1.5.24

M Chanakya Srinivas- EE25BTECH11036

Question

Question:A line intersects the Y-axis and X-axis at the points $P = (0, b)$ and $Q = (c, 0)$ respectively. If $(2, -5)$ is the midpoint of $\overline{PQ}$, then find the coordinates of P and Q.

Problem Statement

A line intersects the Y-axis and X-axis at points

$$P = (0, b), \quad Q = (c, 0).$$

If $(2, -5)$ is the midpoint of $\overline{PQ}$, then find P and Q using a matrix method.

Step 1: Representing the Points as Vectors

$$\mathbf{P} = \begin{pmatrix} 0 \\ b \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} c \\ 0 \end{pmatrix},$$

$$\mathbf{M} = \begin{pmatrix} 2 \\ -5 \end{pmatrix}.$$

Midpoint Formula: $\mathbf{M} = \frac{1}{2}(\mathbf{P} + \mathbf{Q})$.

Step 2: Applying the Midpoint Formula

$$\begin{pmatrix} 2 \\ -5 \end{pmatrix} = \frac{1}{2}\left(\begin{pmatrix} 0 \\ b \end{pmatrix} + \begin{pmatrix} c \\ 0 \end{pmatrix}\right)$$

$$= \frac{1}{2}\begin{pmatrix} c \\ b \end{pmatrix}$$

$$\implies \begin{pmatrix} 4 \\ -10 \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}.$$

Step 3: Writing as a Matrix System

$$c = 4, \quad b = -10.$$

$$\underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} b \\ c \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} -10 \\ 4 \end{pmatrix}}_{\mathbf{B}}.$$

Step 4: Solving the Matrix Equation

$$\mathbf{x} = A^{-1}\mathbf{B}$$

$$= I\begin{pmatrix} -10 \\ 4 \end{pmatrix}$$

$$= \begin{pmatrix} -10 \\ 4 \end{pmatrix}.$$

$$\therefore \quad b = -10, \quad c = 4.$$

Final Answer

$$P = \begin{pmatrix} 0 \\ -10 \end{pmatrix},$$

$$Q = \begin{pmatrix} 4 \\ 0 \end{pmatrix}.$$

$$\boxed{P = (0, -10),\ Q = (4, 0)}$$

C Code - Section formula function

```c
// File: solver.c
void findCoordinates(int mx, int my, int* c_ptr, int* b_ptr) {
    // Calculate c from the x-coordinate of the midpoint
    *c_ptr = 2 * mx;

    // Calculate b from the y-coordinate of the midpoint
    *b_ptr = 2 * my;
}
```

Python Code through shared output

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

# --- Part 1: Interfacing with the C Function ---

# Load the shared library
try:
    solver_lib = ctypes.CDLL('./1.5.24.so')
except OSError as e:
    print(Error loading shared library. Did you compile solver.c?
        )
    print(e)
    exit()

# Define the argument and return types for the C function
# void findCoordinates(int, int, int*, int*)
```

Python Code through shared output

```
solver_lib.findCoordinates.argtypes = [
    ctypes.c_int,
    ctypes.c_int,
    ctypes.POINTER(ctypes.c_int),
    ctypes.POINTER(ctypes.c_int)
]
solver_lib.findCoordinates.restype = None
```

Python Code through shared output

```
# Input: The midpoint coordinates
midpoint_x, midpoint_y = 2, -5

# Create C-type integer variables to store the output from the C
    function
c_val = ctypes.c_int()
b_val = ctypes.c_int()

# Call the C function, passing the addresses of our output
    variables
solver_lib.findCoordinates(
    ctypes.c_int(midpoint_x),
    ctypes.c_int(midpoint_y),
    ctypes.byref(c_val),
    ctypes.byref(b_val)
)
```

Python Code through shared output

```
# Extract the Python values from the C-type objects
c = c_val.value
b = b_val.value

# Define the coordinates of P and Q
P = (0, b)
Q = (c, 0)
M = (midpoint_x, midpoint_y)

print(fCalculation complete.)
print(fCoordinates of P = {P})
print(fCoordinates of Q = {Q})
```

Python Code through shared output

```
# --- Part 2: Plotting the Result ---

# Create arrays for plotting the line
x_points = np.array([P[0], Q[0]])
y_points = np.array([P[1], Q[1]])

# Create the plot
plt.figure(figsize=(8, 7))
plt.plot(x_points, y_points, 'b-', label=f'Line through P and Q')
     # Line
plt.plot(P[0], P[1], 'go', markersize=10, label=f'P = {P}') #
    Point P
plt.plot(Q[0], Q[1], 'ro', markersize=10, label=f'Q = {Q}') #
    Point Q
plt.plot(M[0], M[1], 'm*', markersize=12, label=f'Midpoint = {M}'
    )# Midpoint M
```

Python Code through shared output

```
# Annotate points with their coordinates
plt.text(P[0] + 0.2, P[1], f'P{P}', fontsize=12)
plt.text(Q[0] + 0.2, Q[1], f'Q{Q}', fontsize=12)
plt.text(M[0] + 0.2, M[1], f'M{M}', fontsize=12)

# Formatting the plot
plt.title('Line Intersecting X and Y Axes', fontsize=16)
plt.xlabel('X-axis', fontsize=12)
plt.ylabel('Y-axis', fontsize=12)
plt.axhline(0, color='black', linewidth=0.7) # X-axis
plt.axvline(0, color='black', linewidth=0.7) # Y-axis
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.axis('equal') # Ensure the scale is the same on both axes

# Show the plot
plt.show()
```

Direct Python Code

```
import sys # for path to external scripts
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

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

# if using termux
import subprocess
import shlex
# end if

# Midpoint given
M = np.array(([2, -5])).reshape(-1,1)
```

Direct Python Code

```
# Let P = (0,b), Q = (c,0)
# From midpoint formula: (c/2, b/2) = (2, -5)
c = 4
b = -10

# Coordinates
P = np.array(([0,b])).reshape(-1,1)
Q = np.array(([c,0])).reshape(-1,1)

# Generating line PQ
x_PQ = line_gen(P,Q)

# Plotting the line
plt.plot(x_PQ[0,:], x_PQ[1,:], label='$PQ$')

# Plot midpoint
plt.scatter(M[0,:], M[1,:], color='red', label='Midpoint M')
```

Direct Python Code

```
# Labeling the coordinates
coords = np.block([[P,Q,M]])
vert_labels = ['P','Q','M']
plt.scatter(coords[0,:], coords[1,:])
for i, txt in enumerate(vert_labels):
    plt.annotate(f'{txt}\n({coords[0,i]:.0f}, {coords[1,i]:.0f})'
        ,
                 (coords[0,i], coords[1,i]),
                 textcoords=offset points,
                 xytext=(20,-10),
                 ha='center')

# Axis styling
ax = plt.gca()
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)
```

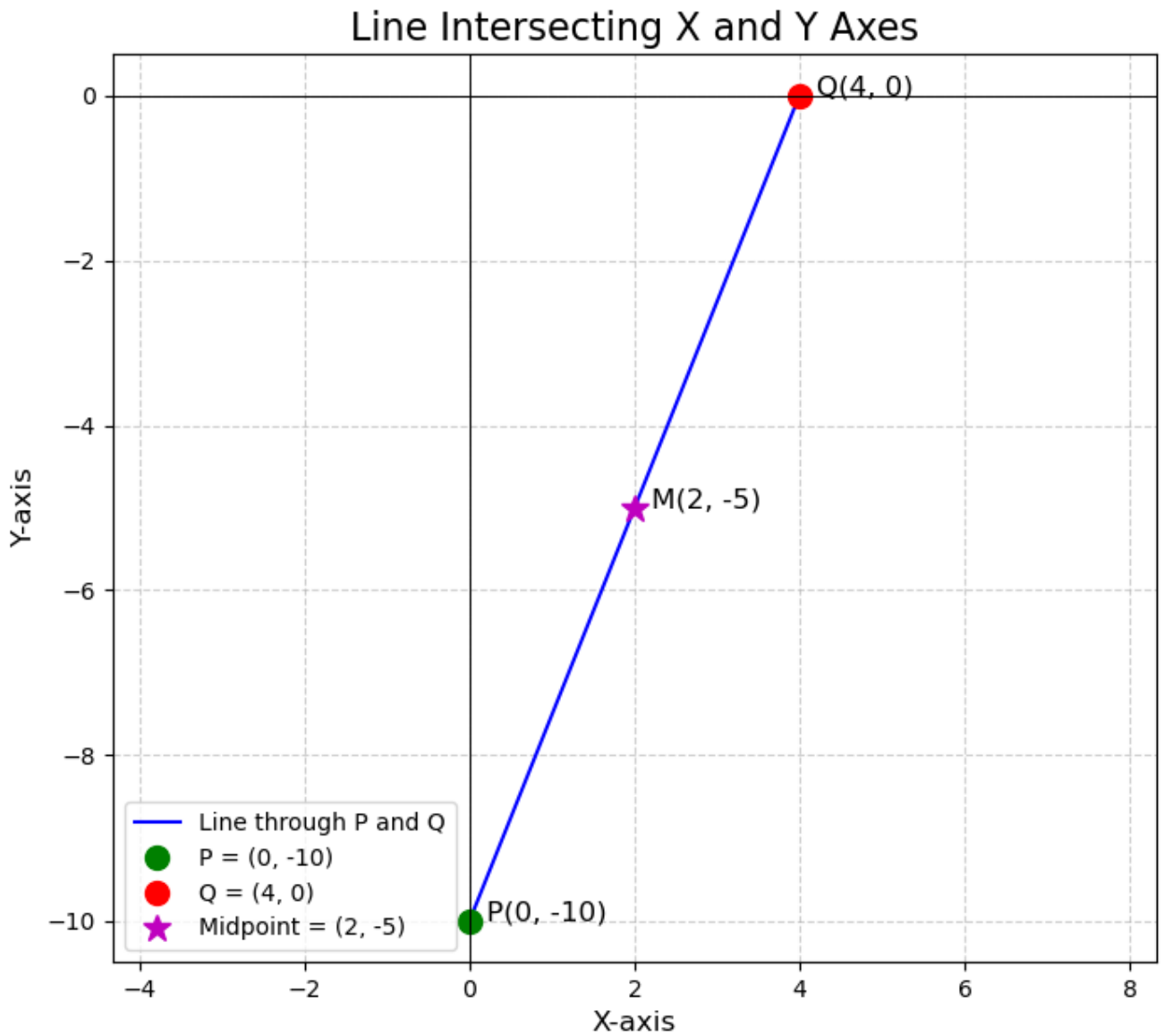
Direct Python Code

```
plt.legend(loc='best')
plt.grid()
plt.axis('equal')

outfile = 'chapters/10/7/2/2/figs/fig.pdf'
plt.savefig(outfile)

# Save figure as PNG
outfile = 'chapters/10/7/2/2/figs/fig.png'
plt.savefig(outfile, dpi=300)

# Open image depending on system
try:
    import platform
    import subprocess, shlex

    if termux in platform.platform().lower(): # Android Termux
        subprocess.run(shlex.split(ftermux-open {outfile}))
```

Direct Python Code

```
    else: # Linux desktop
        subprocess.run(shlex.split(fxdg-open {outfile}))
except Exception as e:
    print(fCould not auto-open file. Saved at {outfile})

#else
#plt.show()
```

Plot by python using shared output from c



Plot by python only