Block Vandermonde Matrices of Matrix Polynomial's Solvents

Malika Yaici
Laboratoire LTII
University of Bejaia
Bejaia, Algeria
yaici_m@hotmail.com

Kamel Hariche
IGEE Institute
University of Boumerdes
Boumerdes, Algeria
khariche@yahoo.com

***Abstract*—Block Vandermonde matrices, considered in this paper, are constructed from a full set of solvents of a corresponding matrix polynomial. These block Vandermonde matrices are encountered especially in control engineering. Some 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 using algorithmic complexity.**

***Index Terms*—Block Vandermonde matrix, Solvents, Matrix polynomial, Matrix inverse, Determinant, parallelization**

I. Introduction

The Vandermonde matrix is ubiquitous in mathematics and engineering. Its uses include polynomial interpolation, coding theory, and signal processing, where the matrix for a discrete Fourier transform is a Vandermonde matrix.

Literature on Vandermonde matrix goes back to 1967, and even before, where many papers deal with the study of its properties, its inverse and its determinant [1]–[5]

Vandermonde matrix may be encountered in many domains as in computer science for the design of cross layer protocols with recovering from errors and packet loss impairments [6], and in [7] the quasi-cyclic (QC) protograph low-density parity-check (LDPC) codes are based on Vandermonde matrices.

The importance of the Vandermonde matrix in control theory has been emphasized in [8]–[10]. Generally, the inverse of the usual Vandermonde matrix as well as the inverse of the generalized Vandermonde matrix is based on using interpolation polynomials.

In [11], an explicit algorithm for the inverse of a Vandermonde matrix was given, based on the Lagrange interpolation and the generalization of the Yang-Hui triangle theory.

The authors of [12] give a new representation for a simple Vandermonde matrix (where all eigenvalues are simple) and a confluent (generalized) Vandermonde matrix and then compute its inverse easily to be used afterwards into resolving Stein equations.

The inversion of the Vandermonde matrix has received much attention for its role in the solution of some problems of numerical analysis and control theory. The work presented in [13] deals with the problem of getting an explicit formula for the generic element of the inverse to result in two algorithms in $O(n^2)$ and $O(n^3)$.

In [14], the numerical properties of the well-known fast Parker-Traub and Bjorck-Pereyra algorithms, which both use the special structure of a Vandermonde matrix to rapidly compute the entries of its inverse, are compared. The results of numerical experiments suggest that the Parker-Traub algorithm allows one not only fast $O(n^2)$ inversion of a Vandermonde matrix, but it also gives more accuracy.

One of the first papers where the term block vandermonde matrix is used, to my knowledge, is [15] but it is in [16] and [17] that the concept is fully studied; the Block Vandermonde matrix (BVM) is defined and its properties are explored. A method, based on the Gaussian elimination, to compute the determinant is also proposed.

In [18], the author gives a method to determine the biggest integer $n = v(q,t)$ for which there exist $t \times t$ matrices $\{A_1 ... A_n\}$ with the highest power q such that the BVM $V = [A_j^{i-1} j \leq n;\ i \leq n]$ is invertible.

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 [19], this condition is overcome by using three new types of symbolic block matrix inversion.

In [20], the properties of block matrices with block banded inverses are investigated to derive efficient matrix inversion algorithms for such matrices. In particular, the following is derived: 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.

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 [21] presents a fine-grained parallel G-J algorithm to settle the problem presented above.

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 [22]).

In this paper new algorithms to compute the inverse and the determinant of such block Vandermonde matrices and their parallelization are given with an implementation using Matlab.

After this introduction, Section 2 deals with definitions, and methods to compute the inverse and determinant of the simple Vandermonde matrix. The block Vandermonde matrix is detailed in section 3, with a recall on matrix polynomials and solvents. The main results are in Section 4, which consist of an iterative construction of the BVM, then proposed algorithms to compute the inverse and the determinant of the matrix. Section 5 is a proposition of a parallel implementation of the two algorithms. Finally a conclusion finishes the paper.

II. Simple Vandermonde matrices

In linear algebra, a Vandermonde matrix, named after Alexandre-Théophile Vandermonde [23], is a matrix with the terms of a geometric progression in each column, i.e., an $m \times n$ matrix

$$V = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_m \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_m^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & \cdots & x_m^{n-1} \end{pmatrix} \quad (1)$$

So for a set of n distinct values $\{x_i i = 1..n\}$ the corresponding Vandermonde square matrix are of two kinds:

Row-Vandermonde matrix of order n:

$$V_r = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} \quad (2)$$

Column-Vandermonde matrix of order n Transpose of the previous matrix:

$$V_c = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} \quad (3)$$

A. Determinant

The determinant of a square Vandermonde matrix can be expressed as:

$$detV(x_1, ..., x_n) = \prod_{1 \le i \le j \le n} (x_i - x_j) \quad (4)$$

This is called the Vandermonde determinant or Vandermonde polynomial. If all the x_i's are distinct, then it is nonzero.

B. Inverse

The inverse of the Vandermonde matrix is, generally, given in the form of the product $U^{-1}L^{-1}$ of two triangular matrices by the display of generating formulas from which the elements of U^{-l} and L^{-1} may be directly computed.

The Vandermonde matrix V has the determinant nonsingular if all values of x_i are distinct. It can, therefore, be factored into a lower triangular matrix L and an upper triangular matrix U where $V = LU$. The factorization is unique if no row or column interchanges are made and if it is specified that the diagonal elements of U are unity.

Another definition of the inverse of V is a matrix W satisfying $WV = I$. If w_i is the ith row of W, thus

$$w_i(x) = \frac{\prod_{j \ne i} (x - x_j)}{\prod_{j \ne i} (x_i - x_j)} \quad \text{for } i = 0, 1, ..., n-1 \quad (5)$$

Where W is normalized to get $w_i(a_i) = 1$. Hence the inverse of V is the matrix W whose rows are the vectors w_i generated by the $w_i(x)$ [5].

III. Block Vandermonde matrix

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

$$V = \begin{pmatrix} I & I & I & \cdots & I \\ A_1 & A_2 & A_3 & \cdots & A_n \\ A_1^2 & A_2^2 & A_3^2 & \cdots & A_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_1^{t-1} & A_2^{t-1} & A_3^{t-1} & \cdots & A_n^{t-1} \end{pmatrix} \quad (6)$$

By analogy to the simple vandermonde matrix, we define the row-BVM and column-BVM as follows.

So for a set of n $m \times m$ matrices $\{A_i i = 1..n\}$ the corresponding square BVM are of two kinds:

Row-BVM of order n:

$$V_r = \begin{pmatrix} I & I & I & \cdots & I \\ A_1 & A_2 & A_3 & \cdots & A_n \\ A_1^2 & A_2^2 & A_3^2 & \cdots & A_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_1^{n-1} & A_2^{n-1} & A_3^{n-1} & \cdots & A_n^{n-1} \end{pmatrix} \quad (7)$$

Column-BVM of order n Transpose of the previous matrix:

$$V_c = \begin{pmatrix} I & A_1 & A_1^2 & \cdots & A_1^{n-1} \\ I & A_2 & A_2^2 & \cdots & A_2^{n-1} \\ I & A_3 & A_3^2 & \cdots & A_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & A_n & A_n^2 & \cdots & A_n^{n-1} \end{pmatrix} \quad (8)$$

Remark 1: The block Vandermonde matrices, we will be dealing with, are constructed from solvents of matrix polynomials.

A. Matrix Polynomials

In this section a recall on matrix polynomials and solvents, and their properties, will be given.

Definition 1: The following $m \times m$ matrix:

$$A(t) = \begin{pmatrix} a_{11}(t) & a_{12}(t) & \cdots & a_{1m}(t) \\ a_{21}(t) & a_{22}(t) & \cdots & a_{2m}(t) \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}(t) & a_{m2}(t) & \cdots & a_{mm}(t) \end{pmatrix} \quad (9)$$

is called a polynomial matrix, of order m, where $a_{ij}(t)$ are scalar polynomials (of degree$\leq r$) over the field of complex numbers.

From a polynomial matrix we can construct a matrix polynomial and vice-versa.

Definition 2: An mth order, rth degree matrix polynomial (also called λ-matrix) is given by:

$$A(t) = A_r t^r + A_{r-1} t^{r-1} + ... + A_1 t + A_0 \quad (10)$$

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

Definition 3: Let X be a $m \times m$ complex matrix.

A right matrix polynomial is defined by:

$$A_R(X) = A_r X^r + A_{r-1} X^{r-1} + ... + A_1 X + A_0 \quad (11)$$

And a left matrix polynomial is defined by:

$$A_L(X) = X^r A_r + X^{r-1} A_{r-1} + ... + X A_1 + A_0 \quad (12)$$

Definition 4: 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 [24].

B. Solvents

Definition 5: A right solvent (or a block root) R of a polynomial matrix $A(t)$ is defined by:

$$A(R) = A_r R^r + A_{r-1} R^{r-1} + ... + A_1 R + A_0 = 0_m \quad (13)$$

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

$$A(L) = L^r A_r + L^{r-1} A_{r-1} + ... + L A_1 + A_0 = 0_m \quad (14)$$

In the following some important facts on solvents:

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

Proof 1: see [16].

Remark 2: A solvent is automatically non-singular. The determinant is non-null because its eigenvalues are distinct; the eigenvectors must be linearly independent.

Remark 3: If $A(t)$ has mr distinct latent roots and the set of right (left) latent vectors verify the condition that every m of them are linearly independent (Haar Condition) then there are exactly $\begin{pmatrix} mn \\ r \end{pmatrix}$ different right (left) solvents [16].

C. Block Vandermonde matrices

As for an eigenvalue system, a block Vandermonde matrix can be defined for solvents with particular properties [24].

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

$$V_R = \begin{pmatrix} I & I & I & \cdots & I \\ R_1 & R_2 & R_3 & \cdots & R_r \\ R_1^2 & R_2^2 & R_3^2 & \cdots & R_r^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_1^{r-1} & R_2^{r-1} & R_3^{r-1} & \cdots & R_r^{r-1} \end{pmatrix} \quad (15)$$

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

$$V_L = \begin{pmatrix} I & L_1 & L_1^2 & \cdots & L_1^{r-1} \\ I & L_2 & L_2^2 & \cdots & L_2^{r-1} \\ I & L_3 & L_3^2 & \cdots & L_3^{r-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ I & L_r & L_r^2 & \cdots & L_r^{r-1} \end{pmatrix} \quad (16)$$

Remark 4: In [26] 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.

Theorem 2: If $V(R_1, ..., R_{k-1})$ is nonsingular and R_k is not a weak solvent of $A(t)$ constructed from $k-1$ solvents, then $V(R_1, V_k)$ is nonsingular and then an $A(t)$ constructed from k solvents exist.

Proof 2: see [16]

Remark 5: W is called a weak solvent of $A(t)$ if $A(W)$ is singular.

Theorem 3: If $A(t)$ has distinct latent roots, then there exists a complete set of right solvents of $A(X)$, $R_1, \cdots, R_m$, and for any such set of solvents, $V(R_1, \cdots, R_m)$ is nonsingular.

Proof 3: see [16]

D. Non-singularity

Theorem 4: A block Vandermonde matrix as defined in equations 15 and 16 are non-singular matrices if and only if the set of r solvents $\{R_1 ... R_r\}$ with multiplicities $\{m_1 ... m_r\}$ is a complete set.

Proof 4: see [16], [26]

Definition 6: If we let $\sigma[A(t)]$ denote the set of all latent roots of $A(t)$ and $\sigma[Ri]$ the set of eigenvalues of the right solvent R_i, then a complete set of right solvents is obtained if we can find r right solvents such that [24]:

$$\begin{cases} \bigcup_{i=1}^{r} \sigma(R_i) = \sigma(A(t)) \\ \sigma(R_i) \cap \sigma(R_j = \emptyset \end{cases} \quad (17)$$

And the block vandermonde matrix thus constructed is non-singular.

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 (covering totally the latent structure of $A(t)$) 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. The following definition summarizes that:

Definition 7: The set of left solvents of $A(t)$, which satisfies the following properties [24]:

$$\begin{cases} r = \sum_{i=1}^{k} m_i, \\ \sigma[A(t)] = \bigcup_{i=1}^{k} \sigma[L_i] \end{cases} \tag{18}$$

and a nonsingular block vandermonde matrix, is called the complete set of the left solvents of $A(t)$.

Remark 6: A complete set of right or left solvents will then describe completely the latent structure of $A(t)$.

IV. Main results

The following sections constitute the author's contribution.

A. 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:

$$V_2 = \begin{pmatrix} V_1 & I_m \\ R_1 & R_2 \end{pmatrix} \tag{19}$$

If we define the following matrices:

$B_1 = I_m$; $C_1 = R_1$ and $D_1 = R_2$

Then a BVM of order 3 (three solvents) is as follows:

$$V_3 = \begin{pmatrix} V_2 & B_2 \\ C_2 & D_2 \end{pmatrix} \tag{20}$$

Where

$B_2 = \begin{pmatrix} I_m \\ R_3 \end{pmatrix}$;

$C_2 = \begin{pmatrix} R_1^2 & R_2^2 \end{pmatrix}$;

$D2 = R_3^2$

The following theorem ia a deduction from previous results:

Theorem 5: A BVM of order r, constructed from r solvents, is as follows

$$V_r = \begin{pmatrix} V_{r-1} & B_{r-1} \\ C_{r-1} & D_{r-1} \end{pmatrix} \tag{21}$$

Where

$$B_{r-1} = \begin{pmatrix} I_m \\ R_r \\ R_r^2 \\ \vdots \\ R_r^{r-2} \end{pmatrix};\ D_{r-1} = R_r^{r-1}$$

and $C_{r-1} = \begin{pmatrix} R_1^{r-1} & R_2^{r-1} & \cdots & R_{r-1}^{r-1} \end{pmatrix}$

Proof 5: It is straight forward from the block partitioning of the BVM V_r as given in equation 15.

$$V_r = \left(\begin{array}{ccc|c} I & \cdots & I & I \\ R_1 & \cdots & R_{r-1} & R_r \\ \vdots & \vdots & \vdots & \vdots \\ R_1^{r-2} & \cdots & R_{r-1}^{r-2} & R_r^{r-2} \\ -- & -- & -- & -- \\ R_1^{r-1} & \cdots & R_{r-1}^{r-1} & R_r^{r-1} \end{array}\right)$$

Remark 7: The previous and 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.

B. Inverse of BVM

From [27] and [19], the inverse of a block partitioned matrix is given as follows:

If A^{-1} exists then:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + E * S_A^{-1} * F & -E * BS_A^{-1} \\ -S_A^{-1} * F & S_A^{-1} \end{pmatrix} \tag{22}$$

where $E = A^{-1}B$, $F = CA^{-1}$ and $S_A = D - CA^{-1}B$.

S_A is the Shur complement of matrix A and should be non-singular.

If D^{-1} exists then:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} S_D^{-1} & -S_D^{-1} * F \\ -E * S_D^{-1} & E * S_D^{-1} * F + D^{-1} \end{pmatrix} \tag{23}$$

where $E = D^{-1}C$, $F = BD^{-1}$, and $S_D = A - BD^{-1}C$.

S_D is the Shur complement of matrix D and should be non-singular.

Let us compute the inverse of a BVM of order r as given in equation 21 by using either equations 22 or 23 both holds! In our case both diagonal entries are non-singular.

$$V_r^{-1} = \begin{pmatrix} V_{r-1}^{-1} + E * S_{r-1}^{-1} * F & -E * S_{r-1}^{-1} \\ -S_{r-1}^{-1} * F & S_{r-1}^{-1} \end{pmatrix} \tag{24}$$

Where S_{r-1} is the Shur complement of matrix V_{r-1} at iteration $r-1$:

$$S_{r-1} = D_{r-1} - C_{r-1}V_{r-1}^{-1}B_{r-1} \tag{25}$$

$E = V_{r-1}^{-1} * B_{r-1}$

$F = C_{r-1} * V_{r-1}^{-1}$

and D_{r-1}, C_{r-1}, B_{r-1} are as given previously.

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

1) Algorithm: Let a complete set of solvents $\{R_1, ..., Rr\}$ and the corresponding BVM V_r as given in 15. From the matrix V_r, all sub-matrices (B_i, C_i, D_i and S_i) will be first constructed, then the inverse is computed. The algorithm uses a function which computes the inverse of the Shur Complement.

Step1: Let $INV = I_m$

Step2:

for $i = 2 * m$ to $r * m$ with step=m

$B_{i-1} = Vr(1 : i-2, i-1 : i)$;

$C_{i-1} = Vr(i-1 : i, 1 : i-2)$;

$D_{i-1} = Vr(i-1:i, i-1:i)$;
$E_{i-1} = INV * B_{i-1}$;
$F_{i-1} = C_{i-1} * INV$;
$S_{i-1} = D_{i-1} - C_{i-1} * INV * B_{i-1}$;
$INV = \begin{pmatrix} INV + E * S_{i-1}^{-1} * F & -E * S_{i-1}^{-1} \\ -S_{i-1}^{-1} * F & S_{i-1}^{-1} \end{pmatrix}$;
`endfor`

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

So the overall complexity of our algorithm is: $O((r-2) * m^3)$.

3) Example: Let a matrix polynomial of order $m = 2$ and degree $r = 4$:

$$A(t) = I_2 t^4 + A_3 t^3 + A_2 t^2 + A_1 t + A_0$$

with $A_3 = \begin{pmatrix} -1 & 2 \\ 3 & -5 \end{pmatrix}$; $A_2 = \begin{pmatrix} 2 & 5 \\ 3 & -1 \end{pmatrix}$
and $A_1 = \begin{pmatrix} -1 & -1 \\ 3 & -2 \end{pmatrix}$; $A_0 = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}$

Then using the "`polyeig`" function of Matlab we obtained the latent values and vectors of $A(t)$, and constructed its solvents. The method to construct solvents from a set of latent values and vectors are given in [28].

This matrix polynomial presents a full set of solvents as follows:

$$R_1 = \begin{pmatrix} 87.40 & -181.70 \\ 42.21 & -87.70 \end{pmatrix}; \quad R_2 = \begin{pmatrix} 0.57 & -1.11 \\ 0.48 & -0.37 \end{pmatrix};$$
$$R_3 = \begin{pmatrix} 0.96 & 4.42 \\ -0.60 & -1.86 \end{pmatrix}; \quad R_4 = \begin{pmatrix} -3.23 & -4.50 \\ 7.75 & 10.25 \end{pmatrix};$$

The associated Vandermonde matrix is as follows:

$$V_4 = \begin{pmatrix} I_2 & I_2 & I_2 & I_2 \\ R_1 & R_2 & R_3 & R_4 \\ R_1^2 & R_2^2 & R_3^2 & R_4^2 \\ R_1^3 & R_2^3 & R_3^3 & R_4^3 \end{pmatrix}$$

The code in Matlab to compute the inverse is as follows:

```
IV=eye(2);
for i=4:2:8
 B=V4(1:i-2,i-1:i);
 C=V4(i-1:i,1:i-2);
 D=V4(i-1:i,i-1:i);
 E=IV*B;
 F=C*IV;
 delta=D-C*IV*B;
 IV=[IV+E/delta*F -E/delta;
-inv(delta)*F inv(delta)];
 endfor
```

And the result of the execution as given by Matlab:

$$IV = \begin{pmatrix} -0.05 & -0.10 & -0.02 & 0.12 & -0.14 & -0.22 & 0.02 & 0.03 \\ -0.02 & -0.05 & -0.01 & 0.06 & -0.07 & -0.11 & 0.01 & 0.01 \\ 0.55 & 0.72 & 0.44 & 0.68 & 0.34 & -0.15 & 0.12 & 0.09 \\ 0.39 & 0.64 & -0.14 & -0.57 & 0.23 & -0.21 & -0.11 & 0.01 \\ 0.03 & -0.30 & -0.59 & 0.10 & -0.59 & 1.10 & -0.02 & -0.21 \\ -0.01 & 0.17 & 0.28 & -0.20 & 0.15 & -0.25 & 0.01 & 0.06 \\ 0.47 & -0.32 & 0.17 & -0.90 & 0.40 & -0.73 & -0.12 & 0.10 \\ -0.37 & 0.24 & -0.13 & 0.70 & -0.31 & 0.57 & 0.09 & -0.08 \end{pmatrix}$$

We used the "`tic/toc`" functions of Matlab to determine the execution time T_1 of our procedure to be compared to the time T_2 of the "`inv`" function of Matlab.

The results are:
$T_1 = 2.2283e-005$
$T_2 = 2.4673e-004$
`speed` = 11.072

Remark 8: The proposed algorithm is 10 times quicker than the procedure used by Matlab.

C. Determinant

From [27], the determinant of a block partitioned matrix is as follows:

If A is non-singular:

$$det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = detA.det[D - CA^{-1}B] \qquad (26)$$

And if D is non-singular:

$$det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = detD.det[A - BD^{-1}C] \qquad (27)$$

Using this equation and the block decompositions given in the previous sections we deduced the following result.

Let us compute the determinant of a BVM of order r as given in equation 21 by using either equations 26 or 28 both holds!

$$detV_r = detV_{r-1}.detS_{r-1} \qquad (28)$$

Where S_{r-1} is the Shur complement of matrix V_{r-1} at iteration $r-1$:

$$S_{r-1} = D_{r-1} - C_{r-1}V_{r-1}^{-1}B_{r-1} \qquad (29)$$

where D_{r-1}, C_{r-1} and B_{r-1} are as given previously.

The same procedure will be used to determine the determinant of the BVM V_{r-1}.

Remark 9: The determinant is, in general, needed with the inverse. So the inverse of V_{r-1} is computed using the previous algorithm.

1) Algorithm: Let a complete set of solvents $R_1 ... R_r$ and the corresponding BVM V_r as given in equation 15. 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`$=m$
$V_{i-1} = Vr(1:i-2, 1:i-2)$;
$B_{i-1} = Vr(1:i-2, i-1:i)$;
$C_{i-1} = Vr(i-1:i, 1:i-2)$;

$D_{i-1} = Vr(i-1:i, i-1:i)$;
$S_{i-1} = D_{i-1} - C_{i-1} * V_{i-1}^{-1} * B_{i-1}$;
$Det = Det * determinant(S_{i-1})$;
endfor

2) Algorithmic complexity:: The number of iterations is $r-2$. 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 $O(m^3)$. So the overall complexity of our algorithm is: $O((r-2) * m^3)$.

3) Example: The same example than in the previous section is used to illustrate the determinant of V_4.

The code in Matlab to compute the determinant is as follows:

```
DD=1;
for i=4:2:8
 V=V4(1:i-2,1:i-2);
 B=V4(1:i-2,i-1:i);
 C=V4(i-1:i,1:i-2);
 delta=V4(i-1:i,i-1:i)-C/V*B;
 DD=DD*det(delta);
end
```

The right matrix divide is used instead of the function "`inv`" as advised by Matlab.

And the result as given by Matlab:

DD = -1.0037e+007

We used the "`tic/toc`" functions of Matlab to determine the execution time T_1 of our procedure to be compared to the time T_2 of the "`det`" function of Matlab. The results are:

T_1 = 7.2925e-006

T_2 = 1.0615e-004

speed =$T1/T2$= 14.556

The proposed algorithm is 14 times quicker than the procedure used by Matlab.

V. 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.

A. Parallel inverse of block Vandermonde matrix

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

B. Parallel determinant of block Vandermonde matrix

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
$V_{i-1} = V_r(1:i-2, 1:i-2)$;
$B_{i-1} = V_r(1:i-2, i-1:i)$;
$C_{i-1} = V_r(i-1:i, 1:i-2)$;
$D_{i-1} = V_r(i-1:i, i-1:i)$;
End
$S_{i-1} = D_{i-1} - C_{i-1} * V_{i-1}^{-1} * B_{i-1}$;
$Det = Det * determinant(S_{i-1})$;
end

C. Algorithmic Complexity

The overall time complexity of the two algorithms is the same as before: $O((r-2) * m^3)$. 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`. The execution time, obtained using "`tic/toc`" functions of Matlab, was greater than the sequential execution time, because of the large amount of data flowing between the cores, at each iteration.

VI. Conclusion

In this paper new results on the computation of the inverse and the determinant of a block Vandermonde matrix are given. Efficient algorithms are proposed with their algorithmic complexities. The parallelization of the algorithms, based on data and task decomposition, is proposed.

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 are needed.

This paper is a part of a project on control methods of dynamic multi-variable systems and their parallelization.

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 and 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] A. A. Al-Shaikhi and J. Ilow, Vandermonde matrix packet-level FEC for joint recovery from errors and packet loss. in proceedings of IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications,Cannes, France, 2008, pp. 1-6

[7] N. Bonello, S. Cheng and L. Hanzo, Construction of Regular Quasi-Cyclic Protograph LDPC codes based on Vandermonde Matrices. in proceedings of IEEE 68th Vehicular Technology Conference, VTC 2008-Fall, Calgary, BC, 2008, pp 1-5.

[8] J. T. Tou, Determination of the inverse Vandermonde matrix. IEEE Trans. Automatic Control (Correspondence). AC-9, pp. 314, 1964.

[9] J. D. Brule, A note on the Vandermonde determinant. IEEE Trans. Automatic Control (Correspondence). AC-9, pp. 314-215, 1964.

[10] G. C. Reis, A matrix formulation for the inverse Vandermonde matrix. IEEE Trans. Automatic Control (Correspondence). AC-12, pp. 793, 1967.

[11] S. Yan and A. Yang, Explicit Algorithm to the Inverse of Vandermonde Matrix. In proceedings of ICTM International Conference on Test and Measurement, Hong Kong, 2009, pp. 176-179.

[12] A. Klein and P. Spreij, Some Results on Vandermonde Matrices With an Application to Time Series Aanalysis. SIAM J. Matrix Aanl. Appl. 25 (1), pp. 213-223, 2003.

[13] A. Eisinberg, G. Fedele, On the inversion of the Vandermonde matrix. Applied Mathematics and Computation. 174, pp. 1384-1397, 2006.

[14] I.Gohberg and V. Olshevsky, The fast generalized Parker-Traub algorithm for inversion of Vandermonde and related matrices. Journal of complexity. 13 (2), pp. 208-234, 1997.

[15] J. E. Dennis, J. F. Traub and 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.

[16] J. E. Dennis, J. F. Traub and R. P. Weber, The algebraic theory of matrix polynomials. SIAM J. Numer. Anal. 13 (6), pp. 831-845, 1976.

[17] J. E. Dennis, J. F. Traub and R. P. Weber, Algorithms for solvents of matrix polynomials. SIAM J. Numer. Anal. 15 (3), pp. 523-533, 1978.

[18] D. R. Richman, A result about block Vandermonde Matrices. Linear and Multilinear Algebra. 21, pp. 181-189, 1987.

[19] Y. Choi and J. Cheong, 2009. New Expressions of 2x2 Block Matrix Inversion and Their Application. IEEE Trans. Autom. Control. 54 (11), 2648-2653.

[20] A. Asif and J. M. F. Moura, Inversion of block matrices with block banded inverses: application to Kalman-Bucy filtering. in proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Istanbul, Turkey. 1, 2000, pp. 608-611.

[21] L. Shang, Z. Wang, S. G. Petiton and F. Xu, Solution of Large Scale Matrix Inversion on Cluster and Grid. Seventh International Conference on Grid and Cooperative Computing, Shenzhen, China, 2008, pp. 33-40.

[22] M. Yaici and K. Hariche, On eigenstructure assignment using block poles placement. European J. Control. 20, pp. 217-226, 2014.

[23] B. Ycart, A case of mathematical eponymy: the Vandermonde determinant. Revue d'Histoire des Mathématiques.9 (1), pp. 43-77, 2013.

[24] L.S. Shieh, F.R. Chang and B.C. Mcinnis, The block partial fraction expansion of a matrix fraction description with repeated block poles. IEEE Trans. Autom. Control. 31 (3), pp. 236239, 1986.

[25] F. Kuo and G. S. Chen, A recursive algorithm for coprime fractions and Diophantine equations. IEEE Trans. Autom. Contr. AC-34 (12), pp. 1276-1279, 1989.

[26] K. Hariche and E. D. Denman, Interpolation theory and Lambda-matrices. J. Math. Anal. Appl. 143, pp. 530-547, 1989.

[27] T. Kailath, Linear Systems. Prentice Hall, New Jersey, 1980.

[28] M. Yaici and K. Hariche, On Solvents of Matrix Polynomials. International Journal of Modeling and Optimization. 4 (4), pp. 273-277, 2014.