MATGEO: 1.1.6.19

Y Siddhanth
EE24BTECH11059
EE1030

November 6, 2024

Problem Statement

The vectors $\lambda\hat{i} + \hat{j} + 2\hat{k}$, $\hat{i} + \lambda\hat{j} - \hat{k}$ and $2\hat{i} - \hat{j} + \lambda\hat{k}$ are coplanar if $\lambda =?$

Information Table

Variable	Description	Formula
A	A vector in terms of λ	$A = \begin{pmatrix} \lambda \\ 1 \\ 2 \end{pmatrix}$
B	A vector in terms of λ	$B = \begin{pmatrix} 1 \\ \lambda \\ -1 \end{pmatrix}$
C	A vector in terms of λ	$C = \begin{pmatrix} 2 \\ -1 \\ \lambda \end{pmatrix}$
M	It is a matrix comprising of vectors A, B, C	$M = [A, B, C]$
λ	It is the variable with which A, B, C are established	$\lvert M \rvert = 0$

Method of Solving

The rank of a matrix M is less than 3, then the matrix is coplanar.

$$Rank\,(M) = 2 \text{ or } 1 \tag{3.1}$$

Equivalently,

$$|M| = 0 \tag{3.2}$$

Finding λ values

$$|M| = \begin{pmatrix} \lambda & 1 & 2 \\ 1 & \lambda & -1 \\ 2 & -1 & \lambda \end{pmatrix} = 0 \tag{3.3}$$

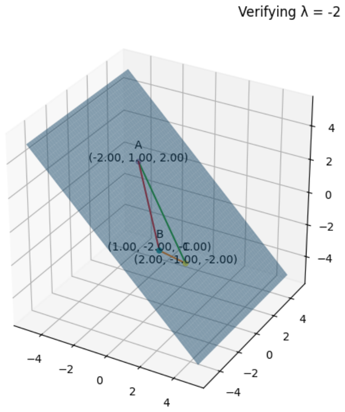
$$\lambda\left(\lambda^2 - 1\right) - 1\left(\lambda + 2\right) + 2\left(-1 - 2\lambda\right) = 0 \tag{3.4}$$

$$\lambda^3 - 6\lambda - 4 = 0 \tag{3.5}$$

$$\left(\lambda + 2\right)\left(\lambda^2 - 2\lambda - 2\right) = 0 \tag{3.6}$$

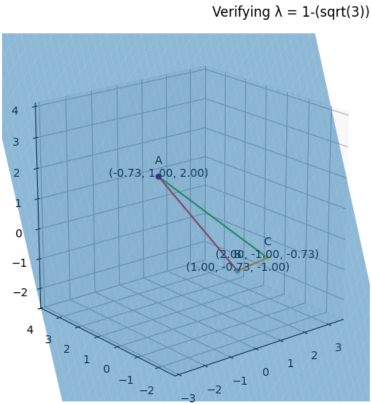
$$\lambda = -2 \text{ or } \lambda = 1 \pm \sqrt{3} \tag{3.7}$$

Verifying Obtained Values



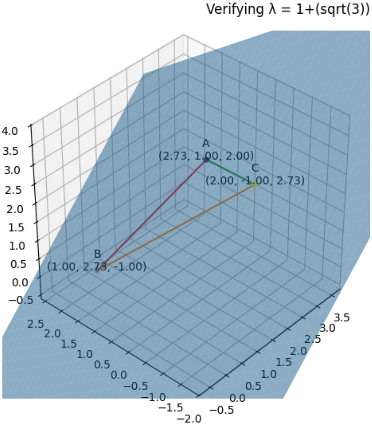
Figure

Verifying Obtained Values



Figure

Verifying Obtained Values



Figure

λ Verification Code

```
import sys #for path to external scripts
sys.path.insert(0, '/home/ysiddhanth/Documents/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
from mpl_toolkits.mplot3d import Axes3D

#local imports
from line.funcs import *
from triangle.funcs import *
from conics.funcs import circ_gen
roots = np.loadtxt("lambda.dat")
#roots are roots[0] = -2; roots[1] and roots[2]
k = roots[2]
A = np.array(([k, 1,2])).reshape(-1,1)
B = np.array(([1,k, -1])).reshape(-1,1)
C = np.array(([2,-1, k])).reshape(-1,1)
```

λ Verification Code

```
P1 = np.array(([k, 1,2]))
P2 = np.array(([1,k, -1]))
P3 = np.array(([2,-1, k]))
v1 = P2 - P1
v2 = P3 - P1
# Compute the normal vector (cross product of v1 and v2)
normal = np.cross(v1, v2)
# Plane equation: ax + by + cz = d
# where [a, b, c] is the normal vector and d is
    calculated using one of the points
a, b, c = normal
d = np.dot(normal, P1)
# Create a figure and a 3D Axes
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
# Generate grid points for x and y
x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
```

λ Verification Code

```
# Calculate corresponding z values for each (x, y) pair
    to satisfy the plane equation
Z = (-a*X - b*Y - d) / c
# Plot the plane
ax.plot_surface(X, Y, Z, alpha=0.5)
#Generating all lines
x_BC = line_gen(B,C)
x_AB = line_gen(A,B)
x_AC = line_gen(A,C)
#Plotting all lines
ax.plot(x_BC[0,:],x_BC[1,:], x_BC[2,:],label='BC')
ax.plot(x_AC[0,:],x_AC[1,:], x_AC[2,:],label='AC')
ax.plot(x_AB[0,:],x_AB[1,:], x_AB[2,:],label='AB')
# Scatter plot
colors = np.arange(2, 5) # Example colors
tri_coords = np.block([A, B, C]) # VStack A, B, C
ax.scatter(tri_coords[0, :], tri_coords[1, :], tri_coords
    [2, :], c=colors)
vert_labels = ['A', 'B', 'C']
```

λ Verification Code

```
for i, txt in enumerate(vert_labels):
# Annotate each point with its label and coordinates
ax.text(tri_coords[0, i], tri_coords[1, i], tri_coords[2,
    i], f'{txt}\n({tri_coords[0, i]:.02f}, {tri_coords[1,
    i]:.02f}, {tri_coords[2, i]:.02f})',
fontsize=10, ha='center', va='baseline')
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
# Set limits and aspect ratio to magnify the plane
ax.set_xlim(-4, 4) # Adjust limits based on your data
ax.set_ylim(-4, 4) # Adjust limits based on your data
ax.set_zlim(-4, 4) # Adjust limits based on your data
ax.set_box_aspect([1,1,1]) # Equal aspect ratio for x, y,
    and z axes
plt.grid() # minor
plt.axis('equal')
plt.show()
```

Methods of Finding λ

Method 1(using sympy) : Use sympy to find the det and solve for lambda.

```
import sympy as sp
λ = sp.symbols('λ')
A = sp.Matrix([[λ, 1, 2], [1, λ, -1], [2, -1, λ]])
det_A = A.det()
lambda_solutions = sp.solve(det_A, λ)
numeric_solutions = [sp.N(sol) for sol in
    lambda_solutions]
```

Methods of Finding λ

Method 2(using the QR algorithm(numpy or C)) :

$$|M| = \begin{pmatrix} \lambda & 1 & 2 \\ 1 & \lambda & -1 \\ 2 & -1 & \lambda \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & -1 \\ 2 & -1 & 0 \end{pmatrix} + \lambda I = |A + \lambda I| \tag{3.8}$$

Thus, the values of λ are the negative of the eigen values of the matrix A.

```
import numpy as np
A = np.array([[0, 1, 2], [1, 0. -1], [2, -1, 0]])
eigenvalues = np.linalg.eigvals(A)
λ-vals = -eigenvals
```