2.10.75

RAVULA SHASHANK REDDY - EE25BTECH11047

September 15, 2025

Question

Prove the points with position vectors $\mathbf{a} + \mathbf{b}$, $\mathbf{a} - \mathbf{b}$ and $\mathbf{a} + k\mathbf{b}$ are collinear for all real values of k.

Theoretical Solution

Given:

$$\mathbf{P} = \begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{Q} = \begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \mathbf{R} = \begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix} \begin{pmatrix} 1 \\ k \end{pmatrix} \tag{1}$$

$$\mathbf{P} - \mathbf{Q} = \begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \tag{2}$$

$$\mathbf{R} - \mathbf{P} = \begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix} \begin{pmatrix} 0 \\ k-1 \end{pmatrix} \tag{3}$$

$$M = \begin{pmatrix} \mathbf{P} - \mathbf{Q} & \mathbf{R} - \mathbf{P} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 2\mathbf{b} & (k-1)\mathbf{b} \end{pmatrix} \tag{4}$$

$$M = \mathbf{b} \begin{pmatrix} 0 & 0 \\ 2 & k-1 \end{pmatrix} \tag{5}$$

$$\text{rank}(M) \leq 1. \tag{6}$$

Therefore, the two difference vectors are linearly dependent.

Hence, the points $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ are collinear for all real k.

For Example:

$$\mathbf{a} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}. \tag{7}$$

For $k = 0$:

$$\mathbf{R} = \begin{pmatrix} 1 + 3(0) \\ 2 + 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}. \tag{8}$$

For $k = 1$:

$$\mathbf{R} = \begin{pmatrix} 1 + 3(1) \\ 2 + 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}. \tag{9}$$

For $k = 2$:

$$\mathbf{R} = \begin{pmatrix} 1 + 3(2) \\ 2 + 2 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}. \tag{10}$$

So the three points are:

$$\mathbf{Q} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}, \mathbf{P} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \tag{11}$$

$$\mathbf{R} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \ (k=0), \ \begin{pmatrix} 4 \\ 3 \end{pmatrix} \ (k=1), \ \begin{pmatrix} 7 \\ 4 \end{pmatrix} \ (k=2). \tag{12}$$

$$M(k) = \begin{pmatrix} \mathbf{P}-\mathbf{Q} & \mathbf{R}-\mathbf{P} \end{pmatrix} = \begin{pmatrix} 6 & 3k-3 \\ 2 & k-1 \end{pmatrix}. \tag{13}$$

For $k = 0$:

$$M(0) = \begin{pmatrix} 6 & -3 \\ 2 & -1 \end{pmatrix}, \quad \operatorname{rank}(M(0)) = 1. \tag{14}$$

Theoretical Solution

For $k = 1$:

$$M(1) = \begin{pmatrix} 6 & 0 \\ 2 & 0 \end{pmatrix}, \quad \text{rank}(M(1)) = 1. \tag{15}$$

For $k = 2$:

$$M(2) = \begin{pmatrix} 6 & 3 \\ 2 & 1 \end{pmatrix}, \quad \text{rank}(M(2)) = 1. \tag{16}$$

C Code

```c
#include <stdio.h>

// Function to compute rank of an n x 2 matrix
// Since columns are multiples, rank is at most 1
int rankMatrix(int n, int col1[], int col2[]) {
    int i;
    // check if both columns are zero
    int zero1 = 1, zero2 = 1;
    for(i=0; i<n; i++) {
        if(col1[i] != 0) zero1 = 0;
        if(col2[i] != 0) zero2 = 0;
    }
    if(zero1 && zero2) return 0; // zero matrix
    // check if col2 is multiple of col1
    int ratio_num = 0, ratio_den = 0;
```

C Code

```
    for(i=0; i<n; i++) {
        if(col1[i] != 0) {
            ratio_num = col2[i];
            ratio_den = col1[i];
            break;
        }
    }
    int dep = 1;
    for(i=0; i<n; i++) {
        if(col1[i]*ratio_num != col2[i]*ratio_den) {
            dep = 0;
            break;
        }
    }
    if(dep) return 1; // dependent columns
    return 2; // independent (wont happen here)
}
```

C Code

```
int main() {
    int n, k, i;

    printf(Enter dimension n: );
    scanf(%d, &n);

    int a[n], b[n];
    printf(Enter vector a (%d values): , n);
    for(i=0; i<n; i++) scanf(%d, &a[i]);

    printf(Enter vector b (%d values): , n);
    for(i=0; i<n; i++) scanf(%d, &b[i]);

    printf(Enter value of k: );
    scanf(%d, &k);

    int col1[n], col2[n];
```

C Code

```
for(i=0; i<n; i++) {
    col1[i] = 2*b[i]; // P - Q
    col2[i] = (k-1)*b[i]; // R - P
}

printf(Matrix M = [col1 | col2]\n);
for(i=0; i<n; i++) {
    printf([%d %d]\n, col1[i], col2[i]);
}
```

C Code

```
    int r = rankMatrix(n, col1, col2);
    printf(Rank(M) = %d\n, r);

    if(r <= 1) {
        printf(=> Points are collinear\n);
    } else {
        printf(=> Points are not collinear\n);
    }

    return 0;
}
```

```
import numpy as np
import matplotlib.pyplot as plt
from libs.rank import rank
from libs.line import line_dir_pt # your definition

# Base vectors
a = np.array([1, 2], dtype=float)
b = np.array([3, 1], dtype=float)

# Fixed points
P = a + b
Q = a - b

# Points for k = 0,1,2
R_points = {fR{k}: a + k*b for k in [0, 1, 2]}

# --- Print ranks and matrices ---
def M_matrix(a, b, k):
    col1 = 2*b
```

Python Direct

```
    col2 = (k-1)*b
    return np.column_stack((col1, col2))

print(P =, P, Q =, Q)
for k, R in R_points.items():
    M = M_matrix(a, b, int(k[1]))
    print(f\n{k}: {R})
    print(M =\n, M)
    print(rank =, rank(M))

# --- Plotting ---
plt.figure(figsize=(7,6))
ax = plt.gca()
ax.set_aspect(equal)

def to_tuple(arr):
    return tuple(int(x) for x in arr)
```

```
# Plot P and Q
ax.scatter(P[0], P[1], color=blue, s=100, zorder=3)
ax.text(P[0]+0.3, P[1]+0.3, fP {to_tuple(P)}, fontsize=12)

ax.scatter(Q[0], Q[1], color=blue, s=100, zorder=3)
ax.text(Q[0]+0.3, Q[1]+0.3, fQ {to_tuple(Q)}, fontsize=12)

# Plot R0, R1, R2
for label, pt in R_points.items():
    ax.scatter(pt[0], pt[1], color=red, s=100, zorder=3)
    ax.text(pt[0]+0.3, pt[1]+0.3, f{label} {to_tuple(pt)},
        fontsize=12)

# Line through P and Q
dir_vec = (P - Q).reshape(-1,1)
P_col = P.reshape(-1,1)
line_points = line_dir_pt(dir_vec, P_col, -10, 10) # smaller
    range
ax.plot(line_points[0,:], line_points[1,:], k--, linewidth=1.8)
```

Python Direct

```
# Zoom into region around points
all_points = np.array([P, Q] + list(R_points.values()))
xmin, ymin = np.min(all_points, axis=0) - 2
xmax, ymax = np.max(all_points, axis=0) + 2
ax.set_xlim(xmin, xmax)
ax.set_ylim(ymin, ymax)

# Labels & grid
ax.set_xlabel(x-axis, fontsize=13)
ax.set_ylabel(y-axis, fontsize=13)
ax.set_title(Collinearity of P, Q, R(k=0,1,2), fontsize=14,
    weight=bold)
ax.grid(True, linestyle=--, alpha=0.7)

plt.show()
```

Python Shared Output

```python
    import ctypes
import numpy as np

# load library
lib = ctypes.CDLL(./librank.so)

# function signature
lib.compute_rank.argtypes = [
    ctypes.c_int,
    ctypes.POINTER(ctypes.c_int),
    ctypes.POINTER(ctypes.c_int),
    ctypes.c_int,
    ctypes.POINTER(ctypes.c_int),
    ctypes.POINTER(ctypes.c_int),
]
lib.compute_rank.restype = ctypes.c_int
```

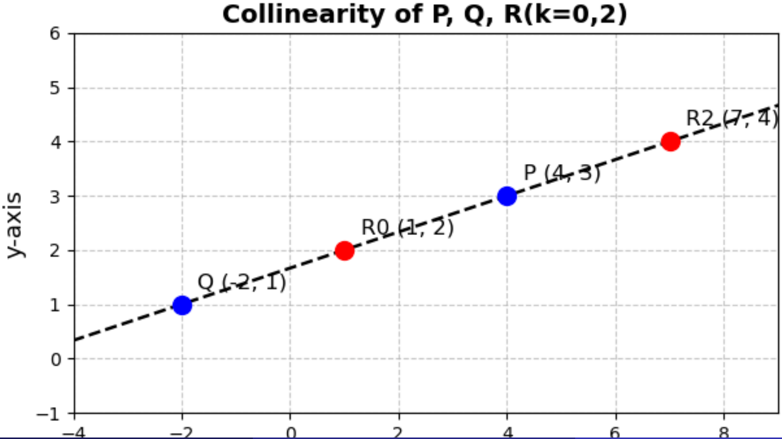
Python Shared Output

```python
def compute_rank(a, b, k):
    n = len(a)
    a = np.array(a, dtype=np.int32)
    b = np.array(b, dtype=np.int32)
    col1 = np.zeros(n, dtype=np.int32)
    col2 = np.zeros(n, dtype=np.int32)

    r = lib.compute_rank(
        n,
        a.ctypes.data_as(ctypes.POINTER(ctypes.c_int)),
        b.ctypes.data_as(ctypes.POINTER(ctypes.c_int)),
        k,
        col1.ctypes.data_as(ctypes.POINTER(ctypes.c_int)),
        col2.ctypes.data_as(ctypes.POINTER(ctypes.c_int)),
    )
```

Python Shared Output

```
    return col1.tolist(), col2.tolist(), r

if __name__ == __main__:
    a = [1, 2]
    b = [3, 1]
    for k in [0, 1, 2]:
        col1, col2, r = compute_rank(a, b, k)
        print(f\nFor k={k}:)
        for i in range(len(a)):
            print(f[{col1[i]:3d} {col2[i]:3d}])
        print(Rank =, r)
        if r <= 1:
            print(=> Collinear)
        else:
            print(=> Not collinear)
```

Plot



Collinearity of P, Q, R(k=0,2)
y-axis
R2 (7, 4)
P (4, 3)
R0 (1, 2)
Q (-2, 1)
6
5
4
3
2
1
0
−1
−4
−2
0
2
4
6
8