ARTICLE IN PRESS

Journal of Computational and Applied Mathematics xx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Q1 Accurate computation of the Moore–Penrose inverse of strictly totally positive matrices

Q2 Ana Marco, José-Javier Martínez *

Departamento de Física y Matemáticas, Universidad de Alcalá, Alcalá de Henares, Madrid 28871, Spain

ARTICLE INFO

Article history:
Received 14 November 2017
Received in revised form 18 June 2018

MSC:
65F20
65F05
65F35
15B05
15B48
15A23
15A09

Keywords:
Moore–Penrose inverse
Inverse
Totally positive matrix
Neville elimination
Bidiagonal decomposition
High relative accuracy

ABSTRACT

The computation of the Moore–Penrose inverse of structured strictly totally positive matrices is addressed. Since these matrices are usually very ill-conditioned, standard algorithms fail to provide accurate results. An algorithm based on the *QR* factorization and which takes advantage of the special structure and the totally positive character of these matrices is presented. The first stage of the algorithm consists of the accurate computation of the bidiagonal decomposition of the matrix. Numerical experiments illustrating the good behavior of our approach are included.

1. Introduction

The *Moore–Penrose inverse* of the matrix $A \in \mathbb{R}^{m\times n}$, usually denoted by $A^\dagger$, is the unique matrix $X \in \mathbb{R}^{n\times m}$ satisfying the four *Penrose conditions*:

(i) $AXA = A$,
(ii) $XAX = X$,
(iii) $(AX)^T = AX$,
(iv) $(XA)^T = XA$,

where B^T is the transpose of the matrix B. It was first introduced and studied by E. H. Moore during the decade of 1910–1920, and later rediscovered independently by A. Bjerhammar and by R. Penrose during the decade of 1950–1960 [1].

One of its main applications is to use it for solving least squares problems, and so the Moore–Penrose inverse arises in several fields such as orthogonal polynomials over discrete domains, computing multilinear regression coefficients [2], designing neural learning algorithms [3] or restoration of digital images [4]. A generalization of the Moore–Penrose inverse, called the weighted Moore–Penrose inverse, has recently been analyzed for instance in [5].

* Corresponding author.
E-mail addresses: ana.marco@uah.es (A. Marco), jjavier.martinez@uah.es (J.-J. Martínez).

https://doi.org/10.1016/j.cam.2018.10.009
0377-0427/

There are several types of methods for computing the Moore–Penrose inverse. The direct methods are usually based on singular value decomposition (as for example the command `pinv` from MATLAB) and QR factorization [6,7]. See also [8] for a comparison of different direct methods. As for the iterative methods, some of them are based on generalizations of the hyper-power method and the Schultz method as its particular case [9,10]. An iterative method based on Penrose equations is presented in [11] and improved in [12].

As recalled in [13], many types of structured matrices arising in applications are extremely ill-conditioned and therefore standard algorithms applied to them do not guarantee relative accuracy. Fortunately, the specific structure of those matrices allows the design of algorithms that, taking that structure into account, will provide a more accurate solution in spite of their high condition number.

In particular we will consider several classes of ill-conditioned totally positive matrices for which we will compute their Moore–Penrose inverse accurately. A matrix is *totally positive* (or *totally nonnegative*) if all its minors are nonnegative. If all its minors are positive the matrix is called *strictly totally positive* [14,15].

As it can be read in [16,17], a crucial preliminary stage for our algorithms for the class of totally positive matrices is the decomposition of the matrix as a product of bidiagonal factors. Starting from an accurate bidiagonal decomposition of A we develop a method for computing $A^\dagger$ based on the QR factorization of A, which includes the inversion of an upper triangular matrix. When A is strictly totally positive, the matrix R in $A = QR$ is nonsingular totally positive.

The remainder of the paper is organized as follows. Section 2 introduces Neville elimination, a key theoretical tool for our approach, and presents the results on the bidiagonal factorization of totally positive matrices, and of their inverses in the square case. In Section 3 we provide an accurate algorithm for computing the Moore–Penrose inverse of a strictly totally positive matrix, while Section 4 contains an algorithm for the computation of the inverse of a nonsingular totally positive matrix. The error analysis of the algorithm is also included in this section. Several numerical experiments are presented in Section 5, and Section 6 is devoted to conclusions.

2. Neville elimination and bidiagonal decomposition

To make this paper as self-contained as possible, we will briefly recall in this section some basic results on Neville elimination, a fundamental theoretical tool for obtaining the results presented in this paper.

Neville elimination is a procedure that makes zeros in a matrix adding to a given row an appropriate multiple of the previous one. Our notation follows the notation used in [18] and [19].

Let $A = (a_{i,j})_{1\leq i,j\leq n}$ be a square matrix of order n. The Neville elimination of A consists of $n-1$ steps resulting in a sequence of matrices $A_1 := A \to A_2 \to \ldots \to A_n$, where $A_t = (a_{i,j}^{(t)})_{1\leq i,j\leq n}$ has zeros below its main diagonal in the $t-1$ first columns. The matrix A_{t+1} is obtained from A_t $(t = 1, \ldots, n-1)$ by using the following formula:

$$a_{i,j}^{(t+1)} := \begin{cases} a_{i,j}^{(t)}, & \text{if} \quad i \leq t \\ a_{i,j}^{(t)} - (a_{i,t}^{(t)}/a_{i-1,t}^{(t)})a_{i-1,j}^{(t)}, & \text{if} \quad i \geq t+1 \text{ and } j \geq t+1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

In this process the element

$$p_{i,j} := a_{i,j}^{(j)} \qquad 1 \leq j \leq n, \ D\,Dj \leq i \leq n$$

is called (i,j) *pivot* of the Neville elimination of A. The process would break down if any of the pivots $p_{i,j}$ $(j \leq i < n)$ is zero. In that case we can move the corresponding rows to the bottom and proceed with the new matrix, as described in [18]. The Neville elimination can be done without row exchanges if all the pivots are nonzero, as it will happen in our situation. The pivots $p_{i,i}$ are called *diagonal pivots*. The element

$$m_{i,j} = \frac{p_{i,j}}{p_{i-1,j}} \qquad 1 \leq j \leq n-1, \ \ j < i \leq n, \tag{2.2}$$

is called *multiplier* of the Neville elimination of A. The matrix $U := A_n$ is upper triangular and has the diagonal pivots on its main diagonal.

The *complete Neville elimination* of a matrix A consists of performing the Neville elimination of A for obtaining U and then continue with the Neville elimination of U^T. The (i,j) pivot (respectively, multiplier) of the complete Neville elimination of A is the (j,i) pivot (respectively, multiplier) of the Neville elimination of U^T, if $j \geq i$. When no row exchanges are needed in the Neville elimination of A and U^T, we say that the complete Neville elimination of A can be done without row and column exchanges, and in this case the multipliers of the complete Neville elimination of A are the multipliers of the Neville elimination of A if $i \geq j$ and the multipliers of the Neville elimination of A^T if $j \geq i$ (see p. 116 of [20]).

The extension of Neville elimination to the rectangular case is straightforward.

Neville elimination characterizes the strictly totally positive matrices and the nonsingular totally positive matrices as follows [19]:

Theorem 2.1. *A matrix is strictly totally positive (resp. nonsingular totally positive) if and only if its complete Neville elimination can be performed without row and column exchanges, the multipliers of the Neville elimination of A and A^T are positive (resp. nonnegative), and the diagonal pivots of the Neville elimination of A are positive.*

Neville elimination also gives the bidiagonal decomposition of the inverse A^{-1} of a nonsingular totally positive matrix A [20,17]:

Theorem 2.2. *Let $A = (a_{i,j})_{1\le i,j\le n}$ be a nonsingular totally positive matrix. Then A^{-1} admits a factorization in the form*

$$A^{-1} = G_1 G_2 \cdots G_{n-1} D^{-1} F_{n-1} F_{n-2} \cdots F_1,$$

where F_i $(i = 1, \ldots, n-1)$ are $n \times n$ bidiagonal matrices of the form

$$F_i = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 0 & 1 & & & & \\ & & & -m_{i+1,i} & 1 & & & \\ & & & & -m_{i+2,i} & 1 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & -m_{n,i} & 1 \end{bmatrix},$$

G_i $(i = 1, \ldots, n-1)$ are $n \times n$ bidiagonal matrices of the form

$$G_i^T = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 0 & 1 & & & & \\ & & & -\widetilde{m}_{i+1,i} & 1 & & & \\ & & & & -\widetilde{m}_{i+2,i} & 1 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & -\widetilde{m}_{n,i} & 1 \end{bmatrix}$$

and D is a diagonal matrix of order n

$$D = \operatorname{diag}\{p_{1,1}, p_{2,2}, \ldots, p_{n,n}\}.$$

The quantities $m_{i,j}$, $\widetilde{m}_{i,j}$ and $p_{i,i}$ are respectively the multipliers of the Neville elimination of A, the multipliers of the Neville elimination of A^T, and the diagonal pivots of the Neville elimination of A.

Now we present the bidiagonal decomposition of a strictly totally positive matrix A given by Neville elimination [20,17]:

Theorem 2.3. *Let $A = (a_{i,j})_{1\le i\le l;\, 1\le j\le n}$ be a strictly totally positive matrix. Then A admits a factorization in the form*

$$A = F_{l-1} F_{l-2} \cdots F_1 D G_1 \cdots G_{n-2} G_{n-1},$$

where F_i $(i = 1, \ldots, l-1)$ are $l \times l$ bidiagonal matrices of the form

$$F_i = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 0 & 1 & & & & \\ & & & m_{i+1,1} & 1 & & & \\ & & & & m_{i+2,2} & 1 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & m_{l,l-i} & 1 \end{bmatrix},$$

G_i^T $(i = 1, \ldots, n-1)$ are $n \times n$ bidiagonal matrices of the form

$$G_i^T = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 0 & 1 & & & & \\ & & & \widetilde{m}_{i+1,1} & 1 & & & \\ & & & & \widetilde{m}_{i+2,2} & 1 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & \widetilde{m}_{n,n-i} & 1 \end{bmatrix},$$

and D is the $l \times n$ diagonal matrix

$$D = (d_{i,j})_{1\le i\le l;\, 1\le j\le n} = \operatorname{diag}\{p_{1,1}, p_{2,2}, \ldots, p_{n,n}\}.$$

The quantities $m_{i,j}$, $\widetilde{m}_{i,j}$ and $p_{i,i}$ are respectively the multipliers of the Neville elimination of A, the multipliers of the Neville elimination of A^T, and the diagonal pivots of the Neville elimination of A.

As it can be seen in [17], given a $m \times n$ strictly totally positive matrix A, its bidiagonal factorization can be stored in a $m \times n$ matrix called $\mathcal{BD}(A)$ whose diagonal elements are the diagonal pivots $p_{i,i}$ of the Neville elimination of A, the elements above its diagonal are the multipliers $m_{i,j}$ of the Neville elimination of A, and the elements below its diagonal are the multipliers $\widetilde{m}_{i,j}$

of the Neville elimination of A^T. Let us observe that by Theorem 2.2 the matrix $\mathcal{BD}(A)$ also serves to represent the bidiagonal factorization of A^{-1} when A is nonsingular.

Let us point out here that, although the Neville elimination is the theoretical tool that allows us to obtain the bidiagonal factorization of a strictly totally positive matrix A, in practice the computation of the matrix $\mathcal{BD}(A)$ representing this decomposition is carried out by means of algorithms that compute the pivots and the multipliers of A and the multipliers of A^T by using explicit expressions for them [17]. These algorithms are designed taking into account the specific structure of the matrix A and guarantee high relative accuracy. Examples of different classes of totally positive matrices for which $\mathcal{BD}(A)$ has been computed accurately are: Cauchy–Vandermonde [21,22], generalized Vandermonde [23], Vandermonde and Cauchy [17], Bernstein–Vandermonde [24,25], collocation matrices of rational bases [26], Said–Ball–Vandermonde [27] and Lupaş matrices [28].

Let us observe that, following the ideas presented in [29], in [22,25,27] structured condition numbers for the computation of the matrix $\mathcal{BD}(A)$ are found.

3. Computation of the Moore–Penrose inverse

Let A be a $m \times n$ ($m \geq n$) strictly totally positive matrix. In this section we present an accurate algorithm for computing the Moore–Penrose inverse $A^\dagger$ of A.

It is based on the explicit expression of $A^\dagger$ obtained by means of the QR factorization of A, as shown in the next theorem:

Theorem 3.1. *Let A be an $m \times n$ matrix, $m \geq n$, $rank(A) = n$. Let*

$$A = QR, \qquad R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

be the QR factorization of A, where Q is an $m \times m$ orthogonal matrix, R is an $m \times n$ upper triangular matrix with positive diagonal entries and R_1 is the $n \times n$ matrix with the first n rows of R. Let Q_1 be the $m \times n$ matrix with the first n columns of Q, then the pseudoinverse of A is

$$A^\dagger = R_1^{-1} Q_1^T.$$

Proof. Taking into account that $A^\dagger = (A^T A)^{-1} A^T$ (the four Moore–Penrose conditions are easily checked) and that

$$A^T A = \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = R_1^T R_1,$$

we have

$$A^\dagger = (A^T A)^{-1} A^T = (R_1^T R_1)^{-1} \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T = R_1^{-1} R_1^{-T} R_1^T Q_1^T = R_1^{-1} Q_1^T. \quad \square$$

For computing R_1 and Q_1 we must start from the accurate computation of the matrix $\mathcal{BD}(A)$ containing the bidiagonal decomposition of A. The algorithm follows:

INPUT: The strictly totally positive matrix A.

OUTPUT: The Moore–Penrose inverse $A^\dagger$ of A.

- *Step 1:* Computation of the matrix $\mathcal{BD}(A)$ containing the bidiagonal decomposition of A. For the structured matrices we will be considering, the construction of A is not necessary for computing $\mathcal{BD}(A)$.
- *Step 2:* Given the result of Step 1, computation of the QR factorization of A by means of the algorithm `TNQR`.
- *Step 3:* Given the result of Step 2, computation of the inverse R_1^{-1} by using the algorithm `TNInverseExpand`.
- *Step 4:* Computation of $R_1^{-1} Q_1^T$.

The algorithm `TNQR` of P. Koev can be obtained from [30], and given $\mathcal{BD}(A)$ computes accurately the QR factorization of A. It returns the $m \times m$ matrix Q and the matrix $\mathcal{BD}(R)$ representing the bidiagonal decomposition of the totally positive matrix R of size $m \times n$. The computational complexity of `TNQR` is of $O(m^2 n)$ arithmetic operations [17].

`TNInverseExpand` is the name that we have given to the algorithm we present in Section 4 for computing the inverse of a nonsingular totally positive matrix starting from its bidiagonal decomposition, and it has been included by P. Koev in his package TNTool [30]. As we will see in the next section, it has high relative accuracy and its computational cost is of $O(n^2)$ arithmetic operations.

The product in Step 4 is carried out by using the standard multiplication command from MATLAB.

Let us recall that when A has full column rank (which implies $m \geq n$), then $x = A^\dagger b$ is the unique solution (in the least squares sense) of the overdetermined linear system $Ax = b$.

We will now see how this approach for computing $A^\dagger$ can also be extended to the case $m < n$. In this situation the least squares problem has not a unique solution, but we can also compute the (unique) Moore–Penrose inverse of A.

If A is a $m \times n$ strictly totally positive matrix with $m < n$, then $B = A^T$ has full column rank and so the results we have presented above can be applied to B. Therefore we have $B^\dagger = (B^T B)^{-1} B^T$, which can be computed by means of our algorithm.

Please cite this article in press as: A. Marco, J.-J. Martínez, Accurate computation of the Moore–Penrose inverse of strictly totally positive matrices, Journal of Computational and Applied Mathematics (2018), https://doi.org/10.1016/j.cam.2018.10.009.

Now we use the fact that $(B^T)^\dagger = (B^\dagger)^T$ [31], and consequently

$$A^\dagger = (B^T)^\dagger = (B^\dagger)^T.$$

For computing the bidiagonal factorization of $B = A^T$ we use the fact that $\mathcal{BD}(A^T) = (\mathcal{BD}(A))^T$ [17].

Remark. In this case ($m < n$ with $rank(A) = m$) the corresponding least squares problem is in fact an underdetermined linear system $Ax = b$. The unique solution with minimum norm $\|x\|_2$ is

$$x^\dagger = A^\dagger b = (B^\dagger)^T b = ((B^T B)^{-1} B^T)^T b = B(B^T B)^{-1} b = A^T (AA^T)^{-1} b.$$

Let us observe that:

(i) $x^\dagger$ is a solution of $Ax = b$, since $Ax^\dagger = AA^T(AA^T)^{-1}b = b$.
(ii) $x^\dagger$ belongs to the row space of A, since it is a linear combination of the columns of A^T.

Consequently, the results included in Section 6.3 of [32] and in Section 2.1 of [33] show (without using the concept of singular value decomposition) that $x^\dagger$ is the unique solution with minimum norm $\|x\|_2$ of the underdetermined linear system $Ax = b$.

4. Computation of the inverse

In this section we present an accurate and fast algorithm for computing the inverse of a nonsingular totally positive matrix A starting from $\mathcal{BD}(A)$, the matrix representing the bidiagonal decomposition of A. This is the algorithm we use for computing the inverse of the upper triangular matrix R_1 involved in the computation of the Moore–Penrose inverse of A (see Section 3). We have called it `TNInverseExpand` and its error analysis is also included in this section.

The fact that the matrix $\mathcal{BD}(A)$ represents both the bidiagonal decomposition of A and of its inverse A^{-1} (see Section 2), and the algorithm of P. Koev `TNExpand`, which given $\mathcal{BD}(A)$ recovers the matrix A [30], have been the starting points in the development of our algorithm `TNInverseExpand`.

Let A be the square matrix whose inverse C we are interested in computing, and let B be the matrix containing $\mathcal{BD}(A)$. The code in MATLAB of the algorithm `TNInverseExpand` for the computation of C starting from B is the following:

```
[n,n]=size(B);
C=zeros(n,n);
for i=1:n
  C(i,i)=1;
end
for i=1:n-1
  for j=n:-1:i+1
    C(:,j)=C(:,j)-B(i,j)*C(:,j-1);
  end
end
for i=1:n
  C(:,i)=C(:,i)/B(i,i);
end
for i=n-1:-1:1
  for j=i:n-1
    C(:,j)=C(:,j)-B(j+1,i)*C(:,j+1);
  end
end
```

Looking at the loops of the algorithm is enough to see that it has a computational cost of $O(n^2)$ arithmetic operations. The fact that all the diagonal entries of B are positive and all the non diagonal entries of B are nonnegative (B is the matrix containing $\mathcal{BD}(A)$, and A is nonsingular totally positive), and the checkerboard sign pattern of the identity matrix and of all the matrices C involved in the computation of A^{-1}, guarantee that `TNInverseExpand` is subtraction free, and therefore, A^{-1} is computed with high relative accuracy once the matrix $BD(A)$ is computed with high relative accuracy.

For the error analysis of the algorithm we use the standard model of floating point arithmetic (see section 2.2 of [34]):
Let x, y be floating point numbers and ϵ be the *machine precision*,

$$fl(x \odot y) = (x \odot y)(1+\delta)^{\pm 1}, \quad \text{where } D|\delta| \leq \epsilon,\ D\ D\odot \in \{+, -, \times, /\}.$$

The following theorem shows the maximum relative error that can be committed by our algorithm `TNInverseExpand` when computing the inverse A^{-1} of a nonsingular totally positive matrix A starting from the matrix $\mathcal{BD}(A)$.

Table 1
Relative errors in the computation of the Moore–Penrose inverse of the Hilbert matrix 12×8 in Example 5.1.

$\kappa_2(A)$	TNBD	MM	pinv	qrginv	QRATS2
$1.6e+09$	$1.4e-08$	$3.0e-16$	$1.5e-08$	$1.5e-08$	$1.6e-08$

Theorem 4.1. *Let A be a nonsingular totally positive matrix and $C = (c_{i,j})_{1\le i,j\le n}$ its exact inverse. Let $(\widehat{c}_{i,j})_{1\le i,j\le n}$ be the matrix representing the inverse of A computed from $\mathcal{BD}(A)$, the matrix representing the bidiagonal decomposition of A, by means of the algorithm* TNInverseExpand *in floating point arithmetic with machine precision ϵ. Then*

$$|\widehat{c}_{i,j} - c_{i,j}| \le \frac{3n\epsilon}{1-3n\epsilon}|c_{i,j}|,\ D\,D\,Di, j = 1,\dots,n.$$

Proof. Looking at the loops of the algorithm and the arithmetic operations involved in each loop we observe that the entries of the inverse matrix of A that require more arithmetic operations to be computed are the ones in the $n-2$ column. As the same type of calculations must be done for computing each $c_{i,n-2}$ (i=1,…,n), analyzing the relative errors accumulated when computing one of them is enough for proving this theorem.

Proceeding in this way and accumulating the relative errors in the style of Higham (see Chapter 3 of [34,29], and [27]) in the computation of $c_{i,n-2}$ $(i = 1,\dots,n)$ by means of the algorithm TNInverseExpand we obtain

$$|\widehat{c}_{i,n-2} - c_{i,n-2}| \le \frac{3n\epsilon}{1-3n\epsilon}|c_{i,n-2}|,\ D\,D\,Di = 1,\dots,n,$$

and therefore

$$|\widehat{c}_{i,j} - c_{i,j}| \le \frac{3n\epsilon}{1-3n\epsilon}|c_{i,j}|,\ D\,D\,Di, j = 1,\dots,n. \quad \square$$

5. Numerical experiments

Several numerical experiments showing the good performance of our algorithm for computing the Moore–Penrose inverse of a strictly totally positive matrix A are presented in this section. We compare the results obtained by using our method for computing the Moore–Penrose inverse $A^\dagger$ with the results obtained by other direct methods: the command pinv from MATLAB, and two algorithms which are good for general (i.e. unstructured) matrices, namely qrginv from [6] and QRATS2 from [7].

Let us observe that pinv is an SVD method for computing $A^\dagger$, while qrginv is based on the QR factorization of A and QRATS2 on the QR factorization of A^T.

We have also compared our results to the ones obtained by the functions ginv from [35] and geninv from [3]. As the results obtained by these two functions are always worst than the ones obtained by using our method (from now on we will denote it by MM), pinv, qrginv and QRATS2, they are not reported here.

All the computations are done in MATLAB and the relative error of the computed Moore–Penrose inverse $A^\dagger$ is obtained by means of:

$$err = \frac{\|A^\dagger - A_e^\dagger\|_2}{\|A_e^\dagger\|_2},$$

where $A_e^\dagger$ is the exact Moore–Penrose inverse computed in *Maple*.

Example 5.1. Let A be the Hilbert matrix of size 12×8. As Hilbert matrices can be seen as Cauchy matrices, the first step of our method, that is, the computation of $\mathcal{BD}(A)$, is developed in this case by using the algorithm for the accurate computation of the bidiagonal decomposition of totally positive Cauchy–Vandermonde matrices presented in [22], an algorithm that works for the particular case of Cauchy matrices and uses as input only the corresponding nodes and poles.

In order to show the importance of computing the matrix $\mathcal{BD}(A)$ with high relative accuracy for obtaining an accurate $A^\dagger$, we also compare in this first experiment the results of our approach with the results obtained by the approach that we will denote by TNBD and whose stages are:

1. The computation of the matrix $\mathcal{BD}(A)$ by means of classical Neville elimination without taking into account the structure of the Hilbert matrix. It is carried out by using the command TNBD in the package TNTool of P. Koev [30], the only function in the package that does not preserve high relative accuracy.
2. Steps 2, 3 and 4 of our algorithm (MM).

The relative errors obtained in the computation of $A^\dagger$ by the methods we have enumerated above are presented in Table 1.

Table 2
Relative errors in the computation of the Moore–Penrose inverse of the Pascal matrix 15×10 in Example 5.2.

$\kappa_2(A)$	MM	pinv	qrginv	QRATS2
$1.3e+09$	$3.7e-16$	$2.3e-09$	$2.0e-10$	$3.1e-09$

Examining Table 1 we see that, although the condition number of the Hilbert matrix is high, our approach computes its Moore–Penrose inverse very accurately (with a relative error of $3.0e-16$). MM obtains the most accurate $A^\dagger$ among all the approaches we have tested.

The difference between the relative error obtained by our approach MM ($3.0e-16$) and by the approach TNBD ($1.4e-08$) shows that computing the matrix $\mathcal{BD}(A)$ with high relative accuracy is essential for obtaining an accurate $A^\dagger$. Classical Neville elimination does not guarantee the computation of $\mathcal{BD}(A)$ with high relative accuracy, and therefore it is important to develop new algorithms that, taking into account the structure of A, compute its $\mathcal{BD}(A)$ with high relative accuracy. Once $\mathcal{BD}(A)$ is computed with high relative accuracy we can do numerical linear algebra accurately for A [17].

This observations do not contradict the results on componentwise backward error analysis of Neville elimination presented in [36]. As the author (following [17]) recalls in Section 3 of that paper, given $\mathcal{BD}(A)$ many matrix computations with A can be performed accurately. But of course this is true only if the entries of $\mathcal{BD}(A)$ are computed with high relative accuracy, which is not the case for Neville elimination.

This is a new example of the situation described in Chapter 16 of [37]. In Theorem 10 of [36] (see also Example 2 related to Hilbert matrices) the product of not accurate bidiagonal factors of A gives an accurate A. As commented in [37] remembering Wilkinson, the errors in the bidiagonal factors must be "diabolically correlated".

The result on backward stability presented in [36] is important when working with general totally positive matrices, but backward stability does not imply high relative accuracy in the computation of the entries of $\mathcal{BD}(A)$. This can only be obtained for structured matrices by means of algorithms which take that structure into account.

Example 5.2. Let us consider the Pascal matrix of order 15, and let A be the submatrix of size 15×10 obtained by removing the last five columns. In this case $\mathcal{BD}(A)$ is the matrix with all its entries equal to 1 [38], and therefore the first step of MM has not to be computed. The relative errors obtained in the computation of the Moore–Penrose inverse $A^\dagger$ of A by means of MM, pinv, qrginv and QRATS2 are presented in Table 2.

The results in Table 2 are analogous to the ones obtained in Example 5.1.

Example 5.3. Let us consider the nodes

$$\frac{1}{16} < \frac{1}{8} < \frac{3}{16} < \frac{1}{4} < \frac{5}{16} < \frac{3}{8} < \frac{7}{16} < \frac{1}{2} < \frac{9}{16} < \frac{5}{8} < \frac{11}{16} < \frac{3}{4} < \frac{13}{16} < \frac{7}{8} < \frac{15}{16},$$

and let V, BV and SBV be respectively the 15×10 strictly totally positive Vandermonde, Bernstein–Vandermonde and Said–Ball–Vandermonde matrices for those nodes.

In this example we compare the results obtained when using MM, pinv, qrginv and QRATS2 for computing the Moore–Penrose inverse for these three matrices.

The accurate computation of the matrix $\mathcal{BD}(A)$ in the first step of MM is done by using:

- In the Vandermonde case, the algorithm that given a rectangular strictly totally positive Cauchy–Vandermonde matrix A with one multiple pole computes $\mathcal{BD}(A)$ with high relative accuracy [21]. This algorithm works for the particular case in which A is a Vandermonde matrix, and its implementation in MATLAB for the square case is called TNBDCV and can be found in the package TNTool of P. Koev [30]. TNBDCV only needs as input the corresponding interpolation nodes.
- In the Bernstein–Vandermonde case, the algorithm that given a rectangular strictly totally positive Bernstein-Vandermonde matrix A computes $\mathcal{BD}(A)$ with high relative accuracy [25]. Its implementation in MATLAB is called TNBDBVR and can be obtained from the package TNTool of P. Koev [30]. TNBDBVR only needs as input the corresponding interpolation nodes.
- In the Said–Ball–Vandermonde case, the algorithm for the accurate computation of the matrix $\mathcal{BD}(A)$ when A is a rectangular strictly totally positive Said–Ball–Vandermonde. Its pseudocode can be found in [27] and it only needs as input the corresponding interpolation nodes.

The relative errors obtained in the computations are included in Table 3.

Looking at Table 3, we observe that MM is the only algorithm that computes the Moore–Penrose inverse of V, BV and SBV accurately with relative errors of $5.9e-16$, $5.2e-16$ and $5.7e-16$, respectively. The other methods obtain better results (but much less accurate than MM) for BV and SBV, matrices that are better conditioned than V.

The following example is similar to Example 5.3, but the matrices involved have greater condition numbers than the ones in Example 5.3. The results of the experiment show how our approach is still accurate when the condition number of the given matrices increases, while the approaches that do not take into account the structure of the initial matrices obtain less accurate results than in the previous example.

Table 3
Relative errors in the computation of the Moore–Penrose inverse of the Vandermonde, Bernstein–Vandermonde and Said–Ball–Vandermonde matrices in Example 5.3.

Matrix	$\kappa_2(A)$	`MM`	`pinv`	`qrginv`	`QRATS2`
V	$1.5e+07$	$5.9e-16$	$2.6e-11$	$5.6e-11$	$2.3e-11$
BV	$2.0e+02$	$5.2e-16$	$3.7e-14$	$6.4e-14$	$5.5e-14$
SBV	$4.0e+03$	$5.7e-16$	$1.1e-13$	$1.9e-13$	$1.3e-13$

Table 4
Relative errors in the computation of the Moore–Penrose inverse of the Vandermonde, Bernstein–Vandermonde and Said–Ball–Vandermonde matrices in Example 5.4.

Matrix	$\kappa_2(A)$	`MM`	`pinv`	`qrginv`	`QRATS2`
V	$5.5e+13$	$2.8e-15$	$2.4e-04$	$3.5e-04$	$3.3e-05$
BV	$1.7e+08$	$3.4e-15$	$1.8e-09$	$4.1e-09$	$4.5e-10$
SBV	$2.1e+09$	$3.7e-15$	$2.3e-08$	$5.1e-08$	$2.3e-08$

Example 5.4. Let us consider the nodes

$$\frac{1}{40} < \frac{1}{35} < \frac{1}{30} < \frac{1}{25} < \frac{1}{20} < \frac{1}{15} < \frac{1}{10} < \frac{1}{8} < \frac{1}{6} < \frac{1}{4} < \frac{1}{2} < \frac{11}{21} < \frac{19}{36} <$$
$$\frac{15}{28} < \frac{13}{24} < \frac{17}{30} < \frac{13}{22} < \frac{3}{5} < \frac{5}{8} < \frac{7}{10} < \frac{9}{10},$$

and let *V*, *BV* and *SBV* be respectively the 21×16 strictly totally positive Vandermonde, Bernstein–Vandermonde and Said–Ball–Vandermonde matrices for those nodes.

The relative errors obtained in the computation of the Moore–Penrose inverses of the matrices *V*, *BV* and *SBV* by means of `MM`, `pinv`, `qrginv` and `QRATS2` are presented in Table 4.

Now we include an example in which the computation of the Moore–Penrose inverse is applied to the solution of a polynomial regression problem in which the monomial basis and the Bernstein basis are considered.

Example 5.5. Let us consider the same nodes $\{x_i\}_{1\le i\le 16}$ as in Example 5.4 and the data vector

$$f = (2, 0, -1, 3, 5, 1, -1, 0, 0, 4, -2, 3, 1, -4, -5, 0, 8, 0, -1, 1, -2)^T.$$

Our first aim is to compute the polynomial $P(x) = \sum_{i=0}^{15} c_i x^i$ such that

$$\sum_{i=1}^{21} |f_i - P(x_i)|^2 \tag{5.1}$$

is minimum, what is equivalent to solve in the least squares sense the overdetermined linear system $Vc = f$, where *V* is the Vandermonde matrix whose Moore–Penrose inverse has been computed in Example 5.4, and *c* the vector containing the coefficients of the polynomial *P* expressed in the monomial basis. The solution of this least squares problem is $c = V^\dagger f$, and we will compute it by using the same methods we have used in the previous examples. The comparison is included in Table 5.

We also consider the same regression problem when the monomial basis is replaced by the Bernstein basis

$$\mathcal{B}_n = \left\{ b_i^{(n)}(x) = \binom{n}{i}(1-x)^{n-i}x^i, \qquad i = 0, \dots, n \right\}.$$

So, now we are interested in computing the coefficients of $P(x) = \sum_{i=0}^{15} c_i \binom{n}{i}(1-x)^{n-i}x^i$ such that (5.1) is minimum. The overdetermined linear system corresponding to this regression problem is $Bc = f$, where *B* is the Bernstein–Vandermonde matrix whose Moore–Penrose inverse has been computed in Example 5.4, and *c* is the vector with the coefficients of *P* in the Bernstein basis. The solution of $Bc = f$ in the least squares sense is $c = B^\dagger f$, and we will compute it by means of the same procedures employed when the monomial basis is taken. The results are also presented in Table 5.

All the computations are done in MATLAB and the relative error of the computed coefficient vector *c* is obtained by means of:

$$err = \frac{\|c - c_e\|_2}{\|c_e\|_2},$$

where c_e is the exact coefficient vector computed in *Maple*.

The results in Table 5 show the better accuracy of our approach, as well as the convenience of using the Bernstein basis (with any method) instead of the monomial basis in this application of the Moore–Penrose inverse to polynomial least squares fitting.

Table 5
Relative errors in the computation of the regression polynomial in Example 5.5 when the monomial and the Bernstein basis are used.

Basis	$\kappa_2(A)$	MM	pinv	qrginv	QRATS2
Monomial	$5.5e+13$	$9.6e-15$	$2.7e-06$	$3.2e-04$	$2.2e-05$
Bernstein	$1.7e+08$	$4.8e-15$	$1.4e-09$	$1.4e-10$	$2.0e-10$

Table 6
Relative errors in the computation of the Moore–Penrose inverse of the Vandermonde and Bernstein–Vandermonde matrices in Example 5.6.

Matrix	$\kappa_2(A)$	MM	pinv	qrginv	QRATS2
V	$1.4e+33$	$2.5e-15$	$1.0e+00$	$1.0e+00$	$1.0e+00$
BV	$3.3e+14$	$3.0e-15$	$1.0e+00$	$1.0e+00$	$1.0e+00$

Finally, we present an example in which, although the considered structured strictly totally positive matrices are of moderate size, their condition numbers are so large that the approaches that do not take into account the structure of these matrices give no accurate answer at all, while our algorithm still gives accurate results.

Example 5.6. Let us consider the nodes $\left\{\frac{i}{51}\right\}_{i=1,\ldots,50}$ sorted in increasing ordering, and let V and BV be respectively the 50×41 strictly totally positive Vandermonde and Bernstein–Vandermonde matrices for those nodes.

The relative errors obtained in the computation of the Moore–Penrose inverses of the matrices V and BV by means of `MM`, `pinv`, `qrginv` and `QRATS2` are presented in Table 6. The spectral condition numbers of both matrices are also included.

Let us observe that the condition number of V computed by using the command `cond(V,2)` of MATLAB is $2.0e+18$ while the correct one is $1.4e+33$. This fact clearly shows that, although the involved matrices could be considered of moderate size, due to their high condition numbers it is difficult to work with them without using algorithm that take into account their structure.

6. Conclusions

In this paper we have presented an algorithm for computing the Moore–Penrose inverse of strictly totally positive matrices, and we have applied it to solve least squares problems with structured matrices (Vandermonde and Bernstein–Vandermonde) related to polynomial regression problems.

As the numerical examples show, although the condition numbers of these matrices are high, our algorithm gives accurate results. This is a consequence of the fact that our approach takes advantage of the total positivity and special structure of the matrices.

In particular, the MATLAB command `pinv` gives less accurate results since it is based on the singular value decomposition and, consequently, the difficulty of computing the small singular values of ill-conditioned matrices with standard algorithms implies that the singular value decomposition is not accurately computed.

Acknowledgments

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 and J. J. Martínez are members of the Research Group ASYNACS (Ref. CCEE2011/R34) of Universidad de Alcalá.

References

[1] A. Ben-Israel, The Moore of the Moore-Penrose inverse, Electron. J. Linear Algebra 9 (2002) 150–157, http://dx.doi.org/10.13001/1081-3810.1083.
[2] T.N.E. Greville, Some applications of the pseudoinverse of a matrix, SIAM Rev. 2 (1960) 15–22.
[3] P. Courrieu, Fast computation of Moore-Penrose inverse matrices, Neural Inf. Process. Lett. Rev. 8 (2005) 25–29.
[4] S. Chountasis, V.N. Katsikis, D. Pappas, Applications of the Moore-Penrose inverse in digital image restoration, Math. Probl. Eng. 2009 (2009) 1–12, http://dx.doi.org/10.1155/2009/170724.
[5] A. Hernández, M. Lattanzi, N. Thome, On a partial order defined by the weighted Moore-Penrose invese, Appl. Math. Comput. 219 (2013) 7310–7318, http://dx.doi.org/10.1016/j.amc.2013.02.010.
[6] V.N. Katsikis, D. Pappas, A. Petralias, An improved method for the computation of the Moore-Penrose inverse matrix, Appl. Math. Comput. 217 (2011) 9828–9834, http://dx.doi.org/10.1016/j.amc.2011.04.080.
[7] P.S. Stanimirović, D. Pappas, V.N. Katsikis, I.P. Stanimirović, Full-rank representations of outer inverses based on the QR decomposition, Appl. Math. Comput. 218 (2012) 10321–10333, http://dx.doi.org/10.1016/j.amc.2012.04.011.
[8] A. Smoktunowicz, I. Wróbel, Numerical aspects of computing the Moore-Penrose inverse of full column rank matrices, BIT 52 (2012) 503–524, http://dx.doi.org/10.1007/s10543-011-0362-0.
[9] P.S. Stanimirović, F. Soleymani, A class of numerical algorithms for computing outer inverses, J. Comput. Appl. Math. 263 (2014) 236–245, http://dx.doi.org/10.1016/j.cam.2013.12.033.
[10] M.D. Petković, M.S. Petković, Hyper-power methods for the computation of outer inverses, J. Comput. Appl. Math. 278 (2015) 110–118, http://dx.doi.org/10.1016/j.cam.2014.09.024.

[11] M.D. Petković, P.S. Stanimirović, Iterative method for computing the Moore-Penrose inverse based on Penrose equations, J. Comput. Appl. Math. 235 (2011) 1604–1613, http://dx.doi.org/10.1016/j.cam.2010.08.042.
[12] M.D. Petković, M.S. Petković, Two improvements of the iterative method for computing Moore-Penrose inverse based on Penrose equations, J. Comput. Appl. Math. 267 (2014) 61–71, http://dx.doi.org/10.1016/j.cam.2014.01.034.
[13] N. Castro-González, F.M. Dopico, J.M. Molera, Multiplicative perturbation theory of the Moore-Penrose inverse and the least squares problem, Linear Algebra Appl. 503 (2016) 1–25, http://dx.doi.org/10.1016/j.laa.2016.03.027.
[14] A. Pinkus, Totally Positive Matrices, in: Cambridge Tracts in. Math., vol. 181, Cambridge University Press, 2010.
[15] S. Fallat, C.R. Johnson, Totally Nonnegative Matrices, in: Princeton Ser. Appl. Math., Princeton University Press, 2011.
[16] J. Demmel, I. Dumitriu, O. Holtz, P. Koev, Accurate and efficient expression evaluation and linear algebra, Acta Numer. 17 (2008) 87–145, http://dx.doi.org/10.1017/S0962492906350015.
[17] P. Koev, Accurate computations with totally nonnegative matrices, SIAM J. Matrix Anal. Appl. 29 (2007) 731–751, http://dx.doi.org/10.1137/04061903X.
[18] M. Gasca, J.M. Peña, Total positivity and Neville elimination, Linear Algebra Appl. 165 (1992) 25–44.
[19] M. Gasca, J.M. Peña, A matricial description of Neville elimination with applications to total positivity, Linear Algebra Appl. 202 (1994) 33–45.
[20] M. Gasca, J.M. Peña, On factorizations of totally positive matrices, in: M. Gasca, C.A. Michelli (Eds.), Total Positivity and Its Applications, Kluwer Academic Publishers, Dordrecht, 1996, pp. 109–130.
[21] J.J. Martínez, J.M. Peña, Factorizations of Cauchy-Vandermonde matrices with one multiple pole, in: O. Pordavi (Ed.), Recent Research on Pure and Applied Algebra, Nova Science Publishers, Hauppauge, NY, 2003, pp. 85–95.
[22] A. Marco, J. Martínez, J.M. Peña, Accurate bidiagonal decomposition of totally positive Cauchy-Vandermonde matrices and applications, Linear Algebra Appl. 517 (2017) 63–84, http://dx.doi.org/10.1016/j.laa.2016.12.003.
[23] J. Demmel, P. Koev, The accurate and efficient solution of a totally positive generalized vandermonde linear system, SIAM J. Matrix Anal. Appl. 27 (2005) 142–152, http://dx.doi.org/10.1137/S0895479804440335.
[24] A. Marco, J. Martínez, A fast and accurate algorithm for solving Bernstein-Vandermonde linear systems, Linear Algebra Appl. 422 (2007) 616–628, http://dx.doi.org/10.1016/j.laa.2006.11.020.
[25] A. Marco, J. Martínez, Accurate computations with totally positive Bernstein-Vandermonde matrices, Electron. J. Linear Algebra 26 (2013) 357–380, http://dx.doi.org/10.13001/1081-3810.1658.
[26] J. Delgado, J.M. Peña, Accurate computations with collocation matrices of rational bases, Appl. Math. Comput. 219 (2013) 4354–4364, http://dx.doi.org/10.1016/j.amc.2012.10.019.
[27] A. Marco, J. Martínez, Bidiagonal decomposition of rectangular totally positive Said-Ball-Vandermonde matrices: Error analysis, perturbation theory and applications, Linear Algebra Appl. 495 (2016) 90–107, http://dx.doi.org/10.1016/j.laa.2016.01.037.
[28] J. Delgado, J.M. Peña, Accurate computations with Lupaş matrices, Appl. Math. Comput. 303 (2017) 171–177, http://dx.doi.org/10.1016/j.amc.2017.01.031.
[29] J. Demmel, P. Koev, Accurate SVDs of polynomial Vandermonde matrices involving orthonormal polynomials, Linear Algebra Appl. 417 (2006) 382–396, http://dx.doi.org/10.1016/j.laa.2005.09.014.
[30] P. Koev, Package TNTool. URL http://www.math.sjsu.edu/~koev/software/TNTool.html.
[31] A. Björck, Numerical Methods for Leasts Squares Problems, SIAM, Philadelphia, 1996.
[32] G. Strang, Linear Algebra and its Applications, fourth ed., Brooks/Cole/Cengage, 2006.
[33] A. Björck, Numerical Methods in Matrix Computations, in: Texts in Applied Mathematics, vol. 59, Springer, 2015.
[34] N.J. Higham, Accuracy and Stability of Numerical Algorithms, second ed., SIAM, Philadelphia, 2002.
[35] V.N. Katsikis, D. Pappas, Fast computing of the Moore-Penrose inverse matrix, Electron. J. Linear Algebra 17 (2008) 637–650, http://dx.doi.org/10.13001/1081-3810.1287.
[36] R. Huang, L. Zhu, Componentwise backward error analysis of Neville elimination, Linear Algebra Appl. 451 (2014) 33–48, http://dx.doi.org/10.1016/j.laa.2014.03.014.
[37] L.N. Trefethen, D.B. III, Numerical Linear Algebra, SIAM, Philadelphia, 1997.
[38] S. Fallat, Bidiagonal factorizations of totally nonnegative matrices, Amer. Math. Monthly 108 (2001) 697–712.