12.894

Sai Krishna Bakki - EE25BTECH11049

Question

The eigenvalues of the matrix

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

are

Theoretical Solution

Given

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

To find eigenvalues of the matrix $\mathbf{A}$

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{2}$$

$$(\mathbf{A} - \lambda\mathbf{I})\,\mathbf{x} = 0 \tag{3}$$

$$\left|\mathbf{A} - \lambda\mathbf{I}\right| = 0 \tag{4}$$

$$\begin{vmatrix} 0-\lambda & -1 \\ 1 & 0-\lambda \end{vmatrix} = 0 \tag{5}$$

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

$$\lambda^2 = -1 \tag{7}$$

$$\lambda = -\sqrt{-1}, \sqrt{-1} \tag{8}$$

C Code

```c
#include <math.h>

void find_2x2_eigenvalues(double a, double b, double c, double d,
                          double* eig1_real, double* eig1_imag
                              ,
                          double* eig2_real, double* eig2_imag
                              ) {
    double trace = a + d;
    double determinant = a * d - b * c;
    double discriminant = trace * trace - 4 * determinant;

    if (discriminant >= 0) {
        // Real eigenvalues
        double sqrt_discriminant = sqrt(discriminant);
        *eig1_real = (trace + sqrt_discriminant) / 2.0;
        *eig1_imag = 0.0;
        *eig2_real = (trace - sqrt_discriminant) / 2.0;
        *eig2_imag = 0.0;
```

C Code

```
    } else {
        // Complex conjugate eigenvalues
        double sqrt_abs_discriminant = sqrt(-discriminant);
        *eig1_real = trace / 2.0;
        *eig1_imag = sqrt_abs_discriminant / 2.0;
        *eig2_real = trace / 2.0;
        *eig2_imag = -sqrt_abs_discriminant / 2.0;
    }
}
```

Python Code Through Shared Output

```
import ctypes
import numpy as np
import os
import platform

# --- Step 1: Compile the C code into a shared library ---
c_file_name = 'eigenv.c'

# Determine the correct file extension for the shared library and
    compile command
if platform.system() == Windows:
    lib_name = 'eigen_lib.dll'
    compile_command = fgcc -shared -o {lib_name} -fPIC {
        c_file_name}
elif platform.system() == Darwin: # macOS
    lib_name = 'eigen_lib.dylib'
    compile_command = fgcc -shared -o {lib_name} -fPIC {
        c_file_name}
else: # Linux
```

Python Code Through Shared Output

```
    lib_name = 'eigenv.so'
    # Add -lm to link the math library on Linux/macOS
    compile_command = fgcc -shared -o {lib_name} -fPIC {
        c_file_name} -lm

# Compile the C code if the library file doesn't exist
if not os.path.exists(lib_name):
    print(fShared library not found. Compiling '{c_file_name}'...
        )
    if os.system(compile_command) != 0:
        print(f\nError: Compilation failed. Please ensure GCC is
            installed.)
        exit()
    print(Compilation successful.)

# --- Step 2: Load the shared library ---
try:
    eigen_lib = ctypes.CDLL(os.path.abspath(lib_name))
except OSError as e:
```

Python Code Through Shared Output

```
    print(fError loading shared library: {e})
    exit()

# --- Step 3: Define the function signature ---
# The modified C function signature is:
# void find_2x2_eigenvalues(double, double, double, double,
    double*, double*, double*, double*)
find_2x2_c = eigen_lib.find_2x2_eigenvalues
find_2x2_c.argtypes = [
    ctypes.c_double, ctypes.c_double, ctypes.c_double, ctypes.
        c_double,
    ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.
        c_double),
    ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.
        c_double)
]
find_2x2_c.restype = None
# --- Step 4: Prepare data and call the C function ---
# The matrix from the image is:
```

Python Code Through Shared Output

```
# [[0, -1],
# [1, 0]]
matrix = np.array([[0, -1], [1, 0]])
a, b = float(matrix[0, 0]), float(matrix[0, 1])
c, d = float(matrix[1, 0]), float(matrix[1, 1])

# Create C-compatible double variables to hold the real and
    imaginary parts
eig1_real, eig1_imag = ctypes.c_double(), ctypes.c_double()
eig2_real, eig2_imag = ctypes.c_double(), ctypes.c_double()

print(fCalling C function to find eigenvalues of the matrix:\n{
    matrix}\n)
```

Python Code Through Shared Output

```
# Call the C function, passing pointers to the result variables
find_2x2_c(a, b, c, d,
           ctypes.byref(eig1_real), ctypes.byref(eig1_imag),
           ctypes.byref(eig2_real), ctypes.byref(eig2_imag))

# --- Step 5: Retrieve the results and combine them into complex
    numbers ---
eigenvalue1 = complex(eig1_real.value, eig1_imag.value)
eigenvalue2 = complex(eig2_real.value, eig2_imag.value)

print(fEigenvalues from C function are: {eigenvalue1} and {
    eigenvalue2})
```

Python Code

```
import numpy as np
A = np.array([[0, -1],
              [1, 0]])

# Use numpy's linear algebra module to find the eigenvalues.
eigenvalues = np.linalg.eigvals(A)

# Print the original matrix.
print(Matrix:)
print(A)
formatted_vals = [f{int(val.imag)}j for val in eigenvalues]

print(\nEigenvalues:)
print(fThe eigenvalues are {formatted_vals[0]} and {
    formatted_vals[1]})
```