Analisis Perbandingan dan Implementasi Metode Least Squares Menggunakan Eliminasi Gauss dan Dekomposisi LU untuk Penyelesaian Sistem Persamaan Linear pada Model Lotka-Volterra

Muh. Rusmin Nurwadin-13523068[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523068@mahasiswa.itb.ac.id, [2]rusmn17@gmail.com

***Abstrak*— Model Lotka-Volterra merupakan sistem persamaan diferensial non-linear yang menggambarkan interaksi predator-prey dalam suatu ekosistem. Dalam penerapannya, model ini menghadapi tantangan dalam penyesuaian parameter berdasarkan data observasi, terutama ketika data mengandung noise atau ketidakpastian. Penelitian ini bertujuan untuk membandingkan efektivitas metode Eliminasi Gauss dan Dekomposisi LU dalam menyelesaikan sistem persamaan linear yang dihasilkan dari penerapan metode Least Squares pada model Lotka-Volterra. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan library NumPy untuk komputasi numerik dan Matplotlib untuk visualisasi. Pengujian dilakukan menggunakan dataset fiktif dengan variasi ukuran data dari 100 hingga 5000 titik. Hasil penelitian menunjukkan bahwa kedua metode memiliki tingkat akurasi yang sebanding dengan error relatif berkisar antara 10^{-16} hingga 10^{-18}. Untuk kasus single RHS, Eliminasi Gauss menunjukkan efisiensi memori yang lebih baik, sementara Dekomposisi LU unggul dalam penyelesaian multiple RHS dengan waktu komputasi 40% lebih cepat. Penelitian ini memberikan panduan dalam pemilihan metode yang tepat berdasarkan karakteristik permasalahan yang dihadapi.**

***Kata Kunci*— Dekomposisi LU, Eliminasi Gauss, Least Squares, Lotka-Volterra, Sistem Persamaan Linear.**

I. PENDAHULUAN

Model Lotka-Volterra merupakan suatu model yang menyajikan dinamika interaksi antara dua populasi yang bersaing untuk persediaan makanan atau sumber alam lainnya, dimana satu populasi berperan sebagai predator, dan yang lainnya sebagai prey dalam suatu ekosistem [1]. Model yang juga dikenal dengan predator-prey ini pertama dikenalkan oleh Alfred J.Lotka pada tahun 1925 dan Vito Volterra pada tahun 1926 [2]. Seiring perkembangannya, model ini tidak hanya digunakan dalam ekologi saja, melainkan juga dalam berbagai bidang lain seperti epidemiologi, ekonomi, dan dinamika sosial, dengan berbagai modifikasi dan pengembangan agar sesuai dengan kompleksitas yang ada kedepannya.

Model Lotka-Volterra menghadapi tantangan besar, terutama saat harus menyesuaikan parameter berdasarkan data observasi yang semakin banyak dan rumit. Sifat nonlinear dari model ini membuat proses penyesuaian parameter menjadi lebih sulit, terutama jika datanya mengandung noise atau ketidakpastian. Metode Least Squares sering digunakan untuk memperkirakan parameter ini, tetapi cara ini menghasilkan sistem persamaan linear yang membutuhkan metode penyelesaian yang cepat dan tepat, terutama untuk data dalam jumlah besar [3].

Pendekatan dalam metode numerik telah menghasilkan berbagai pendekatan baru untuk menylesaikan sistem persamaan linear, salah satunya dengan pemanfaatan metode Eliminasi Gauss, yang dapat meningkatkan efisiensi komputasi menjadi lebih baik. Sementara itu, implementasi dari Dekomposisi LU juga dapat digunakan untuk dalam pengoptimalan algoritma untuk masalah berskala lebih besar [4].

Meskipun telah ada beberapa penelitian tentang metode numerik, namun analisis perbandingan antara metode Eliminasi Gauss dan Dekomposisi LU masih terbatas, khususnya dalam penyelesaian sistem persamaan linear yang muncul pada model Lotka-Volterra. Penelitian sebelumnya lebih banyak membahas aspek teoretis, tanpa memberikan perbandingan langsung antara kedua metode tersebut dalam implementasinya. Hal ini menunjukkan perlunya melakukan perbandingan lebih mendalam antara kedua metode tersebut.

Tujuan dari penelitian ini adalah untuk membandingkan metode eliminasi Gauss dan dekomposisi LU dalam menyelesaikan sistem persamaan linear yang diperoleh dengan menerapkan metode least square pada model Lotka-Volterra. Implementasi yang dilakukan adalah dengan menggunakan bahasa pemrograman Python, yang selanjutnya akan berfokus pada kecepatan perhitungan, keakuratan hasil, dan efisiensi penggunaan memori.

Penelitian ini dilakukan untuk menunjukkan dengan jelas kelebihan dan kekurangan masing-masing metode yang digunakan, sehingga penelitian ini diharapkan dapat

menjadi acuan dalam menentukan metode terbaik untuk menyelesaikan sistem persamaan linear dalam kasus serupa.

II. LANDASAN TEORI

A. Model Lotka-Volterra

Model Lotka-Volterra atau yang juga dikenal dengan model Mangsa-Pemangsa merupakan model dalam bentuk sistem persamaan diferensial biasa yang non-linier yang menggambarkan interaksi antara dua makhluk hidup yang saling berikatan dalam hubungan predasi [5]. Model ini terdiri dari dua persamaan diferensial, yakni sebagai berikut.

1. Dinamika Prey (Mangsa).

$$\frac{dx}{dt} = \alpha x - \beta xy \quad (1)$$

Dengan x = populasi mangsa, y = populasi predator, α = tingkat pertumbuhan mangsa tanpa kehadiran predator, dan β = tingkat kematian mangsa karena interaksi dengan predator.

2. Dinamika Predator.

$$\frac{dy}{dt} = \delta xy - \gamma y \quad (2)$$

Dengan δ = tingkat pertumbuhan predator karena konsumsi mangsa dan γ = tingkat kematian predator tanpa kehadiran mangsa.

Model ini didasari oleh beberapa asumsi, seperti populasi prey akan tumbuh secara eksponensial tanpa adanya predator, interaksi yang terjadi mengurangi populas dengan laju proposional terhadap pertemuan kedua populasi, populasi predator akan menurun secara eksponensial tanpa adanya prey, serta predator dapat mengkonversi interaksi dengan prey menjadi pertumbuhan populasinya.

B. Metode Least Square

Metode Least Square merupakan metode analisis yang digunakan untuk melakukan suatu estimasi terhadap suatu hal. Dalam pengaplikasiannya, metode ini membutuhkan berbagai macam informasi atau data yang cukup banyak dan diamati dalam periode waktu yang relatif panjang, sehingga dari hasil analisis yang dilakukan dapat diketahui seberapa besar fluktuasi yang terjadi, dan faktor apa saja yang mempengaruhi perubahan tersebut [6]. Secara teoritis, keakuratan data yang diperoleh serta rentang waktu pengumpulan data menjadi aspek krusial dalam analisis data berkala. Dengan semakin banyaknya data yang terkumpul, estimasi yang dihasilkan pun cenderung lebih akurat.

Least Square bekerja dengan cara mengurangi kesalahan prediksi yang dihitung sebagai selisih antara nilai aktual (Y) dan nilai prediksi (Y'). Persamaan matematis yang digunakan adalah sebagai berikut.

$$Y' = a + bX \quad (3)$$

Dengan Y' = nilai yang diprediksi, a = konstata atau intercept (nilai Y saat $X = 0$), b = kemiringan garis, menunjukkan tingkat perubahan Y terhadap perubahan X, dan X = variabel independen.

Untuk menghitung a dan b dilakukan dengan persamaan berikut.

$$a = \frac{\Sigma Y}{n} \quad (4)$$

$$b = \frac{\Sigma XY}{\Sigma X^2} \quad (5)$$

Dengan ΣY = jumlah Y, ΣXY = jumlah hasil perkalian X dan Y, ΣX^2 = jumlah nilai X kuadrat, dan n = jumlah data.

Penggunaan metode ini diawali dengan pengumpulan data, lalu menghitung parameter a dan b, lalu menggunakan persamaan untuk menentukan nilai-nilai terkait.

Dalam model lotka-volterra, metode ini digunakan untuk menyesuaikan parameter model, agar sesuai dengan data populasi predator dan mangsa dari pengamatan yang dilakukan. Dengan menghitung selisih antara nilai yang diprediksi oleh model dan nilai aktual, metode ini meminimalkan jumlah kuadrat selisih (residual) melalui proses optimasi. Hasilnya adalah parameter yang menggambarkan dinamika interaksi predator-mangsa yang paling sesuai dengan data nyata, memungkinkan prediksi yang lebih akurat dan relevan terhadap sistem ekologi yang diamati. Metode ini kerap kali melibatkan penyelesaian sistem persamaan linear normal, yang dapat diselesaikan menggunakan eliminasi Gauss atau dekomposisi LU.

C. Sistem Persamaan Linear

Sistem Persamaan Linear (SPL) merupakan salah satu persamaan dari ilmu aljabar yang mana persamaan ini sukunya mengandung konstanta dengan variabel tunggal. Jika di gambarkan dalam koordinat kartesius maka kecenderungan akan membentuk garis lurus, maka dari itu hal ini menjadi salah satu alasan disebut linear [7]. Bentuk umum SPL adalah sebagai berikut.

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

Gambar 1. Bentuk umum SPL
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

SPL dalam bentuk matriks disajikan sebagai berikut.

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

Gambar 2. SPL dalam bentuk matriks
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Atau dalam perkalian matriks, digambarkan sebagai berikut.

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

A **x** **b**

Gambar 3. SPL dalam perkalian matriks
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

D. Metode Eliminasi Gauss

Eliminasi Gauss ditemukan oleh Carl Friderich Gauss, metode ini digunakan dalam memecahkan sistem persamaan linear dengan merepresentasikan SPL menjadi bentuk matriks, selanjutnya matriks tersebut diubah ke dalam bentuk Eselon Baris melalui Operasi Baris Elementer (OBE). Terakhir, sistem diselesaikan dengan substitusi balik. Operasi Baris Elementer dilakukan dengan tiga operasi berikut terhadap matriks augmented [8].

1. Mengalikan sebuah baris dengan konstanta tidak nol.
2. Pertukaran dua buah baris.
3. Tambahkan sebuah baris dengan kelipatan baris lainnya.

Berikut contoh matriks augmented.

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

Gambar 4. Matriks Augmented
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Hasil penerapan OBE pada matriks augmented sampai terbentuk matriks eselon baris.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_n \end{bmatrix} \sim \text{OBE} \sim \begin{bmatrix} 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & 1 & * \end{bmatrix}$$

Gambar 5. Matriks Augmented menjadi Matriks Eselon Baris denga OBE
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Dalam implementasi praktis, untuk meningkatkan stabilitas numerik dan menghindari error yang besar akibat pembagian dengan angka yang sangat kecil, Eliminasi Gauss dapat menggunakan teknik partial pivoting. Teknik ini melakukan penukaran baris sehingga elemen diagonal (pivot) yang akan digunakan memiliki nilai absolut terbesar dalam kolomnya. Proses ini dilakukan sebelum eliminasi pada setiap langkah, dimana program akan mencari elemen dengan nilai absolut terbesar pada kolom di bawah diagonal utama dan menukar barisnya dengan baris yang sedang diproses jika ditemukan elemen yang lebih besar. Partial pivoting sangat penting untuk mengurangi akumulasi error pembulatan yang dapat terjadi selama proses eliminasi, terutama ketika berhadapan dengan sistem persamaan yang memiliki perbedaan signifikan dalam besaran koefisiennya.

E. Metode Dekomposisi LU

Dekomposisi LU adalah suatu metode untuk memfaktorkan (dekomposisi) matriks (A) menjadi hasil kali dua matriks, yaitu matriks segitiga bawah (Lower triangular matrix (L)) dan matriks segitiga atas (Upper triangular matrix (U)) [9]. Dekomposisi LU dapat ditulis sebagai berikut.

$$A = LU \tag{6}$$

Dalam matriks, ditulis sebagai berikut.

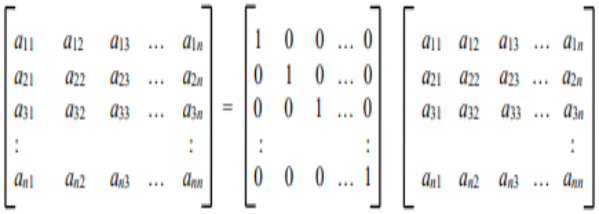
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \vdots & & \vdots & & \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ l_{21} & 1 & 0 & \ldots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

Gambar 6. Dekomposisi LU
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Dalam melakukan dekomposisi LU terdapat dua metode yang dapat digunakan, yaitu dengan metode LU-Gauss dan metode reduksi Crout. Metode LU-Gauss berdasarkan Eliminasi Gauss, sementara metode reduksi Crout berdasarkan kesamaan dua buah matriks.

Langkah-langkah pembentukan *L* dan *U* dari matriks A dengan metode LU-Gauss adalah sebagai berikut.

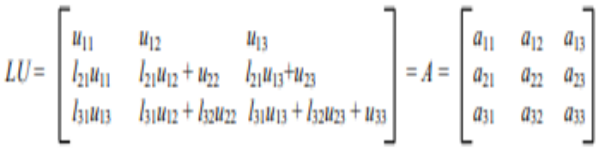
1. Nyatakan *A* sebagai *A* = *IA*. Dimana *A* adalah matriks awal, dan *I* adalah identitas.



Gambar 7. Matriks A sama dengan *IA*
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/

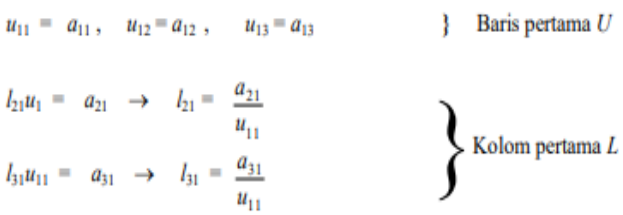
2. Lakukan eliminasi Gauss pada matriks *A* menjadi matrik *U*. Selanjutnya tempatkan faktor pengali (m_{ij}) pada posisi l_{ij} pada matriks *l*.
3. Setelah seluruh proses eliminisai selesai, matriks *l* menjadi matriks *L* dan matriks A menjadi matriks *U*.

Metode selanjutnya adalah metode Crout atau juga yang dikenal dengan metode Cholesky atau Doolittle. Karena *LU* = *A*, maka hasil perkalian *L* dan *U* dapat ditulis sebagai berikut.



Gambar 8. Representasi Matriks A = LU
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Dari kesamaan dua buah matriks *LU* = *A*, diperoleh sebagai berikut.



Gambar 9. Langkah Menentukan *LU*
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Selanjutnya, untuk menentukan matriks L dan U, maka dilakukan langkah berikut sebanyak k kali, dengan k adalah ukuran dari matriks A.

(1) Hitung elemen-elemen baris pertama dari U
(2) Hitung elemen-elemen kolom pertama dari L
(3) Hitung elemen-elemen baris kedua dari U
(4) Hitung elemen-elemen baris kedua dari L
(5) ...dst
(..) Hitung elemen-elemen baris ke-k dari U
(..) Hitung elemen-elemen baris ke-k dari L

Dekomposisi LU dapat digunakan untuk menentukan solusi SPL Ax = b, diawali dengan melakukan dekomposisi A menjadi L dan U. Kemudian menentukan solusi Ly = b, dengan menghitung y dengan teknik penyulihan maju. Terakhir pecahkan Ux = y, dengan menghitung x dengan teknik penyulihan mundur.

Salah satu keunggulan dari Dekomposisi LU adalah efisiensinya dalam menyelesaikan sistem persamaan linear dengan multiple Right Hand Side (RHS). Multiple RHS mengacu pada situasi di mana kita perlu menyelesaikan beberapa sistem persamaan dengan matriks koefisien yang sama tetapi dengan vektor konstanta yang berbeda. Dalam kasus ini, matriks L dan U hanya perlu didekomposisi sekali dan dapat digunakan kembali untuk setiap vektor konstanta yang berbeda, sehingga menghemat waktu komputasi secara signifikan.

F. Error Relatif

Error relatif merupakan metrik yang digunakan untuk mengevaluasi akurasi hasil perhitungan numerik. Dalam konteks penyelesaian sistem persamaan linear, error relatif membantu mengukur seberapa jauh hasil perhitungan menyimpang dari nilai sebenarnya. Error relatif dihitung dengan membandingkan selisih antara nilai hasil perhitungan dengan nilai sebenarnya, dibagi dengan nilai sebenarnya, yang dapat dinyatakan dalam persamaan berikut.

$$error = \frac{|x - x*|}{|x*|} \qquad (7)$$

Dengan *x* = nilai hasil perhitungan dan *x** = nilai sebenarnya.

Semakin kecil nilai error relatif, semakin akurat hasil perhitungan yang diperoleh. Dalam praktiknya, error relatif sering dinyatakan dalam notasi ilmiah untuk memudahkan perbandingan tingkat akurasi antara metode-metode yang berbeda. Hal ini sangat berguna dalam membandingkan keakuratan metode Eliminasi Gauss dan Dekomposisi LU, terutama ketika ukuran sistem persamaan menjadi sangat besar.

III. METODOLOGI

Penelitian ini menggunakan pendekatan kuantitatif yang terdiri atas tiga tahap utama. Tahap pertama melibatkan studi literatur yang mendalam terhadap model Lotka-Volterra, metode Least Squares, metode Eliminasi Gauss, dan Dekomposisi LU untuk membangun landasan teoretis yang kuat. Studi ini bertujuan untuk memahami prinsip dasar, kelebihan, dan kekurangan masing-masing metode dalam menyelesaikan sistem persamaan linear yang dihasilkan dari penerapan model Lotka-Volterra.

Tahap kedua mencakup implementasi program menggunakan bahasa pemrograman Python. Dalam tahap ini, digunakan library NumPy untuk mendukung operasi matriks dan komputasi numerik yang efisien, serta Matplotlib untuk visualisasi data hasil analisis. Implementasi program melibatkan pembuatan fungsi-

fungsi khusus untuk menghasilkan data (generate data), menerapkan metode Eliminasi Gauss dengan partial pivoting, melakukan Dekomposisi LU, serta menghitung error relatif sebagai metrik evaluasi akurasi.

Data fiktif yang digunakan disimulasikan dengan menambahkan noise Gaussian berdistribusi normal ($\mu = 0$, $\sigma = 0.1$) untuk merepresentasikan fluktuasi alami dalam data populasi. Parameter noise ini ditentukan berdasarkan asumsi umum pada data lapangan yang menunjukkan variasi moderat akibat ketidakstabilan ekosistem. Proses ini dilakukan menggunakan fungsi *numpy.random.normal* untuk memastikan distribusi noise yang konsisten.

Tahap ketiga adalah pengujian dan analisis komparatif kedua metode menggunakan dataset fiktif dengan variasi ukuran data dari 100 hingga 5000 titik. Analisis performa dilakukan dengan mempertimbangkan tiga aspek berikut.

1. Kompleksitas waktu: Diukur dari waktu eksekusi program untuk setiap metode.
2. Akurasi numerik: Dievaluasi melalui perhitungan error relatif untuk mengukur deviasi hasil.
3. Efisiensi komputasi: Diperiksa khususnya dalam penanganan kasus multiple RHS, yang mengharuskan penyelesaian sistem persamaan linear dengan berbagai vektor konstanta.

Dengan pendekatan metodologi ini, penelitian berfokus pada pengujian dan perbandingan performa kedua metode untuk mengidentifikasi keunggulan relatif masing-masing dalam konteks penyelesaian sistem persamaan linear pada model Lotka-Volterra. Penelitian ini diharapkan dapat memberikan wawasan praktis bagi pengguna dalam memilih metode yang paling sesuai berdasarkan karakteristik permasalahan yang dihadapi.

IV. IMPLEMENTASI

Implementasi program dilakukan menggunakan bahasa pemrograman Python. Implementasi terdiri atas beberapa komponen utama yang saling terintegrasi untuk melakukan analisis perbandingan metode Eliminasi Gauss dan Dekomposisi LU pada Model Lotka-Volterra.

A. Library Program

```python
import numpy as np
import matplotlib.pyplot as plt
import time
```

Gambar 10. Library Program
Sumber : Dokumen Pribadi

Program ini menggunakan tiga library utama Python yang masing-masing memiliki peran penting dalam perhitungan dan visualisasi. Library NumPy (numpy) digunakan untuk operasi matriks dan array, mendukung komputasi numerik yang efisien, serta menyediakan fungsi-fungsi matematis yang diperlukan seperti gradient, dot, dan zeros. Untuk visualisasi data populasi predator-prey, digunakan library Matplotlib (matplotlib.pyplot) yang memungkinkan pembuatan grafik interaktif dan informatif untuk membantu analisis hasil. Library Time (time) dimanfaatkan khusus untuk pengukuran waktu komputasi dalam analisis performa metode numerik.

B. Generate Data

```python
def generate_data(n_points=1000):
    t = np.linspace(0, 10, n_points)

    base_prey = np.linspace(10, 27, n_points)
    noise_prey = np.random.normal(0, 0.5, n_points)
    x = base_prey + noise_prey

    base_predator = 5 - 0.15 * t
    noise_predator = np.random.normal(0, 0.3, n_points)
    y = base_predator + noise_predator

    return t, x, y
```

Gambar 11. Generate Data
Sumber : Dokumen Pribadi

Fungsi generate_data() membangkitkan dataset fiktif untuk model predator-prey dengan parameter jumlah titik data. Fungsi ini menghasilkan data prey dengan tren naik dan predator dengan tren menurun, masing-masing ditambah noise untuk simulasi fluktuasi alami. Output berupa array waktu, populasi prey, dan populasi predator.

C. Visualisasi Data

```python
def plot_data(t, x, y, title="Data Populasi Predator-Prey"):
    plt.figure(figsize=(12,6))
    plt.plot(t, x, 'b-', label='Prey', alpha=0.7)
    plt.plot(t, y, 'r-', label='Predator', alpha=0.7)
    plt.xlabel('Waktu')
    plt.ylabel('Populasi')
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()
```

Gambar 12. Visualisasi Data
Sumber : Dokumen Pribadi

Fungsi plot_data() menampilkan visualisasi hasil dalam bentuk grafik dua dimensi yang menunjukkan perubahan populasi prey dan predator terhadap waktu. Grafik dilengkapi dengan elemen-elemen pendukung untuk memudahkan interpretasi data.

D. Least Squares

```python
def build_least_squares(t, x, y):
    dx = np.gradient(x, t)
    dy = np.gradient(y, t)

    n = len(t)
    A = np.zeros((2*n, 4))
    b = np.zeros(2*n)

    for i in range(n):
        # Persamaan untuk dx/dt
        A[2*i] = [x[i], -x[i]*y[i], 0, 0]
        b[2*i] = dx[i]

        # Persamaan untuk dy/dt
        A[2*i+1] = [0, 0, -y[i],
x[i]*y[i]]
        b[2*i+1] = dy[i]

    # Bentuk persamaan normal
    ATA = np.dot(A.T, A)
    ATb = np.dot(A.T, b)

    return ATA, ATb

```

Gambar 13. Least Square
Sumber : Dokumen Pribadi

Fungsi build_least_squares() mengimplementasikan metode Least Squares untuk membangun sistem persamaan linear. Fungsi ini menghitung turunan numerik data dan membentuk matriks koefisien serta vektor konstanta yang akan diselesaikan menggunakan metode numerik.

E. Eliminasi Gauss

```python
def gauss_elimination(A, b):
    n = A.shape[0]
    # Buat augmented matrix [A|b]
    Ab = np.column_stack([A, b])
    x = np.zeros(n)

    # Forward elimination
    for i in range(n):
        max_idx = i + np.argmax(np.abs(Ab[i:, i]))
        if max_idx != i:
            Ab[i], Ab[max_idx] = Ab[max_idx].copy(),
Ab[i].copy()
        pivot = Ab[i,i]
        if np.abs(pivot) < 1e-10:
            continue

        for j in range(i+1, n):
            factor = Ab[j,i] / pivot
            Ab[j] -= factor * Ab[i]

    # Back substitution
    for i in range(n-1, -1, -1):
        if np.abs(Ab[i,i]) < 1e-10:
            x[i] = 0
            continue
        x[i] = Ab[i,-1]
        for j in range(i+1, n):
            x[i] -= Ab[i,j] * x[j]
        x[i] /= Ab[i,i]

    return x

```

Gambar 14. Eliminasi Gauss
Sumber : Dokumen Pribadi

Fungsi gauss_elimination() mengimplementasikan metode Eliminasi Gauss dengan partial pivoting. Proses terdiri dari forward elimination untuk mendapatkan matriks segitiga atas dan back substitution untuk memperoleh solusi. Implementasi mencakup penanganan kasus pivot mendekati nol.

F. Dekomposisi LU

```python
def lu_decomposition(A):
    n = A.shape[0]
    L = np.eye(n)
    U = A.copy()
    P = np.eye(n)

    for i in range(n-1):
        pivot_idx = i + np.argmax(np.abs(U[i:, i]))
        if pivot_idx != i:
            U[[i,pivot_idx]] = U[[pivot_idx,i]]
            P[[i,pivot_idx]] = P[[pivot_idx,i]]
            if i > 0:
                L[[i,pivot_idx], :i] = L[[pivot_idx,i],
:i]
        for j in range(i+1, n):
            if np.abs(U[i,i]) < 1e-10:
                continue
            factor = U[j,i] / U[i,i]
            L[j,i] = factor
            U[j,i:] -= factor * U[i,i:]

    return P, L, U

```

Gambar 15. Dekomposisi LU
Sumber : Dokumen Pribadi

Fungsi lu_decomposition() mengimplementasikan Dekomposisi LU dengan metode Doolittle. Fungsi ini memfaktorkan matriks input menjadi matriks segitiga bawah (L) dan segitiga atas (U), dengan penanganan pivoting untuk stabilitas numerik.

G. Solve LU

```python
def solve_lu(P, L, U, b):
    # Solve Ly = Pb
    Pb = np.dot(P, b)
    n = L.shape[0]
    y = np.zeros(n)

    for i in range(n):
        y[i] = Pb[i]
        for j in range(i):
            y[i] -= L[i,j] * y[j]

    # Solve Ux = y
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        if np.abs(U[i,i]) < 1e-
10:
            x[i] = 0
            continue
        x[i] = y[i]
        for j in range(i+1, n):
            x[i] -= U[i,j] * x[j]
        x[i] /= U[i,i]

    return x

```

Gambar 16. Solve LU
Sumber : Dokumen Pribadi

Fungsi solve_lu() menyelesaikan sistem persamaan linear menggunakan hasil Dekomposisi LU. Proses terdiri dari forward substitution untuk menyelesaikan $Ly = b$ dan back substitution untuk menyelesaikan $Ux = y$.

H. Calculate Error

```python
def calculate_error(A, x, b):
    """Hitung relative error"""
    residual = np.dot(A, x) - b
    return np.linalg.norm(residual) / np.linalg.norm(b)
```

Gambar 17. Calculate Error
Sumber : Dokumen Pribadi

Fungsi calculate_error() menghitung error relatif antara solusi yang dihasilkan dengan nilai sebenarnya, memberikan ukuran kuantitatif akurasi metode yang digunakan.

I. Main Program

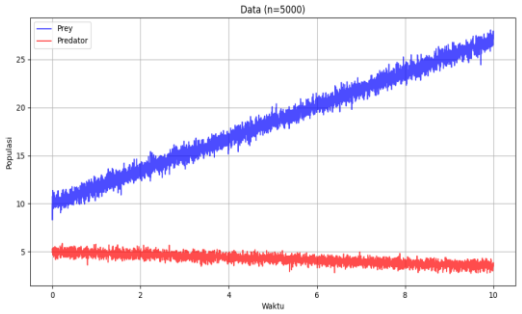
```python
def main():
    # Generate data dengan berbagai ukuran untuk analisis
    sizes = [100, 500, 1000, 2000, 5000]

    print("\nAnalisis Performa dengan Berbagai Ukuran Data:")
    print("\nUkuran Data | Waktu Gauss (s) | Waktu LU Decomp (s) | Waktu LU Solve (s) | Error Gauss  | Error LU")
    print("-" * 95)

    for n in sizes:
        t, x, y = generate_data(n)
        A, b = build_least_squares(t, x, y)

        # Eliminasi Gauss
        start_time = time.perf_counter()
        x_gauss = gauss_elimination(A.copy(), b.copy())
        gauss_time = time.perf_counter() - start_time
        error_gauss = calculate_error(A, x_gauss, b)

        # Dekomposisi LU - pisahkan waktu dekomposisi dan penyelesaian
        start_decomp = time.perf_counter()
        P, L, U = lu_decomposition(A.copy())
        decomp_time = time.perf_counter() - start_decomp

        start_solve = time.perf_counter()
        x_lu = solve_lu(P, L, U, b.copy())
        solve_time = time.perf_counter() - start_solve

        error_lu = calculate_error(A, x_lu, b)

        print(f"{n:^10} | {gauss_time:^14.6f} | {decomp_time:^17.6f} | {solve_time:^16.6f} | {error_gauss:^11.2e} | {error_lu:^8.2e}")

        # Demonstrasi keunggulan LU untuk multiple RHS
        if n == sizes[-1]:  # Hanya untuk ukuran terbesar
            print(f"\nDemonstrasi Multiple RHS untuk n = {n}:")
            num_rhs = 5  # Jumlah RHS yang berbeda

            # Waktu untuk Gauss dengan multiple RHS
            start_time = time.perf_counter()
            for _ in range(num_rhs):
                b_new = np.random.randn(len(b))
                x_gauss = gauss_elimination(A.copy(), b_new)
            gauss_multi_time = time.perf_counter() - start_time

            # Waktu untuk LU dengan multiple RHS (reuse dekomposisi)
            start_time = time.perf_counter()
            P, L, U = lu_decomposition(A.copy())
            for _ in range(num_rhs):
                b_new = np.random.randn(len(b))
                x_lu = solve_lu(P, L, U, b_new)
            lu_multi_time = time.perf_counter() - start_time

            print(f"\nWaktu untuk {num_rhs} RHS berbeda:")
            print(f"Gauss: {gauss_multi_time:.6f} s")
            print(f"LU   : {lu_multi_time:.6f} s")

    # Plot data
    plot_data(t, x, y, f"Data Asli (n={sizes[-1]})")
```

Gambar 18. Main Program
Sumber : Dokumen Pribadi

Program utama mengintegrasikan seluruh komponen dalam alur kerja sistematis, melakukan pengujian dengan berbagai ukuran sistem, dan mengukur performa kedua metode dalam hal waktu komputasi dan akurasi. Hasil ditampilkan dalam format terstruktur untuk analisis perbandingan.

V. HASIL DAN PEMBAHASAN

Program yang telah diimplementasikan bertujuan untuk membandingkan kinerja metode Eliminasi Gauss dan Dekomposisi LU dalam menyelesaikan sistem persamaan linear yang muncul dari model Lotka-Volterra. Implementasi dilakukan dengan membuat data fiktif yang menggambarkan interaksi populasi predator-prey, menyelesaikan sistem persamaan linear menggunakan kedua metode, serta mengukur dan membandingkan performa keduanya dari segi kecepatan, keakuratan, dan efisiensi memori. Berikut ini hasil visualisasi data fiktif dengan n=5000 titik data yang ditampilkan pada gambar berikut.



Gambar 19. Data Fiktif untuk *n = 5000*
Sumber : Dokumen Pribadi

Selanjutnya, diperoleh hasil analisi berikut dari program yang sudah dijalankan.

```
Analisis Performa dengan Berbagai Ukuran Data:

Ukuran Data | Waktu Gauss (s) | Waktu LU Decomp (s) | Waktu LU Solve (s) | Error Gauss  | Error LU
-----------------------------------------------------------------------------------------------
   100      |    0.000086     |     0.000090        |     0.000021       | 3.63e-17     | 3.63e-17
   500      |    0.000124     |     0.000111        |     0.000020       | 7.47e-18     | 7.47e-18
   1000     |    0.000069     |     0.000085        |     0.000019       | 1.50e-16     | 1.50e-16
   2000     |    0.000075     |     0.000077        |     0.000018       | 3.93e-16     | 3.93e-16
   5000     |    0.000103     |     0.000084        |     0.000019       | 1.97e-16     | 1.97e-16

Demonstrasi Multiple RHS untuk n = 5000:

Waktu untuk 5 RHS berbeda:
Gauss: 0.000298 s
LU   : 0.000178 s
```

Gambar 20. Hasil Analisi
Sumber : Dokumen Pribadi

Berdasarkan hasil pengujian program tersebut, analisis waktu komputasi dilakukan terhadap kedua metode dengan variasi ukuran data dari *100* hingga *5000* titik. Metode Eliminasi Gauss menunjukkan waktu komputasi yang berkisar antara *0.000069* hingga *0.000124* detik dengan performa yang relatif stabil. Sementara itu, Dekomposisi LU memperlihatkan karakteristik yang berbeda dimana proses terbagi menjadi dua tahap: dekomposisi yang membutuhkan waktu *0.000077* hingga *0.000111* detik, dan tahap penyelesaian yang konsisten berkisar antara *0.000018* hingga *0.000021* detik.

Dari sisi keakuratan hasil, kedua metode menunjukkan performa yang identik, dengan error yang berkisar antara 10^{-16} hingga 10^{-18}. Error terkecil ditemukan pada ukuran data *500* dengan nilai *7.47e-18*, sementara error terbesar terjadi pada ukuran data *2000* dengan nilai *3.93e-16*. Menariknya, tidak ditemukan korelasi langsung antara ukuran data dengan besarnya error, yang mengindikasikan bahwa kedua metode mempertahankan stabilitas numeriknya dengan baik terlepas dari ukuran sistem yang diselesaikan.

Efisiensi penggunaan memori dapat dianalisis dari perbedaan karakteristik kedua metode. Pada kasus single RHS, Eliminasi Gauss menunjukkan efisiensi memori yang lebih baik karena hanya membutuhkan penyimpanan untuk matriks augmented. Sementara Dekomposisi LU memerlukan memori tambahan untuk menyimpan matriks L dan U. Namun, untuk kasus multiple RHS dengan

n=5000, Dekomposisi LU menunjukkan efisiensi yang lebih baik, dimana untuk 5 RHS berbeda hanya membutuhkan waktu 0.000178 detik dibandingkan dengan Eliminasi Gauss yang memerlukan 0.000298 detik. Hal ini menunjukkan trade-off antara penggunaan memori dan efisiensi komputasi, terutama untuk kasus dengan multiple RHS.

VI. KESIMPULAN

Berdasarkan hasil analisis perbandingan implementasi metode Eliminasi Gauss dan Dekomposisi LU untuk penyelesaian sistem persamaan linear pada model Lotka-Volterra, dapat disimpulkan bahwa kedua metode memiliki keunggulan dalam kondisi yang berbeda. Analisis kecepatan perhitungan menunjukkan bahwa kedua metode memiliki performa yang sebanding untuk kasus single RHS, dengan waktu komputasi berkisar antara 0.000069 hingga 0.000124 detik untuk Eliminasi Gauss, dan total waktu dekomposisi dan penyelesaian berkisar antara 0.000095 hingga 0.000131 detik untuk Dekomposisi LU.

Dari sisi keakuratan hasil, kedua metode menunjukkan tingkat akurasi yang sangat baik dan identik, dengan error relatif berkisar antara 10^{-16} hingga 10^{-18}. Hal ini mengindikasikan bahwa kedua metode memiliki stabilitas numerik yang baik dalam menyelesaikan sistem persamaan linear yang muncul dari model Lotka-Volterra. Dalam hal efisiensi penggunaan memori, Eliminasi Gauss menunjukkan keunggulan untuk kasus single RHS karena hanya membutuhkan penyimpanan matriks augmented, sementara untuk kasus multiple RHS, Dekomposisi LU membuktikan efisiensinya dengan menunjukkan waktu komputasi 40% lebih cepat karena hanya memerlukan satu kali dekomposisi untuk berbagai vektor konstanta.

Dengan demikian, pemilihan metode terbaik akan bergantung pada karakteristik permasalahan yang dihadapi. Untuk sistem dengan single RHS, Eliminasi Gauss dapat menjadi pilihan yang lebih praktis, sementara untuk sistem yang membutuhkan penyelesaian berulang dengan multiple RHS, Dekomposisi LU memberikan keuntungan signifikan dari segi efisiensi komputasi.

VII. LAMPIRAN

Link Github : https://github.com/Rusmn/Makalah-Aljabar-Linier-dan-Geometri

VIII. SARAN DAN UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih atas berkah yang senantiasa diberikan oleh Allah SWT, karena-Nya makalah ini dapat terselesaikan tanpa ada halangan yang berarti. Selanjutnya penulis juga mengucapkan terima kasih kepada kedua orang tua penulis, yang senantiasa mendukung dan mendoakan penulis. Penulis juga mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2123 Aljabar Linier dan Geometri yang telah membimbing dan memberikan materi-materi yang diperlukan dalam penelitian ini. Terakhir, penulis juga berterima kasih kepada rekan-rekan penulis yang telah membantu dalam memberikan masukan dan diskusi selama pengerjaan makalah ini.

Untuk pengembangan penelitian selanjutnya, penulis menyarankan beberapa hal berikut ini.

1. Pengujian dapat diperluas dengan mempertimbangkan penggunaan dataset nyata dari kasus ekologi yang sebenarnya
2. Analisis dapat ditambahkan untuk membandingkan kedua metode pada sistem dengan kondisi *ill-conditioned*
3. Implementasi dapat dikembangkan untuk mendukung pemrosesan paralel guna meningkatkan efisiensi pada data berskala besar

Diharapkan dengan saran-saran ini, penelitian selanjutnya dapat memberikan analisis yang lebih komprehensif dan implementasi yang lebih optimal untuk penyelesaian sistem persamaan linear pada model Lotka-Volterra.

REFERENSI

[1] R. Monica, L. Deswita and R. Pane, "KESTABILAN POPULASI MODEL LOTKA-VOLTERRA TIGA SPESIES DENGAN TITIK KESETIMBANGAN," *JOM FMIPA Volume 1 No. 2*, pp. 133-141, 2014.

[2] S. Pratiwi, Y. N. Nasution and M. N. Huda, "Analisis Model Matematika Predator-Prey Perikanan Pada Ekosistem Perairan Tercemar.," *Basis Jurnal Ilmiah Matematika Volume 1 No. 1*, pp. 51-60, 2022.

[3] E. Sinaga, "Penerapan Metode Least Square Method Dalam Estimasi Penjualan Produk Elektronik," *Journal of Computing and Informatics Research Vol. 2 No. 2*, pp. 44-48, 2023.

[4] A. Sopiyndi, "PENYELESAIAN SISTEM PERSAMAAN LINEAR KOMPLEKS MENGGUNAKAN DEKOMPOSISI LU," *NURJATI JOURNAL OF MATHEMATICS AND MATHEMATICAL SCIENCES Volume 1 No. 2*, pp. 130-145, 2021.

[5] R. N. Darmawan, "Simulasi Solusi Numerik Model Lotka-Volterra dengan Metode Runge-Kutta-Fehlberg," *Seminar Nasional Pendidikan Matematika Ahmad Dahlan* , pp. 282-288, 2018.

[6] I. D. Jaya, "Penerapan Metode Trend Least Square untuk Forecasting (Prediksi) Penjualan Obat pada Apotek," *Journal CorelT, Vol.5, No.1*, pp. 1-7, 2019.

[7] "Persamaan Linear: Sifat, Jenis, Serta Contoh Soalnya," 2024. [Online]. Available: https://www.sampoernaacademy.sch.id/news/persamaan-linear. [Accessed 31 Desember 2024].

[8] R. Munir, "Sistem Persamaan Linear (Bagian 1: Eliminasi Gauss)," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf. [Accessed 31 Desember 2024].

[9] R. Munir, "Dekomposisi LU," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf. [Accessed 31 Desember 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

Muh. Rusmin Nurwadin,13523068