Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 1 (2024), pp. 189-201

http://www.ripublication.com/gjpam.htm
https://dx.doi.org/10.37622/GJPAM/20.1.2024.189-201

Division Free Inversion of Certain Class of Vandermonde Matrices

Anand Aruna Kumar[1], Rajesh Tengli[2] and S.K. Srivatsa[3]

[1]*Senior Staff Engineer, Intel Corporation, Penang, Malaysia*
[2]*Senior Consultant, Capgemini, SE*
[3]*Professor, Purnapramati - A Centre for Integrated Learning, Bangalore, India*

Abstract

The study of matrices and determinants has a long history. The subject has seen great progress with the advent of computers. In the present era of building models of machine learning the use of matrices of large sizes has become imminent. Solving a system of equations using matrices has been a paradigm shift in the theory of equations and fits perfectly well for solving them with computers. One of the challenges that still persist is when the matrix is ill-conditioned (*either very large or small numbers as matrix elements*). As a result, one might encounter instabilities in inversion, either in the form of elemental inaccuracy or the final inverse itself. This challenge paved the way for inventing novel algorithms and numerical approaches to handle the calculation of these ill-conditioned matrices. In this article we propose a unique method for Vandermonde matrices. Here we convert a class of analytically solvable system of equations to a problem that in principle can be numerically friendly and accurate. We show that the method of conversion avoids division completely and is well behaved to handle ill-conditioned matrices. We also provide simple programs that can be used as a model to use on large dimensional matrices and show how the errors can be minimized with the proposed approach.

Keywords: Vandermonde, nilpotent, multinomial expansion, ill-conditioned, LU decomposition, division free.

1. INTRODUCTION

Inverse of a matrix is a fundamental concept in linear algebra. For a square matrix A, if there exists another matrix B such that their product results in the identity matrix (AB = BA = I), then B is said to be the inverse of A, denoted as A^{-1}. Finding the inverse of a matrix is a crucial operation in linear algebra and has various applications in mathematics, sciences and engineering [1], [2], [3], [4]. The inverse of a matrix allows us to solve linear systems of equations, compute determinants, and solve certain optimization problems. There are many methods to finding inverses some of which are Gaussian Elimination with back substitution, Matrix adjoint method, elementary row operations, matrix decomposition methods - LU, QR, Singular value decomposition (SVD), matrix inversion lemma, and pseudo-inverses. Most often, the latter method of pseudo-inverses are used in case of non-square and ill-conditioned matrices [5].

An ill-conditioned matrix is one that is very sensitive to small changes in its elements, resulting in significant changes the elements of its inverse. This sensitivity makes it difficult to compute the inverse accurately using a numerical procedure. Similarly, rounding errors and truncation leads to significant loss of precision. As a result, the computed inverse might not accurately represent the true inverse. When finding inverses of matrices numerically for ill-conditioned ones, it is essential to use techniques that avoid or minimize division by small or large numbers. Often times, direct computation of the inverse of an ill-conditioned matrix is avoided and alternate methods such as Scaling, Regularization, Matrix decomposition and method of Pseudoinverses.

The choice of method depends on the properties of a given matrix and often for a specific application. For small matrices or when precision is a concern, direct methods like Gaussian elimination or LU decomposition are popular methods. For large and sparse matrices, iterative methods pivoting techniques or decomposition-based approaches might be more efficient. In some cases, a matrix may not be invertible. In such a case a pseudo-inverse (also known as the Moore-Penrose inverse) can be used. It is commonly used in applications like least squares regression. However, we should note that this inverse is not a unique solution.

In the next section, we elaborate the specific topic of interest on matrix inversion for Vandermonde type of matrices. The approach presented utilizes in constructing a Vandermonde matrix using algebraic polynomial of distinct roots.

2. PROBLEM FORMULATION

Let a_1, a_2, a_3, ... a_n be any ordered sequence of n distinct natural numbers. Define, functional values by the expression such that $f_n(1)$ as $(a_1 + a_2 + a_3 + ... + a_n)$, $f_n(2)$ be $(a_1a_2 + a_2a_3 + a_3a_4 + ... + a_{n-1}a_n)$, sum of products of numbers taken two at a time without repetitions, $f_n(3)$ being sum of product of combinations of three numbers $(a_1a_2a_3 + a_2a_3a_4 + ...)$, without repetitions, and finally, $f_n(n)$ be products of all n elements a_1, a_2, a_3 and a_n. We use the notation $f_n(k)$ for sum of product of k terms of n elements taken. In one simple example, the a_is can be first n natural numbers. It is easy to show in this case that $f_n(1) = n(n+1)/2$ and $f_n(n) = n!$. Yet, even in this simple case of the choice of a_i's, the calculation of $f_n(k)$ is non-trivial for $1 < k < n$. Using this as the basis of the problem, we explore in detail to provide computation friendly method to evaluate $f_n(k)$ which involves matrix method and inverse computation.

Mathematical interpolation is a method for estimating a function's value at locations when the function is specified at certain points in the domain of the function. In this context too - the estimation of the coefficients of the polynomial expression, these same $f_n(k)$ occur naturally in this interpolation problem and lead to Vandermonde matrices [6].

Consider a polynomial of n^{th} order, $P_n(x)$ of the form given by equation (1):

$$P_n(x) = (x + a_1)(x + a_2)...(x + a_n) \tag{1}$$

with $a_1, a_2, ..., a_n$ being distinct positive integers. Equation (1) can be expanded and written as (2):

$$P_n(x) = x^n + f_n(1)x^{n-1} + f_n(2)x^{n-2} + ... + f_n(n) \tag{2}$$

where $f_n(k)$s for $(1 \leq k \leq n)$ are the terms we are interested in computing. Clearly, $x = -a_k$, are the zeroes of $P_n(x)$. In other words, for each k, we get n equations of the form:

$$P_n(-k) = (-k)^n + f_n(1)(-k)^{n-1} + f_n(2)(-k)^{n-2} + ... + f_n(n) = 0;\ 1 \leq \forall k \leq n \tag{3}$$

These n equations, on re-arrangement, can be written in the following form for each k:

$$f_n(1)/k - f_n(2)/k^2 + f_n(3)/k^3 + ... - f_n(n)/k^n = 1 \tag{4}$$

and can be compactly written as

$$\begin{pmatrix} 1 & -1 & \dots & (-1)^{n-1} \\ 1/2 & -1/2^2 & \dots & (-1)^{n-1}/2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1/n & -1/n^2 & \dots & (-1)^{n-1}/n^n \end{pmatrix} \begin{pmatrix} f_n(1) \\ f_n(2) \\ \vdots \\ f_n(n) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{5}$$

The above matrix contains symmetries in rows and columns, the row terms are in geometric series and the column elements are simply terms of a harmonic p-series, $1/n^p$. The matrix is by definition, a Vandermonde matrix, general form of which is shown below.

$$\begin{pmatrix} 1 & -1 & \dots & (-1)^{n-1} \\ y_2 & -y_2^2 & \dots & (-1)^{n-1}y_2^n \\ y_3 & -y_3^2 & \dots & (-1)^{n-1}y_3^n \\ \vdots & \vdots & \ddots & \vdots \\ y_n & -y_n^2 & \dots & (-1)^{n-1}y_n^n \end{pmatrix} \begin{pmatrix} f_n(1) \\ f_n(2) \\ f_n(3) \\ \vdots \\ f_n(n) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \implies V \cdot \hat{f_n} = \hat{1}_n \tag{6}$$

Here, V, $\hat{f_n}$ and $\hat{1}_n$ are the '$n \times n'$ matrix, column vector $f_n(k)$ and unit column vector respectively. The determinant of V is non singular and its value, up to a constant multiplication factor, can be seen by using the factorization method and is given by

$$det(V) = \begin{vmatrix} 1 & -1 & \dots & (-1)^{n-1} \\ y_2 & -y_2^2 & \dots & (-1)^{n-1}y_2^n \\ y_3 & y_3^2 & \dots & (-1)^{n-1}y_3^n \\ \vdots & \vdots & \ddots & \vdots \\ y_n & -y_n^2 & \dots & (-1)^{n-1}y_n^n \end{vmatrix} = \prod_{i=2,j=2,i\neq j}^{n} c(y_i - y_j)(1 - y_i)y_j$$

Thus, the column vector of $f_n(k)$ can be found analytically by inversion of matrix V.

$$\begin{pmatrix} f_n(1) \\ f_n(2) \\ \vdots \\ f_n(n) \end{pmatrix} = \begin{pmatrix} 1 & -1 & \dots & (-1)^{n-1} \\ 1/2 & -1/2^2 & \dots & (-1)^{n-1}/2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1/n & -1/n^2 & \dots & (-1)^{n-1}/n^n \end{pmatrix}^{-\mathbf{1}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{7}$$

3. METHODS AND DISCUSSIONS

This section focuses on a general method of reduction to obtain inverse of these Vandermonde matrices with an emphasis on large values of n. Brute force approach becomes impractical for numbers $n > 100$, as the matrices become ill-conditioned. A direct computation for n = 50 and n = 100 was envisaged using SageMath/ Octave on a computer with 16GB RAM and 256GB hard disk drive. Though the expected solutions are all integers for large n values, the elements of inverse matrix had fractional solutions or complex numbers. This computational challenge is attributable to floating point limitations of ordinary computers. We provide a method to find an inverse of the Vandermonde matrix as shown below. First, we reduce the Vandermonde matrix, V to an upper triangular matrix using row-Echelon reduction method. The resulting matrix equation (8) is shown below:

$$\begin{pmatrix} 1 & -1 & \dots & (-1)^{n-1} \\ 0 & -1 & a_2+1 & (R_2-R_1)\ terms \\ 0 & 0 & 1 & (R_3-R_1)-(R_2-R_1)\ terms \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (-1)^{n-1} \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(n) \end{pmatrix} = \begin{pmatrix} 1 \\ -1/a_2 \\ 1/(a_2a_3) \\ \vdots \\ (-1)^{n-1}/(a_2a_3..a_n) \end{pmatrix} \tag{8}$$

We note that the equation (8) can be written as a sum of two matrices: a diagonal matrix with the principal diagonal elements as ± 1, which is a Unitary diagonal matrix $E(n)$ with determinant ± 1 and another matrix that is an upper triangle nilpotent matrix of rank $(n-1)$. This can be compactly represented as

$$(E+N)\cdot(F) = C \tag{9}$$

where E is the unitary matrix, N being strictly upper triangular matrix, F is the column vector of interest, and C is the last column of the echelon matrix shown in (8). We also notice that $det(E)$ can either be ± 1, depending on choice of even or odd values of n. To remove the ambiguity, we convert to a equivalent class of systems which have just the identity matrix $I(n)$. Left multiplying equation (9) with E and using $E^2 = I$, $N' = E\cdot N, C' = E\cdot C$, we obtain

$$E\cdot(E+N)\cdot(F) = E\cdot C \implies (I+N')\cdot(F) = C' \tag{10}$$

$$F = (I+N')^{-1}\cdot C' = F = (I+(-N')+(-N')^2\cdots-+(-1)^{n-1}\cdot(N')^{n-1})\cdot C' \tag{11}$$

The series is terminate at $n-1$ order of N as $(N')^n$ is a zero matrix.

The solution for each $f_n(k)$ of column matrix F in question is obtained by assigning a_k values $1/k$, $1 \le \forall k \le n$ in N' and its summation terms multiplied with C' column

vector. We illustrate with an example in the next section to show that the matrix entries of (11) are free of large denominators.

In the above illustration, F is a product of an upper triangular square matrix and a column vector C', is constrained by the way F and C' are related through (7) . It is not difficult to generalize from this to solve for the Vandermonde matrix that is independent constraints.

In order to do that, we consider the $L \cdot U$ decomposition form and assign $(I + K)$ to be an lower diagonal matrix, say, L of the $L \cdot U$ form and another equivalent form for upper diagonal as $U = D \cdot (I + Y)$. We need a diagonal normalizer D to factorize into $(I + Y)$ to be usable for nilpotent expansions during inversion, see [7] for an equivalence with lower triangular matrix. Here, K and Y are strictly lower and upper triangular matrices.

$$V = L \cdot U = (I + K) \cdot D \cdot (I + Y) \tag{12}$$

$$L^{-1} = (I + K)^{-1} = (I + (-K) + (-K)^2 ... + (-1)^{n-1} \cdot (K)^{n-1}) \tag{13}$$

$$U^{-1} = (D \cdot (I+Y))^{-1} = (I+Y)^{-1} \cdot D^{-1} = (I+(-Y)+(-Y)^2 ...+(-1)^{n-1} \cdot (Y)^{n-1}) \cdot D^{-1} \tag{14}$$

We now have two representations for L^{-1} and U^{-1} that is sufficient to use for any generic Vandermonde inverse.

$$\boxed{V^{-1} = (L \cdot U)^{-1} = U^{-1} \cdot L^{-1} = (I + Y)^{-1} \cdot D^{-1} \cdot (I + K)^{-1}} \tag{15}$$

This form provides all the requirements to produce a division free inversion of the Vandermonde matrix. One can independently compute basic elements K, Y and D using terms of V. We refer to work of *Sheng-Liang Yang*, [11] for exemplary illustration of L and U forms which can be generalized to obtain components of nilpotent terms for both strictly upper and lower matrices. To improve computational efficiency, it is required to have numerical elements of nilpotent matrices, subsequent powers of it are faster than symbolic version of computation.

4. ILLUSTRATIONS

Consider an example for a small ($n = 5$) algebraic polynomial of (6). We illustrate a method based on multinomial expansion to obtain the elements of the N' matrix.

$V_5(5 \times 5)$, with $a_1 = 1$. The equation (6) reduces to

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ a_2 & -a_2^2 & a_2^3 & -a_2^4 & a_2^5 \\ a_3 & -a_3^2 & a_3^3 & -a_3^4 & a_3^5 \\ a_4 & -a_4^2 & a_4^3 & -a_4^4 & a_4^5 \\ a_5 & -a_5^2 & a_5^3 & -a_5^4 & a_5^5 \end{pmatrix} \begin{pmatrix} f_5(1) \\ f_5(2) \\ f_5(3) \\ f_5(4) \\ f_5(5) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{16}$$

The corresponding expressions for the Unitary matrix, E_5 and the nilpotent matrix N_5 are therefore,

$$E_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix};$$

$$N_5 = \begin{pmatrix} 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & a_2+1 & -(a_2^2+a_2+1) & (a_2^3+a_2^2+a_2+1) \\ 0 & 0 & 0 & -(a_3+a_2+1) & (a_3^2+a_2^2+a_3a_2+a_3+a_2+1) \\ 0 & 0 & 0 & 0 & (a_4+a_3+a_2+1) \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and the transformed nilpotent matrix N_5' and the column vector C_5' after left multiplication with E_5 are:

$$N_5' = \begin{pmatrix} 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & -(a_2+1) & (a_2^2+a_2+1) & -(a_2^3+a_2^2+a_2+1) \\ 0 & 0 & 0 & -(a_3+a_2+1) & (a_3^2+a_2^2+a_3a_2+a_3+a_2+1) \\ 0 & 0 & 0 & 0 & -(a_4+a_3+a_2+1) \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$C_5' = \begin{pmatrix} 1 \\ 1/a_2 \\ 1/(a_2a_3) \\ 1/(a_3a_4a_5) \\ 1/(a_2a_3a_4a_5) \end{pmatrix}$$

From this example it is clear that all our entries of nilpotent expansion terms are free of large denominators (here it is trivially equal to 1). Now, to compute $f_5(i)$, simply solve by expanding $(I + N')^{-1} \cdot C'$ given in (11) by substituting for N'_5 and term by term matrix addition of non vanishing powers of N'_5, followed by a trivial column matrix multiplication of C'_5.

5. COMPUTATIONAL 'TRICK' FOR OBTAINING ELEMENTS OF N'

We observe that elements of the matrix N' have a certain symmetry in the powers of the various a_is that enables the terms of N' to be culled out of a expansion of multinomial expression but without the corresponding coefficients. The corresponding generating function, for the elements of the N' matrix with $a_1 = 1$, is the multinomial expansion of the expression: $(1 + a_2 + .. + a_k)^n$. On expansion, we get all the terms of the matrix elements of N'. A simple computer program is devised to replace multinomial coefficients with 1 in each term and complete computation of the inverse. We further emphasize on the trick here is to have an analytical form for elements of the nilpotent matrix. Explicitly, it is a symbolic generalization that could have arbitrary a_is of Vandermonde matrix, and the routine deduces all elements of N without having to deal with large valued divisors

For e.g., let us consider a 10 dimensional matrix N'_{10} and compute the multinomial terms for a particular element of the matrix, say $N'_{7,10}$ (7^{th} row, 10^{th} column term). Based on previous discussion, the first non-vanishing term of 7^{th} row will be the term, $N'_{7,8}$, which is first order, the number of variables will be from a_2 through a_7; the $N'_{7,9}$ will be a sum of first and second degree terms of a_i; and the degree of 10^{th} column would be 3. Thus, a multinomial series expansion of $(1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7)^3$ would be contain all possible combinations of a_i, except for multinomial numerical coefficients. If we can set these coefficients to 1, in the expansion, all the terms of matrix element $N'_{7,10}$ can be obtained. There are no analytical expression for doing such a thing, thus needs a computation trick. For example, the polynomial, or generating function for $N'_{7,10}$ is given in (17).

$$(N'_{7,10})_{generator} = (a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7)^3 \tag{17}$$

$$= \sum_{\substack{k_1+k_2+\cdots+k_7=3 \\ k_1,k_2,\ldots,k_7\geq 0}} \binom{3}{k_1, k_2, \ldots, k_7} \prod_{s=1}^{7} a_s^{k_s}$$

with $a_1 = 1$ (*representation to match with convention of multinomial expansion*). The multinomial coefficients are then replaced by 1 to get matrix term, $N'_{7,10}$. (*see next section for details*).

Single term computation: an example-for V_{10}'s version of equation (6):

For computational trick, we use the *MATLAB/Octave* built in routines to identify numerical coefficients in the multinomial expansion and store them as coefficients in an array, i.e., the expansion in *MATLAB/Octave* is stored as $[M_c^T] \cdot [C]$, where M_c^T is transposed array of multinomial coefficients and $[C]$ is a column of multiplicands. We then simply add all the terms of $[C]$ column matrix. Using this built in feature, the matrix elements are computed term by term and the N' matrix is stored. A typical result from computation for a 10×10 nilpotent matrix' $N'_{10_{(7,10)}}$ term is shown below. Here, N'_{10} is a nilpotent matrix of 10×10 with ${}^{10}C_2$ non-vanishing upper triangle elements.

$$
\begin{aligned}
N'_{10_{(7,10)}} &= -(a_2^3 + a_2^2 a_3 + a_2^2 a_4 + a_2^2 a_5 + a_2^2 a_6 + a_2^2 a_7 + a_2^2 + a_2 a_3^2 + a_2 a_3 a_4 \\
&+ a_2 a_3 a_5 + a_2 a_3 a_6 + a_2 a_3 a_7 + a_2 a_3 + a_2 a_4^2 + a_2 a_4 a_5 + a_2 a_4 a_6 + a_2 a_4 a_7 + a_2 a_4 + a_2 a_5^2 \\
&+ a_2 a_5 a_6 + a_2 a_5 a_7 + a_2 a_5 + a_2 a_6^2 + a_2 a_6 a_7 + a_2 a_6 + a_2 a_7^2 + a_2 a_7 + a_2 + a_3^3 + a_3^2 a_4 \\
&+ a_3^2 a_5 + a_3^2 a_6 + a_3^2 a_7 + a_3^2 + a_3 a_4^2 + a_3 a_4 a_5 + a_3 a_4 a_6 + a_3 a_4 a_7 + a_3 a_4 + a_3 a_5^2 \\
&+ a_3 a_5 a_6 + a_3 a_5 a_7 + a_3 a_5 + a_3 a_6^2 + a_3 a_6 a_7 + a_3 a_6 + a_3 a_7^2 + a_3 a_7 + a_3 + a_4^3 + a_4^2 a_5 \\
&+ a_4^2 a_6 + a_4^2 a_7 + a_4^2 + a_4 a_5^2 + a_4 a_5 a_6 + a_4 a_5 a_7 + a_4 a_5 + a_4 a_6^2 + a_4 a_6 a_7 + a_4 a_6 \\
&+ a_4 a_7^2 + a_4 a_7 + a_4 + a_5^3 + a_5^2 a_6 + a_5^2 a_7 + a_5^2 + a_5 a_6^2 + a_5 a_6 a_7 + a_5 a_6 + a_5 a_7^2 \\
&+ a_5 a_7 + a_5 + a_6^3 + a_6^2 a_7 + a_6^2 + a_6 a_7^2 + a_6 a_7 + a_6 + a_7^3 + a_7^2 + a_7 + 1)
\end{aligned}
$$

Likewise, the remaining non-vanishing terms can be computed and stored as matrix elements for a given N'. It is sufficient to use N' and expand the powers, which are all multiplicative and additive by construction (Refer Program 2 for details).

A couple of final notes on the matrix term. The sign factor (± 1) for a given term is simply deduced by evaluating $(-1)^{i+j}$, where i and j are row and column index respectively. Second important note being, the elements of distinct sized matrices, N'_α and N'_β, where $\alpha > \beta$, then all i, j terms of N'_β are contained in N'_α. This might be particularly useful in re-using the terms needed to build the matrix element library. Sign factors do not apply if the V is purely using $L \cdot U$ decomposition, in other words, elements of K obtained through (17) can use absolute value. Lastly, a different generating function/methods are needed to obtain elements of K and Y matrices.

Computational considerations

In case of Vandermonde matrix, inversion operation gets greatly simplified to compute as the terms are in a geometric progression along a row or column. For a smaller dimensions, any method would work well, but here we are explicitly interested in

large factorials and their inverses, yet get integral solutions to the problem in question (say $1000!$ or $1/1000!$), the numbers are large or small depending on how it appears in evaluation. The alternate methods for matrix inversion target towards complete solutions of Gauss-Jordan methods with an identity matrix or diagonalized form, also known as reduced row echelon form (RREF). We have avoided this method, as RREF for large matrix size have the same issue of large denominators and inaccuracies in the matrix elements, since general matrix inverse routines use some or other form of RREF in their subroutines.

In the case of nilpotent matrix N (or N'), the algebraic expression for matrix element of N (or N') in the binomial expansion of equation (11) is shown. In any nilpotent $n \times n$ matrix of rank $(n-1)$, there are ${}^nC_2$ matrix elements. The number of terms in the expansion of equation (11) would have a total of ${}^{n-1}C_2 + {}^{n-2}C_2 + ... + 1 = n \cdot (n^2-1)/6$. Note that this is proportional to the total number of operations which are multiplication, addition or subtraction avoiding any or all division.

The big challenge in computation here is of the length of terms for each matrix element of the nilpotent matrix. Typically, the largest number of terms in any n-dimensional matrix would be ${}^nC_{n/2}$ for even n, or ${}^nC_{(n\pm1)/2}$ for odd n. This is also the term that takes the largest time for computation. Once the entries of nilpotent matrix N' are independently computed and constructed, the rest of process of power series expansion are simply multiplicative and additive. A suitable architecture can support such computations and we refer to an excellent treatise on computation methods here [8] for optimal choice of method for specific applications. Hence, we can thus construct a divisor free matrix inversion for a certain class of matrices like Vandermonde type.

6. CONCLUSION

The subject of computation of inverse of a matrix has been studied intensely and received great deal of attention due to modern use of matrix methods in Machine learning, Neural networks and other subjects. Solving a large systems of equations using computers is a regular feature in the present times. Yet, calculation of inverse of a matrix is difficult due to challenges related to singularity, computational complexity, numerical precision, ill-conditioning, storage requirements, dimensionality, and potential computational errors.

In this article we proposed a numerical-friendly method for the calculation of inverse for Vandermonde type of matrices. This method consists of separating a given matrix into upper and lower Nilpotent matrices through $L \cdot U$ decomposition. The inverse of which

has been shown to be completely division free by large/ small numbers. Therefore, this method or approach, makes it potentially easy and suitable for extending to a wide set of ill-conditioned matrices too. Finally, we have provided simple example programs for ideas, that can be developed to make inversion of Vandermonde matrices division free.

7. SUPPLEMENTARY MATERIAL: NUMERICAL IMPLEMENTATION, PROGRAMS

A limited set of very simple programs/routines are provided for the user to test them out and improvise on specific algorithms for larger matrices, all of the programs can be compiled using open source packages.

Program 1: General Matrix inversion using Sagemath: `https://www.sagemath.org`

```
# Program 1
# Change variables size_row size_col to test change the dim of the matrix
# Compute capability on ordinary PC upto 150x150 dimension.

# Example using 50x50 matrix
var('size_row size_col')
size_row =50
size_col = 50
A = matrix(size_row, size_col, lambda x,y:(-1)^(y)*(1/(x+1)^(y+1)))
AInv = A.inverse()
ColOf1s = matrix(QQ,size_row, 1, lambda x,y:1);
FinalFs = Ainv*ColOf1s;FinalFs
```

Program 2: Compute a specific matrix element of nilpotent matrix, N using modified multinomial expansion using Octave: `https://www.octave.org`.

```
# Program  2:
# Expression for Matrix element of $N_ij$.
# A simple program to generate symmetric polynomial of multiple variables
# This example uses N_{7,10} shown in the article.

clear()
pkg load symbolic # Load symbolic package.

syms a2 a3 a4 a5 a6 a7 # Load 6 variables a2..a7 as symbols---declaration

P = (1 + a2 + a3 + a4 + a5 + a6 + a7)^3; # Multinomial

[c,t] = coeffs(P); # Declaration for co-effecients and terms of polynomial
```

```
size(c) ; # Check the dimension of coeffient matrix.

R = sum(t(1,:)); # Sum of all terms of combinations of a_i polynomial.

L = latex(sym(R));  # This converts the symbolic result to a character

# Reference polynomial terms
M = latex(expand(P)); # Latex formatted polynomial

# Data and file management
fileID1 = fopen('<Filepath/filename.txt>','wt'); # File open to text.
fprintf(fileID1, '%s \n' ,L);
fclose(fileID1)

fileID2 = fopen('<Filepath/filenamepoly.txt>','wt');
fprintf(fileID2, '%s \n',M);
fclose(fileID2)
toc # Computational time

# Use (-1)^{i+j} as a multiplier to get the sign of N''s {ij} term.
```

REFERENCES

[1] Strang, Gilbert, (2022), Introduction to linear algebra, SIAM.

[2] Eldén, Lars, (2019), Matrix methods in data mining and pattern recognition, SIAM.

[3] Zhang, Xian-Da, (2017), Matrix analysis and applications, Cambridge University Press.

[4] Kelly, Alonzo, (2013), Mobile robotics: mathematics, models, and methods, Cambridge University Press.

[5] Teukolsky, Saul A and Flannery, Brian P and Press, WH and Vetterling, WT, (1992), Numerical recipes in C, SMR.

[6] Turner, L Richard, (1966), Inverse of the Vandermonde matrix with applications, (No. NASA-TN-D-3547), NASA.

[7] I. Miranda-Sánchez and J. López-Bonilla , Studies in Nonlinear Science 5(4): pp 57-58, 2020.

[8] Guorong Wang, Yimin Wei, Sanzheng Qiao, (2018), Generalized Inverses: Theory and Computations, Science Press, Beijing, Springer.

[9] Yiu-Kwong Man, Advances in Theoretical and Applied Mathematics, Vol 13, # 1, pp 15-21, Research India Publications.

[10] E. A. Rawashdeh, Matematički Vesnik (Mathematics Society of Serbia), 71, 3, pp 207-213, Sep 2019.

[11] Sheng-liang Yang, Discrete Applied Mathematics, 146, pp 102-105 (2005), Science Direct, Elsevier Publishers, `https://www.elsevier.com/locate/dam`.