

12.859

Bhargav - EE25BTECH11013

October 10, 2025

Question:

Let $\mathbf{O} = \{\mathbf{P} : \mathbf{P} \text{ is a } 3 \times 3 \text{ real matrix with } \mathbf{P}^T \mathbf{P} = \mathbf{I}_3, \det(\mathbf{P}) = 1\}$. Which of the following options is/are correct?

- a) There exists $\mathbf{P} \in \mathbf{O}$ with $\lambda = \frac{1}{2}$ as an eigenvalue.
- b) There exists $\mathbf{P} \in \mathbf{O}$ with $\lambda = 2$ as an eigenvalue.
- c) If λ is the only real eigenvalue of $\mathbf{P} \in \mathbf{O}$, then $\lambda = 1$.
- d) There exists $\mathbf{P} \in \mathbf{O}$ with $\lambda = -1$ as an eigenvalue.

Solution

Let \mathbf{v} be the eigenvector corresponding to the eigenvalue λ .

$$\mathbf{P}\mathbf{v} = \lambda\mathbf{v} \quad (1)$$

Since orthogonal transformations preserve the length of vectors ($|\mathbf{P}| = 1$)

$$\|\mathbf{P}\mathbf{v}\| = \|\mathbf{v}\| \quad (2)$$

$$\|\mathbf{P}\mathbf{v}\| = |\lambda| \|\mathbf{v}\| \quad (3)$$

Solution

Using the above equations,

$$\|\mathbf{v}\| = |\lambda| \|\mathbf{v}\| \quad (4)$$

Thus, $|\lambda| = 1$

Eigenvalues can be either -1 or 1 or both.

Thus, options (c) and (d) are correct.

Solution

This can be verified by examples.

1. For $\lambda_1 = 1$

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{P}^T \mathbf{P} = \mathbf{I}$$

Eigenvalue of \mathbf{P} is 1.

2. For $\lambda_2 = -1$

$$\mathbf{P} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\mathbf{P}^T \mathbf{P} = \mathbf{I}$$

Eigenvalue of \mathbf{P} is -1.

```
#include <stdio.h>

void matmul_transpose(double A[3][3], double result[3][3]) {
    // Compute result = A^T * A
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = 0.0;
            for (int k = 0; k < 3; k++) {
                result[i][j] += A[k][i] * A[k][j];
            }
        }
    }
}
```

Python + C Code

```
import numpy as np
import ctypes

lib = ctypes.CDLL("./libcode.so")
# Define function argument and return types
lib.matmul_transpose.argtypes = [((ctypes.c_double * 3) * 3), ((
    ctypes.c_double * 3) * 3)]

# Convert numpy array to C 2D array
def to_c_matrix(A):
    c_mat = ((ctypes.c_double * 3) * 3)()
    for i in range(3):
        for j in range(3):
            c_mat[i][j] = A[i][j]
    return c_mat

# Convert C 2D array back to numpy
def from_c_matrix(c_mat):
```

```
# Function to verify example
def verify_matrix(P, name):
    print(f"\n--- {name} ---")
    print("Matrix P:\n", P)

    P_c = to_c_matrix(P)
    result_c = ((ctypes.c_double * 3) * 3)()
    lib.matmul_transpose(P_c, result_c)

    PT_P = from_c_matrix(result_c)
    print("\nP^T * P =\n", PT_P)
    print("\nIs P orthogonal?", np.allclose(PT_P, np.eye(3)))

    eigenvalues, _ = np.linalg.eig(P)
    print("\nEigenvalues of P:", eigenvalues)
```



```
# Example 1:  $\lambda = 1$ 
P1 = np.array([[1, 0, 0],
               [0, 1, 0],
               [0, 0, 1]], dtype=float)

# Example 2:  $\lambda = -1$ 
P2 = np.array([[-1, 0, 0],
               [0, 1, 0],
               [0, 0, -1]], dtype=float)

verify_matrix(P1, "Example 1 ( $\lambda = 1$ )")
verify_matrix(P2, "Example 2 ( $\lambda = -1$ )")
```

Python Code

```
import numpy as np

# Example 1:  $\lambda_1 = 1$ 
P1 = np.array([
    [1, 0, 0],
    [0, 1, 0],
    [0, 0, 1]
])

# Example 2:  $\lambda_2 = -1$ 
P2 = np.array([
    [-1, 0, 0],
    [0, 1, 0],
    [0, 0, -1]
])
```

Python Code

```
# Function to verify orthogonality and eigenvalues
def verify_matrix(P, name):
    print(f"--- {name} ---")
    print("Matrix P:\n", P)

    # Check orthogonality
    PT_P = np.dot(P.T, P)
    print("\nP^T * P =\n", PT_P)

    # Check if PT_P is identity
    print("\nIs P^T * P = I ?", np.allclose(PT_P, np.eye(P.shape
        [0])))

    # Eigenvalues of P
    eigenvalues, _ = np.linalg.eig(P)
    print("\nEigenvalues of P:", eigenvalues)
    print()
```