12.338

BEERAM MADHURI - EE25BTECH11012

October 2025

Question

For a real symmetric matrix $\mathbf{A}$, which of the following statements is true?

a) The matrix is always diagonalizable and invertible.
b) The matrix is always invertible but not necessarily diagonalizable.
c) The matrix is always diagonalizable but not necessarily invertible.
d) The matrix is always neither diagonalizable nor invertible.

finding the properties of matrix A:

Checking for diagonalizability of matrix A
given,

$$\mathbf{A} = \mathbf{A}^\top \tag{1}$$

$\therefore$ eigenvalues of $\mathbf{A}$ are real.
for distinct eigenvalues λ_i, λ_j corresponding eigenvectors are $\mathbf{x_i}$, $\mathbf{x_j}$.

$$\mathbf{Ax_i} = \lambda_i \mathbf{x_i} \quad \text{and} \quad \mathbf{Ax_j} = \lambda_j \mathbf{x_j} \tag{2}$$

$$\mathbf{x_j}^\top \mathbf{Ax_i} = \lambda_i \mathbf{x_j}^\top \mathbf{x_i} \tag{3}$$

$$(\mathbf{Ax_j})^\top \mathbf{x_i} = \lambda_i \mathbf{x_j}^\top \mathbf{x_i} \tag{4}$$

$$\because \quad \mathbf{Ax_j} = \lambda_j \mathbf{x_j} \tag{5}$$

$$\lambda_j \mathbf{x_j}^\top \mathbf{x_i} = \lambda_i \mathbf{x_j}^\top \mathbf{x_i} \tag{6}$$

$$(\lambda_j - \lambda_i)\mathbf{x_j}^\top \mathbf{x_i} = 0 \tag{7}$$

$\therefore$ eigenvectors are orthogonal

$\therefore$ We can construct an orthogonal matrix with these eigenvectors

$$Q = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \mathbf{x_3} & \dots & \mathbf{x_n} \end{bmatrix} \tag{8}$$

$$Q^\top Q = I \tag{9}$$

$$A = QMQ^\top \tag{10}$$

Where **M** is diagonal matrix
$\therefore$ **A** is always diagonalizable.

Checking for invertibility of Matrix **A**:

$$\mathbf{A} = QMQ^\top \tag{11}$$

$$|A| = |Q||M||Q^\top||\mathbf{A}| = M_1 M_2 \cdots M_n \tag{12}$$

where $M_1, M_2, \cdots M_n$ are diagonal entries of Matrix M.
A is invertible only when

$$\det(A) \neq 0 \tag{13}$$

that is $M_1, M_2, M_3 \cdots M_n \neq 0$
that is none of its eigenvalues are zero

if $\lambda_i = 0$
then A is non-invertible

$\therefore$ a real symmetric matrix may or may not be invertible.
$\therefore$ Option c is correct.

Example of a real symmetric matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{14}$$

$$\mathbf{A} = \mathbf{A}^\top \tag{15}$$

$\mathbf{A}$ is symmetric and diagonalizable but not invertible as $\det(A) = 0$

Python Code

```
import numpy as np

# --- Example 1: A real symmetric matrix that IS invertible ---
matrix_A = np.array([
    [3, 1],
    [1, 2]
])
print("## Matrix A ##")
print(matrix_A)
```

Python Code

```python
# Check for symmetry: A == A.transpose()
is_symmetric_A = np.all(matrix_A == matrix_A.T)
print(f"Is symmetric? {is_symmetric_A}")

# Check for invertibility by calculating the determinant
det_A = np.linalg.det(matrix_A)
print(f"Determinant: {det_A:.2f}")
print(f"Is invertible? {det_A != 0}")
```

Python Code

```
print("-" * 20)

# --- Example 2: A real symmetric matrix that is NOT invertible
    ---
matrix_B = np.array([
    [2, 4],
    [4, 8]
])
print("## Matrix B ##")
print(matrix_B)
```

Python Code

```python
# Check for symmetry
is_symmetric_B = np.all(matrix_B == matrix_B.T)
print(f"Is symmetric? {is_symmetric_B}")

# Check for invertibility
det_B = np.linalg.det(matrix_B)
print(f"Determinant: {det_B:.2f}")
print(f"Is invertible? {det_B != 0}")
```

C Code

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

// Define a 2x2 matrix structure
typedef struct {
    double elements[2][2];
} Matrix2x2;

// Function to print a 2x2 matrix
```

C Code

```c
void printMatrix(Matrix2x2 m) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%8.2f", m.elements[i][j]);
        }
        printf("\n");
    }
}
// Function to check if a 2x2 matrix is symmetric
// A matrix A is symmetric if A = A^T (its transpose)
// For a 2x2 matrix, this just means element [0][1] must equal
    element [1][0]
```

C Code

```
bool isSymmetric(Matrix2x2 m) {
    if (m.elements[0][1] == m.elements[1][0]) {
        return true;
    }
    return false;
}
// Function to calculate the determinant of a 2x2 matrix
// For a matrix [[a, b], [c, d]], the determinant is ad - bc
double determinant(Matrix2x2 m) {
    return (m.elements[0][0] * m.elements[1][1]) - (m.elements
        [0][1] * m.elements[1][0]);
}
```

C Code

```c
int main() {
    // Example 1: A real symmetric matrix that IS invertible
    Matrix2x2 matrixA = {
        {{3.0, 1.0}, {1.0, 2.0}}
    };
    printf("## Matrix A ##\n");
    printMatrix(matrixA);
    printf("Is symmetric? %s\n", isSymmetric(matrixA) ? "Yes" : "
        No");
```

C Code

```c
    double detA = determinant(matrixA);
    printf("Determinant: %.2f\n", detA);
    printf("Is invertible? %s\n\n", (detA != 0) ? "Yes" : "No");
    // Example 2: A real symmetric matrix that is NOT invertible
    Matrix2x2 matrixB = {
        {{2.0, 4.0}, {4.0, 8.0}}
    };
```

C Code

```
    printf("## Matrix B ##\n");
    printMatrix(matrixB);
    printf("Is symmetric? %s\n", isSymmetric(matrixB) ? "Yes" : "
        No");
    double detB = determinant(matrixB);
    printf("Determinant: %.2f\n", detB);
    printf("Is invertible? %s\n", (detB != 0) ? "Yes" : "No");
    return 0;
}
```

Python and C Code

```python
import ctypes

# Define a 2x2 matrix structure that is compatible with the C struct
class Matrix2x2(ctypes.Structure):
    """A C-compatible 2x2 matrix structure."""
    fields = [
        ("elements", (ctypes.c_double * 2) * 2)
    ]
# --- Python functions that operate on the C-like structure ---
```

Python and C Code

```python
def print_matrix(m: Matrix2x2):
    """Prints the elements of a Matrix2x2 structure."""
    for i in range(2):
        for j in range(2):
            # Use an f-string for formatted output, similar to
                printf
            print(f"{m.elements[i][j]:8.2f}", end="")
        print()
```

Python and C Code

```python
def is_symmetric(m: Matrix2x2) -> bool:
    """
    Checks if a 2x2 matrix is symmetric.
    A matrix A is symmetric if element [0][1] equals element
        [1][0].
    """
    return m.elements[0][1] == m.elements[1][0]
```

Python and C Code

```python
def determinant(m: Matrix2x2) -> float:
    """
    Calculates the determinant of a 2x2 matrix.
    For a matrix [[a, b], [c, d]], the determinant is ad - bc.
    """
    return (m.elements[0][0] * m.elements[1][1]) - (m.elements
        [0][1] * m.elements[1][0])
```

Python and C Code

```
if _name_ == "_main_":
    # Example 1: A real symmetric matrix that IS invertible
    # Instantiate the structure using nested tuples for the 2D
        array
    matrix_a = Matrix2x2(elements=((3.0, 1.0), (1.0, 2.0)))
    print("## Matrix A ##")
    print_matrix(matrix_a)
    # Use a ternary operator in the f-string to mimic C's output
    print(f"Is symmetric? {'Yes' if is_symmetric(matrix_a) else '
        No'}")
```

```
    det_a = determinant(matrix_a)
    print(f"Determinant: {det_a:.2f}")
    print(f"Is invertible? {'Yes' if det_a != 0 else 'No'}\n")


    # Example 2: A real symmetric matrix that is NOT invertible
    matrix_b = Matrix2x2(elements=((2.0, 4.0), (4.0, 8.0)))
```

Python and C Code

```
    print("## Matrix B ##")
    print_matrix(matrix_b)
    print(f"Is symmetric? {'Yes' if is_symmetric(matrix_b) else 'No'}")

    det_b = determinant(matrix_b)
    print(f"Determinant: {det_b:.2f}")
    print(f"Is invertible? {'Yes' if det_b != 0 else 'No'}")
```