EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
Vol. 13, No. 4, 2020, 1035-1054
ISSN 1307-5543 – www.ejpam.com
Published by New York Business Global

Optimized Cramer's rule in WZ factorization and applications

Olayiwola Babarinsa[1,2,*], Azfi Zaidi Mohammad Sofi[2], Mohd Asrul Hery Ibrahim[2], Hailiza Kamarulhaili[3]

[1] *Department of Mathematical Sciences, Federal University Lokoja, 1154 Kogi State, Nigeria*
[2] *Faculty of Bioengineering & Technology, Universiti Malaysia Kelantan, 16100 Kota Bharu, Kelantan, Malaysia*
[3] *School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia*

Abstract. In this paper, WZ factorization is optimized with a proposed Cramer's rule and compared with classical Cramer's rule to solve the linear systems of the factorization technique. The matrix norms and performance time of WZ factorization together with LU factorization are analyzed using sparse matrices on MATLAB via AMD and Intel processor to deduce that the optimized Cramer's rule in the factorization algorithm yields accurate results than LU factorization and conventional WZ factorization. In all, the matrix group and Schur complement for every Z_{system} (2×2 block triangular matrices from Z-matrix) are established.

2020 Mathematics Subject Classifications: 65F05, 65F35, 15A23
Key Words and Phrases: WZ factorization, LU factorization, Linear systems, Cramer's rule, MATLAB

1. Introduction

Evans and Hatzopoulos [24] first posited *WZ factorization* or quadrant interlocking factorization of nonsingular matrix. The factorization decomposes matrices into block forms which are then regrouped and solved as sub-blocks [32]. In WZ factorization of nonsingular matrix B, W-matrix (bow-tie matrix) and *Z-matrix* (hourglass matrix) - which are also known as interlocking quadrant factors of B - coexist in the form [6]

$$W = \begin{bmatrix} 1 & & & & & & & \circ \\ \bullet & 1 & & & & & & \bullet \\ \bullet & \circ & 1 & & & & \circ & \bullet \\ \bullet & \circ & \circ & 1 & & \circ & \circ & \bullet \\ \bullet & \circ & \bullet & & 1 & \bullet & \circ & \bullet \\ \bullet & \bullet & & & & 1 & \bullet & \bullet \\ \bullet & & & & & & 1 & \bullet \\ \circ & & & & & & & 1 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & \circ & \circ & \circ & \circ & \circ & \bullet & \\ & & \circ & \circ & \circ & \bullet & & \\ & & & \circ & \bullet & & & \\ & & & \bullet & \circ & & & \\ & & \bullet & \circ & \circ & \circ & & \\ & \bullet & \circ & \circ & \circ & \circ & \circ & \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

*Corresponding author.
DOI: https://doi.org/10.29020/nybg.ejpam.v13i4.3838

Email addresses: olayiwola@umk.edu.my; olayiwola.babarinsa@fulokoja.edu.ng (O. Babarinsa), azfi.ms@umk.edu.my (A.Z. Sofi), hery.i@umk.edu.my (M.A. Ibrahim), hailiza@usm.my (H. Kamarulhaili)

such that

$$B = WZ. \tag{1}$$

The factorization exists for every nonsingular matrix due to its uniqueness, often with pivoting [33, 37]. Pivoting improves the numerical stability of WZ factorization [34]. Even without pivoting or reordering, WZ factorization will not fail if the matrix is real symmetric, positive definite or diagonally dominant, see [21, 43]. The factorization has been applied in scientific computing - especially in science and engineering - see also [10, 19, 21, 25, 39]. Other variations of WZ factorization are detailed in [15, 20, 23, 36, 38], but block WZ factorization (or its Z_{system}) is discussed in [7, 9, 26]. The newest and alternative form of WZ factorization with applications is the WH factorization, see [4, 6]. In addition, the numerical accuracy $\left(-log_{10}\frac{\|B-WZ\|}{n\cdot\|B\|}\right)$ of WZ factorization depends on the matrix size but more on the matrix norms [11]. The *matrix norm* of WZ factorization is the Frobenius norm [28]. The Frobenius norm of WZ factorization from Equation (1) is given as

$$\|B-WZ\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{n}|b_{i,j} - w_{i,j}z_{i,j}|\right)^{\frac{1}{2}}. \tag{2}$$

Furthermore, WZ factorization proves to be better on Intel processors than on Advanced Micro Devices (AMD) processors [11]. Even though WZ factorization and *LU factorization* have similar computational complexity with LU factorization - $\left(\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n\right)$ and WZ factorization - $\left(\frac{2}{3}n^3 - \frac{7}{6}n - 3\right)$, the WZ factorization still shown to be better than LU factorization (except block LU factorization) irrespective of the version of MATLAB or the number of processors used [22]. However, for a uniprocessor, WZ factorization does not exhibit any advantage over LU factorization since every step performed is in serial [32]. For sparse matrices, LU and WZ factorization generate approximately similar number of nonzero elements. LU factorization relies on leading principal submatrices, whereas WZ factorization relies on nonsingular central submatrices. WZ factorization simultaneously computes two matrix elements (two columns at a time), unlike LU factorization which computes one column at a time [12]. While LU factorization performs elimination in serial with $n-1$ steps, WZ factorization executes components in parallel with $\frac{n}{2}$ steps if n is even or $\frac{n-1}{2}$ steps if n is odd. LU factorization is often known to be implemented in LAPACK library to exploit the standard software library architectures [17]. WZ factorization offers parallelization in solving both sparse and dense linear system to enhance performance using OpenMP, CUDA, BLAS or EDK HW/SW codesign architecture [1, 14]. Then, Yalamov [42] presented that WZ factorization is faster on computer with a parallel architecture than any other matrix factorization methods. Therefore, WZ factorization has the adaptability to solve linear systems on Single Instruction, Multiple Data (*SIMD*) or Multiple Instruction, Multiple Data (*MIMD*) shared memory parallel computers or mesh multiprocessors, see [3, 27, 30] and the references therein. The efficiency of WZ factorization depends on an efficacious use of the memory echelon because computational cost often relies not only on the total number of arithmetic operations used but also the data transferring time between different memory levels [9].

In WZ factorization, there are $\sum_{k=1}^{\lfloor\frac{n}{2}-1\rfloor}(n-2k)$ of 2×2 linear systems to be solved which account for the elements in W-matrix and Z-matrix, for $k = 1,2,...,\lfloor \frac{n}{2} \rfloor$. The direct solver of linear systems in WZ factorization algorithm solely depends on a classical method called *Cramer's rule*. Cramer's rule solves the 2×2 linear systems of WZ factorization under the nonsingularity constraint presumed for their determinants [8]. Though Cramer's rule is assumed to be less practical due to its setbacks, many modifications have been made on Cramer's rule to solve simple and large linear systems, see [5, 29, 41]. Due to round off errors which may become significant on problems with non-integer coefficients, Moler [35] then demonstrated that Cramer's rule is inadequate even for 2×2 linear systems. However, Dunham [18] gave a counter example of 2×2 linear system to show that Cramer's rule is sufficient. Linear systems, especially 2×2 linear equations, solved by Cramer's rule can be forward stable or backward stable depending on the conditioning of the system [31, 40]. Cramer's rule and Gaussian elimination requires about the same amount of arithmetic operations for finding the solution of 2×2 linear systems, but Cramer's rule yields a highly accuracy and stability than Gaussian elimination even with pivoting [16, 29]. For this reason, Cramer's rule has been applied to solve the linear systems in WZ factorization for over three decades. Therefore, in Section 2, we proposed a method to optimize Cramer's rule. While in Section 3, we apply the proposed method in WZ factorization on sparse matrices via MATLAB R2017b and R2019b respectively. Then, the performance time and the matrix norm of optimized Cramer's rule and classical Cramer's rule in WZ factorization and LU factorization are compared on AMD Ryzen 5 1500X and Intel Core i5-7500 processor each having four cores and 16GB RAM with standard hardware. Furthermore, we relate Schur complement and matrix group to the partition of Z-matrix into 2×2 block triangular matrices.

2. Solving simple linear systems with optimized Cramer's rule

A linear system is defined by

$$Bx = c, \tag{3}$$

where

$$det(B) \neq 0,\ \ x = [x_1, x_2, ..., x_n]^T,\ \ c = [c_1, c_2, ..., c_n]^T, \quad B \in \mathbb{R}^{n\times n},\ \ x, c \in \mathbb{R}^n.$$

Theorem 1. *[31][Cramer's rule] Let $Bx = c$ be an $n\times n$ system of linear equation and B an $n\times n$ nonsingular matrix, then the unique solution $x = [x_1, x_2, ..., x_n]^T$ 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 coefficient matrix B by substituting the column vector c to the ith column of B, for $i = 1,2,...,n$.

Let c_1 be the row sum of matrix B. If the ith column of matrix B is replaced with c_1 to obtain a new matrix $B_{i|c_1}$ with all other columns in B and $B_{i|c_1}$ remain the same, for $i = 1,2,...,n$. Then,

$$det(B) = det(B_{i|c_1}). \tag{5}$$

It is a well-established theorem that if the ith column of matrix B is the difference of the ith column of matrix D_i and the ith column of matrix E_i, and all other columns in D and E are equal to the corresponding columns in B, for $i = 1, 2, ..., n$ [2]. Then

$$det(B) = det(D) - det(E). \tag{6}$$

Corollary 1. *Let $Bx = c$ be an $n \times n$ system of linear equation and B an $n \times n$ nonsingular matrix of x, then the ith entry x_i of the unique solution $x = [x_1, x_2, ..., x_n]^T$ to the linear system is given by*

$$x_i = -\frac{det(B_{i-(c+c_1)})}{det(B)}, \tag{7}$$

where $B_{i-(c+c_1)}$ is the matrix obtained by subtracting the sum of column vector c and c_1 the row sum of the coefficient matrix from the ith column of B, for $i = 1, 2, ..., n$.

Proof. Let $c_2 = c + c_1$, where c is the column vector and c_1 the row sum of matrix B. If c_2 is subtracted from the ith column of matrix B, then we can re-write Equation (6) as

$$det(B_{i-c_2}) = det(B) - det(B_{i|c_2}). \tag{8}$$

But

$$det(B_{i|c_2}) = det(B_{i|(c+c_1)}) = det(B_{i|c}) + det(B_{i|c_1}). \tag{9}$$

Substitute Equation (5) in Equation (9) to get

$$det(B_{i|c_2}) = det(B_{i|c}) + det(B). \tag{10}$$

Therefore,

$$det(B_{i-c_2}) = det(B) - (det(B_{i|c}) + det(B)).$$

Now,

$$x_i = -\frac{det(B_{i-c_2})}{det(B)} = -\frac{det(B_{i-(c+c_1)})}{det(B)}. \tag{11}$$

□

The flowchart in Figure 1, the step by step in Algorithm 1, and the MATLAB code of the algorithm in Listing 1 show the computational steps of Corollary 1.

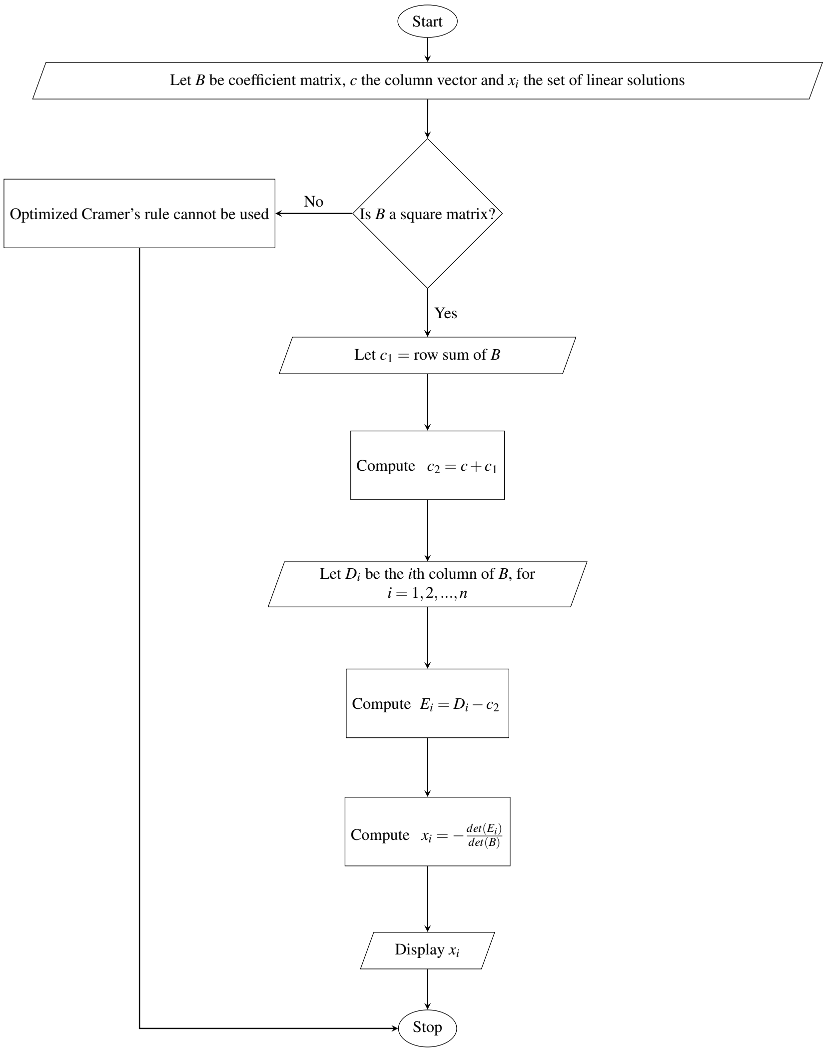


Figure 1: Flowchart of an optimized Cramer's rule

Algorithm 1 An optimized Cramer's rule

procedure
  $B \leftarrow n \times n$ coefficient matrix
  $c \leftarrow$ column vector
  $x_i \leftarrow$ solutions of linear system
  for i **do**1n
    $c_1 \leftarrow$ row sum of B
    $c_2 \leftarrow c + c_1$
    $D_i \leftarrow i$th row of B
    $E_i \leftarrow D_i - c_2$
    $det(E) \leftarrow$ determinant of E_i
    $det(B) \leftarrow$ determinant of B
    $x_i \leftarrow -(det(E)/det(B))$
  end for
end procedure

Listing 1: MATLAB code of optimized Carmer's rule.

```
function x=Optimized Cramer's rule(B,c)
B=input('matrix B =');
c=input('column vector =');
n=size(B,1);
m=size(B,2);
if n~=m
    Error('The matrix is not square.');
    x=[];
else
    detB=det(B);
    if det(B)~=0
        x=zeros(n,1);
        c1=sum(B,2);
        c2=c+c1;
        for j=1:n
            if j~=1 && j~=n
                E=[B(:,1:j-1) B(:,j)-c2 B(:,j+1:n)];
            elseif j==1
                E=[B(:,1)-c2 B(:,2:n)];
            elseif j==n
                E=[B(:,1:n-1) B(:,n)-c2];
            end
             detE=det(E);
            x(j)=-(det(E)/detB);
        end
    else
        Error('Matrix B is singular.');
         x=[];
    end
end
```

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 $x = [x_1, x_2, \ldots, x_n]^T$ and c the column vector. If $x_i = \frac{det(B_{i|c})}{det(B)}$*

and $x_i = -\frac{det(B_{i-(c+c_1)})}{det(B)}$, *then*

$$-\frac{det(B_{i-(c+c_1)})}{det(B)} = \frac{det(B_{i|c})}{det(B)},$$

where $B_{i|c}$ *is the matrix obtained from matrix B by substituting the column vector c to the ith column of B and* $B_{i-(c+c_1)}$ *is the matrix obtained by subtracting the sum of column vector c and* c_1 *the row sum of coefficient matrix from the ith column of B, for* $i = 1,2,...,n$.

Proof. We begin by substituting Equation (8) to the numerator of Equation (11) to obtain

$$\begin{aligned} x_i &= -\frac{\left[det(B) - det(B_{i|c_2})\right]}{det(B)} \\ &= -\frac{\left[det(B) - det(B_{i|c_1+c})\right]}{det(B)} \\ &= -\frac{\left[det(B) - \left(det(B_{i|c_1}) + det(B_{i|c})\right)\right]}{det(B)} \end{aligned}$$

Recall that $det(B_{i|c_1}) = det(B)$.
Thus,

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

□

Corollary 1 as well as Theorem 1 indicates if a system is inconsistent or indeterminate without completely solving the systems, unlike other direct solvers. Notwithstanding, the optimized Cramer's rule use a few more arithmetic operations than classical Cramer's rule for higher linear systems. However, based on our background analysis, the optimized Cramer's rule, especially for examples of 2×2 well and ill-conditioned linear systems lower than the relative residual measurements of Cramer's rule. This distinct advantage makes optimized Cramer's rule suitable for solving the 2×2 linear systems of WZ factorization.

3. Application of optimized Cramer's rule in WZ factorization

For the WZ factorization algorithm, we obtain the the ith to the $(n-1)$th element of the $(i-1)$th and $(n-i+1)$th column of W-matrix by computing $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ from

$$\begin{cases} z_{k,k}^{(k-1)} w_{i,k}^{(k)} + z_{n-k+1,k}^{(k-1)} w_{i,n-k+1}^{(k)} = -z_{i,k}^{(k-1)} \\ z_{k,n-k+1}^{(k-1)} w_{i,k}^{(k)} + z_{n-k+1,n-k+1}^{(k-1)} w_{i,n-k+1}^{(k)} = -z_{i,n-k+1}^{(k-1)} \end{cases} \tag{12}$$

which update the elements of Z-matrix from

$$z_{i,j}^{(k)} = z_{i,j}^{(k-1)} + w_{i,k}^{(k)} z_{k,j}^{(k-1)} + w_{i,n-k+1}^{(k)} z_{n-k+1,j}^{(k-1)} \tag{13}$$

and we then proceed similarly for the central submatrices of size $(n-2k)$ and so on, where $k = 1,2,...,\lfloor\frac{n}{2}\rfloor$, $i,j = k+1,...,n-k$ and $z_{i,j}^{(k)} \in \mathbb{R}$, see [9]. We can now re-write Equation (12) in matrix form as

$$\overbrace{\begin{bmatrix} z_{k,k}^{(k-1)} & z_{n-k+1,k}^{(k-1)} \\ z_{k,n-k+1}^{(k-1)} & z_{n-k+1,n-k+1}^{(k-1)} \end{bmatrix}}^{B} \overbrace{\begin{bmatrix} w_{i,k}^{(k)} \\ w_{i,n-k+1}^{(k)} \end{bmatrix}}^{w} = \overbrace{\begin{bmatrix} -z_{i,k}^{(k-1)} \\ -z_{i,n-k+1}^{(k-1)} \end{bmatrix}}^{c} \tag{14}$$

If we apply Theorem 1 to derive W-matrix by computing $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ (from $Bw = c$) with respect to first and second column of B from Equation (14), we will obtain

$$w_{i,k}^{(k)} = \frac{det\left(B_{1|c}\right)}{det\left(B\right)} \qquad \text{and} \qquad w_{i,n-k+1}^{(k)} = \frac{det\left(B_{2|c}\right)}{det\left(B\right)}. \tag{15}$$

The factorization obtained using Cramer's rule when we grouped and ordered the scalar operations into matrix-vector operation is the vectorized WZ factorization (VWZ factorization), see [13] for its MATLAB code.
Furthermore, if Corollary 1 is applied to compute $w_{i,k}^{(k)}$ and $w_{i,n-k+1}^{(k)}$ in Equation (14). Then,

$$\begin{aligned}
det\left(B\right) =& z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} \\
det\left(B_{1-(c+c_1)}\right) =& - z_{n-k+1,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} \\
& + z_{k,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} + z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{n-k+1,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} \\
& - z_{n-k+1,k}^{(k-1)} z_{i,n-k+1}^{(k-1)} - z_{k,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} \\
=& - z_{n-k+1,k}^{(k-1)} z_{i,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} \\
det\left(B_{2-(c+c_1)}\right) =& - z_{k,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{k,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} \\
& - z_{k,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} - z_{k,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} + z_{k,k}^{(k-1)} z_{i,n-k+1}^{(k-1)} \\
& + z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{k,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} \\
=& - z_{k,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} + z_{k,k}^{(k-1)} z_{i,n-k+1}^{(k-1)}
\end{aligned}$$

where

$$w_{i,k}^{(k)} = -\frac{det\left(B_{1-(c+c_1)}\right)}{det\left(B\right)} \qquad \text{and} \qquad w_{i,n-k+1}^{(k)} = -\frac{det\left(B_{2-(c+c_1)}\right)}{det\left(B\right)}. \tag{16}$$

The W^oZ^o factorization is the factorization obtained from using Corollary 1, where the W-matrix obtained is referred to as W^o-matrix and its Z-matrix as Z^o-matrix. The complete MATLAB code of W^oZ^ofactorization is given in Listing 2.

Listing 2: MATLAB code of W^oZ^o factorization.

```
function optimizedWZfactorization(B,W,Z)
%steps of elimination - from B to Z
```

```
3   B=input('matrix B =');
4   n = size(B, 1);
5   W = zeros(n);
6   for k = 1:ceil((n-1)/2)
7       k2 = n - k + 1;
8       determinant = B(k,k) * B(k2,k2) - B(k2,k) * B(k,k2);
9       if determinant == 0
10          exitflag = 0;
11          for i1 = k:k2
12            for i2 = i1:k2
13              determinant = B(i1,k) * B(i2,k2) - B(i2,k) * B(i1,k2);
14              if determinant ~= 0
15               disp('input matrix cannot be factorized to Z-matrix')
16                      tmp = B(i1,k:k2);
17                      B(i1,k:k2) = B(k,k:k2);
18                      B(k,k:k2) = tmp;
19                      tmp = B(i2,k:k2);
20                      B(i2,k:k2) = B(k2,k:k2);
21                      B(k2,k:k2) = tmp;
22                      exitflag = 1;
23                      break
24                  end
25              end
26          end
27          if exitflag == 0
28              Z = B;
29              return
30          end
31      end
32  %finding elements of W
33     % To compute ith to the (n-1)th element of (i-1)th column of W
34     W(k+1:k2-1,k)=-(-B(k2,k)*B(k+1:k2-1,k2)+B(k2,k2)*B(k+1:k2-1,k))/determinant;
35     % To compute ith to the (n-1)th element of (n-i+1)th column of W
36     W(k+1:k2-1,k2)=-(-B(k,k2)*B(k+1:k2-1,k)+B(k,k)*B(k+1:k2-1,k2))/determinant;
37     for m=1:n
38      W(m,m)=1;
39      W(m, n+1-m);
40    end
41    % updating B
42    B(k+1:k2-1,k) = 0;
43    B(k+1:k2-1,k2) = 0;
44    B(k+1:k2-1,k+1:k2-1) = B(k+1:k2-1,k+1:k2-1) + W(k+1:k2-1,k)* B(k,k+1:k2-1)
45                             + W(k+1:k2-1,k2) * B(k2,k+1:k2-1);
46      Z = B;
47  end
```

Besides, if there is no regrouping or ordering of scalar operations into matrix-vector operation then the factorization is a sequential WZ factorization. For the MATLAB code of WZ factorization, we replace line 32 to line 44 in Listing 2 with line 1 to line 9 of Listing 3.

Listing 3: MATLAB code of sequential WZ factorization.

```
1  % finding elements of W
2   % To compute ith to the (n-1)th element of (i-1)th column of W
3       for i=k+1:k2-1
4     W(i,k)= (B(k2,k2)*B(i,k)-B(k2,k)*B(i,k2))/determinant;
5   % To compute ith to the (n-1)th element of (n-i+1)th column of W
6     W(i,k2)=(B(k,k)*B(i,k2)-B(k,k2)*B(i,k))/determinant;
7   % updating B
```

```
    for j=k+1:k2-1
 B(i,j)= B(i,j) + W(i,k)*B(k,j) + W(i,k2)*B(k2,j);
```

For the computation and analysis, the square sparse matrices used to investigate LU, WZ, VWZ and W^oZ^o factorization, in Table 1, 2 and 3, are obtained from *The SuiteSparse Matrix Collection*. Table 1 gives the basic information about the sparse matrices, Table 2 and Table 3 illustrate the performance time and matrix norm of LU, WZ, VWZ and W^oZ^o factorization on Intel and AMD processor via MATLAB R2017b and R2019b respectively.

Table 1: Basic information of the sparse matrices.

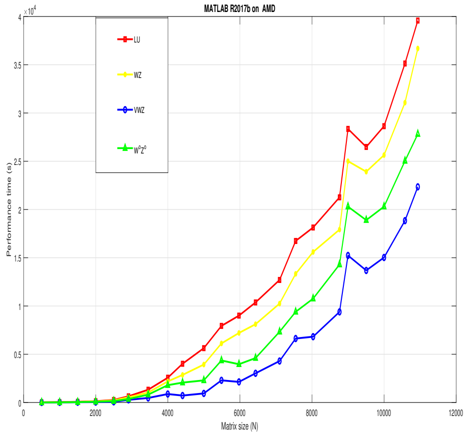
Matrix name	Matrix size	Nonzero entries	Group	Year	kind
$Trefethen\_500$	500	3,996	JGD_Trefethen	2008	Combinatorial problem
$tub1000$	1000	97,645	Bai	1994	Computational fluid dynamic problem
$comsol$	1500	7,996	Langemyr	2002	Structural problem
$olm2000$	2000	12,349	Bai	1994	Computational fluid dynamic problem
$cryg2500$	2500	174,296	Bai	1996	Materials problem
$nasa2910$	2910	66,528	Nasa	1995	Structural problem
$thermal$	3456	28,505	Brunetiere	2000	Thermal problem
$ACTIVSg2000$	4000	219,024	TAMU_SmartGridCenter	2018	Power network problem
$bcsstk28$	4410	29,600	HB	1984	Structural problem
$rdb5000$	5000	262,943	Bai	1994	Computational fluid dynamic problem
$s3rmq4m1$	5489	54,471	Cylshell	1997	Structural problem
$C-32$	5975	51,480	Schenk_IBMNA	2006	Optimization problem
$n3c6-b7$	6435	340,200	JGD_Homology	2008	Combinatorial problem
Kuu	7102	834,226	MathWorks	2006	Structural problem
fp	7548	834,226	MKS	2006	Electromagnetics problem
$bcsstk38$	8032	355,460	Boeing	1995	Structural problem
$Kaufhold$	8765	42,471	MathWorks	2006	Counter example problem
$nd3k$	9000	3,279,690	ND	2003	2D\3D problem
$nemeth19$	9506	818,302	Nemeth	1999	Quantum chemistry problem
$cryg10000$	10000	818,302	Bai	1996	Materials problem
$bundle1$	10581	818,302	Lourakis	2006	Computer graphics problem
$wing\_nodal$	10937	15,0976	DIMACS10	2000	Undirected graph

Table 2: Performance time of LU, WZ, VWZ and W^oZ^o factorization on Intel and on AMD processor.

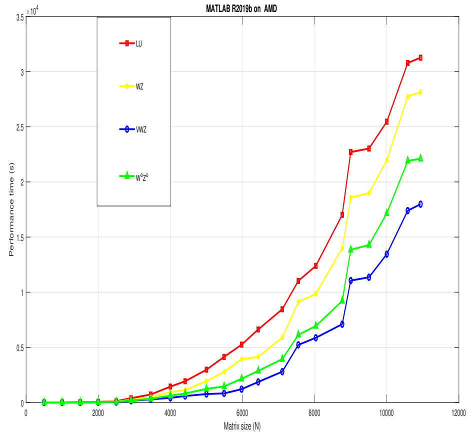
Matrix name	MATLAB R2017b								MATLAB R2019b							
	Intel				AMD				Intel				AMD			
	LU	WZ	VWZ	W^oZ^o	LU	WZ	VWZ	W^oZ^o	LU	WZ	VWZ	W^oZ^o	LU	WZ	VWZ	W^oZ^o
$Trefethen\_500$	8.62	7.46	4.01	8.52	19.46	11.42	5.90	8.52	4.75	4.56	2.75	3.78	8.62	7.46	4.01	6.22
$tub1000$	15.55	13.64	7.04	15.43	39.72	21.75	13.42	15.43	8.37	7.85	5.28	5.93	15.55	13.64	7.04	10.45
$comsol$	45.12	39.18	20.02	35.33	73.75	59.20	28.42	35.33	29.23	23.46	12.82	15.84	45.12	39.18	20.02	28.81
$olm2000$	103.72	89.82	34.51	79.05	132.64	113.14	43.41	79.05	55.15	46.64	21.57	34.73	103.72	89.82	34.51	58.73
$cryg2500$	188.78	158.3	53.22	144.15	263.13	213.23	69.87	144.15	117.23	87.15	40.86	56.55	188.78	158.3	53.22	121.18
$nasa2910$	482.57	403.15	205.18	363.12	639.57	538.32	276.52	363.12	390.42	232.05	155.65	186.42	482.57	403.15	205.18	318.87
$thermal$	929.17	857.85	312.61	813.50	1324.55	1029.13	470.45	813.50	722.43	473.42	281.64	331.53	929.17	857.85	312.61	629.45
$ACTIVSg2000$	1431.20	1125.16	506.07	1796.92	2569.15	2183.07	876.19	1796.92	1438.57	937.45	419.76	623.53	1431.20	1125.16	506.07	981.01
$bcsstk28$	2661.12	1836.67	599.37	2062.15	4013.45	2858.11	719.92	2062.15	1927.61	1123.35	586.45	813.09	2661.12	1836.67	599.37	1135.84
$rdb5000$	3459.22	2659.92	783.84	2292.48	5631.51	3932.91	939.34	2292.48	2973.75	1937.15	767.15	1236.30	3459.22	2659.92	783.84	1689.98
$s3rmq4m1$	4832.34	3832.18	950.92	4354.84	7937.43	6122.84	2295.32	4354.84	4132.98	2786.10	823.72	1454.84	4832.34	3832.18	950.92	2006.14
$C-32$	6489.65	5389.14	1273.51	3965.15	9003.58	7204.77	2134.84	3965.15	5247.42	3927.54	1207.31	2153.65	6489.65	5389.14	1273.51	2515.75
$n3c6-b7$	8126.71	6981.75	1893.01	4603.21	10369.48	8112.60	3025.18	4603.21	6621.45	4132.73	1862.56	2863.52	8126.71	6981.75	1893.01	3631.71
Kuu	10265.34	8265.48	2973.15	7331.64	12698.81	10249.27	4297.12	7331.64	8457.54	5891.14	2793.18	3936.61	10265.34	8265.48	2973.15	5713.94
fp	12823.35	10523.83	4789.81	9393.45	16739.16	13332.05	6629.94	9393.45	11023.35	9113.51	5227.07	6137.01	12823.35	10523.83	4789.81	7703.27
$bcsstk38$	14096.62	12096.5	5527.45	10762.70	18134.61	15599.49	6819.17	10762.70	12389.20	9864.25	5867.55	6935.55	14096.62	12096.5	5527.45	9124.52
$Kaufhold$	17917.45	15617.31	8534.12	14297.41	21260.27	17917.44	9389.54	14297.41	17016.72	13983.48	7098.30	9214.53	17917.45	15617.31	8534.12	12885.85
$nd3k$	24685.45	20685.48	12387.52	20297.53	28351.27	25007.71	15235.02	20297.53	22707.63	18573.15	11053.45	13847.45	24685.45	20685.48	12387.52	16813.89
$nemeth19$	25034.62	22034.34	13087.82	18897.14	26482.15	23917.64	13683.88	18897.14	23025.75	18987.47	11353.52	14282.34	25034.62	22034.34	13087.82	17147.86
$cryg10000$	28818.28	25318.19	16439.45	20297.84	28634.51	25637.24	15026.85	20297.84	25463.54	21984.45	13448.25	17157.22	28818.28	25318.19	16439.45	19884.64
$bundle1$	31724.22	28724.46	19334.35	25031.50	35128.77	31072.54	18843.52	25031.50	30784.45	27738.55	17395.71	21912.56	31724.22	28724.46	19334.35	22746.55
$wing\_nodal$	34465.18	31465.75	22135.19	27815.14	39581.06	36685.84	22348.11	27815.14	31263.52	28149.16	17981.03	22101.33	34465.18	31465.75	22135.19	26698.74

Table 3: Matrix norms of LU, WZ, VWZ and W^oZ^o factorization on MATLAB R2019b.

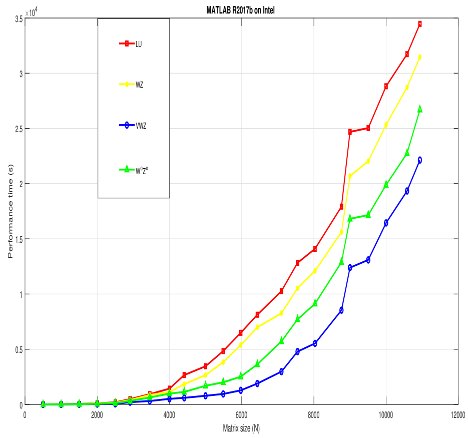
Matrix name	Frobenius norm	MATLAB R2017b								MATLAB R2019b							
		Intel				AMD				Intel				AMD			
		$\|B-LU\|$	$\|B-WZ\|$	$\|B-VWZ\|$	$\|B-W^oZ^o\|$	$\|B-LU\|$	$\|B-WZ\|$	$\|B-VWZ\|$	$\|B-W^oZ^o\|$	$\|B-LU\|$	$\|B-WZ\|$	$\|B-VWZ\|$	$\|B-W^oZ^o\|$	$\|B-LU\|$	$\|B-WZ\|$	$\|B-VWZ\|$	$\|B-W^oZ^o\|$
$Trefethen\_500$	4.39E+04	1.96E-20	1.83E-20	1.14E-20	0.98E-20	1.63E-19	1.13E-19	1.49E-19	0.82E-19	1.46E-21	0.63E-21	0.16E-21	0.08E-21	1.98E-20	1.62E-20	1.24E-20	1.01E-20
$tub1000$	3.86E+06	2.68E-20	2.31E-20	1.79E-20	1.72E-20	2.65E-19	2.37E-19	1.94E-19	1.48E-19	2.23E-21	2.01E-21	1.46E-21	1.08E-21	4.56E-20	4.23E-20	3.81E-20	3.51E-20
$comsol$	11.02	3.75E-20	3.53E-20	2.94E-20	2.19E-20	3.62E-19	3.27E-19	2.31E-19	2.24E-19	2.84E-21	2.56E-21	2.32E-21	2.12E-21	6.67E-20	6.23E-20	5.91E-20	5.57E-20
$olm2000$	7.12E+06	5.62E-20	5.40E-20	5.03E-20	4.31E-20	5.752E-18	5.43E-18	5.15E-18	4.33E-18	4.92E-21	4.11E-21	4.03E-21	3.91E-21	8.75E-20	5.43E-20	5.15E-20	4.33E-20
$cryg2500$	4.29E+04	8.51E-20	8.32E-20	7.27E-20	6.36E-20	8.54E-18	8.25E-18	7.28E-18	6.41E-18	6.72E-21	6.31E-21	6.01E-21	5.79E-21	1.54E-19	0.25E-19	0.28E-19	0.41E-19
$nasa2910$	3.60E+08	9.79E-20	9.48E-20	8.96E-20	8.48E-20	9.93E-18	9.25E-18	8.15E-18	8.73E-18	8.69E-21	8.19E-21	7.91E-21	7.75E-21	3.93E-19	2.25E-19	1.15E-19	0.73E-19
$thermal$	40.03	1.02E-19	0.98E-19	0.82E-19	0.25E-19	1.13E-18	0.78E-18	0.63E-18	0.41E-18	1.52E-20	1.30E-20	1.89E-20	0.68E-20	8.13E-19	8.78E-19	7.63E-19	6.41E-19
$ACTIVSg2000$	2.64E+04	2.57E-19	2.29E-19	1.08E-19	0.30E-19	3.25E-18	2.74E-18	1.27E-18	0.69E-18	3.27E-20	2.95E-20	1.72E-20	1.30E-20	9.25E-19	9.74E-19	8.27E-19	8.69E-19
$bcsstk28$	1.05E+09	4.92E-19	4.53E-19	2.91E-19	2.22E-19	5.19E-17	4.63E-17	2.84E-17	2.07E-17	4.52E-20	4.11E-20	3.87E-20	3.52E-20	1.39E-18	0.61E-18	0.29E-18	0.02E-18
$rdb5000$	5.28E+03	6.21E-19	6.06E-19	4.17E-19	3.19E-19	7.35E-17	6.782E-17	4.15E-17	3.26E-17	6.24E-20	5.68E-20	5.57E-20	5.02E-20	3.75E-18	3.43E-18	3.11E-18	2.76E-18
$s3rmq4m1$	1.12E+05	8.79E-19	8.53E-19	7.42E-19	6.61E-19	8.87E-17	8.24E-17	7.53E-17	6.55E-17	7.39E-20	6.87E-20	6.42E-20	5.81E-20	7.12E-18	6.93E-18	6.73E-18	6.21E-18
$C-32$	5.72E+04	8.47E-19	8.27E-19	6.68E-19	6.03E-19	8.75E-17	8.31E-17	6.82E-17	6.31E-17	9.07E-20	8.24E-20	7.98E-20	6.53E-20	8.82E-18	8.24E-18	7.82E-18	6.67E-18
$n3c6-b7$	226.89	9.98E-19	9.74E-19	8.83E-19	8.06E-19	9.92E-17	9.51E-17	8.91E-17	8.69E-17	9.58E-20	9.13E-20	8.38E-20	7.73E-20	9.13E-18	8.76E-18	8.43E-18	8.01E-18
Kuu	1.38E+03	1.13E-18	1.01E-18	0.98E-18	0.77E-18	1.93E-16	1.71E-16	1.44E-16	0.68E-16	1.76E-19	1.49E-19	1.11E-19	0.83E-19	9.97E-18	9.45E-18	8.99E-18	8.84E-18
fp	3.41E+10	2.29E-18	2.05E-18	1.59E-18	1.37E-18	2.82E-16	2.12E-16	1.82E-16	1.47E-16	2.56E-19	2.17E-19	1.76E-19	1.27E-19	0.92E-17	0.58E-17	0.35E-17	0.07E-17
$bcsstk38$	5.69E+11	3.36E-18	3.11E-18	2.71E-18	2.36E-18	4.32E-16	3.20E-16	2.98E-16	2.73E-16	3.63E-19	3.41E-19	2.91E-19	2.48E-19	1.67E-17	1.23E-17	1.11E-17	0.88E-17
$Kaufhold$	6.84E+16	4.28E-18	4.06E-18	3.13E-18	2.75E-18	5.17E-16	4.62E-16	3.47E-16	2.57E-16	5.38E-19	5.12E-19	4.82E-19	4.67E-19	3.10E-17	2.81E-17	2.52E-17	2.03E-17
$nd3k$	5.01E+03	6.12E-18	5.98E-18	5.21E-18	4.84E-18	6.64E-16	5.841E-16	5.31E-16	4.68E-16	5.60E-19	5.35E-19	5.29E-19	5.07E-19	4.81E-17	4.421E-17	4.21E-17	4.09E-17
$nemeth19$	63.50	7.02E-18	6.86E-18	6.12E-18	5.77E-18	7.78E-16	6.67E-16	6.29E-16	5.08E-16	6.82E-19	6.54E-19	6.09E-19	5.81E-19	6.23E-17	6.01E-17	5.79E-17	5.53E-17
$cryg10000$	3.42E+05	7.54E-18	7.32E-18	6.71E-18	6.16E-18	7.38E-16	7.15E-16	6.94E-16	6.74E-16	8.76E-19	8.42E-19	7.81E-19	7.37E-19	7.67E-17	7.18E-17	6.92E-17	6.68E-17
$bundle1$	2.36E+13	9.01E-18	8.83E-18	8.11E-18	7.71E-18	9.26E-16	8.21E-16	8.52E-16	7.82E-16	9.86E-19	9.26E-19	8.82E-19	8.51E-19	8.54E-17	8.24E-17	8.07E-17	7.81E-17
$wing\_nodal$	388.56	9.54E-18	9.24E-18	8.31E-18	8.08E-18	1.81E-15	1.70E-15	1.34E-15	1.11E-15	1.64E-18	1.34E-18	1.21E-18	1.08E-18	9.86E-17	9.68E-17	9.37E-17	9.12E-17



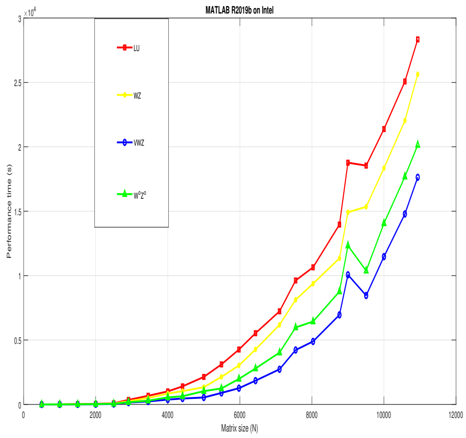
MATLAB R2017b on AMD processor.



MATLAB R2019b on AMD processor.

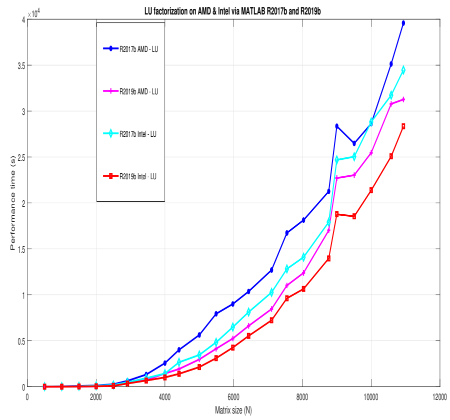


MATLAB R2017b on Intel processor.

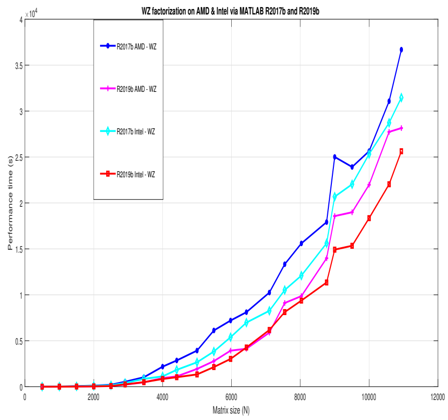


MATLAB R2019b on Intel processor.

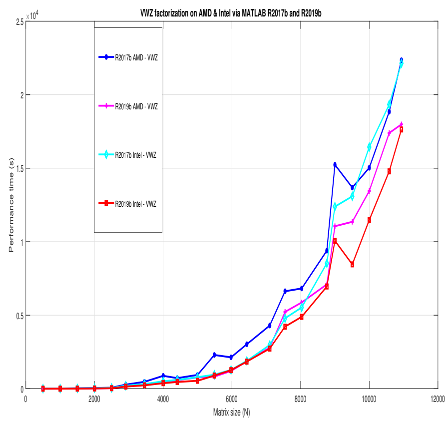
Figure 2: Performance time of LU, WZ, VWZ and W^oZ^o factorization on AMD and Intel processor via MATLAB R2017b and R2019b respectively.

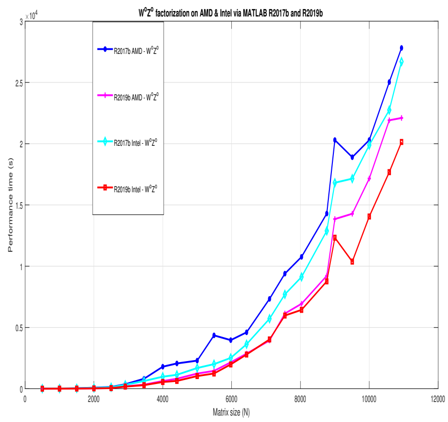


LU on AMD and Intel processor.



WZ on AMD and Intel processor.



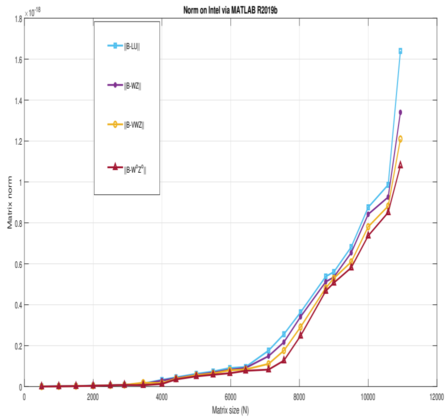


VWZ

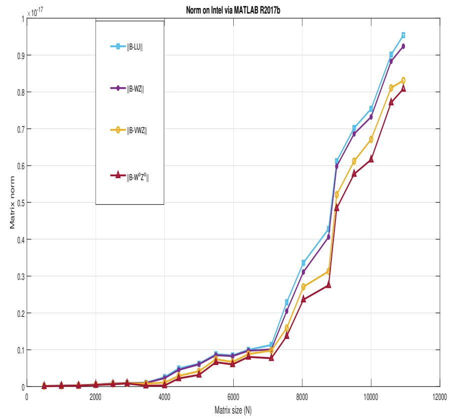
on AMD and Intel processor.

W^oZ^o on AMD and Intel processor.

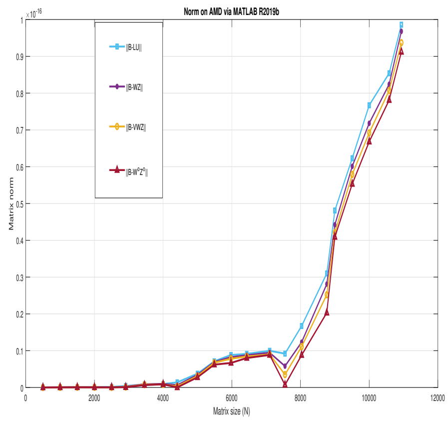
Figure 3: Combined performance time of LU, WZ, VWZ and W^oZ^o factorization on AMD and Intel processor via MATLAB R2017b and R2019b.



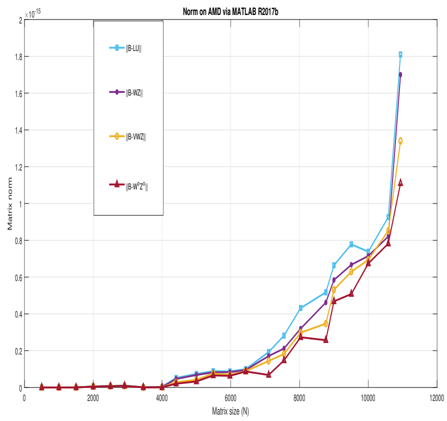
Norms on Intel via MATLAB R2019b.



Norms on Intel via MATLAB R2017b.



Norms on AMD via MATLAB R2019b.



Norms on AMD via MATLAB R2017b.

Figure 4: Matrix norms of LU, WZ, VWZ and W^oZ^o factorization on AMD and Intel processor via MATLAB R2017b and R2019b respectively.

In Figure 2, the sequential WZ factorization, on average for both MATLAB R2017b and 2019b, is about 22% faster than LU factorization on Intel processor and about 17% times on AMD processor. The most preferred factorization algorithm according to the performance time is VWZ factorization while LU factorization is the worst. However, W^oZ^o factorization in general is about 28% faster than WZ factorization and 41% than LU factorization. The performance time of W^oZ^o factorization approaches WZ factorization as the version of MATALB improves. The performance time of all the factorization algorithms increase exponentially with increase in matrix size. The version of MATLAB has minimal influence on the algorithms but the performance time significantly depends on the size of the matrix and architecture of the algorithm. Nevertheless, the higher the version of MATLAB the better the result on performance time. Kuu and $n3c6-b7$ have the highest matrix dimension difference of 667. Even though $Kaufhold$ and $nd3k$ have the least matrix dimension difference of 235, $Kaufhold$ has 1.3% nonzero elements of $nd3k$. The surge in performance time of $nd3k$ is due to the number of nonzero elements in the matrix for the factorization to utilize. $nd3k$ has more than 4% of nonzero elements while other sparse matrices in Table 1 have less than 2% nonzero elements.

Now, Figure 3 shows that the improved version of MATLAB contributes to better performance time of each algorithm. The algorithms on MATLAB R2017b spend more time in execution than on MATLAB R2019b irrespective of the type of processor used. The figure also shows that the the time to execute the algorithms via MATLAB R2017b and MATLAB R2019b on AMD processor is longer than on Intel processor.

Figure 4 displays the matrix norms for AMD and Intel on MATLAB R2017b and R2019b respectively. Our background analysis shows that the matrix norms of LU, WZ, VWZ and W^oZ^o factorization are influenced by the architecture of the algorithm used. Due to minimal round-off error, the matrix norms of W^oZ^o factorization are better than LU, WZ and VWZ factorization. The LU factorization has the worst algorithm for matrix norm. The matrix norms of all the factorization algorithms increase as the size of their matrices increase. Furthermore, the accuracy of our algorithms based on the relative residual depends more on the Frobenius norm than the matrix size. In Table 3, $comsol$, $thermal$, $n3c6-b7$, $nemeth19$ and $wing\_nodal$ have their Frobenius norms below 500 and their numerical accuracy below 25. $Kaufhold$ with 0.06% nonzero entries has the highest Frobenius norm among the given matrices.

Proposition 2. *Schur complement exists for every Z_{system}.*

Proof. For the existence of Z-matrix, the necessary and sufficient condition for WZ factorization is that matrix B must be centro-nonsingular (see [37]). First, let Z-matrix of even order being factorized from nonsingular matrix B be

$$Z = \begin{bmatrix} \alpha_{k,k} & \cdots & \cdots & \alpha_{k,\frac{n}{2}} & \vdots & \beta_{k,\frac{n}{2}+1} & \cdots & \cdots & \beta_{k,n} \\ & \ddots & Z_{1,1} & \vdots & \vdots & \vdots & Z_{1,2} & \iddots & \\ & & \ddots & \vdots & \vdots & \vdots & \iddots & & \\ & & & \alpha_{k,k} & \cdots & \beta_{k,l} & & & \\ \cdots & \cdots & \cdots & \vdots & & \vdots & \cdots & \cdots & \cdots \\ & & & \gamma_{l,k} & \cdots & \delta_{l,l} & & & \\ & & \iddots & \vdots & \vdots & \vdots & \ddots & & \\ & \iddots & Z_{2,1} & \vdots & \vdots & \vdots & Z_{2,2} & \ddots & \\ \gamma_{n,k} & \cdots & \cdots & \gamma_{n,\frac{n}{2}} & \vdots & \delta_{n,\frac{n}{2}+1} & \cdots & \cdots & \delta_{n,n} \end{bmatrix} \tag{17}$$

where $k = 1,2,...,\frac{n}{2}$; $l = n-k+1$. Then, the determinant of Z-matrix is

$$\begin{aligned} det(Z) &= det \begin{bmatrix} \alpha_{k,k} & \cdots & \beta_{k,l} \\ \vdots & & \vdots \\ \gamma_{l,k} & \cdots & \delta_{l,l} \end{bmatrix}_{1 \leq k \leq \frac{n}{2};\, l=n-k+1} \\ &= \prod_{k=1}^{\frac{n}{2}} (\delta_{l,l}\alpha_{k,l} - \gamma_{l,k}\beta_{k,l})_{l=n-k+1} \neq 0. \end{aligned} \tag{18}$$

Next, partition Equation (17) into Z_{system} of 2×2 triangular block matrices $\left([Z_{i,j}]_{i,j=1}^{2}\right)$ with each block containing $\frac{n}{2}\times\frac{n}{2}$ matrix to have

$$Z_{system} = \begin{bmatrix} Z_{1,1} & Z_{1,2} \\ Z_{2,1} & Z_{2,2} \end{bmatrix}.$$

If each 2×2 triangular block matrix is singular (i.e $Z_{1,1}Z_{2,2} = Z_{1,2}Z_{2,1}$), then Z_{system} is not invertible which contradicts Equation (18). Hence, there exists at least two nonsingular triangular block matrices in Z_{system}. If $Z_{1,1}$ is invertible as well as $Z_{2,2}$, then the Schur complement of the block $Z_{1,1}$ in Z_{system} is given as

$$Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2}. \tag{19}$$

The determinant of Equation (19) is nonsingular because $Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2}$ is a lower triangular invertible matrix (see [9]) and

$$\frac{det(Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2})}{det(Z_{1,1})} \neq 0.$$

This implies

$$det(Z_{system}) = det(Z_{1,1})det(Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2}).$$

Hence, the Schur complement of Z_{system} depends on the existence of nonsingular Z-matrix. □

Corollary 2. *Z_{system} is a matrix group of degree 2 over $\mathbb{R}$.*

Proof. Let $GL(n,\mathbb{R})$ be the matrix group of order n over $\mathbb{R}$ satisfying matrix multiplication and $M_n(\mathbb{R})$ the size of the matrix. Let the matrix group of Z_{system} be $GL_{Z_s}(2,\mathbb{R})$ of degree 2 over $\mathbb{R}$ defined as

$$GL_{Z_s}(2,\mathbb{R}) = \left\{ Z_s = \begin{bmatrix} Z_{1,1} & Z_{1,2} \\ Z_{2,1} & Z_{2,2} \end{bmatrix} : det(Z_s) = Z_{1,1}Z_{2,2} - Z_{1,2}Z_{2,1} \neq 0 \right\}$$

Since $GL_{Z_s}(2,\mathbb{R})$ is an invertible matrix based on Proposition 2, then there exists an inverse such that its identity is I_2 (that is $I_2 Z_s I_2 = Z_s$). To see that $GL_{Z_s}(2,\mathbb{R})$ is closed under matrix multiplication, we let $Z_{s(k)}, Z_{s(m)}, Z_{s(n)} \in M_2(\mathbb{R}) = Z_s$ such that $Z_{s(k)} = \{k_{i,j}\}$, $Z_{s(m)} = \{m_{i,j}\}$ and $Z_{s(n)} = \{n_{i,j}\}$. Then the associativity holds as

$$\begin{aligned}
(Z_{s(k)} * Z_{s(m)}) * Z_{s(n)} &= ((k_{i,j}) * (m_{i,j})) * (n_{i,j}) \\
&= \left(\sum_{r=1}^{2} k_{i,r}\, m_{r,j} \right) * (n_{i,j}) \\
&= \left(\sum_{s=1}^{2} \left(\sum_{r=1}^{2} k_{i,r}\, m_{r,s} \right) * (n_{s,j}) \right) \\
&= \left(\sum_{s=1}^{2} k_{i,s} * \left(\sum_{r=1}^{2} m_{s,r} * n_{r,j} \right) \right) \\
&= (k_{i,j}) * ((m_{i,j}) * (n_{i,j})) \\
&= Z_{s(k)} * (Z_{s(m)} * Z_{s(n)}).
\end{aligned}$$

□

Corollary 3. *If $GL_{Z_s}(2,\mathbb{R})$ is the matrix group of Z_{system} with degree 2 over $\mathbb{R}$, then $GL_Z(n,\mathbb{R})$ is the matrix group of Z-matrix with degree n over $\mathbb{R}$.*

Proof. Let $GL_Z(n,\mathbb{R})$ be a matrix group of Z-matrix of order n over $\mathbb{R}$ and $GL_{Z_s}(2,\mathbb{R})$ be the matrix group of Z_{system} of order 2 over $\mathbb{R}$. From Proposition 2, Z_{system} is the 2×2 triangular block matrices partitioned from Z-matrix of order n. Since Z_{system} is a matrix group, based on Corollary 2, in which Z_{system} is a subset Z-matrix. Conspicuously Z-matrix has axioms of a matrix group which is invertible and closed under matrix multiplication with property of associativity. □

4. Conclusions

The advantage of optimized Cramer's rule over classical Cramer's rule to solve 2×2 linear systems in WZ factorization is to obtain good floating points and to minimize round-off error without loss of generality in the coefficient matrix of linear systems. Although, the optimized

Cramer's rule has high performance time than VWZ factorization and low performance time than WZ factorization, the method produces better matrix norms than all other factorization algorithms, irrespective of the processors used. We passionately advocate that W^oZ^o factorization should be compared with LU, WZ and VWZ factorization on shared memory parallel computers or mesh multiprocessors.

Acknowledgements

This research is funded by Ministry of Higher Education Malaysia (FRGS RACE), Grant number R/FRGS/A0100/01258A/003/2019/00670.

References

[1] D. Ahmed and N. Askar. Parallelize and analysis lu factorization and quadrant interlocking factorization algorithm in openmp. *Journal of Duhok University*, 20(1):46–53, 2018.

[2] A. Aitken. *Determinants and matrices*. Interscience Publishers, New York, 1956.

[3] R. Asenjo, M. Ujaldon, and E. Zapata. Parallel wz factorization on mesh multiprocessors. *Microprocessing and Microprogramming*, 38(5):319–326, 1993.

[4] O. Babarinsa, M. Arif, and H. Kamarulhaili. Potential applications of hourglass matrix and its quadrant interlocking factorization. *ASM Science Journal*, 12(5S):28–38, 2019.

[5] O. Babarinsa and H. Kamarulhaili. Modified cramer's rule and its application to solve linear systems in *wz* factorization. *MATEMATIKA*, 35(1):25–38, 2018.

[6] O. Babarinsa and H. Kamarulhaili. Quadrant interlocking factorization of hourglass matrix. In *AIP Conference Proceedings*, volume 1974, pages 030009:1–9. AIP Publishing, 2018.

[7] A. Benaini and D. Laiymani. Generalized wz factorization on a reconfigurable machine. *Parallel Algorithms Appl.*, 3(4):261–269, 1994.

[8] M. Brunetti and A. Renato. Old and new proofs of cramer's rule. *Appl. Math. Sci.*, 8(133):6689–6697, 2014.

[9] B. Bylina. The block wz factorization. *J. Comput. Appl. Math.*, 331:119–132, 2018.

[10] B. Bylina and J. Bylina. Influence of preconditioning and blocking on accuracy in solving markovian models. *Int. J. Appl. Math. Comput. Sci.*, 19(2):207–217, 2009.

[11] B. Bylina and J. Bylina. Mixed precision iterative refinement techniques for the wz factorization. In *Federated Conference on Computer Science and Information Systems*, pages 425–431. IEEE, 2013.

[12] B. Bylina and J. Bylina. The wz factorization in matlab. Federated Conference on Computer Science and Information Systems, pages 561–568. IEEE, 2014.

[13] Beata Bylina and JarosLaw Bylina. Gpu-accelerated wz factorization with the use of the cublas library. In *Federated Conference on Computer Science and Information System*, pages 509–515. IEEE, 2012.

[14] Beata Bylina and JarosLaw Bylina. The parallel tiled wz factorization algorithm for multicore architectures. *Int. J. Appl. Math. Comput. Sci*, 29(2):407–419, 2019.

[15] M. Chawla and R. Khazal. A new wz factorization for parallel solution of tridiagonal systems. *Int. J. Comput. Math.*, 80(1):123–131, 2003.

[16] L. Debnath. A brief historical introduction to matrices and their applications. *Internat. J. Math. Ed. Sci. Tech.*, 45(3):360–377, 2013.

[17] J. Dongarra, M. Faverge, H. Ltaief, and P. Luszczek. Achieving numerical accuracy and high performance using recursive tile lu factorization with partial pivoting. *Concurr. Comp-Pract. E.*, 26(7):1408–1431, 2014.

[18] C. Dunham. Cramer's rule reconsidered or equilibration desirable. *ACM SIGNUM Newsletter*, 15(4):9–9, 1980.

[19] O. Efremides, M. Bekakos, and D. Evans. Implementation of the generalized wz factorization on a wavefront array processor. *Int. J. Comput. Math.*, 79(7):807–815, 2002.

[20] D. Evans. The choleski qif algorithm for solving symmetric linear systems. *Int. J. Comput. Math.*, 72(3):283–288, 1999.

[21] D. Evans. The qif singular value decomposition method. *Int. J. Comput. Math.*, 79(5):637–645, 2002.

[22] D. Evans and R. Abdullah. The parallel implicit elimination (pie) method for the solution of linear systems. *Parallel Algorithms Appl.*, 4(1):153–162, 1994.

[23] D. Evans and A. Hadjidimos. A modification of the quadrant interlocking factorisation parallel method. *Int. J. Comput. Math.*, 8(2):149–166, 1980.

[24] D. Evans and M. Hatzopoulos. A parallel linear system solver. *Int. J. Comput. Math.*, 7(3):227–238, 1979.

[25] D. Evans and G. Oksa. Parallel solution of symmetric positive definite toeplitz systems. *Parallel Algorithms Appl.*, 12(4):297–303, 1997.

[26] E. Golpar-Raboky. A new approach for computing wz factorization. *Appl. Appl. Math.*, 7(2):571–584, 2012.

[27] E. Golpar-Raboky. Abs algorithms for integer wz factorization. *Malaysian J. Math. Sci.*, 8(1):69–85, 2014.

[28] Golub Gene H and Van Loan Charles F. *Matrix computations*. Johns Hopkins University Press, Baltimore MD., 1996.

[29] K. Habgood and I. Arel. A condensation-based application of cramer's rule for solving large-scale linear systems. *J. Discrete Algorithms*, 10:98–109, 2012.

[30] M. Hatzopoulos and N. Missirlis. Advantages for solving linear systems in an asynchronous environment. *J. Comput. Appl. Math.*, 12:331–340, 1985.

[31] N. Higham. *Accuracy and stability of numerical algorithms*. Siam, New York, 2002.

[32] P. Huang, A. MacKay, and D. Teng. A hardwaresoftware codesign of wz factorization to improve time to solve matrices. In *Canadian Conference on Electrical and Computer Engineering*, pages 1–5. IEEE, 2010.

[33] M. Kaps and M. Schlegl. A short proof for the existence of the wz-factorisation. *Parallel Comput.*, 4(2):229–232, 1987.

[34] R. Khazal. Existence and stability of choleski qif for symmetric linear systems. *Int. J. Comput. Math.*, 79(9):1013–1025, 2002.

[35] C. Moler. Cramer's rule on 2-by-2 systems. *ACM SIGNUM Newsletter*, 9(4):13–14, 1974.

[36] S. Rao. Parallel solution of the linear systems by an alternate quadrant interlocking factorization method. *Parallel Algorithms Appl.*, 4(2):1–20, 1994.

[37] S. Rao. Existence and uniqueness of wz factorization. *Parallel Comput.*, 23(8):1129–1139, 1997.

[38] S. Rao and R. Kamra. A hybrid parallel algorithm for large sparse linear systems. *Numer. Linear Algebra Appl.*, 25(6):e2210, 2018.

[39] K. Rhofi, M. Ameur, and A. Radid. Double power method iteration for parallel eigenvalue problem. *Int. J. Pure Appl. Math.*, 108(4):945–955, 2016.

[40] F Stummel. Perturbation theory for evaluation algorithms of arithmetic expressions. *Math. Comput.*, 37(156):435–473, 1981.

[41] O. Ufuoma. A new and simple method of solving large linear systems based on cramer's rule but employing dodgson's condensation. In *Proceedings of the World Congress on Engineering and Computer Science*, volume I, pages 23–25, 2013.

[42] P. Yalamov and D. Evans. The wz matrix factorisation method. *Parallel Comput.*, 21(7):1111–1120, 1995.

[43] Y. Zhong, F. Wu, and Z. Luo. Wz factorization for a kind of special structured matrix. *Journal of National University of Defense Technology*, 32(4):157–164, 2010.