A particular block Vandermonde matrix

Malika Yaici[1,*], and *Kamel* Hariche[2]

[1]Laboratoire LTII, University of Bejaia, Algeria
[2]IGEE Institute, University of Boumerdes, Algeria

Abstract. The Vandermonde matrix is ubiquitous in mathematics and engineering. Both the Vandermonde matrix and its inverse are often encountered in control theory, in the derivation of numerical formulas, and in systems theory. In some cases block vandermonde matrices are used. Block Vandermonde matrices, considered in this paper, are constructed from a full set of solvents of a corresponding matrix polynomial. These solvents represent block poles and block zeros of a linear multivariable dynamical time-invariant system described in matrix fractions. Control techniques of such systems deal with the inverse or determinant of block vandermonde matrices. Methods to compute the inverse of a block vandermonde matrix have not been studied but the inversion of block matrices (or partitioned matrices) is very well studied. In this paper, properties of these matrices and iterative algorithms to compute the determinant and the inverse of a block Vandermonde matrix are given. A parallelization of these algorithms is also presented. The proposed algorithms are validated by a comparison based on algorithmic complexity.

1 Introduction

The Vandermonde matrix is very important and its uses include polynomial interpolation, coding theory, signal processing, etc. Literature on Vandermonde matrix goes back to 1965 and even before, where many papers deal with the study of its properties, its inverse and its determinant [1-5]. In [6], the author proved that every generic nxn matrix is a product of a Vandermonde matrix and its transpose, and in [7] a Vandermonde matrix is decomposed to obtain variants of the Lagrange interpolation polynomial of degree≤n that passes through the n +1points .

One of the first papers where the term block vandermonde matrix (BVM) is used, to my knowledge, is [8] but it is in [9] and [10] that the concept is fully studied; the block Vandermonde matrix is defined and its properties are explored. A method, based on the Gaussian elimination, to compute the determinant is also proposed. In [11], the author gives a method to determine the biggest integer $n=v(q,t)$ for which there exist $t\times t$ matrices $\{A_1 \ldots A_n\}$with the highest power q such that the BVM $V = (A_j^{i-1}\ i, j \le n)$ is invertible. In [12], linear diffusion layers achieving maximal branch numbers called MDS (maximal distance separable) matrices are constructed from block Vandermonde matrices and their transposes. Under some conditions these MDS matrices are involutory its inverse is itself) which is of great value in cryptography.

Methods to compute the inverse of a block vandermonde matrix have not been studied but the inversion of block matrices (or partitioned matrices) is very well studied! The method to compute the inverse of a 2×2 block matrix is known, under the conditions that at least one of the two diagonal matrix entries must be non-singular. In [13], this condition is overcome by using three new types of symbolic block matrix inversion.

In [14], the properties of block matrices with block banded inverses are investigated to derive efficient matrix inversion algorithms for such matrices. In particular, a recursive algorithm to invert a full matrix whose inverse is structured as a block tridiagonal matrix and a recursive algorithm to compute the inverse of a structured block tridiagonal matrix are proposed.

Block Vandermonde matrices constructed using matrix polynomials solvents are very useful in control engineering, for example in control of multi-variable dynamic systems described in matrix fractions (see [15]). It is in these particular BVM that we are interested.

Parallelization may be a solution to problems where large size matrices, as BVM, are used. Large scale matrix inversion has been used in many domains and block-based Gauss-Jordan (G-J) algorithm as a classical method of large matrix inversion has become the focus of many researchers. But the large parallel granularity in existing algorithms restricts the performance of parallel block-based G-J algorithm, especially in the cluster environment consisting of PCs or workstations. The author of [16] presents a fine-grained parallel G-J algorithm to settle the problem presented above.

In this paper a new algorithm to compute the inverse and the determinant of a block Vandermonde matrix constructed from solvents are given. An implementation using Matlab has been undergone, in order to obtain the speed-up of the proposed algorithms compared to Matlab built-in functions.

* Corresponding author:yaici_m@hotmail.com

After this introduction, some needed mathematical preliminaries are presented in section 2, and then the main results come in section 3. A parallelization of the proposed algorithms is given in section 4. A conclusion finishes the paper.

2 Mathematical preliminaries

For a set of n $m \times m$ matrices $\{A_1, A_2, \ldots, A_n\}$, the corresponding block Vandermonde matrix (BVM) of order t is defined as follows:

$$V = \begin{pmatrix} I & I & \cdots & I \\ A_1 & A_2 & \cdots & A_n \\ \vdots & \vdots & \vdots & \vdots \\ A_1^{t-1} & A_2^{t-1} & \cdots & A_n^{t-1} \end{pmatrix} \quad (1)$$

The block Vandermonde matrices, we will be dealing with, are constructed from solvents of matrix polynomials. In this section a recall on matrix polynomials, solvents, and their properties, will be given.

2.1 Matrix polynomials

Definition 1: An mth order, rth degree matrix polynomial (also called λ-matrix) is given by [15]:

$$A(t) = A_r t^r + A_{r-1} t^{r-1} + \cdots + A_1 t + A_0 \quad (2)$$

where A_i are $m \times m$ real matrices and t a complex number.

Definition 2: Let X be an $m \times m$ complex matrix.
A right matrix polynomial is defined by:

$$A_R(t) = A_r X^r + A_{r-1} X^{r-1} + \cdots + A_1 X + A_0 \quad (3)$$

And a left matrix polynomial is defined by:

$$A_L(t) = X^r A_r + X^{r-1} A_{r-1} + \cdots + X A_1 + A_0 \quad (4)$$

Definition 3: The complex number λ_i is called a latent value of $A(t)$ if it is a solution of the scalar polynomial equation $det(A(t)) = 0$. The non-trivial vector v_i, solution of the equation $A(\lambda_i)v_i = 0$, is called a primary right latent vector associated to the latent value λ_i. Similarly, the non trivial row vector w, solution of the equation $wA(\lambda_i) = 0$ is called a primary left latent vector associated with λ_i [15].

2.2 Solvents

Definition 4: A right solvent (or a block root) R of a polynomial matrix $A(t)$ is defined by:

$$A(R) = A_r R^r + A_{r-1} R^{r-1} + \cdots + A_1 R + A_0 = 0_m \quad (5)$$

and the left solvent L of a polynomial matrix $A(t)$ is defined by:

$$A(L) = L^r A_r + L^{r-1} A_{r-1} + \cdots + L A_1 + A_0 = 0_m \quad (6)$$

Theorem 1: If the latent roots of $A(t)$ are distinct, then $A(X)$ has a complete set of solvents.
Proof 1: see [9].

A solvent is automatically non-singular. The determinant is non-null because its latent values (eigenvalues) are distinct; the latent vectors (eigenvectors) must be linearly independent.

2.3 Block Vandermonde matrices

As for an eigenvalue system, a block Vandermonde matrix can be defined for solvents with particular properties [15].

Let a set of r right solvents R_i ($m \times m$ matrices) of a corresponding matrix polynomial $A(t)$. A row block Vandermonde matrix of order r is a $rm \times rm$ matrix defined as:

$$V = \begin{pmatrix} I_m & I_m & \cdots & I_m \\ R_1 & R_2 & \cdots & R_r \\ \vdots & \vdots & \vdots & \vdots \\ R_1^{r-1} & R_2^{r-1} & \cdots & R_r^{r-1} \end{pmatrix} \quad (7)$$

And given a set of r left solvents L_i ($m \times m$ matrices) of a polynomial matrix $A(t)$ a column block Vandermonde matrix of order r is a defined as:

$$V = \begin{pmatrix} I_m & R_1 & \cdots & R_1^{r-1} \\ I_m & R_2 & \cdots & R_2^{r-1} \\ \vdots & \vdots & \cdots & \vdots \\ I_m & R_r & \cdots & R_r^{r-1} \end{pmatrix} \quad (8)$$

Theorem 2: If $A(t)$ has distinct latent roots, then there exists a complete set of right solvents of $A(X)$, $R_1, \ldots, R_m$, and for any such set of solvents, The BVM V is nonsingular.
Proof 2: see [9]

Remark 1: In [17] the general right (left) block Vandermonde matrix constructed by solvents, where a right (left) solvent R_i (L_i) with multiplicity m_i exists, is given.

2.4 Non-singularity

Definition 5: If we let $\sigma[A(t)]$ denote the set of all latent roots of $A(t)$ and $\sigma[R_i]$ the set of eigenvalues of the right solvent Ri, then a complete set of right solvents is obtained if we can find r right solvents such that [10]:

$$\begin{cases} \bigcup_{i=1}^{r} \sigma[R_i] = \sigma[A(t)] \\ \sigma[R_i] \cap \sigma[R_j] = \varnothing \end{cases} \quad (9)$$

and the block Vandermonde matrix thus constructed is nonsingular.

Just as for the right solvents, the existence of a left block root depends on the existence of a set of *m* linearly independent left latent vectors. A complete set of left block roots is obtained if we can find *r* left block roots where each block root involves a distinct set of *m* latent roots of $A(t)$. This in turn requires that for each such a distinct set, we can find a corresponding set of linearly left latent vectors.

Remark 2: A complete set of right or left solvents will then describe completely the latent (eigen) structure of $A(t)$.

3 Main results

The following results are mainly given on a row-BVM constructed from right solvents. The same procedures can be applied to a column-BVM constructed from left solvents.

3.1. Iterative construction of BVM

Let $V_1 = I_m$, where I_m is the $m \times m$ identity matrix, then BVM of order 2 constructed from two solvents is as follows:

$$V_2 = \begin{pmatrix} V_1 & I_m \\ R_1 & R_2 \end{pmatrix} \tag{10}$$

If we define the following matrices: $B_1=I_m$, $C_1=R_1$ and $D_1=R_2$ Then a BVM of order three (3 solvents) is as follows:

$$V_2 = \begin{pmatrix} V_2 & B_2 \\ C_2 & D_2 \end{pmatrix} \tag{11}$$

where $B_2 = \begin{pmatrix} I_m \\ R_3 \end{pmatrix}$, $C_2 = (R_1^2 \quad R_2^2)$ and $D_2 = R_3^2$.

The following theorem is a deduction from previous results:

Theorem 3: A BVM of order *r*, constructed from *r* solvents, is as follows:

$$V_r = \begin{pmatrix} V_{r-1} & B_{r-1} \\ C_{r-1} & D_{r-1} \end{pmatrix} \tag{12}$$

where $B_{r-1} = \begin{pmatrix} I_m \\ R_r \\ \vdots \\ R_r^{r-2} \end{pmatrix}$,
$C_r = (R_1^{r-1} \quad \cdots \quad R_r^{r-1})$
and $D_{r-1} = R_r^{r-1}$.

Proof 3: It is straight forward from the block partitioning of the BVM V_r as follows:

$$V = \left(\begin{array}{cccc|c} I_m & I_m & \cdots & I_m & I_m \\ R_1 & R_2 & \cdots & R_{r-1} & R_r \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_1^{r-2} & R_2^{r-2} & \cdots & R_{r-1}^{r-2} & R_r^{r-2} \\ -- & -- & -- & -- & -- \\ R_1^{r-1} & R_2^{r-1} & \cdots & R_{r-1}^{r-1} & R_r^{r-1} \end{array}\right) \tag{13}$$

3.2 Inverse of a BVM

From [18], the inverse of a block partitioned matrix is given as follows:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + E * S_A^{-1} * F & -E * S_A^{-1} \\ -S_A^{-1} * F & S_A^{-1} \end{pmatrix} \tag{14}$$

if A^{-1} exists and S_A , the Shur Complement of matrix *A*, is nonsingular.

$$E = A^{-1} * B,\ F = C * A^{-1},\ S_A = D - C * A^{-1} * B.$$

A similar inverse formula exists if D^{-1} exists and S_D, the Shur complement of matrix *D*, is nonsingular.

Let us compute the inverse of a BVM of order *r* as given in equation 7 by using equation 14. In our case both diagonal entries are non-singular.

$$V_r^{-1} = \begin{pmatrix} V_{r-1}^{-1} + E * S_{r-1}^{-1} * F & -E * S_{r-1}^{-1} \\ -S_{r-1}^{-1} * F & S_{r-1}^{-1} \end{pmatrix} \tag{15}$$

where S_{r-1} is the Shur complement of matrix V_{r-1} (computed at iteration r-1)

$$S_{r-1} = D_{r-1} - C_{r-1} * V_{r-1}^{-1} * B_{r-1} \tag{16}$$

$$E = V_{r-1}^{-1} * B_{r-1},\ F = C_{r-1} * V_{r-1}^{-1}$$

and D_{r-1}, C_{r-1}, B_{r-1} are as given in equation 12.

The same procedure will be used to determine the inverse of the BVM V_{r-1}. So the algorithm is an iterative procedure.

Algorithm: Let a complete set of solvents $\{R_1, ..., R_r\}$ and the corresponding BVM V_r as given in equation7. From the matrix V_r, all sub-matrices (B_i, C_i, D_i and S_i) will be first constructed, and then the inverse is computed. The algorithm uses a function which computes the inverse of the Shur Complement.

```
Step1: Let INV = Im
Step2:
for i = 2*m to r*m with step=m
Bi-1 = Vr(1:i-2, i-1:i);
Ci-1 = Vr(i-1:I, 1:i-2);
```

D_{i-1} = V_r(i-1:I, i-1:i);
E_{i-1} = INV*B_{i-1};
F_{i-1} = C_{i-1} * INV;
S_{i-1} =D_{i-1}-C_{i-1}*INV*B_{i-1};

$$\text{INV} = \begin{pmatrix} INV + E * S_{i-1}^{-1} * F & -E * S_{i-1}^{-1} \\ -S_{i-1}^{-1} * F & S_{i-1}^{-1} \end{pmatrix}$$

endfor

Algorithmic Complexity: The iterative algorithm requires *r-2* iterations. The procedure consists of a set of affectations and the computation of the inverse of S_r. The size of S_r is $m \times m$ and Matlab uses Gaussian elimination method rather than an inverse algorithm. The Gaussian elimination has a complexity of $O(m^3)$.

The overall complexity of our algorithm is: $O((r\text{-}2) * m^3)$.

3.3 Determinant of a BVM

From [18], the determinant of a block partitioned matrix is as follows if A^{-1} exists:

$$\det\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det A * \det(D - C * A^{-1} * B) \quad (17)$$

A similar determinant formula exists if D^{-1} exists.

Using this equation and the block decompositions given in the previous sections we deduced the determinant of a BVM of order *r* as given in equation 7 by using equation 17.

$$\det V_r = \det V_{r-1} * \det S_{r-1} \quad (18)$$

where S_{r-1} is the Shur complement of matrix V_{r-1} (defined in equation 16).

Remark 2: The determinant is, in general, needed with the inverse. So the inverse of V_{r-1} is computed using the previous algorithm.

Algorithm: Let a complete set of solvents $\{R_1 \ldots R_r\}$ and the corresponding BVM V_r as given in equation 7. From the matrix V_r all sub-matrices (V_i, B_i, C_i, D_i and S_i) can be constructed and the determinant computed. The algorithm uses a function which computes the determinant of the Shur Complement.

Step1: Let Det = 1
Step2:
for i = 2*m to r*m with step=m
V_{i-1} = V_r(1: i-2, 1:i-2) ;
B_{i-1} = V_r(1: i-2, i-1:i);
C_{i-1} = V_r(i-1: I, 1: i-2);
Di-1 = V_r(i-1:I, i-1:i);
S_{i-1} = D_{i-1} - C_{i-1} * V_{i-1}^{-1} * B_{i-1};
Det = Det * determinant(S_{i-1});
endfor

Algorithmic Complexity: As for the inverse algorithm, the determinant iterative algorithm needs *r-2* iterations. The procedure consists of a set of affectations, the computation of the inverse of a BVM and of the determinant of *S*. The size of $S = m \times m$ and Matlab uses the triangular factors of Gaussian elimination method to compute the determinant and the inverse of a square matrix. The Gaussian elimination has a complexity of $O(m3)$.So the complexity depends also on the size of S_r.

The overall complexity of our algorithm is: $O((r\text{-}2) * m^3)$.

4 Parallelization

For both Determinant and Inverse, a parallelization is possible because of the decomposition step in the proposed algorithm. Even though the iterative approach is difficult to optimally parallelize! The above decomposition is useful only if a parallel execution is possible, otherwise the benefits are negligible.

There exist two kinds of parallelization of matrix calculus: data or tasks (calculus) decomposition. Because data decomposition is already performed, so task decomposition is proposed.

4.1 Parallel inverse of BVM

From the data decomposition, a master-slave task decomposition was performed on the sequential algorithm. The master task will execute the data scattering and gathering, and sequential instructions and at least three (3) slave tasks will execute the parallel blocks.

Algorithm:
Step1: Let INV = I_m
Step2: for i = 2*m to r*m with step=m
Parallel block
B_{i-1} = V_r(1:i-2; i-1:i);
C_{i-1} = V_r(i-1:i; 1:i-2);
$D_{i¡1}$ = V_r(i-1:i; i-1:i);
End
Parallel block
E_{i-1} = INV*B_{i-1} ;
F_{i-1} = C_{i-1}*INV ;
S_{i-1} = D_{i-1} -C_{i-1}*INV*B_{i-1};
end
iS = S_{i-1}^{-1};
Parallel block

$$\text{INV} = \begin{pmatrix} INV + E * iS * F & -E * iS \\ -iS * F & iS \end{pmatrix}$$

end
end

4.2 Parallel determinant of BVM

As for the precedent algorithm, a master-slave task decomposition has been performed, and the master task will execute the data scattering and gathering and

sequential parts, and at least four (4) slave tasks will execute the parallel blocks.

Algorithm:

```
Step1: Let Det=1
Step2: for i = 2*m to r*m with step=m
Parallel block
Vi-1 = Vr(1:i-2; 1:i-2);
Bi-1 = Vr(1:i-2; i-1:i);
Ci-1 = Vr(i-1:i; 1:i-2);
Di-1 = Vr(i-1:i; i-1:i);
End
Si-1 = D_{i-1} - C_{i-1} * V_{i-1}^{-1} * B_{i-1} ;
Det = Det*determinant(Si-1);
end
```

4.3 Algoithmic complexity

The overall time complexity of the two algorithms is the same as before: $O((r\text{-}2)*m^3)$. But the detailed complexity is slightly better.

An implementation using Matlab has been done. Matlab (classical) offers parallel execution of a set of instructions (parallel block) using `parfor`. Matlab uses the number of cores available on the used computer using `matlabpool`.

4 Conclusions

In this paper new results on the computation of the inverse and the determinant of a particular block Vandermonde matrix are given. Efficient algorithms are proposed with their algorithmic complexities.

We used the ”tic/toc” functions of Matlab to determine the execution time of our algorithms to be compared to the execution time of Matlab functions, and the proposed inverse algorithm is found 10 times quicker, and the determinant algorithm is 14 times quicker. The parallel execution time was greater than the sequential execution time, because of the large amount of data flowing between the cores at each iteration.

These new computation techniques are very useful in control theory, where systems are described in matrix fractions description and their properties are deduced from solvents. In this case block Vandermonde matrices constructed from solvents, their inverse and determinants are needed. The future work is within this axis and it consists in proposing a parallel algorithm to solve the Compensator (Diophantine) equation where block Vandermonde matrices constructed using matrix polynomial solvents are involved.

References

1. A. Klinger, *The Vandermonde matrix*. The American Mathematical Monthly. **74** (5), pp. 571-574 (1967)
2. V. Pless, *Introduction to the Theory of Error-Correcting Codes*. John Wiley, New York (1982)
3. R. E. Blahut, *Theory and Practice of Error Control Codes*. Addison Wesley, Reading, Mass., USA (1983)
4. G. H. Golub, C. F. Van Loan, *Matrix Computation*, Johns Hopkins Univ. Press, Baltimore, pp. 119-124 (1983)
5. J. J. Rushanan, *On the Vandermonde Matrix*. The American Mathematical Monthly, Published by: Mathematical Association of America. **96** (10), pp. 921-924 (1989)
6. K. Ye, *New classes of matrix decompositions*, Lin. Algeb. and its App., **514**, pp. 47-81, (2017)
7. I.-P.Kim, A. R. Kräuter, *Decompositions of a matrix by means of its dual matrices with applications*, Lin. Algeb. and its App., **537**, pp. 100–117, (2018)
8. J. E. Dennis, J. F. Traub, R. P. Weber, *On the matrix polynomial, lambda-matrix and block eigenvalue problems*. Computer Science Department Tech. rep., Carnegie-Mellon Univ., Pittsburgh, Pa., USA (1971)
9. J. E. Dennis, J. F. Traub, R. P. Weber, *The algebraic theory of matrix polynomials*. SIAM J. Numer. Anal. **13** (6), pp. 831-845 (1976)
10. J. E. Dennis, J. F. Traub, R. P. Weber, *Algorithms for solvents of matrix polynomials*. SIAM J. Numer. Anal. **15** (3), pp. 523-533 (1978)
11. D. R. Richman, *A result about block Vandermonde Matrices*. Lin. and Multilin. Alg. **21**, pp. 181-189 (1987)
12. Q. Li, B. Wu, and Z. Liu, *Direct Constructions of (Involutory) MDS Matrices from Block Vandermonde and Cauchy-like Matrices*, International Workshop on the Arithmetic of Finite Fields (WAIFI) 2018, Bergen, Norway. June 14-16, (2018)
13. Y. Choi, J. Cheong, *New Expressions of 2x2 Block Matrix Inversion and Their Application*. IEEE Trans. Auto. Cont. **54** (11), 2648-2653 (2009)
14. A. Asif, J. M. F. Moura, *Inversion of block matrices with block banded inverses: application to Kalman-Bucy filtering*. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul, Turkey. **1**, pp. 608-611, (2000)
15. M. Yaici, K. Hariche, *On eigenstructure assignment using block poles placement*. Euro. J. Cont. **20**, pp. 217-226 (2014)
16. L. Shang, Z. Wang, S. G. Petiton, F. Xu, *Solution of Large Scale Matrix Inversion on Cluster and Grid*. 7th Int. Conf. on Grid and Cooperative Computing, Shenzhen, China, pp. 33-40 (2008)
17. K. Hariche, E. D. Denman, *Interpolation theory and Lambda matrices*. J. Math. Anal. Appl. **143**, pp. 530-547 (1989)
18. T. Kailath, *Linear Systems*. Prentice Hall, New Jersey (1980)