Bernstein–Bezoutian matrices and curve implicitization

Ana Marco, José-Javier Martínez*

Departamento de Matemáticas, Universidad de Alcalá, Campus Universitario, 28871-Alcalá de Henares (Madrid), Spain

Received 4 April 2006; received in revised form 16 January 2007; accepted 4 February 2007

Communicated by V. Pan

Abstract

A new application of Bernstein–Bezoutian matrices, a type of resultant matrices constructed when the polynomials are given in the Bernstein basis, is presented. In particular, the approach to curve implicitization through Sylvester and Bézout resultant matrices and bivariate interpolation in the usual power basis is extended to the case in which the polynomials appearing in the rational parametric equations of the curve are expressed in the Bernstein basis, avoiding the basis conversion from the Bernstein to the power basis. The coefficients of the implicit equation are computed in the bivariate tensor-product Bernstein basis, and their computation involves the bidiagonal factorization of the inverses of certain totally positive matrices.

Keywords: Bernstein basis; Bezoutian matrices; Curve; Implicitization; Interpolation; Total positivity

1. Introduction

When studying rational plane algebraic curves, there are two standard ways of representation: the implicit equations and the parametric equations. The interest in one or other representation depends on the operations that one wants to do with the curve. For example, the intersection of two curves is more easily computed when we have the implicit equation of one curve and the parametric equations of the other, and hence it is very important to be able to change from one representation to another.

We will concentrate on the implicitization problem, that is to say, on finding an implicit representation starting from a given rational parametrization of the curve. A good introduction to the implicitization problem that contains some of the methods for solving it and several references is [21]. In [16] we have presented an approach to the implicitization problem based on resultants and on interpolation using the usual power basis for the corresponding space of bivariate polynomials.

However, the recent work [2], in which the Bernstein–Bézout matrix is introduced and used for designing a fast algorithm for computing the greatest common divisor (GCD) of two polynomials expressed in Bernstein basis, has shown the importance of evaluating resultants from Bernstein-basis resultant matrices directly, avoiding a basis transformation between the Bernstein basis and the power basis. In this sense, in [2] it is indicated that for...
numerical computations involving polynomials in Bernstein form it is essential to consider algorithms which express all intermediate results using this form only. This fact is a consequence of results concerning the potentially severe loss of accuracy in the basis conversion between Bernstein and power bases [9].

Although in [2] only univariate polynomials are studied, it must be observed that the construction of the resultant matrices can be extended to the case of multivariate polynomials. In this case some care must be taken when using the Bernstein–Bézout matrix for computing the resultant, since there are situations in which the determinant of the Bernstein–Bézout matrix is not the resultant but a polynomial multiple of it.

So, starting from a plane algebraic curve given by its parametric equations in Bernstein form (the usual situation in the case of Bézier curves, a kind of curve which is very popular in computer aided geometric design (CAGD) due to the properties they satisfy [14]), our aim is to use the Bernstein–Bézout matrix and bivariate interpolation for obtaining its implicit equation in the bivariate tensor-product Bernstein basis. So, the basis conversion is also avoided in the bivariate polynomials.

The interest in obtaining the implicit equation of the curve in the bivariate Bernstein basis comes from the fact that this basis has important advantages in the context of plotting implicit algebraic curves, as it is shown for example by various experiments on interval arithmetic methods using the Bernstein basis presented in [18].

Although we present all the details with an example in exact rational arithmetic, it must be taken into account that the process can also be carried out in (high) finite precision arithmetic. In that situation some important results of numerical linear algebra we use will have a major importance. More precisely, the total positivity and the structure of certain matrices will be important issues, as it happens in several instances of CAGD (see, for example, the recent work [23] and references therein).

The rest of the paper is organized as follows. In Section 2 several basic results will be presented. In Section 3 we introduce the interpolation algorithm for computing the implicit equation as a factor of the determinant of the resultant matrix, while in Section 4 we consider some results related to total positivity which will be relevant for the solving of the linear system associated with the interpolation problem. Finally, in Section 5 we briefly examine the computational complexity of the whole algorithm.

2. Preliminaries

Let \( P(t) = (x(t), y(t)) \) be a proper parametrization of a rational plane algebraic curve \( C \), where \( x(t) = \frac{u_1(t)}{v_1(t)} \) and \( y(t) = \frac{u_2(t)}{v_2(t)} \) and \( \gcd(u_1, v_1) = \gcd(u_2, v_2) = 1 \). A parametrization \( P(t) = (x(t), y(t)) \) of a curve \( C \) is said to be proper if every point on \( C \) except a finite number of exceptional points is generated by exactly one value of the parameter \( t \). It is well known that every rational curve has a proper parametrization, so we can assume that the parametrization is proper. Several recent results on the properness of curve parametrizations can be seen in [22].

In connection with the implicitization problem, the following theorem [22] holds:

**Theorem 1.** Let \( P = (x(t) = \frac{u_1(t)}{v_1(t)}, y(t) = \frac{u_2(t)}{v_2(t)}) \) be a proper rational parametrization of an irreducible curve \( C \), with \( \gcd(u_1, v_1) = \gcd(u_2, v_2) = 1 \). Then the polynomial defining \( C \) is \( \text{Res}_t(u_1(t) - xv_1(t), u_2(t) - yv_2(t)) \) (the resultant with respect to \( t \) of the polynomials \( u_1(t) - xv_1(t) \) and \( u_2(t) - yv_2(t) \)).

Our aim is to compute the implicit equation \( F(x, y) = 0 \) of the curve \( C \) by means of polynomial interpolation, which taking into account Theorem 1 is equivalent to computing \( \text{Res}_t(u_1(t) - xv_1(t), u_2(t) - yv_2(t)) \).

First of all, we remark that the concept of interpolation space will be essential. The following result, also in [22], shows which is in our case the most suitable interpolation space:

**Theorem 2.** Let \( P = (x(t) = \frac{u_1(t)}{v_1(t)}, y(t) = \frac{u_2(t)}{v_2(t)}) \) be a proper rational parametrization of the irreducible curve \( C \) defined by \( F(x, y) \), and let \( \gcd(u_1, v_1) = \gcd(u_2, v_2) = 1 \). Then \( \deg_y(F) = \max\{\deg_t(u_1) , \deg_t(v_1)\} \) and \( \deg_x(F) = \max\{\deg_t(u_2) , \deg_t(v_2)\} \).

**Theorem 2** tells us that the polynomial \( F(x, y) \) defining the implicit equation of the curve \( C \) belongs to the polynomial space \( \Pi_{n,m}(x, y) \), where \( n = \max\{\deg_t(u_2) , \deg_t(v_2)\} \) and \( m = \max\{\deg_t(u_1) , \deg_t(v_1)\} \). The dimension of \( \Pi_{n,m}(x, y) \) is \( (n + 1)(m + 1) \), and a basis is given by \( \{x^iy^j | i = 0, \ldots , n; j = 0, \ldots , m\} \). Moreover, \( \deg_x(F(x, y)) = n \) and \( \deg_y(F(x, y)) = m \), and therefore there is no interpolation space \( \Pi_{r,s}(x, y) \) with \( r < n \) or \( s < m \) such that \( F(x, y) \) belongs to \( \Pi_{r,s}(x, y) \).
Let us note that these theorems refer to the degree of polynomials in the power basis, so since now we will be using the Bernstein basis some care will be needed. For the sake of clarity we will illustrate all our results with a small example. Let

\[
\{ \beta_0^{(4)}(t), \beta_1^{(4)}(t), \beta_2^{(4)}(t), \beta_3^{(4)}(t), \beta_4^{(4)}(t) \}
\]

be the (univariate) Bernstein basis of the space of polynomials of degree less than or equal to 4, where the Bernstein polynomials are defined as follows,

\[
\beta_i^{(n)}(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad i = 0, \ldots, n,
\]

and let us consider the algebraic curve given by the parametric equations

\[
x(t) = \frac{4\beta_0^{(4)}(t) + 4\beta_1^{(4)}(t) + 3\beta_2^{(4)}(t) + 3\beta_3^{(4)}(t) + 7\beta_4^{(4)}(t)}{\beta_0^{(4)}(t) + \beta_1^{(4)}(t) + \beta_2^{(4)}(t) + \beta_3^{(4)}(t) + 3\beta_4^{(4)}(t)}
\]

\[
y(t) = 2\beta_0^{(4)}(t) + 3\beta_1^{(4)}(t) + 3\beta_2^{(4)}(t) + 3\beta_3^{(4)}(t) + 4\beta_4^{(4)}(t).
\]

If we call \( p(t) = u_1(t) - xv_1(t) \) and \( q(t) = u_2(t) - yv_2(t) \), their coefficients in the Bernstein basis are given by

\[
p_0 = 4 - x, \quad p_1 = 4 - x, \quad p_2 = 3 - x, \quad p_3 = 3 - x, \quad p_4 = 7 - 3x,
\]

and

\[
q_0 = 2 - y, \quad q_1 = 3 - y, \quad q_2 = 3 - y, \quad q_3 = 3 - y, \quad q_4 = 4 - y.
\]

However, let us observe that if we write \( p \) and \( q \) in the power basis we have

\[
p(t) = 4 - x - 6t^2 + 8t^3 + (-2x + 1)t^4
\]

(a polynomial of degree 4 in \( t \)), while

\[
q(t) = 2 - y + 4t - 6t^2 + 4t^3,
\]

a polynomial of degree 3 in \( t \).

Therefore, taking into account Theorem 2, the polynomial defining the implicit equation will be a polynomial belonging to the space \( \Pi_{n,m}(x, y) \) with \( n = 3 \) and \( m = 4 \). We will use for that space the tensor-product bivariate Bernstein basis given by

\[
\{ B_{ij}^{(n,m)}, i = 0, \ldots, n; j = 0, \ldots, m \} = \{ \beta_i^{(n)}(x)\beta_j^{(m)}(y), i = 0, \ldots, n; j = 0, \ldots, m \}.
\]

Finally, we will recall, following [2], the algorithm for constructing the Bernstein–Bézout matrix of the polynomials \( p(t) \) and \( q(t) \) expressed in the Bernstein basis \( \{ \beta_i^{(n)}(t), i = 0, \ldots, n \} \). Although in [2] the coefficients of the polynomials are always numbers, in our application we will consider the symbolic (i.e. with the entries being polynomials in \( x, y \)) Bernstein–Bézout matrix of \( p(t) \) and \( q(t) \) which we denote by \( BS \). For the reader’s convenience, we present the algorithm written in Maple language:

```maple
for i from 1 to n do
    BS[i,1]:=(n/i)*(p[i]*q[0]-p[0]*q[i]);
od;
for j from 1 to n-1 do
    BS[n,j+1]:=(n/(n-j))*(p[n]*q[j]-p[j]*q[n]);
od;
for j from 1 to n-1 do
    for i from 1 to n-1 do
        BS[i,j+1]:=(n^2/(i*(n-j)))*(p[i]*q[j]-p[j]*q[i]);
    od;
od;
```


\[(j*(n-i))/(i*(n-j))*BS[i+1,j];\]

od;

od;

Let us observe that, if \(m = n\), the resultant is the determinant of the Bernstein–Bézout matrix, while – as a consequence of the corresponding result for the Bézout resultant [21] – if \(m > n\), that determinant is equal to the resultant multiplied by the factor \((\tilde{p}_m)^{m-n}\), where \(\tilde{p}_m\) is the leading coefficient of \(p(t)\) in the power basis. So, in our example, the determinant of \(BS\) will be the implicit equation we are looking for multiplied by the factor \(-2x + 1\), since the degree of \(p\) is 4 and the degree of \(q\) is 3 and the coefficient of \(t^4\) in \(p\) is \(-2x + 1\).

Therefore, the fact of knowing the leading coefficients of \(p\) and \(q\) in the power basis has two important advantages: it allows one to work with an interpolation space of smallest dimension, and it avoids the need of polynomial factorization, which would be a difficult task if the coefficients of the polynomial multiple of the implicit equation are not computed exactly.

In the following section we will show how to compute the coefficients in the bivariate tensor-product Bernstein basis of the implicit equation (which will be a scalar multiple of the resultant computed by using the approach of [16], where the equation is obtained in the usual power basis).

### 3. The interpolation process

Since the expansion of the symbolic determinant is very time and space consuming, our aim is to compute the polynomial defining the implicit equation by means of Lagrange bivariate interpolation, but using the bivariate tensor-product Bernstein basis instead of the power basis. A good introduction to the theory of interpolation can be seen in [6].

If we consider the interpolation nodes \((x_i, y_j)\) \((i = 0, \ldots, n; j = 0, \ldots, m)\) and the interpolation space \(\Pi_{n,m}(x, y)\), the interpolation problem is stated as follows:

Given \((n+1)(m+1)\) values

\[f_{ij} \in \mathbb{K}, \quad i = 0, \ldots, n; \quad j = 0, \ldots, m\]

(the interpolation data), find a polynomial

\[F(x, y) = \sum_{(i,j) \in I} c_{ij} \beta_i^{(n)}(x) \beta_j^{(m)}(y) \in \Pi_{n,m}(x, y)\]

(where \(I\) is the index set \(I = \{(i, j) | i = 0, \ldots, n; \quad j = 0, \ldots, m\}\)) such that

\[F(x_i, y_j) = f_{ij} \quad \forall \quad (i, j) \in I.\]

If we consider for the interpolation space \(\Pi_{n,m}(x, y)\) the basis

\[\{B_{ij}^{(n,m)}, \quad i = 0, \ldots, n; \quad j = 0, \ldots, m\} = \{\beta_i^{(n)}(x) \beta_j^{(m)}(y), \quad i = 0, \ldots, n; \quad j = 0, \ldots, m\}\]

\[= \{B_{00}^{(n,m)}, B_{01}^{(n,m)}, \ldots, B_{0m}^{(n,m)}, B_{10}^{(n,m)}, B_{11}^{(n,m)}, \ldots, B_{1m}^{(n,m)}, \ldots, B_{nm}^{(n,m)}\}\]

with that precise ordering, and the interpolation nodes with the corresponding ordering

\[\{(x_i, y_j) | i = 0, \ldots, n; \quad j = 0, \ldots, m\}\]

\[= \{(x_0, y_0), (x_0, y_1), \ldots, (x_0, y_m), (x_1, y_0), (x_1, y_1), \ldots, (x_1, y_m), \ldots, (x_n, y_0), \ldots, (x_n, y_m)\},\]

then the \((n+1)(m+1)\) interpolation conditions \(F(x_i, y_j) = f_{ij}\) can be written as a linear system

\[Ac = f,\]

where the coefficient matrix \(A\) is given by a Kronecker product

\[B_x \otimes B_y.\]
with
\[ B_x = ((\beta_j^{(n)}(x_i)), \quad i = 0, \ldots, n; \quad j = 0, \ldots, n, \]
\[ B_y = ((\beta_j^{(m)}(y_i)), \quad i = 0, \ldots, m; \quad j = 0, \ldots, m. \]
\[ c = (c_{00}, \ldots, c_{0m}, c_{10}, \ldots, c_{1m}, \ldots, c_{n0}, \ldots, c_{nm})^T. \]

and
\[ f = (f_{00}, \ldots, f_{0m}, f_{10}, \ldots, f_{1m}, \ldots, f_{n0}, \ldots, f_{nm})^T. \]

The Kronecker product \( D \otimes E \) is defined by blocks as \((d_{kl}E)\), with \( D = (d_{kl}) \).

Let us observe that, although the interpolation data can be generated by constructing the symbolic Bernstein–Bézout matrix \( BS \), evaluating it at each interpolation node, and then computing the corresponding numerical determinant, it is more convenient to obtain them in a way that avoids the construction of the symbolic matrix \( BS \).

This is the reason why we compute each interpolation datum by means of the evaluation of \( p(t) \) and \( q(i) \) at the node \((x_i, y_j)\) followed by the computation of the determinant of the corresponding numerical Bernstein–Bézout matrix \( B \) making use of the Bini–Gemignani algorithm which constructs (in \( O(n^2) \) arithmetic operations) the Bernstein–Bézout matrix for the evaluated polynomials. In this way we only work with numbers and not with variables, and the computational cost of the process is smaller. In addition, if the determinant of the symbolic Bernstein–Bézout matrix is not the resultant but a polynomial multiple of it, we must divide the value of the determinant by the polynomial factor evaluated at the node \((x_i, y_j)\) (in our example we must divide by \(-2x_i + 1\)).

As for the interpolation nodes, they must be selected in a way that the matrices \( B_x \) and \( B_y \) are nonsingular because in this case the matrix \( B_x \otimes B_y \) is also nonsingular, and the interpolation problem has a unique solution. In addition, to carry out the computation of the interpolation data as described in the above paragraph we must not use the value of \( x_i \) for which the leading coefficient of \( p(t) \) in the power basis evaluates to 0, and the value \( y_j \) for which the leading coefficient of \( q(t) \) in the power basis evaluates to 0.

An algorithm for solving linear systems with a Kronecker product coefficient matrix is derived in a self-contained way (in a more general setting) in [19]. In our case, it reduces the solution of the linear system of order \((n+1)(m+1)\) with coefficient matrix \( B_x \otimes B_y \) to the solution of \( n+1 \) linear systems with the same matrix \( B_y \) and \( m+1 \) linear systems with the same matrix \( B_x \). For the case of the power basis considered in [16], taking into account that every linear system to be solved was a Vandermonde linear system, it was convenient to use the Björck–Pereyra algorithm [4,12] to solve those linear systems. For the Bernstein basis being used here, an appropriate algorithm which takes advantage of the special properties of the coefficient matrices \( B_x \) and \( B_y \) will be considered in Section 4.

In the example we are considering we will choose as interpolation nodes \((x_i, y_j) = (\frac{i+1}{n+2}, \frac{j+1}{m+2}) \) \((i = 0, \ldots, n; \quad j = 0, \ldots, m)\). With this selection of the nodes the matrices

\[ B_x = \begin{pmatrix}
\frac{64}{125} & \frac{48}{125} & \frac{12}{125} & \frac{1}{125} \\
\frac{27}{125} & \frac{54}{125} & \frac{36}{125} & \frac{8}{125} \\
\frac{12}{125} & \frac{12}{125} & \frac{48}{125} & \frac{64}{125} 
\end{pmatrix}, \]

and

\[ B_y = \begin{pmatrix}
\frac{625}{1296} & \frac{125}{1296} & \frac{35}{1296} & \frac{5}{1296} & \frac{1}{1296} \\
\frac{16}{81} & \frac{32}{81} & \frac{8}{81} & \frac{1}{81} & \frac{1}{81} \\
\frac{1}{16} & \frac{1}{4} & \frac{1}{4} & \frac{1}{16} & \frac{1}{81} \\
\frac{1}{81} & \frac{8}{81} & \frac{8}{81} & \frac{16}{81} \\
\frac{1}{125} & \frac{5}{125} & \frac{25}{125} & \frac{625}{1296}
\end{pmatrix}, \]

are nonsingular and the leading coefficients of \( p(t) \) and \( q(t) \) do not evaluate to zero at any node. Let us notice that in this case \( B_x \) and \( B_y \) are also strictly totally positive matrices. The reasons why we are interested in \( B_x \) and \( B_y \) being strictly totally positive will be explained in Section 4.
The vector \( f \) containing the interpolation data is in this case:

\[
\begin{pmatrix}
2966444009 & 168039769 & 89075963 & 132479888 & 1836508249 & 6917076001 \\
4374000 & 273375 & 162000 & 273375 & 4374000 & 13122000 \\
392870521 & 7730521 & 311002432 & 4317494161 & 5255095199 & 299356319 \\
820125 & 18000 & 820125 & 13122000 & 13122000 & 820125 \\
53144671 & 238032368 & 3309470639 & 1295093591 & 74016751 & 39526277 \\
162000 & 820125 & 13122000 & 4374000 & 273375 & 162000 \\
\end{pmatrix}
\]

After solving the \( n + 1 \) linear systems with coefficient matrix \( B_y \) and the \( m + 1 \) linear systems with the matrix \( B_x \) by means of the algorithm considered in the following section, we obtain the vector \( c \) with the coefficients of the desired implicit equation in the tensor-product bivariate Bernstein basis (using the lexicographical ordering we are considering):

\[
\begin{pmatrix}
25264 & 66256 & 167852 & 45652 & 36137 & 15728 & 125312 & 320120 & 29164 & 69421 \\
27 & 81 & 243 & 81 & 81 & 27 & 243 & 729 & 81 & 243 \\
29440 & 79024 & 203228 & 18580 & 14761 & 2048 & 16640 & 14336 & 3940 & 9391 \\
81 & 243 & 729 & 81 & 81 & 9 & 81 & 27 & 81 \\
\end{pmatrix}
\]

4. Total positivity of \( B_x \) and \( B_y \)

For the sake of clarity, in our example all the computations have been done in exact arithmetic, but usually in practice finite precision arithmetic will be used. So, our aim in this section is to consider the numerical solving of the linear system \((B_x \otimes B_y)c = f\) corresponding to the interpolation problem. As will be shown, the total positivity of the matrices \( B_x \) and \( B_y \) will be an essential property.

Making use of the results of [10,11], we know that performing the complete Neville elimination on a strictly totally positive matrix \( N \) a bidiagonal factorization of its inverse \( N^{-1} \) can be obtained, that is to say, we have

\[
N^{-1} = G_1 G_2 \ldots G_{n-1} D^{-1} F_{n-1} F_{n-2} \ldots F_1,
\]

where \( D^{-1} \) is a diagonal matrix and \( F_i \) and \( G_i \) are bidiagonal matrices.

So, after having obtained that factorization (with a computational cost of \( O(n^3) \) arithmetic operations), all the linear systems \( N z = b \) with the same coefficient matrix \( N \) can be solved (with a cost of \( O(n^2) \) arithmetic operations) by performing the product

\[
G_1 G_2 \ldots G_{n-1} D^{-1} F_{n-1} F_{n-2} \ldots F_1 b.
\]

An early application of these ideas to solve structured linear systems can be seen in [20], and a recent extension has been presented in [7].

On the other hand, from [5] we know that the Bernstein basis of the space of polynomials of degree less than or equal to \( n \) is a strictly totally positive basis on the open interval \((0, 1)\), which implies that all the collocation matrices

\[
M = \begin{pmatrix} \beta_j^{(n)}(t_i) \end{pmatrix}, \quad i, j = 0, \ldots, n
\]

with \( t_0 < t_1 < \cdots < t_n \) in \((0, 1)\) are strictly totally positive, i.e. all their minors are strictly positive.

Therefore, choosing in our situation the interpolation nodes \((x_i, y_j)\) \((i = 0, \ldots, n; j = 0, \ldots, m)\) satisfying \( x_0 < x_1 < \cdots < x_n \) in \((0, 1)\) and \( y_0 < y_1 < \cdots < y_m \) in \((0, 1)\), the matrices \( B_x \) and \( B_y \) are strictly totally positive and the bidiagonal factorization of the matrices \( B_x^{-1} \) and \( B_y^{-1} \) can be obtained by means of Neville elimination. Let us recall that, as we explained in Section 3, if there exists a value of \( x \) at which the leading coefficient of \( p(t) \) in the power basis evaluates to 0, or a value of \( y \) at which the leading coefficient of \( q(t) \) in the power basis evaluates to 0, it must be avoided.

Taking into account that solving the linear system \((B_x \otimes B_y)c = f\) is equivalent to solving \( n + 1 \) linear systems with the matrix \( B_y \) and then \( m + 1 \) linear systems with the matrix \( B_x \) [19], we can proceed by computing the bidiagonal
factorization of the inverse of $B_y$ by means of Neville elimination and then by solving each one of the linear systems $B_yz = b$ in $O(n^2)$ arithmetic operations by computing the product

$$G_1G_2\ldots G_{n-1}D^{-1}F_{n-1}F_{n-2}\ldots F_1b$$

(see [7]). After that, we can proceed analogously for solving the $m+1$ linear systems with the same coefficient matrix $B_x$.

A detailed error analysis of Neville elimination, which shows the advantages of this type of elimination for the class of totally positive matrices, has been carried out in [1], and related work for the case of Vandermonde linear systems can be seen in Chapter 22 of [13].

In addition to being totally positive matrices, $B_x$ and $B_y$ are also structured matrices which are called in [17] Bernstein–Vandermonde matrices. The exploitation of their structure allows us to reduce the cost of computing the bidiagonal factorization of $B_x^{-1}$ and $B_y^{-1}$ from $O(n^3)$ to $O(n^2)$ if the algorithm included in [17] is used. This algorithm uses explicit expressions for computing the determinants involved in the Neville elimination of the Bernstein–Vandermonde matrices, avoiding in this way the subtractive cancellation which appears when working in floating point arithmetic. Therefore the algorithm is both faster and more accurate than the general Neville elimination for this class of matrix. Some numerical experiments that confirm the convenience of using this algorithm for computing the bidiagonal factorization of Bernstein–Vandermonde matrices are reported in [17].

It must be observed that Bernstein–Vandermonde matrices are ill-conditioned matrices, although their ill-conditioning is less severe than that of Vandermonde matrices [17]. So, standard algorithms such as Gaussian elimination are not appropriate for solving linear systems with that type of coefficient matrix.

As for the example included in this paper, the choice of the nodes presented in Section 3 has allowed us to obtain the coefficients of the polynomial defining the implicit equation of the curve by using the approach described in this section.

5. Computational complexity

In this section we will briefly examine the computational complexity of our algorithm in terms of arithmetic operations. In view of the algorithm, we must solve $n+1$ systems of order $m+1$ with the same matrix $B_y$ and $m+1$ systems of order $n+1$ with the same matrix $B_x$.

The bidiagonal factorization of the inverse of a Bernstein–Vandermonde matrix of order $n$ takes $O(n^2)$ operations if the algorithm of [17] is used. That factorization is used for solving all the systems with the same matrix, so each of the systems can be solved with $O(n^2)$ operations.

For the sake of clarity in the comparison, we will consider here the case $m = n$. Then, the interpolation part of the algorithm has computational complexity $O(n^3)$. Let us observe that, in this situation, if we solve the linear system $Ac = f$ of order $(n+1)^2$ by means of Gaussian elimination, without taking into account the special structure of the matrix, we have computational complexity $O(n^6)$. Moreover, using the approach we are describing, there is no need to construct the matrix $A$ (and even the construction of $B_x$ and $B_y$ is not needed if the algorithm of [17] is used), which implies an additional saving in computational cost.

Let us remark that, since the construction of each numerical Bernstein–Bézout matrix requires $O(n^2)$ arithmetic operations and the complexity of the computation of its determinant is $O(n^3)$, the generation of the $(n+1)^2$ interpolation data has a computational complexity of $O(n^6)$. Therefore with our approach, which exploits the Kronecker product structure, the whole process has complexity $O(n^6)$, while using Gaussian elimination it would be $O(n^9)$.

It is worth noting that the main cost of the process corresponds to the generation of the interpolation data, and not to the computation of the coefficients of the interpolating polynomial. So, the main effort to reduce the computational cost must be focused on that stage, i.e. on the computation of the determinants of numerical Bernstein–Bézout matrices. In this sense, an interesting approach would be to take advantage of the displacement structure of the Bernstein–Bézout matrices [2,15] (see also sections 2.11 and 2.12 of [3]) to try to develop an algorithm with complexity $O(n^2)$ for computing each determinant. In this way the complexity of the whole process would be $O(n^4)$. Nevertheless this is not a trivial task (see the difficulties indicated at the end of Section 3 of [2]).

By using different techniques, some improved algorithms for computing determinants of certain structured matrices in a symbolic setting have recently been obtained in [8].
Remark. Finally, let us observe that all the linear systems with matrix $B_y$ can be solved simultaneously, and the same can be said of the systems with matrix $B_x$. Therefore the algorithm exhibits a high degree of intrinsic parallelism. This parallelism is also present in the computation of the interpolation data since we can compute simultaneously the determinants involved in this process.

Acknowledgements

This research has been partially supported by Spanish Research Grant MTM2006-03388 from the Spanish Ministerio de Educación y Ciencia.

References