Programming Assignment 2

Assigned: Sept. 19
Due: Oct. 3

Problem 1

The process that you considered in programming assignment 1, problem 2, is invertible if there are an odd number of points, though not if there are an even number of points (see problem set 2, problem 3). That is, if \( n \) is odd, then given a sequence of \( k \) points in the plane \( \vec{u}_1 \ldots \vec{u}_k \) you can find a sequence \( \vec{v}_1 \ldots \vec{v}_k \) such that

\[
\begin{align*}
\vec{u}_1 & \text{ is the midpoint of } \vec{v}_1 \text{ and } \vec{v}_2, \\
\vec{u}_2 & \text{ is the midpoint of } \vec{v}_2 \text{ and } \vec{v}_3, \\
\vdots & \\
\vec{u}_{k-1} & \text{ is the midpoint of } \vec{v}_{k-1} \text{ and } \vec{v}_k, \\
\vec{u}_k & \text{ is the midpoint of } \vec{v}_k \text{ and } \vec{v}_1.
\end{align*}
\]

Write a MATLAB function \texttt{InvertMidpoints(Q)} which takes as argument a polygon \( Q \), represented in the form used in programming assignment 1, and returns the polygon \( P \) such that each vertex in \( Q \) is the midpoint of a side in \( P \).

Thus, for example, if \( Q \) is the array

\[
\begin{bmatrix}
1 & 3 & 1 & 2 & 3 \\
4 & 4 & 3 & 6 & 3
\end{bmatrix}
\]

then \texttt{InvertMidpoint(Q)} should return

\[
\begin{bmatrix}
0 & 2 & 4 & -2 & 6 \\
0 & 8 & 0 & 6 & 6
\end{bmatrix}
\]

Hint: First, notice that the \( x \) and \( y \) coordinates in this problem are completely decoupled. That is, if we write \( \vec{u}_i = (x_i, y_i) \), and \( \vec{v}_i = (a_i, b_i) \), then we have

\[
\begin{align*}
x_1 &= (a_1 + a_2)/2. \\
x_2 &= (a_2 + a_3)/2. \\
\vdots & \\
x_{k-1} &= (a_{k-1} + a_k)/2. \\
x_k &= (a_k + a_1)/2.
\end{align*}
\]

and the identical equations relates the \( y \)'s to the \( b \)'s. Thus, for example, for \( k = 5 \), the 5-dimensional vectors \( \vec{x}, \vec{y}, \vec{a}, \vec{b} \) satisfy the equations \( \vec{x} = C\vec{a}, \vec{y} = C\vec{b} \) where \( C \) is the matrix of coefficients,

\[
C = \begin{bmatrix}
1/2 & 1/2 & 0 & 0 & 0 \\
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 \\
1/2 & 0 & 0 & 0 & 1/2
\end{bmatrix}
\]
So you can write the program by (a) creating the matrix $C$ for size $k$; (b) solving the two systems of equations; (c) putting the two solutions together. (Do NOT write code to solve systems of linear equations; use the built-in MATLAB solver.)

**Problem 2**

(This is assignment 5.2.a from the textbook.)

Write a function `PolyInterpolate(M)` to do polynomial interpolation, as described in Application 3.9 of section 3.6.1. The input parameter $M$ is a $2 \times n$ matrix, where each column holds the $x$- and $y$-coordinates of one point. You should assume that no two points have the same $x$ coordinates. The function should return the coefficients $a_{n-1}, \ldots, a_0$ of the $(n-1)$-degree $y = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ that fits the point. The function should also generate a plot of the curve with the input points, as shown in Figure 3.4.

For instance, for the example in Application 3.9. the function call `PolyInterpolate([-3,-1,0,2,4; 1,0,5,0,1])` should return the vector $[0.2167, -0.4333, -2.7167, 2.933, 5.0000]$ and generate the solid curve of Figure 3.4.