MatGeo Question 3-3.3-2

Shiven Bajpai

AI24BTECH11030
IIT Hyderabad

November 5, 2024

Problem Statement

Construct a triangle with sides $5cm$, $6cm$ and $7cm$

Calculating Cosine

Let the vertices of triangle be **A**, **B** and **C** and lengths of the sides opposing them be denoted by $a = 5cm$, $b = 6cm$ and $c = 7cm$ respectively.
By Cosine rule in ΔABC,

$$a^2 = b^2 + c^2 - 2bc\cos A \tag{3.1}$$

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc} \tag{3.2}$$

$$\cos A = \frac{60}{84} \tag{3.3}$$

Solving for points

Let

$$\mathbf{A} = \mathbf{0} \text{ and } \mathbf{C} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Then

$$\mathbf{B} = c \begin{pmatrix} \cos A \\ \sin A \end{pmatrix}$$

Substituting values from 3.3 we get,

$$\mathbf{A} = \mathbf{0},\ \mathbf{B} = \begin{pmatrix} 5 \\ \sqrt{24} \end{pmatrix},\ \mathbf{C} = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$$

Code I

This code can be found at

https://github.com/shivenBajpai/EE1030/blob/main/Question_6/Codes/main.c
https://github.com/shivenBajpai/EE1030/blob/main/Question_6/Codes/main.py

```
// main.c

#include <math.h>
#include <stdlib.h>

// Struct to represent 2 dimensional vectors
typedef struct {
    double x;
    double y;
} Vector;

// Calculate vertcies given the lengths of a triangle
Vector* calculateVertices(double a, double b, double c) {
    Vector* vertices = malloc(sizeof(Vector) * 3);
```

Code II

```
    double cos_A = (pow(b,2) + pow(c,2) - pow(a,2)) / (2*b*c);
    double sin_A = sqrt(1 - pow(cos_A, 2));

    vertices[0].x = 0;
    vertices[0].y = 0;
    vertices[1].x = c*cos_A;
    vertices[1].y = c*sin_A;
    vertices[2].x = b;
    vertices[2].y = 0;

    return vertices;
}
```

Code III

```
# main.py

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

lib = ctypes.CDLL('./main.so')

omat = np.array([[0, 1], [-1, 0]])

# Utility functions
# Copied from line library from github.com/gadepall/matgeo
def line_gen(A,B):
  len =10
  dim = A.shape[0]
  x_AB = np.zeros((dim,len))
```

Code IV

```
  lam_1 = np.linspace(0,1,len)
  for i in range(len):
    temp1 = A + lam_1[i]*(B-A)
    x_AB[:,i]= temp1.T
  return x_AB

def circ_gen(O,r):
    len = 50
    theta = np.linspace(0,2*np.pi,len)
    x_circ = np.zeros((2,len))
    x_circ[0,:] = r*np.cos(theta)
    x_circ[1,:] = r*np.sin(theta)
    x_circ = (x_circ + O)
    return x_circ

def line_intersect(n1,A1,n2,A2):
  N=np.block([n1,n2]).T
  p = np.zeros((2,1))
```

Code V

```
  p[0] = n1.T@A1
  p[1] = n2.T@A2
  P=LA.solve(N,p)
  return P

# Define the Vector struct in Python
class Vector(ctypes.Structure):
    _fields_ = [("x", ctypes.c_double),
                ("y", ctypes.c_double)]

# Specify the return type of the get_data function
lib.calculateVertices.restype = ctypes.POINTER(Vector)
lib.calculateVertices.argtypes = [ctypes.c_double, ctypes.
    c_double, ctypes.c_double]

# Call the C function
a = 5
b = 6
```

Code VI

```
c = 7
v_ptr = lib.calculateVertices(a, b, c)

# Extract information from returned values
A = np.array([v_ptr[0].x, v_ptr[0].y]).reshape(-1,1)
B = np.array([v_ptr[1].x, v_ptr[1].y]).reshape(-1,1)
C = np.array([v_ptr[2].x, v_ptr[2].y]).reshape(-1,1)

# Generate and plot lines
x_AB = line_gen(A,B);
x_BC = line_gen(B,C);
x_CA = line_gen(C,A);
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='$CA$')

# Circumcircle
O = line_intersect(A-B,(A+B)/2,A-C,(A+C)/2)
```

Code VII

```
r = np.sqrt((A-O).T@(A-O))
ccircle = circ_gen(O, r)
plt.plot(ccircle[0,:], ccircle[1,:], label="Circumcircle")

# Incircle
I = line_intersect(omat@(B-A)/LA.norm(B-A)+omat@(C-A)/LA.norm(C-A
    ),A,omat@(A-B)/LA.norm(A-B) + omat@(C-B)/LA.norm(C-B),B)
r = (omat@(B-C)).T@(I-B)/LA.norm(B-C)
icircle = circ_gen(I, r)
plt.plot(icircle[0,:], icircle[1,:], label="Incircle")

#Labeling the coordinates
tri_coords = np.block([[A,B,C,O,I]])
plt.scatter(tri_coords[0,:], tri_coords[1,:])
vert_labels = ['A','B','C','O','I']
for i, txt in enumerate(vert_labels):
    plt.annotate(txt,
                 (tri_coords[0,i], tri_coords[1,i]),
```

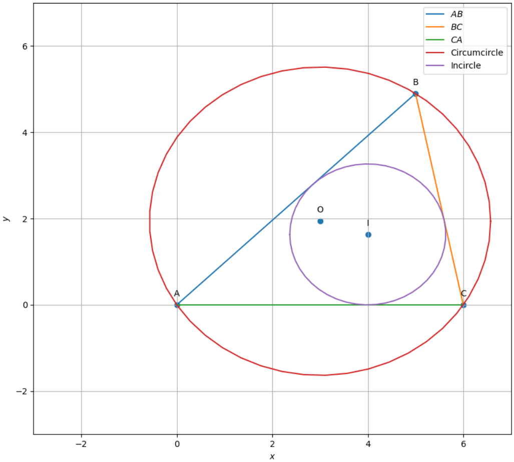
Code VIII

```
                textcoords="offset␣points",
                xytext=(0,10),
                ha='center')

# Configure plot
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.axis([-3, 7, -3, 7])
plt.show()

# Free memory
lib.free(v_ptr)
```

Plot



(Mihir asked me to plot the circles of the triangle)

Bonus Problem: Alternate Method

Do it via intersection of circles, using matrices

Equations

Let the vertices of triangle be $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and lengths of the sides opposing them be denoted by $a = 5cm$, $b = 6cm$ and $c = 7cm$ respectively.

Let $\mathbf{A} = \mathbf{0}$ and $\mathbf{C} = \begin{pmatrix} b \\ 0 \end{pmatrix}$, then $\mathbf{B}$ satisfies

$$\|\mathbf{x} - \mathbf{A}\| = c$$
$$\|\mathbf{x} - \mathbf{C}\| = a$$

Since a point that lies on 2 circles lies on their radical axis, $\mathbf{B}$ lies on the line

$$\|\mathbf{x}\|^2 - 2\mathbf{C}^\top\mathbf{x} + b^2 - a^2 = \|\mathbf{x}\|^2 - 2\mathbf{A}^\top\mathbf{x} - c^2$$
$$(\mathbf{C} - \mathbf{A})^\top\mathbf{x} = \frac{b^2 + c^2 - a^2}{2}$$

Solving Intersection

Let $\mathbf{n} = (\mathbf{C} - \mathbf{A})$ and $d = \frac{b^2+c^2-a^2}{2}$
Let parametric form of line be $\mathbf{x} = \mathbf{h} + k\mathbf{m}$
Where $\mathbf{h} = \frac{\mathbf{n}d}{\|\mathbf{n}\|^2}$ and $\mathbf{m}$ be perpendicular to $\mathbf{n}$ i.e $\mathbf{m} = \begin{pmatrix} -\mathbf{n}_2 \\ \mathbf{n}_1 \end{pmatrix}$
Substituting $\mathbf{x}$ in equation of circle

$$(\mathbf{h} + k\mathbf{m} - \mathbf{A})^\top(\mathbf{h} + \mathbf{k}m - \mathbf{A}) = c^2$$
$$((\mathbf{h} - \mathbf{A}) + k\mathbf{m})^\top((\mathbf{h} - \mathbf{A}) + k\mathbf{m}) = c^2$$
$$(\mathbf{h} - \mathbf{A})^\top(\mathbf{h} - \mathbf{A}) + 2k\mathbf{m}^\top(\mathbf{h} - \mathbf{A}) + k^2\mathbf{m}^\top\mathbf{m} = c^2$$

Since $\mathbf{m}^T\mathbf{h} = 0$

$$(\mathbf{h} - \mathbf{A})^\top(\mathbf{h} - \mathbf{A}) - 2k\mathbf{m}^\top\mathbf{A} + k^2\mathbf{m}^\top\mathbf{m} = c^2$$

Solving for K

Using quadratic formula for k,

$$k = \frac{2\mathbf{m}^\top \mathbf{A} \pm \sqrt{4(\mathbf{m}^\top \mathbf{A})^2 - 4(\mathbf{m}^\top \mathbf{m})((\mathbf{h} - \mathbf{A})^\top (\mathbf{h} - \mathbf{A}) - c^2)}}{2\mathbf{m}^\top \mathbf{m}}$$

$$k = \frac{\mathbf{m}^\top \mathbf{A} \pm \sqrt{(\mathbf{m}^\top \mathbf{A})^2 - (\mathbf{m}^\top \mathbf{m})((\mathbf{h} - \mathbf{A})^\top (\mathbf{h} - \mathbf{A}) - c^2)}}{\mathbf{m}^\top \mathbf{m}}$$

Since we have assumed $\mathbf{A} = \mathbf{0}$, We can simplify this to

$$k = \frac{\pm\sqrt{(c^2 - \mathbf{h}^\top \mathbf{h})}}{\sqrt{\mathbf{m}^\top \mathbf{m}}}$$

Then $\mathbf{x}$ is

$$\mathbf{x} = \mathbf{h} \pm \frac{\sqrt{c^2 - \|\mathbf{h}\|^2}\,\mathbf{m}}{\|\mathbf{m}\|}$$

Solution

Now substituting data,

$$d = 30$$

$$\mathbf{h} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$$

$$\mathbf{m} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

$$k = \pm\frac{\sqrt{24}}{6}$$

$$\mathbf{B} = \begin{pmatrix} 5 \\ \pm\sqrt{24} \end{pmatrix}$$

Code I

This code can be found at

https://github.com/shivenBajpai/EE1030/blob/main/Question_6/Codes/alt.c
https://github.com/shivenBajpai/EE1030/blob/main/Question_6/Codes/alt.py

```
// alt.c

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include "matfun.h"
#include "geofun.h"

// Calculate vertices given the lengths of a triangle, returns
    array of matrices [A, B1, B2, C]
double*** calculateVertices(double a, double b, double c) {

    if (fmax(a,fmax(b,c)) >= (a+b+c)/2) {
        printf("Lengths given cannot form a valid triangle!");
```

Code II

```
        return NULL;
    }

    double** A = createMat(2, 1);
    double** C = createMat(2, 1);
    C[0][0] = b;

    double** n = Matsub(C, A, 2, 1);
    double d = (pow(b, 2) + pow(c, 2) - pow(a, 2))/2;

    double** m = createMat(2, 1);
    m[0][0] = -1 * n[1][0];
    m[1][0] = n[0][0];

    double** h = Matscale(n, 2, 1, d/pow(Matnorm(n, 2),2));
    double** temp = Matscale(m, 2, 1, sqrt(pow(c, 2) - pow(
        Matnorm(h,2), 2))/Matnorm(m, 2));
```

Code III

```
    double** B1 = Matadd(h, temp, 2, 1);
    double** B2 = Matsub(h, temp, 2, 1);

    freeMat(n, 2);
    freeMat(m, 2);
    freeMat(h, 2);
    freeMat(temp, 2);

    double*** returnValues = malloc(sizeof(double**)*2*1*4);
    returnValues[0] = A;
    returnValues[1] = B1;
    returnValues[2] = B2;
    returnValues[3] = C;

    return returnValues;
}
```

Code IV

```
# alt.py

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

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

# Define the argument and return types for the C function
lib.calculateVertices.argtypes = [ctypes.c_double, ctypes.
    c_double, ctypes.c_double]
lib.freeMat.argtypes = [ctypes.c_void_p, ctypes.c_int]
lib.calculateVertices.restype = ctypes.POINTER(ctypes.POINTER(
    ctypes.POINTER(ctypes.c_double)))
```

Code V

```
# Call the C function
a, b, c = 5.0, 6.0, 7.0 # Example values for the sides of the
    triangle
vertices_ptr = lib.calculateVertices(a, b, c)

if not vertices_ptr: sys.exit(1)

A = np.array([vertices_ptr[0][0][0], vertices_ptr[0][1][0]]).
    reshape(-1, 1);
B1 = np.array([vertices_ptr[1][0][0], vertices_ptr[1][1][0]]).
    reshape(-1, 1);
B2 = np.array([vertices_ptr[2][0][0], vertices_ptr[2][1][0]]).
    reshape(-1, 1);
C = np.array([vertices_ptr[3][0][0], vertices_ptr[3][1][0]]).
    reshape(-1, 1);

# Plotting
plt.figure()
```

Code VI

```
# Plot points A, B1, B2, and C
plt.scatter([A[0], B1[0], B2[0], C[0]], [A[1], B1[1], B2[1], C
    [1]], color='k')
plt.text(A[0]-0.5, A[1], 'A', fontsize=12, ha='right')
plt.text(B1[0]-0.5, B1[1], 'B1', fontsize=12, ha='right')
plt.text(B2[0]+1, B2[1]-1, 'B2', fontsize=12, ha='right')
plt.text(C[0]-0.5, C[1]+0.5, 'C', fontsize=12, ha='right')

# Draw lines for triangle A B1 C
plt.plot([A[0], B1[0]], [A[1], B1[1]], color='brown')
plt.plot([B1[0], C[0]], [B1[1], C[1]], color='orange')
plt.plot([A[0], C[0]], [A[1], C[1]], color='red')

# Draw lines for triangle A B2 C
plt.plot([A[0], B2[0]], [A[1], B2[1]], color='brown')
plt.plot([B2[0], C[0]], [B2[1], C[1]], color='orange')
plt.plot([A[0], C[0]], [A[1], C[1]], color='red')
```

Code VII

```
# Plot circles around A and C
circle_A = plt.Circle((A[0], A[1]), c, color='blue', fill=False)
circle_C = plt.Circle((C[0], C[1]), a, color='green', fill=False)

plt.gca().add_patch(circle_A)
plt.gca().add_patch(circle_C)

# Setting the axis limits and labels
plt.xlim(-c-1, C[0]+a+1)
plt.ylim(-max(c, a)-1, max(c, a)+1)
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel('X')
plt.ylabel('Y')

# Display the plot
plt.grid(True)
plt.show()
```

Code VIII

```
#Free the memory later
lib.freeMat(vertices_ptr[0], 2)
lib.freeMat(vertices_ptr[1], 2)
lib.freeMat(vertices_ptr[2], 2)
lib.freeMat(vertices_ptr[3], 2)
lib.free(vertices_ptr)
```

Plot