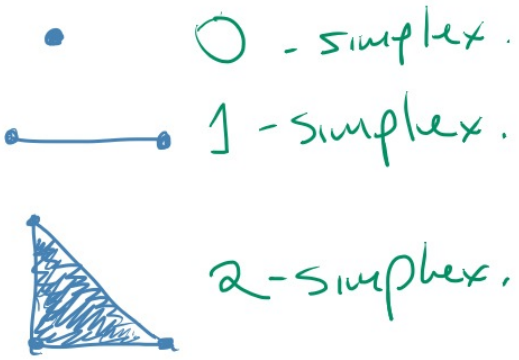
university of groningen

faculty of mathematics and natural sciences

Detecting positivity of multivariable polynomials on the standard simplex using the the Bernstein-Beziér representation

René te Velde



Master Thesis in Applied Mathematics

August 2011

Detecting positivity of multi-variable polynomials on the standard simplex using the the Bernstein-Beziér representation

Summary

One can read about algorithmic copositivity detection by simplicial partition in a paper of Stefan Bundfuss and Mirjam Dür. Copositivity detection is a quadratic problem (for quadratic polynomials). In this paper we did a similar approach, but for general multi-index polynomials instead of quadratic polynomials. Herefore, we use multi-index notations, barycentric coordinates, the Bernstein-Beziér representation of polynomials and simplicial partition. In this thesis, one can read about al the theory behind Bernstein-Beziér with respect to the positivity of polynomials on the standard simplex. Furthermore, we made several algorithms for low degree polynomials with one and two variables. We concluded that the algorithm works well except for some cases that the local minimum of a polynomial is exactly zero. The last conclusion is that the use of the Bernstein-Beziér form for positivity evaluations certainly has potential, but describing a complex polynomial with Bernstein-coefficients is rather hard and complicated.

Master Thesis in Applied Mathematics
Author: René te Velde
Supervisor: Dr. M. Dür
Supervisor 2: Prof. dr. H. L. Trentelman
Date: August 2011

Institute of Mathematics and Computing Science
P.O. Box 407
9700 AK Groningen
The Netherlands

Contents

Chapter 1

Introduction

The object we study in this thesis is the positivity of real multivariate polynomials. Since the late 19th century, the problem of detecting if a polynomial is positive or not on $\mathbb{R}^n$ has been researched. It is a very difficult problem if you want to detect the positivity of a complex polynomial. One could look for criteria which ensure the positivity of a polynomial. For example: a sum of squares of real polynomials with k variables ($k \geq 1$) is positive on $\mathbb{R}^n$. Also, other criteria exist, ensuring the positivity of a polynomial on a subset of $\mathbb{R}^n$. For example: if all the coefficients of a polynomial are positive, then the polynomial is positive on $\mathbb{R}^n_+$. The converse is false in general. These criteria are all very simple and basic, but they give an idea of where we are dealing with.

The primary objective of this thesis is to study the positivity of a polynomial on a standard simplex of $\mathbb{R}^n$. For this, we use a basis of $\mathbb{R}[X_1, ..., X_n]$ which is most suitable as a basis of monomials for this problem: the Bernstein basis. A monomial is a product of positive integer powers of a fixed set of variables (possibly) together with a coefficient, for example: x, $3x_1x_2^2$ or $-2x_1^2x_2^3x_3$. In the same way as before, if the Bernstein-coefficients of a polynomial with respect to the Bernstein basis are positive, then the polynomial is positive on the considered simplex. The general idea of this thesis is that if that not all the Bernstein-coefficients are positive, we can subdivide a simplex several times to detect whether or not a polynomial is positive on a certain simplex. It is true that if the Bernstein-coefficients are positive on the subdivision simplices, then the polynomial is positive on the bigger simplex.

Furthermore, Bernstein polynomials form a basis and the Bernstein coefficients contain lots of information about the geometry of a polynomial. We will also take a look at this in this thesis.

In chapter 2 we will take a look at the theory behind our research. We start with the quadratic form of polynomials in 2.1 which brought the idea for this thesis. After that we define a general polynomial and show the notation behind it in 2.2. In 2.3 we show the basis of our approach and we show what we mean by a simplex

in 2.4. In 2.5 we start by explaining barycentric coordinates which we use in 2.5 and 2.6 which show what the Bernstein-Bezir representation of a polynomial is and what the coefficients of such a polynomial look like. In 2.7 we show how a transformation of one simplex to another looks like and 2.8 gives further conditions for positivity detection. In chapter 3 one can see how our algorithms look like. The results are stated in chapter 4, the conclusions in chapter 5 and the code of our programs in chapter 6. We wrote the code in matlab.

Chapter 2

Theory

2.1 The quadratic form

In [1], one can read about algorithmic copositivity detection by simplicial partition. In this paper, (strict) copositivity means the following:

Let $\mathscr{S}$ denote the set of symmetric matrices in $\mathbb{R}^{n\times n}$ and let $\mathbb{R}^n_+$ denote the non-negative orthant.

A matrix $A \in \mathscr{S}$ is called copositive if

$$x^T Ax \geq 0 \; \forall \, x \in \mathbb{R}^n_+ \tag{2.1}$$

A matrix $A \in \mathscr{S}$ is called strictly copositive if

$$x^T Ax > 0 \; \forall \, x \in \mathbb{R}^n_+ \setminus \{0\} \tag{2.2}$$

Here, $q(x) = x^T Ax$ stands for the following n-ary quadratic form:

$$q(x_1,...,x_n) = \sum_{i,j=1}^{n} a_{ij} x_i x_j, \quad a_{ij} \in \mathbb{R}$$

in which $x = (x_1,...,x_n)^T$ is a column vector and $A = (a_{ij})$ is the $n \times n$ matrix over the field $\mathbb{R}$ whose entries are the coefficients of q. This n-ary quadratic form over $\mathbb{R}$ is a homogeneous polynomial of total degree 2. It has n variables with coefficients in $\mathbb{R}$.

In this thesis, we want to do a similar approach as in [1], not for the quadratic form, but for a general multi-index polynomial. Herefore, we have to define such a polynomial.

2.2 The general polynomial

We are going to define the general polynomial in the same way as in [2]. We start with defining $\mathbb{N} := \{0, 1, ...\}$ and for compactness, we define an n-dimensional multi-index β as an ordered n-tuple: $\beta = (\beta_1, ..., \beta_n)$ with $\beta \in \mathbb{N}^n$ i.e. $\beta_i \in \mathbb{N} \ \forall \ i \in (1, ..., n)$ (β_i is a nonnegative integer). For example, we will use multi-indices to shorten power products: for $x = (x_1, ..., x_n) \in \mathbb{R}^n$ one defines:

$$x^\beta = x_1^{\beta_1} \cdots x_n^{\beta_n} \tag{2.3}$$

Further adopting the standard multi-index notations we write:

$$|\beta| = \sum_{i=0}^{n} \beta_i, \quad \beta! = \beta_1! \cdots \beta_n! \tag{2.4}$$

We can use $|\beta|$ to define the total degree of a polynomial:

$$d = \max\{|\beta| \mid c_\beta \neq 0\} \tag{2.5}$$

in which $c_\beta \in \mathbb{R}$ is a constant of a polynomial. Finally, with the help of (2.5), we can write a general n-variable polynomial $p_n(x)$ in the form

$$p_n(x) = \sum_{|\beta| \leq d} c_\beta x^\beta, \quad x \in \mathbb{R}^n \tag{2.6}$$

in which $p_n(x) \in \mathbb{R}[x] = \mathbb{R}[x_1, ..., x_n]$. In short: let $P_n(x)$ be the general n-variable polynomial set of polynomials $p_n(x)$ with total degree d from (2.5) i.e. $\{|\beta| \mid c_\beta \neq 0\} \leq d$. A polynomial in this set has terms like: $c_\beta x_1^{\beta_1} \cdots x_n^{\beta_n}$ with c_β a constant and $|\beta| \leq d$ if $c_\beta \neq 0$.

As an example throughout this chapter, we will take the following polynomial:

$$p(x_1, x_2) = x_1 + 2x_1x_2 + x_2^3 \tag{2.7}$$

So, in this case, $d = 3$. Now we are done with defining the general polynomial and we want to take a look at what is called copositivity when we were talking about the quadratic form (section 2.1). This is the basis of our approach.

2.3 The basis of our approach

In [1], the following observation was the basis of their approach.

Lemma 1. Let $||\cdot||$ denote any norm on $\mathbb{R}^n$. We have

$$(a) \qquad A \text{ is copositive} \Leftrightarrow x^T Ax \geq 0 \; \forall \, x \in \mathbb{R}^n_+ \qquad \text{with } ||x|| = 1.$$
$$(b) \qquad A \text{ is strictly copositive} \Leftrightarrow x^T Ax > 0 \; \forall \, x \in \mathbb{R}^n_+ \setminus \{0\} \qquad \text{with } ||x|| = 1.$$

Proof. [$\Rightarrow$] is obvious. [$\Leftarrow$]: Take $x \in \mathbb{R}^n_+$ with $||x|| \neq 1$. If $||x|| = 0$ then $x = 0$ and $x^T Ax = 0$. If $||x|| > 0$ then $\tilde{x} := \frac{x}{||x||}$ fullfills $||\tilde{x}|| = 1$, whence $x^T Ax = ||x||^2 \tilde{x}^T A\tilde{x}$, and the statement follows. □

Unfortunately, $p_n(x) = p_n(||x|| \cdot \tilde{x}) = ||x||^d p_n(\tilde{x})$ does not hold for a general polynomial, so we cannot show a similar proof as above. See the next example.

Example 1. We take our example polynomial (2.7): $p(x_1, x_2) = x_1 + 2x_1x_2 + x_2^3$. We will show: $\exists \, x : p(x_1, x_2) \neq ||x||^3 p(\tilde{x_1}, \tilde{x_2})$. We take $x = (2,2)$ It follows that $p(x) = p(2,2) = 2 + 8 + 8 = 18$ and with x we get $||x|| = \sqrt{2^2 + 2^2} = \sqrt{8}$ and $\tilde{x} = \frac{1}{\sqrt{8}}(2,2)$, then

$$\begin{aligned}
||x||^3 \cdot p(\tilde{x}) &= \left(\sqrt{8}\right)^3 \cdot \left(\frac{2}{\sqrt{8}} + 2 \cdot \frac{2}{\sqrt{8}} \cdot \frac{2}{\sqrt{8}} + \left(\frac{2}{\sqrt{8}}\right)^3 \right) \\
&= \left(\sqrt{8}\right)^3 \cdot \left(\frac{2}{\sqrt{8}} + \frac{8}{8} + \frac{8}{8 \cdot \sqrt{8}} \right) \\
&= \left(\sqrt{8}\right)^3 \cdot \left(\frac{8 \cdot 2 + 8 \cdot \sqrt{8} + 8}{8 \cdot \sqrt{8}} \right) \\
&= \left(\sqrt{8}\right)^3 \cdot \left(\frac{24 + 8 \cdot \sqrt{8}}{8 \cdot \sqrt{8}} \right) \\
&= \left(\sqrt{8}\right)^3 \cdot \left(\frac{3 + \sqrt{8}}{\sqrt{8}} \right) \\
&\neq 18
\end{aligned}$$

What we are going to do in this thesis is a little bit different from [1]. We only take a look at positivity for $||x|| \leq 1$. The conditions for a general polynomial $p_n(x)$ (comparing to the quadratic form (section 2.1), equations 2.1 and 2.2) are stated as follows:

$$p_n(x) \geq 0 \; \forall \, x \in \mathbb{R}^n_+ \qquad \text{with } ||x|| \leq 1 \qquad (2.8)$$
$$p_n(x) > 0 \; \forall \, x \in \mathbb{R}^n_+ \setminus \{0\} \qquad \text{with } ||x|| \leq 1 \qquad (2.9)$$

with $\mathbb{R}^n_+$ is still the nonnegative orthant.

We will denote the set of polynomials for which the first holds by $\mathscr{R}$ and we will denote the set of polynomials for which the second holds by $\mathscr{R}^+$. In short:

$$\mathscr{R} = \{p_n(x) \in P_n(x) | p_n(x) \geq 0 \; \forall \, x \in \mathbb{R}^n_+ \qquad \text{with } ||x|| \leq 1\}$$
$$\mathscr{R}^+ = \{p_n(x) \in P_n(x) | p_n(x) > 0 \; \forall \, x \in \mathbb{R}^n_+ \setminus \{0\} \qquad \text{with } ||x|| \leq 1\}$$

Clearly, our example polynomial (2.7) belongs to $\mathscr{R}$ and even to $\mathscr{R}^+$: if you fill in a positive number for x_1 and x_2 in the polynomial, you get a positive outcome because of all the '+'-signs.

For simplicity, we choose the 1-norm and we define the set $\Delta_S := \{x \in \mathbb{R}^n_+ | \; ||x||_1 \leq 1\} = \{x \in \mathbb{R}^n_+ | \forall i, \, x_i \geq 0 \text{ and } \sum_{i=1}^{n} x_i \leq 1\}$ as the so-called *standard simplex*, whose vertices are the vector $\vec{0}$ and the unit vectors $e_1, ..., e_n$.

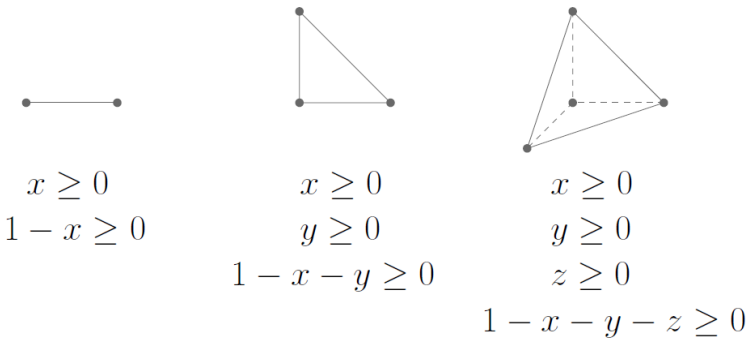


Figure 2.1: The standard simplex in $\mathbb{R}^1, \mathbb{R}^2$ and $\mathbb{R}^3$

With the help of the standard simplex, one can write for example

$$p_n(x) \geq 0 \; \forall \, x \in \Delta_S \text{ instead of } p_n(x) \geq 0 \; \forall \, x \in \mathbb{R}^n_+ \text{ with} ||x|| \leq 1$$

This means we search for conditions which ensure that that a polynomial $p_n(x)$ is nonnegative over a simplex. A convenient way to describe polynomials with respect to a simplex is to use barycentric coordinates and the Bernstein-Bézier representation, which we are going to do in the same way as in [3].

2.4 Barycentric coordinates

We will describe barycentric coordinates with the help of [4]. Let $p_n(x) \in P_n(x)$ be a polynomial with $x = (x_1, ..., x_n)^T$ and total degree d. Let us denote $\text{conv}\{A\}$ as the convex hull of the set A. We also have a basis non degenerate n-simplex

$$\Delta = \text{conv}\{v_0, ..., v_n\} \subset \mathbb{R}^n$$

with $n+1$ affinely independent known vertices $v_0, ..., v_n \in \mathbb{R}^n$. The dimension of an n-simplex is n. Given Δ, we can represent each point $x \in \mathbb{R}^n$ in the affine hull of Δ by its uniquely defined barycentric coordinates $\lambda = \lambda(x) = (\lambda_0, ..., \lambda_n)^T$ (λ_i are polynomials of degree 1) with respect to the simplex Δ:

$$x = \sum_{i=0}^{n} \lambda_i v_i \quad \text{with} \quad \sum_{i=0}^{n} \lambda_i = 1 \quad \text{and} \quad x \in \Delta \Leftrightarrow \forall i, \lambda_i \geq 0$$

One can see in figure 2.1 that you can write the barycentric coordinates with respect to the standard simplex (figure 2.1) as

$$\lambda_i = x_i \quad (i = 1, ..., n) \tag{2.10}$$

and because $\sum_{i=0}^{n} \lambda_i = 1$ we can write λ_0 as:

$$\lambda_0 = 1 - x_1 - ... - x_n \tag{2.11}$$

Barycentric coordinates are unique up to a permutation of the vertices of a simplex Γ. In n dimensions we can write a vertex v_i like $v_i = (v_{1i}, ..., v_{ni})$ (v_{ji} are the coordinates of the vertex v_i). We get

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} = P_\Gamma \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix} \tag{2.12}$$

in which the matrix P_Γ is the matrix

$$P_\Gamma = \begin{pmatrix} v_{10} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{n0} & \cdots & v_{nn} \\ 1 & \cdots & 1 \end{pmatrix} \tag{2.13}$$

A simplex Γ is nondegenerate, i.e. the vertices $v_0, ..., v_n$ are affinely independent and the matrix P_Γ is invertible. Because of the invertibility, one can write barycentric coordinates with respect to the simplex Γ out of the barycentric coordinates with respect to the simplex Δ in the following way:

$$\begin{pmatrix} \lambda_0^{[\Gamma]} \\ \vdots \\ \lambda_n^{[\Gamma]} \end{pmatrix} = P_\Gamma^{-1} P_\Delta \begin{pmatrix} \lambda_0^{[\Delta]} \\ \vdots \\ \lambda_n^{[\Delta]} \end{pmatrix} \tag{2.14}$$

2.5 The Bernstein-Bézier representation

In the sequel we will describe a polynomial by barycentric coordinates and the Bernstein-Bézier representation. With the help of the standard multi-index notations (2.3) and (2.4) the standard Bernstein polynomials of degree $d = |\alpha|$, with respect to the simplex Δ, are given by

$$B_\alpha^d(\Delta) = \frac{d!}{\alpha!}\lambda^\alpha \text{ with } \alpha \in \mathbb{N}^n \tag{2.15}$$

With the help of (2.10) and (2.11) and the multinomial theorem: $(x_1 + ... + x_m)^n = \sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha$, one can see the standard Bernstein polynomials appear naturally in the following expansion:

$$1 = 1^d = (\lambda_0 + ... + \lambda_n)^d = \sum_{|\alpha|=d} \frac{d!}{\alpha!}\lambda^\alpha = \sum_{|\alpha|=d} B_\alpha^d(\Delta)$$

The Bernstein polynomials form a basis of all multi-index polynomials of degree d (basis of $\mathbb{R}_d[x]$) and they are nonnegative on Δ. Each polynomial $p_n(x)$ has a **unique** Bernstein-Bézier (BB) representation with respect to a simplex Δ:

$$p_n(x) = \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) B_\alpha^d(\Delta) \tag{2.16}$$

The coefficients $b_\alpha(p,d,\Delta) \in \mathbb{R}$ are the BB coefficients of the polynomial p with respect to a basis simplex Δ. We are going to analyze these BB coefficients with respect to the standard simplex (figure 2.1) in the same way as in [4].

2.6 The BB coefficients

In this section, we are going to write a general polynomial (2.6) in the Bernstein-Bézier form (2.16) to see how one can get the BB coefficients. First, we write 1 in Bernstein-Bézier form by using (2.10) and (2.11):

$$1 = \sum_{i=1}^n x_i + (1 - \sum_{i=1}^n x_1) = \sum_{i=1}^n \lambda_i + \lambda_0 = \sum_{i=0}^n \lambda_i \tag{2.17}$$

Now we start with our general polynomial:

$$\begin{aligned} p_n(x) &= \sum_{|\beta|\le d} c_\beta x^\beta \\ p_n(x) &= \sum_{|\beta|\le d} c_\beta x_1^{\beta_1} \cdots x_n^{\beta_n} \\ p_n(x) &= \sum_{|\beta|\le d} c_\beta x_1^{\beta_1} \cdots x_n^{\beta_n} \cdot 1^{d-|\beta|} \end{aligned}$$

and finally, with the help of (2.10) and (2.17), we can write $p_n(x)$ as

$$p_n(x) = \sum_{|\beta|\leq d} c_\beta \lambda_1^{\beta_1} \cdots \lambda_n^{\beta_n} \cdot (\sum_{i=0}^{k} \lambda_i)^{d-|\beta|} \tag{2.18}$$

Next, we are going to show three examples of getting the BB coefficients. In the first one, we show that there are special cases of polynomials that you can write in Bernstein-Bézier form very fast. This is an extended version of example 1.7 out of [4].

Example 2. Let $n \geq 1$ and $f = (1-2x_1)^d \in \mathbb{R}[x_1,...,x_n]$. We want to write this as

$$f = \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) B_\alpha^d(\Delta)$$

Now, we are going to obtain f in Bernstein-Bézier form. We use (2.17), the same multinomial theorem as in paragraph 2.5 and (2.15):

$$\begin{aligned}
f &= (1-2x_1)^d \\
&= [(1-\sum_{i=1}^{n} x_i) + \sum_{i=1}^{n} x_i - 2x_1]^d \\
&= [(1-\sum_{i=1}^{n} x_i) - x_1 + \sum_{i=2}^{n} x_i]^d \\
&= \sum_{|\alpha|=d} \frac{d!}{\alpha!} (1-\sum_{i=1}^{n} x_i)^{\alpha_0} (-x_1)^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \\
&= \sum_{|\alpha|=d} (-1)^{\alpha_1} \frac{d!}{\alpha!} \lambda_0^{\alpha_0} \lambda_1^{\alpha_1} \lambda_2^{\alpha_2} \cdots \lambda_n^{\alpha_n} \\
&= \sum_{|\alpha|=d} (-1)^{\alpha_1} B_\alpha^d(\Delta_S)
\end{aligned}$$

One can see that $\forall\ |\alpha| = d$ the BB coefficients are $b_\alpha(f,d,\Delta_S) = (-1)^{\alpha_1}$.

In the second example, we find the BB coefficients of a very simple polynomial.

Example 3. Let $f = 6x^2 - 6x + 2$. In this case, $d = 2$, $\lambda = (\lambda_0, \lambda_1)$ and $\Delta_S = [0,1]$ (see figure 2.1). Because of $d = 2$ and the two λ's, we have 3 α's: $\alpha 1 = (2,0)$, $\alpha 2 = (1,1)$ and $\alpha 3 = (0,2)$.

We want to write f in the following way:

$$f = \sum_{|\alpha|=d} B_\alpha^d(\Delta) b_\alpha(p,d,\Delta) = \sum_{|\alpha|=2} B_\alpha^2([0,1]) b_\alpha(f,2,[0,1]) \tag{2.19}$$

with $B_\alpha^2([0,1]) = \frac{2!}{\alpha!}\lambda^\alpha$.

The standard Bernstein polynomials are

$$\begin{aligned} B^2_{\alpha 1}([0,1]) &= 1\lambda_0^2 \\ B^2_{\alpha 2}([0,1]) &= 2\lambda_0^1\lambda_1^1 \\ B^2_{\alpha 3}([0,1]) &= 1\lambda_1^2 \end{aligned}$$

With (2.18) we write

$$f = \sum_{|\beta|\leq 2} c_\beta \lambda_1^{\beta_1} (\sum_{i=0}^{1} \lambda_i)^{2-|\beta|}$$

In our case, $\beta 1 = \{2\}, \beta 2 = \{1\}$ and $\beta 3 = \{0\}$. Furthermore, $c_{\beta 1} = 6, c_{\beta 2} = -6$ and $c_{\beta 3} = 2$. Now, we get rid off the sum.

$$\begin{aligned} f &= \sum_{|\beta|\leq 2} c_\beta \lambda_1^{\beta_1} (\sum_{i=0}^{1} \lambda_i)^{2-|\beta|} \\ &= 6\lambda_1^2 (\sum_{i=0}^{1} \lambda_i)^0 - 6\lambda_1^1 (\sum_{i=0}^{1} \lambda_i)^1 + 2(\sum_{i=0}^{1} \lambda_i)^2 \\ &= 6\lambda_1^2 - 6\lambda_1^1(\lambda_0+\lambda_1) + 2(\lambda_0+\lambda_1)^2 \\ &= 6\lambda_1^2 - 6\lambda_0\lambda_1 - 6\lambda_1^2 + 2\lambda_0^2 + 4\lambda_0\lambda_1 + 2\lambda_1^2 \\ &= 2\lambda_0^2 - 2\lambda_0\lambda_1 + 2\lambda_1^2 \\ &= 2B^2_{\alpha 1}([0,1]) - 1B^2_{\alpha 2}([0,1]) + 2B^2_{\alpha 3}([0,1]) \end{aligned}$$

So, the BB coefficients have to be $b(f,2,[0,1]) = (2,-1,2)$ to make equation (2.19) right.

In the third example, we find the BB coefficients of our example polynomial (2.7). We do this in the same way as in example 3.

Example 4. Let $p(x_1,x_2) = x_1 + 2x_1x_2 + x_2^3$. In this case, $d = 3$. $\lambda = (\lambda_0,\lambda_1,\lambda_2)$ and we use the standard simplex Δ_S (see figure 2.1). Because of $d = 3$ and the three λ's, we have 10 α's: $\alpha 1 = (3,0,0)$, $\alpha 2 = (0,3,0)$, $\alpha 3 = (0,0,3)$, $\alpha 4 = (2,1,0)$, $\alpha 5 = (2,0,1)$, $\alpha 6 = (1,2,0)$, $\alpha 7 = (0,2,1)$, $\alpha 8 = (1,0,2)$, $\alpha 9 = (0,1,2)$ and $\alpha 10 = (1,1,1)$.

We want to write $p(x_1,x_2)$ in the following way:

$$p(x_1,x_2) = \sum_{|\alpha|=d} B^d_\alpha(\Delta) b_\alpha(p,d,\Delta) = \sum_{|\alpha|=3} B^3_\alpha(\Delta_S) b_\alpha(p,3,\Delta_S) \tag{2.20}$$

with $B^3_\alpha(\Delta_S) = \frac{3!}{\alpha!}\lambda^\alpha$.

The standard Bernstein polynomials are

$$\begin{aligned}
B^3_{\alpha 1}(\Delta_S) &= 1\lambda_0^3 \\
B^3_{\alpha 2}(\Delta_S) &= 1\lambda_1^3 \\
B^3_{\alpha 3}(\Delta_S) &= 1\lambda_2^3 \\
B^3_{\alpha 4}(\Delta_S) &= 3\lambda_0^2\lambda_1^1 \\
B^3_{\alpha 5}(\Delta_S) &= 3\lambda_0^2\lambda_2^1 \\
B^3_{\alpha 6}(\Delta_S) &= 3\lambda_1^2\lambda_0^1 \\
B^3_{\alpha 7}(\Delta_S) &= 3\lambda_1^2\lambda_2^1 \\
B^3_{\alpha 8}(\Delta_S) &= 3\lambda_2^2\lambda_0^1 \\
B^3_{\alpha 9}(\Delta_S) &= 3\lambda_2^2\lambda_1^1 \\
B^3_{\alpha 10}(\Delta_S) &= 6\lambda_0^1\lambda_1^1\lambda_2^1
\end{aligned}$$

With (2.18) we write

$$p(x_1,x_2) = \sum_{|\beta|\leq 3} c_\beta \lambda_1^{\beta_1}\lambda_2^{\beta_2}(\sum_{i=0}^{2}\lambda_i)^{3-|\beta|}$$

In our case, $\beta 1 = \{1,0\}, \beta 2 = \{1,1\}$ and $\beta 3 = \{0,3\}$. Furthermore, $c_{\beta 1} = 1, c_{\beta 2} = 2$ and $c_{\beta 3} = 1$. Now, we get rid off the sum.

$$\begin{aligned}
p(x_1,x_2) &= \sum_{|\beta|\leq 3} c_\beta \lambda^\beta (\sum_{i=0}^{2}\lambda_i)^{3-|\beta|} \\
&= 1\lambda_1^1(\sum_{i=0}^{2}\lambda_i)^2 + 2\lambda_1^1\lambda_2^1(\sum_{i=0}^{2}\lambda_i)^1 + 1\lambda_2^3(\sum_{i=0}^{2}\lambda_i)^0 \\
&= \lambda_1(\lambda_0+\lambda_1+\lambda_2)^2 + 2\lambda_1\lambda_2(\lambda_0+\lambda_1+\lambda_2) + \lambda_2^3 \\
&= \lambda_1(\lambda_0^2+2\lambda_0\lambda_1+2\lambda_0\lambda_2+\lambda_1^2+2\lambda_1\lambda_2+\lambda_2^2) + 2\lambda_1\lambda_2(\lambda_0+\lambda_1+\lambda_2) + \lambda_2^3 \\
&= \lambda_0^2\lambda_1 + 2\lambda_0\lambda_1^2 + 2\lambda_0\lambda_1\lambda_2 + \lambda_1^3 + 2\lambda_1^2\lambda_2 + \lambda_1\lambda_2^2 + 2\lambda_0\lambda_1\lambda_2 + 2\lambda_1^2\lambda_2 + 2\lambda_1\lambda_2^2 + \lambda_2^3 \\
&= 0\lambda_0^3 + 1\lambda_1^3 + 1\lambda_2^3 + 1\lambda_0^2\lambda_1 + 0\lambda_0^2\lambda_2 + 2\lambda_1^2\lambda_0 + 4\lambda_1^2\lambda_2 + 0\lambda_2^2\lambda_0 + 3\lambda_2^2\lambda_1 + 4\lambda_0\lambda_1\lambda_2 \\
&= 0B^3_{\alpha 1} + 1B^3_{\alpha 2} + 1B^3_{\alpha 3} + \tfrac{1}{3}B^3_{\alpha 4} + 0B^3_{\alpha 5} + \tfrac{2}{3}B^3_{\alpha 6} + \tfrac{4}{3}B^3_{\alpha 7} + 0B^3_{\alpha 8} + 1B^3_{\alpha 9} + \tfrac{2}{3}B^3_{\alpha 10}
\end{aligned}$$

So, the BB coefficients have to be $b(p,3,\Delta_S) = (0,1,1,\frac{1}{3},0,\frac{2}{3},\frac{4}{3},0,1,\frac{2}{3})$ to make equation (2.20) right.

The barycentric coordinates and the Bernstein-Bézier representation give rise to easily verifiable sufficient conditions for our problem (2.8). We will describe these conditions in paragraph 2.8. First, we are going to describe how a transformation from one simplex to another simplex (see paragraph 2.4) works.

2.7 The transformation of one simplex to another ($\Delta \to \Gamma$)

We start with a polynomial in Bernstéin Bezier form with respect to a simplex Δ. Such a polynomial looks as follows (with the help of 2.3, 2.15 and 2.16):

$$p_n(x) = \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) B_\alpha^d(\Delta) = \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) \frac{d!}{\alpha!} \lambda_0^{\alpha_0} \cdots \lambda_n^{\alpha_n}$$

Now we introduce a new simplex Γ. The coordinates with respect to the simplex Δ were $\lambda_0, ..., \lambda_n$ and the new coordinates with respect to the simplex Γ are $\gamma_0, ..., \gamma_n$. We already know from (2.14) that

$$\begin{pmatrix} \gamma_0 \\ \vdots \\ \gamma_n \end{pmatrix} = P_\Gamma^{-1} P_\Delta \begin{pmatrix} \lambda_0 \\ \vdots \\ \lambda_n \end{pmatrix}$$

This means that $\lambda = P_\Delta^{-1} P_\Gamma \gamma$. From now on, we define $P_\Delta^{-1} P_\Gamma =: T$ such that we get $\lambda_i = \sum_{j=0}^{n} T_{ij} \gamma_j$. We can replace the λ_i's in the equation for our polynomial:

$$p_n(x) = \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) \frac{d!}{\alpha!} (\sum_{j=0}^{n} T_{0j}\gamma_j)^{\alpha_0} \cdots (\sum_{j=0}^{n} T_{nj}\gamma_j)^{\alpha_n} \tag{2.21}$$

And finally, this equation has to become

$$p_n(x) = \sum_{|\alpha|=d} b_\alpha(p,d,\Gamma) \frac{d!}{\alpha!} \gamma_0^{\alpha_0} \cdots \gamma_n^{\alpha_n} \tag{2.22}$$

which shows that in theory it is possible to calculate Bernstein coefficients with respect to the new simplex out of the Bernstein coefficients with respect to the old one, but in practice it is very difficult if you have to deal with complicated polynomials. Example 8 in paragraph 2.8 is an example of a transformation of one simplex to another one.

Barycentric coordinates and the Bernstein-Bézier representation give rise to sufficient conditions for our problem (2.8).

2.8 Further conditions for our problem (2.8)

Lemma 5. *Let $\Delta = conv\{v_1, ..., v_n\}$ be a simplex and let $p_n(x)$ be a polynomial as defined in (2.6). If*

$$b_\alpha(p, d, \Delta) \geq 0$$

(resp. > 0), then $p_n(x) \geq 0$ (resp. > 0) $\forall\ x \in \Delta$.

Proof. According to (2.16), we can write a polynomial in the form

$$p_n(x) = \sum_{|\alpha|=d} b_\alpha(p, d, \Delta) B_\alpha^d(\Delta)$$

with $B_\alpha^d(\Delta) = \frac{d!}{\alpha!}\lambda^\alpha$. For $x \in \Delta$, we have $\lambda(x) \geq 0$, whence $B_\alpha^d(\Delta) \geq 0$, whence $p_n(x) \geq 0$ for all $x \in \Delta$ if $b_\alpha(p, d, \Delta) \geq 0$ for all $x \in \Delta$. □

With this lemma we cannot say anything about example 2. This is because there is always an α_1 which is an odd number (see examples 3 and 4). Our example polynomial is positive (example 4). **Warning:** the converse of this lemma is false in general! Take a look at example 3: $f = 6x^2 - 6x + 2 > 0$ on $[0, 1]$ (see figure 2.2), but $b(f, 2, [0, 1]) = [2, -1, 2]$!

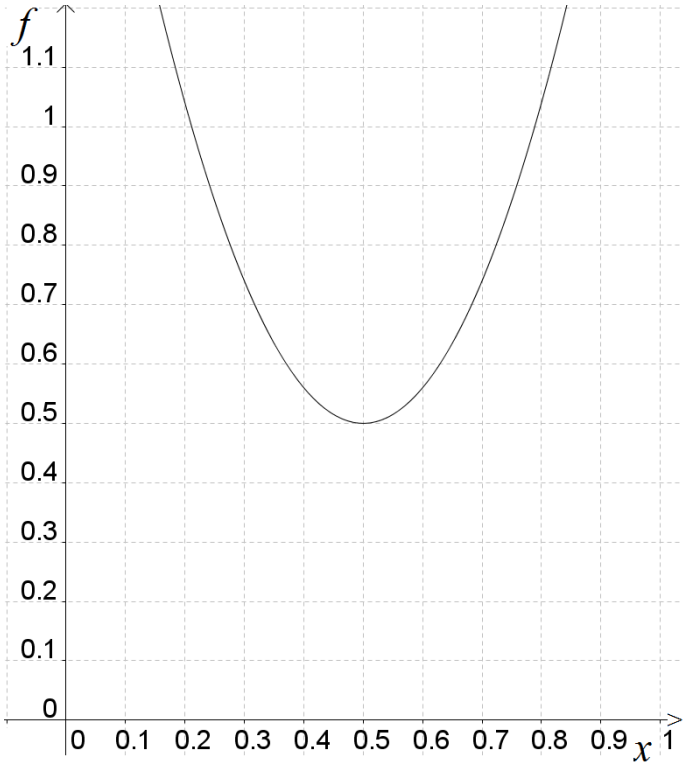


Figure 2.2: Graph of the function $f = 6x^2 - 6x + 2$ on $[0, 1]$.

Definition 6. Let Δ be a simplex in $\mathbb{R}^n$. A family $\mathscr{P} = \{\Delta_1, ..., \Delta_m\}$ of simplices satisfying

$$\Delta = \bigcup_{i=1}^{m} \Delta_i \quad \text{and} \quad \text{int}\,\Delta_i \cap \text{int}\,\Delta_j = \emptyset \quad \text{for } i \neq j$$

is called a *simplicial partition* of Δ.

Using this concept, the following theorem gives sufficient conditions for positivity which generalize the aforementioned relation of positivity with respect to a simplex.

Theorem 7. *Let $p_n(x) \in P_n(x)$, let $\mathscr{P} = \{\Delta_1, ..., \Delta_m\}$ be a simplicial partition of Δ_S into m simplices and let $b_\alpha(p, d, \Delta_k)$ denote the BB coefficients with respect to the simplex Δ_k.*

$$(a)\ \textit{If } \forall\, k \textit{ we have } b_\alpha(p,d,\Delta_k) \geq 0, \textit{ then } p_n(x) \in \mathscr{R}$$
$$(b)\ \textit{If } \forall\, k \textit{ we have } b_\alpha(p,d,\Delta_k) > 0, \textit{ then } p_n(x) \in \mathscr{R}^+$$

Proof. We only prove (a). It is sufficient to prove nonnegativity of $p_n(x)$ for $x \in \Delta_S$. So choose an arbitrary $x \in \Delta_S$. Then $x \in \Delta_k$ for some $\Delta_k \in \mathscr{P}$. By assumption, $b_\alpha(p, d, \Delta_k) \geq 0$ which, by lemma 5, implies $p_n(x) \in \mathscr{R}$. □

We will show an example in which we proof the nonnegativity of the polynomial of example 3 with the help of the transformation of one simplex to another one and simplicial partition.

Example 8. We take the polynomial with 1 variable of example 3: $f : \mathbb{R} \to \mathbb{R} = 6x^2 - 6x + 2$. This polynomial has BB coefficients $b(f, 2, [0,1]) = (2, -1, 2)$ for $(\alpha 1, \alpha 2, \alpha 3) = ((2,0),(1,1),(0,2))$. We have to do simplicial partition steps in order to confirm the positiviness of f on [0, 1] with our theory. In 1 dimension, we have the standard simplex $\Delta_s = [0,1]$, see figure 2.3.

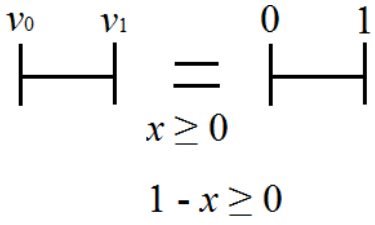


Figure 2.3: The standard simplex in 1 dimension

With the help of (2.12) and (2.13), we get the following equation:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} = \begin{pmatrix} v_{10} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{n0} & \cdots & v_{nn} \\ 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$$

In our case, this will be the following:

$$\begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} v_0 & v_1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \end{pmatrix}$$

This means that

$$P_{[0,1]} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \text{ and } P^{-1}_{[0,1]} = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$$

We are going to do a simplicial partition and take the simplex $[0, \frac{1}{2}]$. This means that

$$P_{[0,\frac{1}{2}]} = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & 1 \end{pmatrix} \text{ and } P^{-1}_{[0,\frac{1}{2}]} = \begin{pmatrix} -2 & 1 \\ 2 & 0 \end{pmatrix}$$

We calculate T out of paragraph 2.7 with

$$T = P^{-1}_{[0,1]} P_{[0,\frac{1}{2}]} = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}$$

Now we calculate with (2.21):

$$\begin{aligned} f(x) &= \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) \frac{d!}{\alpha!} (\sum_{j=0}^{n} T_{0j}\gamma_j)^{\alpha_0} \cdots (\sum_{j=0}^{n} T_{nj}\gamma_j)^{\alpha_n} \\ &= \sum_{|\alpha|=2} b_\alpha(f,2,[0,1]) \frac{2!}{\alpha!} (\sum_{j=0}^{1} T_{0j}\gamma_j)^{\alpha_0} (\sum_{j=0}^{1} T_{1j}\gamma_j)^{\alpha_1} \\ &= 2\cdot 1\cdot (1\gamma_0 + \frac{1}{2}\gamma_1)^2 \cdot (\frac{1}{2}\gamma_1)^0 - 1\cdot 2\cdot (1\gamma_0 + \frac{1}{2}\gamma_1)^1 \cdot (\frac{1}{2}\gamma_1)^1 + 2\cdot 1\cdot (1\gamma_0 + \frac{1}{2}\gamma_1)^0 \cdot (\frac{1}{2}\gamma_1)^2 \\ &= 2\gamma_0^2 + 2\gamma_0\gamma_1 + \frac{1}{2}\gamma_1^2 - \gamma_0\gamma_1 - \frac{1}{2}\gamma_1^2 + \frac{1}{2}\gamma_1^2 \\ &= 2\gamma_0^2 + \gamma_0\gamma_1 + \frac{1}{2}\gamma_1^2 \end{aligned}$$

With the help of (2.22), we get that $b(f,2,[0,\frac{1}{2}]) = (2, \frac{1}{2}, \frac{1}{2})$.

Now we take the other half of the standard simplex in 1 dimension: the simplex $[\frac{1}{2}, 1]$. This means that

$$P_{[\frac{1}{2},1]} = \begin{pmatrix} \frac{1}{2} & 1 \\ 1 & 1 \end{pmatrix} \text{ and } P^{-1}_{[\frac{1}{2},1]} = \begin{pmatrix} -2 & 2 \\ 2 & -1 \end{pmatrix}$$

We calculate T again with

$$T = P_{[0,1]}^{-1} P_{[\frac{1}{2},1]} = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 1 \end{pmatrix}$$

Now we calculate with (2.21):

$$\begin{aligned} f(x) &= \sum_{|\alpha|=d} b_\alpha(p,d,\Delta) \frac{d!}{\alpha!} (\sum_{j=0}^{n} T_{0j}\gamma_j)^{\alpha_0} \cdots (\sum_{j=0}^{n} T_{nj}\gamma_j)^{\alpha_n} \\ &= \sum_{|\alpha|=2} b_\alpha(f,2,[0,1]) \frac{2!}{\alpha!} (\sum_{j=0}^{1} T_{0j}\gamma_j)^{\alpha_0} (\sum_{j=0}^{1} T_{1j}\gamma_j)^{\alpha_1} \\ &= 2\cdot 1\cdot (\frac{1}{2}\gamma_0)^2 \cdot (\frac{1}{2}\gamma_0 + 1\gamma_1)^0 - 1\cdot 2\cdot (\frac{1}{2}\gamma_0)^1 \cdot (\frac{1}{2}\gamma_0 + 1\gamma_1)^1 + 2\cdot 1\cdot (\frac{1}{2}\gamma_0)^0 \cdot (\frac{1}{2}\gamma_0 + 1\gamma_1)^2 \\ &= \frac{1}{2}\gamma_0^2 - \frac{1}{2}\gamma_0^2 - \gamma_0\gamma_1 + \frac{1}{2}\gamma_0^2 + 2\gamma_0\gamma_1 + 2\gamma_1^2 \\ &= \frac{1}{2}\gamma_0^2 + 1\gamma_0\gamma_1 + 2\gamma_1^2 \end{aligned}$$

With the help of (2.22), we get that $b(f,2,[\frac{1}{2},1]) = (\frac{1}{2},\frac{1}{2},2)$. We saw that $b_\alpha(f,2,[0,\frac{1}{2}]) \geq 0 \;\forall\; \alpha$ and that $b_\alpha(f,2,[\frac{1}{2},1]) \geq 0 \;\forall\; \alpha$, so $f \geq 0$ for $x \in [0,1]$.

By now, we know that $b_\alpha(p,d,\Delta_k)$ is a function of $v_0, v_1, ..., v_k$, which are the vertices of Δ_k. In the lemma 9, we are going to proof continuity of the Bernstein-coefficients with respect to these vertices.

Lemma 9. *$b_\alpha(p,d,v_0,...,v_n)$ is a continuous function of $v_0,...,v_n$*

Proof. One can proof this easily with the help of paragraph 2.7. You can see that $b_\alpha(p,d,\Gamma)$ is a continuous function of $b_\alpha(p,d,\Delta)$ and T by construction. This means that $b_\alpha(p,d,v_0,...,v_n)$ is a continuous function of $v_0,...,v_n$. □

Furthermore, we tested that if $p_n(x) \in \mathbb{R} \Rightarrow$ there exist finitely many simplicial partitions Δ_k for which $b_\alpha(p,d,\Delta_k) \geq 0 \;\forall\; k$. This holds for all the polynomials we've tried and we use this conjecture in our algorithms.

The next theorem holds for a polynomial of degree 1 with 1 variable. We need it for one of our algorithms.

Theorem 10. *Let $p_1(x) = ax+b$ be a polynomial of degree 1 with 1 variable. Such a polynomial has 2 Bernstein-coefficients $b_{\alpha 1}$ and $b_{\alpha 2}$ on Δ_S.*

$$\textit{If } b_{\alpha 1} < 0 \textit{ and/or } b_{\alpha 2} < 0 \Rightarrow p_n(x) \notin \mathscr{R}.$$

Proof. With the help of (2.16), one can write $p_1(x) = ax+b = b_{\alpha 1}\lambda_0 + b_{\alpha 2}\lambda_1$ in barycentric coordinates. Because of (2.10) and (2.11), we can write $\lambda_1 = x$ and $\lambda_0 = 1-x$ in the 1 variable case. If $x = \lambda_1 = 0 \Rightarrow \lambda_0 = 1$ and $b_{\alpha 1}$ has to be greater then 0 in order to ensure the positivity of the polynomial on Δ_S. On the other hand, if $x = \lambda_1 = 1 \Rightarrow \lambda_0 = 0$ and $b_{\alpha 2}$ has to be greater then 0 to ensure the positivity. □

The same argument holds for every polynomial with 1 variable. For example, one can write a polynomial of degree 2 with 1 variable as $p_2(x) = ax^2+bx+c = b_{\alpha 1}\lambda_0^2 + 2b_{\alpha 2}\lambda_0\lambda_1 + b_{\alpha 3}\lambda_1^2$. If $\lambda_0 = 0$, then $\lambda_1 = 1$, hence $b_{\alpha 3}$ has to be greater or equal to zero to ensure the positivity of the polynomial on Δ_S. The same holds for $b_{\alpha 1}$. It is always true that the first and the last Bernstein-coefficient has to be greater or equal to zero if you write a polynomial of degree n with 1 variable in the following way: $p_n(x) = b_{\alpha 1}\lambda_0^n + ... + b_{\alpha(n+1)}\lambda_1^n$.

Another remark we make is that a polynomial has to be greater or equal then zero for every vertex of the simplicial partition. If this is not true, then the polynomial will not be positive on Δ_S. We will use this in our algorithms. Unfortunately, the algorithms do not work in some cases, namely when the minimum of a polynomial on the standard simplex Δ_S is exactly zero, just as in [1]. This is why we let the algorithms stop when when we have a function value between 0 and -10^{-6} on a vertex to catch matlab errors.

The last remark we make is that the algorithms for 1 variable would also work on another interval instead of the standard simplex in 1 dimension: $\Delta_S = [0,1]$, but we always use the standard simplex.

Chapter 3

The algorithms

Theorem 7 en the related conjecture give rise to algorithms which decide whether or not a polynomial is positive on the standard simplex and how many simplicial partitions are needed.

3.1 The constant function

We start very basic with the algorithm for the constant function. The Bernstein-coefficient (there is only one) of this constant function on the standard simplex in 0 dimensions is the constant itself. This is the input of the algorithm. The output is whether or not the function is positive on the standard simplex.

Algorithm 1. Test whether or not a constant function is positive on the standard simplex in 0 dimensions

---

Input: A constant c

1: **if** $c < 0$ **then return** "This function is not positive on the standard simplex."
2: **else return** "This function is positive on the standard simplex."
3: **end if**

Output: "This function is not positive on the standard simplex." or "This function is positive on the standard simplex.".

3.2 The polynomial of degree 1 with 1 variable

With the help of theorem 10 we get the following algorithm for a polynomial of degree 1 with 1 variable like $p_1(x) = ax + b$. The input are a and b from the polynomial or the two Bernstein-coefficients of the polynomial with respect to the standard simplex Δ_S. The output is whether or not the polynomial is positive on the standard simplex.

Algorithm 2. Test whether or not a polynomial of degree 1 with 1 variable is positive on the standard simplex in 1 dimension

---

Input: 2 polynomial (p) coefficients or 2 Bernstein (b) coefficients $b_{\alpha 1}$ and $b_{\alpha 2}$.

1: **if** input = p **then**
2: Calculate Bernstein-coefficients
3: **end if**
4: **if** $b_{\alpha 1} < 0 \;||\; b_{\alpha 2} < 0$ **then return**
5: "This polynomial is not positive on the standard simplex."
6: **else return** "This polynomial is positive on the standard simplex."
7: **end if**

Output: "This polynomial is not positive on the standard simplex." or "This polynomial is positive on the standard simplex.".

3.3 The polynomial of degree 2 with 1 variable

The next algorithm is made for a polynomial of degree 2 with 1 variable like $p_2(x) = ax^2 + bx + c$. We make use of the same argument as in theorem 10. The input are a, b and c from the polynomial or the three Bernstein-coefficients of the polynomial with respect to the standard simplex Δ_S. The output gives the number of times the algorithm used simplicial partition before all the Bernstein-coefficients are positive, or it gives back that the polynomial is positive (without simplicial partition), or it gives back that the polynomial is not positive on the standard simplex at all.

Algorithm 3. Test whether or not a polynomial of degree 2 with 1 variable is positive on the standard simplex in 1 dimension

Input: 3 polynomial (p) coefficients or 3 Bernstein (b) coefficients $b_{\alpha 1}$,$b_{\alpha 2}$ and $b_{\alpha 3}$.

if input = p **then**
    Calculate Bernstein-coefficients
end if
if $b_\alpha \geq 0 \; \forall \; \alpha$ **then return**
    "This polynomial is positive on the standard simplex."
end if
if $b_{\alpha 1} < 0 \; || \; b_{\alpha 3} < 0$ **then return**
    "This polynomial is not positive on the standard simplex."
end if
$i = 0$
while $\exists \; b_\alpha < 0 \; \forall \; \alpha$ with respect to the simplicial partitions **do**
    $i = i + 1$
    Simplicial partition
    for $\forall$ partitions **do**
        Calculate Bernstein-coefficients
        if $\exists$ a vertex v with $-10^{-6} < f(v) < 0$ in the simpl. part. **then return**
            "The algorithm failed."
        end if
        if $\exists$ a vertex v with $f(v) < 0$ in the simplical partition **then return**
            "This polynomial is not positive on the standard simplex."
        end if
    end for
end while
return i

Output: "This polynomial is positive on the standard simplex." or "This polynomial is negative on the standard simplex." or the number of simplicial partition steps i in which the algorithm converges.

3.4 The polynomial of degree $n > 2$ with 1 variable

The algorithm for a polynomial of degree $n > 2$ with 1 variable works in the same way as the algorithm for the polynomial of degree 2 with 1 variable.

Algorithm 4. Test whether or not a polynomial of degree n with 1 variable is positive on the standard simplex in 1 dimension

Input: $n+1$ polynomial (p) coefficients or $n+1$ Bernstein (b) coefficients $b_{\alpha 1}$, ..., $b_{\alpha(n+1)}$.

if input = p **then**
    Calculate Bernstein-coefficients
end if
if $b_\alpha \geq 0 \; \forall \; \alpha$ **then return**
    "This polynomial is positive on the standard simplex."
end if
if $b_{\alpha 1} < 0 \; || \; b_{\alpha(n+1)} < 0$ **then return**
    "This polynomial is not positive on the standard simplex."
end if
$i = 0$
while $\exists \; b_\alpha < 0 \; \forall \; \alpha$ with respect to the simplicial partitions **do**
    $i = i + 1$
    Simplicial partition
    for $\forall$ partitions **do**
        Calculate Bernstein-coefficients
        if $\exists$ a vertex v with $-10^{-6} < f(v) < 0$ in the simpl. part. **then return**
            "The algorithm failed."
        end if
        if $\exists$ a vertex v with $f(v) < 0$ in the simplical partition **then return**
            "This polynomial is not positive on the standard simplex."
        end if
    end for
end while
return i

Output: "This polynomial is positive on the standard simplex." or "This polynomial is negative on the standard simplex." or the number of simplicial partition steps i in which the algorithm converges.

3.5 The polynomial of degree 1 with 2 variables

With the same arguments of the 1 variable case we get the following algorithm for a polynomial of degree 1 with 2 variables like $p_1(x_1,x_2) = ax_1 + bx_2 + c$. The input are a, b and c from the polynomial or the three Bernstein-coefficients of the polynomial with respect to the standard simplex Δ_S. The output is whether or not the polynomial is positive on the standard simplex.

Algorithm 5. Test whether or not a polynomial of degree 1 with 2 variables is positive on the standard simplex in 2 dimensions

---

Input: 3 polynomial (p) coefficients or 3 Bernstein (b) coefficients $b_{\alpha 1}$, $b_{\alpha 2}$ and $b_{\alpha 3}$.

1: **if** input = p **then**
2: Calculate Bernstein-coefficients
3: **end if**
4: **if** $b_{\alpha 1} < 0 \,||\, b_{\alpha 2} < 0 \,||\, b_{\alpha 3} < 0$ **then return**
5: "This polynomial is not positive on the standard simplex."
6: **else return** "This polynomial is positive on the standard simplex."
7: **end if**

Output: "This polynomial is not positive on the standard simplex." or "This polynomial is positive on the standard simplex.".

3.6 The polynomial of degree 2 with 2 variables

The next algorithm is made for a polynomial of degree 2 with 2 variables like $p_2(x_1,x_2) = ax_1^2 + bx_2^2 + cx_1x_2 + dx_1 + ex_2 + f$. We make use of the same arguments as in the 1 variable case. The input are six polynomial coefficients or six Bernstein coefficients of the polynomial with respect to the standard simplex Δ_S from $p(\lambda) = b_{\alpha 1}\lambda_0^2 + b_{\alpha 2}\lambda_1^2 + b_{\alpha 3}\lambda_2^2 + 2b_{\alpha 4}\lambda_0\lambda_1 + 2b_{\alpha 5}\lambda_0\lambda_2 + 2b_{\alpha 6}\lambda_1\lambda_2$. The output gives the number of times the algorithm used simplicial partition before all the Bernstein-coefficients are positive, or it gives back that the polynomial is positive (without simplicial partition), or it gives back that the polynomial is not positive on the standard simplex at all.

Algorithm 6. Test whether or not a polynomial of degree 2 with 2 variables is positive on the standard simplex in 2 dimension

Input: 6 polynomial (p) coefficients or 6 Bernstein (b) coefficients $b_{\alpha 1}$, $b_{\alpha 2}$ ... $b_{\alpha 6}$.

if input = p **then**
    Calculate Bernstein-coefficients
end if
if $b_\alpha \geq 0 \; \forall \; \alpha$ **then return**
    "This polynomial is positive on the standard simplex."
end if
if $b_{\alpha 1} < 0 \;||\; b_{\alpha 2} < 0 \;||\; b_{\alpha 3} < 0$ **then return**
    "This polynomial is not positive on the standard simplex."
end if
$i = 0$
while $\exists \; b_\alpha < 0 \; \forall \; \alpha$ with respect to the simplicial partitions **do**
    $i = i + 1$
    Simplicial partition
    for $\forall$ partitions **do**
        Calculate Bernstein-coefficients
        if $\exists$ a vertex v with $-10^{-6} < f(v) < 0$ in the simpl. part. **then return**
            "The algorithm failed."
        end if
        if $\exists$ a vertex v with $f(v) < 0$ in the simplical partition **then return**
            "This polynomial is not positive on the standard simplex."
        end if
    end for
end while
return i

Output: "This polynomial is positive on the standard simplex." or "This polynomial is negative on the standard simplex." or the number of simplicial partition steps i in which the algorithm converges.

3.7 The polynomial of degree $n > 2$ with 2 variables

The next algorithm is made for a polynomial of degree $n > 2$ with 2 variables. One can extract it from the algorithm for the polynomial of degree 2 with 2 variables. We included it for the completeness of this thesis.

Algorithm 7. Test whether or not a polynomial of degree $n > 2$ with 2 variables is positive on the standard simplex in 2 dimension

Input: The polynomial (p) coefficients or the Bernstein (b) coefficients

if input = p **then**
    Calculate Bernstein-coefficients
end if
if $b_\alpha \geq 0 \; \forall \; \alpha$ **then return**
    "This polynomial is positive on the standard simplex."
end if
if $b_{\alpha 1} < 0 \; ||...|| \; b_{\alpha n+1} < 0$ **then return**
    "This polynomial is not positive on the standard simplex."
end if
$i = 0$
while $\exists \; b_\alpha < 0 \; \forall \; \alpha$ with respect to the simplicial partitions **do**
    $i = i + 1$
    Simplicial partition
    for $\forall$ partitions **do**
        Calculate Bernstein-coefficients
        if $\exists$ a vertex v with $-10^{-6} < f(v) < 0$ in the simpl. part. **then return**
            "The algorithm failed."
        end if
        if $\exists$ a vertex v with $f(v) < 0$ in the simplical partition **then return**
            "This polynomial is not positive on the standard simplex."
        end if
    end for
end while
return i

Output: "This polynomial is positive on the standard simplex." or "This polynomial is negative on the standard simplex." or the number of simplicial partition steps i in which the algorithm converges.

Chapter 4

Results

For the output: ”NP” means ”not positive on the standard simplex”, ”P” means ”positive on the standard simplex”, ”F” means ”the algorithm failed” and and a number is the number of times simplicial partition is used before the algorithm concludes that the polynomial is positive on the standard simplex.

4.1 The constant function

The input is a constant c.

Input	$c < 0$	$c = 0$	$c > 0$
Output	NP	P	P

4.2 The polynomial of degree 1 with 1 variable

The input are 2 Bernstein-coefficients $b_{\alpha 1}$ and $b_{\alpha 2}$.

Input	$b_{\alpha 1} < 0, b_{\alpha 2} \in \mathbb{R}$	$b_{\alpha 1} \in \mathbb{R}, b_{\alpha 2} < 0$	rest
Output	NP	NP	P

4.3 The polynomial of degree 2 with 1 variable

The input are 3 Bernstein-coefficients [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$].

Input	$b_\alpha \geq 0 \; \forall \; \alpha$	$b_{\alpha 1} < 0, b_{\alpha 2}, b_{\alpha 3} \in \mathbb{R}$	$b_{\alpha 1}, b_{\alpha 2} \in \mathbb{R}, b_{\alpha 3} < 0$	rest
Output	P	NP	NP	?

We tried different combinations of $b_{\alpha 1}$, $b_{\alpha 2}$ and $b_{\alpha 3}$, with $b_{\alpha 1} \geq 0$, $b_{\alpha 2} < 0$ and $b_{\alpha 3} \geq 0$. We started with the polynomial of the examples 3 and 8: $f = 6x^2 - 6x + 2$ which we found positive after 1 simplicial partition step according to example 8. The Bernstein-coefficients of this polynomial are [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$] = [2 -1 2] according to example 3. You can see our findings in the next table.

	Input	**Output**
Example	[2 -1 2]	1
different $b_{\alpha 1}$	[1 -1 2]	1
	[0.51 -1 2]	4
	[0.5000000000000001 -1 2]	27
	[0.50000000000000001 -1 2]	F
	[0.5 -1 2]	F
	[0 -1 2]	NP
different $b_{\alpha 2}$	[2 -2 2]	1
	[2 -2.0000000000000001 2]	1
	[2 -2.000000000000001 2]	NP
different $b_{\alpha 3}$	[2 -1 1]	1
	[2 -1 0.51]	4
	[2 -1 0.5000000000000001]	27
	[2 -1 0.50000000000000001]	F
	[2 -1 0.5]	F
	[2 -1 0]	NP

4.4 The polynomial of degree 3 with 1 variable

The input are 4 Bernstein-coefficients [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$ $b_{\alpha 4}$].

Input	$b_\alpha \geq 0 \; \forall \; \alpha$	$b_{\alpha 1} < 0, b_{\alpha 2}, b_{\alpha 3}, b_{\alpha 4} \in \mathbb{R}$	$b_{\alpha 1}, b_{\alpha 2}, b_{\alpha 3} \in \mathbb{R}, b_{\alpha 4} < 0$	rest
Output	P	NP	NP	?

We tried different combinations of $b_{\alpha 1}$, $b_{\alpha 2}$, $b_{\alpha 3}$ and $b_{\alpha 4}$ with $b_{\alpha 1} \geq 0$, $b_{\alpha 2} < 0$ or $b_{\alpha 3} < 0$ and $b_{\alpha 4} \geq 0$. Again, we started with the polynomial of the examples 3 and 8: $f = 6x^2 - 6x + 2$. The four Bernstein-coefficients of this polynomial are [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$ $b_{\alpha 4}$] = [2 0 0 2], thus we conclude immediately that this polynomial is positive on the standard simplex. Furthermore, we looked at the same polynomials (almost) as in paragraph 4.3 and some other polynomials of degree 3. You can see our findings in the next table.

	Input	**Output**
Example	[2 0 0 2]	1
Degree 2	[1 -1/3 0 2]	1
	[0.51 -0.5 0 2]	4
	[0.5 -0.5 0 2]	F
	[0 -0.5 0 2]	NP
	[2 -2/3 -2/3 2]	1
Degree 3	[1.1 1/3 -5/3 3]	1
	[1.00000000001 1/3 -5/3 3]	1
	[1 1/3 -5/3 3]	”F”
	[0.9 1/3 -5/3 3]	NP

4.5 The polynomial of degree 1 with 2 variables

The input are 3 Bernstein-coefficients $b_{\alpha 1}$, $b_{\alpha 2}$ and $b_{\alpha 3}$.

Input	$b_{\alpha 1} < 0, b_{\alpha 2} \in \mathbb{R}, b_{\alpha 3} \in \mathbb{R}$	$b_{\alpha 1} \in \mathbb{R}, b_{\alpha 2} < 0, b_{\alpha 3} \in \mathbb{R}$	$b_{\alpha 1} \in \mathbb{R}, b_{\alpha 2} \in \mathbb{R}, b_{\alpha 3} < 0$	rest
Output	NP	NP	NP	P

4.6 The polynomial of degree 2 with 2 variables and 3 variables

The input are 6 Bernstein-coefficients [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$ $b_{\alpha 4}$ $b_{\alpha 5}$ $b_{\alpha 6}$].

Input	$b_\alpha \geq 0 \; \forall \; \alpha$	$b_{\alpha 1} < 0$	$b_{\alpha 2} < 0$	$b_{\alpha 3} < 0$	rest
Output	P	NP	NP	NP	?

Note that we changed the sequence of Bernstein coefficients comparing to the 1 variable cases. We tried different combinations of $b_{\alpha 1}$, $b_{\alpha 2}$, $b_{\alpha 3}$, $b_{\alpha 4}$, $b_{\alpha 5}$ and $b_{\alpha 6}$ with $b_{\alpha 1}, b_{\alpha 2}, b_{\alpha 3} \geq 0$ and at least one of $b_{\alpha 4}, b_{\alpha 5}$ and $b_{\alpha 6} < 0$ Again, we started with the polynomial of the examples 3 and 8: $f = 6x^2 - 6x + 2$. The six Bernstein-coefficients of this polynomial are [$b_{\alpha 1}$ $b_{\alpha 2}$ $b_{\alpha 3}$ $b_{\alpha 4}$ $b_{\alpha 5}$ $b_{\alpha 6}$] = [2 2 2 -1 2 -1]. This time, the algorithm finds after two times simplicial partition that the polynomial is positive on the standard simplex. Most of the second degree polynomials with 2 variables that are positive will be found within 2 simplicial partition steps with our algorithm, but some will be found in 1 step. We even found an example which used 53 simplicial partition steps! You can see some examples and the rest of our findings in the following table. We dit not find Bernstein-coefficients for which the algorithm did not stop, but maybe this is possible.

	Input	**Output**
Example	[2 2 2 -1 2 -1]	2
	[2 2 2 0 2 -1]	1
	[2 2 2 0 2 0]	P
	[2/3 1+2/3 1+2/3 -1/3 -1/3 -1]	3
Round-off error	$[2/3(1+2/3)(1+2/3)-1/3-1/3(-1-1/3)]$	55
Exact	$[2/3(1+2/3)(1+2/3)-1/3-1/3(-1-1/3)]$	F
	[3 3 0 -3 -2 2/3]	NP

4.7 The polynomial of degree 3 with 2 variables

This works in the same way as the polynomials of degree 2 with 2 variables. It leads to the same findings and errors. Most of the positive polynomials (on the standard simplex) will be found within 2 simplicial partition steps. Some need one step and polynomials with a local minimum close to zero need more steps. The algorithm gives false if a local minimum is at exactly 0. The positive P and the not positive NP both work perfect.

Chapter 5

Conclusions

5.1 The constant function

Of course, this algorithm works perfectly. If the constant is smaller then zero, the function is not positive on the standard simplex. The rest of the constants are positive on the standard simplex.

5.2 The polynomial of degree 1 with 1 variable

This algorithm also works perfectly. If the first and/or the second Bernstein-coefficients is smaller then zero, the function is not positive on the standard simplex. The rest of the functions are positive on the standard simplex.

5.3 The polynomial of degree 2 with 1 variable

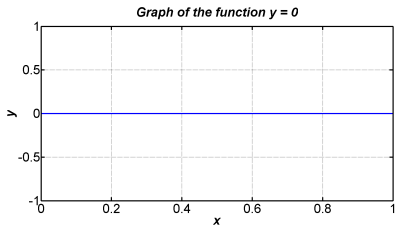
The algorithm works within 1 simplicial partition step for the polynomial of the examples 3 and 8: $f = 6x^2 - 6x + 2$ with Bernstein-coefficients [2 -1 2]. This polynomial has a minimum of 0.5. We changed the first Bernstein-coefficient $b_{\alpha 1}$ and we concluded that the algorithm also works within 1 simplicial partition step for $b_{\alpha 1} > 2$. For $b_{\alpha 1} < 2$ and getting close to 0.5, we see that we need more and more simplicial partition steps in order to find the positivity on the standard simplex. When $b_{\alpha 1} = 1$, the function is $f = 5x^2 - 4x + 1$. This function has a minimum of 0.2 and the algorithm still stops after 1 simplicial partition step. When $b_{\alpha 1} = 0.51$, the function is $f = 4.51x^2 - 3.02x + 0.51$. This function has a minimum close to 0 and the algorithm stops after 4 simplicial partition steps. When $b_{\alpha 1} = 0.5000000000000001$, the function is very close to $f = 4.5x^2 - 3x + 0.5$ and it has a minimum very close to 0. The algorithm still stops (after 27 simplicial partition

steps). If we put in another zero in $b_{\alpha 1}$, the algorithm concludes that it has failed. The same happens when $b_{\alpha 1}$ is exactly 0.5. The algorithm should have concluded that the function is positive on the standard simplex, because the mimimum of $f = 4.5x^2 - 3x + 0.5$ on the standard simplex is exactly zero! The same things appear when we differ $b_{\alpha 3}$ instead of $b_{\alpha 1}$.

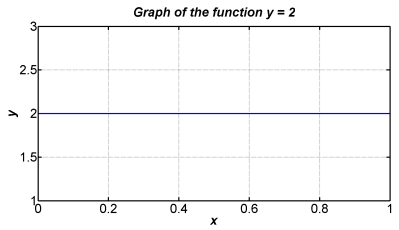
So, what happens exactly? 0.50000000000000001 gets rounded off to 0.5 by matlab. That explains the identical output of the two input values, but what happens when the input is exactly 0.5? We take a closer look at the function with input: Bernstein-coefficients [2 -1 0.5]. This function is $f = 4.5x^2 - 6x + 2$. The output is "The algorithm failed". If we didn't implement this failing outcome in our algorithm, something goes wrong in line 16 and 17 of algorithm 3: **if** $\exists$ a vertex v with $f(v) < 0$ in the simplicial partition **then return** "This polynomial is not positive on the standard simplex." We know from the proof of theorem 10 that for a simplex v, $v = x = \lambda_1$ holds in 1 dimension with 1 variable, furthermore, $\lambda_0 = 1 - \lambda_1 = 1 - x = 1 - v$. This means we can write $f = b_{\alpha 1}\lambda_0^2 + b_{\alpha 2}\lambda_0\lambda_1 + b_{\alpha 3}\lambda_1^2 = b_{\alpha 1}(1-v)^2 + 2b_{\alpha 2}(1-v)v + b_{\alpha 3}v^2$ for a vertex v. We use this in our algorithm. After 28 simplicial partition steps, we get a vertex $v = 0.6666666679084301$. According to matlab, $4.5v^2 - 6v + 2 = 0$ and $2(1-v)^2 + 2 \cdot -1(1-v)v + 0.5v^2 = -2.7756 \cdot 10^{-17}$, while $2(1-v)^2 + 2 \cdot -1(1-v)v + 0.5v^2 = 4.5v^2 - 6v + 2$! It turns out that matlab made another round off error. If the top of the graph of a function is 0, it takes infinite simplicial partition steps to determine if the function is positive on the standard simplex, just as in [1]. In our algorithm, we get a round off error when we reach 28 simplicial partition steps.

The argument above is not true if we differ $b_{\alpha 2}$. The top of the graph of the function with Bernstein-coefficients [2 -2 2]: $f = 8x^2 - 8x + 2$ is exactly 0, but in 1 simplicial partition step we find that the function is positive. Of course, we still get the round off error with many significant digits.

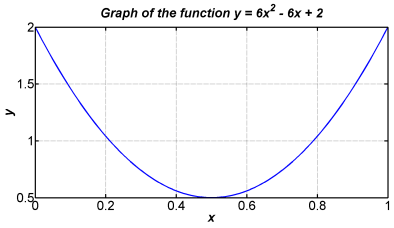
Hereunder, one can see graphs of functions with different Bernstein-coefficients.



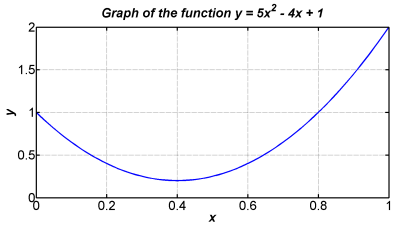
(a) Bernstein-coefficients (0 0 0)



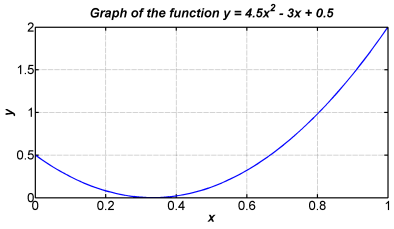
(b) Bernstein-coefficients (2 2 2)



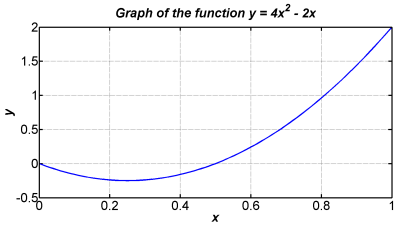
(c) Bernstein-coefficients (2 -1 2)



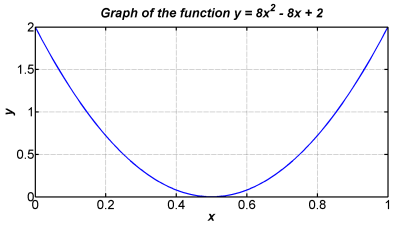
(d) Bernstein-coefficients (1 -1 2)



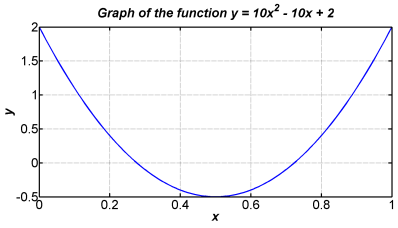
(e) Bernstein-coefficients (0.5 -1 2)



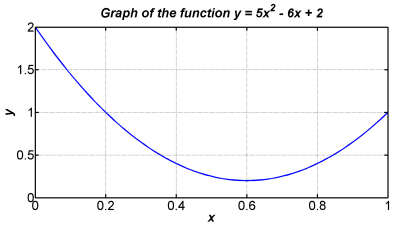
(f) Bernstein-coefficients (0 -1 2)



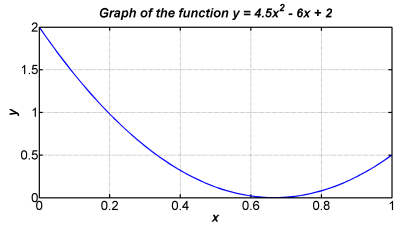
(g) Bernstein-coefficients (2 -2 2)



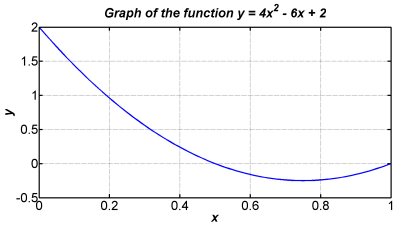
(h) Bernstein-coefficients (2 -3 2)



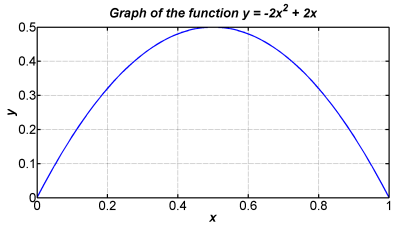
(i) Bernstein-coefficients (2 -1 1)



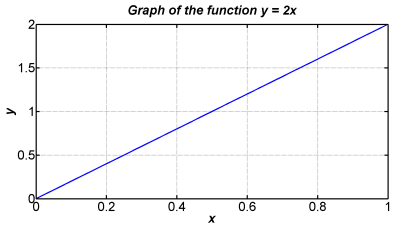
(j) Bernstein-coefficients (2 -1 0.5)



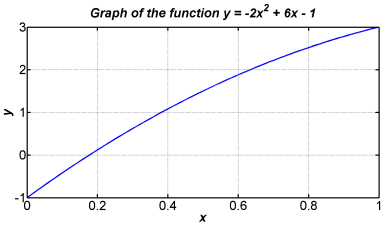
(k) Bernstein-coefficients (2 -1 0)



(l) Bernstein-coefficients (0 1 0)



(m) Bernstein-coefficients (0 1 2)



(n) Bernstein-coefficients (-1 2 3)

If you take a look at the figures above, you see that the first Bernstein-coefficient with respect to the standard simplex $[0,1]$ of a function y is equal to the function-value at $x = 0$: $y(0)$. Furthermore, The third Bernstein-coefficient is equal to $y(1)$. The second Bernstein-coefficient works in the following way: if $b_{\alpha 2} < \frac{1}{2} b_{\alpha 1} b_{\alpha 3}$ then the graph is a parabola which opens downwards and if $b_{\alpha 2} > \frac{1}{2} b_{\alpha 1} b_{\alpha 3}$ then the graph is a parabola which opens upwards and if $b_{\alpha 2} = \frac{1}{2} b_{\alpha 1} b_{\alpha 3}$ then the graph is a line. We don't know for sure if the algorithm always has an outcome. It could be possible that the algorithm doesn't stop partitioning simplices for some Bernstein coefficients like an example of a polynomial of degree 3 with 1 variable (the one with Bernstein coefficients [1 1/3 -5/3 3]).

5.4 The polynomial of degree 3 with 1 variable

For the polynomial of the examples 3 and 8 it follows immediately that this polynomial is positive on the standard simplex because all the Bernstein-coefficients are greater then or equal to zero. Furthermore, for strictly positive polynomials the algorithm works within one simplicial partition step for degree 3 polynomials. The degree 3 algorithm works the same for degree 2 polynomials as the square algorithm. When a degree 3 polynomial has a minimum equal to zero on the standard simplex then the algorithm doesn't work (marked with "F"). The test variable which tests if the algorithm doesn't stop is always exactly zero in the last steps. Hereunder, one can see the graph of the function for which our algorithm failed.

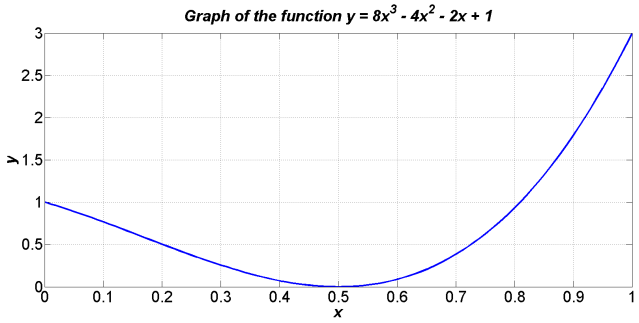


Figure 5.1: Bernstein-coefficients (1 1/3 -5/3 3)

With four Bernstein-coefficients, we have that the first Bernstein-coefficient is equal to the function-value at $x = 0$: $y(0)$ and the fourth Bernstein-coefficient is equal to $y(1)$. With the second Bernstein-coefficient one can predict if the graph is going to increase or decrease: if $b_{\alpha 2} < b_{\alpha 1}$ then the graph starts decreasing, if $b_{\alpha 2} = b_{\alpha 1}$ then the graph starts flat and if $b_{\alpha 2} > b_{\alpha 1}$ then the graph starts increasing. The same holds (only mirrored) if you compare the fourth Bernstein-coefficient with the third.

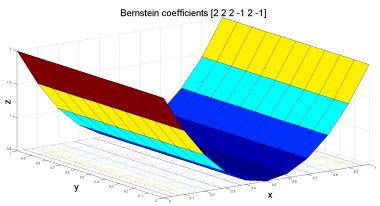
5.5 The polynomial of degree 1 with 2 variables

This algorithm works perfectly. If the first and/or the second and/or the third Bernstein-coefficients is smaller then zero, the function is not positive on the standard simplex. The rest of the functions are positive on the standard simplex.

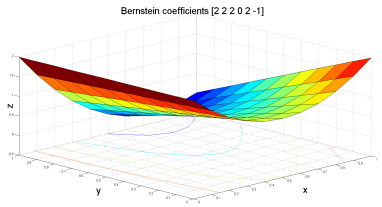
5.6 The polynomial of degree 2 with 2 and 3 variables

For the polynomial of the examples 3 and 8 it follows rapidly that this polynomial is positive on the standard simplex (2 simplicial partition steps). Furthermore, for

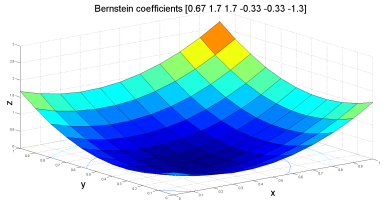
most positive polynomials the algorithm works within one simplicial partition step. When a polynomial polynomial has a local minimum equal to zero on the standard simplex then the algorithm doesn't work (marked with "F"). The test variable which tests if the algorithm doesn't stop is always exactly zero in the last steps. Hereunder, one can some graphs of functions which we tested. One can see that



(a) $6x_1^2 - 6x_1 + 2$



(b) $4x_1^2 - 2x_1x_2 - 4x + 2$



(c) $3x_1^2 + 3x_2^2 - 2x_1 - 2x_2 + 2/3$

the first Bernstein coefficient is the value at the point (0, 0). The second Bernstein coefficient is the value at the point (1, 0) and the third Bernstein coefficient is the value at the point (0, 1). The other Bernstein-coefficients give how the the graph is bending.

5.7 General conclusion

All the algorithms work very well except for some special cases. There are some cases with a local minima at exactly zero for which an algorithm does not stop and if a local minimum of a polynomial is at exactly 0, the outcome of the algorithm is "failure". Most of the time there is only 1 simplicial partition step needed for a 1-dimensional polynomial and 2 simplicial partition steps for 2-dimensional polynomials, except for the cases that a polynomial has a local minimum close to zero. In that case, the number of simplicial partition steps can reach up to above 50. The use of the Bernstein-Beziér form of polynomials certainly has potential outside Beziér curves. The big disadvantage is that describing a complex polynomial with Bernstein-coefficients in an efficient way is hard and complicated. Nevertheless, describing polynomials with the help of Bernstein coefficients is useful, even compared to the normal polynomial coefficients.

Chapter 6

Matlab code

6.1 The constant function

```
function [] = constant()
% Input: constant. Output: positivity/negativity on the standard simplex.
% This algorithm only holds for constant functions.

constant = input('Enter a constant: ');

syms x
ezplot(constant + 0 * x,[0 1]);
grid on;
xlabel('\it{\bf{x}}');
ylabel('\it{\bf{y}}');
title(sprintf('\\it{\\bf{Graph of the function y = %g}}', constant));

if constant < 0
    disp('This function is not positive on the standard simplex.');
else
    disp('This function is positive on the standard simplex.');
end
```

6.2 The polynomial of degree 1 with 1 variable

```
function [] = degree1()
% Input: [a b]. Output: positivity/negativity on the standard simplex.
% This algorithm only holds for polynomials with a degree of max 1 with 1
% variable. The input are [a b] from p = ax + b or 2 Bernstein-coefficients
% [a b] from p = a * lambda_0 + b * lambda_1.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter p = ax + b as [a b]: ');
    Ber = [pol(2), pol(1) + pol(2)];
elseif porb == 'b'
    Ber = input('Enter p = a * lambda_0 + b * lambda_1 as [a b]: ');
    pol = [Ber(2) - Ber(1), Ber(1)];
end

x = 0 : 0.1 : 1;
y = pol(1) * x + pol(2);
plot(x, y);
grid on;
xlabel('\it{\bf{x}}');
ylabel('\it{\bf{y}}');
graphtitledegree1(pol);

if Ber(1) < 0 || Ber(2) < 0 % Checking the function value of vertex 0 and vertex 1.
    disp('This polynomial is not positive on the standard simplex.');
else
    disp('This polynomial is positive on the standard simplex.');
end
```

6.3 The polynomial of degree 2 with 1 variable

```
  function [] = square()
% Input: [a b c]. Output: # times simplicial partition.
% This only holds for polynomials of maximal degree 2 with 1 variable.
% The input are [a b c] from p = ax^2 + bx + c or 3 Bernstein-coefficients
% [a b c] from p = a * lambda_0^2 + b * 2 * lambda_0 * lambda_1 + c *
% lambda_1^2. The output gives the number of times the algorithm used
% simplicial partition before all the Bernstein-coefficients are positve or
% it gives back that the polynomial is not positive on the standard simplex
% at all.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter p = ax^2 + bx + c as [a b c]: ');
    Ber = [pol(3), (pol(2) + 2 * pol(3)) / 2, pol(1) + pol(2) + pol(3)];
elseif porb == 'b'
    Ber = input('Enter the Bernstein-coefficients as [a b c]:  ');
    pol = [Ber(1) - 2 * Ber(2) + Ber(3), -2 * Ber(1) + 2 * Ber(2), Ber(1)];
end

x = 0 : 0.01 : 1; % Plotting the graph of the function.
y = pol(1) * x.^2 + pol(2) * x + pol(3);
plot(x, y);
grid on;
xlabel('\it{\bf{x}}');
ylabel('\it{\bf{y}}');
graphtitlesquare(pol);

if Ber >= 0
    disp('This polynomial is positive on the standard simplex.');
    return
end

if Ber(1) < 0 || Ber(3) < 0
    disp('This polynomial is not positive on the standard simplex.');
    return
end

Pinv = [-1 1; 1 0];
i = 0; % # times we already did simplicial partition.
j = 1; % The size of the partitions.
spot = [0 1]; % Indication of the partition where the Bern.-coeff. are < 0
stop = 1; % Stopping criterium.

while stop
    stop = 0;
    m = 1; % The spot in the spot-matrix in row i;
    i = i + 1;
    j = j / 2;
    numberofpart = 1 / j; % The number of different partitions.
    Bernstein = [0 0 0];

    for k = spot(i, :) % Simp. partition and calculating Bernstein-coeff.
        vertex1 = k / numberofpart;
        vertex2 = (k + 1) / numberofpart;
```

```
60          P = [vertex1, vertex2; 1 1];
61          T = Pinv * P;
62          Bernstein(1) = Ber(1) * T(1,1)^2 + 2 * Ber(2) * T(1,1) * T(2,1) + Ber(3) * T(2,1)^2;
63          Bernstein(2) = Ber(1) * T(1,1) * T(1,2) + Ber(2) * T(1,1) * T(2,2) + Ber(2) * ...
64          T(1,2) * T(2,1) + Ber(3) * T(2,1) * T(2,2);
65          Bernstein(3) = Ber(1) * T(1,2)^2 + 2 * Ber(2) * T(1,2) * T(2,2) + Ber(3) * T(2,2)^2;
66
67          if sum(Bernstein<0) > 0
68              spot(i + 1, m) = k * 2;
69              spot(i + 1, m + 1) = k * 2 + 1;
70              m = m + 2;
71              stop = 1;
72          end
73
74          for a = [vertex1 vertex2] % Testing vertices.
75          test = Ber(1) * (1 - a)^2 + Ber(2) * 2 * (1 - a) * a + Ber(3) * a^2;
76              if test > -10^-6 && test < 0
77                  fprintf('The algorithm failed.\n');
78                  return
79              elseif test < -10^-6
80                  fprintf('This polynomial is not positive on the standard simplex.\n');
81                  return
82              end
83          end
84      end
85  end
86
87  fprintf('Number of times simplicial partition is used: %d\n', i);
```

6.4 The polynomial of degree 3 with 1 variable

```
  function [] = degree3()
% Input: [a b c d]. Output: # times simplicial partition.
% This only holds for polynomials of maximal degree 3 with 1 variable.
% The input are [a b c d] from p = ax^3 + bx^2 + cx + d or 4 Bernstein-
% coefficients [a b c d] from p = a * lambda_0^3 + b * 3 * lambda_0^2 *
% lambda_1 + c * 3 * lambda_0 * lambda_1^2 + d * lambda_1^3. The output
% gives the number of times the algorithm used simplicial partition before
% all the Bernstein-coefficients are positve or it gives back that the
% polynomial is not positive on the standard simplex at all.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter p = ax^3 + bx^2 + cx + d as [a b c d]: ');
    Ber = [pol(4), (pol(3) + 3 * pol(4)) / 3, (pol(2) + 2 * pol(3) + 3 * pol(4)) / 3, ...
        pol(1) + pol(2) + pol(3) + pol(4)];
elseif porb == 'b'
    Ber = input('Enter the Bernstein-coefficients as [a b c d]:  ');
    pol = [- Ber(1) + 3 * Ber(2) - 3 * Ber(3) + Ber(4), 3 * (Ber(3) - 2 * Ber(2) + Ber(1)), ...
        -3 * Ber(1) + 3 * Ber(2), Ber(1)];
end

x = 0 : 0.01 : 1; % Plotting the graph of the function.
y = pol(1) * x.^3 + pol(2) * x.^2 + pol(3) * x + pol(4);
plot(x, y);
grid on;
xlabel('\it{\bf{x}}');
ylabel('\it{\bf{y}}');
graphtitledegree3(pol);

if Ber >= 0
    disp('This polynomial is positive on the standard simplex.');
    return
end

if Ber(1) < 0 || Ber(4) < 0
    disp('This polynomial is not positive on the standard simplex.');
    return
end

Pinv = [-1 1; 1 0];
i = 0; % # times we already did simplicial partition.
j = 1; % The size of the partitions.
spot = [0 1]; % Indication of the partition where the Bern.-coeff. are < 0
stop = 1; % Stopping criterium.

while stop
    stop = 0;
    m = 1; % The spot in the spot-matrix in row i;
    i = i + 1;
    j = j / 2;
    numberofpart = 1 / j; % The number of different partitions.
    Bernstein = [0 0 0 0];

    for k = spot(i, :) % Simp. partition and calculating Bernstein-coeff.
```

```
60          vertex1 = k / numberofpart;
61          vertex2 = (k + 1) / numberofpart;
62          P = [vertex1, vertex2; 1 1];
63          T = Pinv * P;
64          Bernstein(1) = Ber(1) * T(1,1)^3 + 3 * Ber(2) * T(1,1)^2 * T(1,2) + 3 * Ber(3) * ...
65              T(1,1) * T(1,2)^2 + Ber(4) * T(2,1)^3;
66          Bernstein(2) = Ber(1) * T(1,1)^2 * T(1,2) + Ber(2) * T(1,1)^2 * T(2,2) + 2 * ...
67              Ber(2) * T(1,1) * T(1,2) * T(2,1) + Ber(3) * T(1,2) * T(2,1)^2 + 2 * Ber(3) * ...
68              T(1,1) * T(2,1) * T(2,2) + Ber(4) * T(2,1)^2 * T(2,2);
69          Bernstein(3) = Ber(1) * T(1,1) * T(1,2)^2 + 2 * Ber(2) * T(1,1) * T(1,2) * ...
70              T(2,2) + Ber(2) * T(1,2)^2 * T(2,1) + 2 * Ber(3) * T(1,2) * T(2,1) * T(2,2) + ...
71              Ber(3) * T(1,1) * T(2,2)^2 + Ber(4) * T(2,1) * T(2,2)^2;
72          Bernstein(4) = Ber(1) * T(1,2)^3 + 3 * Ber(2) * T(1,2)^2 * T(2,2) + 3 * Ber(3) * ...
73              T(1,2) * T(2,2)^2 + Ber(4) * T(2,2)^3;
74
75          if sum(Bernstein<0) > 0
76              spot(i + 1, m) = k * 2;
77              spot(i + 1, m + 1) = k * 2 + 1;
78              m = m + 2;
79              stop = 1;
80          end
81
82          for a = [vertex1 vertex2] % Testing vertices.
83              test = Ber(1) * (1 - a)^3 + 3 * Ber(2) * (1 - a)^2 * a + 3 * Ber(3) * (1 - a) * ...
84                  a^2 + Ber(4) * a^3;
85              if test > -10^-6 && test < 0
86                  fprintf('The algorithm failed.\n');
87                  return
88              elseif test < -10^-6
89                  fprintf('This polynomial is not positive on the standard simplex.\n');
90                  return
91              end
92          end
93      end
94  end
95
96  fprintf('Number of times simplicial partition is used: %d\n', i);
```

6.5 The polynomial of degree 1 with 2 variables

```
function [] = degree12var()
% Input: [a b c]. Output: positivity/negativity on the standard simplex.
% This algorithm only holds for polynomials with a degree of max 1 with 2
% variables. The input are [a b c] from p = ax1 + bx2 + c or 2
% Bernstein-coefficients [a b c] from p = a * lambda_0 + b * lambda_1 +
% c * lambda_2.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter p = ax1 + bx2 + c as [a b c]: ');
    Ber = [pol(3), pol(1) + pol(3), pol(2) + pol(3)];
elseif porb == 'b'
    Ber = input('Enter p = a * lambda_0 + b * lambda_1 + c * lambda_2 as [a b c]: ');
end

if Ber(1) < 0 || Ber(2) < 0 || Ber(3) < 0 % Checking for vertices 1,2 and 3
    disp('This polynomial is not positive on the standard simplex.');
else
    disp('This polynomial is positive on the standard simplex.');
end
```

6.6 The polynomial of degree 2 with 2 variables

```
  function [] = square2var()
% Input: [a b c d e f]. Output: # times simplicial partition.
% This only holds for polynomials of maximal degree 2 with 2 variables.
% The input are [a b c d e f] from p = ax1^2 + bx2^2 + cx1x2 + dx1 + ex2 + f
% or 3 Bernstein-coefficients [a b c d e f] accordint to paragraph 3 of the
% thesis. The output gives the number of times the algorithm used
% simplicial partition before all the Bernstein-coefficients are positve or
% it gives back that the polynomial is not positive on the standard simplex
% at all.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter the polynomial as [a b c d e f]: ');
    Ber = [pol(6), pol(1) + pol(4) + pol(6), pol(2) + pol(5) + pol(6), (pol(4) + 2 * ...
        pol(6)) / 2, (pol(5) + 2 * pol(6)) / 2, (pol(3) + pol(4) + pol(5) + 2 * pol(6)) / 2]
elseif porb == 'b'
    Ber = input('Enter the Bernstein-coefficients as [a b c d e f]:  ');
    pol = [Ber(2) - 2 * Ber(4) + Ber(1), Ber(3) - 2 * Ber(5) + Ber(1), 2 * Ber(6) - 2 * Ber(4)
        + 2 * Ber(1) - 2 * Ber(5), 2 * Ber(4) - 2 * Ber(1), 2 * Ber(5) - 2 * Ber(1), Ber(1)]
end

x = 0 : 0.1 : 1;
y = 0 : 0.1 : 1;
[X, Y] = meshgrid(x, y);
Z = pol(1) * X.^2 + pol(2) * Y.^(2) + pol(3) * X.*Y.*1 + pol(4) * X.*1 + pol(5) * Y.*1 ...
    + pol(6);
surfc(X, Y, Z);
title(sprintf('Bernstein coefficients [%0.2g %0.2g %0.2g %0.2g %0.2g %0.2g]', Ber(1), ...
    Ber(2), Ber(3), Ber(4), Ber(5), Ber(6)))
xlabel('x'); ylabel('y'); zlabel('z');

if Ber >= 0
    disp('This polynomial is positive on the standard simplex.');
    return
end

if Ber(1) < 0 || Ber(2) < 0 || Ber(3) < 0
    disp('This polynomial is not positive on the standard simplex.');
    return
end

j = 0; % Aantal keer simplicial partition.
driehoeken(1, :) = [0 0 1 0 0 1];
stop = 1;

while stop

    stop = 0;
    j = j + 1;
    a = size(driehoeken, 1);

    for i = 1 : a

        [va, vb, vc] = maxdistance(driehoeken(i, :));
```

```
60
61          b = [va vb vc];
62          for c = 1 : 2 : 5
63              test = Ber(1) * (1 - b(c) - b(c+1))^2 + Ber(2) * b(c)^2 + Ber(3) * b(c+1)^2 + 2 * ...
64                  Ber(4) * (1 - b(c) - b(c+1)) * b(c) + 2 * Ber(5) * (1 - b(c) - b(c+1)) * b(c+1) ...
65                  + 2 * Ber(6) * b(c) * b(c+1);
66              if test > -10^-6 && test < 0
67                  fprintf('The algorithm failed.\n');
68                  return
69              elseif test < -10^-6
70                  fprintf('This polynomial is not positive on the standard simplex.\n');
71                  return
72              end
73          end
74
75          driehoek1 = [[(va(1) + vb(1)) / 2 (va(2) + vb(2)) / 2] va vc];
76          driehoek2 = [[(va(1) + vb(1)) / 2 (va(2) + vb(2)) / 2] vb vc];
77
78          if controleer(driehoek1, Ber) == 0
79              driehoeken(size(driehoeken, 1) + 1, :) = driehoek1;
80              stop = 1;
81          end
82
83          if controleer(driehoek2, Ber) == 0
84                  driehoeken(size(driehoeken, 1) + 1, :) = driehoek2;
85              stop = 1;
86          end
87
88      end
89
90      for i = 1 : a
91          driehoeken(1, :) = [];
92      end
93
94  end
95
96  fprintf('Number of times simplicial partition is used: %d\n', j);
```

6.7 The polynomials of degree 3 with 2 variables

```
  function [] = degree32var()
% Input: [a b c d e f g h i j]. Output: # times simplicial partition.
% This only holds for polynomials of maximal degree 3 with 2 variables.
% The input are [a b c d e f g h i j] from p = ax1^3 + bx2^3 + cx1^2x2 + dx1x2^2 + ex1^2
% + fx2^2 + gx1x2 + hx1 + ix2 + j or 10 Bernstein-coefficients [a b c d e f g h i j]
% according to chapter 2 of the thesis. The output gives the number of times the
% algorithm used simplicial partition before all the Bernstein-coefficients are positve
% or it gives back that the polynomial is not positive on the standard simplex at all.

porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');

while ~(strcmp(porb,'p') || strcmp(porb,'b'))
    porb = input('Polynomial (p) or Bernstein-coefficients (b)? ', 's');
end

if porb == 'p'
    pol = input('Enter the polynomial as [a b c d e f g h i j]: ');
    Ber = [pol(10), pol(1) + pol(5) + pol(8) + pol(10), pol(2) + pol(6) + pol(9) + pol(10), ...
        (pol(8) + 3 * pol(10)) / 3, (pol(9) + 3 * pol(10)) / 3, (pol(5) + 2 * pol(8) + ...
        3 * pol(10)) / 3, (pol(3) + pol(5) + pol(7) + 2 * pol(8) + pol(9) + 3 * pol(10)) ...
        / 3, (pol(6) + 2 * pol(9) + 3 * pol(10)) / 3, (pol(4) + pol(6) + pol(7) + pol(8) + ...
        2 * pol(9) + 3 * pol(10)) / 3, (pol(7) + 2 * pol(8) + 2 * pol(9) + 6 * pol(10)) / 6];
elseif porb == 'b'
    Ber = input('Enter the Bernstein-coefficients as [a b c d e f g h i j]:  ');
    pol = [Ber(2) + 51 * Ber(1) - 21 * Ber(4) - 3 * Ber(6), Ber(3) + 51 * Ber(1) - ...
        21 * Ber(5) - 3 * Ber(8), 3 * Ber(7) + 297 * Ber(1) - 90 * Ber(4) - 27 * Ber(5) - ...
        9 * Ber(6) - 18 * Ber(10), 3 * Ber(9) + 297 * Ber(1) - 27 * Ber(4) - 90 * Ber(5) - ...
        9 * Ber(8) - 18 * Ber(10), 3 * Ber(6) + 18 * Ber(4) - 45 * Ber(1), 3 * Ber(8) + ...
        18 * Ber(5) - 45 * Ber(1), 6 * Ber(10) + 6 * Ber(4) + 6 * Ber(5) - 30 * Ber(1), ...
        3 * Ber(5) - 9 * Ber(1), 3 * Ber(5) - 9 * Ber(1), Ber(1)];
end

x = 0 : 0.1 : 1;
y = 0 : 0.1 : 1;
[X, Y] = meshgrid(x, y);
Z = pol(1) * X.^3 + pol(2) * Y.^(3) + pol(3) * X.^2*Y.*1 + pol(4) * X.*Y.^2*1 + pol(5) * ...
    X.^2*1 + pol(6) * Y.^2*1 + pol(7) * X. * Y. * 1 + pol(8) * X. + pol(9) * Y. + pol(10);

surfc(X, Y, Z);
title('a mesh plot of a function of two variables');
xlabel('x'); ylabel('y'); zlabel('z');

if Ber >= 0
    disp('This polynomial is positive on the standard simplex.');
    return
end

if Ber(1) < 0 || Ber(2) < 0 || Ber(3) < 0
    disp('This polynomial is not positive on the standard simplex.');
    return
end

j = 0; % Aantal keer simplicial partition.
driehoeken(1, :) = [0 0 1 0 0 1];
stop = 1;

while stop

    stop = 0;
```

```
60      j = j + 1;
61      a = size(driehoeken, 1);
62
63      for i = 1 : a
64
65          [va, vb, vc] = maxdistance(driehoeken(i, :));
66
67          b = [va vb vc];
68          for c = 1 : 2 : 5
69              test = Ber(1) * (1 - b(c) - b(c + 1))^3 + Ber(2) * b(c)^3 + Ber(3) * b(c + 1)^3 ...
70                  + 3 * Ber(4) * (1 - b(c) - b(c + 1))^2 * b(c) + 3 * Ber(5) * (1 - b(c) - ...
71                  b(c + 1))^2 * b(c + 1) + 3 * Ber(6) * (1 - b(c) - b(c + 1)) * b(c)^2 + ...
72                  3 * Ber(7) * b(c)^2 * b(c + 1) + 3 * Ber(8) * (1 - b(c) - b(c + 1)) * b(c-1)^2 ...
73                  + 3 * Ber(9) * b(c) * b(c + 1)^2 + 6 * (1 - b(c) - b(c + 1)) * b(c) * b(c + 1);
74              if test > -10^-6 && test < 0
75                  fprintf('The algorithm failed.\n');
76                  return
77              elseif test < -10^-6
78                  fprintf('This polynomial is not positive on the standard simplex.\n');
79                  return
80              end
81          end
82
83          driehoek1 = [[(va(1) + vb(1)) / 2 (va(2) + vb(2)) / 2] va vc];
84          driehoek2 = [[(va(1) + vb(1)) / 2 (va(2) + vb(2)) / 2] vb vc];
85
86          if controleer(driehoek1, Ber) == 0
87              driehoeken(size(driehoeken, 1) + 1, :) = driehoek1;
88              stop = 1;
89          end
90
91          if controleer(driehoek2, Ber) == 0
92                  driehoeken(size(driehoeken, 1) + 1, :) = driehoek2;
93              stop = 1;
94          end
95
96      end
97
98      for i = 1 : a
99          driehoeken(1, :) = [];
100     end
101
102 end
103
104 fprintf('Number of times simplicial partition is used: %d\n', j);
```

Bibliography

[1] S. Bundfuss; M. Dür, *Algorithmic copositivity detection by simplicial partition*, Linear Algebra and its Applications **428** (2008), no. 7, 1511 – 1523.

[2] M. Zettler; J. Garloff, *Robustness analysis of polynomials with polynomial parameter debendency using bernstein expansion*, Automatic Control **43** (1998), no. 3, 425 – 431.

[3] B. Jüttler, *Arbitrarily weak linear convexity conditions for multivariate polynomials*, Studia Scientiarum Mathematicarum Hungarica **36** (2000), no. 1-2, 165 – 184.

[4] Richard Leroy, *Certificates of positivity and polynomial minimization in the multivariate bernstein basis*, Ph.D. thesis, University of Rennes 1, 2008.