5.13.74

Sai Krishna Bakki - EE25BTECH11049

Question

The trace of a square matrix is defined to be the sum of its diagonal entries. If $\mathbf{A}$ is a 2 x 2 matrix, such that the trace of $\mathbf{A}$ is 3 and the trace of $\mathbf{A}^3$ is -18, then the value of the determinant of $\mathbf{A}$ is

Theoretical Solution

Given:

$$\text{tr}(\mathbf{A}) = 3 \tag{1}$$
$$\text{tr}(\mathbf{A^3}) = -18 \tag{2}$$

Let the eigenvalues of 2x2 matrix A be λ_1 and λ_2, we know that trace is the sum of eigenvalues.

$$\lambda_1 + \lambda_2 = 3 \tag{3}$$

we are given that $\text{tr}(\mathbf{A^3}) = -18$. Since the eigenvalues of $\mathbf{A}^3$ are λ_1^3 and λ_2^3, the trace of $\mathbf{A}^3$ is their sum.

$$\lambda_1^3 + \lambda_2^3 = -18 \tag{4}$$

We can use the algebraic identity for the sum of cubes to connect our two equations (3) and (4).

$$\lambda_1^3 + \lambda_2^3 = (\lambda_1 + \lambda_2)((\lambda_1 + \lambda_2)^2 - 3\lambda_1\lambda_2) \tag{5}$$

Substituting the equations (3) and (4) in above equation, we get

$$\lambda_1\lambda_2 = 5 \tag{6}$$

But determinant of **A** is $\lambda_1\lambda_2$.
Therefore, the value of the determinant of **A** is 5.

C Code

```c
#include <math.h>
#if defined(_WIN32)
    #define DLLEXPORT __declspec(dllexport)
#else
    #define DLLEXPORT
#endif

DLLEXPORT double solve_determinant_2x2(double trace_A, double trace_A3) {
    // Ensure we don't divide by zero if trace_A is 0.
    if (trace_A == 0) {
        return 0.0; // Or handle as an error, e.g., return NAN.
    }
    // Formula: det(A) = (tr(A)^3 - tr(A^3)) / (3 * tr(A))
    return (pow(trace_A, 3) - trace_A3) / (3.0 * trace_A);
}
```

Python Code Through Shared Output

```python
import ctypes
import os

# Define the name of the shared library based on the operating system
if os.name == 'nt': # Windows
    lib_name = 'solver.dll'
else: # Linux, macOS, etc.
    lib_name = 'solver.so'

# Construct the full path to the library file in the current directory
lib_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), lib_name)

try:
    # 1. Load the shared library
    solver_lib = ctypes.CDLL(lib_path)
except OSError as e:
```

Python Code Through Shared Output

```
    print(fError: Could not load the shared library '{lib_name}'.)
    print(Please make sure you have compiled the C code first.)
    print(fDetails: {e})
    exit()

# 2. Define the function signature to match the C code
# Specify the argument types (argtypes)
solver_lib.solve_determinant_2x2.argtypes = [ctypes.c_double, ctypes.c_double]
# Specify the return type (restype)
solver_lib.solve_determinant_2x2.restype = ctypes.c_double

# 3. Define the input values from the problem
trace_A = 3.0
trace_A3 = -18.0

# 4. Call the C function from Python
```

Python Code Through Shared Output

```
# Python floats will be automatically converted to ctypes.c_double
determinant = solver_lib.solve_determinant_2x2(trace_A, trace_A3)

# 5. Print the result
print(--- Calling C function from Python using ctypes ---)
print(fGiven tr(A) = {trace_A})
print(fGiven tr(A^3) = {trace_A3})
print(- * 25)
print(fThe calculated determinant of A is: {determinant})
```

Python Code

```
import sympy

# 1. Define the unknown variable and knowns as symbolic objects
# d represents the determinant of A, which we want to find.
d = sympy.Symbol('d')

# tr_A is the trace of A.
tr_A = 3
# tr_A3 is the trace of A^3.
tr_A3 = -18
# For a 2x2 matrix, the trace of the identity matrix (I) is 2.
tr_I = 2

# 2. Set up the equation based on the Cayley-Hamilton theorem
# The derived formula is: tr(A^3) = (tr(A)**2 - d)*tr(A) - d*tr(A
    )*tr(I)
```

Python Code

```
# We create an equation object that is equal to zero.
equation = sympy.Eq((tr_A**2 - d)*tr_A - d*tr_A*tr_I, tr_A3)

# 3. Solve the equation for our unknown variable 'd'
# sympy.solve takes the equation and the variable to solve for.
solution = sympy.solve(equation, d)

# 4. Print the result
# The solution is a list, so we print the first element.
print(fThe equation to solve is: {equation})
print(fThe calculated determinant of A is: {solution[0]})
```