

Division Algorithms for Bernstein Polynomials

Laurent Busé* Ron Goldman†

November 2, 2007

Abstract

Three division algorithms are presented for univariate Bernstein polynomials: an algorithm for finding the quotient and remainder of two univariate polynomials, an algorithm for calculating the GCD of an arbitrary collection of univariate polynomials, and an algorithm for computing a μ -basis for the syzygy module of an arbitrary collection of univariate polynomials. Division algorithms for multivariate Bernstein polynomials and analogues in the multivariate Bernstein setting of Gröbner bases are also discussed. All these algorithms are based on a simple ring isomorphism that converts each of these problems from the Bernstein basis to an equivalent problem in the monomial basis. This isomorphism allows all the computations to be performed using only the original Bernstein coefficients; no conversion to monomial coefficients is required.

1 Introduction

The Bernstein bases are the bases of choice for polynomials in Computer Aided Design. Univariate Bernstein bases are B-bases [2]; thus the univariate Bernstein bases are the most variation diminishing polynomial bases – their control polygons are closer to the corresponding polynomial curves than the control polygons relative to any other polynomial basis. Bezier curves and surfaces are simply polynomial curves and surfaces represented in the Bernstein bases.

Computations in the Bernstein bases are more stable than computations in other polynomial bases [11]. Moreover change of basis algorithms are themselves often numerically unstable. Therefore when polynomials are represented in the Bernstein bases, it is best to perform all calculations using only the Bernstein coefficients. Formulas for multiplication, differentiation, and subdivision of polynomials represented in Bernstein form are well known [12]. The purpose of this paper is to develop three additional algorithms for univariate polynomials represented in the Bernstein bases: an algorithm for dividing two univariate polynomials, an algorithm for calculating the GCD of an arbitrary collection of univariate polynomials, and an algorithm for computing a μ -basis for the syzygy module of an arbitrary collection of univariate polynomials. We shall also introduce division algorithms for multivariate Bernstein polynomials and we will provide as well analogues of Gröbner bases in the multivariate Bernstein setting.

*INRIA Sophia Antipolis

†Rice University

A *Gröbner basis* is a special basis for an ideal generated by a collection of polynomials in several variables. Gröbner bases have wide ranging applications in Algebraic Geometry and Geometric Modeling, including implicitizing rational surfaces, determining ideal membership, and solving systems of polynomial equations; see [8, 9] for further details.

In contrast, a μ -*basis* is a special basis for the syzygy module of a collection of polynomials in one variable. The notion of a μ -basis was first introduced into Geometric Modeling by Sederberg, Cox and their collaborators in order to investigate the properties of rational planar curves for use in Computer Aided Design [4, 10, 15, 17, 16]. Efficient algorithms for implicitization, inversion, and intersection of rational planar curves as well as straightforward procedures for the detection and analysis of their singular points have all been developed using μ -bases [10, 18]. Recently μ -bases have also been applied to study both nonplanar rational curves [19] and rational ruled surfaces [1, 3, 6, 7]. Straightforward algorithms to compute a μ -basis for the syzygy module of three univariate polynomials represented in the monomial basis are presented in [5, 20] and an extension of the algorithm in [5] to an arbitrary number of univariate polynomials in monomial form is presented in [19].

We shall begin our discussion of Bernstein polynomials in Section 2 with some background material on computations with univariate Bernstein polynomials. To derive division algorithms for Bernstein polynomials, we discuss in Section 3 the conceptual link between the Bernstein bases and the monomial bases, a link which allows us at least theoretically to transform any division algorithm in the monomial bases to a division algorithm in the Bernstein bases. As our first illustrations of this approach, we present in Section 4.1 a variant of Euclid's division algorithm along with a GCD algorithm for two univariate Bernstein polynomials. More advanced illustrations of this approach are presented in Section 4.2 where we provide an algorithm for computing a μ -basis for the syzygy module of an arbitrary collection of univariate Bernstein polynomials, and in Section 4.3 where we provide an algorithm for computing the GCD of an arbitrary collection of univariate Bernstein polynomials based on this μ -basis algorithm. In Section 5 we extend division algorithms from univariate Bernstein polynomials to multivariate Bernstein polynomials and we show as well how to generate the analogue of a Gröbner basis for the ideal generated by a collection of multivariate Bernstein polynomials. We conclude in Section 6 with a brief summary of our work.

2 Bernstein Bases Basics

The *univariate Bernstein basis functions of degree n* are defined by

$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k} \quad k = 0, \dots, n.$$

A *Bernstein polynomial $P(t)$* is a polynomial represented in the Bernstein basis:

$$P(t) = \sum_{k=0}^n c_k^n B_k^n(t).$$

Below we present some background material on computations with Bernstein polynomials. In particular, we provide formulas for multiplication and division of Bernstein polynomials

by powers of t or $1 - t$, and for degree elevation of Bernstein polynomials. For computational efficiency, we present these formulas in a way that avoids unnecessary growth of the size of the coefficients. Also to avoid large numerators and denominators, quotients of binomial coefficients are systematically replaced by fractions or by products of fractions with smaller numerators and denominators.

2.1 Multiplication and Division by Powers of t and $1 - t$

The main formulas we shall require for Bernstein polynomials are formulas for multiplication and division by powers of t and $1 - t$.

We begin with multiplication. Observe that for the Bernstein basis functions,

$$\begin{aligned} t^d B_k^n(t) &= \binom{n}{k} t^{k+d} (1-t)^{n-k} = \frac{\binom{n}{k}}{\binom{n+d}{k+d}} B_{k+d}^{n+d}(t), \\ (1-t)^d B_k^n(t) &= \binom{n}{k} t^k (1-t)^{n+d-k} = \frac{\binom{n}{k}}{\binom{n+d}{k}} B_k^{n+d}(t). \end{aligned}$$

Therefore if

$$P(t) = \sum_{k=0}^n c_k^n B_k^n(t),$$

then

$$\begin{aligned} t^d P(t) &= \sum_{k=0}^n \frac{\binom{n}{k}}{\binom{n+d}{k+d}} c_k^n B_{k+d}^{n+d}(t), \\ (1-t)^d P(t) &= \sum_{k=0}^n \frac{\binom{n}{k}}{\binom{n+d}{k}} c_k^n B_k^{n+d}(t). \end{aligned}$$

Thus we have the following formulas for multiplication by powers of t and $1 - t$:
Multiplication by Powers of t^d

$$c_k^{n+d} = \frac{\binom{n}{k-d}}{\binom{n+d}{k}} c_{k-d}^n = \left(\prod_{i=0}^{d-1} \frac{k-i}{n+d-i} \right) c_{k-d}^n. \quad (2.1)$$

Multiplication by Powers of $(1-t)^d$

$$c_k^{n+d} = \frac{\binom{n}{k}}{\binom{n+d}{k}} c_k^n = \left(\prod_{i=0}^{d-1} \frac{n+d-k-i}{n+d-i} \right) c_k^n. \quad (2.2)$$

Next we consider division. Suppose that for a given integer $j > 0$,

$$P(t) = \sum_{k=j}^n c_k^n B_k^n(t).$$

Since t^j divides $B_k^n(t)$ for all $k \geq j$, it follows that t^j divides $P(t)$. But for $k \geq j$,

$$\frac{B_k^n(t)}{t^j} = \binom{n}{k} t^{k-j} (1-t)^{n-k} = \frac{\binom{n}{k}}{\binom{n-j}{k-j}} B_{k-j}^{n-j}(t);$$

therefore

$$\frac{P(t)}{t^j} = \sum_{k=j}^n \frac{\binom{n}{k}}{\binom{n-j}{k-j}} c_k^n B_{k-j}^{n-j}(t).$$

Similarly, for $j \leq n - k$,

$$\frac{B_k^n(t)}{(1-t)^j} = \binom{n}{k} t^k (1-t)^{n-k-j} = \frac{\binom{n}{k}}{\binom{n-j}{k}} B_k^{n-j}(t).$$

Therefore if

$$P(t) = \sum_{k=0}^{n-j} c_k^n B_k^n(t),$$

then

$$\frac{P(t)}{(1-t)^j} = \sum_{k=0}^{n-j} \frac{\binom{n}{k}}{\binom{n-j}{k}} c_k^n B_k^{n-j}(t).$$

Thus we have the following formulas for removing common powers of t or $1 - t$:

Removing Common Powers of t^j

$$c_k^{n-j} = \frac{\binom{n}{k+j}}{\binom{n-j}{k}} c_{k+j}^n = \left(\prod_{i=0}^{j-1} \frac{n-i}{k+j-i} \right) c_{k+j}^n. \quad (2.3)$$

Removing Common Powers of $(1-t)^j$

$$c_k^{n-j} = \frac{\binom{n}{k}}{\binom{n-j}{k}} c_k^n = \left(\prod_{i=0}^{j-1} \frac{n-i}{n-k-i} \right) c_k^n. \quad (2.4)$$

2.2 Degree Elevation

Given a polynomial represented in the univariate Bernstein basis of degree n , degree elevation computes representations of the same polynomial in the univariate Bernstein bases of degree greater than n . Degree elevation facilitates adding two or more Bernstein polynomials that are not represented in the same degree Bernstein bases.

We give the formula for *degree one elevation*; arbitrary degree elevation is computed by iteration. In this way, we avoid unnecessary growth of the size of the coefficients due to compact formulas involving binomial coefficients. We begin with the Bernstein basis functions. Observe that

$$tB_k^n(t) = \binom{n}{k} t^{k+1} (1-t)^{n-k} = \frac{\binom{n}{k}}{\binom{n+1}{k+1}} B_{k+1}^{n+1}(t) = \frac{k+1}{n+1} B_{k+1}^{n+1}(t),$$

$$(1-t)B_k^n(t) = \binom{n}{k} t^k (1-t)^{n+1-k} = \frac{\binom{n}{k}}{\binom{n+1}{k}} B_k^{n+1}(t) = \frac{n+1-k}{n+1} B_k^{n+1}(t).$$

Adding these two equations gives the degree elevation formula for the Bernstein basis functions:

$$B_k^n(t) = \frac{n+1-k}{n+1} B_k^{n+1}(t) + \frac{k+1}{n+1} B_{k+1}^{n+1}(t).$$

Therefore if

$$P(t) = \sum_{k=0}^n c_k^n B_k^n(t),$$

then

$$\begin{aligned} tP(t) &= \sum_{k=j}^n c_k^n t B_k^n(t) = \sum_{k=0}^n \frac{k+1}{n+1} c_k^n B_{k+1}^{n+1}(t), \\ (1-t)P(t) &= \sum_{k=j}^n c_k^n (1-t) B_k^n(t) = \sum_{k=0}^n \frac{n+1-k}{n+1} c_k^n B_k^{n+1}(t). \end{aligned}$$

Adding these two equations gives the degree elevation formula for Bernstein polynomials:

$$P(t) = \sum_{k=0}^n \left(\frac{n+1-k}{n+1} c_k^n + \frac{k}{n+1} c_{k-1}^n \right) B_k^{n+1}(t),$$

that is

$$c_k^{n+1} = \frac{n+1-k}{n+1} c_k^n + \frac{k}{n+1} c_{k-1}^n. \quad (2.5)$$

3 Conversion Between Monomial and Bernstein Bases

The *univariate monomial basis functions of degree n* are given by t^k , $k = 0, \dots, n$. Formulas to pass between the monomial basis and Bernstein basis of a given degree are easy to derive [12], but we will not use these formulas here because our goal is to perform all the computations in the Bernstein basis. Nevertheless, understanding the conceptual link between these two polynomial bases will help us to derive division algorithms in the Bernstein basis.

Key to understanding this conceptual link is the notion of degree. When we are given a particular Bernstein representation for a polynomial $P(t)$, we shall write $\deg(P)$ for the degree of this Bernstein representation, whereas we shall write $\deg_t(P)$ for the degree of $P(t)$ in the monomial representation. Notice that $\deg(P) \geq \deg_t(P)$ because the Bernstein representation of $P(t)$ may be degree elevated. For instance, consider $3t = \sum_{k=0}^3 k B_k^3(t)$. In this example, $\deg(3t) = 3$, whereas $\deg_t(3t) = 1$. We call $\deg(P)$ the *naive degree* of a polynomial represented in the Bernstein basis.

One way to unify these two notions of degree is to pass to the homogeneous setting. Let s denote a new homogenizing variable. The *homogeneous monomial basis of degree n* is $t^k s^{n-k}$, $k = 0, \dots, n$; similarly, the *homogeneous Bernstein basis of degree n* is $B_k^n(t, s) = \binom{n}{k} t^k (s-t)^{n-k}$, $k = 0, \dots, n$. Now let *degree* mean the total degree with respect to s and t . From this perspective, consider the ring homomorphism

$$\begin{aligned} \phi : \mathbb{R}[t, s] &\rightarrow \mathbb{R}[t, s] \\ t &\mapsto t \\ s &\mapsto s - t. \end{aligned} \quad (3.1)$$

It is easy to see that ϕ is an isomorphism; its inverse ψ sends s to $s + t$ and leaves t invariant. Moreover, if we grade the ring $\mathbb{R}[t, s]$ by setting $\deg(t) = \deg(s) = 1$, then ϕ becomes a *graded* isomorphism – that is ϕ sends a homogeneous polynomial of degree d to a homogeneous

polynomial of degree d . Therefore, for all integers d , the maps ϕ and ψ provide a correspondence between homogeneous polynomials of degree d in Bernstein form and homogeneous polynomials of degree d in monomial form:

$$\psi \left(\sum_{i=0}^d a_i B_i^d(t, s) \right) = \sum_{i=0}^d a_i \binom{d}{i} t^i s^{d-i} \quad \text{and} \quad \phi \left(\sum_{j=0}^d b_j t^j s^{d-j} \right) = \sum_{j=0}^d \frac{b_j}{\binom{d}{j}} B_j^d(t, s).$$

This correspondence sheds light on the way to transform known division algorithms in the monomial basis to division algorithms in the Bernstein basis: any polynomial relation between a collection of Bernstein polynomials f_1, \dots, f_n , corresponds to a polynomial relation between the polynomials $\psi(f_1), \dots, \psi(f_n)$ in the monomial bases, and vice-versa. Notice that the polynomials g and $\phi(g)$ are, in general, distinct – that is, they are not the same polynomials represented in different bases, rather they are really different polynomials.

After dehomogenization, the maps ψ and ϕ can be interpreted as the rational linear change of parameters $u = t/(1-t)$. Indeed, given a Bernstein polynomial $f(t) = \sum_{i=0}^d a_i B_i^d(t)$, the map ψ associates to $f(t)$ the polynomial in u given by

$$\frac{f(t)}{(1-t)^d} = \sum_{i=0}^d a_i \binom{d}{i} \frac{t^i}{(1-t)^i} = \sum_{i=0}^d a_i \binom{d}{i} u^i.$$

Similarly, given a polynomial $g(u) = \sum_{j=0}^d b_j u^j$, the map ϕ associates to $g(u)$ the Bernstein polynomial in t given by

$$(1-t)^d g \left(\frac{t}{1-t} \right) = \sum_{j=0}^d \frac{b_j}{\binom{d}{j}} B_j^d(t).$$

It follows that the correspondence (3.1) remains a correspondence after dehomogenization if and only if we restrict to polynomials in the monomial basis with a non-zero coefficient of t^d and to Bernstein polynomials with a non-zero coefficient of $B_d^d(t)$.

Therefore, one possible way to generate division algorithms for Bernstein polynomials is simply to divide each Bernstein polynomial $f_i(t)$ by $(1-t)^{\deg(f_i)}$ and apply the substitution $u = t/(1-t)$ to convert to a polynomial in monomial form. One can then use a division algorithm we already know in the monomial basis (in the variable u) and reverse this process to convert back from the result in the monomial basis to get a result in the Bernstein basis. However, proceeding this way, we have to build polynomials in the monomial basis that may have very large coefficients compared to the original Bernstein coefficients. For instance, the Bernstein polynomial $f(t) = \sum_{i=0}^d B_i^d(t)$ maps to the polynomial $\psi(f) = \sum_{i=0}^d \binom{d}{i} u^i$: as n increases, the coefficients of $\psi(f)$ increase very quickly but the coefficients of f remain constant. Moreover, the same problem occurs when converting back from the monomial basis to the Bernstein basis but this time with division by large integers.

Consequently, we shall not invoke the substitution $u = t/(1-t)$. Rather, we will use the correspondence in (3.1) as a guideline to interpret division algorithms purely in terms of Bernstein coefficients. In particular, this correspondence suggests that we should define the *leading coefficient* of a Bernstein polynomial of degree d as the coefficient of $B_d^d(t)$.

4 Division Algorithms

We are now ready to formally present our division algorithms for univariate Bernstein polynomials. We begin with the division algorithm for two univariate Bernstein polynomials. Next we present an algorithm for the GCD of two univariate Bernstein polynomials based on this division algorithm. We then go on to develop an algorithm for generating a μ -basis for the syzygy module of an arbitrary collection of univariate Bernstein polynomials. The GCD algorithm for three or more univariate Bernstein polynomials is a consequence of this μ -basis algorithm for Bernstein polynomials.

In the illustrative examples, we will use the following notation. Let

$$g(t) = \sum_{k=0}^n c_k^n B_k^n(t).$$

When we want to emphasize the Bernstein coefficients, we shall write

$$g(t) = [c_n^n, \dots, c_0^n].$$

4.1 Division and GCD Algorithms for Two Bernstein Polynomials

Given two univariate polynomials $f(t), g(t)$ such that $1 \leq \deg_t(f) \leq \deg_t(g)$, the classical Euclidian algorithm for division of g by f returns two uniquely determined polynomials: the quotient $q(t)$ and the remainder $r(t)$ that satisfy $g = qf + r$ with $\deg_t(q) = \deg_t(g) - \deg_t(f)$ and $\deg_t(r) < \deg_t(f)$. The way this result is stated actually depends implicitly on the usual representation of polynomials in the monomial basis. Recalling the discussion in Section 3, we introduce a new homogenizing variable s and denote by $f^h(t, s)$ and $g^h(t, s)$ the homogenization of $f(t)$ and $g(t)$ – that is, we set

$$f^h(t, s) := s^{\deg_t(f)} f\left(\frac{t}{s}\right) \in \mathbb{R}[t, s], \quad g^h(t, s) := s^{\deg_t(g)} g\left(\frac{t}{s}\right) \in \mathbb{R}[t, s].$$

Note that $f^h(1, 0) \neq 0$ and $g^h(1, 0) \neq 0$ – that is f and g do not vanish at infinity. Now the Euclidian division algorithm corresponds to the following equality between homogeneous polynomials of degree $\deg_t(g)$:

$$g^h(t, s) = q(t, s)f^h(t, s) + s^{\deg_t(g) - \deg_t(f) + 1} r(t, s), \quad (4.1)$$

where $q(t, s)$ and $r(t, s)$ are uniquely determined homogeneous polynomials. Observe it is possible that $r(1, 0) = 0$, so $\deg_t(r(t, 1))$ may be strictly less than $\deg_t(f) - 1$, although the homogeneous degree of $r(t, s)$ is $\deg_t(f) - 1$. Using the correspondence in (3.1) and dehomogenizing the polynomials by setting s to 1, leads to a formula for division in the Bernstein basis.

Proposition 4.1 *Let $f(t)$ and $g(t)$ be two univariate polynomials given in Bernstein form*

$$f(t) = \sum_{i=0}^d a_i B_i^d(t), \quad g(t) = \sum_{i=0}^e b_i B_i^e(t)$$

with $a_d = f(1) \neq 0$, $b_e = g(1) \neq 0$, and $e \geq d \geq 0$. Then there exist two uniquely determined univariate polynomials $q(t)$ and $r(t)$ such that

$$g(t) = q(t)f(t) + (1-t)^{e-d+1}r(t) \quad (4.2)$$

where $q(t) = \sum_{i=0}^{e-d} q_i B_i^{e-d}(t)$ and $r(t) = \sum_{i=0}^{d-1} r_i B_i^{d-1}(t)$.

(Observe that here, as in Equation (4.1), it is possible that $r(1) = 0$.)

Interpreting Euclid's classical division algorithm through the correspondence in (3.1), we get an algorithm to compute the quotient and remainder in Equation (4.2). Before presenting this algorithm, we explain why we focus on the case described in Proposition 4.1 where $f(1) \neq 0$ and $g(1) \neq 0$.

If $f(1) = 0$ or $g(1) = 0$, then define c_f and c_g to be the integers such that

$$f^*(t) := \frac{f(t)}{(1-t)^{c_f}} \in \mathbb{R}[t], \quad g^*(t) := \frac{g(t)}{(1-t)^{c_g}} \in \mathbb{R}[t]$$

and $f^*(1) \neq 0$ and $g^*(1) \neq 0$. Let $\deg(h)$ denote the naive Bernstein degree of a Bernstein polynomial h , in contrast to the actual monomial degree $\deg_t(h)$ of h . If $\deg(g^*) < \deg(f^*)$, then the equality

$$g(t) = 0 \times f(t) + (1-t)^{c_g} g^*(t)$$

is the analogue of Equation (4.2), since

$$c_g > c_g - c_f - (\deg(f^*) - \deg(g^*)) = \deg(g) - \deg(f).$$

If $\deg(g^*) \geq \deg(f^*)$, then we can compute the polynomials $r(t)$ and $q(t)$ in Equation (4.2),

$$g^*(t) = q(t)f^*(t) + (1-t)^{\deg(g^*) - \deg(f^*) + 1} r(t),$$

and multiplying this equation by $(1-t)^{\max(c_f, c_g)}$ we get a division formula. Observe however that if $c_g < c_f$ we only get a division formula of the form

$$(1-t)^{c_f - c_g} g(t) = q(t)f(t) + (1-t)^{c_f + \deg(g^*) - \deg(f^*) + 1} r(t).$$

INPUT: Two Bernstein polynomials $f(t)$ and $g(t)$ such that $f(1) \neq 0$, $g(1) \neq 0$, and $1 \leq d := \deg(f) \leq e := \deg(g)$. Denote by $a \neq 0$ the coefficient of $B_d^d(t)$ in $f(t)$.

ALGORITHM:

1. Define the two Bernstein polynomials $r(t) := g(t)$ with $\deg(r) = e$ and $q(t) := 0$ with $\deg(q) = e - d$.
2. While $\deg(r) \geq \deg(f)$
 - (a) Set the coefficient of $B_{\deg(r)-d}^{e-d}(t)$ in $q(t)$ to

$$\frac{b}{\binom{e-d}{\deg(r)-d} a}$$

where $r(t) = bB_{\deg(r)}^{\deg(r)}(t) + \dots$ with $b \neq 0$.

- (b) Replace $r(t)$ by $r(t) - \frac{b}{a}t^{\deg(r)-d}f(t)$.
- (c) Remove all common powers of $1 - t$ from $r(t)$ (which reduces the degree of r)

OUTPUT: Two Bernstein polynomials $q(t)$ and $r(t)$ satisfying Equation (4.2); more precisely we have

$$g(t) = q(t)f(t) + (1 - t)^{e-\deg(r)}r(t)$$

where $\deg(q) = \deg(g) - \deg(f)$, $\deg(r) < \deg(f)$, and $r(1) \neq 0$.

Remark 4.2 Notice that step 2(a) can be skipped if one's only aim is to compute the remainder $r(t)$.

Example 4.3 Consider the following simple example:

$$g(t) = \left[2, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0\right] = t^4 + t, \quad f(t) = [1, 0, 0, 0] = t^3.$$

Notice that the leading coefficient of f is 1 and the leading coefficient of g is 2; therefore in the division algorithm $a = 1$ and $b = 2$. We begin with the initialization

$$q(t) = [0, 0] = 0, \quad r(t) = g(t) = \left[2, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0\right].$$

In the first iteration of the while loop we have $b = 2$ and hence the first coefficient of $q(t)$ is set to 2. Also

$$r(t) = \left[2, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0\right] - [2, 0, 0, 0, 0] = \left[0, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0\right]$$

so after division by $1 - t$

$$r(t) = \left[3, 1, \frac{1}{3}, 0\right].$$

Therefore, for the second iteration of the while loop we have $b = 3$ and the second coefficient of $q(t)$ is set to 3: hence

$$q(t) = [2, 3] = 3 - t.$$

Next we have

$$r(t) = \left[3, 1, \frac{1}{3}, 0\right] - [3, 0, 0, 0] = \left[0, 1, \frac{1}{3}, 0\right]$$

which after reduction yields $r(t) = 2t^2 + t = [3, 1/2, 0]$. Since $\deg(r) < \deg(f)$, the algorithm terminates and we have

$$g(t) = t^4 + t = (3 - t) \times t^3 + (1 - t)^2 \times (2t^2 + t) = q(t)f(t) + (1 - t)^2 r(t).$$

Notice that, in general, when we divide $g(t)$ by $f(t)$ in the Bernstein basis, the quotient and the remainder are not equal as polynomials to the quotient and the remainder that we get when we divide $g(t)$ by $f(t)$ in the monomial basis (see Example 4.3). Moreover, Equation (4.2) and the division algorithm for Bernstein polynomials are valid even in the case where $\deg_t(g) < \deg_t(f)$. For instance, if we divide $g(t) = t$ by $f(t) = t^2$ using the Bernstein representations

$$g(t) = B_3^3(t) + \frac{2}{3}B_2^3(t) + \frac{1}{3}B_1^3(t), \quad f(t) = B_2^2(t),$$

then we get

$$g(t) = t = (2 - t)t^2 + (1 - t)^2t.$$

Nevertheless, this division algorithm for two Bernstein polynomials leads immediately to the following GCD algorithm for two Bernstein polynomials.

INPUT: Two Bernstein polynomials $f(t)$ and $g(t)$ such that $1 \leq \deg(f) \leq \deg(g)$.

ALGORITHM:

1. Determine how many common factors c there are of $1 - t$ in f and g .
2. Remove all common powers of $1 - t$ from f and from g so that $f(1) \neq 0$ and $g(1) \neq 0$.
3. Exchange f and g if $\deg(f) > \deg(g)$.
4. While $\deg(f) > 0$
 - (a) Compute the remainder r of the division of g by f using the division algorithm for Bernstein polynomials.
 - (b) Replace $g \leftarrow f$ and $f \leftarrow r$.
5. If $f \neq 0$ set $d(t) = 1$, otherwise set $d(t) := g(t)$.

OUTPUT: A Bernstein polynomial $d(t)$ and an integer c such that $\text{GCD}(f, g) = (1 - t)^c d(t)$.

Example 4.4 *We consider the following example:*

$$g(t) = [0, 0, 1, 0, 0, 0] = 10t^3 - 20t^4 + 10t^5, \quad f(t) = \left[0, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0\right] = t - t^4.$$

The first step of the GCD algorithm is to compute the integer c , the highest common factor in f and g of $1 - t$, which here is equal to 1, and then to reduce both g and f with respect to the factor $1 - t$. We get

$$g(t) = \left[-10, -\frac{5}{3}, 0\right] = \frac{10t^3 - 20t^4 + 10t^5}{(1 - t)^2}, \quad f(t) = \left[3, 1, \frac{1}{3}, 0\right] = \frac{t - t^4}{1 - t}.$$

Since $\deg(f) > \deg(g)$, we reverse the roles of f and g – that is, we set $g \leftarrow f$ and $f \leftarrow g$. Next the remainder of the division of $g(t)$ by $f(t)$ is found to be $[1/3, 0]$, so we are now left with

$$g(t) = \left[-10, -\frac{5}{3}, 0\right] = -\frac{20}{3}t^2 - \frac{10}{3}t, \quad f(t) = \left[\frac{1}{3}, 0\right] = \frac{1}{3}t.$$

Finally dividing $g(t)$ by $f(t)$ returns a zero remainder, which implies that

$$\text{GCD}(f, g) = (1 - t) \times \left[\frac{1}{3}, 0\right] = \left[0, \frac{1}{6}, 0\right] = \frac{1}{3}t(1 - t).$$

Notice that although the result of division is not the same in the Bernstein basis and the monomial basis, the GCD that emerges from the Bernstein GCD algorithm is the same polynomial, up to multiplication by a non-zero constant, as the GCD computed using Euclid's algorithm in the monomial basis. The GCD of two polynomials is independent of the basis used in the computation, whereas the quotient and the remainder evidently depend on the choice of basis.

4.2 A μ -Basis Algorithm for an Arbitrary Collection of Bernstein Polynomials

Let $f(t) := (f_1(t), \dots, f_k(t))$ be a collection of univariate polynomials in $\mathbb{R}[t]$, the ring of univariate polynomials in t with real coefficients. The *syzygy module* is

$$\text{syz}(f_1(t), \dots, f_k(t)) := \left\{ p(t) = (p_1(t), \dots, p_k(t)) \mid p(t) \cdot f(t) := \sum_{i=1}^k p_i(t) f_i(t) = 0 \right\} \subset \mathbb{R}[t]^k.$$

The set $\text{syz}(f_1(t), \dots, f_k(t))$ is known to be a free module with $k - 1$ generators over $\mathbb{R}[t]$: this result is a consequence of Hilbert's Syzygy Theorem [9, Chapter 6, §2] – see also [14, 19] and [13] for Hilbert's original paper.

There are several ways to define the notion of a μ -basis. Recalling our discussion in Section 3, we choose a definition that relies on the homogeneous setting and refer the reader to [19] for further details. Introducing the homogenizing variable s , we will denote the homogenization of any polynomial $P(t) \in \mathbb{R}[t]$ by

$$P^h(t, s) := s^{\deg_t(P)} P\left(\frac{t}{s}\right) \in \mathbb{R}[t, s].$$

Now for all integers $i = 1, \dots, k$, we denote by $\bar{f}_i(t, s)$ the homogenization of $f_i(t)$ of degree $d := \max_i(\deg_t(f_i))$, that is we set

$$\bar{f}_i(t, s) := s^{d - \deg_t(f_i)} f_i^h(t, s) \in \mathbb{R}[t, s], \quad i = 1, \dots, k.$$

The syzygy module of $\bar{f}(t, s) := (\bar{f}_1(t, s), \dots, \bar{f}_k(t, s))$,

$$\text{syz}(\bar{f}_1(t, s), \dots, \bar{f}_k(t, s)) := \{p(t, s) = (p_1(t, s), \dots, p_k(t, s)) \mid p(t, s) \cdot \bar{f}(t, s) = 0\} \subset \mathbb{R}[t, s]^k,$$

is also a free module of rank $k - 1$, but is moreover a *graded* module over $\mathbb{R}[t, s]$.

Proposition 4.5 *The vectors of polynomials $u_1(t), \dots, u_{k-1}(t)$ form a μ -basis of $\text{syz}(f(t))$ if and only if the vectors of homogeneous polynomials $\bar{u}_1(t, s), \dots, \bar{u}_{k-1}(t, s)$ form a basis of $\text{syz}(\bar{f}(t, s))$. Moreover, in this case we have*

$$\sum_{i=1}^{k-1} \max(\deg_t(u_i(t))) = \max(\deg_t(f_1), \dots, \deg_t(f_k)) - \deg_t(\text{GCD}(f_1, \dots, f_k)) \quad (4.3)$$

where $\deg_t(u_i(t))$ denotes the maximum of the degrees of the components of the vector of polynomials $u_i(t)$ with respect to the variable t .

This result is a standard consequence of the Hilbert-Burch Theorem and some basic properties of graded modules; see for instance [9, Chapter 6] and Exercise 17 therein. From a computational point of view, an important corollary of this result is that $u_1(t), \dots, u_{k-1}(t)$ is a μ -basis of $\text{syz}(f(t))$ if and only if the *leading coefficient vectors* of $u_1(t), \dots, u_{k-1}(t)$, that is the vectors $\bar{u}_1(1, 0), \dots, \bar{u}_{k-1}(1, 0)$, are linearly independent (see [19]). This is the key property needed to validate the algorithm for computing a μ -basis for polynomials in the monomial basis [19], and hence in the Bernstein basis through the correspondence in (3.1).

Our procedure for constructing a μ -basis for $\text{syz}(f_1(t), \dots, f_k(t))$ will be to start with a collection of generators for $\text{syz}(f_1(t), \dots, f_k(t))$ and then to systematically reduce degrees and eliminate extraneous elements until we are left with a μ -basis. To proceed, we need a simple set of generators and an easy method for reducing degree. The obvious syzygies

$$(0, \dots, 0, -f_j(t), 0, \dots, 0, f_i(t), 0, \dots, 0) \quad 1 \leq i < j \leq k$$

are a simple set of generators of the free $\mathbb{R}[t]$ -module

$$\text{GCD}(f_1, \dots, f_k) \cdot \text{syz}(f_1, \dots, f_k), \quad (4.4)$$

see [14, 19].

To reduce degrees and eliminate extraneous elements, suppose that $S_1(t), \dots, S_m(t)$ are a collection of generators of $\text{syz}(f_1(t), \dots, f_k(t))$ polynomially dependent over the ring $\mathbb{R}[t]$. Then there are polynomials $p_1(t), \dots, p_m(t)$ such that

$$p_1(t)S_1(t) + \dots + p_m(t)S_m(t) \equiv 0. \quad (4.5)$$

We can assume that $1 - t$ is not a factor of $p_j(t)$ for all $j = 1, \dots, m$; otherwise we can simply remove this common factor from each polynomial $p_j(t)$ and Equation (4.5) would remain valid. Let

$$S(t) = p_1(1)S_1(t) + \dots + p_m(1)S_m(t). \quad (4.6)$$

Then by Equation (4.5), $S(1) = 0$, so $(1 - t) \mid S(t)$. Now to reduce the degree of one of the syzygies, find the syzygy $S_h(t)$ of highest degree among $S_1(t), \dots, S_m(t)$ with $p_h(1) \neq 0$, and set

$$S_h(t) \leftarrow \frac{S(t)}{1 - t}.$$

This new set of syzygies is still a collection of generators for $\text{syz}(f_1(t), \dots, f_k(t))$, but one of the syzygies has lower degree than one of the syzygies in the original collection. If we iterate this procedure, we must eventually arrive at a basis with $k - 1$ generators. We can then invoke the correspondence in (3.1) and the algorithm in [19] to conclude that this basis is a μ -basis.

Notice that we do not need to find the polynomials $p_1(t), \dots, p_m(t)$; to compute $S(t)$ all we need are the constants $p_1(1), \dots, p_m(1)$. But by Equation (4.5)

$$p_1(1)S_1(1) + \dots + p_m(1)S_m(1) \equiv 0.$$

Thus the constant vectors $S_1(1), \dots, S_m(1)$ are linearly dependent. Hence to find the constant scalars $p_1(1), \dots, p_m(1)$, we need only find a relation between the constant vectors $S_1(1), \dots, S_m(1)$, which is easily done using standard techniques from linear algebra.

Before proceeding with our algorithm, notice that the μ -basis of a single non-zero polynomial $f_1(t)$ is empty, since the map

$$\mathbb{R}[t] \rightarrow \mathbb{R}[t] : p(t) \mapsto p(t)f_1(t)$$

is injective. Also, the μ -basis of two polynomials $f_1(t), f_2(t)$ is generated by $(-\tilde{f}_2(t), \tilde{f}_1(t))$ with $\tilde{f}_i(t) := f_i(t)/\text{GCD}(f_1, f_2)$. Finally, notice that if u_1, \dots, u_{k-1} is a μ -basis of the module $\text{syz}(f_1, \dots, f_k)$, then

$$(1, 0, \dots, 0), (0, u_1), \dots, (0, u_{k-1})$$

is a μ -basis of $\text{syz}(0, f_1, \dots, f_k)$. Therefore, we will restrict our attention to the case of $k \geq 3$ non-zero polynomials.

INPUT: A collection of $k \geq 3$ non-zero Bernstein polynomials $f_1(t), \dots, f_k(t)$ with $\text{GCD}(f_1, \dots, f_k) = 1$.

ALGORITHM:

1. Construct a list $S = \{S_1(t), \dots, S_s(t)\}$ containing the $s = \binom{k}{2}$ obvious syzygies. Since $\text{GCD}(f_1, \dots, f_k) = 1$, this list contains a system of generators for $\text{syz}(f_1(t), \dots, f_k(t))$.
2. For each element in S , remove the common powers of $1 - t$.
3. While $s > k - 1$
 - (a) Compute constants c_1, \dots, c_s such that $c_1 S_1(1) + \dots + c_s S_s(1) = 0$ (recall that $S_i(1)$ can be read directly from the Bernstein form). Observe that such constants always exist since $s > k - 1$, and that these constants can easily be computed with standard exact linear algebra routines.
 - (b) Find the integer p such that $\deg(S_p)$ is maximal with $c_p \neq 0$.
 - (c) Replace

$$S_p(t) \leftarrow \sum_{\substack{i=1, \dots, s \\ c_i \neq 0}} c_i t^{\deg(S_p) - \deg(S_i)} S_i(t).$$

Notice that the degree elevation procedure described in Section 2.2 is used here to add polynomials that are, in general, not expressed in Bernstein bases of the same degree – see the illustrative example below.

- (d) Remove common powers of $1 - t$ from $S_p(t)$ ($1 - t$ appears at least once).
- (e) Remove S_p from S if S_p is the null vector.

OUTPUT: A μ -basis $S_1(t), \dots, S_{k-1}(t)$ of $\text{syz}(f_1(t), \dots, f_k(t))$ in the Bernstein basis.

Remark 4.6 *In step 3(c), we could also replace $S_p(t)$ by $\sum_{i=1, \dots, s} c_i S_i(t)$ using the degree elevation procedure described in Section 2.2 to perform the addition. However, an immediate comparison of Equations (2.1) and (2.5) shows that multiplication by powers of t is faster than degree elevation. Therefore, as much as possible we choose to use multiplication by powers of t rather than degree elevation.*

To prove that this algorithm terminates with a μ -basis, we use the correspondence in (3.1). Indeed, passing this algorithm through the map ψ gives exactly the algorithm in [19] to compute a μ -basis of a collection of polynomials in the monomial basis. Moreover, since ϕ is a graded isomorphism, ϕ sends a basis of $\text{syz}(\psi(\bar{f}_1), \dots, \psi(\bar{f}_k))$ to a basis of $\text{syz}(\bar{f}_1, \dots, \bar{f}_k)$.

Example 4.7 *We consider the following illustrative example:*

$$f_1(t) = [1] = 1, \quad f_2(t) = [1, 0] = t, \quad f_3(t) = [1, 0, 0] = t^2.$$

At the beginning of the algorithm the three obvious generators for $\text{syz}(f_1, f_2, f_3)$ are:

$$S_1(t) = ([1, 0, 0], [0], [-1]), \quad S_2(t) = ([1, 0], [-1], [0]), \quad S_3(t) = ([0], [1, 0, 0], [-1, 0]).$$

In the first iteration of the while loop we have to compute an element in the kernel of the matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{pmatrix}$$

which is constructed by taking the first coefficient in each component of each of the vectors S_1, S_2 and S_3 . Choosing the vector

$$(c_1 = 1 \quad c_2 = -1 \quad c_3 = -1),$$

we replace S_1 by

$$S_1 - tS_2 - S_3 = \left([0, 0, 0], \left[0, \frac{1}{2}, 0 \right], \left[0, -\frac{1}{2}, -1 \right] \right).$$

Notice that we must use degree elevation to perform this addition. Now we remove common powers of $1 - t$ and get

$$S_1 \leftarrow ([0, 0], [1, 0], [-1, -1]).$$

In the second iteration of the while loop ($c_1 = 1, c_2 = 0, c_3 = -1$); hence we replace S_3 by

$$tS_1 - S_3 = ([0, 0, 0], [0, 0, 0], [0, 0, 0]) = ([0], [0], [0]).$$

Therefore, the algorithm terminates here and returns the following μ -basis for $\text{syz}(f_1, f_2, f_3)$:

$$(([0, 0], [1, 0], [-1, -1]), ([1, 0], [-1], [0])) = ((0, t, -1), (t, -1, 0)).$$

4.3 A GCD Algorithm for Three or More Bernstein Polynomials

If $\deg_t(\text{GCD}(f_1, \dots, f_k)) > 0$, then Equation (4.4) implies that the μ -basis algorithm in Section 4.2 begins with a set of generators for $\text{GCD}(f_1, \dots, f_k) \cdot \text{syz}(f_1, \dots, f_k)$. Moreover, by construction, $\text{GCD}(f_1, \dots, f_k)$ will remain in the generators in each step of the μ -basis algorithm. Therefore, the output of this μ -basis algorithm will be $\text{GCD}(f_1, \dots, f_k) \times \mu$ -basis.

We can take advantage of this observation to compute $\text{GCD}(f_1(t), \dots, f_k(t))$ for univariate Bernstein polynomials. We begin by finding and removing common powers of $1 - t$, which is easy to do for polynomials represented in the Bernstein basis. Any other factors of $\text{GCD}(f_1(t), \dots, f_k(t))$ will remain in the generators constructed in each stage of this μ -basis algorithm. Hence each element in the output of this μ -basis algorithm will also contain these common factors. Therefore iterating this μ -basis algorithm on any element of the output of this μ -basis algorithm, for instance the basis element with the lowest degree, will eventually generate these common factors.

INPUT: A collection of $k \geq 3$ univariate Bernstein polynomials $f_1(t), \dots, f_k(t)$.

ALGORITHM:

1. Determine the largest integer c such that $(1 - t)^c$ is a common factor of $f_1(t), \dots, f_k(t)$.
2. Remove the common factor $(1 - t)^c$ from all the polynomials $f_1(t), \dots, f_k(t)$.

3. Apply the algorithm in Section 4.2 to get a list S of $k - 1$ polynomially independent vectors that generate the free module $\text{GCD}(f_1, \dots, f_k) \cdot \text{syzy}(f_1, \dots, f_k)$.
4. Choose a vector f in S with minimal Bernstein degree. The vector f is composed of k univariate polynomials whose GCD is equal to $\text{GCD}(f_1, \dots, f_k)$.
5. Remove the null components from f .
6. Set d to the Bernstein degree of f , that is the maximum Bernstein degree of the components of f .
7. While $d > 0$ and the number of components of f is greater than or equal to 3
 - (a) Compute the result S of the μ -basis algorithm in Section 4.2 applied to f .
 - (b) Replace f by a vector in S of minimal Bernstein degree.
 - (c) Delete the null components of f .
 - (d) If the Bernstein degree of f is equal to d , then set d to 0; otherwise set d to the Bernstein degree of f .
8. If the number of components of f is 2 (observe that the number of components cannot be less than 2), then return $d(t)$, the GCD of these two components, by applying the GCD algorithm in Section 4.1; otherwise return one non-zero component $d(t)$ of f .

OUTPUT: A Bernstein polynomial $d(t)$ and an integer c such that $\text{GCD}(f_1(t), \dots, f_k(t)) = (1 - t)^c d(t)$.

This algorithm is guaranteed to terminate because, by construction, at each iteration of the algorithm the sum of the degrees of the output polynomials is strictly decreasing. This property fails to hold only when all the functions h_i at the start of the iteration are scalar multiples of $\text{GCD}(h_1, \dots, h_l)$, but in this case we terminate the algorithm.

Example 4.8 *We illustrate our GCD algorithm with the following example:*

$$f_1(t) = t = [1, 0], f_2(t) = t^2 = [1, 0, 0], f_3(t) = t^3 = [1, 0, 0, 0].$$

Applying the μ -basis algorithm in Section 4.2, we get the following two vectors

$$\left([-1, 0, 0], \left[2, \frac{1}{2}, 0 \right], [-1, 0] \right), ([-1, 0, 0], [1, 0], [0]). \quad (4.7)$$

We then deduce that $\text{GCD}(f_1, f_2, f_3)$ is equal to $\text{GCD}([-1, 0, 0], [1, 0])$. Applying the GCD algorithm in Section 4.1, we find that $\text{GCD}(f_1, f_2, f_3) = [1, 0] = t$.

5 Extension to Multivariate Bernstein Polynomials

The correspondence in (3.1) between the univariate monomial bases and the univariate Bernstein bases is easily extended to the multivariate setting. For instance, bivariate Bernstein polynomials are of the form

$$f(s, t) = \sum_{0 \leq i, j; i+j \leq d} a_{i,j} B_{i,j}^d(s, t) = \sum_{0 \leq i, j; i+j \leq d} a_{i,j} \frac{d!}{i!j!(d-i-j)!} s^i t^j (1-s-t)^{d-i-j}$$

and, introducing a new indeterminate u , their homogenization is

$$f^h(s, t, u) = \sum_{0 \leq i, j; i+j \leq d} a_{i,j} B_{i,j}^d(s, t, u) = \sum_{0 \leq i, j; i+j \leq d} \frac{d!}{i!j!(d-i-j)!} s^i t^j (u-s-t)^{d-i-j}.$$

Thus bivariate Bernstein polynomials correspond to homogeneous polynomials in the monomial basis of the same degree through the graded isomorphism

$$\begin{aligned} \phi : \mathbb{R}[s, t, u] &\rightarrow \mathbb{R}[s, t, u] & (5.1) \\ s &\mapsto s \\ t &\mapsto t \\ u &\mapsto u - s - t. \end{aligned}$$

Let ψ denote the inverse of ϕ . Then the maps ϕ and ψ can be used to derive division algorithms for bivariate Bernstein polynomials from standard division algorithms for polynomials expressed in the monomial basis.

Instead of providing all the details of such a theory, we simply give an example that should clarify the situation. Suppose that we want to divide the Bernstein polynomial

$$f(s, t) = B_{2,1}^3(s, t) + B_{1,2}^3(s, t) + B_{0,2}^3(s, t)$$

by the two Bernstein polynomials

$$g_1(s, t) = B_{1,1}^2(s, t) - B_{0,0}^2(s, t), \quad g_2(s, t) = B_{0,2}^2(s, t) - B_{0,0}^2(s, t).$$

We choose the lexicographic ordering $s > t > u$; this order corresponds, through ϕ , to the relations $s > t > (u - s - t)$ for Bernstein polynomials – that is, after dehomogenization, to the relations $s > t > 1 - s - t$. Then, since $\frac{3}{2}sB_{1,1}^2(s, t) = B_{2,1}^3(s, t)$ we compute

$$\begin{aligned} f(s, t) - \frac{3}{2}sg_1(s, t) &= f(s, t) - \frac{3}{2} \left(\frac{2}{3}B_{2,1}^3(s, t) - \frac{1}{3}B_{1,0}^3(s, t) \right) \\ &= B_{1,2}^3(s, t) + \frac{1}{2}B_{1,0}^3(s, t) + B_{0,2}^3(s, t). \end{aligned}$$

Also, since $\frac{3}{2}tB_{1,1}^2(s, t) = B_{1,2}^3(s, t)$ we have

$$\begin{aligned} f(s, t) - \frac{3}{2}sg_1(s, t) - \frac{3}{2}tg_1(s, t) &= \\ B_{1,2}^3(s, t) + \frac{1}{2}B_{1,0}^3(s, t) + B_{0,2}^3(s, t) - \frac{3}{2} \left(\frac{2}{3}B_{1,2}^3(s, t) - \frac{1}{3}B_{0,1}^3(s, t) \right) &= \\ \frac{1}{2}B_{1,0}^3(s, t) + \frac{1}{2}B_{0,1}^3(s, t) + B_{0,2}^3(s, t). \end{aligned}$$

Finally, since $3(1-s-t)B_{0,2}^2(s,t) = B_{0,2}^3(s,t)$ we deduce that

$$\begin{aligned} f(s,t) - \frac{3}{2}sg_1(s,t) - \frac{3}{2}tg_1(s,t) - 3(1-s-t)g_2(s,t) = \\ \frac{1}{2}B_{1,0}^3(s,t) + \frac{1}{2}B_{0,1}^3(s,t) + B_{0,2}^3(s,t) - (B_{0,2}^3(s,t) - 3B_{0,0}^3(s,t)) = \\ \frac{1}{2}B_{1,0}^3(s,t) + \frac{1}{2}B_{0,1}^3(s,t) + 3B_{0,0}^3(s,t), \end{aligned}$$

that is, the Bernstein polynomial $f(s,t)$ is equal to

$$\begin{aligned} \left(\frac{3}{2}B_{1,0}^1(s,t) + \frac{3}{2}B_{0,1}^1(s,t) \right) g_1(s,t) + (3B_{0,0}^1(s,t)) g_2(s,t) + \\ \left(\frac{1}{2}B_{1,0}^3(s,t) + \frac{1}{2}B_{0,1}^3(s,t) + 3B_{0,0}^3(s,t) \right). \end{aligned}$$

Observe that as in the univariate setting we only needed multiplication of the multivariate Bernstein basis functions by powers of s , t and $1-s-t$.

Once we have a multivariate polynomial division algorithm, we can compute S -polynomials and apply Buchberger's algorithm to compute Gröbner bases. Notice that all the computations can be made purely in terms of the original Bernstein coefficients. The result of such a computation will not be a Gröbner basis for the ideal generated by the original Bernstein polynomials; nevertheless the result will have similar interesting properties. For instance, consider a collection of bivariate Bernstein polynomials f_1, \dots, f_k and denote by g_1, \dots, g_r the bivariate Bernstein polynomials generated by applying the Bernstein polynomial variant of Buchberger's algorithm with a given monomial order \succ . In general, g_1, \dots, g_r is not a Gröbner basis of the ideal generated by f_1, \dots, f_k for the monomial order \succ . But $\psi(g_1), \dots, \psi(g_r)$ is a Gröbner basis of the ideal generated by $\psi(f_1), \dots, \psi(f_k)$ for the monomial order \succ . Therefore g_1, \dots, g_r can be used in a manner similar to a Gröbner basis, since these polynomials are just the image under an isomorphism of a true Gröbner basis. For instance, the ideal membership problem can be solved by using the polynomials g_1, \dots, g_r ; indeed if we divide a bivariate Bernstein polynomial f by g_1, \dots, g_r , then the remainder r is zero if and only if f belongs to the ideal generated by the polynomials f_1, \dots, f_r because $\psi(f)$ belongs to the ideal generated by the polynomials $\psi(f_1), \dots, \psi(f_r)$ if and only if $\psi(r) = 0$. The situation is similar to the univariate setting, where the quotient and the remainder of the division algorithm are different in the Bernstein and monomial bases, but the result of the GCD algorithm for two univariate polynomials is essentially the same in both bases. Everything that can be computed in terms of Gröbner bases can be computed using the equivalent computation in the Bernstein basis through the isomorphism ϕ .

We end this section with a simple example showing that we can determine ideal membership working purely in Bernstein form. Consider the two bivariate Bernstein polynomials

$$f_1(s,t) = B_{2,0}^2(s,t), \quad f_2(s,t) = B_{1,0}^1(s,t) + B_{0,1}^1(s,t).$$

We want to determine if the bivariate polynomial

$$f(s,t) = \frac{1}{2}B_{1,0}^2(s,t) + B_{0,2}^2(s,t) + \frac{1}{2}B_{0,1}^2(s,t)$$

belongs to the ideal I generated by $f_1(s, t)$ and $f_2(s, t)$. To this end, we use the correspondence (5.1) and choose the lexicographic order $s > t > u$, as we did in the previous example. The Bernstein division of f by f_1 and f_2 returns

$$f(s, t) = 0 \times f_1(s, t) + B_{0,0}^1(s, t)f_2(s, t) + B_{0,2}^2(s, t).$$

Although the remainder here is not zero, we can not deduce that f does not belong to the ideal I since we did not compute a Gröbner basis for I . Applying the Bernstein polynomial variant of Buchberger’s algorithm, we find that the two bivariate Bernstein polynomials

$$g_1(s, t) = f_2(s, t), \quad g_2(s, t) = B_{0,2}^2(s, t),$$

form the analogue of a Gröbner basis for I . Now dividing f by g_1 and g_2 we find that

$$f(s, t) = B_{0,0}^1(s, t)g_1(s, t) + g_2(s, t),$$

which proves that f actually does belong to the ideal I .

6 Conclusion

We have presented three division algorithms for univariate Bernstein polynomials: an algorithm for dividing two univariate polynomials, an algorithm for calculating the GCD of an arbitrary collection of univariate polynomials and an algorithm for calculating a μ -basis for the syzygy module of an arbitrary collection of univariate polynomials. The division and GCD algorithms for two Bernstein polynomials are simple variants of Euclid’s classical division and GCD algorithms for two polynomials in monomial form; the GCD algorithm for three or more Bernstein polynomials is a consequence of the μ -basis algorithm for Bernstein polynomials. We also extended division algorithms for Bernstein polynomials to the multivariate setting, where we developed an analogue of Buchberger’s algorithm to compute bases comparable to Gröbner bases for ideals generated by multivariate Bernstein polynomials. The main idea behind each of these algorithms is to apply the ring isomorphism ψ to convert the problem from the Bernstein basis to an equivalent problem in the monomial basis without performing a change of basis procedure. This approach allows us to perform all the calculations using only the original Bernstein coefficients. The only operations required in these division algorithms for Bernstein polynomials are multiplication by powers of t , removing common powers of $1 - t$, and degree elevation.

Acknowledgment

This work was partially supported by the French ANR grant “GECKO” and by NSF grant OISE-0421771.

References

- [1] Laurent Busé, Mohamed Elkadi, and André Galligo. A computational study of ruled surfaces. To appear in *J. Symbolic Comput.*, 2007.

- [2] J. M. Carnicer and J. M. Peña. Shape preserving representations and optimality of the Bernstein basis. *Adv. Comput. Math.*, 1(2):173–196, 1993.
- [3] Falai Chen. Reparametrization of a rational ruled surface using the μ -basis. *Comput. Aided Geom. Design*, 20(1):11–17, 2003.
- [4] Falai Chen and Tom Sederberg. A new implicit representation of a planar rational curve with high order singularity. *Comput. Aided Geom. Design*, 19(2):151–167, 2002.
- [5] Falai Chen and Wenping Wang. The μ -basis of a planar rational curve - properties and computation. *Graphical Models*, 64(6):368–381, 2002.
- [6] Falai Chen and Wenping Wang. Revisiting the μ -basis of a rational ruled surface. *J. Symbolic Comput.*, 36(5):699–716, 2003.
- [7] Falai Chen, Jianmin Zheng, and Thomas W. Sederberg. The mu-basis of a rational ruled surface. *Comput. Aided Geom. Design*, 18(1):61–72, 2001.
- [8] David Cox, John Little, and Donal O’Shea. *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1997.
- [9] David Cox, John Little, and Donal O’Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.
- [10] David A. Cox, Thomas W. Sederberg, and Falai Chen. The moving line ideal basis of planar rational curves. *Comput. Aided Geom. Design*, 15(8):803–827, 1998.
- [11] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Comput. Aided Geom. Design*, 4(3):191–216, 1987.
- [12] Ron Goldman. *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*. Morgan Kaufmann Publishers, Academic Press, San Diego, 2002.
- [13] David Hilbert. Ueber die Theorie der algebraischen Formen. *Math. Ann.*, 36(4):473–534, 1890.
- [14] Zhiping Lin. On syzygy modules for polynomial matrices. *Linear Algebra Appl.*, 298(1-3):73–86, 1999.
- [15] Thomas W. Sederberg and Falai Chen. Implicitization using moving curves and surfaces. volume SIGGRAPH’95 of *Computer Graphics, Annual Conference Series*, pages 301–308, 1995.
- [16] Thomas W. Sederberg, Ron Goldman, and Hang Du. Implicitizing rational curves by the method of moving algebraic curves. *J. Symbolic Comput.*, 23(2-3):153–175, 1997.
- [17] Thomas W. Sederberg, Takafumi Saito, Dong Xu Qi, and Krzysztof S. Klimaszewski. Curve implicitization using moving lines. *Comput. Aided Geom. Design*, 11(6):687–706, 1994.

- [18] Ning Song, Falai Chen, and Ron Goldman. Axial moving lines and singularities of rational planar curves. To appear in *Comput. Aided Geom. Design*, 2007.
- [19] Ning Song and Ron Goldman. μ -bases for polynomial systems in one variable. Submitted to *Comput. Aided Geom. Design*, 2007.
- [20] Jianmin Zheng and Thomas W. Sederberg. A direct approach to computing the μ -basis of planar rational curves. *J. Symbolic Comput.*, 31(5):619–629, 2001.