4.8.14

Hemanth Reddy-AI25BTECH11018

September 30, 2025

Question

If $\mathbf{A} = \begin{pmatrix} 3 & -2 \\ 4 & -2 \end{pmatrix}$ and $\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, find k so that $\mathbf{A}^2 = k\mathbf{A} - 2\mathbf{I}$.

Solution:

The characteristic equation for a matrix **A** is f(λ) = **A** - λ **I** = 0

$$\mathbf{A} - \lambda\mathbf{I} = \begin{vmatrix} 3-\lambda & -2 \\ 4 & -2-\lambda \end{vmatrix} = 0 \tag{1}$$

Upon expanding we get λ^2- λ+2=0

$$\lambda^2 = \lambda - 2 \tag{2}$$

Using the Cayley-Hamilton theorem f(λ) = f(**A**)=0

$$\mathbf{A}^2 = \mathbf{A} - 2\mathbf{I} \tag{3}$$

Value of k=1

C Code

```c
#include <stdio.h>

// Function to multiply two 2x2 matrices (A * A)
void multiply_matrices(double result[2][2], double A[2][2],
    double B[2][2]) {
    result[0][0] = A[0][0] * B[0][0] + A[0][1] * B[1][0];
    result[0][1] = A[0][0] * B[0][1] + A[0][1] * B[1][1];
    result[1][0] = A[1][0] * B[0][0] + A[1][1] * B[1][0];
    result[1][1] = A[1][0] * B[0][1] + A[1][1] * B[1][1];
}

// Function to multiply a 2x2 matrix by a scalar
void scalar_multiply(double result[2][2], double matrix[2][2],
    double k) {
    result[0][0] = k * matrix[0][0];
    result[0][1] = k * matrix[0][1];
    result[1][0] = k * matrix[1][0];
    result[1][1] = k * matrix[1][1];
```

C Code

```c

// Function to subtract two 2x2 matrices with a scalar multiplication
void subtract_matrices(double result[2][2], double A[2][2], double I[2][2]) {
    result[0][0] = A[0][0] - 2 * I[0][0];
    result[0][1] = A[0][1] - 2 * I[0][1];
    result[1][0] = A[1][0] - 2 * I[1][0];
    result[1][1] = A[1][1] - 2 * I[1][1];
}

int main() {
    // Define the matrix A and the identity matrix I
    double A[2][2] = {{3.0, -2.0}, {4.0, -2.0}};
    double I[2][2] = {{1.0, 0.0}, {0.0, 1.0}};

    // Calculate A^2
    double A_squared[2][2];
    multiply_matrices(A_squared, A, A);
```

C Code

```
// Solve for k using one element of the matrix equation
    // Equating the (0,0) elements: (A^2)[0][0] = k*A[0][0] - 2*I
        [0][0]
    // A^2[0][0] = k*3 - 2*1
    // A_squared[0][0] + 2 = 3*k
    double k_numerator = A_squared[0][0] + 2.0;
    double k_denominator = A[0][0];

    double k_val = k_numerator / k_denominator;

    // Verify the solution with another element for consistency
    // Equating the (1,1) elements: (A^2)[1][1] = k*A[1][1] - 2*I
        [1][1]
    // A^2[1][1] = k*(-2) - 2*1
    // A^2[1][1] + 2 = -2*k
    double k_val_check = (A_squared[1][1] + 2.0) / A[1][1];
```

C Code

```c
printf("Calculated value of k from (0,0) element: %.2f\n", k_val);
    printf("Calculated value of k from (1,1) element: %.2f\n\n", k_val_check);

    // Print the final result and verify the equation
    printf("The value of k that satisfies A^2 = kA - 2I is k = %.2f\n", k_val);

    return 0;
}
```

Python Code

```



import sys
#for path to external scripts
sys.path.insert(0, '/sdcard/github/matgeo/codes/CoordGeo')
#path to my scripts
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

#local imports
#from line.funcs import *
#from triangle.funcs import *
#from matrix.funcs import *
```

Python Code

```python
#from conics.funcs import circ_gen


#if using termux
import subprocess
import shlex
#end if

def solve_for_k_cayley_hamilton(A, I):
    """
    Solves for the scalar k in the matrix equation A^2 = kA - 2I
    by using the Cayley-Hamilton Theorem.

    Args:
        A (np.array): A 2x2 NumPy array representing matrix A.
        I (np.array): A 2x2 NumPy array representing the identity
            matrix I.
```

Python Code

```
Returns:
        float: The value of k that satisfies the equation.
    """
    # Step 1: Find the characteristic polynomial coefficients of
        A
    # The characteristic polynomial is lambda^2 - (tr(A))lambda +
         det(A) = 0
    coeffs = np.poly(A)
    # The coefficients will be [1, -tr(A), det(A)]

    # Step 2: Apply the Cayley-Hamilton Theorem
    # The matrix A satisfies its characteristic equation:
    # A^2 + coeffs[1]*A + coeffs[2]*I = 0

    # Rearranging the Cayley-Hamilton equation:
    # A^2 = -coeffs[1]*A - coeffs[2]*I
```

Python Code

```
# Step 3: Compare with the given equation
    # Given: A^2 = kA - 2I
    # By comparing the coefficients of A and I, we find:
    # k = -coeffs[1]
    # -2 = -coeffs[2] => coeffs[2] = 2

    # The value of k is the negative of the second coefficient
        from np.poly(A)
    k_value = -coeffs[1]

    # Verification check to see if the determinant is correct
    if np.isclose(coeffs[2], 2):
        print("Verification successful: The determinant from the
            characteristic polynomial matches the problem
            statement.")
    else:
        print("Verification failed: The determinant does not
            match the problem statement.")
```

Python Code

```python
    return k_value

# Given matrices
A = np.array([[3, -2],
              [4, -2]])
I = np.array([[1, 0],
              [0, 1]])

# Find the value of k
k_value = solve_for_k_cayley_hamilton(A, I)

# Print the final result
if k_value is not None:
    print(f"\nThe value of k is: {k_value}")
```