Comparative Analysis of LU and QR Decomposition for Solving Linier Systems in Terms of Computational Time and Numerical Accuracy

Dave Daniell Yanni - 13523003[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]d06163606@gmail.com, 13523003@std.stei.itb.ac.id

***Abstract*—Matrix decomposition, as the name suggests, is a method or process of breaking down a matrix into several simpler matrices. It is often used to simplify computations in various applications, one of which is solving linier systems. Examples of commonly used matrix decomposition methods include LU decomposition and QR decomposition. This paper provides a comparative analysis of LU and QR decomposition techniques for solving linier systems, focusing on their computational efficiency and numerical accuracy**

***Keywords*—Linier systems, LU, matrix decomposition, QR**

I. Introduction

Solving linear systems is crucial in many fields due to its wide-ranging applications. For instance, in computer science, linear systems are integral to algorithms used in search engines, data mining, and machine learning, where solving large-scale systems is essential for processing and analyzing data efficiently.

Linear systems can be solved using various methods, one of which involves matrix decomposition. Among the numerous decomposition techniques, two of the most widely used are LU decomposition and QR decomposition.

This research paper aims to perform a comparative analysis of LU and QR decomposition methods for solving linear systems, with a focus on their computational efficiency and numerical accuracy.

II. Theoretical Basis

A. Linier Equations

A linier equation is an equation in which the highest power of a variable is 1. The standard form of a linier equation is $Ax = B$, here x is a variable, and A is a coefficient and B is constant.

B. Linier Systems

A linier system is a system made of two of more linier equations. The picture below is a linier system made of n number of variables starting from x_1 to x_n and m number of equations.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

Fig. 2.1 Linier System
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Fig. 2.2 Linier System in Matrix form
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$A \qquad x \qquad b$

Fig. 2.3 Linier System as Multiplication of Matrices

Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf

A system of linier equation can be represented in the form of a matrix. As shown in the figures above, the same system of linier equation can be expressed as a matrix and can also be interpreted as a multiplication of matrices.

C. Matrix Decomposition

Matrix decomposition is the process of factorizing a matrix into the product of two or more simpler matrices. Some key types of matrix decomposition are LU decomposition, QR decomposition, and Singular Value Decomposition.

D. LU Decomposition

LU decomposition factorizes a matrix into the product of a lower triangular matrix L and an upper triangular matrix U.

$$A = LU$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & \vdots & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & & \vdots & & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & & \vdots & & \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

L = matriks segitiga bawah (*lower triangular matrix*),
U = matriks segitiga atas (*upper triangular matrix*)

Fig. 2.4 Matrix as Lower and Upper Matrix
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

There are two methods to factorize A into L and U, the first method is Gauss, and the second method is Crout.

E. LU Decomposition with Gauss

To factorize a matrix into L and U with Gauss there are several steps.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

Fig. 2.5 Matrix as Multiplication of Identity and Matrix
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

(*LU* Gauss naif – tidak ada pertukaran baris)

$$A = \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix}$$

Eliminasikan matriks A di ruas kanan menjadi matriks segitiga atas U, dan tempatkan faktor pengali m_{ij} pada posisi l_{ij} di matriks I.

$$\begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} \begin{matrix} R_2 - (^{-2}/_4)R_1 \\ \sim \\ R_3 - (^{1}/_4)R_1 \end{matrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 1.25 & 6.25 \end{bmatrix}$$

Tempatkan $m_{21} = -2/4 = -0.5$ dan $m_{31} = 1/4 = 0.25$ ke dalam matriks L:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & m_{32} & 1 \end{bmatrix}$$

Fig. 2.6 Example of LU Decomposition Using Gauss
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

Teruskan proses eliminasi Gauss pada matriks A,

$$\begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 1.25 & 6.25 \end{bmatrix} \begin{matrix} R_3 - (^{1.25}/_{-2.5})R_2 \\ \sim \end{matrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 0 & 8.5 \end{bmatrix} = U$$

Tempatkan $m_{32} = 1.25/-2.5 = -0.5$ ke dalam matriks L:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{bmatrix}$$

Jadi,

$$A = \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 0 & 8.5 \end{bmatrix}$$

Fig. 2.7 Example of LU Decomposition Using Gauss
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

The first step in LU decomposition is to represent matrix A as IA, then use Gauss elimination on matrix A until it becomes upper matrix U. The multiplication factor m_{ij} is placed in position l_{ij} inside the lower matrix L. After the whole Gauss elimination is done the identity matrix will turn into matrix L and matrix A will turn into matrix U.

F. LU Decomposition with Crout

Another way of solving LU decomposition is using the Crout method.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{3,2} & 1 \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{2,2} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Karena $LU = A$, maka hasil perkalian L dan U itu dapat ditulis sebagai

$$LU = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} \\ l_{31}u_{13} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} \end{bmatrix} = A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Dari kesamaan dua buah matriks $LU = A$, diperoleh

$$u_{11} = a_{11}, \quad u_{12} = a_{12}, \quad u_{13} = a_{13} \quad \} \text{ Baris pertama } U$$

$$\left.\begin{aligned} l_{21}u_1 = a_{21} \rightarrow l_{21} &= \frac{a_{21}}{u_{11}} \\ l_{31}u_{11} = a_{31} \rightarrow l_{31} &= \frac{a_{31}}{u_{11}} \end{aligned}\right\} \text{ Kolom pertama } L$$

$$\left.\begin{aligned} l_{21}u_{12} + u_{22} = a_{22} &\rightarrow u_{22} = a_{22} - l_{21}u_{12} \\ l_{21}u_{13} + u_{23} = a_{23} &\rightarrow u_{23} = a_{23} - l_{21}u_{13} \end{aligned}\right\} \text{ Baris kedua } U$$

$$l_{31}u_{12} + l_{32}u_{22} = a_{32} \rightarrow \quad l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} \qquad \text{Kolom kedua } L$$

$$l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33} \rightarrow u_{33} = a_{33} - (l_{31}u_{13} + l_{32}u_{23}) \quad \} \text{ Baris ketiga } U$$

Fig. 2.8 Example of LU Decomposition Using Crout

Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

Solving using the Crout method follows a pattern, first is finding the first row of the upper matrix U, then finding the first column of the lower matrix L, then finding the second row of the upper matrix U and so on. The following general equation is used to find the elements for upper matrix U and lower matrix L.

$$u_{pj} = a_{pj} - \sum_{k=1}^{p-1} l_{pk}\, u_{kj}, \qquad \begin{matrix} p = 1, 2, 3, \ldots, n \\ j = p, p+1, \ldots, n \end{matrix}$$

Fig. 2.9 General Equation for Upper Matix in Crout
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

$$l_{iq} = \frac{a_{iq} - \sum_{k=1}^{q-1} 1_{ik}\, u_{kq}}{u_{qq}}, \quad \begin{matrix} q = 1, 2, 3, \ldots, n\text{-}1 \\ i = q+1, q+2, \ldots, n \\ \text{dengan syarat } u_{qq} \neq 0 \end{matrix}$$

Fig. 2.10 General Equation for Lower Matix in Crout
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

F. Solving Linier Systems with LU Decomposition

To solve a linier system $Ax = B$, matrix A is first decomposed into L and U matrices. Once A has been decomposed, it can be replaced by LU in the equation, resulting in $LUx = B$. This can be solved in two steps. Solve $Ly = B$ for y where y is LU, then solve $Ux = y$ for x.

$$Ly = b \rightarrow \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ l_{21} & 1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ l_{n1} & l_{n2} & l_{n3} & \ldots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}$$

Fig. 2.11 Ly = B Matrix
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

$$Ux = y \rightarrow \begin{bmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ 0 & u_{22} & u_{23} & \ldots & u_{2n} \\ \ldots & & & & \ldots \\ 0 & 0 & 0 & \ldots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Fig. 2.12 Ux = y Matrix
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf

G. QR Decomposition

QR decomposition factorizes a matrix into the product of a orthonormal matrix Q and an upper triangular matrix R. QR decomposition involves the Gram-Schmidt method.

$$A = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \right]$$

Fig. 2.13 Matrix A
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{a}_1, \quad \mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}, \\ \mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1, \quad \mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}, \\ \mathbf{u}_{k+1} &= \mathbf{a}_{k+1} - (\mathbf{a}_{k+1} \cdot \mathbf{e}_1)\mathbf{e}_1 - \cdots - (\mathbf{a}_{k+1} \cdot \mathbf{e}_k)\mathbf{e}_k, \quad \mathbf{e}_{k+1} = \frac{\mathbf{u}_{k+1}}{\|\mathbf{u}_{k+1}\|}. \end{aligned}$$

Fig. 2.14 General Equation for QR Decomposition
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Columns in matrix A are labeled as a_1 to a_n which is then used to find orthonormal vectors for matrix Q.

$$A = \left[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \right] = \left[\mathbf{e}_1 \mid \mathbf{e}_2 \mid \cdots \mid \mathbf{e}_n \right] \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_n \end{bmatrix} = QR.$$

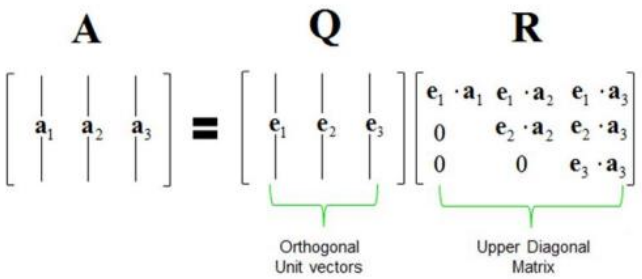

Fig. 2.15 QR Decomposition
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Upper matrix R follows the general equation $e_n \,.\, a_m$ where n is the current row and m is the current column and $m \geq n$.

H. Solving Linier Systems with QR Decomposition

To solve a linier system $Ax = B$,

1. Matrix A is first decomposed into Q and R matrices.
2. Once A has been decomposed, it can be replaced by LU in the equation, resulting in $QRx = B$.
3. Multiply both sides with Q^T

$$Q^T QRx = Q^T B$$

$$IRx = Q^T B$$
$$Rx = Q^T B$$

4. Solve for x.

$$A = \begin{bmatrix} -1 & 3 \\ 1 & 5 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Fig. 2.16 Matrix A Example
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Figure above is matrices used for example of solving linier systems using QR decomposition.

$$Q = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \approx \begin{bmatrix} -0.707106781186548 & 0.707106781186548 \\ 0.707106781186548 & 0.707106781186548 \end{bmatrix}$$

$$R = \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & 4\sqrt{2} \end{bmatrix} \approx \begin{bmatrix} 1.414213562373095 & 1.414213562373095 \\ 0 & 5.65685424949238 \end{bmatrix}$$

Fig. 2.17 QR Decomposition
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Result of QR decomposition of matrix A.

$$Rx = Q^T b$$

$$\begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & 4\sqrt{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

Fig. 2.18 Result System
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Final system after following steps 1 to 3.

Lakukan penyulihan mundur:

Persamaan baris ke-2: $4\sqrt{2}\,x_2 = \frac{\sqrt{2}}{2} \rightarrow x_2 = \frac{\frac{\sqrt{2}}{2}}{4\sqrt{2}} = \frac{1}{8}$

Persamaan baris ke-1: $\sqrt{2}\,x_1 + \sqrt{2}\,x_2 = -\frac{\sqrt{2}}{2} \rightarrow x_1 = \frac{-\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{8}}{\sqrt{2}} = \frac{\frac{-5\sqrt{2}}{8}}{\sqrt{2}} = \frac{-5}{8}$

Fig. 2.19 Solving the equation
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf

Solve for both variables using backward substitution.

III. METHODOLOGY

A. Code Description

The experiment is done in Python with NumPy and SciPy libraries for matrix decomposition.

```
import numpy as np
from scipy.linalg import lu, qr
import time

# Function to solve linear system using LU decomposition
def solve_with_lu(A, B):
    P, L, U = lu(A)
    y = np.linalg.solve(L, np.dot(P, B))  # Forward substitution
    x = np.linalg.solve(U, y)  # Backward substitution
    return x

# Function to solve linear system using QR decomposition
def solve_with_qr(A, B):
    Q, R = qr(A)
    x = np.linalg.solve(R, np.dot(Q.T, B))
    return x

# Function to calculate error percentage and accuracy percentage
def calculate_error_accuracy(x_computed, x_actual):
    error_percentage = (np.linalg.norm(x_computed - x_actual) / np.linalg.norm(x_actual)) * 100
    accuracy_percentage = 100 - error_percentage
    return error_percentage, accuracy_percentage

# Function to compare methods with accuracy and stability
def compare_methods_with_accuracy(matrix_size, trials):
    lu_times, qr_times = [], []
    lu_errors, qr_errors = [], []
    lu_accuracies, qr_accuracies = [], []

    np.random.seed(0)  # Ensure reproducibility

    for _ in range(trials):
        # Generate a random matrix A and solution x_actual
        A = np.random.rand(matrix_size, matrix_size)
        # Improve conditioning by adding a diagonal matrix
        A += np.eye(matrix_size) * 10
        x_actual = np.random.rand(matrix_size)
        B = np.dot(A, x_actual)  # Generate B to ensure consistent solution

        # Solve using LU decomposition
        start_time = time.time()
        x_lu = solve_with_lu(A, B)
        lu_times.append(time.time() - start_time)
        # Calculate error and accuracy for LU
        lu_error, lu_accuracy = calculate_error_accuracy(x_lu, x_actual)
        lu_errors.append(lu_error)
        lu_accuracies.append(lu_accuracy)

        # Solve using QR decomposition
        start_time = time.time()
        x_qr = solve_with_qr(A, B)
        qr_times.append(time.time() - start_time)
        # Calculate error and accuracy for QR
        qr_error, qr_accuracy = calculate_error_accuracy(x_qr, x_actual)
        qr_errors.append(qr_error)
        qr_accuracies.append(qr_accuracy)
```

```
    # Compute averages
    avg_lu_time = np.mean(lu_times)
    avg_qr_time = np.mean(qr_times)
    avg_lu_error = np.mean(lu_errors)
    avg_qr_error = np.mean(qr_errors)
    avg_lu_accuracy = np.mean(lu_accuracies)
    avg_qr_accuracy = np.mean(qr_accuracies)

    return avg_lu_time, avg_qr_time, avg_lu_error, avg_qr_error, avg_lu_accuracy, avg_qr_accuracy

matrix_size = 500
trials = 10
results = compare_methods_with_accuracy(matrix_size, trials)

print(f"Matrix Size: {matrix_size}x{matrix_size}, Trials: {trials}")
print(f"LU Decomposition: Avg Time = {results[0]:.6f}s, Avg Error = {results[2]:.6f}%, Avg Accuracy = {results[4]:.6f}%")
print(f"QR Decomposition: Avg Time = {results[1]:.6f}s, Avg Error = {results[3]:.6f}%, Avg Accuracy = {results[5]:.6f}%")
```

There are several functions in the code:

1. solve_with_lu(A, B)
   Function to solve matrix $Ax = B$ using LU decomposition
2. solve_with_qr(A, B)
   Function to solve matrix $Ax = B$ using QR decomposition.
3. calculate_error_accuracy(x_computed, x_actual)
   Function to calculate error and accuracy of the solving methods.
4. compare_methods_with_accuracy(matrix_size, trials)
   Function to compare LU and QR decomposition

B. Experimental Setup

The purpose of this experiment is to compare and observe the computational time and accuracy of both LU and QR decomposition in solving linear systems. The variables used in this experiment are matrix_size and trials, where matrix_size determines the size of the matrix to be tested, and trials represent the number of runs or computations of matrices the code performs. The matrix is generated as a random dense matrix using np.random.rand(matrix_size, matrix_size).

The metrics measured in the experiment are computational time and accuracy. Computational time is calculated by taking the difference between the start time and the time after the calculations are completed. Accuracy is measured by the difference between the computed results and the actual results.

IV. Results and Analysis

```
Matrix Size: 500x500, Trials: 100
LU Decomposition: Avg Time = 0.036049s, Avg Error = 0.000000%, Avg Accuracy = 100.000000%
QR Decomposition: Avg Time = 0.058775s, Avg Error = 0.000000%, Avg Accuracy = 100.000000%
```

Fig. 4.1 Experiment Results 1

```
Matrix Size: 500x500, Trials: 10
LU Decomposition: Avg Time = 0.061627s, Avg Error = 0.000000%, Avg Accuracy = 100.000000%
QR Decomposition: Avg Time = 0.040341s, Avg Error = 0.000000%, Avg Accuracy = 100.000000%
```

Fig. 4.2 Experiment Results 2

```
Matrix Size: 500x500, Trials: 10
LU Decomposition: Avg Time = 0.032714s, Avg Error = 29835.187326%, Avg Accuracy = -29735.187326%
QR Decomposition: Avg Time = 0.065895s, Avg Error = 0.000000%, Avg Accuracy = 100.000000%
```

Fig. 4.3 Experiment Results 3

A. Computational Time

In the first experiments, LU decomposition had a comparatively lower computational time. However, in the second experiment, LU resulted in a longer computational time. Generally, when solving linear systems, LU is faster than QR decomposition, but in some cases, LU can lead to longer computational times. When solving using the Gauss or Crout method in LU, if the process encounters a row that is empty, Gauss requires additional time to swap rows, while Crout requires both row swapping and recalculating from the beginning. In contrast, QR decomposition maintains a consistent flow, and thus LU might result in slower computational times for certain matrices.

B. Numerical Accuracy

In the first and second experiments, both LU and QR decompositions demonstrated 0% error and 100% accuracy. However, in the third experiment, where the matrix was poorly conditioned, LU decomposition exhibited significantly worse accuracy, with an error of 29,835.19%. In contrast, QR decomposition maintained its stability and accuracy. This highlights the greater consistency and stability of QR decomposition, particularly when dealing with poorly conditioned matrices.

V. Discussion

The results demonstrated that LU decomposition generally outperforms QR in terms of computational time. However, QR decomposition exhibited more consistent and stable performance across all trials, which can be attributed to the inherent stability of the QR algorithm. Both methods achieved 100% accuracy, indicating that they provide reliable solutions when solving linear systems with well-conditioned

Despite these insights, the experimental setup was limited by the use of randomly generated dense matrices. Such matrices may not fully capture the complexities of real-world problems, as they do not account for factors like matrix conditioning or sparsity, which can significantly impact the performance of both methods. In particular, the condition number of a matrix plays a crucial role in the stability of LU decomposition, while QR decomposition tends to perform better with poorly conditioned matrices.

Future work should focus on exploring a wider variety of matrix types, including sparse and ill-conditioned matrices, to assess how these factors influence the performance of LU and QR decomposition. Additionally, scaling the problem size and incorporating parallelization could provide further insights into the scalability and efficiency of both methods for large-scale systems. By addressing these factors, a more comprehensive understanding of the strengths and weaknesses of LU and QR decomposition in different contexts can be achieved.

VI. Conclusion

Both LU and QR decomposition are effective methods for solving linier systems, but the choice between them should

depend on the specific characteristics of the matrix and the problem at hand.

VI. ACKNOWLEDGMENT

The author extends heartfelt gratitude to God for providing wisdom, perseverance, and opportunity to complete this paper successfully. Sincere appreciation is all extended to Ir. Rila Mandala, M.Eng., Ph.D., the lecturer of the IF2123 Linier Algebra and Geometry course.

REFERENCES

[1] Munir, Rinaldi. 2023. "Sistem Persamaan Linier (Bagian 1: Metode Eliminasi Gauss)". https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf (accessed on 2 December 2024).

[2] Munir, Rinaldi. 2023. "Dekomposisi LU". https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf (accessed on 2 December 2024).

[3] Munir, Rinaldi. 2023. "Sistem Persamaan Linier (Bagian 1: Metode Eliminasi Gauss)". https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf (accessed on 2 December 2024).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Januari 2025

Dave Daniell Yanni 13523003