Reflective Report

Design and Creative Technologies

Torrens University, Australia

Student: Luis Guilherme de Barros Andrade Faria - A00187785

Subject Code: MFA 502

Subject Name: Mathematical Foundations of Artificial Intelligence

Assessment No.: 2A

Title of Assessment: Reflective Report, Set 1, Problem 2

Lecturer: Dr. James Vakilian

Date: Oct 2025

Table of Contents

1. Introduction and Overview

The second component of EigenAI focused on computing **eigenvalues** and **eigenvectors** for a 2×2 matrix **without using external libraries**. After finishing the determinant recursion logic, I wanted to extend the same clarity and interactivity to another essential linear-algebra concept.

I structured this challenge to produce correct eigenpairs and **teach the underlying steps** interactively through *EigenAI's* Streamlit interface.

The **EigenAI architecture** remained consistent with Set 1 Problem 1:

- **Frontend / Presentation layer:** Streamlit UI for matrix input, progress animations, and result display.
- **Logic layer:** Pure-Python solver (`eigen_solver.py`) implementing characteristic-polynomial computation and vector derivation.
- **Integration:** `set1Problem2.py` connects both layers, displaying tutor-style explanations during execution.

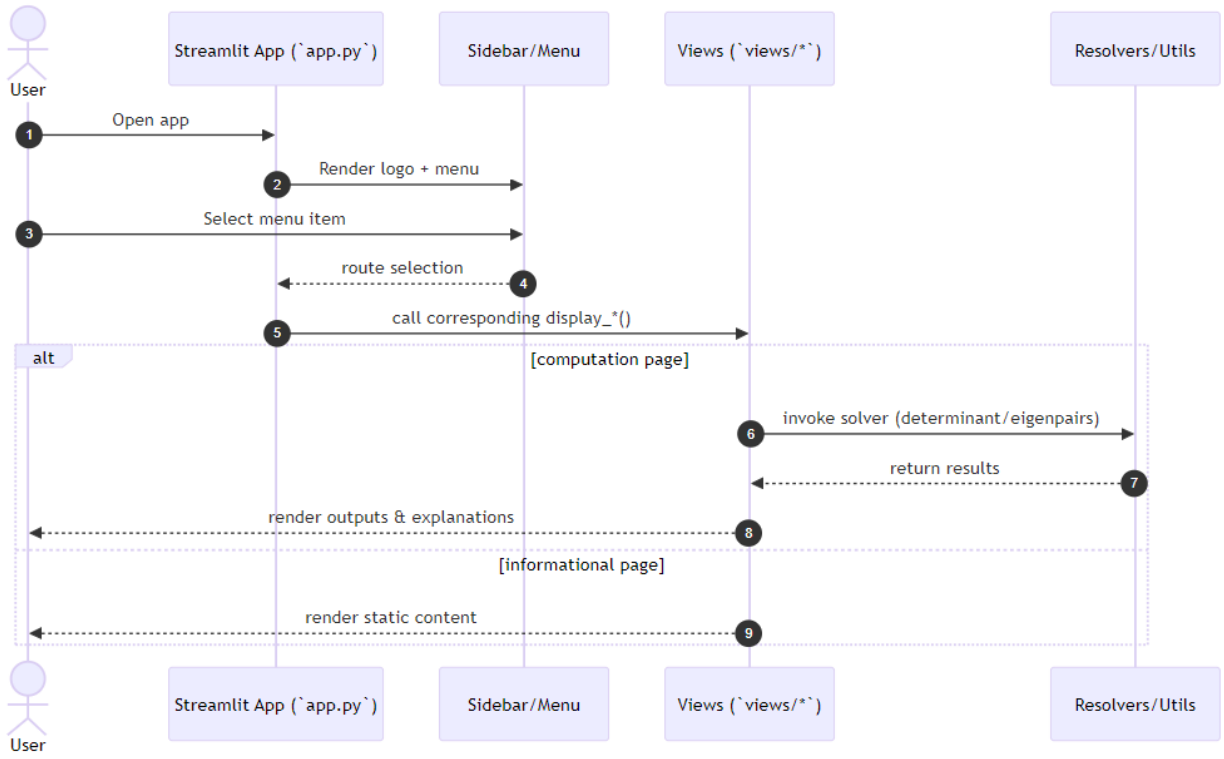

Figure 1: Sequence diagram showing interaction between* `app.py`, `views/set1Problem2.py`, *and* `utils/eigen_solver.py`.

2. Mathematical Approach

The algorithm is grounded in the characteristic equation of a 2×2 matrix A = [[a,b],[c,d]]:

$$\det(A - \lambda I) = \lambda^2 - (a + d)\lambda + (ad - bc) = 0$$

Solving this quadratic yields the **eigenvalues (λ_1, λ_2)**.

Each eigenvector v satisfies $(A - \lambda I)v = 0$, which provides the direction that remains unchanged by the transformation.

The program solves these steps explicitly:

1. Compute **trace ($t = a + d$)** and **determinant ($\Delta = ad - bc$)**.
2. Derive **discriminant ($t^2 - 4\Delta$)**.
3. Apply a manual **Newton–Raphson square-root** (`my_sqrt`) to avoid math-library dependencies.
4. Substitute results in the quadratic formula to find both λ values.
5. For each λ, compute the corresponding eigenvector ratio (e.g., $(y = -((a - \lambda)/b)x)$).

This procedure demonstrates mastery of both the **algebraic definition** and the **computational flow** underlying eigen decomposition.

3. Programming Methods

`eigen_solver.py` was developed using the same step-by-step documentation style as `determinant.py`, ensuring readability and instructional value.

Key functions include:

- `my_sqrt(x)` – Newton–Raphson approximation.
- `eigenvalues_2x2(A)` – computes λ_1, λ_2.
- `eigenvector_for_lambda(A, λ)` – solves $(A - \lambda I)v = 0$ and normalizes v.
- `eigenpairs(A)` – combines results into (λ, v) tuples.

`set1Problem2.py` integrates this logic into EigenAI's UI: Users input a 2×2 matrix, click **"Compute Eigenpairs"**, and the tutor walks them through three animated steps:

1. Forming $(A - \lambda I)$

2. Solving $\det(A - \lambda I) = 0$
3. Computing eigenvectors

This aligns with **pedagogical UX principles**, making abstract math transparent and engaging.

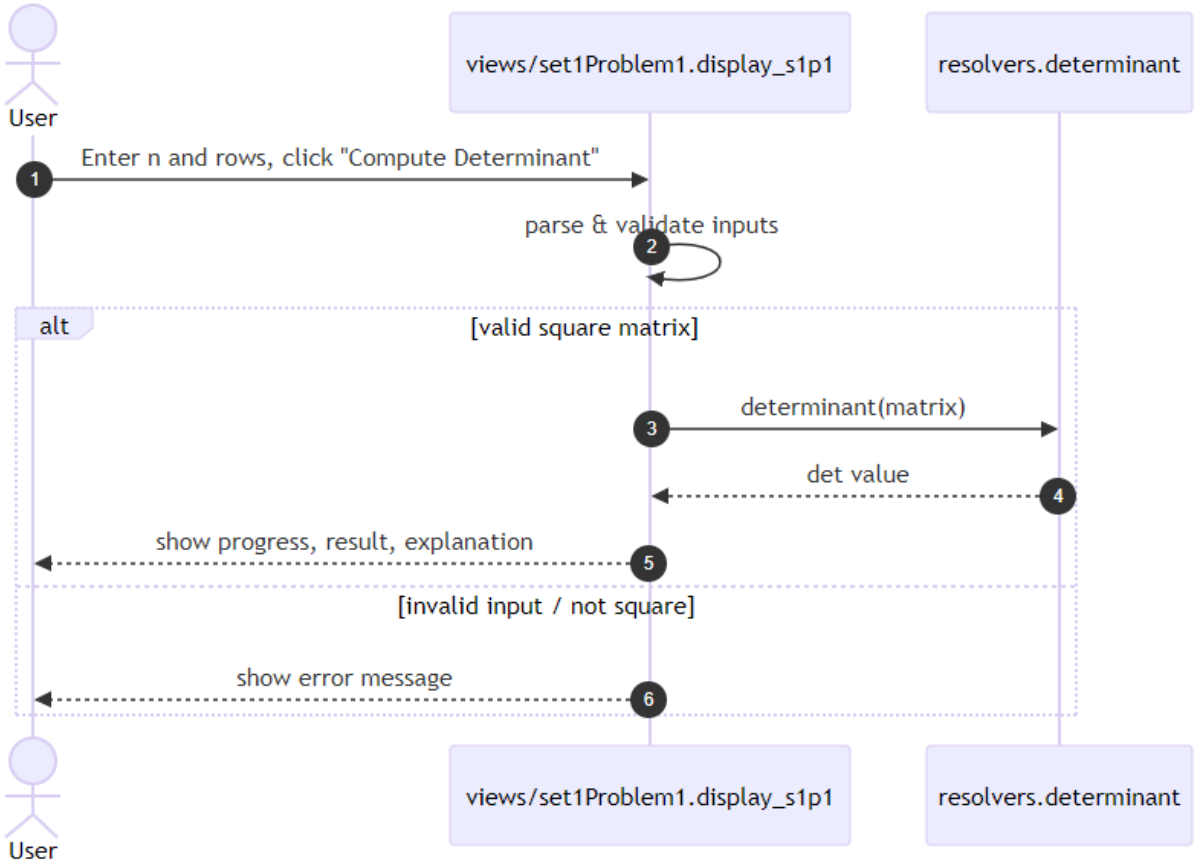

Figure 2: Sequence diagram illustrating user input, progress bar animation, and eigenpair resolution flow.

4. What Went Right

- **Consistent modularity** between determinant and eigen modules allowed quick integration.

- **Manual sqrt implementation** improved understanding of numerical methods.
- **Tutor-style UI** successfully presented intermediate symbolic steps, supporting learners visually.
- Maintained **separation of concerns**, simplifying debugging and testing.

5. What Went Wrong

- Early confusion over handling cases where b = 0 or c = 0 (required conditional vector logic).
- Minor rounding inconsistencies when normalizing eigenvectors.
- Attempting to generalize to 3×3 matrices exposed the quadratic-formula limitation.

6. Uncertainties

Performance was never the issue here; rather, the uncertainty lies in **extending symbolic solvers** to higher-dimensional cases and future exploration could involve iterative eigenvalue methods such as **Power Iteration** or integrating **NumPy** if external-library use becomes permissible.

7. Personal Insight

Before this project, eigenvalues and eigenvectors felt abstract, I understood their definition but not their behavior. Implementing the solver revealed their **geometric meaning**: certain directions in space remain fixed, merely scaled by λ.

Developing `my_sqrt` and manually solving quadratic equations gave me a deeper appreciation for how **core AI libraries perform matrix decompositions under the hood** and I

can also point out that the experience mirrored the principle behind machine learning itself — breaking complexity into small, learnable steps.

Seeing the Streamlit progress bar narrate each phase felt like bridging the gap between **mathematics, computation, and pedagogy**.

8. Conclusion

Set 1 Problem 2 strengthened my mathematical intuition and numerical reasoning and helped me learn to treat equations as algorithms to implement, not as symbols to memorize.

Reading about it helped me grasp fundamental concepts to AI engineering that I've used in the past, but that with a lot of abstractions, where linear algebra underpins models like **PCA**, **SVD**, and **neural-network weight transformations**.

The EigenAI project thus serves both as a personal milestone and as a reusable educational framework for future MFA501 students.

Statement of Acknowledgment

I acknowledge that I have used the following AI tool(s) in the creation of this report:

- OpenAI ChatGPT (GPT-5): Used to assist with outlining, refining structure, improving clarity of academic language, and supporting APA 7th referencing conventions.

I confirm that the use of the AI tool has been in accordance with the Torrens University Australia Academic Integrity Policy and TUA, Think and MDS's Position Paper on the Use of AI. I confirm that the final output is authored by me and represents my own critical thinking, analysis, and synthesis of sources. I take full responsibility for the final content of this report.

9. References

Dash, R. B., & Dalai, D. K. (2008). *Fundamentals of Linear Algebra*. ProQuest Ebook Central.

Torrens University Australia (2025). *MFA501 Module Notes – Linear Transformations and Matrix Operations.*

EigenAI Project (2025). *GitHub Repository.* Retrieved from https://github.com/lfariabr/masters-swe-ai/tree/master/2025-T2/T2-MFA/projects/eigenai

Streamlit, Inc. (n.d.). *Streamlit documentation.* Retrieved from https://docs.streamlit.io/