12.442

BEERAM MADHURI - EE25BTECH11012

October 2025

Question

Eigen values of the matrix $\begin{pmatrix} 5 & 3 \\ 1 & 4 \end{pmatrix}$ are

a) -6.3 and -2.7 b) -2.3 and -6.7 c) 6.3 and 2.7 d) 2.3 and 6.7

finding the eigen values of given matrix:

Let

$$A = \begin{bmatrix} 5 & 3 \\ 1 & 4 \end{bmatrix} \tag{1}$$

$$A = \lambda I \tag{2}$$

where 'λ' are eigen values.

$$\left| A - \lambda I \right| = 0 \tag{3}$$

$$\left| \begin{bmatrix} 5 & 3 \\ 1 & 4 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0 \tag{4}$$

$$\left| \begin{bmatrix} 5-\lambda & 3 \\ 1 & 4-\lambda \end{bmatrix} \right| = 0 \tag{5}$$

$$(5-\lambda)(4-\lambda)-3=0 \tag{6}$$

$$20-9\lambda+\lambda^2-3=0 \tag{7}$$

$$\lambda^2-9\lambda+17=0 \tag{8}$$

$$\lambda_1=6.3 \tag{9}$$

$$\lambda_2=2.7 \tag{10}$$

Hence eigen values of given matrix are 2.7 and 6.3.
$\therefore$ Option C is correct.

Python Code

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

# Define the matrix
A = np.array([[5, 3],
              [1, 4]])

# Calculate the eigenvalues using numpy for verification
eigenvalues = np.linalg.eigvals(A)
# Sort for consistent plotting
eigenvalues = np.sort(eigenvalues)
```

Python Code

```
print(f"Calculated Eigenvalues: {eigenvalues}")

# Define the characteristic polynomial: lambda^2 - 9*lambda + 17
def characteristic_polynomial(lmbda):
    return lmbda**2 - 9*lmbda + 17

# Generate lambda values for plotting the curve
lmbda_values = np.linspace(1, 8, 400)
poly_values = characteristic_polynomial(lmbda_values)
```

Python Code

```
# Create the plot
plt.style.use('seaborn-v0_8-whitegrid')
fig, ax = plt.subplots(figsize=(10, 6))
# Plot the characteristic polynomial
ax.plot(lmbda_values, poly_values, label=r'$f(\lambda) = \lambda
    ^2 - 9\lambda + 17$', color='royalblue', linewidth=2)
# Plot the x-axis (y=0)
ax.axhline(0, color='black', linewidth=0.75)
# Plot the roots (eigenvalues)
ax.plot(eigenvalues, characteristic_polynomial(eigenvalues), 'o',
     color='crimson', markersize=8, label=f'Eigenvalues (Roots)')
```

Python Code

```
# Annotate the eigenvalues
for eig in eigenvalues:
    ax.annotate(f'$\\lambda \\approx {eig:.2f}$',
                xy=(eig, 0),
                xytext=(eig, -3), # Position the text slightly
                    below the point
                textcoords='data',
                arrowprops=dict(arrowstyle="->", connectionstyle="
                    arc3,rad=.2", color='black'),
                fontsize=12,
                ha='center')
```

Python Code

```
# Set titles and labels for clarity
ax.set_title('Graph of the Characteristic Polynomial', fontsize
    =16)
ax.set_xlabel(r'$\lambda$ (Lambda)', fontsize=12)
ax.set_ylabel(r'$f(\lambda)$', fontsize=12)
ax.legend(fontsize=11)
ax.grid(True)
# Set plot limits
ax.set_ylim(-4, 10)
# Display the plot
plt.show()
```

C Code

```c
#include <stdio.h>
#include <math.h>

int main() {
    // Given 2x2 matrix
    float a = 5, b = 3, c = 1, d = 4;
    // Variables for trace, determinant, and eigenvalues
    float trace, det, lambda1, lambda2, discriminant;
    // Trace = a + d
    trace = a + d;
```

C Code

```
// Determinant = ad - bc
det = a * d - b * c;

// Characteristic equation:  - (trace) + det = 0
// =>  = [trace  sqrt(trace - 4det)] / 2
discriminant = trace * trace - 4 * det;
```

C Code

```
    if (discriminant >= 0) {
        lambda1 = (trace + sqrt(discriminant)) / 2;
        lambda2 = (trace - sqrt(discriminant)) / 2;

        printf("Eigenvalues are: %.2f and %.2f\n", lambda1,
            lambda2);
    } else {
        printf("Eigenvalues are complex.\n");}
    return 0;
}
```

Python and C Code

```
import ctypes
import math

def main():
    # Use ctypes to declare C-style float variables
    a = ctypes.c_float(5)
    b = ctypes.c_float(3)
    c = ctypes.c_float(1)
    d = ctypes.c_float(4)
```

Python and C Code

```
    trace = ctypes.c_float(a.value + d.value)
    det = ctypes.c_float(a.value * d.value - b.value * c.value)
    discriminant = ctypes.c_float(trace.value * trace.value - 4 *
        det.value)

    if discriminant.value >= 0:
        sqrt_disc = math.sqrt(discriminant.value)
        lambda1 = ctypes.c_float((trace.value + sqrt_disc) / 2)
```

Python and C Code

```
        lambda2 = ctypes.c_float((trace.value - sqrt_disc) / 2)
        print(f"Eigenvalues are: {lambda1.value:.2f} and {lambda2
            .value:.2f}")
    else:
        print("Eigenvalues are complex.")
if __name__ == "__main__":
    main()
```

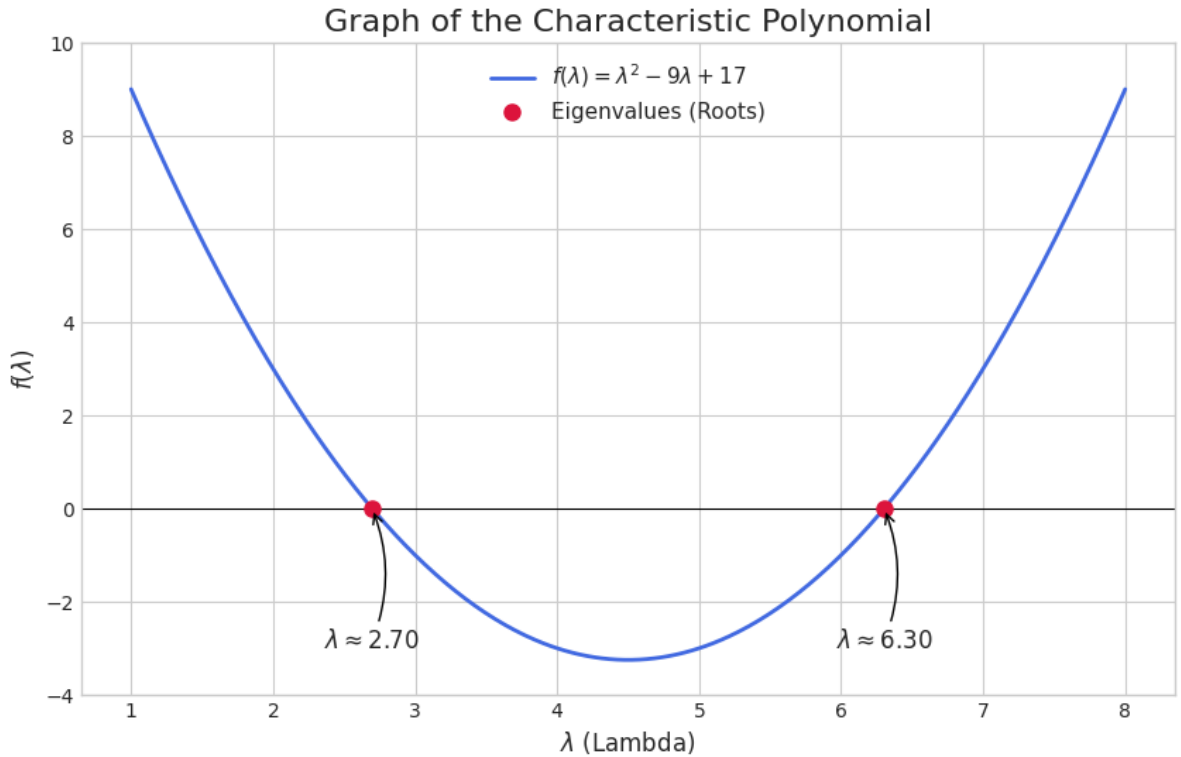
Graph of the Characteristic Polynomial
$f(\lambda) = \lambda^2 - 9\lambda + 17$
Eigenvalues (Roots)
$\lambda \approx 2.70$
$\lambda \approx 6.30$
$f(\lambda)$
λ (Lambda)
10
8
6
4
2
0
−2
−4
1
2
3
4
5
6
7
8

Figure: Plot