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications," *Series Nonconvex Optimization and its Applications Vol. 65*, Kluwer Acad. Publ., Dordrecht, Boston, London.

[13] Zettler M. and Garloff J. (1998), "Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion," *IEEE Trans. Automat. Contr. Vol. 43*, 425–431.