# 4.8.6

Dhanush Kumar A - AI25BTECH11010

September 29, 2025

# Question

Find the coordinates of the foot of the perpendicular **Q** drawn from $P(3,2,1)$ to the plane $2x - y + z + 1 = 0$. Also find the distance **PQ** and the image of the point **P** treating this plane as a mirror.

# Solution

The point and the plane normal are

$$\mathbf{P} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, \qquad \mathbf{n} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}, \tag{1}$$

$$\mathbf{n}^T \mathbf{x} = -1. \tag{2}$$

Let $\mathbf{Q}$ be the foot of the perpendicular from $\mathbf{P}$ to the plane and let $\lambda$ be the scalar such that

$$\mathbf{P} - \mathbf{Q} = \lambda \mathbf{n} \tag{3}$$

$$\mathbf{n}^T \mathbf{Q} = -1. \tag{4}$$

we have $\mathbf{Q} = \mathbf{P} - \lambda \mathbf{n}$. Therefore,

$$\mathbf{n}^T (\mathbf{P} - \lambda \mathbf{n}) = -1 \tag{5}$$

$$\mathbf{n}^T \mathbf{P} - \lambda \|\mathbf{n}\|^2 = -1. \tag{6}$$

# Solution

Thus

$$5 - 6\lambda = -1 \quad \Rightarrow \quad \lambda = -1. \tag{7}$$

Therefore

$$\mathbf{Q} = \mathbf{P} - \lambda \mathbf{n} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} - (-1) \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix}. \tag{8}$$

Distance:

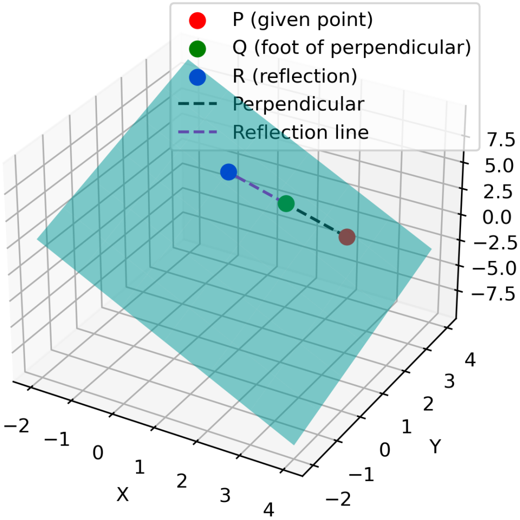
$$PQ = \|\mathbf{P} - \mathbf{Q}\| = \|\lambda \mathbf{n}\| = |\lambda| \|\mathbf{n}\| = 1 \cdot \sqrt{6} = \sqrt{6}. \tag{9}$$

Image of **P** in the plane (reflection) is

$$\mathbf{R} = 2\mathbf{Q} - \mathbf{P} = 2 \begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 4 \\ -1 \end{pmatrix}. \tag{10}$$

**Answer:** $\mathbf{Q} = (1, 3, 0),\ PQ = \sqrt{6},\ \mathbf{R} = (-1, 4, -1).$

# Plot


● P (given point)
● Q (foot of perpendicular)
● R (reflection)
Perpendicular
Reflection line
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−2
−1
0
1
2
3
4
X
−2
−1
0
1
2
3
4
Y

# Python code - To find n

```python



import numpy as np
import matplotlib.pyplot as plt
import os
from mpl_toolkits.mplot3d import Axes3D

# Function to compute foot of perpendicular, distance, and reflection
def solve_point_plane(P, n, d):
    P = np.array(P, dtype=float)
    n = np.array(n, dtype=float)

    #  = (n^T P - d) / (n^T n)
    lam = (np.dot(n, P) - d) / np.dot(n, n)

    # Foot of perpendicular
    Q = P - lam * n
```

# Python code - To find n

```
    # Distance
    dist = abs(lam) * np.linalg.norm(n)

    # Reflection
    R = 2 * Q - P

    return lam, Q, dist, R


# ---------------- MAIN CODE ----------------
P = np.array([3, 2, 1])
n = np.array([2, -1, 1])
d = -1 # Plane equation: n^T x = -1

# Solve
lam, Q, dist, R = solve_point_plane(P, n, d)
```

# Python code - To find n

```

print(" =", lam)
print("Foot of perpendicular Q =", Q)
print("Distance =", dist)
print("Reflection R =", R)
```

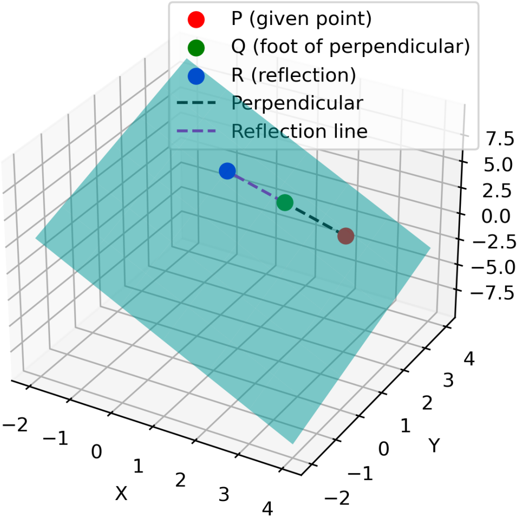
# Python code - Plotting the Plane

```
# ---------- Plotting ----------
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Create a grid for the plane
xx, yy = np.meshgrid(range(-2, 5), range(-2, 5))
zz = (-d - n[0]*xx - n[1]*yy) / n[2]

# Plot the plane surface
ax.plot_surface(xx, yy, zz, alpha=0.5, color='cyan')

# Plot points P, Q, R
ax.scatter(*P, color='red', s=60, label='P (given point)')
ax.scatter(*Q, color='green', s=60, label='Q (foot of
    perpendicular)')
ax.scatter(*R, color='blue', s=60, label='R (reflection)')
```

# Python code - Plotting the Plane

```python

# Draw perpendicular line P-Q
ax.plot([P[0], Q[0]], [P[1], Q[1]], [P[2], Q[2]], 'k--', label='
    Perpendicular')

# Draw line Q-R
ax.plot([Q[0], R[0]], [Q[1], R[1]], [Q[2], R[2]], 'm--', label='
    Reflection line')

# Labels
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.legend()

# ---------- Save figure ----------
os.makedirs("figs", exist_ok=True) # create folder if not exists
save_path = os.path.join("../figs", "point_plane.png")
plt.savefig(save_path, dpi=300, bbox_inches='tight')
```

# Plot-Using Python



# C code - Solving

```c

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "/home/dhanush-kumar-a/ee1030-2025/ai25btech11010/matgeo/4.8.6/codes/libs/matfun.h"

// Function to compute distance and reflection
// Inputs: P (3x1), n (3x1), d
// Outputs: distance (pointer), R (3x1 array)
void point_plane_info(double **P, double **n, double d, double *distance, double *R) {
    // Compute lambda
    double lam = (Matdot(n,P,3) - d)/(Matdot(n,n,3));

    // Compute foot of perpendicular Q = P - lambda*n
    double **Q = Matsub(P, Matscale(n,3,1,lam), 3,1);
```

# C code - Solving

```c

    // Compute distance = |lambda| * ||n||
    *distance = fabs(lam) * Matnorm(n,3);

    // Compute reflection R = 2*Q - P
    double **Rmat = Matsub(Matscale(Q,3,1,2.0), P, 3,1);

    // Copy results to output array
    for(int i=0;i<3;i++)
        R[i] = Rmat[i][0];

    // Free memory
    for(int i=0;i<3;i++){
        free(Q[i]);
        free(Rmat[i]);
    }
    free(Q);
    free(Rmat);
}
```

# Python code -Ploting the plane using c function

```python
import ctypes
import numpy as np
import matplotlib.pyplot as plt
import os


# Load the shared library (use absolute path if needed)
lib = ctypes.CDLL("./main.so") # or absolute path

# Prepare point P and normal n
P_vals = [3.0, 2.0, 1.0]
n_vals = [2.0, -1.0, 1.0]

# Allocate pointers for P and n (double**)
P = (ctypes.POINTER(ctypes.c_double) * 3)()
n = (ctypes.POINTER(ctypes.c_double) * 3)()
for i in range(3):
    P[i] = ctypes.pointer(ctypes.c_double(P_vals[i]))
```

# Python code - Plotting the Plane

```

# Prepare outputs
distance = ctypes.c_double()
R = (ctypes.c_double * 3)()

# Call the C function
lib.point_plane_info(
    P, # double**
    n, # double**
    ctypes.c_double(-1.0), # plane constant d
    ctypes.byref(distance), # double* output
    R # double* output
)

# Convert R to numpy array
R_vals = np.array([R[i] for i in range(3)])
P_arr = np.array(P_vals)
n_arr = np.array(n_vals)
d = -1.0
```

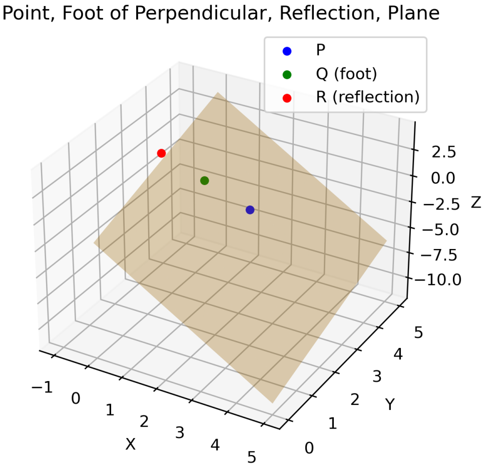
# Python code - Plotting the Plane

```python

# Compute Q = foot of perpendicular
lam = (np.dot(n_arr, P_arr) - d) / np.dot(n_arr, n_arr)
Q_arr = P_arr - lam * n_arr

# Print results
print("Distance from P to plane:", distance.value)
print("Foot of perpendicular Q:", Q_arr)
print("Reflection R:", R_vals)

# --- Plotting ---
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

# Python code - Plotting the Plane

```python

# Plot points P, Q, R
ax.scatter(*P_arr, color='blue', label='P')
ax.scatter(*Q_arr, color='green', label='Q (foot)')
ax.scatter(*R_vals, color='red', label='R (reflection)')

# Plot plane
xx, yy = np.meshgrid(np.linspace(0,5,10), np.linspace(0,5,10))
zz = (-n_arr[0]*xx - n_arr[1]*yy + d)/n_arr[2]
ax.plot_surface(xx, yy, zz, alpha=0.3, color='orange')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
ax.set_title("Point, Foot of Perpendicular, Reflection, Plane")

# Save figure
plt.savefig("../figs/point_plane_plot.png", dpi=300)
```

# Plot-Using Python and C


Point, Foot of Perpendicular, Reflection, Plane
● P
● Q (foot)
● R (reflection)
X
Y
Z