

2.10.12

Pratik R-AI25BTECH11023

September 9, 2025

# Question

A unit vector perpendicular to the plane determined by the points  $P(1, -1, 2)$ ,  $Q(2, 0, -1)$  and  $R(0, 2, 1)$  is

# Solution

According to the question,  
Given the position vectors,

$$\mathbf{P} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}; \mathbf{Q} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix}; \mathbf{R} = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \quad (1)$$

$$\mathbf{A} = \mathbf{Q} - \mathbf{P} = \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix} \quad (2)$$

$$\mathbf{B} = \mathbf{R} - \mathbf{P} = \begin{pmatrix} -1 \\ 3 \\ -1 \end{pmatrix} \quad (3)$$

# Equation

we need to find the unit vector which is perpendicular to the vectors **A** and **B**. The vector perpendicular to **A** and **B** is given by their cross-product.

Let the perpendicular vector be  $\mathbf{X}^T = \begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix}$

$$\therefore \mathbf{A}^T \mathbf{X} = 0 \quad (4)$$

$$\mathbf{B}^T \mathbf{X} = 0, \quad (5)$$

$$\therefore \begin{pmatrix} \mathbf{A}^T \\ \mathbf{B}^T \end{pmatrix} \mathbf{X} = 0 \quad (6)$$

$$\begin{pmatrix} 1 & 1 & -3 \\ -1 & 3 & -1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = 0 \quad (7)$$

# Solution

This can be represented as,

$$\begin{pmatrix} 1 & 1 & -3 \\ -1 & 3 & -1 \end{pmatrix} \xleftrightarrow{R_2 \leftarrow R_2 + R_1} \begin{pmatrix} 1 & 1 & -3 \\ 0 & 4 & -4 \end{pmatrix} \quad (8)$$

yielding,

$$x_1 + x_2 - 3x_3 = 0 \quad (9)$$

$$4x_2 - 4x_3 = 0 \quad (10)$$

$$\implies x_2 = x_3 \quad (11)$$

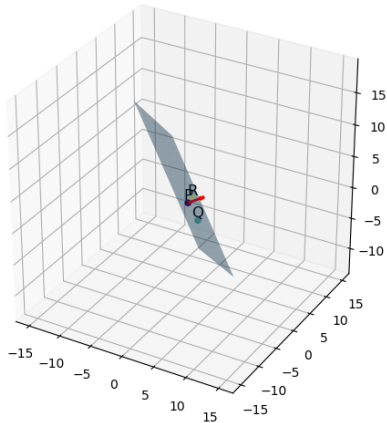
$$x_1 = 2x_3 \quad (12)$$

$$\mathbf{x} = x_3 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad (13)$$

The unit vector perpendicular to the plane is given by

$$\mathbf{x} = \frac{1}{\sqrt{6}} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad (14)$$

# Plot



# C Code - unit normal vector for plane

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

int main() {
    // Coordinates of the points
    double P[3] = {1, -1, 2};
    double Q[3] = {2, 0, -1};
    double R[3] = {0, 2, 1};
```



## C Code - unit normal vector for plane

```
    / Store points to a file (no labels, just coordinates)
FILE *fptr;
fptr = fopen("output.data", "w");
if (fptr == NULL) {
    printf("Error opening file!\n");
    return 1;
}
fprintf(fptr, "%lf %lf %lf\n", P[0], P[1], P[2]);
fprintf(fptr, "%lf %lf %lf\n", Q[0], Q[1], Q[2]);
fprintf(fptr, "%lf %lf %lf\n", R[0], R[1], R[2]);
fclose(fptr);
```

## C Code - unit normal vector for plane

```
/
// Compute vectors PQ and PR
double PQ[3], PR[3];
for (int i = 0; i < 3; i++) {
    PQ[i] = Q[i] - P[i];
    PR[i] = R[i] - P[i];
}

// Cross product PQ x PR
double N[3];
N[0] = PQ[1]*PR[2] - PQ[2]*PR[1];
N[1] = PQ[2]*PR[0] - PQ[0]*PR[2];
N[2] = PQ[0]*PR[1] - PQ[1]*PR[0];
```

## C Code - unit normal vector for plane

```
// Magnitude of the vector N
double mag = sqrt(N[0]*N[0] + N[1]*N[1] + N[2]*N[2]);

// Unit vector
double unit[3];
for (int i = 0; i < 3; i++) {
    unit[i] = N[i] / mag;
}

printf("Unit vector perpendicular to the plane: (%lf, %lf, %lf)\n", unit[0], unit[1], unit[2]);
return 0;
}
```

# Python Code

```
import sys #for path to external 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

#if using termux
import subprocess
import shlex
#end if
```

```
# Read points from output.data file
points = []
with open('output.data', 'r') as file:
    for line in file:
        coords = list(map(float, line.split()))
        points.append(coords)

# Convert to numpy arrays with required shape
P = np.array(points[0]).reshape(-1, 1)
Q = np.array(points[1]).reshape(-1, 1)
R = np.array(points[2]).reshape(-1, 1)
```

```
# Create a figure and a 3D Axes
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

a, b, c, d = 2, 1, 1, -3 # coefficients of the plane equation: ax
    + by + cz + d = 0

# 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)

# 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(Q,R)

#Plotting all lines
#ax.plot(x_QR[0,:],x_QR[1,:], x_QR[2,:],label='$BC$')

# Scatter plot
colors = np.arange(2, 5) # Example colors
tri_coords = np.block([P,Q,R]) # Stack P,Q,R vertically
ax.scatter(tri_coords[0, :], tri_coords[1, :], tri_coords[2, :],
           c=colors)
vert_labels = ['P', 'Q', 'R']
ax.quiver(1,-1,2, 2,1,1, color='r', linewidth=2, label='n')
```

```
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}', fontsize=12, ha='center', va='bottom')
    #ax.text(tri_coords[0, i], tri_coords[1, i], tri_coords[2, i],
            f'{txt}\n({tri_coords[0, i]:.0f}, {tri_coords[1, i]:.0f}, {tri_coords[2, i]:.0f})',
            # fontsize=12, ha='center', va='bottom')

ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
```



# Python Code

```
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])
'''
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
'''
plt.grid() # minor
plt.axis('equal')

plt.savefig('../figs/fig.png')
```