Department of Civil and Environmental Engineering, Spring Semester, 2024

ENCE 201 Midterm 1, Open Notes and Open Book

Name : _____

E-mail (print neatly!): _____

Exam Format and Grading. This take home midterm exam is open notes and open book. You need to comply with the **university regulations for academic integrity**.

There are three questions. Partial credit will be given for partially correct answers, so please show all your working.

Please see the **class web page for instructions on how to submit your exam paper**.

Question	Points	Score
1	10	
2	15	
3	15	
Total	40	

Question 1: 10 points. This question covers linear matrix equations and their solution.

Consider the family of equations:

$$\begin{aligned} x + ky &= 1 \\ kx + y &= 1 \end{aligned} \tag{1}$$

where x and y are variables and k is a number.

[1a] (2 pts) Write equations 1 in matrix form A.X = B.

[1b] (2 pts) In part [1a], what are the dimensions of matrices A, X and B?

[1c] (2 pts) Write down the augmented matrix $[A|B]$.

[1d] (4 pts) Find values of k for which equations 1 will have exactly one solution. What are the corresponding values of x and y?

Question 2 (15 points). This question covers use of Python to compute and print the maximum and minimum coordinate positions, and perimeter of the six-sided polygon shown in Figure 1.

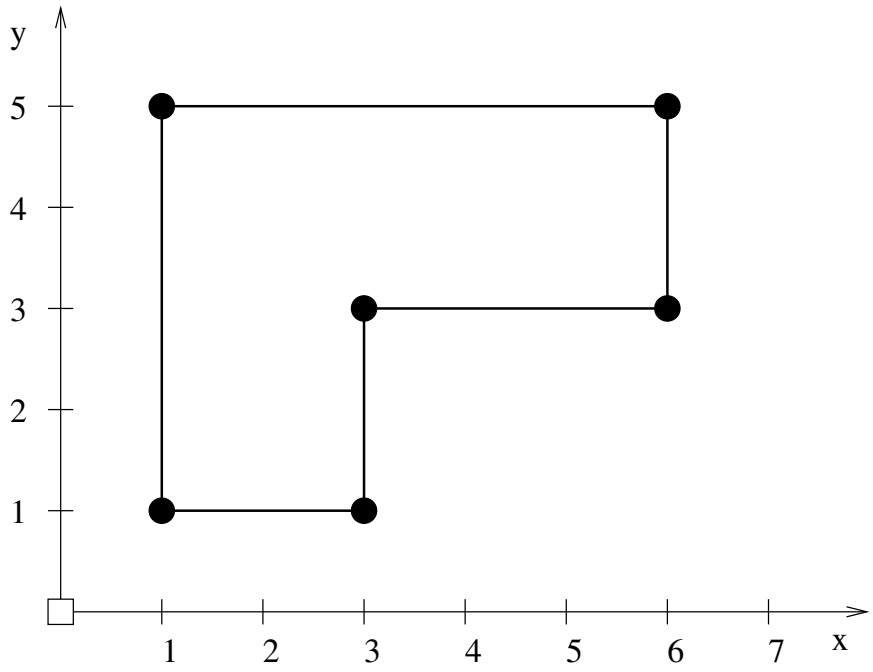


Figure 1. Six-sided irregular polygon.

Look the Python code over carefully and answer the questions that follow:

```
# ================================================================
# TestPolygonSixSides.py: Compute max/min nodal coordinatees and
# perimeter of six-sided polygon.
#
# Written by: Mark Austin                                March 2024
# ================================================================

import math
import numpy as np

# Function to print two-dimensional matrices ...

def PrintMatrix(name, a):
    print("Matrix: {:s} ".format(name) );
    for row in a:
        for col in row:
            print("{:8.4f}".format(col), end=" ")
        print("")

# Compute polygon perimeter ...

def perimeter( coord ):
    norows = coord.shape[0];
```

```
    dperimeter = 0.0;
    for i in range(1,norows):
        dx = coord[i][0] - coord[i-1][0];
        dy = coord[i][1] - coord[i-1][1];
        dperimeter = dperimeter + math.sqrt( dx*dx + dy*dy );

    dx = coord[norows-1][0] - coord[0][0];
    dy = coord[norows-1][1] - coord[0][1];
    dperimeter = dperimeter + math.sqrt( dx*dx + dy*dy );

    return dperimeter;

# ============================================================
# main method ...
# ============================================================

def main():
    print("--- Enter TestPolygonSixSides.main()         ... ");
    print("--- ======================================== ... ");

    print("--- Part 1: Initialize coefficients for matrix equations ... ");

    coord = np.array( [ ( 1.0, 1.0 ),
                        ( 1.0, 5.0 ),
                        ( 6.0, 5.0 ),
                        ( 6.0, 3.0 ),
                        ( 3.0, 3.0 ),
                        ( 3.0, 1.0 ) ] );

    PrintMatrix("Polygon Coordinates", coord);

    print("--- Part 2: Max/min coordinate positions ... ");

    print("--- Min x = {:f} ...".format( min ( coord[:,0] ) ) );
    print("--- Max x = {:f} ...".format( max ( coord[:,0] ) ) );
    print("--- Min y = {:f} ...".format( min ( coord[:,1] ) ) );
    print("--- Max y = {:f} ...".format( max ( coord[:,1] ) ) );

    print("--- Part 3: Compute and print distance of coords from origin ... ");

    distance = np.zeros( coord.shape[0] )
    for i in range(6):
        x = coord[i][0];
        y = coord[i][1];
        distance[i] = math.sqrt(x**2 + y**2)

    print("--- Min distance from origin = {:f} ...".format( min ( distance ) ) );
    print("--- Max distance from origin = {:f} ...".format( max ( distance ) ) );

    print("--- Part 4: Compute and print perimeter and area of polygon ... ");

    print("--- Polygon perimeter = {:f} ...".format( perimeter(coord) ) );

    print("--- ======================================== ... ");
```

```
    print("--- Enter TestPolygonSixSides.main()         ... ");

# call the main method ...

if __name__ == "__main__":
    main()
```

The program output is:

```
--- Enter TestPolygonSixSides.main()         ...
--- ======================================== ...
--- Part 1: Initialize coefficients for matrix equations ...
Matrix: Polygon Coordinates
  1.0000   1.0000
  1.0000   5.0000
  6.0000   5.0000
  6.0000   3.0000
  3.0000   3.0000
  3.0000   1.0000
--- Part 2: Max/min coordinate positions ...
--- Min x = 1.000000 ...
--- Max x = 6.000000 ...
--- Min y = 1.000000 ...
--- Max y = 5.000000 ...
--- Part 3: Compute and print distance of coords from origin ...
--- Min distance from origin = 1.414214 ...
--- Max distance from origin = 7.810250 ...
--- Part 4: Compute and print perimeter and area of polygon ...
--- Polygon perimeter = 18.000000 ...
--- ======================================== ...
--- Enter TestPolygonSixSides.main()         ...
```

[2a] (2 pts) What does the line:

```
import numpy as np
```

do, and why is it needed in this program?

[2b] (2 pts) What does the line:

```
import math
```

do, and why is it needed in this program?

[2c] (2 pts) Write a small script of Python code to retrieve and print the matrix dimensions, and number of rows and columns in the coord matrix.

[2d] (3 pts) Briefly explain how the various parts (i.e., use of builtin functions and formatting specification) of the statement:

```
print("--- Min x = {:f} ...".format( min ( coord[:,0] ) ) );
```

work.

[2e] (3 pts) Now let's consider the block of code:

```
distance = np.zeros( coord.shape[0] )
for i in range(6):
    x = coord[i][0];
    y = coord[i][1];
    distance[i] = math.sqrt(x**2 + y**2)
```

Create a table that shows the iteration value i, and values of x, y and distance[i], for each iteration of the loop computation.

[2f] (3 pts) Finally, consider the statement:

```
print("--- Polygon perimeter = {:f} ...".format( perimeter(coord) ) );
```

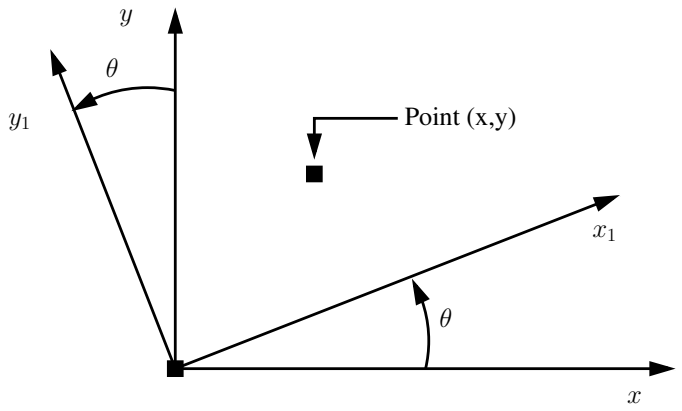
and the block of code:

```
dperimeter = 0.0;
for i in range(1,norows):
    dx = coord[i][0] - coord[i-1][0];
    dy = coord[i][1] - coord[i-1][1];
    dperimeter = dperimeter + math.sqrt( dx*dx + dy*dy );

dx = coord[norows-1][0] - coord[0][0];
dy = coord[norows-1][1] - coord[0][1];
dperimeter = dperimeter + math.sqrt( dx*dx + dy*dy );
```

within the function perimeter. Construct a table that shows how the perimeter computation systematically assembles the polygon perimeter.

Question 3 (15 points). Suppose that a x-y coordinate system is rotated anticlockwise by an angle θ to create a new coordinate system x_1-y_1.



The matrix product:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2}$$

describes how points in the x-y coordinate system are transformed into the x_1-y_1 coordinate system. Let us denote the 2-by-2 coordinate transformation matrix A(θ).

[3a] (2 pts) Is matrix A(θ) skew-symmetric?

[3b] (3 pts) For two rotations θ_1 and θ_2 verify that:

$$A(\theta_1)A(\theta_2) = A(\theta_1 + \theta_2). \tag{3}$$

Show all of your working ...

[3b] continued ...

[3c] (2 pts) Now suppose that the coordinate axes are rotated anticlockwise by angle θ **three** times. Draw and label a sequence of coordinate systems (similar to the diagram above) that illustrate the essential features of this scenario.

[3d] (5 pts) Using the **ideas** and **results** of **Parts [3b] and [3c]**, derive a formula for $\cos(3\theta)$ in terms of $\cos(\theta)$ alone, i.e.,

$$\cos(3\theta) = 4\cos^3(\theta) - 3\cos(\theta). \tag{4}$$

Show all of your working.

[3e] (3 pts) Show that for all values of x, y, z and $a \geq 0$ the triple product of matrices:

$$\begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} 1+a & -a & 0 \\ -a & 1+a & -a \\ 0 & -a & 1+a \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \geq 0 \tag{5}$$