12.888

EE25BTECH11043 - Nishid Khandagre

October 10, 2025

Question

Which one of the following matrices has eigenvalues 1 and 6?

- a) $\begin{pmatrix} 5 & -2 \\ -2 & 2 \end{pmatrix}$
- b) $\begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix}$
- c) $\begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix}$
- d) $\begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}$

Theoretical Solution

To find the eigenvalues λ of a matrix $\mathbf{M}$, we solve the characteristic equation

$$\det\left(\mathbf{M} - \lambda\mathbf{I}\right) = 0 \tag{1}$$

For $\mathbf{A} = \begin{pmatrix} 5 & -2 \\ -2 & 2 \end{pmatrix}$

$$\det\left(\mathbf{A} - \lambda\mathbf{I}\right) = \begin{vmatrix} 5-\lambda & -2 \\ -2 & 2-\lambda \end{vmatrix} \tag{2}$$

$$= (5-\lambda)(2-\lambda) - (-2)(-2) \tag{3}$$

$$= \lambda^2 - 7\lambda + 6 \tag{4}$$

Theoretical Solution

Now, we solve the characteristic equation:

$$\lambda^2 - 7\lambda + 6 = 0 \tag{5}$$

$$(\lambda - 1)(\lambda - 6) = 0 \tag{6}$$

$$\lambda = 1, 6 \tag{7}$$

The eigenvalues are 1 and 6. This matches the requirement.

For $\mathbf{B} = \begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix}$

$$\det\left(\mathbf{B} - \lambda \mathbf{I}\right) = \begin{vmatrix} 3-\lambda & -1 \\ -2 & 2-\lambda \end{vmatrix} \tag{8}$$

$$= (3-\lambda)(2-\lambda) - (-1)(-2) \tag{9}$$

$$= \lambda^2 - 5\lambda + 4 \tag{10}$$

Theoretical Solution

$$\lambda^2 - 5\lambda + 4 = 0 \quad (11)$$
$$(\lambda - 1)(\lambda - 4) = 0 \quad (12)$$
$$\lambda = 1, 4 \quad (13)$$

The eigenvalues are 1 and 4. This does not match the requirement.

For $\mathbf{C} = \begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix}$

$$\det\left(\mathbf{C} - \lambda\mathbf{I}\right) = \begin{vmatrix} 3-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} \quad (14)$$
$$= (3-\lambda)(2-\lambda) - (-1)(-1) \quad (15)$$
$$= \lambda^2 - 5\lambda + 5 \quad (16)$$

Theoretical Solution

$$\lambda^2 - 5\lambda + 5 = 0 \tag{17}$$

$$\lambda = \frac{5 \pm \sqrt{5}}{2} \tag{18}$$

The eigenvalues are $\frac{5+\sqrt{5}}{2}$ and $\frac{5-\sqrt{5}}{2}$. This does not match the requirement.

For $\mathbf{D} = \begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}$

$$\det\left(\mathbf{D} - \lambda\mathbf{I}\right) = \begin{vmatrix} 2-\lambda & -1 \\ -1 & 3-\lambda \end{vmatrix} \tag{19}$$

$$= (2-\lambda)(3-\lambda) - (-1)(-1) \tag{20}$$

$$= \lambda^2 - 5\lambda + 5 \tag{21}$$

Theoretical Solution

$$\lambda^2 - 5\lambda + 5 = 0 \tag{22}$$

$$\lambda = \frac{5 \pm \sqrt{5}}{2} \tag{23}$$

The eigenvalues are $\frac{5+\sqrt{5}}{2}$ and $\frac{5-\sqrt{5}}{2}$. This does not match the requirement.

Only matrix **A** has eigenvalues 1 and 6.

C Code

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

// Function to find eigenvalues of a 2x2 matrix
// mat = [[a, b], [c, d]]
// Returns 0 if real eigenvalues are found, -1 if complex
int findEigenvalues(double a, double b, double c, double d,
    double *lambda1, double *lambda2) {
    double trace = a + d;
    double det = a * d - b * c;
```

C Code

```
    // Characteristic equation: lambda^2 - (trace)lambda + (det) = 0
    // Using quadratic formula: lambda = (-B  sqrt(B^2 - 4AC)) / 2A
    // Here A=1, B=-trace, C=det
    double discriminant = trace * trace - 4 * det;

    if (discriminant >= 0) {
        *lambda1 = (trace + sqrt(discriminant)) / 2.0;
        *lambda2 = (trace - sqrt(discriminant)) / 2.0;
        return 0; // Real eigenvalues
    } else {
        // Complex eigenvalues (not relevant for this problem, but good to handle)
        return -1; // Complex eigenvalues
    }
}
```

Python Code using C Shared Output

```
import ctypes
import numpy as np

# Load the shared library
lib_eigen = ctypes.CDLL(./code27.so)

# Define the argument types and return type for the C function
lib_eigen.findEigenvalues.argtypes = [
    ctypes.c_double, # a
    ctypes.c_double, # b
    ctypes.c_double, # c
    ctypes.c_double, # d
    ctypes.POINTER(ctypes.c_double), # lambda1
    ctypes.POINTER(ctypes.c_double) # lambda2
]
lib_eigen.findEigenvalues.restype = ctypes.c_int
```

Python Code using C Shared Output

```
# Given target eigenvalues
target_eigenvalues = {1.0, 6.0}

matrices = {
    a: np.array([[5, -2], [-2, 2]], dtype=float),
    b: np.array([[3, -1], [-2, 2]], dtype=float),
    c: np.array([[3, -1], [-1, 2]], dtype=float),
    d: np.array([[2, -1], [-1, 3]], dtype=float)
}

print(Checking matrices for eigenvalues 1 and 6:\n)
for label, mat in matrices.items():
    a, b = mat[0, 0], mat[0, 1]
    c, d = mat[1, 0], mat[1, 1]

    lambda1_result = ctypes.c_double()
    lambda2_result = ctypes.c_double()
```

Python Code using C Shared Output

```python
    # Call the C function
    status = lib_eigen.findEigenvalues(
        a, b, c, d,
        ctypes.byref(lambda1_result),
        ctypes.byref(lambda2_result)
    )
    if status == 0:
        found_eigenvalues = {round(lambda1_result.value, 6),
            round(lambda2_result.value, 6)}
        print(fMatrix {label}:\n{mat})
        print(f Calculated eigenvalues: {found_eigenvalues})
        if found_eigenvalues == target_eigenvalues:
            print( This matrix has eigenvalues 1 and 6!\n)
        else:
            print( Eigenvalues do not match 1 and 6.\n)
    else:
        print(fMatrix {label}:\n{mat})
        print(Could not find real eigenvalues)
```

Python Code: Direct

```python
import numpy as np

def check_eigenvalues(matrix, target_eigenvalues):

    Calculates the eigenvalues of a 2x2 matrix and checks if they match
    the target eigenvalues.

    eigenvalues, _ = np.linalg.eig(matrix)

    calculated_eigenvalues = set(round(val.real, 6) for val in eigenvalues)

    return calculated_eigenvalues == target_eigenvalues, calculated_eigenvalues
```

Python Code: Direct

```
# Define the target eigenvalues
target_eigenvalues = {1.0, 6.0}

# Define the matrices as NumPy arrays
matrices = {
    a: np.array([[5, -2], [-2, 2]]),
    b: np.array([[3, -1], [-2, 2]]),
    c: np.array([[3, -1], [-1, 2]]),
    d: np.array([[2, -1], [-1, 3]])
}

print(fSearching for a matrix with eigenvalues: {
    target_eigenvalues}\n)
```

Python Code: Direct

```python
# Iterate through each matrix and check its eigenvalues
for label, matrix in matrices.items():
    print(fChecking Matrix {label}:\n{matrix})
    match, calculated_eigs = check_eigenvalues(matrix, target_eigenvalues)

    if match:
        print(f Result: This matrix has the target eigenvalues {calculated_eigs}!\n)
    else:
        print(f Result: Calculated eigenvalues are {calculated_eigs}, which do NOT match the target.\n)
```