12.547

Bhargav - EE25BTECH11013

October 5, 2025

Question

Question:
Consider $\mathbf{R^3}$ with the usual inner product. If d is the distance from (1,1,1) to the subspace span $\{(1,1,0),(0,1,1)\}$ of $\mathbf{R^3}$, then $3d^2$ is

Solution

Let $\mathbf{W} = \text{span}\{u_1, u_2\}$
Where $\mathbf{U} = \begin{pmatrix} u_1 & u_2 \end{pmatrix}$

The distance from $\mathbf{P} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ to the subspace span $\mathbf{W}$ can be found by finding the projection of $\mathbf{P}$ onto $\mathbf{W}$.

Let $\mathbf{Ux}$ be the projection of $\mathbf{P}$ on the span $\mathbf{W}$
Where $\mathbf{x}$ is the column vector containing the coefficients that scale the basis vectors of the subspace to give the projection point.

Let $\mathbf{Ux}$ be the projection of $\mathbf{P}$ on the span $\mathbf{W}$

$$\mathbf{U}^{\mathbf{T}}(\mathbf{P} - \mathbf{Ux}) = 0 \tag{1}$$

(since $\mathbf{U}$ is perpendicular to $\mathbf{P} - \mathbf{Ux}$)

$$\implies \mathbf{U}^{\mathbf{T}}\mathbf{Ux} = \mathbf{U}^{\mathbf{T}}\mathbf{P} \tag{2}$$

Solution

Since the columns of $\mathbf{U}$ are Linearly independent, so are the columns of $\mathbf{U^T U}$ and hence $\mathbf{U^T U}$ is invertible

$$\mathbf{x} = \left(\mathbf{U^T U}\right)^{-1} \mathbf{U^T P} \tag{3}$$

Hence the projection of $\mathbf{P}$ on the span $\mathbf{W}$ is

$$\mathbf{Ux} = \mathbf{U}\left(\mathbf{U^T U}\right)^{-1} \mathbf{U^T P} \tag{4}$$

Solution

The distance of $\mathbf{P}$ from the span $\mathbf{W}$ is:

$$d = \|\mathbf{P} - \mathbf{U}\mathbf{x}\| \tag{5}$$

$$d = \left\|\mathbf{P} - \mathbf{U}\left(\mathbf{U}^\mathsf{T}\mathbf{U}\right)^{-1}\mathbf{U}^\mathsf{T}\mathbf{P}\right\| \tag{6}$$

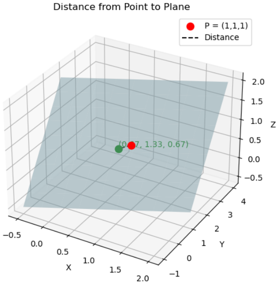
$$\mathbf{P} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{U} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{7}$$

Substituting the values in (0.6):

$$d = \frac{1}{\sqrt{3}} \tag{8}$$

$$\boxed{3d^2 = 1}$$

Plot


Distance from Point to Plane
P = (1,1,1)
Distance
(0.67, 1.33, 0.67)
X
Y
Z
−0.5
0.0
0.5
1.0
1.5
2.0
−1
0
1
2
3
4
−0.5
0.0
0.5
1.0
1.5
2.0

C Code

```c
#include <math.h>

double projection(const double *u1, const double *u2, const
    double *P, double *P_proj) {
    double U_TU[2][2], invU[2][2], UTP[2], coeff[2];

    // Compute Gram matrix U^T U
    U_TU[0][0] = u1[0]*u1[0] + u1[1]*u1[1] + u1[2]*u1[2];
    U_TU[0][1] = u1[0]*u2[0] + u1[1]*u2[1] + u1[2]*u2[2];
    U_TU[1][0] = U_TU[0][1];
    U_TU[1][1] = u2[0]*u2[0] + u2[1]*u2[1] + u2[2]*u2[2];

    double det = U_TU[0][0]*U_TU[1][1] - U_TU[0][1]*U_TU[1][0];
    invU[0][0] = U_TU[1][1]/det;
    invU[0][1] = -U_TU[0][1]/det;
    invU[1][0] = -U_TU[1][0]/det;
    invU[1][1] = U_TU[0][0]/det;
```

C Code

```c
    // Compute U^T P
    UTP[0] = u1[0]*P[0] + u1[1]*P[1] + u1[2]*P[2];
    UTP[1] = u2[0]*P[0] + u2[1]*P[1] + u2[2]*P[2];

    // Compute coefficients
    coeff[0] = invU[0][0]*UTP[0] + invU[0][1]*UTP[1];
    coeff[1] = invU[1][0]*UTP[0] + invU[1][1]*UTP[1];

    for(int i=0;i<3;i++)
        P_proj[i] = coeff[0]*u1[i] + coeff[1]*u2[i];

    double dist = 0.0;
    for(int i=0;i<3;i++)
        dist += (P[i] - P_proj[i]) * (P[i] - P_proj[i]);
    return sqrt(dist);
}
```

Python + C Code

```python
import ctypes
import numpy as np
import matplotlib.pyplot as plt
lib = ctypes.CDLL("./libdist.so")
lib.projection.argtypes = [
    np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags="C_CONTIGUOUS"),
    np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags="C_CONTIGUOUS"),
    np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags="C_CONTIGUOUS"),
    np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags="C_CONTIGUOUS")
]
lib.projection.restype = ctypes.c_double
```

Python + C Code

```
u1 = np.array([1.0, 1.0, 0.0])
u2 = np.array([0.0, 1.0, 1.0])
P = np.array([1.0, 1.0, 1.0])
P_proj = np.zeros(3, dtype=np.double)
# Compute projection and distance
distance = lib.projection(u1, u2, P, P_proj)
print(f"Distance from P to plane: {distance:.6f}")
print(f"Projection point: {P_proj}")

s = np.linspace(-0.5, 2, 10)
t = np.linspace(-0.5, 2, 10)
S, T = np.meshgrid(s, t)
X = S*u1[0] + T*u2[0]
Y = S*u1[1] + T*u2[1]
Z = S*u1[2] + T*u2[2]

fig = plt.figure(figsize=(7,6))
ax = fig.add_subplot(111, projection='3d')
```

Python + C Code

```
ax.plot_surface(X, Y, Z, color='lightblue', alpha=0.5)
ax.scatter(*P, color='red', s=80, label='P = (1,1,1)')
ax.scatter(*P_proj, color='green', s=80)
ax.text(P_proj[0], P_proj[1], P_proj[2],
        f'({P_proj[0]:.2f}, {P_proj[1]:.2f}, {P_proj[2]:.2f})',
        color='green', fontsize=10, ha='left', va='bottom')
ax.plot([P[0], P_proj[0]], [P[1], P_proj[1]], [P[2], P_proj[2]],
    'k--', label='Distance')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Distance from Point to Plane')
ax.legend()
plt.savefig("/mnt/c/Users/bharg/Documents/backupmatrix/
    ee25btech11013/matgeo/12.547/figs/Figure_1.png")
plt.show()
```

Python Code

```
import numpy as np
import matplotlib.pyplot as plt

# Define vectors and point
u1 = np.array([1.0, 1.0, 0.0])
u2 = np.array([0.0, 1.0, 1.0])
P = np.array([1.0, 1.0, 1.0])

# Stack u1 and u2 as columns to form U
U = np.column_stack((u1, u2))

# Compute projection coefficients: inv(U^T U) * U^T * P
coeff = np.linalg.inv(U.T @ U) @ (U.T @ P)

# Compute projection point
P_proj = U @ coeff
```

Python Code

```
# Compute distance
distance = np.linalg.norm(P - P_proj)
print(f"Distance from P to plane: {distance:.6f}")
print(f"Projection point: {P_proj}")

# Create plane grid
s = np.linspace(-0.5, 2, 10)
t = np.linspace(-0.5, 2, 10)
S, T = np.meshgrid(s, t)
X = S*u1[0] + T*u2[0]
Y = S*u1[1] + T*u2[1]
Z = S*u1[2] + T*u2[2]

# Plotting
fig = plt.figure(figsize=(7,6))
ax = fig.add_subplot(111, projection='3d')

# Plane
ax.plot_surface(X, Y, Z, color='lightblue', alpha=0.5)
```

Python Code

```python
# Original point
ax.scatter(*P, color='red', s=80, label='P = (1,1,1)')
ax.scatter(*P_proj, color='green', s=80)
ax.text(P_proj[0], P_proj[1], P_proj[2],
        f'({P_proj[0]:.2f}, {P_proj[1]:.2f}, {P_proj[2]:.2f})',
        color='green', fontsize=10, ha='left', va='bottom')
ax.plot([P[0], P_proj[0]], [P[1], P_proj[1]], [P[2], P_proj[2]],
    'k--', label='Distance')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Distance from Point to Plane')
ax.legend()
plt.savefig("/mnt/c/Users/bharg/Documents/backupmatrix/
    ee25btech11013/matgeo/12.547/figs/Figure_1.png")
plt.show()
```