2.9.26

Dhanush Kumar A - AI25BTECH11010

September 10, 2025

Question

If $f(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$, prove that $f(\alpha)f(-\beta) = f(\alpha - \beta)$.

Solution

We have

$$f(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

$$f(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$f(-\beta) = \begin{pmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

$$f(\alpha)f(-\beta) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4}$$

Solution

$$= \begin{pmatrix} \cos\alpha\cos\beta + \sin\alpha\sin\beta & \cos\alpha\sin\beta - \sin\alpha\cos\beta & 0 \\ \sin\alpha\cos\beta - \cos\alpha\sin\beta & \sin\alpha\sin\beta + \cos\alpha\cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{6}$$

$$= \begin{pmatrix} \cos(\alpha-\beta) & -\sin(\alpha-\beta) & 0 \\ \sin(\alpha-\beta) & \cos(\alpha-\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{7}$$

$$= f(\alpha-\beta). \tag{8}$$

Thus proved.

Python code - Verify the result

```python
import numpy as np

def f(theta):
    return np.array([
        [np.cos(theta), -np.sin(theta), 0],
        [np.sin(theta), np.cos(theta), 0],
        [0, 0, 1]
    ])

# Take input
alpha = float(input("Enter alpha (in radians): "))
beta = float(input("Enter beta (in radians): "))

# Compute both sides
lhs = f(alpha) @ f(-beta)
rhs = f(alpha - beta)
```

Python code - Verify the result

```python


# Check equality (within tolerance, since floats may not be exact
    )
if np.allclose(lhs, rhs, atol=1e-9):
    print("Verified: f(alpha) f(-beta) = f(alpha - beta)")
else:
    print(" Not equal")
    print("LHS =\n", lhs)
    print("RHS =\n", rhs)
```

Output of Python code

```
Enter alpha (in radians): 4
Enter beta (in radians): 5
Verified: f(alpha) f(-beta) = f(alpha - beta)
```

C code - Verify the result

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

#define SIZE 3
#define EPS 1e-9 // tolerance for floating-point comparison

// Function to build matrix f(theta)
void f(double theta, double M[SIZE][SIZE]) {
    M[0][0] = cos(theta); M[0][1] = -sin(theta); M[0][2] = 0;
    M[1][0] = sin(theta); M[1][1] = cos(theta); M[1][2] = 0;
    M[2][0] = 0; M[2][1] = 0; M[2][2] = 1;
}
```

C code - Verify the result

```c

// Multiply two 3x3 matrices
void multiply(double A[SIZE][SIZE], double B[SIZE][SIZE], double
    C[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            C[i][j] = 0;
            for (int k = 0; k < SIZE; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

C code - Verify the result

```c

// Check if two matrices are approximately equal
bool equal(double A[SIZE][SIZE], double B[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (fabs(A[i][j] - B[i][j]) > EPS)
                return false;
        }
    }
    return true;
}
```

C code - Verify the result

```c

// Print a 3x3 matrix
void printMatrix(double M[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%10.6f ", M[i][j]);
        }
        printf("\n");
    }
}
```

C code - Verify the result

```c

int main() {
    double alpha, beta;
    printf("Enter alpha (in radians): ");
    scanf("%lf", &alpha);
    printf("Enter beta (in radians): ");
    scanf("%lf", &beta);

    double F_alpha[SIZE][SIZE], F_minus_beta[SIZE][SIZE],
        F_alpha_minus_beta[SIZE][SIZE];
    double lhs[SIZE][SIZE], rhs[SIZE][SIZE];
```

C code - Verify the result

```c

// Build matrices
f(alpha, F_alpha);
f(-beta, F_minus_beta);
f(alpha - beta, F_alpha_minus_beta);

// Compute lhs = f(alpha) * f(-beta)
multiply(F_alpha, F_minus_beta, lhs);

// rhs = f(alpha - beta)
for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
        rhs[i][j] = F_alpha_minus_beta[i][j];
```

C code - Verify the result

```c

    // Compare
    if (equal(lhs, rhs)) {
        printf("\n Verified: f(alpha) f(-beta) = f(alpha - beta)\
            n");
    } else {
        printf("\nNot equal!\n");
        printf("\nLHS =\n"); printMatrix(lhs);
        printf("\nRHS =\n"); printMatrix(rhs);
    }

    return 0;
}
```

Output of C code

```
Enter alpha (in radians): 2
Enter beta (in radians): 1.23

 Verified: f(alpha) f(-beta) = f(alpha - beta)
```