2.5.25

Vaishnavi - EE25BTECH11059

September 29, 2025

Question

Let $\mathbb{R}^3$ denote the three-dimensional space. Take two points $P = (1, 2, 3)$ and $Q = (4, 2, 7)$. Let $\text{dist}(X, Y)$ denote the distance between two points X and Y in $\mathbb{R}^3$.
Let

$$S = \{X \in \mathbb{R}^3 : (\text{dist}(X, P))^2 - (\text{dist}(X, Q))^2 = 50\}$$

and

$$T = \{Y \in \mathbb{R}^3 : (\text{dist}(Y, Q))^2 - (\text{dist}(Y, P))^2 = 50\}.$$

Then which of the following statements are TRUE?

Solution

Variable	Value
P	$(1, 2, 3)$
Q	$(4, 2, 7)$

Table: Variables Used

Solution

$$\mathbf{P} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \tag{1}$$

$$\mathbf{Q} = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix} \tag{2}$$

$$\text{dist}(X, P))^2 - (\text{dist}(X, Q))^2 = 50 \tag{3}$$

$$(\|\mathbf{X} - \mathbf{P}\|_2)^2 - (\|\mathbf{X} - \mathbf{Q}\|_2)^2 = 50 \tag{4}$$

solution

$$\mathbf{X}^{\mathbf{T}}\mathbf{X} - 2\mathbf{P}^{\mathbf{T}}\mathbf{X} + \mathbf{P}^{\mathbf{T}}\mathbf{P} - \mathbf{X}^{\mathbf{T}}\mathbf{X} + 2\mathbf{Q}^{\mathbf{T}}\mathbf{X} - \mathbf{Q}^{\mathbf{T}}\mathbf{Q} = 50 \tag{5}$$

$$2(\mathbf{Q}^{\mathbf{T}} - \mathbf{P}^{\mathbf{T}})\mathbf{X} + \mathbf{P}^{\mathbf{T}}\mathbf{P} - \mathbf{Q}^{\mathbf{T}}\mathbf{Q} = 50 \tag{6}$$

(0.6)is the eq of plane S

$$\text{dist}(Y, Q))^2 - (\text{dist}(Y, P))^2 = 50 \tag{7}$$

$$(\|\mathbf{Y} - \mathbf{Q}\|_2)^2 - (\|\mathbf{Y} - \mathbf{P}\|_2)^2 = 50 \tag{8}$$

Solution

$$\mathbf{Y}^\mathbf{T}\mathbf{Y} - 2\mathbf{Q}^\mathbf{T}\mathbf{Y} + \mathbf{Q}^\mathbf{T}\mathbf{Q} - \mathbf{Y}^\mathbf{T}\mathbf{Y} + 2\mathbf{P}^\mathbf{T}\mathbf{Y} - \mathbf{P}^\mathbf{T}\mathbf{P} = 50 \tag{9}$$

$$2(\mathbf{P}^\mathbf{T} - \mathbf{Q}^\mathbf{T})\mathbf{Y} + \mathbf{Q}^\mathbf{T}\mathbf{Q} - \mathbf{P}^\mathbf{T}\mathbf{P} = 50 \tag{10}$$

(0.10) is the eq of plane T
S And T are parallel planes

Solution

S is an infinite plane. In any plane one can choose three non collinear points whose triangle has unit area.Therefore a) is correct

let eq of T be $\mathbf{n^T X} = c$
let $\mathbf{A}, \mathbf{B}$ be two points on T

$$\mathbf{n^T A} = c \tag{11}$$
$$\mathbf{n^T B} = c \tag{12}$$

Solution

any point on line AB can be written as

$$\mathbf{C} = (1-k)\mathbf{A} + k\mathbf{B} \tag{13}$$

$$\mathbf{n}^\mathbf{T}[(1-k)\mathbf{A} + k\mathbf{B}] = \mathbf{n}^\mathbf{T}\mathbf{A} + k[\mathbf{n}^\mathbf{T}\mathbf{B} - \mathbf{n}^\mathbf{T}\mathbf{A}] = c \tag{14}$$

$$\mathbf{n}^\mathbf{T}\mathbf{C} = c \tag{15}$$

Hence C lies on T
Therefore b) is correct

Solution

distance between S and T$=\frac{|c_1-c_2|}{|n|}=10$
let $\mathbf{A},\mathbf{B}$ lie on S and $\mathbf{A}',\mathbf{B}'$ where

$$\|\mathbf{A}-\mathbf{B}\|_2=\|\mathbf{A}'-\mathbf{B}'\|_2=d(say) \tag{16}$$
$$2(d+10)=48 \tag{17}$$
$$d=14 \tag{18}$$

There are infinitely many ways to pick two points in plane S that are 14 units apart (because the plane is infinite).
Therefore c) is correct

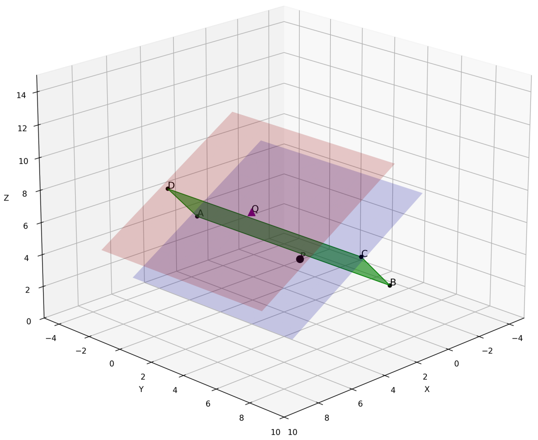
Solution

The distance between the planes S and T is 10. A square with perimeter 48 has side length 12. If two opposite vertices lie on S and the other two on T, then each side of the square must connect the two planes. Since the vertical separation is 10, and the side is 12, the square can be tilted so that part of the side length is vertical (10 units) and the rest is horizontal: ($\sqrt{12^2 - 10^2} = \sqrt{44}$ units)
Hence d) is correct

Graph

Refer to Figure

Option (d): Planes S & T with Square of Perimeter 48



Python Code

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

# Define vectors
a = np.array([2, -1, -2])
b = np.array([7, 2, -3])
b1 = np.array([4, -2, -4])
b2 = np.array([3, 4, 1])

# Function to draw vectors
def draw_vector(ax, start, vec, color, label):
    ax.quiver(*start, *vec, color=color, label=label,
        arrow_length_ratio=0.1)

# Create 3D plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection='3d')
```

Python Code

```
# Draw from origin
origin = np.array([0,0,0])
draw_vector(ax, origin, a, 'blue', 'a')
draw_vector(ax, origin, b, 'red', 'b')
draw_vector(ax, origin, b1, 'green', 'b1 (parallel to a)')
draw_vector(ax, origin, b2, 'purple', 'b2 (perpendicular to a)')

# Show b as b1 + b2 (parallelogram completion)
draw_vector(ax, b1, b2, 'orange', 'b1 + b2 = b')

# Labels and title
ax.set_xlabel('X', fontsize=12)
ax.set_ylabel('Y', fontsize=12)
ax.set_zlabel('Z', fontsize=12)
ax.set_title( 3D Representation of Vectors a, b, b1, and b2 , fontsize=14)
ax.legend()
```

Python Code

```
# Grid and aspect ratio
ax.grid(True)
ax.set_box_aspect([1,1,1])

# Axis limits
ax.set_xlim(0,8)
ax.set_ylim(-3,6)
ax.set_zlim(-5,2)

# Save figure
plt.savefig( Graph3.png , dpi=300, bbox_inches='tight')
plt.show()
```

C Code

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

#define MAX_POINTS 10000
#define STEP 1.0
#define TOL 0.5  // tolerance
#define SIDE_LENGTH 12.0

typedef struct {
    double x, y, z;
} Point;

double dist2(Point a, Point b) {
    return (a.x - b.x)*(a.x - b.x) +
           (a.y - b.y)*(a.y - b.y) +
           (a.z - b.z)*(a.z - b.z);
}
```

C Code

```
void solve_vectors() {
  Point P = {1, 2, 3};
  Point Q = {4, 2, 7};

  Point S[MAX_POINTS];
  Point T[MAX_POINTS];
  int s_count = 0, t_count = 0;

  for (double x = -10; x <= 10; x += STEP) {
      for (double y = -10; y <= 10; y += STEP) {
          for (double z = -10; z <= 10; z += STEP) {
              Point A = {x, y, z};
              double lhs = dist2(A, P);
              double rhs = dist2(A, Q);
              double d = lhs - rhs;
```

```
if (fabs(d - 50.0) < TOL && s_count < MAX_POINTS) {
                    S[s_count++] = A;
                } else if (fabs(-d - 50.0) < TOL &&
                   t_count < MAX_POINTS) {
                    T[t_count++] = A;
                }
            }
        }
    }

    printf( Found %d points on Set S and %d points on
        Set T.\n , s_count, t_count);

    for (int i = 0; i < s_count; i++) {
        for (int j = 0; j < t_count; j++) {
            double d = sqrt(dist2(S[i], T[j]));
            if (fabs(d - SIDE_LENGTH) < TOL) {
                printf( Found square side length %.2f\
                   n , d);
```

Python and C Code

```
iimport ctypes

# Load the shared library
lib = ctypes.CDLL('./code.so')

# Call the function
lib.solve_vectors()
```