1.6.18

AI25BTECH11003 - Bhavesh Gaikwad

August 26,2025

Question

Prove that points A(2,1), B(0,5) and C(-1,2) are not collinear.

Theoretical Solution

$$\text{B-A}=\begin{pmatrix}0-2\\5-1\end{pmatrix}=\begin{pmatrix}-2\\4\end{pmatrix} \qquad C-A=\begin{pmatrix}-1-2\\2-1\end{pmatrix}=\begin{pmatrix}-3\\1\end{pmatrix}$$

$$\text{M}=\begin{pmatrix}B-A & C-A\end{pmatrix}=\begin{pmatrix}-2 & -3\\4 & 1\end{pmatrix}$$

Row-reduce to compute the rank:

$$\begin{pmatrix}-2 & -3\\4 & 1\end{pmatrix}\xrightarrow{R_2\leftarrow R_2+2R_1}\begin{pmatrix}-2 & -3\\0 & -5\end{pmatrix}$$

The echelon form has two nonzero rows, hence rank(M)=2≠1

$$\boxed{\textit{Therefore, The points } A(2,1),\ B(0,5) \textit{ and } C(-1,2) \textit{ are not collinear.}} \tag{1}$$

Python + C Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "libs/matfun.h"
#include "libs/geofun.h"

int main(void) {
    // Points as 2x1 column vectors
    double **A = createMat(2,1);
    double **B = createMat(2,1);
    double **C = createMat(2,1);

    // Set coordinates correctly
    A[0][0] = 2.0; A[1][0] = 1.0;
    B[0][0] = 0.0; B[1][0] = 5.0;
    C[0][0] = -1.0; C[1][0] = 2.0;

    // Calculate direction vectors B-A and C-A
    double **BA = Matsub(B, A, 2, 1);
```

Python + C Code

```c
// Calculate determinant to find rank
    double det = M[0][0] * M[1][1] - M[0][1] * M[1][0];

    int rank;
    if (fabs(det) > 1e-10) { // Use fabs for absolute value
        rank = 2;
    } else {
        // Check if at least one row is non-zero
        if ((fabs(M[0][0]) > 1e-10 || fabs(M[0][1]) > 1e-10) ||
            (fabs(M[1][0]) > 1e-10 || fabs(M[1][1]) > 1e-10)) {
            rank = 1;
        } else {
            rank = 0;
        }
    }
```

Python + C Code

```
// Output result
    if (rank == 2) {
        printf("The points A, B, and C are NOT collinear (Rank =
            2)\n");
    } else if (rank == 1) {
        printf("The points A, B, and C are collinear (Rank = 1)\n
            ");
    } else {
        printf("All points coincide (Rank = 0)\n");
    }

    // Write points to .dat file
    FILE *f = fopen("points.dat", "w");
    if (f == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
```

Python + C Code

```
    fprintf(f, "%.6f\t%.6f\n", A[0][0], A[1][0]);
    fprintf(f, "%.6f\t%.6f\n", B[0][0], B[1][0]);
    fprintf(f, "%.6f\t%.6f\n", C[0][0], C[1][0]);
    fclose(f);

    printf("Points written to points.dat\n");

    // Free memory
    freeMat(A, 2);
    freeMat(B, 2);
    freeMat(C, 2);
    freeMat(BA, 2);
    freeMat(CA, 2);
    freeMat(M, 2);

    return 0;
}
```

Python + C Code

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

# Read the points from points.dat
points = []
with open("points.dat") as f:
    for line in f:
        x, y = map(float, line.split())
        points.append((x, y))

# Labels and colors for each point
labels = ['A(2,1)', 'B(0,5)', 'C(-1,2)']
colors = ['blue', 'orange', 'green']

plt.figure(figsize=(6, 6))
```

Python + C Code

```
# Plot points and annotate
for (x, y), label, color in zip(points, labels, colors):
    plt.scatter(x, y, color=color, s=100, zorder=5)
    plt.text(x + 0.1, y + 0.1, label, fontsize=12, fontweight='
        bold')

# Draw dashed lines between B to A and C to A
# Assuming order in points.dat is A, B, C
A, B, C = points
plt.plot([B[0], A[0]], [B[1], A[1]], '--', color='gray', alpha
    =0.7, linewidth=1.5)
plt.plot([C[0], A[0]], [C[1], A[1]], '--', color='gray', alpha
    =0.7, linewidth=1.5)
```

Python + C Code

```
# Grid, axes, and ticks
plt.grid(True, alpha=0.4, linestyle=':')
plt.axhline(0, color='black', linewidth=1)
plt.axvline(0, color='black', linewidth=1)
plt.xlim(-2, 4)
plt.ylim(0, 7)
plt.xticks(np.arange(-2, 5, 1))
plt.yticks(np.arange(0, 8, 1))
# Title only
plt.title('Points A, B, C', fontsize=14, fontweight='bold', pad
    =20)
# Remove axis labels
plt.xlabel('')
plt.ylabel('')
# Save the figure
plt.savefig('fig1.png', dpi=150, bbox_inches='tight')
plt.close()
```

Python Code

```
# Code by GVV Sharma
# September 12, 2023
# Revised July 21, 2024
# released under GNU GPL

# Point Vectors
import sys # for path to external scripts
sys.path.insert(0, '/home/bhavesh-g/Documents/matgeo/1.6.18/codes
    /line') # path to my scripts

import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Python Code

```
# local imports
from line.funcs import *
from triangle.funcs import *
from conics.funcs import circ_gen

# --- Using given points directly ---
A = np.array(([2, 1])).reshape(-1, 1) # A(2,1)
B = np.array(([0, 5])).reshape(-1, 1) # B(0,5)
C = np.array(([-1, 2])).reshape(-1, 1) # C(-1,2)

# Generating all lines
x_AB = line_gen(A, B)
x_BC = line_gen(B, C)
x_CA = line_gen(C, A)
```

Python Code

```
# Plotting all lines
plt.plot(x_AB[0, :], x_AB[1, :], label='$AB$')
plt.plot(x_BC[0, :], x_BC[1, :], label='$BC$')
plt.plot(x_CA[0, :], x_CA[1, :], label='$CA$')

# Labeling the coordinates
colors = np.arange(1, 4)
tri_coords = np.block([[A, B, C]])
plt.scatter(tri_coords[0, :], tri_coords[1, :], c=colors)
vert_labels = ['A', 'B', 'C']
for i, txt in enumerate(vert_labels):
    plt.annotate(
        f'{txt}\n({tri_coords[0,i]:.2f}, {tri_coords[1,i]:.2f})',
        (tri_coords[0, i], tri_coords[1, i]),
        textcoords="offset points",
        xytext=(25, 5),
        ha='center'
    )
```

Python Code

```
# Axes crossing at origin
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['bottom'].set_position('zero')

plt.grid()
plt.axis('equal')

# if using termux
plt.savefig('fig1.png')
# subprocess.run(shlex.split("termux-open fig1.pdf")) # uncomment
    on Termux
# else
# plt.show()
```

Graph