Affine Lower Bound Functions for Polynomials and their use in Global Optimisation

HPOPT 2004
CWI, Amsterdam
June 23–25, 2004

Jürgen Garloff
University of Applied Sciences /
Fachhochschule Konstanz
Department of Computer Science
and
University of Konstanz
Department of Mathematics and Statistics

Andrew P. Smith
Institute for Applied Research

Affine Lower Bound Functions for Polynomials and their use in Global Optimisation

Outline

- Relaxations of constrained global optimisation problems
- The Bernstein expansion for polynomials and its properties
- Affine lower bound functions for polynomials (5 methods)
- One algorithm in detail with an error bound
- Examples and a comparison of methods
- Future work and conclusions

Constrained Global Optimization Problem (GOP)

$$\min_{x \in M} f(x),$$

where the set M of feasible solutions is given by inequalities and equations

$$g_i(x) \leq 0,\ i = 1, \ldots, m,\ x \in X,$$

$$h_j(x) = 0,\ j = 1, \ldots, l,\ x \in X,$$

with $D \subseteq \mathbf{R}^n$, X being a box in D, and f, g_i, h_j are real-valued functions defined on D.

A frequently used approach is the generation of *relaxations* and their use in a branch and bound framework. Generally speaking, a relaxation of the given problem has the properties that

1. each feasible point of the given problem is feasible for the relaxation,

2. the relaxation is easier to solve than the original problem, and

3. the solutions of the relaxation converge to the solutions of the original problem, provided the maximal width of the set of feasible points converges to zero.

We obtain a relaxation for the GOP if we proceed as follows:

1. Replace the objective function f and the functions g_i by affine lower bound functions $\underline{f}$ and $\underline{g_i}$, respectively.
2. If the function h_j defining the j-th equality constraint is affine, then this equation is added to the constraints that define the relaxation. The remaining equations are written as two inequalities and these are treated according to (1).

Then we obtain the following optimization problem: $\min_{x \in N} \underline{f}(x)$ with the set N of feasible solutions given by

$$\underline{g_i}(x) \leq 0,\ i = 1, \ldots, m',\ x \in X,$$
$$h_j(x) = 0,\ j = 1, \ldots, l',\ x \in X,$$

References for recent relaxation techniques: Adjiman and Floudas ('96)

I.P. Androulakis, C.D. Maranas, and C.A. Floudas ('95)

C. Jansson ('00, '01) C.D. Maranas and C.A. Floudas ('94)

Bernstein Polynomials

$B_i(x) = \binom{l}{i} x^i (1-x)^{l-i}, \quad 0 \le i \le l$

given: $p(x) = \sum_{i=0}^{l} a_i x^i$

wanted: $p(I) = \{p(x) \mid x \in I\}$ w.l.o.g. $I = [0,1]^n$

power form $\longrightarrow$ Bernstein form

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

$$b_i = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} a_j, \quad 0 \le i \le l$$ *Bernstein coefficients*

in particular, $b_0 = a_0 = p(0), \quad b_l = \sum_{i=0}^{l} a_i = p(1)$

can be calculated economically by difference table method (similarly to the de Casteljau algorithm).

Range Enclosing Property

$$p(I) \subseteq B_p := \left[\min_{i=0}^{l} b_i, \max_{i=0}^{l} b_i \right]$$

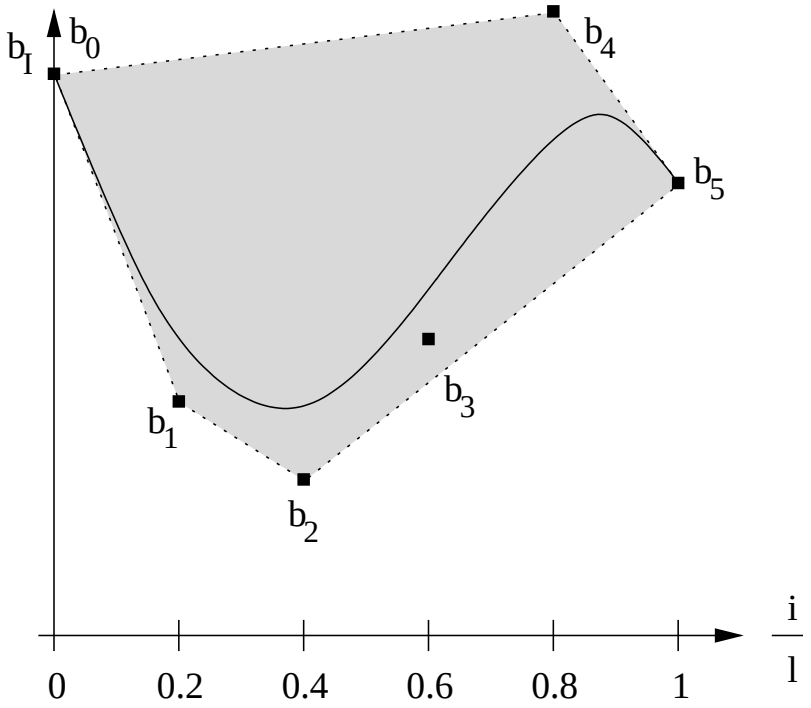
Bounds are sharp (i.e. no overestimation) if and only if the min/max is attained at b_0 or b_l.

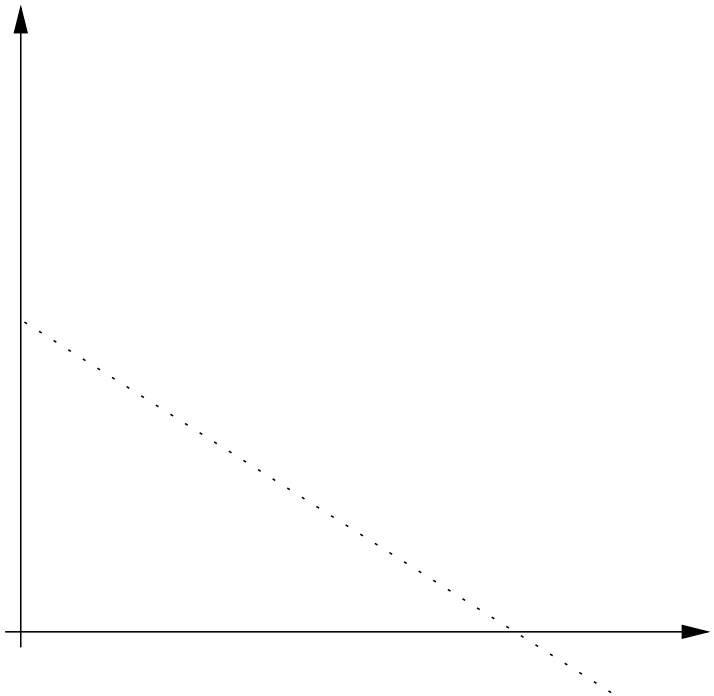
In fact,

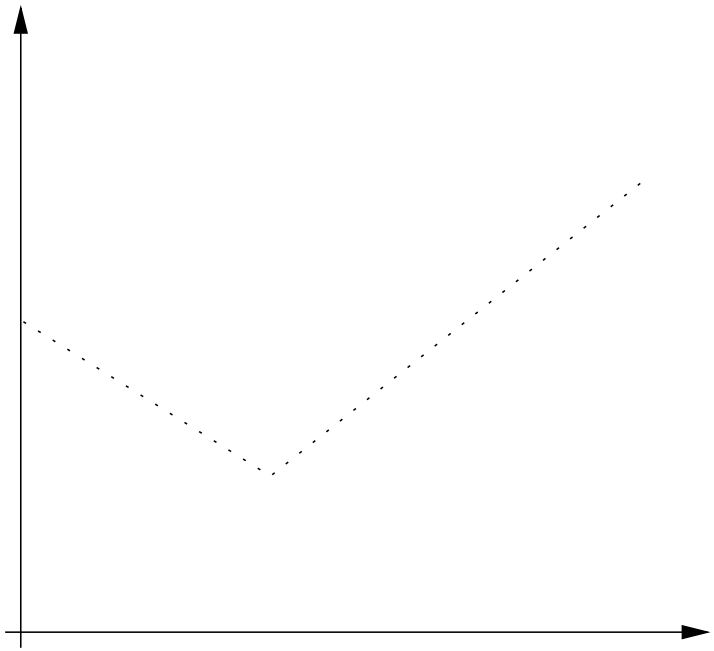
$$\begin{aligned} b_0 &= a_0 &&= p(0) \\ b_l &= \sum_{i=0}^{l} a_i &&= p(1) \end{aligned}$$

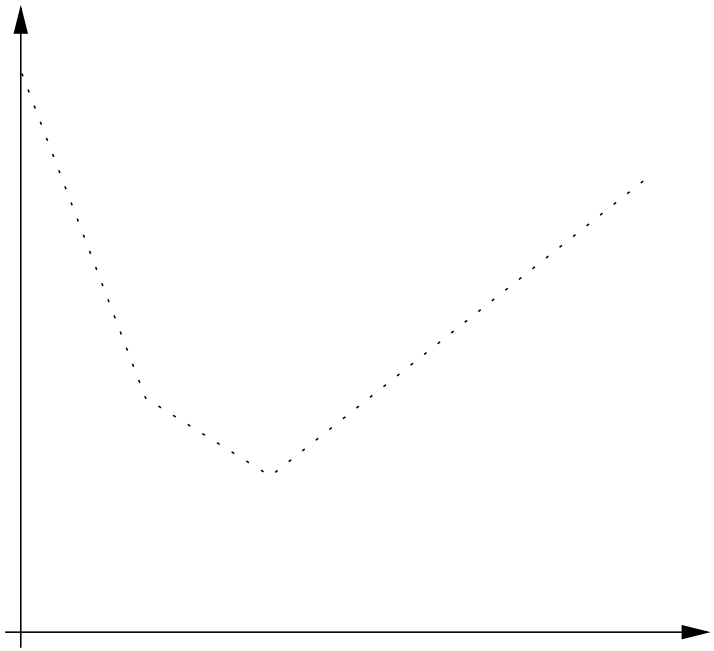
Convex Hull Property

$$\left\{ \binom{x}{p(x)} \;:\; x \in I \right\} \quad \subseteq \quad conv \left\{ \binom{i/l}{b_i} \;:\; 0 \leq i \leq l \right\}.$$









Notations

Multiindices $i = (i_1, \ldots, i_n)^T$ are vectors, where the components are nonnegative integers. The vectors 0 and 1 denote the multiindices with all components equal to 0 or 1, resp.
Comparisons and the absolute value $|.|$ are used entrywise.
The division of multiindices i, l with $l > 0$ is defined componentwise as $i/l := (i_1/l_1, \ldots, i_n/l_n)^T$. For $x \in \mathbf{R}^n$ its multipowers are $x^i := \prod_{\mu=1}^{n} x_\mu^{i_\mu}$.

We use the notations $\sum_{i=0}^{l} := \sum_{i_1=0}^{l_1} \ldots \sum_{i_n=0}^{l_n}$ and $\binom{l}{i} := \prod_{\mu=1}^{n} \binom{l_\mu}{i_\mu}$.

Method 1 - Constant bound function

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

Then

$$c_0(x) \leq p(x) \quad \forall x \in I$$

Method 2 (our previous approach) computes a bound function pasing through b_{i^0} and relies on the computation of gradients and the solution of an LP problem.

Garloff, Jansson, Smith ('03)

Method 3 - Algorithm, First Iteration:

Let $u^1 = (1, 0, \ldots, 0)^T$.

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 $b^1 = \left(\frac{i^1}{l}, b_{i^1}\right)^T$ and the vector $w^1 = \frac{i^1 - i^0}{l}$. Define the lower bound function

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

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

Let $\tilde{u}^j = (\beta_1^j, \ldots, \beta_{j-1}^j, 1, 0, \ldots, 0)^T$
such that $\tilde{u}^j \cdot w^k = 0$, $k = 1, \ldots, j-1$. Normalise $\tilde{u}^j$.

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

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

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

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

For the n iterations of the 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 a single $(n-1) \times (n-1)$ matrix. 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 in all iterations requires at most $n^2 L^n + L^n O(n)$ arithmetic operations.

Theorem 1 *We have that*

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

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.

Theorem 2 *The affine lower bound function c_n satisfies the a posteriori error bound*

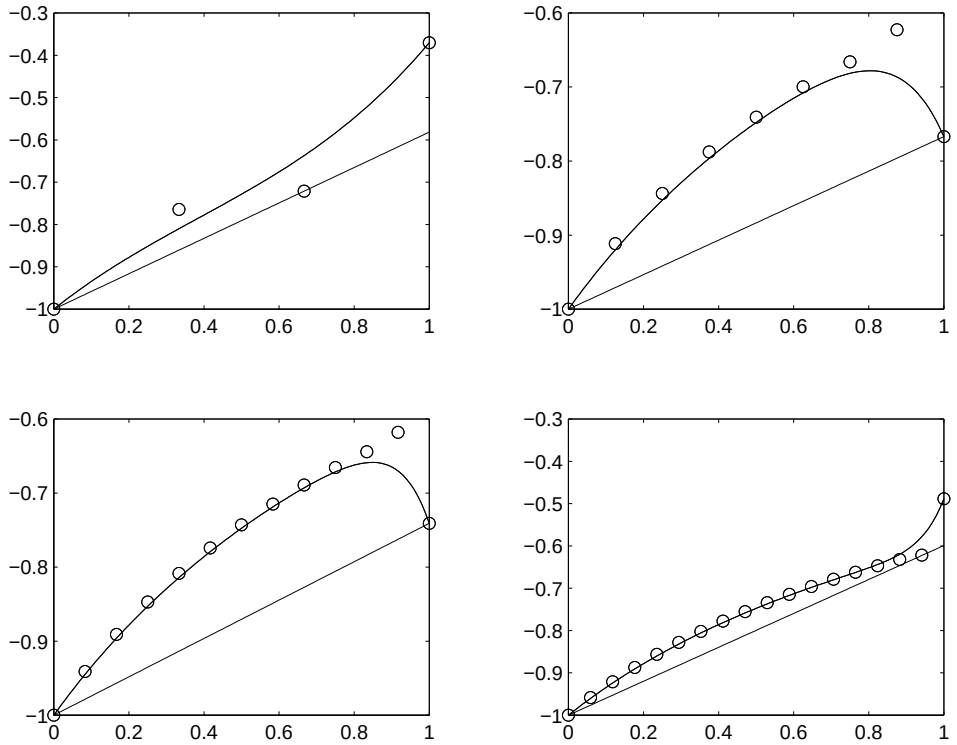
$$0 \leq p(x) - c_n(x) \leq \max\left\{ b_i - c_n\left(\frac{i}{l}\right) \ : \ 0 \leq i \leq l \right\}, \ x \in I,$$

which specifies in the univariate case to ($x \in I$)

$$0 \leq p(x) - c_1(x) \leq \max\left\{ \left(\frac{b_i - b^0}{i - i^0} - \frac{b^1 - b^0}{i^1 - i^0} \right)(i - i^0) : \ \ 0 \leq i \leq l, \ i \neq i^0 \right\}.$$

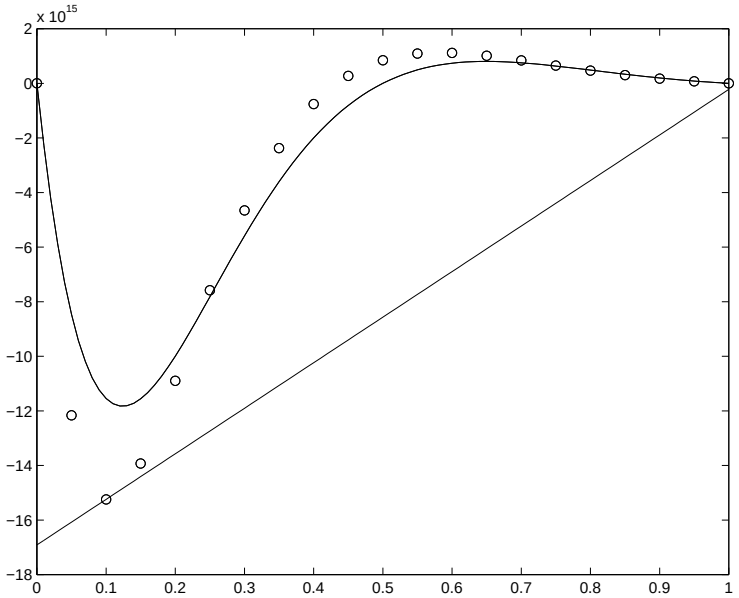
Remark: If we extend the construction of affine lower bound functions in the univariate case from I to arbitrary intervals $[\underline{a}, \overline{a}]$ with $\underline{a} < \overline{a}$, then we can show that this error bound is quadratic w.r.t. the width of the interval , i.e., the right-hand side can be bounded from above by $C(p)(\overline{a} - \underline{a})^2$, where $C(p)$ is an constant depending only on p. The question whether quadratic convergence holds true in the multivariate case is open, but seems likely also to hold.

Example 1: For $l = 3, 8, 13, 17$: $p(x) = \sum_{i=0}^{l} \frac{(-1)^{i+1}}{i+1} x^i, \quad x \in [0, 1]$



l	3	8	12	17
error	0.2113	0,1735	0.1446	0.1103

Example 2: $p(x) = \prod_{i=1}^{20} (x - i), \quad x \in [1, 3]$



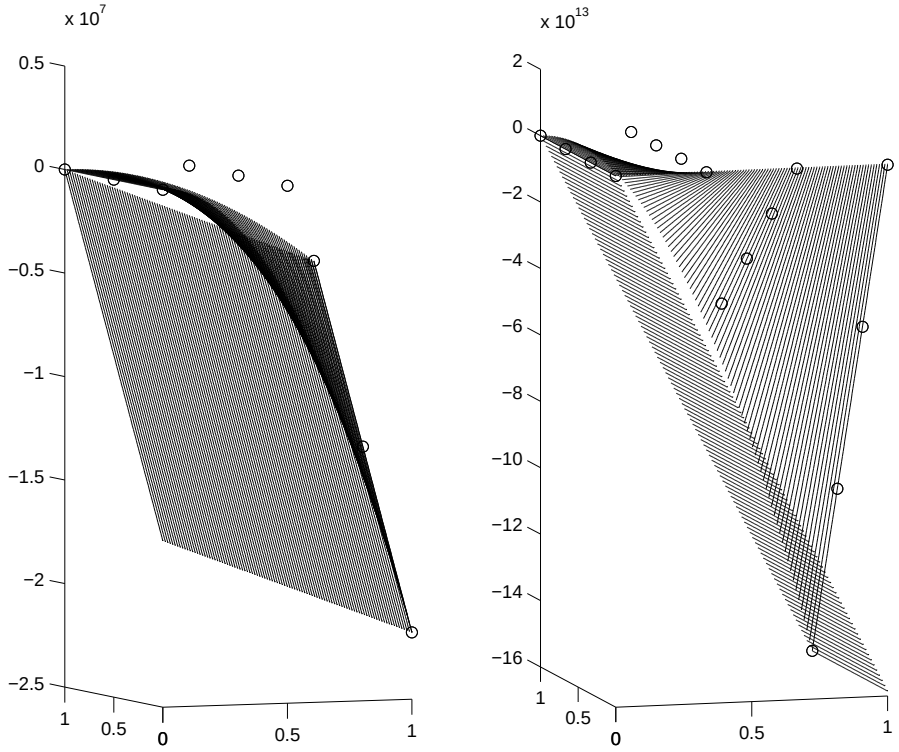
The error bound is 1.6912×10^{16}.

Example 3:

$$
\begin{aligned}
p_1(x_1, x_2) &= \alpha_1 x_1^2 x_2 + \alpha_2 x_1^2 + \alpha_3 x_1 x_2 + \alpha_4 x_1 + \alpha_5 x_2, \\
p_2(x_1, x_2) &= \alpha_6 x_1^2 x_2 + \alpha_7 x_1 x_2^2 + \alpha_8 x_1 x_2 + \alpha_9 x_2^3 + \alpha_{10} x_2^2 + \alpha_{11} x_2 + \alpha_{12},
\end{aligned}
$$

where

$$
\begin{aligned}
\alpha_1 &= -1.697 \times 10^7 & \alpha_7 &= 4.126 \times 10^7 \\
\alpha_2 &= 2.177 \times 10^7 & \alpha_8 &= -8.285 \times 10^6 \\
\alpha_3 &= 0.5500 & \alpha_9 &= 2.284 \times 10^7 \\
\alpha_4 &= 0.4500 \times 10^7 & \alpha_{10} &= 1.918 \times 10^7 \\
\alpha_5 &= -1.0000 \times 10^7 & \alpha_{11} &= 48.40 \\
\alpha_6 &= 1.585 \times 10^{14} & \alpha_{12} &= -27.73
\end{aligned}
$$



The error bounds for p_1 and p_2 are 1.937×10^7 and 1.585×10^4, respectively.

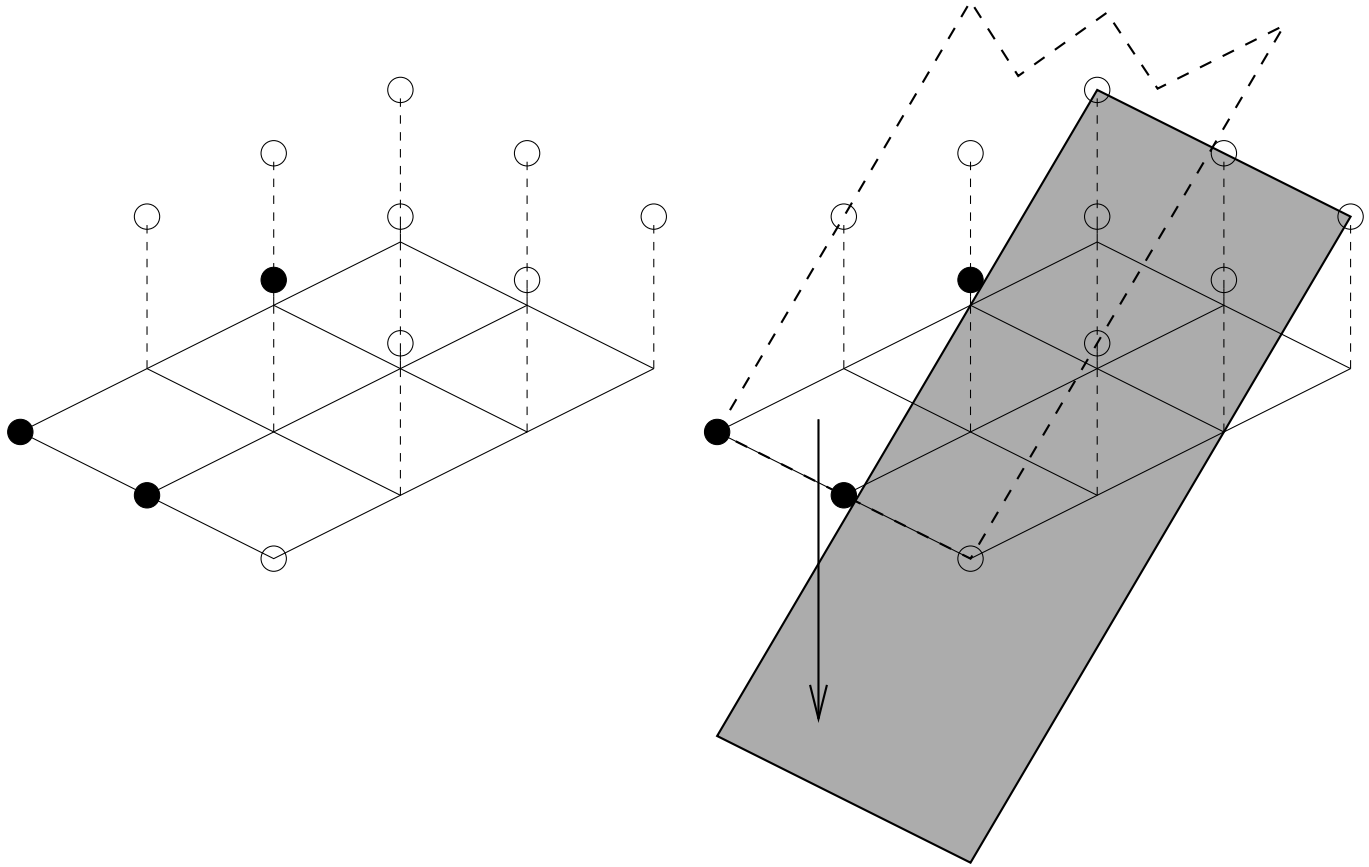
Method 4 - Designate control points corresponding to the $n+1$ smallest$*$ Bernstein coefficients.

Method 5 - Designate the control point corresponding to the minimum Bernstein coefficient and n others which connect to it with minimum$*$ absolute value of gradient.

In either case, a lower bound function is obtained by interpolating the designated control points (solution of a linear system). This bound function may be invalid, in which case it is corrected by the computation of an error term followed by a downward shift.

($*$ excluding degenerate cases)

Method 4 - Example of poor lower bound function



Method 5 - Example of poor lower bound function

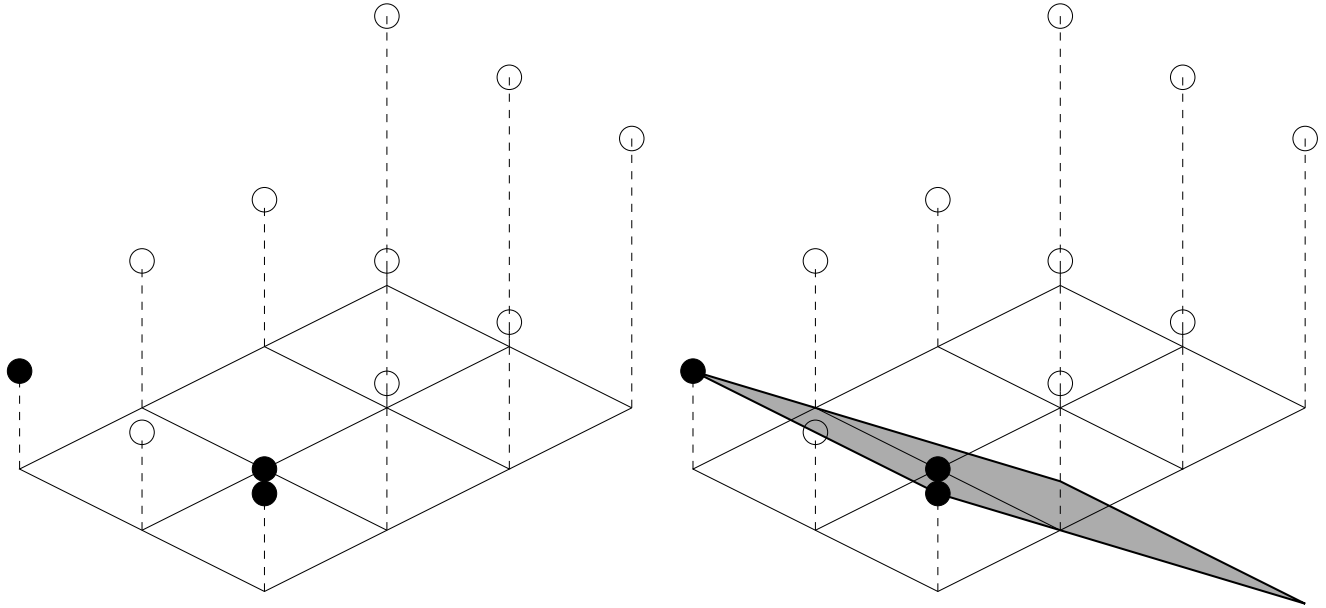


Table 1. Results for random polynomials

Method				**1** (Constant)		**2** (LP problems)		**3** (Linear eqs)	
n	D	k	$(D+1)^n$	time (s)	δ	time (s)	δ	time (s)	δ
2	2	5	9	0.000040	1.414	0.00020	0.976	0.000069	0.981
2	6	10	49	0.00013	1.989	0.0025	1.695	0.00031	1.677
2	10	20	121	0.00039	2.867	0.023	2.543	0.00074	2.511
4	2	20	81	0.00037	3.459	0.0082	2.847	0.0012	2.797
4	4	50	625	0.0024	5.678	2.82	5.056	0.0093	5.045
6	2	20	729	0.0011	4.043	4.48	3.403	0.016	3.353
8	2	50	6561	0.0093	6.941	greater than		0.24	6.291
10	2	50	59049	0.091	7.143	1 minute		3.43	6.503
				4 (min BCs)		**5** (min gradients)			
2	2	5	9	0.000085	1.147	0.00011	0.961		
2	6	10	49	0.00031	4.914	0.00044	1.910		
2	10	20	121	0.00090	11.49	0.0012	3.014		
4	2	20	81	0.0012	4.797	0.0015	3.199		
4	4	50	625	0.0088	14.05	0.011	5.940		
6	2	20	729	0.015	5.921	0.017	3.687		
8	2	50	6561	0.21	14.33	0.24	7.360		
10	2	50	59049	2.69	17.11	3.11	7.680		

Comparison of Methods - Summary

- Method 1 is the fastest, but constant bound functions are crude, with no shape information and a mediocre error bound.
- Method 2 is too slow, requiring the solution of LP problems.
- Method 3 is recommended, with the best error bound.
- Methods 4 and 5 are unreliable, giving extremely poor bound functions in some cases.
- The best method for any given polynomial may vary.

Future Work

- Verified bound functions can be obtained, either by using interval arithmetic, or by the computation of error terms and adjustment.
- Bound functions for derivatives can also be computed - first- and second-order information is easily obtainable from the Bernstein coefficients.
- Taylor expansion can be used to compute bound functions for problems involving general functions. A high-degree Taylor polynomial is computed (introduction of higher degree polynomial terms is not problematic for our approach), for which Bernstein coefficients and a bound function can be computed. The remainder of the Taylor expansion can be enclosed in an interval by interval computation methods. Subtracting this interval from the lower bound function of the Taylor polynomial provides the lower bound function for the given function.

Conclusions

- Bernstein expansion can deliver tight affine bounding functions for polynomials which may be of use in global optimisation or constraint programming.

- A package for the computation of bound functions for polynomials based on Bernstein expansion will shortly be contributed to the COCONUT project.

- For more information and papers, please see
  `http://www-home.fh-konstanz.de/~garloff/`

- Thank you for your attention!