MatGeo Presentation

EE24BTECH11044 - Muthyala Koushik

Question

AOBC is a rectangle whose three vertices are vertices $\mathbf{A}\left(0,3\right)$, $\mathbf{O}\left(0,0\right)$ and $\mathbf{B}\left(5,0\right)$. The length of its diagonal is

Solution:

Direction vector of $\mathbf{AB}$: $\mathbf{m} = \mathbf{B} - \mathbf{A}$

$$\mathbf{AB} = \begin{pmatrix} 5 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ -3 \end{pmatrix} \tag{1}$$

Solution:

length of $\mathbf{AB}$(Diagonal):$\left\|\mathbf{m}\right\|^2 = \mathbf{m}\mathbf{m}^t$

$$\left\|\mathbf{AB}\right\|^2 = \begin{pmatrix} 5 & -3 \end{pmatrix} \begin{pmatrix} 5 \\ -3 \end{pmatrix} \tag{2}$$

$$\left\|\mathbf{AB}\right\|^2 = 5^2 + (-3)^2 \tag{3}$$

$$\left\|\mathbf{AB}\right\|^2 = 25 + 9 \tag{4}$$

$$\left\|\mathbf{AB}\right\|^2 = 34 \tag{5}$$

$$\left\|\mathbf{AB}\right\| = \sqrt{34} \tag{6}$$

so, length of diagonal$=\sqrt{34}$.

Plot

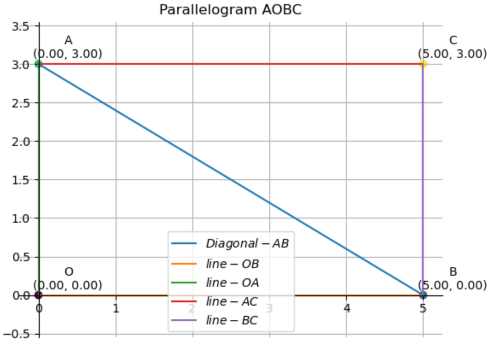

Figure: The plot of the rectangle AOBC

C-Code

```c
#include <stdio.h>

// Function to calculate the C vertex
void find_c_vertex(double O[2], double A[2], double B[2], double
↪  C[2]) {
    C[0] = A[0] + B[0] - O[0];
    C[1] = A[1] + B[1] - O[1];
}}
//use this command gcc -shared -fPIC -o vertex.so vertex.c
// to generate vertex.so lib
```

Python Code

```python
import sys                                          #for path to external scripts
sys.path.insert(0, '/home/koushik/matgeo/codes/CoordGeo')        #path to my scripts
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

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

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

# Helper function to convert NumPy arrays to C arrays
def to_c_array(np_array):
    return (ctypes.c_double * 2)(*np_array)

# Define the Python function to call the C function
def calculate_c_vertex(O, A, B):
    C = (ctypes.c_double * 2)()  # Create an empty C array for the output (vertex C)

    # Convert NumPy arrays to C arrays using the helper function
    O_c = to_c_array(O)
    A_c = to_c_array(A)
    B_c = to_c_array(B)

    # Call the C function to calculate the vertex C
    lib.find_c_vertex(O_c, A_c, B_c, C)
```

Python Code

```
 # Return the C vertex as a Python list
    return [C[0], C[1]]

# Generating points O, A, B
O = np.array([0, 0]).reshape(-1, 1)
B = np.array([5, 0]).reshape(-1, 1)
A = np.array([0, 3]).reshape(-1, 1)

# Call the function to calculate the fourth vertex C
C = np.array(calculate_c_vertex(O.flatten(), A.flatten(), B.flatten())).reshape(-1, 1)

# Generating lines for the parallelogram
from line.funcs import line_gen

x_OB = line_gen(O, B)
x_OA = line_gen(O, A)
x_BC = line_gen(B, C)
x_AC = line_gen(A, C)
x_AB = line_gen(A, B)

# Plotting the lines
plt.plot(x_AB[0,:], x_AB[1,:], label='$Diagonal-AB$')
plt.plot(x_OB[0,:], x_OB[1,:], label='$line-OB$')
plt.plot(x_OA[0,:], x_OA[1,:], label='$line-OA$')
plt.plot(x_AC[0,:], x_AC[1,:], label='$line-AC$')
plt.plot(x_BC[0,:], x_BC[1,:], label='$line-BC$')
```

Python Code

```python
# Labeling the coordinates
colors = np.arange(1, 5)
tri_coords = np.block([[O, B, A, C]])
plt.scatter(tri_coords[0,:], tri_coords[1,:], c=colors)
vert_labels = ['O', 'B', 'A', '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')

# Adjust plot aesthetics
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.grid()
plt.axis('equal')
plt.title('Parallelogram AOBC')
plt.legend()
plt.show()
```