# MATH 6310 (FALL 2025) HOMEWORK SOLUTIONS

VANDIT GOEL
**UTA ID:**1002245699

## 1. HOMEWORK 2 - WEDNESDAY 1 OCTOBER, 2025

**Problem** (3)**.** Prove that for any $A \in \mathbb{R}^{n\times n}$, $\text{tr}(A) = \sum \lambda_i(A)$.

**Solution.** By Spectral Theorem,

$$A = V\Lambda V^{-1} = V\Lambda V^{\top},$$

where $V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}$ and $\Lambda$ is diagonal and contains the eigenvectors of $A$.

$$\text{tr}(A) = \text{tr}(V\Lambda V^{\top}) \tag{1}$$

By cyclic property of trace, $\text{tr}(ABC) = \text{tr}(CAB)$

We can rewrite (1) as,

$$\text{tr}(A) = \text{tr}(V^{\top} V\Lambda)$$

Since V is an orthogonal matrix, $VV^{\top} = I$

$$\text{tr}(A) = \text{tr}(\Lambda) = \sum \lambda_i$$

where $\lambda_i$ are the eigenvalues of $A$.

**Problem** (4)**.** A symmetric matrix $A = A^{\top}$ has orthonormal eigenvectors $v_1, \dots, v_n$. Define the **Rayleigh quotient** of $A$ via:

$$R(x) := \frac{\langle Ax, x\rangle}{\langle x, x\rangle}.$$

(a) Prove that

$$\max_{x\in\mathbb{R}^n} R(x) = \lambda_1.$$

(b) Now prove that $\lambda_2$ is the solution to the following constrained optimization problem

$$\max_{x\in\mathbb{R}^n} R(x) \quad \text{subject to} \quad \langle v_1, x\rangle = 0.$$

(c) Similar to the previous part, under what constraints is $\lambda_3$ the solution to $\max R(x)$ ?

**Solution.** (a)

**Claim.**

$$\max_{x\in\mathbb{R}^n} R(x) = \lambda_1.$$

*Proof.* Given $A = A^{\top}$ and $v_1, \dots, v_n$ are orthonormal eigenbasis with eigenvalues ordered $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. For any $x \in \mathbb{R}^n$ let

$$x = \sum_{i=1}^{n} \alpha_i v_i, \qquad \alpha_i = \langle v_i, x\rangle$$

Then, $\langle x, x\rangle = \sum_i \alpha_i^2$ and since $Av_i = \lambda_i v_i$,

$$\langle Ax, x\rangle = \sum_{i=1}^{n} \lambda_i \alpha_i^2$$

Hence the Rayleigh quotient becomes

$$R(x) = \frac{\sum_{i=1}^{n} \lambda_i \alpha_i^2}{\sum_{i=1}^{n} \alpha_i^2} = \sum_{i=1}^{n} \lambda_i w_i, \quad \text{where } w_i := \frac{\alpha_i^2}{\sum_j \alpha_j^2} \tag{1}$$

Note that $w_i \geq 0$ and $\sum_i w_i = 1$, so $R(x)$ is weighted average of the eigenvalues $\lambda_i$. From this representation, the following facts are immediate.

$$\min_i \lambda_i \leq R(x) \leq \max_i \lambda_i = \lambda_1 \qquad \forall x$$

□

(b) If we impose $\langle v_1, x\rangle = 0$, then $\alpha_1 = 0$ and the weights satisfy $w_1 = 0$. From (1), $R(x)$ is a combination of $\lambda_2, \ldots, \lambda_n$, so for such $x$

$$R(x) \leq \max_{i\geq 2} \lambda_i = \lambda_2$$

(c) By the same reasoning, if $x$ is orthogonal to $v_1$ and $v_2$ (i.e. $\langle v_1, x\rangle = \langle v_2, x\rangle = 0$), then $\alpha_1 = \alpha_2 = 0$ and $R(x)$ is a combination of $\lambda_3, \ldots, \lambda_n$. Thus, the maximum under those constraints is $\lambda_3$, achieved at $x = v_3$.

**Problem** (5)**.** In class we proved (or will prove) the first equality in the following theorem.

**Theorem 1.1** (Courant–Fischer Minimax Theorem)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Then*

$$\lambda_k = \max_{\substack{S\subset\mathbb{R}^n \\ \dim(S)=k}} \min_{x\in S} \frac{\langle Ax, x\rangle}{\langle x, x\rangle} = \min_{\substack{T\subset\mathbb{R}^n \\ \dim(T)=n-k+1}} \max_{x\in T} \frac{\langle Ax, x\rangle}{\langle x, x\rangle}.$$

Prove the second equality (by proving that the far right-hand side gives $\lambda_k$). This is a generalization of the Rayleigh quotient problem above.

**Solution.**

**Claim.**

$$\lambda_k = \min_{\substack{T\subset\mathbb{R}^n \\ dim T=n-k+1}} \max_{x\in T} R(x), \qquad R(x) = \frac{\langle Ax, x\rangle}{\langle x, x\rangle},$$

*Proof.* Let $S_k := \operatorname{span}(v_1, \ldots, v_k)$ (so $\dim S_k = k$) and $T_k := \operatorname{span}(v_k, \ldots, v_n)$ (so $\dim T_k = n - k + 1$)

For any nonzero $x \in T_k$ it can be written $x = \sum_{i=k}^{n} \alpha_i v_i$, hence □

$$R(x) = \frac{\sum_{i=k}^{n} \lambda_i \alpha_i^2}{\sum_{i=k}^{n} \alpha_i^2}$$

From the proof in problem (4) we know $R(x) \leq \lambda_k$ for all $x \in T_k$ and the maximum over $T_k = \lambda_k$ (attained at $x = v_k$). Thus,

$$\min_{\dim T=n-k+1} \max_{x\in T} R(x) \leq \max_{x\in T_k} R(x) = \lambda_k \tag{1}$$

Now, we know

$$\dim S_k + \dim T = k + (n - k + 1) = n + 1 > n$$

that the intersection $S_k \cap T$ is non-trivial, so there exists a non-zero $x \in S_k \cap T$. For such $x$ we can write $x = \sum_{i=1}^{k} \alpha_i v_i$, hence

$$R(x) = \frac{\sum_{i=1}^{k} \lambda_i \alpha_i^2}{\sum_{i=1}^{k} \alpha_i^2}$$

From the proof in Problem (4) we know $R(x) \geq \min\{\lambda_1, \ldots, \lambda_k\} = \lambda_k$

$$\max_{y \in T} R(y) \geq R(x); \geq; \lambda_k$$

This is true since $x \in T$ as well so $R(x)$ is one of the candidate values in that set. The maximum over the whole set must be at least as large as any one candidate. This holds for every subspace $T$ of dimension $n-k+1$, so taking the minimum over such $T$ yields

$$\min_{\dim T = n-k+1} \max_{x \in T} R(x) \geq \lambda_k \tag{2}$$

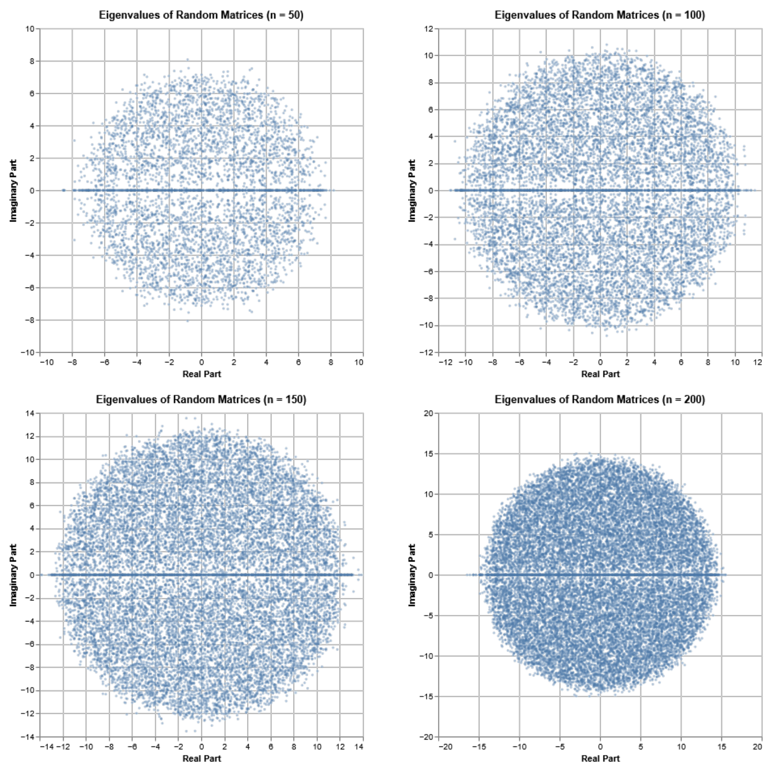
From (1) and (2)

$$\lambda_k = \min_{\substack{T \subset \mathbb{R}^n \\ dim T = n-k+1}} \max_{x \in T} R(x)$$

**Problem** (31)**.** [Requires Programming] Generate 100 matrices of size $100 \times 100$ whose entries are random Gaussians (i.e., drawn from $\mathcal{N}(0,1)$). For each matrix, compute its eigenvalues, and after all matrices are generated, plot all of the eigenvalues on one plot (note they will be complex in general). What do you notice? Try this experiment for different sizes of matrices; what do you notice?

Your code must be turned in as an appendix at the end of your homework. In the space below this problem, you should put any figures you generate and your answers to the questions.

**Solution.** Generated plots for different sizes of $n \times n$ matrices



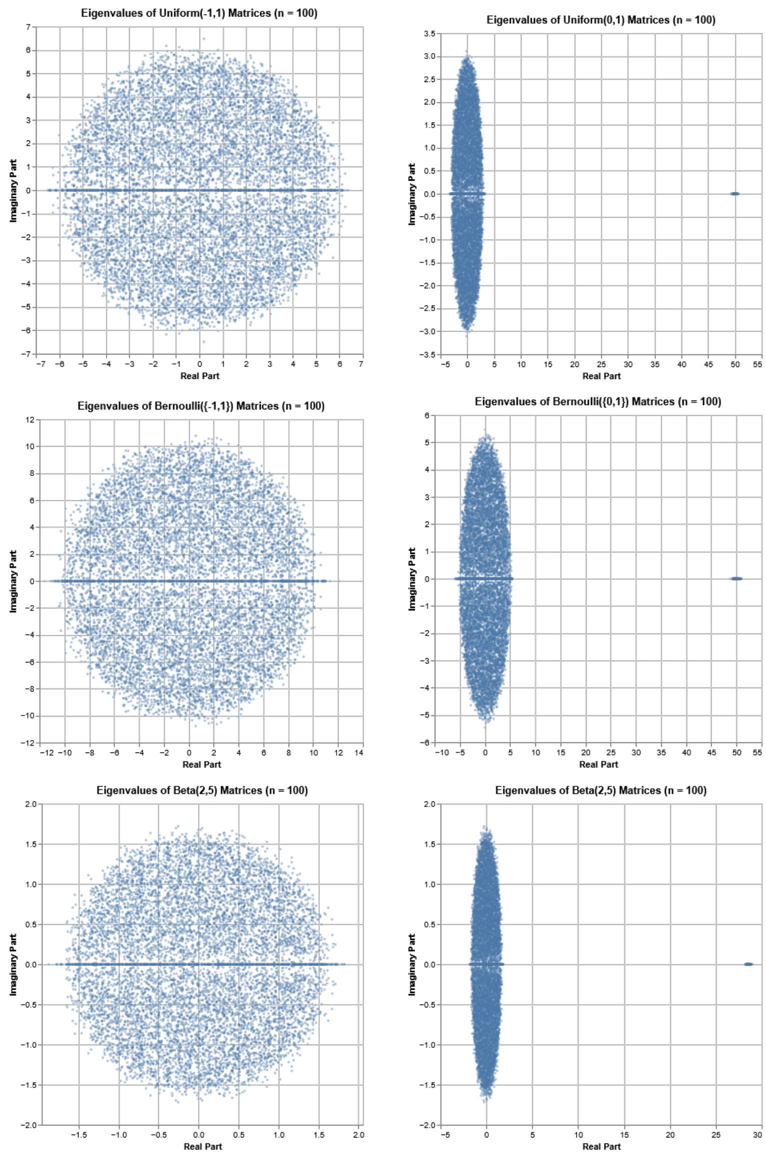

**Observations:**

- Eigenvalues are plotted in the complex plane (real part on x-axis, imaginary part on y-axis).
- The plotted points form a roughly circular band centered near the origin.
- The radius increases with the size of the matrices.
  - For $n = 50$, radius $\approx 7$
  - For $n = 100$, radius $\approx 10$
  - For $n = 150$, radius $\approx 13$
  - For $n = 200$, radius $\approx 14$
- Point density appears higher near the center and decreases toward the perimeter.
- Some variability and gaps are visible; the distribution is not perfectly uniform.
- The plot is less uniform for smaller values of $n$.

**Problem** (32)**.** [Requires Programming] Repeat the process of Problem 1 but for random matrices with entries drawn i.i.d. from some other distribution and write your conclusions. (Examples would be Uniform[0,1], Uniform[a,b] for other choices of $a, b \in \mathbb{R}$, Bernoulli $\{\pm 1\}$ (with equal probability), $\beta$ distributions, but feel free to choose whatever you like).

Your code must be turned in as an appendix at the end of your homework. In the space below this problem, you should put any figures you generate and your answers to the questions.

**Solution.** Generated plots for various other distributions:




**Observations:**

- The chart for Uniform (0,1) distributions seem to be circular/disk as well. But interestingly has a few outliers around (50,0).
- The outliers disappear for Uniform (-1, 1) and is predominantly circular/disk.
- The raidus also seem to increase with the size of the matrix for both Uniform distributions.
- Similar observations noted for Bernoulli distribution.
- Beta Distribution seem to always have outliers. But is predominantly circular/disk shaped.
- The disk shape for these distribution can be attributed to the fact that they can be approximated as a Normal Distribution.

## Appendix A. Code for Problems 31 and 32

Listing 1. Marimo application used to generate eigenvalue visualizations

```
import marimo

__generated_with = "0.16.4"
app = marimo.App()


@app.cell(hide_code=True)
def _(mo):
    mo.md(
        r"""
␣␣␣␣Homework␣2
␣␣␣␣==========
␣␣␣␣"""
    )
    return


@app.cell(hide_code=True)
def _(mo):
    mo.md(
        rf"""
    **Name:** Vandit Goel
    **ID:** 1002245699
    **Email:** vxg5699@mavs.uta.edu
    **Repo:** git@github.com:vandyG/math-6310.git
    """
    )
    return


@app.cell(hide_code=True)
def _(mo):
    mo.md(
        r"""
␣␣␣␣##␣Problem␣31

␣␣␣␣Generate␣100␣matrices␣of␣size␣$100\times␣100$␣whose␣entries␣are␣random␣Gaussians␣(i.e.,␣drawn␣from␣$\mathcal{N}(0,1)$).␣␣For␣each␣matrix,␣compute␣its␣eigenvalues,␣and␣after␣all␣matrices␣are␣generated,␣plot␣all␣of␣the␣eigenvalues␣on␣one␣plot␣(note␣they␣will␣be␣complex␣in␣general).␣␣What␣do␣you␣notice?␣␣Try␣this␣experiment␣for␣different␣sizes␣of␣matrices;␣what␣do␣you␣notice?

␣␣␣␣Your␣code␣must␣be␣turned␣in␣as␣an␣appendix␣at␣the␣end␣of␣your␣homework.␣In␣the␣space␣below␣this␣problem,␣you␣should␣put␣any␣figures␣you␣generate␣and␣your␣answers␣to␣the␣questions.
```

```python
    """
    )
    return


@app.cell
def _():
    import numpy as np
    import matplotlib.pyplot as plt
    import marimo as mo
    import pandas as pd
    import altair as alt
    return alt, mo, np, pd


@app.cell
def _(mo):
    matrix_size_slider = mo.ui.slider(
        start=20,
        stop=200,
        value=100,
        step=10,
        label="Matrix dimension (n × n)",
        show_value=True,
    )

    distribution_dropdown = mo.ui.dropdown(
        options=[
            "Normal(0,1)",
            "Uniform(0,1)",
            "Uniform(-1,1)",
            "Bernoulli({0,1})",
            "Beta(2,5)",
        ],
        value="Normal(0,1)",
        label="Select Distribution",
    )
    return distribution_dropdown, matrix_size_slider


@app.cell
def _(distribution_dropdown, matrix_size_slider):
    dist = distribution_dropdown.value
    size = matrix_size_slider.value
    return dist, size


@app.cell
def _(dist, np, size):
    if dist == "Normal(0,1)":
        matrices = [np.random.normal(0, 1, (size, size)) for _ in range(100)]
    elif dist == "Uniform(0,1)":
```

```python
        matrices = [np.random.uniform(0, 1, (size, size)) for _ in range(100)]
    elif dist == "Uniform(-1,1)":
        matrices = [np.random.uniform(-1, 1, (size, size)) for _ in range(100)]
    elif dist == "Bernoulli({0,1})":
        matrices = [np.random.choice([0, 1], (size, size)) for _ in range(100)]
    elif dist == "Beta(2,5)":
        matrices = [np.random.beta(5, 2, (size, size)) for _ in range(100)]
    return (matrices,)


@app.cell
def _(matrices, np):
    eigenvalues = [np.linalg.eigvals(matrix) for matrix in matrices]
    all_eigenvalues = np.concatenate(eigenvalues)
    return (all_eigenvalues,)


@app.cell
def _(
    all_eigenvalues,
    alt,
    dist,
    distribution_dropdown,
    matrix_size_slider,
    mo,
    pd,
):
    alt.data_transformers.enable("vegafusion")

    n = matrix_size_slider.value

    # Prepare a DataFrame with real and imaginary parts
    df = pd.DataFrame({
        "real": all_eigenvalues.real,
        "imag": all_eigenvalues.imag,
    })

    # Create an Altair scatter plot with semi-transparent points
    chart = (
        alt.Chart(df)
        .mark_point(filled=True, opacity=0.45, size=10)
        .encode(
            x=alt.X("real:Q", title="Real Part"),
            y=alt.Y("imag:Q", title="Imaginary Part"),
            tooltip=[
                alt.Tooltip("real:Q", format=".3f"),
                alt.Tooltip("imag:Q", format=".3f"),
```

```
            ],
        )
        .properties(
            title=f"Eigenvalues of {dist} Matrices (n = {n})",
            height=400,
            width=400,
        )
        .resolve_scale(x="independent", y="independent")
        .interactive(bind_y=False)
    )

    controls = mo.vstack(
        [
            distribution_dropdown,
            matrix_size_slider,

        ],
        align="start",
        gap="0.75rem",
    )

    layout = mo.vstack([controls, chart], align="center", gap="2rem")
    layout
    return


@app.cell(hide_code=True)
def _(mo):
    mo.md(
        r"""
    ### Observations
    #### Problem 31

    - Eigenvalues are plotted in the complex plane (real part on x-axis, imaginary part on y-axis).
    - The plotted points form a roughly circular band centered near the origin.
    - The radius increases with the size of the matrices.
        - For n=50, radius ~ 7
        - For n=100, radius ~ 10
        - For n=150, radius ~ 13
        - For n=200, radius ~ 14
    - Point density appears higher near the center and decreases toward the perimeter.
    - Some variability and gaps are visible; the distribution is not perfectly uniform.
    - The plot is less uniform for smaller values of n.

    #### Problem 32
    - The chart for Uniform (0,1) distributions seem to be circular/disk as well. But interestingly has a few outliers around (50,0).
```

```
    - The outliers disappear for Uniform (-1, 1) and is predominantly circular/disk.
    - The raidus also seem to increase with the size of the matrix for both Uniform distributions.
    - Similar observations noted for Bernoulli distribution.
    - Beta Distribution seem to always have outliers. But is predominantly circular/disk shaped.
    - The disk shape for these distribution can be attributed to the fact that they can be approximated as a Normal Distribution.
    """
    )
    return


if __name__ == "__main__":
    app.run()
```