1.5.36

Sai Krishna Bakki - EE25BTECH11049

Question

Find the unit vector in the direction of the vector $a = \hat{i} + \hat{j} + 2\hat{k}$

Theoretical Solution

Given

$$\mathbf{A} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \tag{1}$$

$$||\mathbf{a}|| = \sqrt{(1)^2 + (1)^2 + (2)^2} = \sqrt{6} \tag{2}$$

The unit vector in the direction of **a** is

$$\frac{\mathbf{a}}{||\mathbf{a}||} = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \tag{3}$$

C Code

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


// Function to calculate the unit vector.
// It takes an input vector, its size, and an output array for
    the result.
void calculate_unit_vector_c(double *vector, int size, double *
    unit_vector_out) {
    double magnitude = 0.0;
    int i;

    // Calculate the magnitude of the vector
    for (i = 0; i < size; i++) {
        magnitude += vector[i] * vector[i];
    }
    magnitude = sqrt(magnitude);
```

C Code

```c

    // To avoid division by zero, if magnitude is 0, return a
        zero vector.
    if (magnitude == 0) {
        for (i = 0; i < size; i++) {
            unit_vector_out[i] = 0.0;
        }
    } else {
        // Calculate the unit vector
        for (i = 0; i < size; i++) {
            unit_vector_out[i] = vector[i] / magnitude;
        }
    }
}
```

Python Code through shared output

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import ctypes
from numpy.ctypeslib import ndpointer

# This script uses the compiled C library 'vector_lib.so'
# to calculate the unit vector.
#
# Before running this script, you must compile the C code by
    running
# the following command in your terminal:
# python setup.py build_ext --inplace

try:
    # --- 1. Load the C Shared Library ---
    # This loads the .so (Linux/macOS) or .dll (Windows) file.
    vector_lib = ctypes.CDLL('./vector_lib.so')
```

Python Code through shared output

```

# --- 2. Define the C function's argument and return types
    ---
# This tells Python how to correctly call the C function.
calculate_unit_vector_c = vector_lib.calculate_unit_vector_c
calculate_unit_vector_c.restype = None # The C function
    returns void
calculate_unit_vector_c.argtypes = [
    ndpointer(dtype=np.float64, flags=C_CONTIGUOUS), #
        Pointer to the input vector
    ctypes.c_int, # Integer for the vector size
    ndpointer(dtype=np.float64, flags=C_CONTIGUOUS) # Pointer
         to the output array
]

# --- 3. Prepare data and call the C function ---
from params import a_vector
```

Python Code through shared output

```
# Ensure the vector is a float64 numpy array for the C
    function
a = a_vector.astype(np.float64)

# Create an empty numpy array to store the result from C
unit_a = np.zeros_like(a, dtype=np.float64)

# Call the C function
calculate_unit_vector_c(a, a.size, unit_a)

# --- 4. Print and Plot the results ---
magnitude_a = np.linalg.norm(a) # Calculate magnitude in
    Python for display
```

Python Code through shared output

```
print(fOriginal vector a: {a})
print(fMagnitude of a: {magnitude_a:.4f})
print(fUnit vector  (from C library): {unit_a})
print(fMagnitude of the unit vector: {np.linalg.norm(unit_a)
    :.4f})


# Create the 3D plot
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

# Plot the original vector 'a' and the unit vector ''
ax.quiver(0, 0, 0, a[0], a[1], a[2], color='b',
    arrow_length_ratio=0.1, label=f'Vector a = {a}')
ax.quiver(0, 0, 0, unit_a[0], unit_a[1], unit_a[2], color='r'
    , arrow_length_ratio=0.2, label=f'Unit Vector   [{unit_a
    [0]:.2f}, {unit_a[1]:.2f}, {unit_a[2]:.2f}]')
```

Python Code through shared output

```
# Configure and display the plot
limit = max(np.max(np.abs(a)), 1.5)
ax.set_xlim([-limit, limit]); ax.set_ylim([-limit, limit]);
    ax.set_zlim([0, limit])
ax.set_title('Unit Vector in 3D (Calculated in C)', fontsize
    =16)
ax.set_xlabel('x-axis'); ax.set_ylabel('y-axis'); ax.
    set_zlabel('z-axis')
ax.legend(fontsize=12)
ax.view_init(elev=20., azim=30)
plt.grid(True)
plt.show()
```

Python Code through shared output

```python

except FileNotFoundError:
    print(Error: Could not find 'vector_lib.so' or 'vector_lib.
        dll'.)
    print(Please ensure you have compiled the C library first by
        running this command in your terminal:)
    print(python setup.py build_ext --inplace)
except Exception as e:
    print(fAn unexpected error occurred: {e})
```

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from funcs import calculate_unit_vector
from params import a_vector # Import the specific vector

# The vector a = (1, 1, 2) is now defined in params.py
# This script calculates its unit vector and creates a 3D plot
    for visualization.

try:
    # Calculate the unit vector using the function from funcs.py
    a, magnitude_a, unit_a = calculate_unit_vector(a_vector)

    # Print the results
    print(fOriginal vector a: {a})
    print(fMagnitude of a: {magnitude_a:.4f})
    print(fUnit vector : {unit_a})
```

Python Code

```
print(fMagnitude of the unit vector: {np.linalg.norm(unit_a)
    :.4f})


# --- Plotting the Vectors ---

# Create the 3D plot
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

# Plot the original vector 'a' in blue
ax.quiver(0, 0, 0, a[0], a[1], a[2], color='b',
    arrow_length_ratio=0.1, label=f'Vector a = {a}')

# Plot the unit vector '' in red
ax.quiver(0, 0, 0, unit_a[0], unit_a[1], unit_a[2], color='r'
    , arrow_length_ratio=0.2, label=f'Unit Vector   [{unit_a
    [0]:.2f}, {unit_a[1]:.2f}, {unit_a[2]:.2f}]')
```

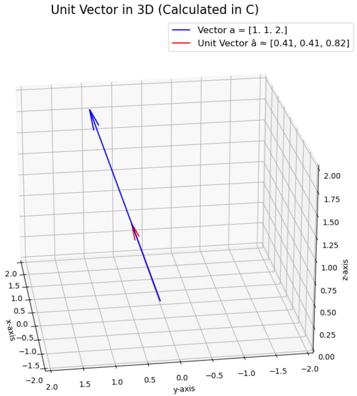
Python Code

```

# Set the plot limits for better visualization
limit = max(np.max(np.abs(a)), 1.5)
ax.set_xlim([-limit, limit])
ax.set_ylim([-limit, limit])
ax.set_zlim([0, limit])

# Set plot title and labels
ax.set_title('Unit Vector in 3D', fontsize=16)
ax.set_xlabel('x-axis', fontsize=12)
ax.set_ylabel('y-axis', fontsize=12)
ax.set_zlabel('z-axis', fontsize=12)

# Add a legend
ax.legend(fontsize=12)

# Set the view angle
ax.view_init(elev=20., azim=30)
```

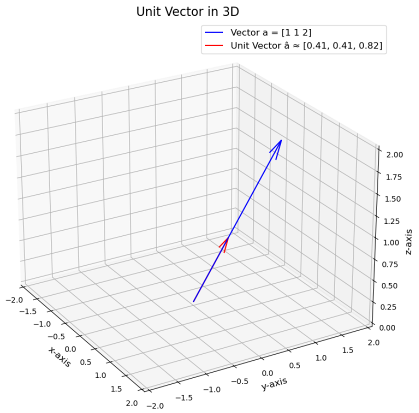
Python Code

```

    # Display the plot
    plt.show()

except (ValueError, NameError) as e:
    print(fAn error occurred: {e})
    print(Please ensure that 'params.py' contains 'a_vector' and
        'funcs.py' contains 'calculate_unit_vector'.)
```

Plot by python using shared output from c



Plot by using Python only


Unit Vector in 3D
Vector a = [1 1 2]
Unit Vector â ≈ [0.41, 0.41, 0.82]
x-axis
y-axis
z-axis