2.10.77

Sai Krishna Bakki - EE25BTECH11049

Question

The points $(-a, -b)$, $(0, 0)$, (a, b) and (a^2, ab) are

1. Collinear
2. Vertices of a parallelogram
3. Vertices of a rectangle
4. None of these

Checking For The Points To Be Vertices Of A Parallelogram

$$\mathbf{A} = \begin{pmatrix} -a \\ -b \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{C} = \begin{pmatrix} a \\ b \end{pmatrix}, \mathbf{D} = \begin{pmatrix} a^2 \\ ab \end{pmatrix} \tag{1}$$

Condition for the points to be vertices of a parallelogram is

$$\mathbf{B} - \mathbf{A} = \mathbf{C} - \mathbf{D} \tag{2}$$

$$\mathbf{B} - \mathbf{A} = \begin{pmatrix} a \\ b \end{pmatrix}, \mathbf{C} - \mathbf{D} = \begin{pmatrix} a - a^2 \\ b - ab \end{pmatrix} \tag{3}$$

But

$$\mathbf{B} - \mathbf{A} \neq \mathbf{C} - \mathbf{D} \tag{4}$$

If $\mathbf{B} - \mathbf{A} \neq \mathbf{C} - \mathbf{D}$ then the points cannot be vertices of a rectangle too because every rectangle is a specific type of parallelogram.

Checking For The Points To Be Collinear

Condition for the points to be collinear is

$$\text{rank}\begin{pmatrix}\mathbf{B}-\mathbf{A} & \mathbf{C}-\mathbf{D}\end{pmatrix} = 1 \tag{5}$$

$$\tag{6}$$

$$\text{rank}\begin{pmatrix} a & a-a^2 \\ b & b-ab \end{pmatrix} \tag{7}$$

By transformation $R_2 \to \frac{-b}{a}R_2 + 2R_1$

$$\text{rank}\begin{pmatrix} a & a-a^2 \\ 0 & 0 \end{pmatrix} = 1 \tag{8}$$

The number of non zero rows in the row reduced matrix (also known as *echelon form*) is defined as the rank.
For the above matrix, Rank is one.
Therefore, we can conclude that four points are collinear.

C Code

```
#include <stdio.h>
#ifdef __cplusplus
extern C {
#endif

    double check_collinearity(double x1, double y1, double x2,
        double y2, double x3, double y3) {
       // Calculate the determinant
       double determinant = x1 * (y2 - y3) + x2 * (y3 - y1) + x3
            * (y1 - y2);
       return determinant;
    }

#ifdef __cplusplus
}
#endif
```

Python Code through shared output

```
# Code by GVV Sharma
# September 12, 2023
# released under GNU GPL
# This script checks if the points (-a,-b), (0,0), (a,b), and (a
    ^2, ab) are collinear.

import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import subprocess
import shlex
import ctypes
import os

# local imports (assuming funcs.py is in the same directory)
from libs.funcs import *

# --- COMPILE AND LOAD C FUNCTION ---
# This block compiles line.c into a shared library and loads it.
```

Python Code through shared output

```
c_file = line.c
lib_file = line.so if os.name != 'nt' else line.dll

# Compile C code into a shared library
# The -fPIC flag is needed for creating a shared library
compile_command = fgcc -shared -o {lib_file} -fPIC {c_file}
try:
    subprocess.run(shlex.split(compile_command), check=True)
    print(fSuccessfully compiled {c_file} to {lib_file}\n)
except (subprocess.CalledProcessError, FileNotFoundError):
    print(Error: C compilation failed. Make sure gcc is installed
         and in your PATH.)
    exit()

# Load the shared library
try:
    c_lib = ctypes.CDLL(os.path.abspath(lib_file))
```

Python Code through shared output

```
except OSError as e:
    print(fError loading shared library: {e})
    exit()

# Define the C function's signature (argument types and return
    type)
check_collinearity_c = c_lib.check_collinearity
check_collinearity_c.argtypes = [ctypes.c_double] * 6 # six
    double arguments
check_collinearity_c.restype = ctypes.c_double # returns a double

# --- PROBLEM SETUP ---

# For a tangible example, let's choose non-zero values for a and
    b
a = 2.0
b = 3.0
```

Python Code through shared output

```
# Define the four points using numpy arrays
P1 = np.array([-a, -b]).reshape(-1, 1)
P2 = np.array([0.0, 0.0]).reshape(-1, 1)
P3 = np.array([a, b]).reshape(-1, 1)
P4 = np.array([a**2, a*b]).reshape(-1, 1)

# --- METHOD 1: MATRIX RANK COLLINEARITY CHECK (Original Method)
    ---
print(--- Method 1: NumPy Matrix Rank ---)
# Form a matrix with the vectors as columns.
vec_matrix = np.hstack([P1, P3, P4])
rank = LA.matrix_rank(vec_matrix)

print(fChecking points with a={a}, b={b})
print(fMatrix of vectors (from origin to other points):\n{
    vec_matrix})
print(fRank of the matrix: {rank})
```

Python Code through shared output

```
if rank == 1:
    print(Conclusion: A rank of 1 means the points are COLLINEAR.
        )
else:
    print(Conclusion: The points are NOT collinear. )

print(- * 40)

# --- METHOD 2: C FUNCTION COLLINEARITY CHECK (New Method) ---
print(--- Method 2: Calling C function via ctypes ---)
# To check if 4 points are collinear, we can check 2 sets of 3
    points.
# For example, are P1, P2, P3 collinear? And are P1, P2, P4
    collinear?

# Check collinearity for points P1, P2, and P3
det1 = check_collinearity_c(
```

Python Code through shared output

```
    P1[0,0], P1[1,0], # P1(x, y)
    P2[0,0], P2[1,0], # P2(x, y)
    P3[0,0], P3[1,0] # P3(x, y)
)

# Check collinearity for points P1, P2, and P4
det2 = check_collinearity_c(
    P1[0,0], P1[1,0], # P1(x, y)
    P2[0,0], P2[1,0], # P2(x, y)
    P4[0,0], P4[1,0] # P4(x, y)
)

print(fDeterminant for P1, P2, P3: {det1})
print(fDeterminant for P1, P2, P4: {det2})

# For floating point numbers, check if the determinant is very
    close to zero
if abs(det1) < 1e-9 and abs(det2) < 1e-9:
```

```
    print(\nConclusion: Both determinants are zero, so the points
        are COLLINEAR.)
else:
    print(\nConclusion: At least one determinant is non-zero, so
        the points are NOT collinear.)

# --- PLOTTING SECTION (No changes needed here) ---
print(\nGenerating plot...)
x_line = line_gen(P1, P4)
plt.plot(x_line[0,:], x_line[1,:], label='Line of Collinearity')
all_coords = np.hstack([P1, P2, P3, P4])
plt.scatter(all_coords[0,:], all_coords[1,:])
vert_labels = ['P1(-a, -b)', 'P2(0, 0)', 'P3(a, b)', 'P4(a, ab)']
for i, txt in enumerate(vert_labels):
    plt.annotate(f'{txt}\n({all_coords[0,i]:.1f}, {all_coords[1,i
        ]:.1f})',
                 (all_coords[0,i], all_coords[1,i]),
                 textcoords=offset points, xytext=(0,10), ha='
                     center')
```

```python
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
plt.legend(loc='best')
plt.grid()
plt.axis('equal')
plt.show()

# Clean up the compiled library file
if os.path.exists(lib_file):
    os.remove(lib_file)
```

Python Code

```
# Code by GVV Sharma
# September 12, 2023
# Revised September 28, 2025
# released under GNU GPL
# This script checks if the points (-a,-b), (0,0), (a,b), and (a
    ^2, ab) are collinear.

import sys
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt

# local imports
from libs.funcs import *

# if using termux
import subprocess
import shlex
# end if
```

```
# For a tangible example, let's choose non-zero values for a and
    b
a = 2
b = 3

# Define the four points based on the problem statement
P1 = np.array([-a, -b]).reshape(-1, 1)
P2 = np.array([0, 0]).reshape(-1, 1)
P3 = np.array([a, b]).reshape(-1, 1)
P4 = np.array([a**2, a*b]).reshape(-1, 1)

# To check if all points are collinear, we can check the rank of
    a matrix
# formed by the vectors from the origin (P2) to the other points.
# The vectors are P1-P2 (i.e., P1), P3-P2 (i.e., P3), and P4-P2 (
    i.e., P4).
# If all these vectors lie on the same line, the rank of the
    matrix will be 1.
```

Python Code

```
# Form a matrix with the vectors as columns
# Note: P2 is the origin, so it's not needed to form the vectors.
vec_matrix = np.block([P1, P3, P4])

# Calculate and print the rank
rank = LA.matrix_rank(vec_matrix)
print(fThe points are defined with a={a} and b={b})
print(fMatrix of vectors:\n{vec_matrix})
print(fRank of the matrix: {rank})

if rank == 1:
    print(Conclusion: A rank of 1 means the vectors are linearly dependent, so the points are COLLINEAR. )
else:
    print(Conclusion: The points are NOT collinear. )


# --- Plotting Section ---
```

Python Code

```
# Generate a line passing through the points to visualize.
# We can use the two outer points, P1 and P4, to draw the line.
x_line = line_gen(P1, P4)

# Plot the generated line
plt.plot(x_line[0,:], x_line[1,:], label='Line of Collinearity')

# Combine all points into one array for plotting
all_coords = np.block([[P1, P2, P3, P4]])
plt.scatter(all_coords[0,:], all_coords[1,:])

# Label the coordinates
vert_labels = ['P1(-a, -b)', 'P2(0, 0)', 'P3(a, b)', 'P4(a, ab)']
for i, txt in enumerate(vert_labels):
    plt.annotate(f'{txt}\n({all_coords[0,i]:.0f}, {all_coords[1,i]:.0f})',
 (all_coords[0,i], all_coords[1,i]), # Point to label
```

Python Code

```
 textcoords=offset points, # How to position the text
 xytext=(0,10), # Distance from text to points (x,y)
    ha='center') # Horizontal alignment

# use set_position
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
plt.legend(loc='best')
plt.grid()
plt.axis('equal')
plt.show()
```

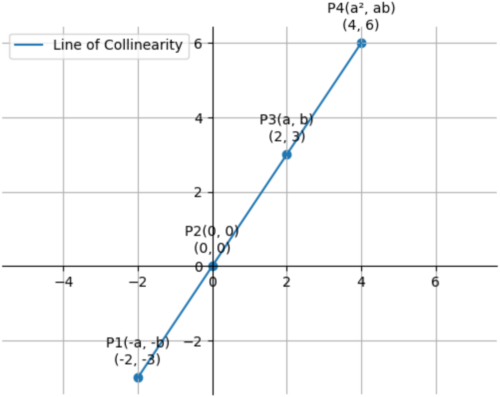
Line of Collinearity
P4(a², ab)
(4, 6)
P3(a, b)
(2, 3)
P2(0, 0)
(0, 0)
P1(-a, -b)
(-2, -3)
−4
−2
0
2
4
6
−2
2
4
6

Figure: 1