Modified Cramer’s Rule and its Application to Solve Linear Systems in WZ Factorization

Olayiwola Babarinsa* and Hailiza Kamarulhaili

School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia
Department of Mathematical Sciences, Federal University Lokoja, P.M.B 1154, Nigeria

*Corresponding author: olayiwola.babarinsa@fulokoja.edu.ng

Article history
Received: 2 May 2018
Received in revised form: 27 August 2018
Accepted: 15 November 2018
Published on line: 1 April 2019

Abstract The proposed modified methods of Cramer’s rule consider the column vector as well as the coefficient matrix concurrently in the linear system. The modified methods can be applied since Cramer’s rule is typically known for solving the linear systems in WZ factorization to yield Z-matrix. Then, we presented our results to show that there is no tangible difference in performance time between Cramer’s rule and the modified methods in the factorization from improved versions of MATLAB. Additionally, the Frobenius norm of the modified methods in the factorization is better than using Cramer’s rule irrespective of the version of MATLAB used.

Keywords Linear systems; Cramer’s rule; WZ factorization; Z-matrix; Frobenius norm

Mathematics Subject Classification 5A06, 15A23, 15B99.

1 Introduction

If a linear system is defined by

\[ Bx = c, \]

then the unique solutions \( x = (x_1, x_2, \ldots, x_n)^T \), where \( B \) is the coefficient matrix of the system and \( c \) the column vector, to equation (1) can be obtained from an old method called Cramer’s rule [1]. Cramer’s rule has many drawbacks: fails when the coefficient matrix is singular, requires \((n+1)\) determinants in its computation and has high computational time [2]. Cramer’s rule and Gaussian elimination (GE) requires about the same amount of arithmetic for finding the solution of \( 2 \times 2 \) linear systems. Moler [3] expressed that Cramer’s rule is unsatisfactory even for \( 2 \times 2 \) linear systems because of round off errors. However, Dunham [4] gives a counter example to the statement to show that Cramer’s rule is satisfactory. Thus, accurate methods to evaluate determinants make Cramer’s rule numerically stable [5]. Notwithstanding its high computational complexity, Cramer’s rule is truly intriguing and it is of hypothetical significance for solving linear systems [6]. One of the advantages of Cramer’s rule is its application in...
WZ factorization or quadrant interlocking factorization (QIF) to check if the matrix being factorized has non-singular inner submatrix (centro-nonsingular) and to solve its linear systems [7, 8].

WZ factorization is first proposed by Evans and Hatzopoulos [9]. During WZ factorization of non-singular matrix \( B \), Z-matrix exists together with W-matrix [10], such that

\[
B = WZ. \tag{2}
\]

The matrix norm of WZ factorization is the Frobenius norm of matrix \( B \) given as

\[
||B||_F = ||B - WZ||.
\]

WZ factorization is known for the adaptability of its direct method for solving linear systems on shared memory parallel computers with many integrated core, see [11–14] and the references therein. WZ factorization has been shown to be better than the GE and LU factorization irrespective of the number of processors used and is also better on Intel processors than AMD processors, see [15, 16]. WZ factorization is extensively applied in finding the numerical solutions of Markov chains [17, 18]. Though, the efficiency of WZ factorization depends on an efficacious use of the memory echelon because computational cost often relies on both the total number of arithmetic operations used and the data transferring time between different memory levels [19]. For the WZ factorization, we compute \( w^{(k)}_{i,k} \) and \( w^{(k)}_{i,n-k+1} \) from equation (3) by solving its \( 2 \times 2 \) linear systems via Cramer’s rule for every update of matrix \( B \),

\[
\begin{cases}
\zeta^{(k-1)}_{k,k} w^{(k)}_{i,k} + \zeta^{(k-1)}_{n-k+1,k} w^{(k)}_{i,n-k+1} = -\zeta^{(k-1)}_{i,k} \\
\zeta^{(k-1)}_{k,n-k+1} w^{(k)}_{i,k} + \zeta^{(k-1)}_{n-k+1,n-k+1} w^{(k)}_{i,n-k+1} = -\zeta^{(k-1)}_{i,n-k+1},
\end{cases} \tag{3}
\]

where \( k = 1, 2, \ldots, |\frac{n}{2}| \); \( i = k + 1, \ldots, n - k \). Then, we update matrix \( B \) for every computed \( i \)th to the \( (n - 1) \)th element of the \((i - 1)\)th and \((n - i + 1)\)th column of W-matrix, see [20].

In Section 2, we propose two modified methods from Cramer’s rule and show how they are equivalent to classical Cramer’s rule. While Section 3 applies the proposed methods of Cramer’s rule in WZ factorization. To obtain the matrix norm of the factorization, we evaluate the Frobenius norm of the applied modified methods and Cramer’s rule. Furthermore, the performance time of the proposed methods and Cramer’s rule in the factorization were compared. Due to the lack of parallel computer or mesh multiprocessors, the MATLAB codes application of this article are limited to Intel processor (Core i7-4600U 2.1GHz).

2 Modified Cramer’s Rule

**Theorem 1 (Cramer’s rule)** Let \( Bx = c \) be an \( n \times n \) system of linear equation and \( B \) an \( n \times n \) non-singular matrix, then the unique solution \( x = (x_1, x_2, \ldots, x_n)^T \) to the linear system is given by

\[
x_i = \frac{\det(B_{i;c})}{\det(B)}.
\] \tag{4}

Where \( B_{i;c} \) is the matrix obtained from \( B \) by substituting the column vector \( c \) to the \( i \)th column of \( B \), for \( i = 1, 2, \ldots, n \).
It is a well-established theorem that if the \( i \)th column of matrix \( B \) is the sum or difference of the \( i \)th column of matrix \( C \) and the \( i \)th column of matrix \( D \) and all other columns in \( C \) and \( D \) are equal to the corresponding columns in \( B \). Then

\[
det(B) = det(C) \pm det(D).
\] (5)

In addition, if for \( i = 1, 2, ..., n \), the column matrix \( B \) is replaced with the row sum of its matrix to obtain a new matrix \( B^{\alpha_i} \) with all other columns in \( B \) and \( B^{\alpha_i} \) remain the same, then the determinant of the matrix and the obtained matrix are equal [21]. That is,

\[
det(B) = det(B^{\alpha_i}).
\] (6)

Now, we can deduce that if column vector \( c \) is added to or subtracted from the \( i \)th column of matrix \( B^{\alpha_i} \) (i.e the \( i \)th column of matrix \( B \) where its row sum replaced), then we can re-write equation (5) via (6) as

\[
det(B^{\alpha_i}_{i+c}) = det(B^{\alpha_i}) \pm det(B^{\alpha_i}_{i|c}),
\] (7)

where \( B^{\alpha_i}_{i+c} \) is the matrix obtained from \( B^{\alpha_i} \) by adding (or subtracting) the column vector \( c \) to (or from) the \( i \)th column of \( B^{\alpha_i} \), \( B^{\alpha_i}_{i|c} \) is the matrix obtained from \( B^{\alpha_i} \) by substituting column vector \( c \) to the \( i \)th column of \( B^{\alpha_i} \). While \( B^{\alpha_i} \) is the matrix obtained from matrix \( B \), where the \( i \)th column of matrix \( B \) is replaced by its row sum. It is important to note note that if

\[
det(B) = det(B^{\alpha_i}),
\]

then

\[
det(B_{i|c}) = det(B^{\alpha_i}_{i|c}).
\] (8)

2.1 Method I

**Corollary 1** Let \( Bx = c \) be an \( n \times n \) system of linear equation and \( B \) a square matrix of \( x \), then the \( i \)th entry \( x_i \) of the unique solution \( x = (x_1, x_2, ..., x_n)^T \) to the linear system is given by

\[
x_i = \frac{\det(B^{\alpha_i}_{i+c})}{\det(B^{\alpha_i})} - 1,
\] (9)

where \( B^{\alpha_i}_{i+c} \) is the matrix obtained from \( B^{\alpha_i} \) by adding the column vector \( c \) to the \( i \)th column of \( B^{\alpha_i} \) and \( B^{\alpha_i}_{i|c} \) is the matrix obtained from \( B \) with its \( i \)th column being replaced by the row sum of \( B \), for \( i = 1, 2, ..., n \).

**Proof** First, we assume that matrix \( B \) is non-singular. Now, we consider only the positive part of equation (7) to prove Corollary 1. We have

\[
det(B^{\alpha_i}_{i+c}) = det(B^{\alpha_i}) + det(B^{\alpha_i}_{i|c}).
\] (10)

Then, we solve for \( det(B^{\alpha_i}_{i|c}) \) from equation (10) and substitute it in equation (8) to have

\[
det(B_{i|c}) = det(B^{\alpha_i}_{i+c}) - det(B^{\alpha_i}).
\] (11)

Now, substitute equation (11) and equation (6) in equation (4) to get

\[
x_i = \frac{det(B^{\alpha_i}_{i+c}) - det(B^{\alpha_i})}{det(B^{\alpha_i})} = \frac{det(B^{\alpha_i}_{i+c})}{det(B^{\alpha_i})} - 1.
\]
2.2 Method II

Corollary 2 Let $Bx = c$ be an $n \times n$ system of linear equation and $B$ a square matrix of $x$, then the $i$th entry $x_i$ of the unique solution $x = (x_1, x_2, ..., x_n)^T$ to the linear system is given by

$$x_i = 1 - \frac{\det(B_{i-c})}{\det(B_{i-c})},$$

(12)

where $B_{i-c}$ is the matrix obtained from $B_i$ by subtracting the column vector $c$ from the $i$th column of $B_i$ and $B_i$ is the matrix obtained from $B$ with its $i$th column being replaced by the row sum of $B$ for $i = 1, 2, ..., n$.

Proof Now, we consider the negative part of equation (7) based on Corollary 2 assumption to have

$$\det(B_{i-c}) = \det(B_i) - \det(B_{i|c}).$$

(13)

Then, we solve for $\det(B_{i|c})$ from equation (13) and substitute it in equation (8) to get

$$\det(B_{i|c}) = \det(B_i) - \det(B_{i-c}).$$

(14)

Then, we substitute equation (14) and equation (6) in equation (4) to have

$$x_i = \frac{\det(B_i) - \det(B_{i-c})}{\det(B_i)} = 1 - \frac{\det(B_{i-c})}{\det(B_i)}.$$

We give the steps for implementing Method I and Method II in Algorithm 1, while their MATLAB codes are provided in Listings 1 and 2 respectively.

Listing 1: MATLAB Code of Method I

```matlab
function x=methodI(B,c)
B=input('matrix B =');
c=input('vector c =');
n=size(B,1);
m=size(B,2);
if n˜=m
    Error('The matrix is not square.');
x=[];
else
    if det(B)˜=0
        x=zeros(n,1);
        k1=sum(B,2);
        k2=k1+c;
        for j=1:n
            if j˜=1 & j˜=n
                Bc=[B(:,1:j-1) k2 B(:,j+1:n)];
                D=[B(:,1:j-1) k1 B(:,j+1:n)];
            end
        end
    end
end
```
Algorithm 1 Modified Cramer’s rule algorithm

1: **procedure** MODIFIED CRAMER’S RULE
2: \( B \leftarrow n \times n \) coefficient matrix
3: \( c \leftarrow \) column vector
4: \( x_i \leftarrow \) solutions of linear system
5: **for** \( i \) **do**
6: \( D \leftarrow \) row sum of \( B \)
7: \( E \leftarrow D \pm c. \)
8: \( F_i \leftarrow \) replace \( i \)th column of \( B \) with \( E \).
9: \( S_i \leftarrow \) replace \( i \)th column of \( B \) with \( D \).
10: \( \text{det}(S) \leftarrow \) determinant of \( S_i \).
11: \( \text{det}(F) \leftarrow \) determinant of \( F_i \).
12: **if** \( E \leftarrow D + c \) **then**
13: \( x_i \leftarrow 1 - (\text{det}(F)/\text{det}(S)) \)
14: **else**
15: \( x_i \leftarrow (\text{det}(F)/\text{det}(S)) - 1 \)
16: **end if**
17: **end for**
18: **end procedure**

Listing 2: MATLAB Code of Method II

```
function x=methodII(B,c)
B=input('matrix B =');
c=input('vector c =');
n=size(B,1);
m=size(B,2);
if n~=m
    Error('The matrix is not square.');
x=[];
else
    if det(B)~=0
```
Proposition 1 Let \( Bx = c \) be an \( n \times n \) system of linear equation, where \( B \) is an \( n \times n \) non-singular matrix of \( x \) for the distinct solution of \( x = (x_1, x_2, ..., x_n)^T \) and \( c \) the column vector.

If \( x_i = 1 - \frac{\det(B_{i-1}^{\alpha i})}{\det(B^{\alpha i})} \) when the column vector \( c \) is subtracted from the column of matrix \( B^{\alpha i} \) and \( x_i = \frac{\det(B_{i+1}^{\alpha i})}{\det(B^{\alpha i})} - 1 \) when the column vector \( c \) is added to the column of matrix \( B^{\alpha i} \). Then

\[
1 - \frac{\det(B_{i-c}^{\alpha i})}{\det(B^{\alpha i})} = \frac{\det(B_{i|c})}{\det(B)} = \frac{\det(B_{i+c}^{\alpha i})}{\det(B^{\alpha i})} - 1,
\]

where \( B_{i|c} \) is the matrix obtained from \( B \) by substituting the column vector \( c \) to the \( i \)th column of \( B \) and \( B^{\alpha i} \) is the matrix obtained from \( B \) from its \( i \)th column being replaced by the row sum of \( B \) for \( i = 1, 2, ..., n \).

Proof We begin by substituting equation (10) in equation (9) of Corollary 1 to obtain

\[
x_i = \frac{\det(B^{\alpha i}) + \det(B_{i|c}^{\alpha i})}{\det(B^{\alpha i})} - 1.
\]

Substitute equation (6) and equation (8) in equation (16) to have Cramer’s rule

\[
x_i = \frac{\det(B_{i|c})}{\det(B)}.
\]

Similarly, substitute equation (13) in equation (12) to get

\[
x_i = 1 - \frac{\det(B^{\alpha i}) - \det(B_{i|c}^{\alpha i})}{\det(B^{\alpha i})}.
\]
We will obtain Cramer’s rule if equation (6) and equation (8) are substituted in equation (18). Thus, equations (16) and (18) are equivalent.

It is apparent that Cramer’s rule has better computational time than its modifications due to less computation operations. The modified methods may only be compared with Cramer’s rule, such as their residual error measurements, but our objective is to apply the methods in WZ factorization. Moreover, the advantage of these methods like Cramer’s rule over direct methods (such as GE or LU decomposition) is that they indicate if a system is incompatible or indeterminate without completely solving the systems. This specific advantage is useful in solving the $2 \times 2$ linear systems in WZ factorization.

3 Application of Modified Cramer’s Rule in WZ Factorization

The MATLAB code to compute the elements of W-matrix and Z-matrix is given in [20]. However, for future research of interested readers, we give the complete MATLAB code of WZ factorization in Listing 3.

Listing 3: MATLAB Code for WZ Factorization

```matlab
function Z = WZfactorization(B,W,Z)
    % step of elimination - from B to Z
    B=input('matrix B =');
    n = size(B, 1);
    W = zeros(n);
    for k = 1:ceil((n-1)/2)
        k2 = n - k + 1;
        determinant = B(k,k) * B(k2,k2) - B(k2,k) * B(k,k2);
        if determinant == 0
            exitflag = 0;
            for i1 = k:k2
                for i2 = i1:k2
                    determinant = B(i1,k) * B(i2,k2) - B(i2,k) * B(i1,k2);
                    if determinant ~= 0
                        tmp = B(i1,k:k2);
                        B(i1,k:k2) = B(k,k:k2);
                        B(k,k:k2) = tmp;
                        tmp = B(i2,k:k2);
                        B(i2,k:k2) = B(k2,k:k2);
                        B(k2,k:k2) = tmp;
                        exitflag = 1;
                        break
                    end % end if determinant ~= 0
                end % end of i2
            end % end of i1
        end % end if determinant == 0
    end % end of k
    Z = B;
    return
end % end if determinant == 0
% finding elements of W
W(k+1:k2-1,k)=(B(k2,k2)*B(k+1:k2-1,k)-B(k2,k)*B(k+1:k2-1,k2))/determinant;
```
To apply the modified methods of Cramer’s rule in $WZ$ factorization, we begin by expressing equation (3) like equation (1) as

\[
\begin{bmatrix}
  z_{k,k}^{(k-1)} \\
  z_{k,n-k+1}^{(k-1)} \\
  z_{n-k+1,n-k+1}^{(k-1)}
\end{bmatrix}
\begin{bmatrix}
  w_{i,k}^{(k)} \\
  w_{i,n-k+1}^{(k)}
\end{bmatrix}
= \begin{bmatrix}
  z_{i,k}^{(k-1)} \\
  z_{i,n-k+1}^{(k-1)}
\end{bmatrix}
\] (19)

and then compute $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ using method I and method II respectively. To do this, we first apply Corollary 1 to compute $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ in equation (19) as

\[
w_{i,k}^{(k)} = \frac{\text{det} \left( \begin{bmatrix} w_{i,k}^{(k)} \\ 1+c \end{bmatrix} \right)^{\alpha_1}}{\text{det} (z^{\alpha_1})} - 1 \quad \text{and} \quad w_{i,n-k+1}^{(k)} = \frac{\text{det} \left( \begin{bmatrix} w_{i,n-k+1}^{(k)} \\ 2+c \end{bmatrix} \right)^{\alpha_2}}{\text{det} (z^{\alpha_2})} - 1, \tag{20}
\]

where

\[
\begin{align*}
\text{det} (z^{\alpha_1}) &= -z_{n-k+1,k}^{(k-1)}z_{k,k}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)}z_{k,k}^{(k-1)} \\
\text{det} \left( \begin{bmatrix} w_{i,k}^{(k)} \\ 1+c \end{bmatrix} \right)^{\alpha_1} &= z_{n-k+1,k}^{(k-1)}z_{i,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)}z_{i,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)}z_{i,k}^{(k-1)} \\
\text{det} \left( \begin{bmatrix} w_{i,n-k+1}^{(k)} \\ 2+c \end{bmatrix} \right)^{\alpha_2} &= z_{k,k}^{(k-1)}z_{k,n-k+1}^{(k-1)} - z_{i,n-k+1}^{(k-1)}z_{k,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)}z_{k,n-k+1}^{(k-1)}.
\end{align*}
\]

The $W^*$-matrix obtained from equation (20) will be referred to as $W^{m_1}$-matrix and its counterpart $Z$-matrix as $Z^{m_1}$-matrix. The $W^{m_1}Z^{m_1}$ factorization is the factorization obtained from using Corollary 1. For the MATLAB code of $W^{m_1}Z^{m_1}$ factorization, we replace line 32 and line 33 in Listing 3 with line 2 to line 5 of Listing 4.

Listing 4: MATLAB Code for $W^{m_1}Z^{m_1}$ Factorization

```matlab
% finding elements of W
W(k+1:k2-1,k2) = (B(k,k) * B(k+1:k2-1,k2) - B(k,k) * B(k+1:k2-1,k));
end
```
Furthermore, if we apply Corollary 2 to compute $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ in equation (19) then

$$w_{i,k}^{(k)} = 1 - \frac{\text{det} \left( \begin{bmatrix} w_{i,k}^{(k)} \end{bmatrix} \right)_{1-c}^{{a_1}}} {\text{det} \left( z^{a_1} \right)}$$

and

$$w_{i,n-k+1}^{(k)} = 1 - \frac{\text{det} \left( \begin{bmatrix} w_{i,n-k+1}^{(k)} \end{bmatrix} \right)_{2-c}^{{a_2}}} {\text{det} \left( z^{a_2} \right)}$$

where

$$\text{det} \left( \begin{bmatrix} w_{i,k}^{(k)} \end{bmatrix} \right)_{1-c}^{{a_1}} = -z_{n-k+1,k}^{(k-1)} z_{k,k}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)}$$

$$\text{det} \left( \begin{bmatrix} w_{i,k}^{(k)} \end{bmatrix} \right)_{2-c}^{{a_2}} = z_{i,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)}$$

The $W^*$-matrix obtained from equation (21) will be referred to as $W^{m_2}$-matrix and its $Z$-matrix as $Z^{m_2}$-matrix. The $W^{m_2}Z^{m_2}$ factorization is the factorization obtained from using Corollary 2. For the MATLAB code of $W^{m_2}Z^{m_2}$ factorization, we replace line 32 and line 33 in Listing 3 with line 32 to line 5 of Listing 5.

Listing 5: MATLAB Code for $W^{m_2}Z^{m_2}$ Factorization

```
1 %finding elements of W
2 W(k+1:k2-1,k)=1−((B(k2,k2)∗B(k+1:k2−1,k)+B(k2,k2)∗B(k,k))
3 −B(k2,k)∗B(k+1:k2−1,k2)−B(k2,k)∗B(k,k))/\text{determinant};
4 W(k+1:k2−1,k2)=1−(B(k1:k2−1,k2)∗B(k,k)+B(k2,k2)∗B(k,k))
5 −B(k2,k)∗B(k+1:k2−1,k)−B(k2,k)∗B(k,k))/\text{determinant};
```

The algorithms of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization are implemented on Intel processor (Core i7-4600U 2.1GHz) with standard hardware. Then, we investigate the performance time of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization on different versions of MATLAB (R2013b, R2015b and R2017b) respectively and the results were recorded in Table 1.

In Figure 1, 2 and 3, the performance time of $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization are similar but higher than performance time of $WH$ factorization for all the versions of MATLAB used. However, the performance time of $WH$, $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization reduces as the versions of MATLAB improve. The performance time of $WH$, $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization decreases about 16% for MATLAB R2015b and decreases about 32% for MATLAB R2017b. Figure 3 shows the performance time of $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization...
Table 1: Performance Time of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>$WZ$</th>
<th>$W^{m_1}Z^{m_1}$</th>
<th>$W^{m_2}Z^{m_2}$</th>
<th>$WZ$</th>
<th>$W^{m_1}Z^{m_1}$</th>
<th>$W^{m_2}Z^{m_2}$</th>
<th>$WZ$</th>
<th>$W^{m_1}Z^{m_1}$</th>
<th>$W^{m_2}Z^{m_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$500 \times 500$</td>
<td>7.26</td>
<td>9.65</td>
<td>9.45</td>
<td>5.29</td>
<td>6.56</td>
<td>7.06</td>
<td>2.25</td>
<td>2.76</td>
<td>3.15</td>
</tr>
<tr>
<td>$1000 \times 1000$</td>
<td>19.95</td>
<td>28.24</td>
<td>27.45</td>
<td>17.01</td>
<td>19.51</td>
<td>18.93</td>
<td>14.28</td>
<td>13.88</td>
<td>14.26</td>
</tr>
<tr>
<td>$1500 \times 1500$</td>
<td>48.93</td>
<td>63.98</td>
<td>70.26</td>
<td>41.43</td>
<td>42.56</td>
<td>41.91</td>
<td>40.12</td>
<td>40.40</td>
<td>40.21</td>
</tr>
<tr>
<td>$2000 \times 2000$</td>
<td>135.84</td>
<td>198.15</td>
<td>205.88</td>
<td>117.33</td>
<td>123.84</td>
<td>127.10</td>
<td>88.32</td>
<td>89.66</td>
<td>86.22</td>
</tr>
<tr>
<td>$2500 \times 2500$</td>
<td>271.03</td>
<td>342.54</td>
<td>351.46</td>
<td>226.01</td>
<td>249.14</td>
<td>253.93</td>
<td>163.83</td>
<td>173.24</td>
<td>181.21</td>
</tr>
<tr>
<td>$3000 \times 3000$</td>
<td>498.27</td>
<td>552.89</td>
<td>563.01</td>
<td>423.54</td>
<td>441.34</td>
<td>448.34</td>
<td>281.44</td>
<td>294.72</td>
<td>301.69</td>
</tr>
<tr>
<td>$3500 \times 3500$</td>
<td>673.90</td>
<td>749.30</td>
<td>762.91</td>
<td>593.56</td>
<td>623.16</td>
<td>631.90</td>
<td>439.30</td>
<td>458.97</td>
<td>462.56</td>
</tr>
<tr>
<td>$4000 \times 4000$</td>
<td>908.43</td>
<td>1032.40</td>
<td>1051.13</td>
<td>882.23</td>
<td>921.57</td>
<td>927.67</td>
<td>656.59</td>
<td>672.19</td>
<td>679.41</td>
</tr>
<tr>
<td>$4500 \times 4500$</td>
<td>1643.64</td>
<td>1730.84</td>
<td>1753.39</td>
<td>1394.23</td>
<td>1430.55</td>
<td>1436.01</td>
<td>998.10</td>
<td>1013.20</td>
<td>1019.08</td>
</tr>
<tr>
<td>$5000 \times 5000$</td>
<td>2239.65</td>
<td>2305.10</td>
<td>2329.12</td>
<td>1739.87</td>
<td>1809.54</td>
<td>1820.47</td>
<td>1263.74</td>
<td>1274.60</td>
<td>1281.69</td>
</tr>
</tbody>
</table>

Figure 1: Execution Time of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2013b
Figure 2: Execution Time of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2015b

Figure 3: Execution Time of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b
approaches $WH$ factorization. In spite of more arithmetic operations used in $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization as a result of applying Corollary 1 and Corollary 2 respectively, we can deduce that a better version of MATLAB in the near future will show that there would be no tangible difference in performance time between $WH$, $W^{m_1}H^{m_1}$ and the $W^{m_2}H^{m_2}$ factorization.

In Table 2, we only display our result from MATLAB R2017b because our background analysis shows that irrespective of the version of MATLAB used, the Frobenius norms of $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization are better than $WZ$ factorization, see Figure 4. More so, the Frobenius norms of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization increase as the size of their matrices increase.

Table 2: Matrix Norm of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b

<table>
<thead>
<tr>
<th>Matrix size (N)</th>
<th>$|B - WZ|$</th>
<th>$|B - W^{m_1}Z^{m_1}|$</th>
<th>$|B - W^{m_2}Z^{m_2}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 $\times$ 500</td>
<td>5.33E-14</td>
<td>4.92E-14</td>
<td>4.91E-14</td>
</tr>
<tr>
<td>1000 $\times$ 1000</td>
<td>1.18E-13</td>
<td>0.92E-13</td>
<td>0.90E-13</td>
</tr>
<tr>
<td>1500 $\times$ 1500</td>
<td>5.67E-13</td>
<td>4.76E-13</td>
<td>4.76E-13</td>
</tr>
<tr>
<td>2000 $\times$ 2000</td>
<td>7.89E-13</td>
<td>6.73E-13</td>
<td>6.71E-13</td>
</tr>
<tr>
<td>2500 $\times$ 2500</td>
<td>1.28E-12</td>
<td>0.97E-12</td>
<td>0.94E-12</td>
</tr>
<tr>
<td>3000 $\times$ 3000</td>
<td>1.39E-12</td>
<td>0.95E-12</td>
<td>0.96E-12</td>
</tr>
<tr>
<td>3500 $\times$ 3500</td>
<td>3.06E-12</td>
<td>2.36E-12</td>
<td>2.36E-12</td>
</tr>
<tr>
<td>4000 $\times$ 4000</td>
<td>3.93E-12</td>
<td>2.71E-12</td>
<td>2.73E-12</td>
</tr>
<tr>
<td>4500 $\times$ 4500</td>
<td>5.51E-12</td>
<td>4.83E-12</td>
<td>4.80E-12</td>
</tr>
<tr>
<td>5000 $\times$ 5000</td>
<td>6.52E-12</td>
<td>5.25E-12</td>
<td>5.26E-12</td>
</tr>
</tbody>
</table>

Figure 4: Matrix Norms of $WZ$, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b
4 Conclusion

Like Cramer’s rule, the modified methods may be impracticable and inappropriate for higher linear systems. However, we pointed out their advantage over Cramer’s rule in quadrant interlocking factorization even though they are theoretically equivalent. The Frobenius norm of the applied modified Cramer’s rule over classical Cramer’s rule in the factorization shows that \( W^{m_1}Z^{m_1} \) and \( W^{m_2}Z^{m_2} \) factorization would be better in numerical solution of Markov chains. Therefore, we passionately advocate that \( W^{m_1}Z^{m_1} \) and \( W^{m_2}Z^{m_2} \) factorization should be compared with \( WZ \) factorization on parallel computer or mesh multiprocessors such as Intel Xeon Phi, Sunway Taihulight or OLCF-4.

Acknowledgments

This research is funded by RU (Research University) Grant, Universiti Sains Malaysia, Grant number 1001/PMATHS/811337.

References


