A New Recursive Algorithm For Inverting A General Comrade Matrix

A. A. KARAWIA*
Computer science unit, Deanship of educational services,
Qassim University, P.O.Box 6595, Buraidah 51452, Saudi Arabia.
E-mail: kraoieh@qu.edu.sa

Abstract

In this paper, a reliable symbolic computational algorithm is presented for inverting a general comrade matrix by using parallel computing along with recursion. The computational cost of the algorithm is $O(n^2)$. The algorithm is implementable to the Computer Algebra System (CAS) such as MAPLE, MATLAB and MATHEMATICA. Three examples are presented for the sake of illustration.

Keywords: Comrade matrices; LU factorization; Inverse matrix; Computer algebra systems(CAS).
AMS Subject Classification:15A15; 15A23; 65F05; 68W30; 11Y05; 33F10; F.2.1; G.1.0.

*Home Address: Mathematics Department, Faculty of Science, Mansoura University, Mansoura 35516, Egypt. E-mail:abibka@mans.edu.eg

1 Introduction

The general $n \times n$ comrade matrix, denoted by C, takes the form

$$C = \begin{pmatrix} \beta_1 & \alpha_1 & 0 & \cdots & \cdots & & 0 \\ \gamma_2 & \beta_2 & \alpha_2 & \ddots & & & 0 \\ 0 & \gamma_3 & \beta_3 & \alpha_3 & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \gamma_{n-1} & \beta_{n-1} & \alpha_{n-1} \\ a_n & a_{n-1} & \cdots & a_4 & a_3 & \gamma_n & \beta_n \end{pmatrix}, \quad n \geq 3. \tag{1.1}$$

The comrade matrix is a generalization of the companion matrix and is associated with a polynomial expressed as a linear combination of an arbitrary orthogonal basis. This matrix appears frequently in many areas of science and engineering, for example in linear multivariable systems theory[1], computing the Greatest Common Divisor of polynomials[2] and division of generalized polynomials [3]. The solution of a comrade linear system has been investigated by many authors (see for instance, [4-6]). Finding the inverse of a comrade matrix is usually required to solve this linear system. The motivation of the current paper is to establish an efficient algorithm for inverting a companion matrix of the form (1.1). The algorithm to compute the inverse of a general comrade matrix is based on the LU factorization of the matrix C in (1.1), which may fail if one of the denominators in the L matrix equals zero. The development of a symbolic algorithm is considered in order to remove all cases where the algorithm fails. Many algorithms for inverting a general comrade matrix require pivoting, for example the Gauss-Jordan elimination algorithm. Overall, pivoting adds more operations to the computational cost of an algorithm. These additional operations are sometimes necessary for the algorithm to work.

The paper is organized as follows. In Section 2, a new symbolic computational algorithm that will not break down, is constructed. In Section 3, three illustrative examples are given. Conclusions of the work are presented in Section 4.

2 Algorithm Construction

In this section we shall focus on the construction of a new symbolic computational algorithm for computing the determinant and the inverse

of a general comrade matrix. Firstly, we begin with computing the LU factorization of the matrix C.

Let

$$C = LU \tag{2.1}$$

where

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ \frac{\gamma_2}{\mu_1} & 1 & 0 & \ddots & & & 0 \\ 0 & \frac{\gamma_3}{\mu_2} & 1 & 0 & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \frac{\gamma_{n-1}}{\mu_{n-2}} & 1 & 0 \\ x_1 & x_2 & \cdots & \cdots & x_{n-2} & x_{n-1} & 1 \end{pmatrix}, \quad \mu_i \neq 0, \quad i = 1, 2, ..., n-2, \tag{2.2}$$

and

$$U = \begin{pmatrix} \mu_1 & \alpha_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \mu_2 & \alpha_2 & \ddots & & & 0 \\ 0 & 0 & \mu_3 & \alpha_3 & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 0 & \mu_{n-1} & \alpha_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & 0 & \mu_n \end{pmatrix} \tag{2.3}$$

The elements in the matrices L and U in (2.2) and (2.3) satisfy:

$$\mu_i = \begin{cases} \beta_1 & \text{if} \quad i = 1 \\ \beta_i - \frac{\alpha_{i-1}}{\mu_{i-1}}\gamma_i & \text{if} \quad i = 2, 3, ..., n-1 \\ \beta_n - \alpha_{n-1}x_{n-1} & \text{if} \quad i = n, \end{cases} \tag{2.4}$$

and

$$x_i = \begin{cases} \frac{a_n}{\mu_1} & \text{if} \quad i = 1 \\ \frac{1}{\mu_i}(a_{n-i+1} - \alpha_{i-1}x_{i-1}) & \text{if} \quad i = 2, 3, ..., n-2 \\ \frac{1}{\mu_{n-1}}(\gamma_n - \alpha_{n-2}x_{n-2}) & \text{if} \quad i = n-1. \end{cases} \tag{2.5}$$

We also have:

$$Det(C) = \prod_{i=1}^{n} \mu_i. \tag{2.6}$$

At this point it is convenient to formulate our first result. It is a symbolic algorithm for computing the determinant of a comrade matrix C of the form (1.1).

Algorithm 2.1 *To compute Det(C) for the comrade matrix C in (1.1), we may proceed as follows:*

INPUT *order of the matrix n and the components* α_i, $i = 1, 2, ..., n-1$, β_i, $i = 1, 2, ..., n$, γ_i, $i = 2, 3, ..., n$, *and* a_i, $i = 3, 4, ..., n$.

OUTPUT *The determinant of comrade matrix C.*

Step 1: *Set* $\mu_1 = \beta_1$. *If* $\mu_1 = 0$ *then* $\mu_1 = t$*(t is just a symbolic name) end if. Set* $x_1 = \frac{a_n}{\mu_1}$.

Step 2: *For* $i = 2, 3, ..., n-2$

Compute $\mu_i = \beta_i - \frac{\gamma_i}{\mu_{i-1}}\alpha_{i-1}$, *if* $\mu_i = 0$ *then* $\mu_i = t$,

Compute $x_i = \frac{1}{\mu_i}(a_{n-i+1} - \alpha_{i-1}x_{i-1})$,

Step 3: *Set* $\mu_{n-1} = \beta_{n-1} - \frac{\gamma_{n-1}}{\mu_{n-2}}\alpha_{n-2}$, *if* $\mu_{n-1} = 0$ *then* $\mu_{n-1} = t$ *end if,*

Set $x_{n-1} = \frac{1}{\mu_{n-1}}(\gamma_n - x_{n-2}\alpha_{n-2})$.

Set $\mu_n = \beta_n - \alpha_{n-1}x_{n-1}$, *if* $\mu_n = 0$ *then* $\mu_n = t$ *end if.*

Step 4: *Compute* $Det(C) = \left(\prod_{i=1}^{n}\mu_i\right)_{t=0}$.

The symbolic Algorithm 2.1 will be referred to as the **DETSGCM** algorithm. The computational cost of the **DETSGCM** algorithm is $7n-10$ operations. The new algorithm **DETSGCM** is very useful to check the nonsingularity of the matrix C

Now, when the matrix C is nonsingular, its inversion is computed as follows:

Let

$$C^{-1} = [S_{i,j}]_{1\le i,j\le n} = [Col_1, Col_2, ..., Col_n]. \tag{2.7}$$

where Col_m denotes the m_{th} column of C^{-1}, $m = 1, 2, ..., n$.

Since the Doolittle LU factorization of the matrix C in (1.1) is always possible we can use parallel computations to obtain the elements of the last two columns $Col_i = (S_{1,i}, S_{2,i}, ..., S_{n,i})^T$, $i = n$ and $n-1$ of C^{-1} as follows [7]:

Solving in parallel the standard linear systems whose coefficient matrix

L is given by (2.2)

$$L \begin{pmatrix} Q_1^{(n)} & Q_1^{(n-1)} \\ Q_2^{(n)} & Q_1^{(n-1)} \\ \vdots & \vdots \\ Q_{n-2}^{(n)} & Q_{n-2}^{(n-1)} \\ Q_{n-1}^{(n)} & Q_{n-1}^{(n-1)} \\ Q_n^{(n)} & Q_n^{(n-1)} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.8}$$

we obtain

$$\begin{pmatrix} Q_1^{(n)} & Q_1^{(n-1)} \\ Q_2^{(n)} & Q_1^{(n-1)} \\ \vdots & \vdots \\ Q_{n-2}^{(n)} & Q_{n-2}^{(n-1)} \\ Q_{n-1}^{(n)} & Q_{n-1}^{(n-1)} \\ Q_n^{(n)} & Q_n^{(n-1)} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \\ 1 & -x_{n-1} \end{pmatrix} \tag{2.9}$$

Hence, solving the following standard linear systems whose coefficient matrix U is given by (2.3)

$$U \begin{pmatrix} S_{1,n} & S_{1,n-1} \\ S_{2,n} & S_{2,n-1} \\ \vdots & \vdots \\ S_{n-2,n} & S_{n-2,n-1} \\ S_{n-1,n} & S_{n-1,n-1} \\ S_{n,n} & S_{n,n-1} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \\ 1 & -x_{n-1} \end{pmatrix} \tag{2.10}$$

gives the two columns Col_i, $i = n$ and $n-1$ in the forms:

$$S_{n,n} = \frac{1}{\mu_n}, \tag{2.11}$$

$$S_{i,n} = -\frac{\alpha_i S_{i+1,n}}{\mu_i}, \quad i = n-1, n-2, ..., 1, \tag{2.12}$$

$$S_{n,n-1} = -\frac{x_{n-1}}{\mu_n}, \tag{2.13}$$

$$S_{n-1,n-1} = \frac{1 - \alpha_{n-1} S_{n,n-1}}{\mu_{n-1}}, \tag{2.14}$$

$$S_{i,n-1} = \frac{-\alpha_i S_{i+1,n-1}}{\mu_i}, \quad i = n-2, n-3, ..., 1. \tag{2.15}$$

Using equations (2.11)-(2.15) and the fact $C^{-1}C = I_n$ where I_n is the $n \times n$ identity matrix, the elements in the remaining (n - 2) columns of C^{-1} may be obtained recursively using

$$Col_{n-2} = \frac{1}{\alpha_{n-2}}(E_{n-1} - \beta_{n-1}Col_{n-1} - \gamma_n Col_n), \tag{2.16}$$

$$Col_j = \frac{1}{\alpha_j}(E_{j+1} - \beta_{j+1}Col_{j+1} - \gamma_{j+2}Col_{j+2} - a_{n-j}Col_n), \quad j = n-3, n-4, ..., 1. \tag{2.17}$$

Here $E_r = (\delta_{1r}, \delta_{2r}, ..., \delta_{nr})^T$, $r = 1, 2, ..., n$, where δ_{ir} is the Kronecker symbol.

Note 2.1. Since the numeric algorithm to compute the inverse of a general comrade matrix is based on the LU factorization of the matrix C in (1.1), then it may be fail if $\mu_j = 0$, $j \in \{1, 2, ..., n\}$ or $\alpha_j = 0$, $j \in \{1, 2, ..., n-2\}$ as can be seen from (2.1) to (2.15) and from (2.16) to (2.17) respectively. To remove all cases in which numeric algorithm fails, we set all these variables equal to t(t is just a symbolic name) and after compute all elements of the inverse matrix we will set t equals to 0. it is convenient to give the following symbolic computational algorithm to compute the inverse of a general comrade matrix of the form (1.1) when it exists.

Algorithm 2.2 *To determine the inverse matrix of the general $n \times n$ comrade matrix C in (1.1) by using the relations (2.11)-(2.17).*

INPUT *Order of the matrix n and the components α_i, $i = 1, 2, ..., n-1$, β_i, $i = 1, 2, ..., n$, γ_i, $i = 2, 3, ..., n$, and a_i, $i = 3, 4, ..., n$.*

OUTPUT *Inverse comrade matrix C^{-1}.*

Step 1: *Use the **DETSGCM** algorithm to check the nonsingularity of the matrix C. If the matrix C is singular then OUTPUT (The matrix C is singular), Stop.*

Step 2: *If $\mu_i = 0$ for any $i = 1, 2, ..., n$, set $\mu_i = t$ (t is just a symbolic name).*

Step 3: *If $\alpha_i = 0$ for any $i = 1, 2, ..., n-2$, set $\alpha_i = t$.*

Step 4: *For $i = 1, 2, ..., n$, compute and simplify the components $S_{i,n}$ and $S_{i,n-1}$ of the columns Col_j, $j = n$, and $n-1$, respectively, by using (2.11)-(2.15).*

Step 5: *For $i = 1, 2, ..., n$, compute and simplify the components $S_{i,n-2}$ by using (2.16).*

Step 6: *For $j = n-3, n-4, ..., 1$, do*

For $i = 1, 2, ..., n$, *do*

Compute and simplify the components $S_{i,j}$ *by using (2.17).*

End do

End do

Step 7: *Substitute the actual value* $t = 0$ *in all expressions to obtain the elements,* $S_{i,j}$, $i, j = 1, 2, ..., n$.

The symbolic Algorithm 2.2 will be referred to as the **SGCMINV**. The computational cost of the **SGCMINV** algorithm is $7n^2 - 5n - 11$ operations. In[8], recurrence relations for the rows of an inverse comrade matrix were presented but it was assumed that $\alpha_i \neq 0$ for $i = 1, 2, ..., n-1$ and the first row of an inverse comrade matrix is known. The computational cost of this method is $O(n^3)$ operations. On the other hand, if $a_i = 0, \quad i = 3, 4, ..., n$, the Algorithm 2.3 in [7] will be special case of the **SGCMINV** algorithm.

3 ILLUSTRATIVE EXAMPLES

In this section we give three examples for illustration.

Example 3.1. Consider the 7×7 matrix C given by

$$C = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{3}{5} & -\frac{4}{5} & \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{4}{2} & -\frac{5}{2} & \frac{1}{2} \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{3}{3} \end{pmatrix}.$$

By applying the **SGCMINV** algorithm:

- $\mu = (-\frac{1}{2}, -\frac{1}{5}, -\frac{1}{3}, -\frac{1}{2}, -\frac{4}{3})$,
  $Det(C) = -\frac{1}{45}$(Step 1).

- $C^{-1} = \begin{pmatrix} -24 & -\frac{75}{4} & -\frac{39}{4} & -\frac{3}{2} & -34 \\ -22 & -\frac{75}{4} & -\frac{39}{4} & -\frac{3}{2} & -34 \\ -16 & -\frac{55}{4} & -\frac{39}{4} & -\frac{3}{2} & -34 \\ -10 & -\frac{35}{4} & -\frac{27}{4} & -\frac{3}{2} & -34 \\ 14 & \frac{45}{4} & \frac{21}{4} & \frac{1}{2} & -34 \end{pmatrix}$ (Steps 2-7).

Example 3.2. Consider the 4×4 matrix C given by

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & -1 & 5 & 0 \\ 0 & 3 & 1 & 2 \\ -1 & 1 & 5 & 3 \end{pmatrix}.$$

By applying the **SGCMINV** algorithm:

- $\mu = (t, -\frac{t+2}{t}, \frac{2(8t+1)}{t+2}, \frac{2(7t-6)}{8t+1})$,
  $Det(C) = \Big[-4(7t-6) \Big]_{t=0} = 24$(Step 1).

- $C^{-1} = \begin{pmatrix} \frac{7}{7t-6} & -\frac{7}{28t-24} & -\frac{15}{28t-24} & \frac{5}{14t-12} \\ -\frac{6}{7t-6} & \frac{7t}{4(7t-6)} & \frac{15t}{4(7t-6)} & -\frac{5t}{2(7t-6)} \\ -\frac{4}{7t-6} & \frac{7t-2}{4(7t-6)} & \frac{3(t+2)}{4(7t-6)} & -\frac{t+2}{2(7t-6)} \\ \frac{11}{7t-6} & -\frac{14t-1}{4(7t-6)} & -\frac{5(2t+3)}{4(7t-6)} & \frac{8t+1}{2(7t-6)} \end{pmatrix}_{t=0}$
  $= \begin{pmatrix} -\frac{7}{6} & \frac{7}{24} & \frac{3}{8} & -\frac{5}{12} \\ 1 & 0 & 0 & 0 \\ \frac{2}{3} & \frac{1}{12} & -\frac{1}{4} & \frac{1}{6} \\ -\frac{11}{6} & -\frac{1}{24} & \frac{5}{8} & -\frac{1}{12} \end{pmatrix}$ (Steps 2-7).

Example 3.3. We consider the following $n \times n$ comrade matrix in order to demonstrate the efficiency of the **SGCMINV** algorithm.

$$C = \begin{pmatrix} -3/2 & 1/2 & 0 & \cdots & \cdots & 0 \\ 1/2 & -3/2 & 1/2 & 0 & \ddots & 0 \\ 0 & 1/2 & -3/2 & 1/2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1/2 & -3/2 & 1/2 \\ -1/2 & -1/2 & \cdots & -1/2 & (1-1)/2 & (-3-1)/2 \end{pmatrix}.$$

We use the **SGCMINV** algorithm to compute the inverse of the comrade matrix C. Results are given in Table 1 in which $\varepsilon = ||C^{-1}_{\mathbf{exact}} - C^{-1}_{\mathbf{SGCMINV}}||_\infty$.

Table 1: Error and CPU time(s)

	n		
	50	100	500
ϵ	1.1631×10^{-9}	1.1215×10^{-9}	1.6078×10^{-9}
CPU time(s)	0.421(using inverse function in Maple 13.0)	3.448	336.338
	0.109(Using our Algorithm)	0.609	33.899

4 CONCLUSIONS

In this work new symbolic computational algorithms have been developed for computing the determinant and inverse of a general comrade matrix. The algorithms are reliable, computationally efficient and remove the cases where the numeric algorithms break down.

References

[1] J. Maroulas, S. Barnett, Applications of the comrade matrix to linear multivariable systems theory, INT. J. Control, 28(1978) 129-145.

[2] N. Aris, A. Abd Rahman, Computing the greatest common divisor of polynomials using the comrade matrix, Lecture Notes Series in Computer Science, 5081(2008)87-96.

[3] S. Barnett, Division of generalized polynomials using the comrade matrix,Linear Algebra and its Applications, 60(1984)159-175.

[4] A.A. Karawia, Two algorithms for solving comrade linear systems, Appl. Math. Comput. 189 (2007)291-297.

[5] T. Sogabe, Numerical algorithms for solving comrade linear systems based on tridiagonal solvers, Appl. Math. Comput. 198 (2008)117-122.

[6] A.A. Karawia, Symbolic Algorithm for Solving Comrade Linear systems Based on Modified Stair-Diagonal Approach, Appl. Math. Lett, accepted.

[7] M. El-Mikkawy, E. Rahmo, A new recursive algorithm for inverting general tridiagonal and anti-tridiagonal matrices, Appl. Math. Comput. 204(2008)368-372.

[8] S. Barnett, Inverse Comrade Matrices, Linear and Multilinear Algebra, 22(1988)325-333.