A Comparison of Methods for the Computation of Affine Lower Bound Functions for Polynomials*

Jürgen Garloff and Andrew P. Smith

University of Applied Sciences / FH Konstanz
Postfach 100543, D-78405 Konstanz, Germany
{garloff,smith}@fh-konstanz.de

Dedicated to Professor Dr. Karl Nickel on the occasion of his eightieth birthday.

Abstract. In this paper the problem of finding an affine lower bound function for a multivariate polynomial is considered. For this task, a number of methods are presented, all based on the expansion of the given polynomial into Bernstein polynomials. Error bounds and numerical results for a series of randomly-generated polynomials are given.

Keywords: Bernstein polynomials, control points, convex hull, bound functions, complexity, 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. [5], [15], [21].

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. [10], [13], [20]. 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.

Here 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. Properties of Bernstein polynomials are introduced in Section 2; the reader is also referred to [4], [6], [18], [22]. In Section 3 we present a number of variant methods, together with a suitable transformation that may be applied to improve the results. Numerical results for a series of randomly-generated polynomials are given in Section 4, with a comparison of the error bounds.

* This work has been supported by the German Research Council (DFG).

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 vector 0 denotes the multiindex with all components equal to 0, 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 instance, $i/l, 0 \leq i \leq l$, defines the Greville abscissae. 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[1] $I = [0,1]^n$. 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

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

[1] Without loss of generality we consider in the sequel the unit box since any nonempty box in $\mathbf{R}^n$ can be mapped affinely thereupon. For the respective formulae for general boxes in the univariate case see [19] and their extensions to the multivariate case, e.g., Section 7.3.2 in [2].

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 $b^j = b_{i^j}$), and $u^1, \ldots, u^n$ is a sequence of n vectors.

3 Affine lower bound functions

In this section we explore a number of different methods for the computation of affine lower bound functions for polynomials. In each case it is assumed that we have a multivariate polynomial p given by (3) and that its Bernstein coefficients b_i, $0 \leq i \leq l$, have been computed.

Theorems 1 and 2 below are independent of any particular method. They characterize an affine lower bound function as the solution of a linear programming problem. There is a degree of freedom in that the statements contain an index set $\hat{J}$ which corresponds to a facet of the convex hull of the control points of p. According to the choice of $\hat{J}$ and the way in which the linear programming problem is posed (either all inequalities in (12) are considered or only a few), numerous related methods can be designed. We discuss a few in the sequel.

3.1 Method 1

Constant bound functions can be computed easily and cheaply from the Bernstein coefficients: 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. However, due to the lack of shape information, these bound functions usually perform relatively poorly.

3.2 Method 2

This method was presented in [7] and relies on the following construction: Choose a control point b_{i^0} with minimum Bernstein coefficient, cf. (9). Let $\hat{J}$ be a set of at least n multiindices such that the slopes between b_{i^0} and the control points with Greville abscissae associated with $\hat{J}$ are smaller than or equal to the slopes between b_{i^0} and the remaining control points. Then the desired affine lower bound function is provided as the solution of the linear programming problem to maximize the affine function at the Greville abscissae associated with $\hat{J}$ under the constraints that this affine function remains below all control points and passes through b_{i^0}. More precisely, the following theorem holds true.

Theorem 1. *Let $\{b_i\}_{i=0}^{l}$ denote the Bernstein coefficients of the polynomial p given by (3). Choose i^0 as in (9) and let $\hat{J} \subseteq \{\hat{j} : 0 \leq \hat{j} \leq l, \hat{j} \neq i^0\}$ be a set of at least n multiindices such that*

$$\frac{b_{\hat{j}} - b_{i^0}}{\|\hat{j}/l - i^0/l\|} \leqslant \frac{b_i - b_{i^0}}{\|i/l - i^0/l\|} \quad \text{for each } \hat{j} \in \hat{J},\ 0 \leq i \leq l,\ i \neq i^0,\ i \notin \hat{J}. \tag{10}$$

Here, $\|\cdot\|$ denotes some vector norm. Then the linear programming problem

$$\min \quad \Big(\sum_{\hat{j}\in\hat{J}} (\hat{j}/l - i^0/l)\Big)^T \cdot s \qquad \text{subject to} \tag{11}$$

$$(i/l - i^0/l)^T \cdot s \geq b_{i^0} - b_i \text{ for } 0 \leq i \leq l, i \neq i^0 \tag{12}$$

has the following properties:

1. *It has an optimal solution $\hat{s}$.*
2. *The affine function*

$$c(x) := -\hat{s}^T \cdot x + (\hat{s}^T \cdot (i^0/l) + b_{i^0}) \tag{13}$$

is a lower bound function for p on I.

In the univariate case, by definition (10), $\hat{J}$ can be chosen such that it consists of exactly one element $\hat{j}$ which may not be uniquely defined. The slope of the affine lower bound function c is equal to the smallest possible slope between the control points. Moreover, the optimal solution of the linear programming problem (11) and (12) can be given explicitly in the univariate case.

Theorem 2. *Suppose that all assumptions of Theorem 1 are satisfied, where $n = 1$ and where $\|\cdot\|$ denotes the absolute value. Choose $\hat{J} = \{\hat{j}\}$, where $\hat{j}$ satisfies*

$$\frac{b_{\hat{j}} - b_{i^0}}{|\hat{j}/l - i^0/l|} = \min\left\{\frac{b_i - b_{i^0}}{|i/l - i^0/l|} :\ 0 \leq i \leq l,\ i \neq i^0\right\}.$$

There then exists an optimal solution $\hat{s}$ of the linear programming problem (11),(12) which satisfies

$$\hat{s} = -\frac{b_{\hat{j}} - b_{i^0}}{\hat{j}/l - i^0/l}. \tag{14}$$

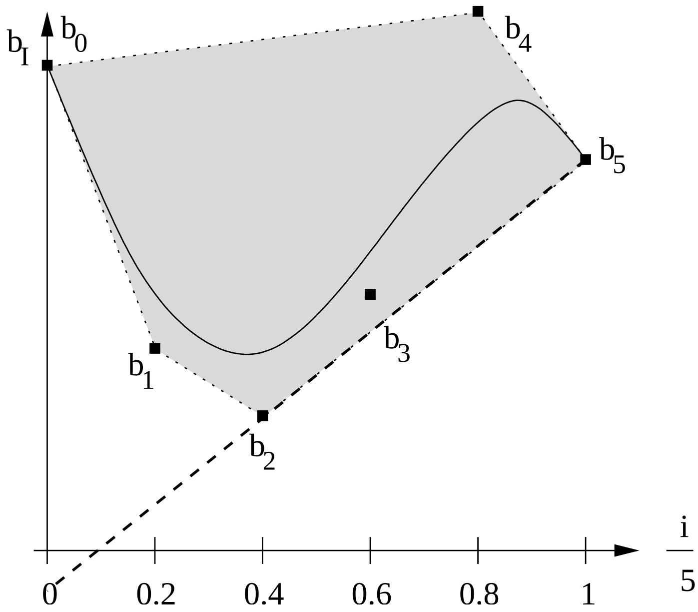

Fig. 1. The curve of a polynomial of fifth degree (bold), the convex hull (shaded) of its control points (marked by squares), and an affine lower bound function constructed as in Theorem 2.

Figure 1 illustrates the construction of such an affine lower bound function.

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.

3.3 Method 3

Overview This method was introduced in [9]. It only requires the solution of a system of linear equations together with a sequence of back substitutions. The following construction aims to find hyperplanes passing through the control point b^0 (associated with the minimum Bernstein coefficient b_{i^0}, cf. (9)) which approximate from below the lower part of the convex hull of the control points increasingly well. In addition to b^0, we designate n additional control points $b^1, \ldots, b^n$. Starting with c_0, cf. (9), we 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 generate a set of linearly independent vectors $\{u^1, \dots, u^n\}$ and we compute slopes from b^0 to b^j in direction u^j. Also, w^j denotes the vector connecting b^0 and b^j.

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

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

Algorithm - jth Iteration, $j = 2, \dots, n$:

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

$$\text{such that } \tilde{u}^j \cdot w^k = 0, \; k = 1, \dots, j-1. \tag{15}$$

Normalize this vector thusly:

$$u^j = \frac{\tilde{u}^j}{||\tilde{u}^j||}. \tag{16}$$

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

$$g_i^j = \frac{b_i - c_{j-1}(\frac{i}{l})}{\frac{i-i^0}{l} \cdot u^j} \quad \text{for all } i, \text{ except where } \frac{i - i^0}{l} \cdot u^j = 0. \tag{17}$$

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

$$c_j(x) = c_{j-1}(x) + \alpha_j u^j \cdot \left(x - \frac{i^0}{l}\right). \tag{18}$$

Remark: Solving (15) 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 $w^1, \ldots, w^n$, as proven in [9].

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 (17) 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 Method 2, which is based on the solution of a linear programming problem with upto $L^n - 1$ constraints. [2]

The following results were given in [9]:

Theorem 3. *With the notation of the above algorithm, it holds for all* $j = 0, \ldots, n$ *that*

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

In particular, we have that

$$c_n\left(\frac{i^k}{l}\right) = b^k, \ k = 0, \ldots, n, \tag{19}$$

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.

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

We obtain a pointwise error bound for the underestimating function c_n which also holds true for c_n replaced by the affine lower bound function c constructed by Method 2, cf. [7].

Theorem 4. *Let $\{b_i\}_{i=0}^{l}$ denote the Bernstein coefficients of the polynomial p given by (3). Then the affine lower bound function c_n satisfies the a posteriori error bound*

$$0 \leq p(x) - c_n(x) \leq \max\left\{ b_i - c_n\left(\frac{i}{l}\right) \;:\; 0 \leq i \leq l \right\},\; x \in I. \tag{20}$$

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 [7].

Theorem 5. *Suppose $n = 1$ and that the assumptions of Theorem 4 hold, then the affine lower bound function c_n satisfies the error bound ($x \in I$)*

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

Theorem 5 also holds true for c_n replaced by the affine lower bound function c of Method 2 and i^1 replaced by $\hat{j}$, cf. Theorem 2.

It was shown in [7] and [9] that affine polynomials coincide with their affine lower bound functions constructed therein. This suggests that almost affine polynomials should be approximated rather well by their affine lower bound functions. This is confirmed by our numerical experiences.

In [8] we introduced 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.

It is worth noting that in the current version of our algorithm the choice of the direction vectors u^j (16) 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 (15), there are $n - j$ degrees of freedom in this selection.

3.4 Methods 4 and 5

We also propose two simpler methods for the construction of affine lower bound functions based on the Bernstein expansion, with the computation of slopes and differences only, with still lower complexity. Method 4 is based on a choice of control points corresponding to $n+1$ smallest Bernstein coefficients and Method 5 is based on a choice of a control point corresponding to the minimum Bernstein coefficient and n others which connect to it with minimum absolute value of gradient. In both cases, a lower bound function interpolating the designated control points is computed, requiring the solution of a single system of linear equations. A degenerate case may arise when this system has no unique solution — with the terminology of Method 3, the set of vectors $\{w_j\}$ is linearly dependent. Such cases are tested for and excluded from consideration during the designation of the control points.

Additionally, both methods (unmodified) are not guaranteed to deliver a valid lower bound function — exceptionally there may still occur control points below it. Therefore an error term (20) is computed. If this is negative, it is necessary to adjust the bound function by a downward shift: the absolute value of this error is subtracted from its constant term.

As will become evident from the numerical results in the following section, both of these methods may perform unexpectedly poorly under certain configurations of control points. Two such examples are illustrated in the following figures, where the small circles are the control points of a bivariate polynomial. Those control points filled in black are those which are designated, leading to the construction of a lower bound function (the shaded plane), in the first case after a necessary downward shift. Although both methods usually deliver a bound function with correct shape information (i.e. an improvement over Method 1), this is seen not always to be the case. For this reason, there are no worthwhile error bounds that can be presented for these two methods.

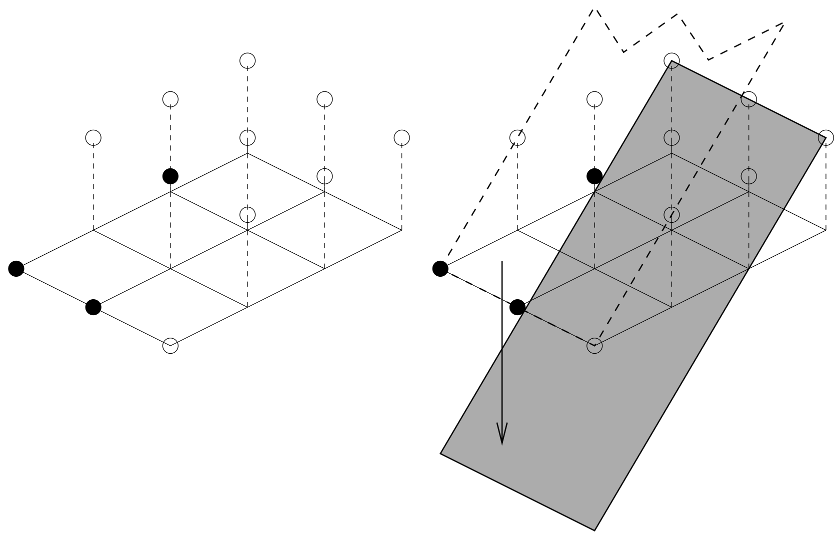

Fig. 2. Method 4 - Example of poor lower bound function

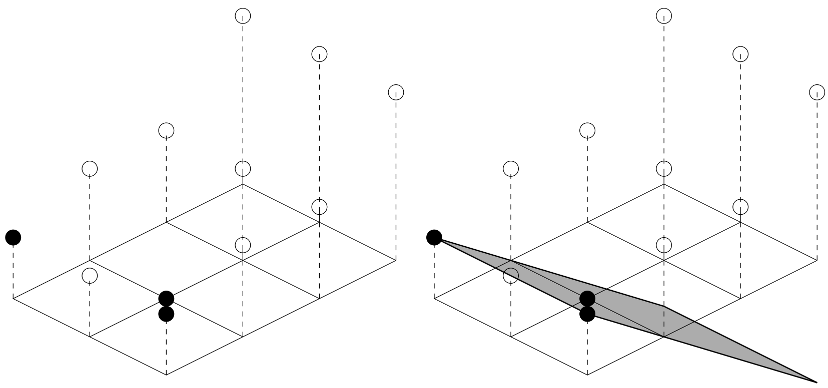

Fig. 3. Method 5 - Example of poor lower bound function

3.5 An equilibriation transformation

A limitation of all the above methods is that the resultant lower bound function must pass through the minimum control point b_{i^0} (except in cases where a downward shift is necessary for Methods 4 and 5). Whilst this is often a good choice, it is not always so. Figure 4 gives a simple example where the optimal lower bound function does not in fact pass through the minimum control point. In this case it would seem sensible to utilise the shape information provided by a broad spread of the control points (global shape information over the box) in addition to that already given by a small number of specially designated control points (which may be clustered) as per the above algorithms (local shape information near the minimum control point). We can lift the restriction that the lower bound function must pass through b_{i^0}. Indeed, if there are many Bernstein coefficients (i.e. for polynomials of high degree) the global shape information may be at least as important, if not more so, as the local information. This is especially evident in the cases where Methods 4 and 5 perform poorly (see Figures 2 and 3).

To this end, we can envisage the determination of the lower bound function as a three-stage process. Firstly, we apply an affine transformation to the control points, which we call the *equilibriation transformation*, derived from the control points on the edges of the box, and approximating the global shape information. Secondly, we compute an affine lower bound function c^* for the transformed polynomial p^* (and its control points b_i^*), by using one of Methods 1-5 above. Lastly, we apply the transformation in reverse to obtain an affine lower bound function c for the original polynomial.

We define the equilibriation transformation on the control points as follows:

$$b_i \;\mapsto\; b_i^* := b_i - \sum_{j=1}^{n} \frac{i_j}{l_j} \left(b_{\left(\lfloor \frac{l_1}{2} \rfloor, \ldots, l_j, \ldots, \lfloor \frac{l_n}{2} \rfloor\right)} - b_{\left(\lfloor \frac{l_1}{2} \rfloor, \ldots, 0, \ldots, \lfloor \frac{l_n}{2} \rfloor\right)} \right), \quad 0 \le i \le l.$$

After applying this transformation, the global shape (i.e. the shape over the whole box) of the polynomial has been approximately flattened, i.e.

$$\begin{aligned}
b^*_{(0,\lfloor\frac{l_2}{2}\rfloor,\ldots,\lfloor\frac{l_n}{2}\rfloor)} &= b^*_{(l_1,\lfloor\frac{l_2}{2}\rfloor,\ldots,\lfloor\frac{l_n}{2}\rfloor)}, \\
&\vdots \\
b^*_{(\lfloor\frac{l_1}{2}\rfloor,\ldots,0,\ldots,\lfloor\frac{l_n}{2}\rfloor)} &= b^*_{(\lfloor\frac{l_1}{2}\rfloor,\ldots,l_j,\ldots,\lfloor\frac{l_n}{2}\rfloor)}, \\
&\vdots \\
b^*_{\left(\lfloor\frac{l_1}{2}\rfloor,\ldots,\lfloor\frac{l_{n-1}}{2}\rfloor,0\right)} &= b^*_{\left(\lfloor\frac{l_1}{2}\rfloor,\ldots,\lfloor\frac{l_{n-1}}{2}\rfloor,l_n\right)}.
\end{aligned}$$

The effect of this transformation is illustrated in Figure 4 with a univariate polynomial of degree 6, yielding an optimal bound function which does not pass through the minimum control point.

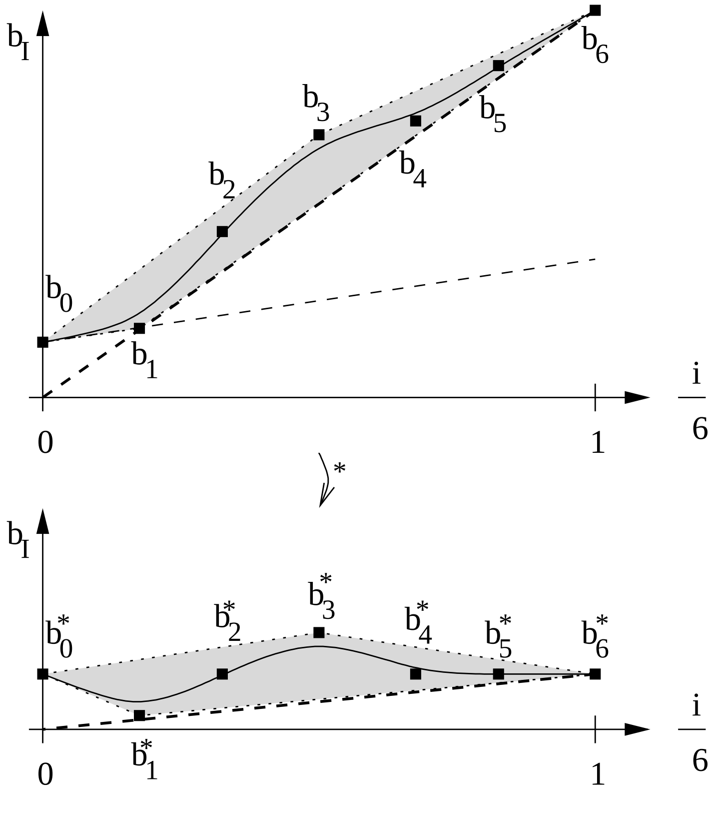

Fig. 4. Result of applying the equilibriation transformation; the improved/transformed bound function is given in bold dashed.

3.6 Verification

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. Suggestions for the way in which one can obtain functions which are guaranteed to be lower bound functions also in the presence of rounding errors are given in [7]. One such approach is to compute an error term (20) followed by a downward shift, if necessary, as in Methods 4 and 5. For a different approach see [3], [11], [14].

4 Examples

The above methods for computing lower bound functions, both with and without the equilibriation transformation, were tested with 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 were randomly generated with $a_i \in [-1, 1]$.

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 results were produced with `C++` on a 2.4 GHz PC. Method 2 utilizes the linear programming solver `LP_SOLVE` [1].

The time required for the computation of the Bernstein coefficients is included; this is equal to the time for Method 1 (constant bound functions). An upper bound on the discrepancy between the polynomial and its lower bound function over I is computed according to Theorem 4 as

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

The error bounds for the bound functions resulting from application of the equilibriation transformation are labelled δ_E and are computed identically. Note that after application of the equilibriation transformation, Method 1 delivers an affine function instead of a constant.

The mean δ values for Methods 2 and 3 are very similar, with Method 3 exhibiting a slight improvement in all but the first case. The poor mean δ values for Methods 4 and 5 are greatly skewed by a small minority of cases where the shape information is incorrect. These methods are unreliable. However for any given individual polynomial, any one method may deliver a significantly superior bound function to the other, with the results only frequently identical in the $n = 2$ case. The equilibriation transformation is effective in reducing the mean error bound in almost all cases, i.e. typically $\delta > \delta_E$. For $n \leq 4$ the computation time for Methods 3-5 is of the same order of magnitude as for Method 1 (constant bound function), and is faster by orders of magnitude than Method 2. Under that method, one can typically compute bound functions in less than a second only for $n \leq 4$; for Methods 3-5 this can be done for $n \leq 8$.

Table 1. Results for random polynomials

Method				**1** (Constant/Affine)					
n	D	k	$(D+1)^n$	time (s)	δ	δ_E	time (s)	δ	δ_E
2	2	5	9	0.000040	1.414	0.777			
2	6	10	49	0.00013	1.989	1.570			
2	10	20	121	0.00039	2.867	2.505			
4	2	20	81	0.00037	3.459	2.841			
4	4	50	625	0.0024	5.678	5.145			
6	2	20	729	0.0011	4.043	3.333			
8	2	50	6561	0.0093	6.941	6.505			
10	2	50	59049	0.091	7.143	6.583			
				2 (LP problems)			**3** (Linear eqs)		
2	2	5	9	0.00020	0.976	0.840	0.000069	0.981	0.866
2	6	10	49	0.0025	1.695	1.536	0.00031	1.677	1.533
2	10	20	121	0.023	2.543	2.383	0.00074	2.511	2.410
4	2	20	81	0.0082	2.847	2.690	0.0012	2.797	2.659
4	4	50	625	2.82	5.056	4.963	0.0093	5.045	4.880
6	2	20	729	4.48	3.403	3.292	0.016	3.353	3.201
8	2	50	6561	greater than			0.24	6.291	6.129
10	2	50	59049	1 minute			3.43	6.503	6.371
				4 (min BCs)			**5** (min gradients)		
2	2	5	9	0.000085	1.147	0.905	0.00011	0.961	0.885
2	6	10	49	0.00031	4.914	3.165	0.00044	1.910	1.514
2	10	20	121	0.00090	11.49	8.175	0.0012	3.014	2.514
4	2	20	81	0.0012	4.797	4.609	0.0015	3.199	2.766
4	4	50	625	0.0088	14.05	14.91	0.011	5.940	5.843
6	2	20	729	0.015	5.921	5.921	0.017	3.687	3.453
8	2	50	6561	0.21	14.33	15.41	0.24	7.360	7.313
10	2	50	59049	2.69	17.11	19.84	3.11	7.680	7.966

5 Conclusions

We have presented several methods for the computation of affine lower bound functions for multivariate polynomials based on Bernstein expansion. A simple constant bound function based on the minimum Bernstein coefficient (Method 1) can be computed cheaply, but performs poorly. It is possible to improve this by exploiting the valuable shape information inherent in the Bernstein coefficients. With Methods 4 and 5, we have demonstrated that a naive attempt to derive such shape information based on simple differences and gradients is unreliable. Methods 2 and 3 do this reliably and in general deliver a better quality bound function. The principal difference between these two lies in the computational complexity; the general construction of Method 2 requires the solution of a linear programming problem, whereas affine bound functions according to Method 3 can be computed much more cheaply, and may therefore be of greater practical use. Indeed one may compute up to 2^n of these 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 Method 2, in less time. Method 3 is therefore our current method of choice.

Methods 1-5 are limited by focussing on the shape information provided by a small number of designated control points, especially the minimum. Their performance can therefore be improved by incorporating the wider shape information provided by a broad spread of the control points. Our currently best overall results are thus obtained by combining Method 3 with the equilibriation transformation given in Section 3.5.

A fundamental limitation of our approach remains the exponential growth of the number of underlying Bernstein coefficients with respect to the number of variables. This means that many-variate (12 variables or more) polynomials cannot currently be handled in reasonable time. Future work will seek to address this limitation.

We have implemented the use of affine lower bound functions in a branch and bound framework for solving constrained global optimization problems involving a polynomial objective function and polynomial constraint functions. Relaxations based on these bound functions lead to linear programs. In practical problems, quite often only a few variables appear in the objective function and in each constraint. In this case, Method 3 may be highly suitable. If validated results are required, the solution of the linear program must be verified. This can be accomplished by using the results of [12], [16], [17].

References

1. Berkelaar M., `LP_SOLVE`: Linear Programming Code. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/
2. Berchtold J. (2000), "The Bernstein Form in Set-Theoretical Geometric Modelling," PhD thesis, University of Bath.
3. Borradaile G. and Van Hentenryck P. (2004), "Safe and tight linear estimators for global optimization," *Mathematical Programming*, to appear.
4. Cargo G. T. and Shisha O. (1966), "The Bernstein form of a polynomial," *J. Res. Nat. Bur. Standards Vol. 70B*, 79–81.
5. 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.
6. 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.
7. Garloff J., Jansson C. and Smith A. P. (2003), "Lower bound functions for polynomials," *J. Computational and Applied Mathematics Vol. 157*, 207–225.
8. Garloff J., Jansson C. and Smith A. P. (2003), "Inclusion isotonicity of convex-concave extensions for polynomials based on Bernstein expansion," *Computing Vol. 70*, 111-119.
9. Garloff J. and Smith A. P. (2004), "An improved method for the computation of affine lower bound functions for polynomials," in "Frontiers in Global Optimization," Floudas C. A. and Pardalos P. M., Eds., *Series Nonconvex Optimization with its Applications Vol. 74*, Kluwer Acad. Publ., Dordrecht, Boston, London, 135–144.

10. Hansen E. R. (1992), "Global Optimization Using Interval Analysis," Marcel Dekker, Inc., New York.
11. Hongthong S. and Kearfott R. B. (2004), "Rigorous linear overestimators and underestimators," submitted to *Mathematical Programming B*.
12. Jansson, C. (2004), "Rigorous lower and upper bounds in linear programming," *SIAM J. Optim. Vol. 14 (3)*, 914–935.
13. Kearfott R. B. (1996), "Rigorous Global Search: Continuous Problems," *Series Nonconvex Optimization and its Applications Vol. 13*, Kluwer Acad. Publ., Dordrecht, Boston, London.
14. Kearfott R. B. (2004), "Empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization," submitted to *Optimization Methods and Software*.
15. Meyer C. A. and Floudas C. A. (2004), "Trilinear monomials with mixed sign domains: facets of the convex and concave envelopes," *Journal of Global Optimization Vol. 29 (2)*, 125–155.
16. Michel C., Lebbah, Y. and Rueher M. (2003), "Safe embeddings of the simplex algorithm in a CSP framework," in Proc. 5th Int. Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2003), Université de Montréal, 210–210.
17. Neumaier A. and Shcherbina O. (2004), "Safe bounds in linear and mixed-integer programming," *Math. Programming A Vol. 99*, 283–296.
18. Prautzsch H., Boehm W. and Paluszny M. (2002), "Bézier and B-Spline Techniques," Springer, Berlin, Heidelberg.
19. Rokne J. (1977), "Bounds for an interval polynomial," *Computing Vol. 18*, 225–240.
20. Ratschek H. and Rokne J. (1988), "New Computer Methods for Global Optimization," Ellis Horwood Ltd., Chichester.
21. 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.
22. 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.