Marco García, A., Martínez Fernández de las Heras, J.J. & Viaña Fernández, R. 2019, "Least squares problems involving generalized Kronecker products and application to bivariate polynomial regression", Numerical Algorithms, vol. 82, pp. 21-39.

Available at http://dx.doi.org/10.1007/s11075-018-0592-1

Noname manuscript No.
(will be inserted by the editor)

Least squares problems involving generalized Kronecker products and application to bivariate polynomial regression

Ana Marco · José-Javier Martínez · Raquel Viaña

Received: date / Accepted: date

Abstract A method for solving least squares problems $(A \otimes B_i)x = b$ whose coefficient matrices have generalized Kronecker product structure is presented. It is based on the exploitation of the block structure of the Moore-Penrose inverse and the reflexive minimum norm g-inverse of the coefficient matrix, and on the QR method for solving least squares problems. Firstly, the general case where A is a rectangular matrix is considered, and then the special case where A is square is analyzed. This special case is applied to the problem of bivariate polynomial regression, in which the involved matrices are structured matrices (Vandermonde or Bernstein-Vandermonde matrices). In this context, the advantage of using the Bernstein basis instead of the monomial basis is shown. Numerical experiments illustrating the good behaviour of the proposed algorithm are included.

Keywords Least squares · Generalized Kronecker product · Generalized inverse · Bernstein-Vandermonde matrices · Total positivity

Mathematics Subject Classification (2000) MSC 65F20 · MSC 65F05 · 62J05 · 15A09 · 15B48

A. Marco · J.-J. Martínez · R. Viaña
Departamento de Física y Matemáticas, Universidad de Alcalá, Campus Universitario, 28871-Alcalá de Henares (Madrid), Spain

A. Marco
E-mail: ana.marco@uah.es

J.-J. Martínez
E-mail: jjavier.martinez@uah.es

R. Viaña
E-mail: raquel.viana@uah.es

1 Introduction

As it can be read in Section 2.5 of [4] the Kronecker product structure arises in several application areas, including signal and image processing, photogrammetry, and multidimensional approximation. In particular, it appears in a natural way in least squares surface fitting of multivariate data on a rectangular grid. When more flexibility is allowed on the point distribution, as seen for the interpolation case in [24], a generalized Kronecker product structure arises.

Least squares problems involving the Kronecker product structure are analyzed in [3,4,10,11,15,26].

Another important field in which Kronecker product arises is Tikhonov regularization for bidimensional problems, as seen in Section 6 of [16]. As in our work, one important issue in that paper is to take advantage of the block structure of the Kronecker product to avoid working with the large matrix $A \otimes B$.

In this work we present a new method for solving bivariate least squares problems whose coefficient matrices have generalized Kronecker product structure. Our approach is inspired by our work on solving bivariate Lagrange polynomial interpolation problems by using the Bernstein basis [24]. In that paper, the solution of the interpolation problem is obtained by solving a linear system whose coefficient matrix is the generalized Kronecker product of Bernstein-Vandermonde matrices. In [27] a generalized Kronecker product of matrices has already appeared (in fact, the authors start from a left Kronecker product) and some of its applications in the area of signal processing have been analyzed (see also [28]). A generalized Kronecker product has also been considered in [13] (in fact, in that paper submatrices of matrices with generalized Kronecker product structure are used), and an algorithm for solving linear systems with that structure has been presented in [25].

All the least squares problems involving generalized Kronecker products we consider will have unique solution, and we will compute it with the help of the Moore-Penrose inverse or a reflexive minimum norm g-inverse of the coefficient matrix. As we will show, these Moore-Penrose inverses and minimum norm g-inverses are block matrices and the exploitation of their structure will be crucial for obtaining a fast algorithm for solving these least squares problems.

The case in which the coefficient matrix of the least squares problem has (standard) Kronecker product structure is analyzed as a particular case of the least squares problems we are considering. Algorithms for solving least squares problems involving Kronecker products can be found in [10,11].

As we will show, a situation in which the solution of a least squares problem with generalized Kronecker product structure appears naturally is when solving a bivariate polynomial regression problem when the bivariate tensor product monomial basis or the bivariate tensor product Bernstein basis are considered. If the nodes and the basis polynomials are adequately ordered, the coefficient matrix of the associated least squares problem will be in the first case the generalized Kronecker product of Vandermonde matrices and in

the second one, the generalized Kronecker product of Bernstein-Vandermonde matrices (see [19,21] for a description of Bernstein-Vandermonde matrices). Since Bernstein-Vandermonde matrices are better conditioned than Vandermonde matrices [7,20], considering the tensor product Bernstein basis is more convenient for solving the regression problem accurately. The implementation in MATLAB of a fast and accurate algorithm for solving the bivariate regression problem when the tensor product Bernstein basis is considered is included. In this algorithm not only the generalized Kronecker product structure of the coefficient matrix corresponding to the least squares problem is exploited, but also the structure of the Bernstein-Vandermonde matrices and their total positivity, which is a consequence of the selected ordering of the nodes and the basis [19].

The rest of the paper is organized as follows. The general least squares problem we are interested in solving is analyzed in Section 2. Section 3 is devoted to a particular case which, as we will show in Section 4, appears naturally when bivariate polynomial regression problems are considered. An algorithm for solving the bivariate polynomial regression problem when the bivariate tensor product Bernstein basis is used is given in Section 4. In Section 5 numerical experiments illustrating the good properties of our approach are included. Finally, Section 6 presents the conclusions.

2 Least squares problems with generalized Kronecker product structure

From now on, let $A \in \mathbb{R}^{m\times p}$ be a matrix with $m \geq p$ and $rank(A) = p$, and let $B_i \in \mathbb{R}^{n\times q}$ $(i = 1, \ldots, m)$ be matrices with $n \geq q$ and $rank(B_i) = q$. We consider the linear system

$$(A \otimes B_i)x = b,$$

where $A \otimes B_i$ is the generalized Kronecker product of the matrices A and B_i.

Definition 1 Let M be a matrix of size $m \times p$ and let N_i $(i = 1, \ldots, m)$ be matrices of size $n \times q$. The *generalized Kronecker product* of M and N_i is the matrix of size $mn \times pq$, denoted by $M \otimes N_i$, defined by the mp blocks of size $n \times q$ as

$$(m_{kl}N_k),$$

with $M = (m_{kl})$. When $N_i = N$ $(i = 1, \ldots, m)$ we have $M \otimes N_i = M \otimes N$, the standard Kronecker product of M and N.

Our aim in this section is to solve the system $(A \otimes B_i)x = b$ in the least squares sense, by taking advantage of the Kronecker product structure of some of the generalized inverses of its coefficient matrix $A \otimes B_i$. We start by presenting the following definition [1,2]:

Definition 2 The *Moore-Penrose inverse* of the matrix $M \in \mathbb{R}^{m\times p}$, usually denoted by $M^\dagger$, is the unique matrix $G \in \mathbb{R}^{p\times m}$ satisfying the four *Penrose conditions*:

$$(1)\ MGM = M,\ \ (2)\ GMG = G,\ \ (3)\ (MG)^T = MG,\ \ (4)\ (GM)^T = GM,$$

where N^T is the transpose of the matrix N.

A matrix $G \in \mathbb{R}^{p\times m}$ satisfying (1) is called a *generalized inverse* or *g-inverse* of M.

A matrix $G \in \mathbb{R}^{p\times m}$ satisfying (1) and (2) is called a *reflexive g-inverse* of M.

A matrix $G \in \mathbb{R}^{p\times m}$ satisfying (1) and (3) is called a *least squares g-inverse* of M.

A matrix $G \in \mathbb{R}^{p\times m}$ satisfying (1) and (4) is called a *minimum norm g-inverse* of M.

The next definition will be very useful along this work.

Definition 3 Let M be a matrix of size $m \times p$ and let N_j $(j = 1, \ldots, p)$ be matrices of size $n \times q$. The *generalized Kronecker product by columns* of M and N_j is the matrix of size $mn \times pq$, denoted by $M \otimes_c N_j$, defined by the mp blocks of size $n \times q$ as

$$(m_{kl}N_l),$$

with $M = (m_{kl})$. When $N_j = N$ $(j = 1, \ldots, p)$ we have $M \otimes_c N_j = M \otimes N$, the standard Kronecker product of M and N.

The next theorem will be important in the solution of the problem considered in this section in the specific situation in which b belongs to the column space of $A \otimes B_i$, denoted by $\mathcal{R}(A \otimes B_i)$.

Theorem 1 *Let $A \in \mathbb{R}^{m\times p}$ be a matrix with $m \geq p$ and $rank(A) = p$, and let $B_i \in \mathbb{R}^{n\times q}$ $(i = 1, \ldots, m)$ be matrices with $n \geq q$ and $rank(B_i) = q$. The matrix*

$$G = A^\dagger \otimes_c B_i^\dagger$$

is a reflexive minimum norm g-inverse of the matrix $A \otimes B_i$.

Proof Since it is easily seen that $(A^\dagger \otimes_c B_i^\dagger)(A \otimes B_i) = I_{pq}$, the Penrose conditions (1), (2) and (4) are proved straightforward. The Penrose condition (3) is not satisfied in this case.

The following small example illustrates this situation.

Example 1 Let us consider the matrices

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, B_1 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}, B_2 = \begin{pmatrix} 5 \\ 6 \end{pmatrix} \text{ and } M = A \otimes \ B_i = \begin{pmatrix} 3 \\ 4 \\ 10 \\ 12 \end{pmatrix}.$$

The matrix

$$G = A^\dagger \otimes_c B_i^\dagger = \left(\tfrac{3}{125} \ \tfrac{4}{125} \ \tfrac{2}{61} \ \tfrac{12}{305} \right)$$

is not $M^\dagger$, since it does not satisfy the Penrose condition (3):

$$MG = \begin{pmatrix} \frac{9}{125} & \frac{12}{125} & \frac{6}{61} & \frac{36}{305} \\ \frac{12}{125} & \frac{16}{125} & \frac{8}{61} & \frac{48}{305} \\ \frac{6}{25} & \frac{8}{25} & \frac{20}{61} & \frac{24}{61} \\ \frac{36}{125} & \frac{48}{125} & \frac{24}{61} & \frac{144}{305} \end{pmatrix} \neq (MG)^T$$

Conditions (1), (2) and (4) are guaranteed by Theorem 1.

Remark 1 The fact that $(A^\dagger \otimes_c B_i^\dagger)(A \otimes B_i)$ is the identity matrix of order pq is a consequence of $A^\dagger A = I_p$ and $B_i^\dagger B_i = I_q$, which only can happen when $rank(A) = p$ and $rank(B) = q$. Therefore, the proof of Theorem 1 cannot be extended to the case where A or B_i have not full rank. Indeed, without the full rank restriction $A^\dagger \otimes_c B_i^\dagger$ is not necessarily a reflexive minimum norm g-inverse of the matrix $A \otimes B_i$, as can be seen by considering the following matrices:

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad B_3 = \begin{pmatrix} 3 & 3 \\ 3 & 3 \\ 3 & 3 \end{pmatrix}.$$

Now, let us observe here that, although $A^\dagger \otimes_c B_i^\dagger$ does not satisfy the Penrose condition (3) for $A \otimes B_i$, and in consequence $x = (A^\dagger \otimes_c B_i^\dagger)b$ is not a least squares solution of the linear system $(A \otimes B_i)x = b$, $A^\dagger \otimes_c B_i^\dagger$ is a minimum norm g-inverse of $A \otimes B_i$ (see Theorem 1) and therefore, when $b \in \mathcal{R}(A \otimes B_i)$, $x = (A^\dagger \otimes_c B_i^\dagger)b$ is the solution with minimum norm of $(A \otimes B_i)x = b$ (see, for example, Section 4.2 in [1]).

In the situation we are considering, i.e. when A and B_i $(i = 1, \ldots, m)$ have full rank, the equation $(A^\dagger \otimes_c B_i^\dagger)(A \otimes B_i) = I_{pq}$ is satisfied, and therefore $A \otimes B_i$ has full rank. Consequently, the least squares problem we are interested in solving has unique solution, and when $b \in \mathcal{R}(A \otimes B_i)$ it is the solution with minimum norm $x = (A^\dagger \otimes_c B_i^\dagger)b$. Let us show how to compute it by taking advantage of the generalized Kronecker product structure of $A^\dagger \otimes_c B_i^\dagger$.

$$x = (A^\dagger \otimes_c B_i^\dagger)b = \begin{pmatrix} \alpha_{11}B_1^\dagger & \alpha_{12}B_2^\dagger & \cdots & \alpha_{1m}B_m^\dagger \\ \alpha_{21}B_1^\dagger & \alpha_{22}B_2^\dagger & \cdots & \alpha_{2m}B_m^\dagger \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{p1}B_1^\dagger & \alpha_{p2}B_2^\dagger & \cdots & \alpha_{pm}B_m^\dagger \end{pmatrix} \begin{pmatrix} b^{(1)} \\ b^{(2)} \\ \\ \vdots \\ b^{(m)} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{11}B_1^\dagger b^{(1)} + \alpha_{12}B_2^\dagger b^{(2)} + \cdots + \alpha_{1m}B_m^\dagger b^{(m)} \\ \alpha_{21}B_1^\dagger b^{(1)} + \alpha_{22}B_2^\dagger b^{(2)} + \cdots + \alpha_{2m}B_m^\dagger b^{(m)} \\ \\ \vdots \\ \\ \alpha_{p1}B_1^\dagger b^{(1)} + \alpha_{p2}B_2^\dagger b^{(2)} + \cdots + \alpha_{pm}B_m^\dagger b^{(m)} \end{pmatrix},$$

where $A^\dagger = (\alpha_{kl})_{1\le k<p;1<l\le m}$, $b = (b_1, b_2, \ldots, b_{mn})^T$ and $b^{(i)} = (b_{(i-1)\cdot n+1}, b_{(i-1)\cdot n+2}, \ldots, b_{i\cdot n})^T$ for $i = 1, \ldots, m$.

The computation of $B_i^\dagger b^{(i)}$ is equivalent to solve the least squares problem $B_i y = b^{(i)}$. Denoting by $y^{(i)} = (y_1^{(i)}, y_2^{(i)}, \ldots, y_q^{(i)})^T$ the solution of this least squares problem, we obtain

$$x = \begin{pmatrix} \alpha_{11}y^{(1)} + \alpha_{12}y^{(2)} + \cdots + \alpha_{1m}y^{(m)} \\ \alpha_{21}y^{(1)} + \alpha_{22}y^{(2)} + \cdots + \alpha_{2m}y^{(m)} \\ \\ \vdots \\ \\ \alpha_{p1}y^{(1)} + \alpha_{p2}y^{(2)} + \cdots + \alpha_{pm}y^{(m)} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{11}\begin{pmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_q^{(1)} \end{pmatrix} + \alpha_{12}\begin{pmatrix} y_1^{(2)} \\ y_2^{(2)} \\ \vdots \\ y_q^{(2)} \end{pmatrix} + \cdots + \alpha_{1m}\begin{pmatrix} y_1^{(m)} \\ y_2^{(m)} \\ \vdots \\ y_q^{(m)} \end{pmatrix} \\ \alpha_{21}\begin{pmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_q^{(1)} \end{pmatrix} + \alpha_{22}\begin{pmatrix} y_1^{(2)} \\ y_2^{(2)} \\ \vdots \\ y_q^{(2)} \end{pmatrix} + \cdots + \alpha_{2m}\begin{pmatrix} y_1^{(m)} \\ y_2^{(m)} \\ \vdots \\ y_q^{(m)} \end{pmatrix} \\ \vdots \\ \alpha_{p1}\begin{pmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_q^{(1)} \end{pmatrix} + \alpha_{p2}\begin{pmatrix} y_1^{(2)} \\ y_2^{(2)} \\ \vdots \\ y_q^{(2)} \end{pmatrix} + \cdots + \alpha_{pm}\begin{pmatrix} y_1^{(m)} \\ y_2^{(m)} \\ \vdots \\ y_q^{(m)} \end{pmatrix} \end{pmatrix}.$$

Taking into account that

$$A^\dagger \begin{pmatrix} y_j^{(1)} \\ y_j^{(2)} \\ \vdots \\ y_j^{(m)} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{p1} & \alpha_{p2} & \cdots & \alpha_{pm} \end{pmatrix} \begin{pmatrix} y_j^{(1)} \\ y_j^{(2)} \\ \vdots \\ y_j^{(m)} \end{pmatrix},$$

for $j = 1, \ldots, q$, and that this computation is equivalent to find the solution of the least squares problem

$$Az = y_{(j)}, \quad \text{where} \quad y_{(j)} = \begin{pmatrix} y_j^{(1)} \\ y_j^{(2)} \\ \vdots \\ y_j^{(m)} \end{pmatrix},$$

the computation of x ends by solving these q least squares problems, all of them with the same coefficient matrix A.

In this way, the algorithm for solving $(A \otimes B_i)x = b$ ($b \in \mathcal{R}(A \otimes B_i)$) is the one presented in Algorithm 1.

Algorithm 1

Input: The matrices $A \in \mathbb{R}^{m \times p}$, $B_i \in \mathbb{R}^{n \times q}$ $(i = 1, \ldots, m)$ and the matrix $F \in \mathbb{R}^{m \times n}$ containing the data vector b sorted by rows.

Output: A matrix $X \in \mathbb{R}^{p \times q}$ containing the unique solution x of the least squares problem sorted by rows.

1: Solve the m least squares problems $B_i y = F^i$, where F^i is the i-th row of the matrix F. The unique solution y of the i-th least squares problem is stored in the i-th row of a matrix we call $M \in \mathbb{R}^{m \times q}$.

2: Solve the q least squares problems $Az = M^j$, where M^j is the j-th column of the matrix M. The unique solution z of the j-th least squares problem is stored in the j-th column of a matrix called $X \in \mathbb{R}^{p \times q}$. X is the output of the algorithm.

Looking at the algorithm is enough to observe that the solution of the bivariate least squares problem $(A \otimes B_i)x = b$, with $b \in \mathcal{R}(A \otimes B_i)$, is reduced to solving only m univariate least squares problems $B_i y = F^i$, and then q univariate least squares problems $Az = M^j$ with the same coefficient matrix A. The matrix $A \otimes B_i \in \mathbb{R}^{mn \times pq}$ is not constructed, which means additional savings in computational cost.

As all the univariate least squares problems we want to solve have unique solution (A and B_i have full rank), the method based on the QR decomposition is adequate for solving them [3,4]. In the specific case in which A and B_i $(i = 1, \ldots, m)$ are structured matrices, special algorithms that take advantage of the structure of A and B_i can be used for improving the QR method to solve the univariate problems [18,20,22,23].

We consider now the particular case of the least squares problem $(A \otimes B_i)x = b$ in which $B_i = B$ $(i = 1, \ldots, m)$.

Taking into account that $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$ (see, for instance, [2–4]), and proceeding in a similar way as we did to solve $(A \otimes B_i)x = b$ ($b \in \mathcal{R}(A \otimes B_i)$), we obtain Algorithm 2 for solving the least squares problem $(A \otimes B)x = b$, A and B being matrices with full column rank.

In this particular situation, the solution of the bivariate least squares problem $(A \otimes B)x = b$ is reduced to solving only m univariate least squares

Algorithm 2

Input: The matrices $A \in \mathbb{R}^{m\times p}$, $B \in \mathbb{R}^{n\times q}$ and the matrix $F \in \mathbb{R}^{m\times n}$ containing the data vector b sorted by rows.

Output: A matrix $X \in \mathbb{R}^{p\times q}$ containing the unique solution x of the least squares problem sorted by rows.

1: Solve the m least squares problems $By = F^i$, where F^i is the i-th row of the matrix F. The solution y of the i-th least squares problem is stored in the i-th row of a matrix we call $M \in \mathbb{R}^{m\times q}$.

2: Solve the q least squares problems $Az = M^j$, where M^j is the j-th column of the matrix M. The solution z of the j-th least squares problem is stored in the j-th column of a matrix called $X \in \mathbb{R}^{p\times q}$. X is the output of the algorithm.

problems $By = F^i$ with the same coefficient matrix B, and then q univariate least squares problems $Az = M^j$ with the same coefficient matrix A. Since A and B have full rank, it is convenient to solve each univariate least squares problem by using the QR method [3,4]. Taking into account that the computation of the QR factorization of a matrix $N \in \mathbb{R}^{m\times p}$ when the Q matrix is computed requires $4m^2p - 2p^2m + \frac{2}{3}p^3$ arithmetic operations [14], and that the QR factorization of the matrices A and B only have to be computed once, the computational complexity of our algorithm is of $O\big(p^2\big(q-2m+\frac{2}{3}p\big)+q^2\big(m-2n+\frac{2}{3}q\big)+4(m^2p+n^2q)+2mq(n+p)\big)$ arithmetic operations.

Let us point out here that the cost of solving this least squares problem by using the QR algorithm without exploiting the generalized Kronecker product structure of the $mn \times pq$ matrix $A \otimes B$ is much greater. This cost is of $O(4(mn)^2pq-2(pq)^2mn+\frac{2}{3}(pq)^3)$ arithmetic operations, since it is dominated by the cost of computing the QR factorization of $A \otimes B$ (Q is required).

A different method for solving the least squares problem $(A \otimes B)x = b$ can be found in [10]. Although it is also based on the QR factorization of the matrices A and B, this approach reduces the solution of the least squares problem to solving a block diagonal system with upper triangular diagonal blocks. A modified parallel implementation of the algorithm included in [10] is presented in [11]. Its computational complexity is of $O\big(p^2\big(q-2m+\frac{2}{3}p\big)+q^2\big(p-2n+\frac{2}{3}q\big)+4(m^2p+n^2q)+2mq(n+p)\big)$, and therefore similar to the complexity of our approach.

Let us observe here that our algorithm is also easily paralellizable. Once the QR factorization of B is computed, the other steps of the algorithm for solving the m least squares problems $By = F^i$ can be computed simultaneously [20]. The same happens to the solution of the q least squares problems $Az = M^j$: once the QR factorization of A is computed, the other steps of the algorithm for solving these least squares problems can be computed simultaneously. This high degree of intrinsic parallelism is a common feature of all algorithms presented in this work.

3 The particular case in which A is square and nonsingular

In this section we consider the least squares problem

$$(A \otimes B_i)x = b,$$

but in the particular situation in which A is a nonsingular square matrix of order m. The matrices B_i are of size $n \times q$ with $n \geq q$ and $rank(B_i) = q$.

As we will show in Section 4, this specific situation will be very interesting from the point of view of the applications. The following theorem will be crucial for its solution:

Theorem 2 *Let $A \in \mathbb{R}^{m\times m}$ nonsingular, and let $B_i \in \mathbb{R}^{n\times q}$ $(i = 1, \ldots, m)$ with $n \geq q$ and $rank(B_i) = q$. Then*

$$(A \otimes B_i)^\dagger = A^\dagger \otimes_c B_i^\dagger.$$

Proof From Theorem 1 we obtain that $A^\dagger \otimes_c B_i^\dagger$ satisfies the Penrose conditions (1), (2) and (4) for the matrix $A \otimes B_i$.

Now we prove the Penrose condition (3) is also satisfied in this case. As A is non nonsingular, $A^\dagger = A^{-1}$. Consequently $(A \otimes B_i)(A^\dagger \otimes_c B_i^\dagger)$ is a block diagonal matrix with symmetric diagonal blocks $B_i B_i^\dagger$, and therefore symmetric. So, $A^\dagger \otimes_c B_i^\dagger$ satisfies the four Penrose conditions for $A \otimes B_i$, and in consequence $(A \otimes B_i)^\dagger = A^\dagger \otimes_c B_i^\dagger$.

Taking Theorem 2 into account, the unique solution of $(A \otimes B_i)x = b$ when A is a nonsingular matrix can be expressed as $x = (A^\dagger \otimes_c B_i^\dagger)b$.

Proceeding as we did in Section 2 for computing $x = (A^\dagger \otimes_c B_i^\dagger)b$, but now taking into account that A is a nonsingular square matrix, we obtain Algorithm 3 for computing the unique solution $x = (A^\dagger \otimes_c B_i^\dagger)b$ of the least squares problem considered in this section.

Algorithm 3

Input: The matrices $A \in \mathbb{R}^{m\times m}$, $B_i \in \mathbb{R}^{n\times q}$ $(i = 1, \ldots, m)$ and the matrix $F \in \mathbb{R}^{m\times n}$, containing the data vector b sorted by rows.

Output: A matrix $X \in \mathbb{R}^{m\times q}$ containing the unique solution x of the least squares problem sorted by rows.

1: Solve the m least squares problems $B_i y = F^i$, where F^i is the i-th row of the matrix F. The solution y of the i-th least squares problem is stored in the i-th row of a matrix we call $M \in \mathbb{R}^{m\times q}$.

2: Solve the q linear systems $Az = M^j$, where M^j is the j-th column of the matrix M. The unique solution z of the j-th linear system is stored in the j-th column of a matrix called $X \in \mathbb{R}^{m\times q}$. X is the output of the algorithm.

As we will see in Section 4, in the particular case in which A and B_i $(i = 1, \ldots, m)$ are structured matrices, special algorithms that take advantage of the structure of these matrices can be used for solving the corresponding univariate linear systems and least squares problems.

4 Application to bivariate polynomial regression

In this section we will show how the solution of a bivariate regression problem can be reduced to the solution of a least squares problem of the type considered in the previous section, in which the matrices A and B_i will be certain structured matrices.

Our starting point is our work on bivariate interpolation in [24], which can be seen as a particular case of our bivariate polynomial regression problem. Throughout that paper, when the considered interpolation space is $\Pi_{mn}(s,t)$, the space of polynomials of degree less than or equal to m in s and less than or equal to n in t, the interpolation nodes are located along lines $s = s_i$ $(i = 0, \ldots, m)$ and denoted by $\{(s_i, t_{ij}) | i = 0, \ldots, m; j = 0, \ldots, n\}$. A remark explaining that the situation in which the interpolation nodes are located along lines $t = t_j$ is completely analogous is also included in that paper.

Following [24], a natural selection for the nodes involved in the computation of the regression polynomial $P(s,t) \in \Pi_{m,q}(s,t)$ is along lines $s = s_i$ $(i = 0, \ldots, m)$, where $s_i \neq s_j$ for $i \neq j$. Proceeding in this way, given the nodes $\{(s_i, t_{ij}) | i = 0, \ldots, m; j = 0, \ldots, n\} \in (0,1) \times (0,1)$, where $t_{ij} \neq t_{ik}$ for $j \neq k$, and $f_{ij} \in \mathbb{R}$ $(i = 0, \ldots, m; j = 0, \ldots, n)$ we are interested in computing the polynomial

$$P(s,t) = \sum_{(i,j) \in I} c_{ij} s^i t^j \in \Pi_{m,q}(s,t), \quad q \leq n$$

(where I is the index set $I = \{(i,j) |\ i = 0, \ldots, m; j = 0, \ldots, q\}$) which is a *least squares polynomial fit* for those data, i.e, which minimizes the sums of the squares of the deviations from the data:

$$\sum_{(i,j) \in I^*} |f_{ij} - P(s_i, t_{ij})|^2,$$

where $I^* = \{(i,j) |\ i = 0, \ldots, m; j = 0, \ldots, n\}$.

If we consider in the polynomial space $\Pi_{m,q}(s,t)$ the tensor product monomial basis

$$\begin{aligned} &\{s^i t^j | i = 0, \ldots, m; j = 0, \ldots, q\} = \\ &= \{1, t, \ldots, t^q, s, st, \ldots, st^q, \ldots, s^m, s^m t, \ldots, s^m t^q\} \end{aligned}$$

with this specific ordering, and the nodes with the corresponding ordering

$$\begin{aligned} &\{(s_i, t_{ij}) |\ i = 0, \ldots, m; j = 0, \ldots, n\} = \\ &= \{(s_0, t_{00}), (s_0, t_{01}), \ldots, (s_0, t_{0n}), \\ &\quad (s_1, t_{10}), (s_1, t_{11}), \ldots, (s_1, t_{1n}), \ldots, (s_m, t_{m0}), (s_m, t_{m1}), \ldots, (s_m, t_{mn})\} \end{aligned}$$

then, the computation of the coefficients c_{ij} $(i = 0, \ldots, m; j = 0, \ldots, q)$ of $P(s,t)$ is equivalent to solve, in the least squares sense, the overdetermined linear system

$$(V_s \otimes V_i) c = f,$$

where V_s and V_i $(i = 0, \ldots, m)$ are the Vandermonde matrices

$$V_s = \begin{pmatrix} 1 & s_0 & s_0^2 & \cdots & s_0^m \\ 1 & s_1 & s_1^2 & \cdots & s_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s_m & s_m^2 & \cdots & s_m^m \end{pmatrix}, \quad V_i = \begin{pmatrix} 1 & t_{i0} & t_{i0}^2 & \cdots & t_{i0}^q \\ 1 & t_{i1} & t_{i1}^2 & \cdots & t_{i1}^q \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_{in} & t_{in}^2 & \cdots & t_{in}^q \end{pmatrix}.$$

It must be observed that in this situation the matrix V_s is a nonsingular square matrix and the matrices V_i are rectangular matrices with full rank. Consequently $V_s \otimes V_i$ (with size $(m+1)(n+1) \times (m+1)(q+1)$) has full rank and, since V_s is a square matrix, the unique solution of the least squares problem can be obtained by using Algorithm 3.

In the specific situation in which $q = n$, the linear system $(V_s \otimes V_i)c = f$ (now with a square coefficient matrix) has a unique solution and it is the linear system corresponding to a bivariate interpolation problem in the tensor product monomial basis. The analysis of this interpolation problem and the one in which the tensor product monomial basis is replaced by the tensor product Bernstein basis has been developed in [24].

Let us also point out here that when $\{s_i\}_{0 \le i \le m}$ and $\{t_{ij}\}_{0 \le j \le n}$ $(i = 0, \ldots, m)$ are sorted in increasing order, i.e., $0 < s_0 < s_1 < \ldots < s_m < 1$ and $0 < t_{i0} < t_{i1} < \ldots < t_{in} < 1$, the Vandermonde matrices V_s and V_i are strictly totally positive. Taking advantage of this fact, Algorithm 3 can be improved by solving the univariate problems by using the accurate and efficient algorithms for totally positive matrices in the package TNTool of P. Koev [17].

Now we consider the same polynomial regression problem as before, but now taking in the polynomial space $\Pi_{m,q}(s,t)$ the tensor product Bernstein basis instead of the tensor product monomial basis.

Definition 4 The *Bernstein basis* of the space $\Pi_n(t)$ of the polynomials of degree less than or equal to n on the interval $[0,1]$ is:

$$\mathcal{B}_n = \left\{ \beta_i^{(n)}(t) = \binom{n}{i} (1-t)^{n-i} t^i, \ i = 0, \ldots, n \right\}.$$

The regression polynomial $P(s,t)$ we are interested in computing is expressed in the tensor product Bernstein basis as

$$P(s,t) = \sum_{(i,j) \in I} d_{ij} \beta_i^{(m)}(s) \beta_j^{(q)}(t)$$

and our aim now is computing its coefficients d_{ij} $(i = 0, \ldots, m; j = 0, \ldots, q)$.

If we consider the tensor product Bernstein basis

$$\{\beta_{ij}^{(m,q)} |\ i = 0, \ldots, m; j = 0, \ldots, q\} = \{\beta_i^{(m)}(s)\beta_j^{(q)}(t) |\ i = 0, \ldots, m; j = 0, \ldots, q\} =$$
$$= \{\beta_{00}^{(m,q)}, \beta_{01}^{(m,q)}, \ldots, \beta_{0q}^{(m,q)}, \beta_{10}^{(m,q)}, \beta_{11}^{(m,q)}, \ldots, \beta_{1q}^{(m,q)}, \ldots, \beta_{m0}^{(m,q)}, \beta_{m1}^{(m,q)}, \ldots, \beta_{mq}^{(m,q)}\}$$

with this precise ordering, and the nodes sorted as before, then the computation of the d_{ij} is equivalent to solve, in the least squares sense, the overdetermined linear system

$$(W_s \otimes W_i)d = f, \tag{4.1}$$

where W_s and W_i $(i = 0, \ldots, m)$ are the Bernstein-Vandermonde matrices

$$W_s = \begin{pmatrix} \binom{m}{0}(1-s_0)^m & \binom{m}{1}s_0(1-s_0)^{m-1} & \cdots & \binom{m}{m}s_0^m \\ \binom{m}{0}(1-s_1)^m & \binom{m}{1}s_1(1-s_1)^{m-1} & \cdots & \binom{m}{m}s_1^m \\ \vdots & \vdots & \ddots & \vdots \\ \binom{m}{0}(1-s_m)^m & \binom{m}{1}s_m(1-s_m)^{m-1} & \cdots & \binom{m}{m}s_m^m \end{pmatrix}, \tag{4.2}$$

$$W_i = \begin{pmatrix} \binom{q}{0}(1-t_{i0})^q & \binom{q}{1}t_{i0}(1-t_{i0})^{q-1} & \cdots & \binom{q}{q}t_{i0}^q \\ \binom{q}{0}(1-t_{i1})^q & \binom{q}{1}t_{i1}(1-t_{i1})^{q-1} & \cdots & \binom{q}{q}t_{i1}^q \\ \vdots & \vdots & \ddots & \vdots \\ \binom{q}{0}(1-t_{in})^q & \binom{q}{1}t_{in}(1-t_{in})^{q-1} & \cdots & \binom{q}{q}t_{in}^q \end{pmatrix}. \tag{4.3}$$

As the Bernstein-Vandermonde square matrix W_s is nonsingular and the rectangular Bernstein-Vandermonde matrices W_i have full rank [19], the matrix $W_s \otimes W_i$ (with size $(m+1)(n+1) \times (m+1)(q+1)$) has full rank and the unique solution of the least squares problem can be obtained by using Algorithm 3.

Let us observe that when $\{s_i\}_{0 \le i \le m}$ and $\{t_{ij}\}_{0 \le j \le n}$ $(i = 0, \ldots, m)$ are sorted in increasing order, i.e., $0 < s_0 < s_1 < \ldots < s_m < 1$ and $0 < t_{i0} < t_{i1} < \ldots < t_{in} < 1$, the Bernstein-Vandermonde matrices W_s and W_i are strictly totally positive [19].

The relationship between solving the regression problem we have considered when the monomial basis is used and when the Bernstein basis is used is analyzed in the following theorem. Before the theorem statement we introduce the following proposition, necessary for its proof.

Proposition 1 *Let $A \in \mathbb{R}^{m \times m}$, $C \in \mathbb{R}^{m \times m}$, $D \in \mathbb{R}^{q \times p}$ and $B_i \in \mathbb{R}^{n \times q}$ (where $i = 1, \ldots, m$ and $n \ge q$) be matrices with full rank. Then*

$$(A \otimes B_i)(C \otimes D) = AC \otimes B_i D$$

Proof

$$(A \otimes B_i)(C \otimes D) = \begin{pmatrix} a_{11}B_1 & a_{12}B_1 & \cdots & a_{1m}B_1 \\ a_{21}B_2 & a_{22}B_2 & \cdots & a_{2m}B_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B_m & a_{m2}B_m & \cdots & a_{mm}B_m \end{pmatrix} \begin{pmatrix} c_{11}D & c_{12}D & \cdots & c_{1m}D \\ c_{21}D & c_{22}D & \cdots & c_{2m}D \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1}D & c_{m2}D & \cdots & c_{mm}D \end{pmatrix}.$$

Multiplying the matrices in the right hand side by blocks we obtain that the ij block of $(A \otimes B_i)(C \otimes D)$ is

$$a_{i1}B_i c_{1j}D + a_{i2}B_i c_{2j}D + \ldots + a_{im}B_i c_{mj}D = (a_{i1}c_{1j} + a_{i2}c_{2j} + \ldots + a_{im}c_{mj})B_i D,$$

i.e., the ij block of $AC \otimes B_i D$. Therefore $(A \otimes B_i)(C \otimes D) = AC \otimes B_i D$.

Theorem 3 *Let $G = V_s \otimes V_i$ $(i = 1, \ldots, m)$ be the coefficient matrix of the linear system corresponding to the polynomial regression problem when the monomial basis is considered, and let $N = W_s \otimes W_i$ $(i = 1, \ldots, m)$ be the coefficient matrix of the linear system corresponding to the polynomial regression problem when the Bernstein basis is considered. Then*

$$N = GH \quad \textit{and} \quad c = Hd,$$

where $H = M_s \otimes M_t$ with M_s being the lower triangular matrix corresponding to the change of basis from the Bernstein basis to the monomial basis in $\Pi_m(s)$ and M_t being the lower triangular matrix corresponding to the change of basis form the Bernstein basis to the monomial basis in $\Pi_q(t)$.

Proof We start by proving $N = GH$ by using Proposition 1:

$$N = W_s \otimes W_i = V_s M_s \otimes V_i M_t = (V_s \otimes V_i)(M_s \otimes M_t) = GH.$$

In consequence, taking into account that $c = G^\dagger f$ and $d = N^\dagger f$, the second equality is proved as follows:

$$N = GH \Rightarrow N^\dagger = (GH)^\dagger = H^\dagger G^\dagger \Rightarrow$$
$$(W_s \otimes W_i)^\dagger = (M_s \otimes M_t)^\dagger (V_s \otimes V_i)^\dagger = (M_s \otimes M_t)^{-1}(V_s \otimes V_i)^\dagger \Rightarrow$$
$$(M_s \otimes M_t)(W_s \otimes W_i)^\dagger = (V_s \otimes V_i)^\dagger \Rightarrow$$
$$(M_s \otimes M_t)(W_s \otimes W_i)^\dagger f = (V_s \otimes V_i)^\dagger f \Rightarrow HN^\dagger f = G^\dagger f \Rightarrow Hd = c.$$

It is important to observe that, although Theorem 3 gives a matricial relationship between the regression polynomial in the Bernstein basis and the corresponding regression polynomial in the monomial basis, which can be used both for theoretical purposes and for computing in exact arithmetic, when computing in floating point arithmetic it is convenient to avoid basis conversion, since there is a potentially severe loss of accuracy in that basis conversion (see [8,9]).

Along this section we have considered the nodes $\{(s_i, t_{ij})\}$ in $(0,1) \times (0,1)$, and in this specific situation we have seen that each bivariate polynomial regression problem can always be solved by using the bivariate tensor product Bernstein basis. The extension of this result to the general situation in which the nodes $\{(s_i, t_{ij})\} \in [a,b] \times [c,d]$ can be obtained by proceeding in the same way as in our work on bivariate interpolation in [24]. An example illustrating how to proceed in this case is included in Section 5.

Now we present an algorithm for solving the least squares problem corresponding to the bivariate regression problem in the bivariate tensor product Bernstein basis. Given the nodes $\{(s_i, t_{ij}) | i = 0, \ldots, m; j = 0, \ldots, n\} \in (0,1) \times (0,1)$ and the data $\{f_{ij} | i = 0, \ldots, m; j = 0, \ldots, n\}$, the algorithm solves the overdetermined linear system $(W_s \otimes W_i)d = f$ in the least squares sense (see (4.1), (4.2) and (4.3)). From now on we consider $\{s_i\}$ and $\{t_{ij}\}$ $(i = 0, \ldots, m)$ sorted in increasing order, so that the Bernstein-Vandermonde

matrices involved in the solution of the regression problem will be strictly totally positive.

The algorithm presented here is based on Algorithm 3. As in this case the coefficient matrices of the least squares problems involved in the Step 1 of this algorithm are strictly totally positive Bernstein-Vandermonde matrices, we will solve them following [20]. We include here the implementation in MATLAB of the corresponding algorithm which we have called `TNLSB`. It solves the least squares problem $Ax = b$ where $A \in \mathbb{R}^{n\times q}$ is a strictly totally positive Bernstein-Vandermonde matrix and $B \in \mathbb{R}^{n\times q}$ is the matrix containing the bidiagonal decomposition of A [20], as follows:

Algorithm 4

```
1: function z=TNLSB(B,b)
2: [n,q]=size(B);
3: [Q,C]=TNQR(B);
4: C2=C(1:q,:);
5: Q1=Q(:,1:q);
6: d=Q1'*b';
7: z=TNSolve(C2,d);
```

The command `TNQR` computes the QR decomposition of a totally positive matrix A starting from its bidiagonal decomposition [18]. `TNSolve` solves the linear system $Ax = b$, where A strictly totally positive, starting from the bidiagonal decomposition of A and using backward substitution. The implementation in MATLAB of `TNQR` and `TNSolve` is available in the package for doing accurate computations with totally positive matrices TNTool of P. Koev [17].

The computational cost of `TNLSB` is dominated by the computational cost of computing the QR factorization by `TNQR`, and consecuently it is of $O(n^2q)$ arithmetic operations [20].

As for Step 2 of Algorithm 3, we will solve the Bernstein-Vandermonde linear systems involved in this step following [19].

So, let S be the $1 \times (m + 1)$ matrix with $\{s_i\}_{0\leq i\leq m}$ sorted as follows $0 < s_0 < s_1 < \ldots < s_m < 1$. Let T be the $(m + 1) \times (n + 1)$ matrix containing $\{t_{ij}\}_{0\leq i\leq m;0\leq j\leq n}$ in such a way that its (i, j) entry is $t_{i-1,j-1}$ and $0 < t_{i0} < t_{i1} < \ldots < t_{in} < 1$ for $i = 0, \ldots, m$. Let F be the $(m + 1) \times (n + 1)$ matrix with the data $\{f_{ij}\}$ sorted by rows, i.e., $f_{i-1,j-1}$ is the (i, j) entry of F. And let $q = deg_t(P(s, t))$, where $P(s, t)$ is the regression polynomial we are interested in computing. Algorithm 5 returns a matrix D of size $(m+1)\times(q+1)$ with the coefficients of the regression polynomial $P(s, t)$ sorted by rows, that is, the (i, j) entry of D is $d_{i-1,j-1}$, the coefficient of $P(s, t)$ corresponding to $\beta^{(m)}_{i-1}\beta^{(q)}_{j-1}$.

The command `TNBDBVR` computes the bidiagonal decomposition of a strictly totally positive rectangular Bernstein-Vandermonde matrix [21]. Its implementation in MATLAB can be obtained from [17].

Algorithm 5

```
 1: function D=LSKronGBVBD(S,T,F,q);
 2: [m1,n1]=size(F);
 3: M=zeros(m1,q+1);
 4: D=zeros(m1,q+1);
 5: for i=1:m1
 6:    Bt=TNBDBVR(T(i,:),q+1)
 7:    M(i,:)=TNLSB(Bt, F(i,:))'
 8: end
 9: Bs=TNBDBV(S);
10: for j=1:q+1
11:    D(:,j)=TNSolve(Bs, M(:,j));
12: end
13: D;
```

The computational cost of Algorithm 5 is of order $O(mn^2q)$ arithmetic operations, since the cost of solving each one of least squares problems involved in the process is of order $O(n^2q)$ arithmetic operations and the cost of solving each Bernstein-Vandermonde linear system of order m is of order $O(m^2)$ arithmetic operations.

As for the accuracy, the computation of the bidiagonal decomposition of the Bernstein-Vandermonde matrices by means of `TNBDBVR` is performed with high relative accuracy [21]. The computation of the QR factorization of a Bernstein-Vandermonde matrix by using `TNQR` is also performed accurately [18]. The high relative accuracy when `TNSolve` is used is only guaranteed when the data vector of the Bernstein-Vandermonde linear system has alternating sign pattern.

Finally, let us point out here that although an analogous algorithm can be developed for solving the bivariate regression problem when the tensor product monomial basis is used, the matrices involved in the process would be Vandermonde matrices instead of Bernstein-Vandermonde matrices. The fact that Bernstein-Vandermonde matrices are better conditioned than Vandermonde matrices [7] shows the convenience of solving the regression problem in the tensor product Bernstein basis instead of using the tensor product monomial basis.

Remark 2 As indicated in the proof of Theorem 2, if C is the coefficient matrix of the linear system corresponding to the polynomial regression problem (with the monomial basis or the Bernstein basis), then $CC^\dagger$ is a block diagonal matrix. Now we observe that $CC^\dagger$ is the corresponding projection matrix P (the *hat matrix* of statistics [22]). In addition, the diagonal blocks of $CC^\dagger$ are (in the notation of Theorem 3) $V_iV_i^\dagger$ or $W_iW_i^\dagger$.

Consequently, the projection $p = Pf$ and the *residual vector* $e = f - p$ can be computed very efficiently in this way.

5 Numerical examples

Two numerical examples illustrating the accuracy of the algorithm for computing a regression polynomial in the tensor product Bernstein basis presented in the previous section (Algorithm 5) are included. In the first one the nodes belong to the interval $(0,1)\times(0,1)$, while in the second the nodes are Padua points [5,6], and therefore they belong to $[-1,1]\times[-1,1]$.

Padua points are particularly well suited in this work because they are located along vertical lines, but not on a rectangular grid. This fact implies that the coefficient matrix of the linear system corresponding to the regression problem has a generalized Kronecker product structure, but not a Kronecker product structure.

Example 2 Let us consider the nodes

$$\{(s_i,t_{ij}),\ i=0,\dots,25;\ j=0,\dots,35\},$$

where $s_i,t_{ij}\in(0,1)$ are given by

$$\{s_i\}_{0\le i\le 25}=\{\tfrac{1}{50},\tfrac{3}{100},\tfrac{37}{1000},\tfrac{3}{50},\tfrac{1}{10},\tfrac{3}{20},\tfrac{1}{5},\tfrac{13}{50},\tfrac{3}{10},\tfrac{9}{25},\tfrac{2}{5},$$

$$\tfrac{43}{100},\tfrac{1}{2},\tfrac{27}{50},\tfrac{3}{5},\tfrac{31}{50},\tfrac{16}{25},\tfrac{71}{100},\tfrac{3}{4},\tfrac{17}{20},\tfrac{22}{25},\tfrac{9}{10},\tfrac{23}{25},\tfrac{19}{20},\tfrac{24}{25},\tfrac{99}{100}\},$$

$$\{t_{ij}\}_{0\le j\le 35}=\{\tfrac{21}{1000},\tfrac{27}{1000},\tfrac{1}{20},\tfrac{81}{1000},\tfrac{9}{100},\tfrac{3}{25},\tfrac{27}{200},\tfrac{4}{25},\tfrac{19}{100},\tfrac{1}{5},\tfrac{11}{50},\tfrac{6}{25},\tfrac{7}{25},\tfrac{3}{10},\tfrac{33}{100},$$

$$\tfrac{37}{100},\tfrac{39}{100},\tfrac{2}{5},\tfrac{9}{20},\tfrac{47}{100},\tfrac{1}{2},\tfrac{13}{25},\tfrac{3}{5},\tfrac{16}{25},\tfrac{27}{40},\tfrac{7}{10},\tfrac{73}{100},\tfrac{77}{100},\tfrac{79}{100},\tfrac{41}{50},\tfrac{17}{20},\tfrac{7}{8},\tfrac{23}{25},\tfrac{47}{50},\tfrac{193}{200},\tfrac{99}{100}\}$$

in the case of i even, and

$$\{t_{ij}\}_{0\le j\le 35}=\{\tfrac{1}{100},\tfrac{3}{100},\tfrac{1}{25},\tfrac{2}{25},\tfrac{1}{10},\tfrac{3}{20},\tfrac{17}{100},\tfrac{21}{100},\tfrac{23}{100},\tfrac{13}{50},\tfrac{29}{100},\tfrac{31}{100},\tfrac{8}{25},\tfrac{73}{200},\tfrac{41}{100},\tfrac{43}{100},$$

$$\tfrac{23}{50},\tfrac{19}{40},\tfrac{51}{100},\tfrac{53}{100},\tfrac{14}{25},\tfrac{57}{100},\tfrac{61}{100},\tfrac{63}{100},\tfrac{33}{50},\tfrac{17}{25},\tfrac{71}{100},\tfrac{3}{4},\tfrac{39}{50},\tfrac{4}{5},\tfrac{83}{100},\tfrac{43}{50},\tfrac{87}{100},\tfrac{91}{100},\tfrac{93}{100},\tfrac{49}{50}\}$$

in the case of i odd.

Given the data

$$\{f_{ij}=f(s_i,t_{ij}),\ i=0,\dots,25;\ j=0,\dots,35\},$$

where

$$f(s,t)=10(s+t)\sin(100(s+t)),$$

we compute the coefficients of the regression polynomial in the bivariate tensor product Bernstein basis $P(s,t)\in\Pi_{25,q}(s,t)$, for $q=15,17,19,21,23,25,27,29$, in *Maple* with 50-digit arithmetic, and use them for comparing the accuracy of the results obtained in MATLAB by means of:

(1) Algorithm 5.
(2) The command $A\backslash f$ of MATLAB.
(3) The command `pinv` of MATLAB.

The maximum relative errors obtained when using the approaches (1), (2) and (3) to compute the coefficients of $P(s,t)$ are included in Table 1. The spectral condition number κ_2 of the coefficient matrices corresponding to the considered regression problems are also presented in Table 1. These condition numbers are computed by using the standard MATLAB command `cond` and therefore it is likely that they are not accurate due to the severe ill conditioning of the matrices. Since we are only interested in showing that the matrices are ill conditioned, but not in computing their spectral condition numbers accurately, the results shown in Table 1 are good enough for our purpose. In order to compute these spectral condition numbers accurately, special algorithms that take into account the structure of the coefficient matrices must be used [21].

q	$\kappa_2(A)$	Algorithm 5	$A\backslash f$	`pinv`
15	3.9e+13	1.4e-14	9.2e-03	1.3e+01
17	1.8e+14	1.5e-14	5.0e+00	1.9e+00
19	9.4e+14	3.8e-14	1.4e+00	1.5e+00
21	5.3e+15	5.0e-15	1.1e+00	1.3e+00
23	2.3e+16	6.9e-15	1.6e+00	1.0e+00
25	3.4e+16	5.4e-15	1.1e+00	1.1e+00
27	6.2e+16	1.2e-14	1.0e+00	1.0e+00
29	9.1e+16	3.4e-15	1.0e+00	1.0e+00

Table 1: Relative errors in Example 2

The results presented in Table 1 show that our algorithm computes the coefficients of the regression polynomial accurately for every value of q. The results obtained when using the other two methods are not accurate at all.

Example 3 In this example the nodes

$$\{(x_i, y_{ij}),\ i = 0, \ldots, 20;\ j = 0, \ldots, 10\},$$

are the Padua points [5,6] for $N = 20$, that is,

$$x_i = \cos\left(\tfrac{(20-i)\pi}{20}\right),\ \text{with } i = 0, \ldots, 20,$$

$$y_{ij} = \begin{cases} \cos\left(\tfrac{(21-2j)\pi}{21}\right), & i \text{ even},\ j = 0, \ldots, 10, \\ \cos\left(\tfrac{(20-2j)\pi}{21}\right), & i \text{ odd},\ j = 0, \ldots, 10, \end{cases}$$

and therefore $(x_i, y_{ij}) \in [-1,1] \times [-1,1]$.

Let us observe here that the Padua points are properly ordered so that the Bernstein-Vandermonde matrices involved in the process are strictly totally positive.

Proceeding as in [24], two changes of variables are performed,

$$s = \frac{1}{20}(9x + 10), \quad t = \frac{1}{20}(9y + 10),$$

taking the Padua points in [-1,1]×[-1,1] to (0,1)×(0,1).

Given the data

$$\{f_{ij} = f(x_i, y_{ij}),\ i = 0, \ldots, 20;\ j = 0, \ldots, 10\},$$

where

$$f(x,y) = \tfrac{3}{4}e^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \tfrac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}} + \tfrac{1}{2}e^{-\frac{(9x-7)^2+(9y-3)^2)}{4}} - \tfrac{1}{5}e^{-(9x-4)^2-(9y-7)^2}$$

is the Franke test function presented in [12], we compute the coefficients of the regression polynomial in the bivariate tensor product Bernstein basis $R(s,t) \in \Pi_{20,q}(s,t)$, for $q = 5, 6, 7, 8, 9$, in *Maple* with 50 digit arithmetic, and use them for comparing the accuracy of the results obtained in MATLAB by Algorithm 5 and the results obtained by using the commands $A\backslash f$ and `pinv` of MATLAB. The maximum relative errors in the computed coefficients, as well as the spectral condition numbers κ_2 of the coefficient matrices corresponding to the considered regression problems are shown in Table 2.

q	$\kappa_2(A)$	Algorithm 5	$A\backslash f$	`pinv`
5	9.7e+07	1.3e-11	1.4e-06	1.1e-06
6	2.1e+08	1.2e-12	6.8e-07	1.1e-06
7	4.6e+08	6.9e-12	1.8e-06	5.2e-07
8	1.0e+09	5.0e-12	1.7e-06	5.0e-07
9	2.2e+09	1.1e-13	6.5e-08	8.2e-08

Table 2: Relative errors in Example 3

6 Conclusions and final remarks

In this work we have presented a fast and accurate method for solving bivariate least squares problems involving generalized Kronecker product structure $(A \otimes B_i)x = b$. The good properties of our approach are a consequence of the generalized Kronecker product structure (by columns) of the Moore-Penrose inverse of the coefficient matrix. In addition to the Moore-Penrose inverse, in the general case where A is not square the reflexive minimum norm g-inverse of the coefficient matrix arises naturally when considering the generalized Kronecker product.

The application to the important problem of computing the bivariate regression polynomial when a bivariate tensor product basis is taken is also developed, and it is shown that in this case it is natural to consider that A is a square matrix. In this case the projection matrix has a block diagonal structure.

When the bivariate tensor product Bernstein basis is considered, the coefficient matrix $A \otimes B_i$ of the least squares problem corresponding to the regression problem is the generalized Kronecker product of Bernstein-Vandermonde matrices. Such matrices do not have to be constructed since our algorithm only works with the nodes and the data, which means an additional saving of computational costs.

Finally, let us observe that our algorithm is easily paralellizable. All the univariate least squares problems in Step 1 can be solved simultaneously, and the same happens to the univariate least squares problems in Step 2.

Acknowledgements This research has been partially supported by Spanish Research Grant MTM2015-65433-P (MINECO/FEDER) from the Spanish Ministerio de Economía y Competitividad. A. Marco, J. J. Martínez and R. Viaña are members of the Research Group ASYNACS (Ref. CCEE2011/R34) of Universidad de Alcalá.

We are also grateful to the anonymous reviewers for their suggestions, which have contributed to improve our paper.

References

1. Bapat, R. B.: Linear Algebra and Linear Models, 3rd edition. Springer (2012)
2. Ben-Israel, A., Greville, T. N. E.: Generalized Inverses, Theory and Applications, 2nd edition. Springer-Verlag (2003)
3. Björck, A.: Numerical Methods for Least Squares Problems, SIAM, Philadelphia (1996)
4. Björck, A.: Numerical Methods in Matrix Computations, Texts in Applied Mathematics, Springer (2016)
5. Caliari, M., De Marchi, S., Vianello, M.: Algorithm 886: Padua2D-Lagrange interpolation at Padua points on bivariate domains. ACM Transactions on Mathematical Software 35, Article 21 (2008)
6. Caliari, M., De Marchi, S., Sommariva, A., Vianello, M.: Padua2DM: fast interpolation and cubature at the Padua points in Matlab/Octave. Numer. Algor. 56, 45–60 (2011)
7. Delgado, J., Peña, J. M.: Optimal conditioning of Bernstein collocation matrices. SIAM J. Matrix Anal. Appl. 31, 990–996 (2009)
8. Farouki, R. T., Rajan, V. T.: On the numerical condition of polynomials in Bernstein form. Comput. Aided Geom. Design 4, 191-216 (1987)
9. Farouki, R. T., Rajan, V. T.: Algorithms for polynomials in Bernstein form. Comput. Aided Geom. Design 5, 1-26 (1988)
10. Fausett, D. W., Fulton, C. T.: Large least squares problems involving Kronecker products. SIAM J. Matrix Anal. Appl. 15, 219-227 (1994)
11. Fausett, D. W., Fulton, C. T., Hashish, H.: Improved parallel QR method for large least squares problems involving Kronecker products. J. Comput. Appl. Math. 78, 63-78 (1997)
12. Franke, R.: A critical comparison of some methods for interpolation of scattered data. Naval Postgraduate School Tech. Rep. NPS-53-79-003 (1979)
13. Gasca, M., Martínez J.-J.: On the solvability of bivariate Hermite-Birkhoff interpolation problems. J. Comput. Appl. Math. 32, 77-82 (1990)
14. Golub, G. H., Van Loan, C. F.: Matrix Computations, 4th edition, The Johns Hopkins University Press, Baltimore (2012)
15. Grosse, E.: Tensor spline approximations, Linear Algebra Appl. 34, 29–41 (1980)
16. Hansen P. C.: Deconvolution and regularization with Toepliz matrices, Numer. Algor. 29, 323–378 (2002)
17. Koev, P.: `http://www.math.sjsu.edu/∼koev`
18. Koev, P.: Accurate computations with totally nonnegative matrices. SIAM J. Matrix Anal. Appl. 29, 731–751 (2007)
19. Marco, A., Martínez J.-J.: A fast and accurate algorithm for solving Bernstein-Vandermonde linear systems. Linear Algebra Appl. 422, 616–628 (2007)

20. Marco, A., Martínez J.-J.: Polynomial least squares fitting in the Bernstein basis. Linear Algebra Appl. 433, 1254–1264 (2010)
21. Marco, A., Martínez J.-J.: Accurate computations with totally positive Bernstein-Vandermonde matrices. Electronic Journal of Linear Algebra 26, 357–380 (2013)
22. Marco, A., Martínez J.-J.: Ajuste polinómico por mínimos cuadrados usando la base de Bernstein. La Gaceta de la Real Sociedad Matemática Española 18, 135–153 (2015)
23. Marco, A., Martínez J.-J.: Bidiagonal decomposition of rectangular totally positive Said-Ball-Vandermonde matrices: Error analysis, perturbation theory and applications. Linear Algebra Appl. 495, 90–107 (2016)
24. Marco, A., Martínez J.-J., Viaña, R.: Accurate polynomial interpolation by using the Bernstein basis. Numer. Algor. 75, 665–674 (2017)
25. Martínez, J.-J.: A generalized Kronecker product and linear systems. International Journal of Mathematical Education in Science and Technology 30, 137–141 (1999)
26. Pisinger, G., Zimmermann, A.: Linear least squares problems with data over incomplete grids. BIT Numerical Mathematics 47, 809–824 (2007)
27. Regalia, P. A., Mitra, S. K.: Kronecker products, unitary matrices and signal processing applications. SIAM Review 31, 586–613 (1989)
28. Van Loan C. F.: The ubiquitous Kronecker product. J. Comput. Appl. Math. 123, 85–100 (2000)