Analisis Prediksi Dinamika Populasi Menggunakan Model Markov Berbasis Diagonalisasi Matriks

Alya Nur Rahmah 13524081[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524081@std.stei.itb.ac.id, [2]alyanurrhmah@gmail.com

***Abstrak*— Dinamika populasi merupakan fenomena yang penting untuk dianalisis karena memiliki peranan penting dalam berbagai bidang, seperti perencanaan sumber daya, perumusan kebijakan publik, serta upaya menjaga keberlanjutan lingkungan. Untuk memahami dan memprediksi perubahan populasi dari waktu ke waktu, diperlukan suatu model matematis yang mampu merepresentasikan transisi antar kondisi populasi secara sistematis. Salah satu pendekatan yang efektif adalah model rantai Markov, yang memodelkan evolusi populasi berdasarkan probabilitas perpindahan antar keadaan melalui suatu matriks transisi. Meskipun demikian, analisis prediksi populasi dalam jangka waktu panjang menuntut perhitungan pangkat matriks transisi, yang secara komputasional dapat menjadi rumit dan tidak efisien apabila dilakukan secara langsung. Oleh karena itu, makalah ini membahas analisis prediksi dinamika populasi menggunakan model Markov dengan pendekatan diagonalisasi matriks. Melalui proses diagonalisasi, matriks transisi direpresentasikan ke dalam bentuk diagonal yang memungkinkan perhitungan pangkat matriks dilakukan dengan lebih sederhana dan terstruktur. Pendekatan ini tidak hanya mempermudah proses komputasi, tetapi juga memberikan kerangka analitis yang lebih jelas dalam mengkaji perilaku populasi jangka panjang, termasuk kestabilan sistem dan kecenderungan menuju keadaan tunak (steady-state). Hasil analisis menunjukkan bahwa penggunaan diagonalisasi matriks dalam model Markov mampu meningkatkan efisiensi perhitungan sekaligus memperdalam pemahaman terhadap struktur matematis dan karakteristik dinamika populasi yang dimodelkan.**

***Kata Kunci*—Diagonalisasi matriks, dinamika populasi, matriks transisi, rantai Markov.**

I. PENDAHULUAN

Dinamika populasi merupakan salah satu permasalahan fundamental yang banyak dikaji dalam berbagai disiplin ilmu, seperti demografi, ekologi, ekonomi, dan ilmu sosial. Perubahan jumlah dan komposisi populasi dari waktu ke waktu memiliki implikasi langsung terhadap perencanaan sumber daya, pembangunan berkelanjutan, serta perumusan kebijakan publik. Oleh karena itu, diperlukan suatu pendekatan analitis yang mampu menggambarkan dan memprediksi perilaku populasi secara sistematis dan terukur.

Pemodelan matematika menjadi alat yang penting dalam menganalisis dinamika populasi. Melalui model matematika, fenomena populasi yang kompleks dapat direpresentasikan ke dalam bentuk yang lebih terstruktur sehingga memungkinkan dilakukan analisis kuantitatif. Salah satu model yang banyak digunakan adalah model rantai Markov. Model ini didasarkan pada asumsi bahwa keadaan sistem pada waktu tertentu hanya bergantung pada keadaan sebelumnya, tanpa dipengaruhi oleh sejarah yang lebih panjang. Asumsi ini, yang dikenal sebagai sifat Markov, menjadikan model ini relatif sederhana namun cukup kuat untuk menggambarkan berbagai fenomena perubahan populasi.

Dalam model Markov, perubahan populasi direpresentasikan menggunakan matriks transisi yang memuat probabilitas perpindahan individu dari satu keadaan ke keadaan lain dalam satu periode waktu. Keadaan-keadaan tersebut dapat berupa kelompok umur, tingkat pendidikan, status kesehatan, atau kategori populasi lainnya, tergantung pada konteks permasalahan yang dikaji. Dengan menggunakan matriks transisi, kondisi populasi pada periode selanjutnya dapat diprediksi melalui operasi perkalian matriks terhadap vektor populasi awal.

Permasalahan muncul ketika prediksi populasi dilakukan untuk jangka waktu yang panjang. Prediksi semacam ini memerlukan perhitungan pangkat matriks transisi yang cukup besar. Perhitungan pangkat matriks secara langsung tidak hanya membutuhkan biaya komputasi yang tinggi, tetapi juga menyulitkan analisis matematis terhadap perilaku sistem dalam jangka panjang. Akibatnya, diperlukan metode alternatif yang mampu menyederhanakan perhitungan sekaligus memberikan pemahaman yang lebih mendalam terhadap karakteristik sistem.

Salah satu pendekatan yang dapat digunakan untuk mengatasi permasalahan tersebut adalah diagonalisasi matriks. Diagonalisasi memungkinkan suatu matriks transisi direpresentasikan ke dalam bentuk matriks diagonal melalui transformasi berbasis nilai eigen dan vektor eigen. Dengan bentuk diagonal ini, perhitungan pangkat matriks menjadi jauh lebih sederhana karena cukup dilakukan terhadap elemen-elemen diagonalnya. Selain meningkatkan efisiensi perhitungan, pendekatan ini juga memberikan landasan teoritis yang kuat untuk menganalisis kestabilan sistem dan kecenderungan populasi menuju keadaan tunak atau steady-state.

Berdasarkan latar belakang tersebut, makalah ini bertujuan untuk membahas analisis prediksi dinamika populasi menggunakan model Markov berbasis diagonalisasi matriks. Pembahasan difokuskan pada konsep dasar model Markov, peran matriks transisi dalam pemodelan populasi, serta pemanfaatan diagonalisasi matriks untuk analisis jangka panjang. Selain itu, makalah ini juga menyajikan implementasi program sebagai ilustrasi penerapan metode yang dibahas. Diharapkan makalah ini dapat memberikan pemahaman yang komprehensif mengenai keterkaitan antara teori aljabar linear dan geometri dalam pemodelan dinamika populasi, serta menunjukkan relevansi penerapannya untuk permasalahan nyata.

II. DASAR TEORI

A. Dinamika Populasi

Dinamika populasi merupakan bidang kajian yang mempelajari perubahan jumlah, distribusi, dan komposisi populasi dalam suatu sistem sepanjang waktu. Perubahan tersebut dipengaruhi oleh berbagai faktor demografis, seperti kelahiran, kematian, serta perpindahan individu antarwilayah atau antarkategori. Kajian dinamika populasi bertujuan untuk memahami pola perubahan tersebut dan kecenderungan yang muncul sebagai akibat dari interaksi antar faktor-faktor tersebut.

Dalam kajian matematika, dinamika populasi sering dimodelkan sebagai sistem diskrit waktu, di mana kondisi populasi diamati pada selang waktu tertentu. Populasi dibagi ke dalam sejumlah keadaan (state) yang merepresentasikan kategori, wilayah, atau kelompok tertentu. Distribusi populasi pada suatu waktu dinyatakan dalam bentuk vektor yang menunjukkan jumlah atau proporsi populasi pada masing-masing keadaan.

Untuk mempermudah analisis, sistem populasi dalam banyak studi diasumsikan sebagai sistem tertutup, yaitu sistem di mana total populasi dianggap konstan selama periode pengamatan. Dengan asumsi ini, perubahan distribusi populasi hanya disebabkan oleh perpindahan individu antar keadaan, sementara faktor kelahiran dan kematian diabaikan. Pendekatan ini memungkinkan fokus analisis diarahkan pada mekanisme perpindahan dan interaksi antar keadaan dalam sistem.

Salah satu aspek penting dalam dinamika populasi adalah perilaku jangka panjang dari distribusi populasi. Dalam kondisi tertentu, meskipun individu terus berpindah antar keadaan, sistem dapat mencapai suatu kondisi keseimbangan dinamis. Pada kondisi ini, distribusi populasi antar keadaan tidak lagi berubah dari waktu ke waktu. Kondisi tersebut dikenal sebagai keadaan tunak atau steady-state.

Keadaan tunak merepresentasikan keseimbangan struktural dalam sistem populasi, di mana arus perpindahan individu antar keadaan saling mengimbangi. Konsep ini menjadi penting dalam kajian dinamika populasi karena memberikan gambaran mengenai distribusi populasi yang stabil dan kecenderungan alami sistem dalam jangka panjang.

B. Rantai Markov dan Matriks Transisi

Rantai Markov merupakan model matematis yang digunakan untuk menggambarkan proses stokastik diskrit, di mana sistem berpindah dari satu keadaan ke keadaan lain dalam langkah waktu tertentu. Ciri utama dari rantai Markov adalah terpenuhinya sifat Markov (*Markov property*), yaitu bahwa probabilitas keadaan sistem pada waktu mendatang hanya bergantung pada keadaan saat ini dan tidak bergantung pada riwayat keadaan sebelumnya.

Secara matematis, suatu proses stokastik diskrit $\{X_t\}$ dikatakan sebagai rantai Markov apabila memenuhi:

$$P\left(X_{\{t+1\}} = j \mid X_t = i, X_{\{t-1\}}, \dots, X_0\right) = P\left(X_{\{t+1\}} = j \mid X_t = i\right),$$

untuk setiap pasangan keadaan (i) dan (j). Keadaan (*state*) dalam rantai Markov merepresentasikan kondisi-kondisi yang mungkin dialami oleh sistem. Dalam kajian populasi, keadaan tersebut dapat berupa wilayah geografis, kelompok umur, atau kategori sosial tertentu.

Rantai Markov banyak digunakan dalam pemodelan dinamika sistem karena kemampuannya menangkap proses transisi yang bersifat acak namun terstruktur. Dengan asumsi bahwa aturan perpindahan antar keadaan bersifat tetap dari waktu ke waktu, rantai Markov menyediakan kerangka formal untuk menganalisis evolusi distribusi sistem secara probabilistik.

Representasi utama dari rantai Markov dilakukan melalui matriks transisi. Matriks transisi adalah matriks persegi yang elemen-elemennya menyatakan probabilitas perpindahan dari satu keadaan ke keadaan lain dalam satu langkah waktu. Jika sistem memiliki (n) keadaan, maka matriks transisi dinyatakan sebagai matriks berukuran $(n \times n)$:

$$\mathrm{P} = \left[\mathrm{p_{ij}}\right],$$

dengan $p_{ij} = P(X_{t+1} = j \mid X_t = i)$.

Matriks transisi memiliki sifat-sifat khusus, yaitu:

1. Setiap elemen matriks bernilai tidak negatif, $p_{ij} \geq 0$.
2. Jumlah elemen pada setiap baris sama dengan satu:
   $\sum_{j=1}^{n} p_{ij} = 1$.

Sifat-sifat tersebut mencerminkan karakter probabilistik dari proses transisi, di mana seluruh kemungkinan perpindahan dari suatu keadaan tercakup secara lengkap. Dalam konteks dinamika populasi, matriks transisi merepresentasikan pola perpindahan individu antar keadaan dalam satu periode waktu.

Distribusi populasi pada waktu ke-(t) dapat dinyatakan dalam bentuk vektor status x_t. Evolusi distribusi populasi dari satu waktu ke waktu berikutnya dinyatakan melalui transformasi linear

$$x_{t+1} = P x_t$$

Dengan melakukan iterasi, distribusi populasi pada waktu ke-k dapat dituliskan sebagai:

$$\mathrm{x_k} = P^k x_0,$$

di mana x_0 merupakan distribusi populasi awal.

Analisis terhadap matriks transisi memungkinkan kajian lebih lanjut mengenai perilaku jangka panjang sistem, khususnya terkait keberadaan dan sifat distribusi yang tidak berubah terhadap transformasi matriks tersebut. Hal

ini berkaitan erat dengan konsep keadaan tunak, nilai eigen, dan vektor eigen dari matriks transisi, yang merupakan bagian penting dalam teori rantai Markov.

D. Nilai Eigen dan Vektor Eigen

Misalkan A adalah matriks persegi berukuran $n \times n$. Suatu vektor tak-nol $x \in R^n$ disebut vektor eigen dari A jika hasil perkalian A dengan x hanya mengubah panjang atau arah vektor tersebut dengan faktor skalar, yaitu:

$$Ax = \lambda x,$$

di mana λ adalah suatu skalar yang disebut nilai eigen yang berkorespondensi dengan vektor eigen x. Pernyataan ini menunjukkan bahwa transformasi linear yang direpresentasikan oleh A hanya "memperbesar", "memperkecil", atau "membalik" arah vektor x, tanpa mengubah arah vektor itu sendiri secara relatif.

Untuk menentukan nilai-nilai eigen dari suatu matriks A, diturunkan persamaan karakteristik dari kondisi $A\mathrm{x} = \lambda\mathrm{x}$. Mengalihkan kedua ruas dengan matriks identitas Idan memindahkan suku menghasilkan :

$$(A - \lambda I)x = 0.$$

Agar persamaan ini memiliki solusi tak-trivial (vektor $x \neq 0$), determinan dari koefisien matriks harus sama dengan nol, sehingga diperoleh *persamaan karakteristik*:

$$\det(\lambda I - A) = 0,$$

yang merupakan polinom dalam λ. Akar-akar dari persamaan karakteristik ini merupakan nilai-nilai eigen dari matriks A, dan untuk setiap nilai eigen λ, vektor-vektor yang memenuhi $(A - \lambda I)x = 0$adalah vektor-vektor eigen yang berkorespondensi.

Suatu nilai eigen λ dapat berulang secara aljabar, dan setiap nilai eigen memiliki himpunan vektor eigen yang bersesuaian, yang membentuk sebuah subruang vektor yang disebut ruang eigen (*eigenspace*). Ruang eigen untuk suatu nilai eigen λdidefinisikan sebagai

$$E(\lambda) = \{\mathrm{x} \in \mathbb{R}^n \mid (A - \lambda I)x = 0\}.$$

Konsep nilai eigen dan vektor eigen memiliki sejumlah sifat penting yang sering digunakan dalam analisis matriks:

1. Nilai-nilai eigen dari suatu matriks persegi Amerupakan akar-akar dari polinom karakteristik $\det(\lambda I - A)$, yang berderajat n.
2. Jika suatu matriks A memiliki invers, maka tidak terdapat nilai eigen yang bernilai nol, karena adanya nilai eigen nol berarti determinan Asama dengan nol.
3. Nilai eigen mempengaruhi perilaku transformasi linear yang direpresentasikan oleh matriks, khususnya dalam konteks sistem dinamis. Nilai eigen dengan nilai mutlak lebih besar dari satu menunjukkan komponen pertumbuhan, sedangkan nilai eigen dengan nilai mutlak kurang dari satu menunjukkan arah yang memudar.

Dalam bahasa geometris, vektor-vektor eigen memberikan arah di mana transformasi linear yang diberikan oleh A hanya mengubah skala vektor tanpa memutar arah. Hal ini membuat nilai eigen dan vektor eigen menjadi alat analitis yang penting untuk memahami karakteristik struktur matriks, terutama ketika matriks tersebut dipangkatkan atau digunakan dalam dekomposisi matriks, seperti pada diagonalisasi

E. Diagonalisasi Matriks

Diagonalisasi matriks merupakan konsep penting dalam aljabar linear yang berkaitan langsung dengan nilai eigen dan vektor eigen. Suatu matriks persegi dikatakan dapat didiagonalisasi apabila matriks tersebut dapat dinyatakan dalam bentuk matriks diagonal melalui transformasi kesebangunan (*similarity transformation*). Diagonalisasi bertujuan untuk merepresentasikan suatu matriks dalam bentuk yang lebih sederhana tanpa mengubah sifat-sifat aljabarnya yang mendasar.

Secara formal, sebuah matriks persegi $A \in \mathbb{R}^{n \times n}$ dikatakan dapat didiagonalisasi jika terdapat matriks invertibel P dan matriks diagonal Dsedemikian sehingga

$$A = PDP^{-1}.$$

Matriks diagonal D berisi nilai-nilai eigen dari matriks A pada elemen-elemen diagonal utamanya, sedangkan matriks P dibentuk dari vektor-vektor eigen yang bersesuaian sebagai kolom-kolomnya.

Keberhasilan proses diagonalisasi sangat bergantung pada keberadaan vektor eigen yang saling bebas linear. Suatu matriks berukuran $n \times n$ dapat didiagonalisasi jika dan hanya jika matriks tersebut memiliki n vektor eigen yang saling bebas linear

Jika suatu matriks memiliki nilai eigen yang semuanya berbeda, maka matriks tersebut pasti dapat didiagonalisasi. Namun, apabila terdapat nilai eigen yang berulang, matriks belum tentu dapat didiagonalisasi. Dalam kasus tersebut, perlu diperiksa apakah jumlah vektor eigen bebas linear yang dihasilkan mencukupi untuk membentuk basis ruang vektor berdimensi n.

Salah satu sifat penting dari matriks yang dapat didiagonalisasi adalah kegunaannya dalam menghitung perpangkatan matriks. Jika $A = PDP^{-1}$, maka untuk setiap bilangan bulat positif $k$berlaku

$$A^k = PD^kP^{-1},$$

dengan D^kmerupakan matriks diagonal yang elemen-elemennya adalah pangkat ke-k dari nilai eigen yang bersesuaian.

III. PEMBAHASAN

Dinamika populasi berkaitan erat dengan perubahan distribusi individu dalam suatu sistem dari waktu ke waktu. Perubahan tersebut dapat terjadi akibat perpindahan antar wilayah, perubahan status demografis, maupun faktor sosial dan ekonomi lainnya. Untuk memahami dan memprediksi pola perubahan tersebut secara sistematis, diperlukan suatu model matematika yang mampu merepresentasikan perpindahan populasi secara terstruktur dan konsisten. Salah satu pendekatan yang dapat digunakan adalah model rantai Markov, yang memandang dinamika populasi sebagai proses diskrit dengan peluang perpindahan tertentu antar keadaan.

Model rantai Markov memungkinkan dinamika populasi dinyatakan dalam bentuk hubungan matematis yang sederhana namun kuat, yaitu melalui matriks transisi. Dengan menggunakan pendekatan ini, distribusi populasi pada suatu waktu dapat diprediksi berdasarkan distribusi pada waktu sebelumnya. Namun, prediksi untuk jangka waktu yang panjang menuntut analisis yang

lebih mendalam terhadap struktur matriks transisi tersebut. Oleh karena itu, konsep nilai eigen, vektor eigen, dan diagonalisasi matriks digunakan untuk menyederhanakan analisis sekaligus memberikan pemahaman mengenai perilaku jangka panjang populasi yang dimodelkan.

A. Pemodelan Dinamika Populasi dan Formulasi Rantai Markov

Analisis dinamika populasi dalam makalah ini dimulai dengan memodelkan populasi sebagai suatu sistem diskrit waktu yang berkembang secara bertahap. Populasi dibagi ke dalam sejumlah keadaan (*state*) yang saling eksklusif dan mencakup seluruh sistem. Setiap keadaan dapat merepresentasikan kategori tertentu, seperti wilayah geografis, kelompok umur, atau klasifikasi demografis lainnya.

Distribusi populasi pada waktu ke-t dinyatakan dalam bentuk vektor keadaan:

$$x_t = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \\ \vdots \\ x_n^{(t)} \end{bmatrix},$$

dengan $x_i^{(t)}$ menyatakan jumlah atau proporsi populasi pada keadaan ke-i pada waktu tersebut. Dalam pembahasan ini diasumsikan bahwa sistem bersifat tertutup, sehingga total populasi tetap konstan sepanjang waktu. Secara matematis, asumsi ini dinyatakan sebagai:

$$\sum_{i=1}^{n} x_i^{(t)} = C,$$

dengan C merupakan konstanta yang menyatakan total populasi.

Perubahan distribusi populasi dari satu waktu ke waktu berikutnya diasumsikan mengikuti sifat Markov, yaitu bahwa keadaan populasi pada waktu selanjutnya hanya bergantung pada keadaan saat ini. Berdasarkan asumsi tersebut, mekanisme perpindahan antar keadaan dapat direpresentasikan secara matematis menggunakan matriks transisi Markov.

Matriks transisi $P = [p_{ij}]$ didefinisikan dengan p_{ij} sebagai probabilitas individu berpindah dari keadaan ke-j menuju keadaan ke-i dalam satu satuan waktu, sehingga:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}.$$

Matriks ini memenuhi sifat matriks stokastik, yaitu setiap elemen bernilai tidak negatif dan jumlah elemen pada setiap kolom bernilai satu yang dinyatakan dengan :

$$p_{ij} \geq 0 \text{ dan } \sum_{j=1}^{n} p_{ij} = 1$$

untuk setiap baris ke-i. Sifat ini menjamin bahwa seluruh individu dari suatu keadaan akan berpindah ke salah satu keadaan yang tersedia.

Dengan demikian, dinamika populasi dapat dirumuskan sebagai transformasi linear:

$$x_{t+1} = Px_t.$$

Relasi ini menjadi dasar utama dalam analisis dinamika populasi berbasis rantai Markov.

B. Prediksi Dinamika Populasi dan Analisis Nilai Eigen

Dengan menggunakan matriks transisi, evolusi distribusi populasi dapat dinyatakan sebagai transformasi linear:

$$x_{t+1} = Px_t.$$

Persamaan ini menunjukkan bahwa distribusi populasi pada waktu berikutnya diperoleh dengan mengalikan matriks transisi dengan vektor distribusi populasi saat ini.

Dengan melakukan iterasi, distribusi populasi pada waktu ke-k dapat dituliskan sebagai:

$$x_k = P^k x_0,$$

di mana x_0 merupakan distribusi populasi awal. Persamaan ini menjadi dasar bagi analisis prediksi dinamika populasi dalam jangka panjang.

Perhitungan langsung P^k untuk nilai k yang besar tidak efisien dan tidak memberikan gambaran struktural mengenai perilaku sistem. Oleh karena itu, analisis dilakukan dengan memanfaatkan nilai eigen dan vektor eigen dari matriks transisi. Nilai eigen λ dan vektor eigen v dari matriks Pdidefinisikan melalui persamaan:

$$P\mathrm{v} = \lambda\mathrm{v}.$$

Dalam konteks rantai Markov, matriks transisi selalu memiliki nilai eigen $\lambda = 1$. Nilai eigen ini memiliki peran penting karena berkaitan dengan kestabilan sistem dan kemungkinan tercapainya distribusi populasi yang tidak berubah terhadap waktu. Nilai eigen lainnya menentukan laju perubahan dan kecepatan konvergensi sistem menuju kondisi stabil tersebut.

C. Diagonalisasi Matriks dan Keadaan Tunak Dinamika Populasi

Apabila matriks transisi P memiliki cukup banyak vektor eigen yang saling bebas linear, maka matriks tersebut dapat didiagonalisasi. Secara matematis, hal ini dinyatakan sebagai:

$$P = VDV^{-1},$$

di mana:

- D adalah matriks diagonal yang elemen-elemen diagonalnya merupakan nilai eigen dari P,
- V adalah matriks yang kolom-kolomnya merupakan vektor-vektor eigen dari P.

Dengan bentuk ini, pemangkatan matriks P dapat disederhanakan menjadi:

$$P^k = VD^kV^{-1}.$$

Karena D merupakan matriks diagonal, pemangkatan D^k dilakukan dengan menaikkan setiap elemen diagonal ke pangkat k:

$$D^k = \begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^k \end{bmatrix}.$$

Bentuk ini memungkinkan analisis dinamika populasi dilakukan secara sistematis karena kontribusi masing-masing nilai eigen terhadap distribusi populasi dapat diamati secara terpisah.

Distribusi populasi dikatakan mencapai keadaan tunak (*steady-state*) apabila terdapat vektor x^*yang memenuhi:

$$Px^* = x^*.$$

Secara aljabar, keadaan tunak merupakan vektor eigen dari matriks transisi yang bersesuaian dengan nilai eigen $\lambda = 1$. Jika semua nilai eigen lainnya memenuhi $| \lambda_i | < 1$, maka untuk setiap distribusi awal $x_0$berlaku:

$$\lim_{k\to\infty} P^k x_0 = x^*.$$

Hal ini menunjukkan bahwa distribusi populasi akan konvergen menuju keadaan tunak, terlepas dari kondisi awalnya.

Dalam konteks dinamika populasi, keadaan tunak merepresentasikan distribusi populasi yang stabil, di mana proporsi populasi di setiap keadaan tidak lagi berubah meskipun waktu terus berjalan.

Pembahasan keadaan tunak menjadi penting karena memberikan gambaran kondisi jangka panjang populasi. Dengan mengetahui keadaan tunak, pembuat kebijakan atau perencana sumber daya dapat memahami kecenderungan populasi tanpa harus melakukan simulasi berulang dalam jangka waktu yang panjang.

Hasil ini menunjukkan bahwa dinamika populasi akan konvergen menuju suatu distribusi stabil yang ditentukan oleh struktur matriks transisi, bukan oleh kondisi awal populasi. Dengan demikian, pendekatan diagonalisasi matriks memberikan pemahaman matematis yang kuat mengenai perilaku jangka panjang dan kestabilan sistem populasi yang dimodelkan.

Secara keseluruhan, pembahasan ini menunjukkan bahwa dinamika populasi dapat dianalisis sebagai sistem linear diskrit yang evolusinya dikendalikan oleh matriks transisi Markov. Dengan mengekspresikan perpindahan antar keadaan secara eksplisit dalam bentuk matriks, serta memanfaatkan konsep nilai eigen dan diagonalisasi, prediksi dinamika populasi jangka panjang dapat dilakukan secara sistematis dan matematis. Pendekatan ini tidak hanya memberikan efisiensi dalam perhitungan, tetapi juga memberikan pemahaman struktural mengenai kestabilan dan kecenderungan distribusi populasi yang dimodelkan.

IV. IMPLEMENTASI PROGRAM

Implementasi ini bertujuan untuk menunjukkan bagaimana model rantai Markov yang direpresentasikan dalam bentuk matriks transisi dapat digunakan secara praktis dalam memprediksi distribusi populasi pada beberapa periode waktu ke depan, serta dalam menentukan keadaan tunak yang dicapai oleh sistem. Program implementasi ini menggunakan bahasa pemrograman Python dengan pustaka numpy dan matplotlib. Pustaka NumPy digunakan untuk melakukan perhitungan vektor dan matriks, termasuk perkalian matriks dan perhitungan nilai eigen yang diperlukan dalam analisis diagonalisasi, sedangkan pustaka Matplotlib dimanfaatkan untuk memvisualisasikan hasil prediksi distribusi populasi agar pola konvergensi dan kestabilan sistem dapat diamati secara lebih jelas. Melalui implementasi program ini, diperoleh hasil berupa prediksi distribusi populasi dari waktu ke waktu serta distribusi populasi jangka panjang yang merepresentasikan keadaan tunak sistem, sehingga analisis teoretis yang dilakukan sebelumnya dapat dibuktikan dan dipahami secara lebih konkret.

A. Pendefinisian Data Awal dan Studi Kasus

Implementasi program diawali dengan pendefinisian studi kasus dan data awal populasi. Studi kasus yang digunakan adalah migrasi penduduk antar tiga wilayah dalam sebuah kota metropolitan, yaitu Wilayah 1 (pusat kota), Wilayah 2 (suburban), dan Wilayah 3 (pinggiran). Pembagian wilayah ini merepresentasikan kondisi nyata di mana perpindahan penduduk dipengaruhi oleh faktor ekonomi, aksesibilitas, dan kualitas hidup.

Distribusi populasi awal dinyatakan dalam bentuk vektor:

$$x_0 = \begin{bmatrix} 1000 \\ 800 \\ 600 \end{bmatrix},$$

yang menyatakan jumlah penduduk (dalam ribuan jiwa) pada masing-masing wilayah. Penggunaan vektor kolom ini sesuai dengan pemodelan dinamika populasi menggunakan aljabar linear.

Selanjutnya, perpindahan penduduk antar wilayah dimodelkan menggunakan matriks transisi Markov P, di mana setiap elemen P_{ij}menyatakan probabilitas penduduk berpindah dari wilayah ke-j ke wilayah ke-i dalam satu periode waktu. Matriks transisi yang digunakan adalah:

$$P = \begin{bmatrix} 0.80 & 0.10 & 0.15 \\ 0.15 & 0.85 & 0.10 \\ 0.05 & 0.05 & 0.75 \end{bmatrix}.$$

Interpretasi dari matriks tersebut adalah :

- 80% penduduk W1 tetap, 15% ke W2, 5% ke W3
- 85% penduduk W2 tetap, 10% ke W1, 5% ke W3
- 75% penduduk W3 tetap, 15% ke W1, 10% ke W2

```python
def definisi_studi_kasus():
    # Definisi populasi awal
    x0 = np.array([1000, 800, 600], dtype=float)  # dalam ribuan jiwa

    print("\nPOPULASI AWAL (x₀):")
    print("-" * 70)
    print(f"  Wilayah 1 (Pusat Kota) : {x0[0]:>8.0f} ribu jiwa")
    print(f"  Wilayah 2 (Suburban)   : {x0[1]:>8.0f} ribu jiwa")
    print(f"  Wilayah 3 (Pinggiran)  : {x0[2]:>8.0f} ribu jiwa")
    print(f"  {'-'*40}")
    print(f"  Total Populasi         : {np.sum(x0):>8.0f} ribu jiwa")

    # Definisi matriks transisi
    # P[i,j] = probabilitas berpindah dari wilayah j ke wilayah i
    P = np.array([
        [0.80, 0.10, 0.15],  # ke Wilayah 1
        [0.15, 0.85, 0.10],  # ke Wilayah 2
        [0.05, 0.05, 0.75]   # ke Wilayah 3
    ])

    print("\nMATRIKS TRANSISI (P):")
    print("-" * 70)
    print("P[i,j] = probabilitas berpindah dari wilayah j ke wilayah i\n")
    print("            Dari W1    Dari W2    Dari W3")
    print("          ┌                                    ┐")
    for i in range(3):
        print(f"  Ke W{i+1} │  {P[i,0]:.2f}       {P[i,1]:.2f}       {P[i,2]:.2f}    │")
    print("          └                                    ┘")

    # Verifikasi sifat matriks stokastik
    print("\nVERIFIKASI MATRIKS STOKASTIK:")
    print("-" * 70)
    col_sums = np.sum(P, axis=0)
    print(f"  Jumlah setiap kolom: {col_sums}")
    print(f"  Semua kolom = 1? {np.allclose(col_sums, 1)}")
    print("  Matriks memenuhi sifat stokastik kolom")

    return P, x0
```

Gambar 4.1 Fungsi definisi_studi_kasus
Sumber : *Dokumen Penulis*

Pada kode program, pendefinisian studi kasus dilakukan melalui fungsi definisi_studi_kasus(). Fungsi ini bertugas untuk:

1. Menjelaskan konteks permasalahan secara naratif,
2. Mendefinisikan vektor populasi awal,
3. Membentuk matriks transisi Markov,
4. Memverifikasi bahwa matriks transisi memenuhi sifat stokastik.

B. Prediksi Distribusi Populasi Beberapa Periode ke Depan

Setelah data awal dan matriks transisi didefinisikan, tahap berikutnya adalah melakukan prediksi distribusi populasi untuk beberapa periode waktu ke depan.

```python
def prediksi_iteratif(P, x0, n_periode):
    print("\n" + "="*70)
    print("BAGIAN B.1: PREDIKSI DISTRIBUSI POPULASI (Metode Iterasi)")
    print("="*70)

    print(f"Prediksi untuk {n_periode} periode ke depan\n")

    # Simulasi iteratif
    riwayat_populasi = [x0.copy()]

    print("{:<8} {:>15} {:>15} {:>15} {:>15}".format(
        "Periode", "Wilayah 1", "Wilayah 2", "Wilayah 3", "Total"))
    print("-" * 70)

    for t in range(n_periode + 1):
        if t == 0:
            x_t = x0
        else:
            x_t = P @ riwayat_populasi[-1]
            riwayat_populasi.append(x_t)

        print("{:<8} {:>15.2f} {:>15.2f} {:>15.2f} {:>15.2f}".format(
            t, x_t[0], x_t[1], x_t[2], np.sum(x_t)))

    return riwayat_populasi
```

Gambar 4.2 Fungsi prediksi_iteratif
Sumber : *Dokumen Penulis*

Prediksi distribusi populasi jangka pendek dilakukan melalui fungsi prediksi_iteratif(P, x0, n_periode). Fungsi ini mengimplementasikan persamaan di atas secara iteratif, dimulai dari vektor populasi awal x_0. Pada setiap iterasi, vektor populasi periode sebelumnya dikalikan dengan matriks transisi untuk memperoleh distribusi periode berikutnya. Fungsi ini digunakan untuk menunjukkan evolusi populasi secara bertahap, mengamati perubahan distribusi penduduk pada setiap periode, serta memverifikasi kestabilan total populasi.

Hasil iterasi ditampilkan dalam bentuk tabel yang memuat distribusi populasi pada setiap wilayah dan total populasi pada setiap periode, sehingga memudahkan analisis dinamika populasi secara kuantitatif.

Untuk memahami perilaku jangka panjang sistem, program selanjutnya menghitung nilai eigen dan vektor eigen dari matriks transisi melalui fungsi analisis_nilai_eigen(P). Nilai eigen digunakan untuk menentukan kestabilan sistem dan mengidentifikasi apakah sistem akan konvergen menuju suatu keadaan tertentu.

```python
def analisis_nilai_eigen(P):
    print("\n" + "="*70)
    print("BAGIAN B.2: ANALISIS NILAI EIGEN DAN VEKTOR EIGEN")
    print("="*70)

    # Hitung nilai eigen dan vektor eigen
    eigenvalues, eigenvectors = np.linalg.eig(P)

    print("\nNILAI-NILAI EIGEN (λ):")
    print("-" * 70)
    for i, val in enumerate(eigenvalues):
        if np.abs(val.imag) < 1e-10:
            print(f"  λ_{i+1} = {val.real:>10.6f}")
        else:
            print(f"  λ_{i+1} = {val.real:>10.6f} + {val.imag:.6f}i")

    print("\nVEKTOR-VEKTOR EIGEN:")
    print("-" * 70)
    for i in range(len(eigenvalues)):
        print(f"\nVektor eigen v_{i+1} (untuk λ_{i+1}):")
        if np.all(np.abs(eigenvectors[:, i].imag) < 1e-10):
            vektor = eigenvectors[:, i].real
            print(f"  [{vektor[0]:>10.6f}]")
            print(f"  [{vektor[1]:>10.6f}]")
            print(f"  [{vektor[2]:>10.6f}]")
        else:
            print(f"  {eigenvectors[:, i]}")

    return eigenvalues, eigenvectors
```

Gambar 4.3 Fungsi analisis_nilai_eigen
Sumber : *Dokumen Penulis*

Berdasarkan hasil nilai eigen tersebut, dilakukan proses diagonalisasi matriks melalui fungsi diagonalisasi_matriks(P, eigenvalues, eigenvectors). Diagonalisasi didasarkan pada persamaan:

$$P = VDV^{-1},$$

dengan Dmerupakan matriks diagonal yang berisi nilai eigen dan Vmerupakan matriks yang kolom-kolomnya adalah vektor eigen. Program juga memverifikasi keberhasilan diagonalisasi dengan merekonstruksi matriks Pdan menghitung galat maksimum antara matriks asli dan hasil rekonstruksi.

```python
def diagonalisasi_matriks(P, eigenvalues, eigenvectors):
    print("\n" + "="*70)
    print("BAGIAN B.3: DIAGONALISASI MATRIKS")
    print("="*70)

    print("\nFormula: P = V D V⁻¹")
    print("  • D = matriks diagonal berisi nilai eigen")
    print("  • V = matriks yang kolom-kolomnya adalah vektor eigen")

    # Matriks V (vektor eigen sebagai kolom)
    V = eigenvectors
    print("\nMatriks V (Vektor Eigen):")
    print("-" * 70)
    print(V.real)

    # Matriks D (nilai eigen pada diagonal)
    D = np.diag(eigenvalues)
    print("\nMatriks D (Nilai Eigen Diagonal):")
    print("-" * 70)
    print(D.real)

    # Inverse dari V
    V_inv = np.linalg.inv(V)
    print("\nMatriks V⁻¹:")
    print("-" * 70)
    print(V_inv.real)

    # Verifikasi: P = V D V^(-1)
    P_reconstructed = (V @ D @ V_inv).real
    error = np.max(np.abs(P - P_reconstructed))

    print("\nVERIFIKASI: P = V D V⁻¹")
    print("-" * 70)
    print("Matriks P hasil rekonstruksi:")
    print(P_reconstructed)
    print(f"\nError maksimum: {error:.2e}")

    if error < 1e-10:
        print("Diagonalisasi berhasil (error < 10⁻¹⁰)")

    return V, D, V_inv
```

Gambar 4.4 Fungsi diagonalisasi_matriks
Sumber : *Dokumen Penulis*

Selanjutnya, fungsi prediksi_jangka_panjang() memanfaatkan hasil diagonalisasi untuk menghitung distribusi populasi pada periode yang lebih besar menggunakan:

$$x_k = P^k x_0 = V D^k V^{-1} x_0.$$

Pendekatan ini lebih efisien secara komputasi dibandingkan metode iteratif, khususnya untuk nilai k yang besar.

```
def prediksi_jangka_panjang(P, x0, eigenvalues, V, V_inv, periode_list):
    print("\n" + "="*70)
    print("BAGIAN B.4: PREDIKSI JANGKA PANJANG (Metode Diagonalisasi)")
    print("="*70)

    print("\n{:<12} {:>18} {:>18} {:>18}".format(
        "Periode ke-", "Wilayah 1", "Wilayah 2", "Wilayah 3"))
    print("-" * 70)

    hasil_prediksi = {}

    for k in periode_list:
        # Hitung D^k
        D_k = np.diag(eigenvalues ** k)

        # Hitung P^k = V D^k V^(-1)
        P_k = (V @ D_k @ V_inv).real

        # Hitung x_k = P^k x_0
        x_k = P_k @ x0

        hasil_prediksi[k] = x_k

        print("{:<12} {:>18.2f} {:>18.2f} {:>18.2f}".format(
            k, x_k[0], x_k[1], x_k[2]))

    return hasil_prediksi
```

Gambar 4.5 Fungsi prediksi_jangka_panjang
Sumber : *Dokumen Penulis*

C. Penentuan Keadaan Tunak

Tahap akhir dari implementasi program adalah penentuan keadaan tunak (*steady state*) dari sistem dinamika populasi. Keadaan tunak didefinisikan sebagai distribusi populasi x^*yang memenuhi:

$$Px^* = x^*.$$

```
def hitung_keadaan_tunak(P, x0, eigenvalues, eigenvectors):
    print("\n" + "="*70)
    print("BAGIAN C: PENENTUAN KEADAAN TUNAK (STEADY STATE)")
    print("="*70)

    # Cari indeks nilai eigen = 1
    idx_steady = np.argmax(np.abs(eigenvalues - 1) < 1e-10)
    steady_eigenvector = eigenvectors[:, idx_steady].real

    # Normalisasi agar total populasi tetap sama dengan populasi awal
    steady_state = steady_eigenvector / np.sum(steady_eigenvector) * np.sum(x0)

    print(f"\nPENCARIAN NILAI EIGEN λ = 1:")
    print("-" * 70)
    print(f"  Nilai eigen ke-{idx_steady+1}: λ_{idx_steady+1} = {eigenvalues[idx_steady].real:.6f}")
    print(f"  Nilai eigen ini sama dengan 1 (|λ - 1| < 10⁻¹⁰)")

    print(f"\nVEKTOR EIGEN UNTUK λ = 1:")
    print("-" * 70)
    print(f"  v = [{steady_eigenvector[0]:>10.6f}]")
    print(f"      [{steady_eigenvector[1]:>10.6f}]")
    print(f"      [{steady_eigenvector[2]:>10.6f}]")

    print(f"\nKEADAAN TUNAK (STEADY STATE):")
    print("-" * 70)
    print(f"{'Wilayah':<15} {'Populasi':<20} {'Proporsi':<15}")
    print("-" * 70)
    total = np.sum(steady_state)
    for i in range(3):
        proporsi = steady_state[i] / total * 100
        print(f"Wilayah {i+1:<7} {steady_state[i]:>15.2f} ribu  {proporsi:>10.2f}%")
    print("-" * 70)
    print(f"{'Total':<15} {total:>15.2f} ribu  {100.0:>10.2f}%")

    # Verifikasi: P * x* = x*
    print("\nVERIFIKASI: P × x* = x*")
    print("-" * 70)
    P_x_steady = P @ steady_state
    error = np.max(np.abs(P_x_steady - steady_state))
    print(f"Error maksimum: {error:.2e}")

    if error < 1e-10:
        print("Verifikasi berhasil: P × x* = x* (error < 10⁻¹⁰)")

    return steady_state
```

Gambar 4.6 Fungsi hitung_keadaan_tunak
Sumber : *Dokumen Penulis*

Penentuan keadaan tunak dilakukan melalui fungsi hitung_keadaan_tunak(P, x0, eigenvalues, eigenvectors). Fungsi ini mencari vektor eigen yang bersesuaian dengan nilai eigen $\lambda = 1$, kemudian menormalisasikannya agar total populasi tetap sama dengan populasi awal. Hasil normalisasi tersebut merupakan distribusi populasi pada keadaan tunak.

Program juga melakukan verifikasi numerik dengan menghitung selisih antara Px^*dan x^*. Galat yang sangat kecil menunjukkan bahwa keadaan tunak telah diperoleh dengan benar.

```
def analisis_kecepatan_konvergensi(eigenvalues, x0, steady_state):
    print("\n" + "="*70)
    print("ANALISIS KECEPATAN KONVERGENSI")
    print("="*70)

    # Urutkan nilai eigen berdasarkan nilai absolutnya
    eigenvalues_sorted = sorted(eigenvalues, key=lambda x: abs(x), reverse=True)
    lambda_1 = eigenvalues_sorted[0]
    lambda_2 = eigenvalues_sorted[1]

    print("\nNILAI EIGEN DOMINAN:")
    print("-" * 70)
    print(f"  λ₁ (dominan pertama)  = {abs(lambda_1):.6f}")
    print(f"  λ₂ (dominan kedua)    = {abs(lambda_2):.6f}")

    # Estimasi waktu untuk mencapai konvergensi tertentu
    if abs(lambda_2) > 0 and abs(lambda_2) < 1:
        # Waktu untuk error berkurang menjadi ε kali nilai awal
        epsilon_90 = 0.10  # 90% konvergensi
        epsilon_95 = 0.05  # 95% konvergensi
        epsilon_99 = 0.01  # 99% konvergensi

        time_90 = np.log(epsilon_90) / np.log(abs(lambda_2))
        time_95 = np.log(epsilon_95) / np.log(abs(lambda_2))
        time_99 = np.log(epsilon_99) / np.log(abs(lambda_2))

        print("\nESTIMASI WAKTU KONVERGENSI:")
        print("-" * 70)
        print(f"  90% konvergensi: ~{time_90:.0f} periode")
        print(f"  95% konvergensi: ~{time_95:.0f} periode")
        print(f"  99% konvergensi: ~{time_99:.0f} periode")
```

Gambar 4.7 Prosedur analisis_kecepatan_konvergensi
Sumber : *Dokumen Penulis*

Sebagai analisis tambahan, fungsi analisis_kecepatan_konvergensi() digunakan untuk mengkaji kecepatan sistem dalam mencapai keadaan tunak. Analisis ini didasarkan pada nilai eigen dominan kedua, di mana semakin kecil nilai absolutnya, semakin cepat sistem mencapai kesetimbangan.

```
def visualisasi_dinamika_populasi(P, x0, steady_state, n_simulasi=50):
    print("\n" + "="*70)
    print("VISUALISASI DINAMIKA POPULASI")
    print("="*70)

    # Simulasi untuk visualisasi
    pop_sim = [x0.copy()]
    for t in range(n_simulasi):
        pop_sim.append(P @ pop_sim[-1])
    pop_sim = np.array(pop_sim)

    # Buat figure dengan 4 subplot
    fig, axes = plt.subplots(2, 2, figsize=(14, 10))
    fig.suptitle('Analisis Dinamika Populasi dengan Model Markov',
                 fontsize=16, fontweight='bold', y=0.995)

    # Plot 1: Evolusi Populasi per Wilayah
    ax1 = axes[0, 0]
    ax1.plot(range(n_simulasi + 1), pop_sim[:, 0], 'b-', linewidth=2, label='Wilayah 1', marker='o', markersize=3, markevery=5)
    ax1.plot(range(n_simulasi + 1), pop_sim[:, 1], 'g-', linewidth=2, label='Wilayah 2', marker='s', markersize=3, markevery=5)
    ax1.plot(range(n_simulasi + 1), pop_sim[:, 2], 'r-', linewidth=2, label='Wilayah 3', marker='^', markersize=3, markevery=5)
    ax1.axhline(y=steady_state[0], color='b', linestyle='--', alpha=0.5, label='Steady W1')
    ax1.axhline(y=steady_state[1], color='g', linestyle='--', alpha=0.5, label='Steady W2')
    ax1.axhline(y=steady_state[2], color='r', linestyle='--', alpha=0.5, label='Steady W3')
    ax1.set_xlabel('Periode (tahun)', fontsize=11)
    ax1.set_ylabel('Populasi (ribu jiwa)', fontsize=11)
    ax1.set_title('Evolusi Populasi per Wilayah', fontsize=12, fontweight='bold')
    ax1.legend(loc='best', fontsize=9)
    ax1.grid(True, alpha=0.3)

    # Plot 2: Proporsi Populasi
    ax2 = axes[0, 1]
    proportions = pop_sim / np.sum(pop_sim, axis=1, keepdims=True) * 100
    ax2.plot(range(n_simulasi + 1), proportions[:, 0], 'b-', linewidth=2, label='Wilayah 1')
    ax2.plot(range(n_simulasi + 1), proportions[:, 1], 'g-', linewidth=2, label='Wilayah 2')
    ax2.plot(range(n_simulasi + 1), proportions[:, 2], 'r-', linewidth=2, label='Wilayah 3')
    ax2.set_xlabel('Periode (tahun)', fontsize=11)
    ax2.set_ylabel('Proporsi (%)', fontsize=11)
    ax2.set_title('Proporsi Populasi per Wilayah', fontsize=12, fontweight='bold')
    ax2.legend(loc='best', fontsize=9)
    ax2.grid(True, alpha=0.3)
    ax2.set_ylim([0, 100])

    # Plot 3: Konvergensi ke Steady State
    ax3 = axes[1, 0]
    distance_to_steady = np.linalg.norm(pop_sim - steady_state, axis=1)
    ax3.semilogy(range(n_simulasi + 1), distance_to_steady, 'purple', linewidth=2.5)
    ax3.set_xlabel('Periode (tahun)', fontsize=11)
    ax3.set_ylabel('Jarak ke Steady State (skala log)', fontsize=11)
    ax3.set_title('Konvergensi ke Keadaan Tunak', fontsize=12, fontweight='bold')
    ax3.grid(True, alpha=0.3, which='both')
```

```python
# Plot 4: Perbandingan Distribusi
ax4 = axes[1, 1]
x_labels = ['Wilayah 1', 'Wilayah 2', 'Wilayah 3']
x_pos = np.arange(len(x_labels))
width = 0.35

bars1 = ax4.bar(x_pos - width/2, x0, width, label='Populasi Awal (t=0)',
                alpha=0.8, color='steelblue', edgecolor='black')
bars2 = ax4.bar(x_pos + width/2, steady_state, width, label='Steady State (t→∞)',
                alpha=0.8, color='coral', edgecolor='black')

# Tambahkan nilai di atas bar
for bars in [bars1, bars2]:
    for bar in bars:
        height = bar.get_height()
        ax4.text(bar.get_x() + bar.get_width()/2., height,
                f'{height:.0f}',
                ha='center', va='bottom', fontsize=9)

ax4.set_xlabel('Wilayah', fontsize=11)
ax4.set_ylabel('Populasi (ribu jiwa)', fontsize=11)
ax4.set_title('Perbandingan: Awal vs Steady State', fontsize=12, fontweight='bold')
ax4.set_xticks(x_pos)
ax4.set_xticklabels(x_labels)
ax4.legend(loc='best', fontsize=9)
ax4.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.savefig('dinamika_populasi_markov.png', dpi=300, bbox_inches='tight')
print("\nGrafik berhasil disimpan: 'dinamika_populasi_markov.png'")
plt.show()
```

Gambar 4.8 Prosedur visualisasi_dinamika_populasi
Sumber : *Dokumen Penulis*

Prosedur di atas adalah visualisasi dinamika populasi yang disajikan melalui grafik untuk memperlihatkan proses konvergensi distribusi populasi menuju keadaan tunak secara visual.

```python
def main():
    print("\n")
    print("╔" + "="*68 + "╗")
    print("║" + " "*68 + "║")
    print("║" + "  ANALISIS PREDIKSI DINAMIKA POPULASI".center(68) + "║")
    print("║" + "  Model Markov dengan Diagonalisasi Matriks".center(68) + "║")
    print("║" + " "*68 + "║")
    print("╚" + "="*68 + "╝")
    print()

    # ========== BAGIAN A ==========
    P, x0 = definisi_studi_kasus()

    # ========== BAGIAN B ==========
    # B.1: Prediksi Iteratif
    n_periode_pendek = 10
    riwayat = prediksi_iteratif(P, x0, n_periode_pendek)

    # B.2: Analisis Nilai Eigen
    eigenvalues, eigenvectors = analisis_nilai_eigen(P)

    # B.3: Diagonalisasi
    V, D, V_inv = diagonalisasi_matriks(P, eigenvalues, eigenvectors)

    # B.4: Prediksi Jangka Panjang
    periode_jangka_panjang = [20, 50, 100]
    hasil_prediksi = prediksi_jangka_panjang(P, x0, eigenvalues, V, V_inv,
                                             periode_jangka_panjang)

    # ========== BAGIAN C ==========
    steady_state = hitung_keadaan_tunak(P, x0, eigenvalues, eigenvectors)

    # Analisis tambahan
    analisis_kecepatan_konvergensi(eigenvalues, x0, steady_state)

    # Visualisasi
    visualisasi_dinamika_populasi(P, x0, steady_state, n_simulasi=50)
```

Gambar 4.9 Prosedur utama
Sumber : *Dokumen Penulis*

Kode main() tersebut berperan sebagai pengendali utama yang mengatur seluruh alur eksekusi program analisis dinamika populasi menggunakan model Markov berbasis diagonalisasi matriks. Fungsi main() tersebut mengintegrasikan seluruh tahapan analisis secara terstruktur dan sesuai dengan dasar teori serta pembahasan yang telah dibahas dalam makalah. Di dalam fungsi ini didefinisikan vektor distribusi populasi awal dan matriks transisi sebagai dasar perhitungan, kemudian dilakukan prediksi distribusi populasi untuk beberapa periode ke depan melalui perkalian matriks. Selanjutnya, fungsi main() memanggil proses perhitungan nilai eigen dan vektor eigen untuk menentukan keadaan tunak sistem, serta menampilkan hasil analisis baik dalam bentuk numerik maupun visualisasi grafik.

V. KESIMPULAN

Pemodelan dinamika populasi menggunakan rantai Markov berbasis diagonalisasi matriks memberikan kerangka analitis yang kuat untuk memahami perilaku sistem populasi dalam jangka panjang. Melalui representasi perpindahan antar keadaan dalam bentuk matriks transisi, dinamika populasi dapat dianalisis sebagai sistem linear diskrit yang evolusinya ditentukan oleh struktur matriks tersebut. Pendekatan ini memungkinkan analisis kestabilan dan konvergensi sistem dilakukan secara lebih terstruktur melalui nilai eigen dan vektor eigen, sehingga karakteristik keadaan tunak dapat ditentukan secara matematis tanpa bergantung pada simulasi berulang dalam jumlah besar.

Implementasi program yang dilakukan memperjelas keterkaitan antara teori aljabar linear dan penerapannya dalam pemodelan dinamika populasi. Program tidak hanya merealisasikan proses prediksi distribusi populasi secara iteratif, tetapi juga memanfaatkan hasil diagonalisasi matriks untuk menghitung prediksi jangka panjang secara lebih efisien. Perhitungan nilai eigen digunakan untuk mengidentifikasi keadaan tunak sistem serta menganalisis kecepatan konvergensi menuju kondisi tersebut, sementara visualisasi hasil prediksi membantu memperlihatkan proses konvergensi secara konkret. Metode diagonalisasi terbukti meningkatkan efisiensi komputasi dan memberikan wawasan mengenai bagaimana struktur matriks transisi memengaruhi kestabilan dan laju perubahan distribusi populasi. Pendekatan ini berpotensi dikembangkan lebih lanjut untuk sistem dengan jumlah keadaan yang lebih besar, matriks transisi yang bergantung pada waktu, atau integrasi faktor demografis lain seperti kelahiran dan kematian, sehingga model yang dihasilkan menjadi lebih fleksibel, realistis, dan relevan untuk analisis populasi yang lebih kompleks.

VI. LAMPIRAN

Kode program secara lengkap dapat dilihat pada link github berikut: https://github.com/alyanrrhma/Makalah-Algeo.git serta video penjelasan dari makalah ini dapat dilihat pada link youtube berikut: https://bit.ly/81MakalahAlgeo88.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga makalah berjudul *"Analisis Prediksi Dinamika Populasi*

Menggunakan Model Markov Berbasis Diagonalisasi Matriks" ini dapat diselesaikan dengan baik. Penulis juga menyampaikan terima kasih kepada Bapak Ir. Rila Mandala, M.Eng., Ph.D, selaku dosen mata kuliah IF2123 Aljabar Linear dan Geometri, atas bimbingan, materi perkuliahan, serta penjelasan yang menjadi landasan penting dalam penyusunan makalah ini. Penulis juga mengucapkan beribu terima kasih atas cara pengajaran Bapak Rila yang penuh perhatian, ketulusan, dan canda tawa, dan banyak latihan soal sehingga kelas perkuliahan ini memiliki makna tersendiri yang membekas di hati penulis. Penulis juga mengucapkan terima kasih untuk Bapak Dr. Rinaldi Munir, M.T. yang telah memfasilitasi dan membuat website pembelajaran yang sangat baik serta terarsip sempurna, sehingga penulis selalu bisa belajar dari mana saja.

Selain itu, penulis juga ingin menyampaikan rasa terima kasih yang sebesarnya kepada saudara kandung, kakak asuh, saudara perasuhan, dan teman penulis yang selalu membahas dan merencanakan jalan jalan ataupun main sehabis masa UAS ini sehingga mendorong motivasi penulis untuk menyelesaikan makalah lebih cepat dan lebih bersemangat. Penulis juga ingin menyampaikan rasa terima kasih untuk kucing penulis, Miko, yang selalu ada dan tidur nyenyak menemani penulis dalam keadaan apapun itu, baik saat penulis sedang belajar ataupun saat penulis sedang mengerjakan tugas. Rasa terima kasih juga tidak luput ditujukan terutama untuk teman penulis yang selalu setia bercengkerama di rulat membagikan banyak ceritanya dan menjadi alasan penulis memilih topik makalah ini. Terima kasih banyak juga untuk ITB yang menjadi kampus ghibli dengan suasana yang nyaman untuk belajar dan mengerjakan tugas, semoga bisa ditingkatkan lagi fasilitasnya

Meskipun tidak ada kaitannya dengan makalah ini, penulis ingin mengucapkan banyak terima kasih untuk semua orang yang telah penulis temui dan kenali di semester 3 ini. Terima kasih banyak untuk semua teman sekelompok karena telah mengerjakan bagiannya dengan baik dan memberikan pelajaran penting bagi penulis. Terima kasih juga untuk himpunan atas pengalaman yang tidak terlupakannya terutama di masa masa osjur yang rasanya penulis ingin mengulangi kembali pengalaman di masa tersebut. Terima kasih juga untuk kakak tingkat dan teman-teman yang selalu menjawab pertanyaan penulis dan mengajarkan apa yang belum penulis ketahui. Terima kasih banyak UKM yang penulis ikuti karena menghadirkan warna baru serta menjadi keluarga baru penulis. Terima kasih juga untuk kepanitiaan dan organisasi yang penulis ikuti karena telah menambah pengalaman baru dan mengajarkan *skill* penting serta menjadi tempat penulis bertumbuh. Terima kasih untuk pengalaman yang tidak terlupakan di semester 3 ini.

Akhir kata, penulis menyadari bahwa makalah ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, penulis memohon maaf apabila terdapat kesalahan atau ketidaktepatan dalam penyajian maupun pembahasan, serta terbuka terhadap kritik dan saran yang membangun.

REFERENSI

[1] Munir, Rinaldi. *Nilai Eigen dan Vektor Eigen (Bagian 1): Bahan Kuliah IF2123 Aljabar Linear dan Geometri 2025-2026*. STEI ITB.https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf, diakses pada 20 Desember 2025 pukul 10.38.

[2] Munir, Rinaldi. *Nilai Eigen dan Vektor Eigen (Bagian 2): Bahan Kuliah IF2123 Aljabar Linear dan Geometri 2025-2026*. STEI ITB.https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-20-Nilai-Eigen-dan-Vektor-Eigen-Bagian2-2025.pdf, diakses pada 20 Desember 2025 pukul 10.44.

[3] Sani, Muhammad Yamin dan Lebba. *Dinamika Kependudukan dan Pembangunan Sosial*. Editor oleh Lebba. Ciputat: Mazhab Ciputat. https://repository.uinjkt.ac.id/dspace/bitstream/123456789/77929/1/Dinamika%20Kependudukan%20dan%20Pembangunan%20Sosial-2.pdf, diakses pada 24 Desember 2025 pada pukul 06.27.

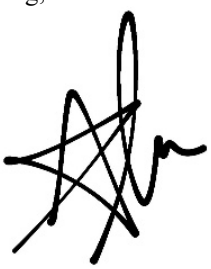
[4] Fewster, Rachel. *Chapter 8: Markov Chains*. Department of Statistics, University of Auckland. https://www.stat.auckland.ac.nz/~fewster/325/notes/ch8.pdf, diakses pada 24 Desember 2025 pada pukul 08.04.

[5] Meyn, S.P. dan Tweedie, R.L. *Markov Chains and Stochastic Stability*. Cambridge University Press (Digital Edition). https://probability.ca/MT/BOOK.pdf, diakses pada 24 Desember 2025 pada pukul 08.15.

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, 24 Desember 2025



Alya Nur Rahmah 13524081