Programming Assignment: Midpoints of Polygons

Problem 1:

There is a theorem that states that, if you carry out the following procedure:

```
P = any polygon (this can be concave or even cross itself).
loop {
   compute the midpoint of each side of P
   P = the polygon formed by connecting these midpoints in sequence;
  }
```

Then `P` will converge toward a series of points that lie on an ellipse. (Picture on next page. Note: Because of the symmetry in this particular case, two of the points collapse into one, and the figure becomes a perfect rectangle, but neither of this happens in general.)

A. Assume that P is represented as a $2 \times n$ matrix, where each column is the coordinates of one vertex of P. For example, the polygon with vertices at $\langle 0, 0\rangle$, $\langle 2, 8\rangle$, $\langle 4, 0\rangle$, - $\langle -2, 6\rangle$, $\langle 6, 6\rangle$ would be represented as the array,

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

Write a MATLAB function `ConnectMidpoints(P)` that, given a polygon `P` constructs the polygon that results from connecting the midpoints of `P` in sequence. For instance if `P` is the matrix above then `ConnectMidpoints(P)` would return the array

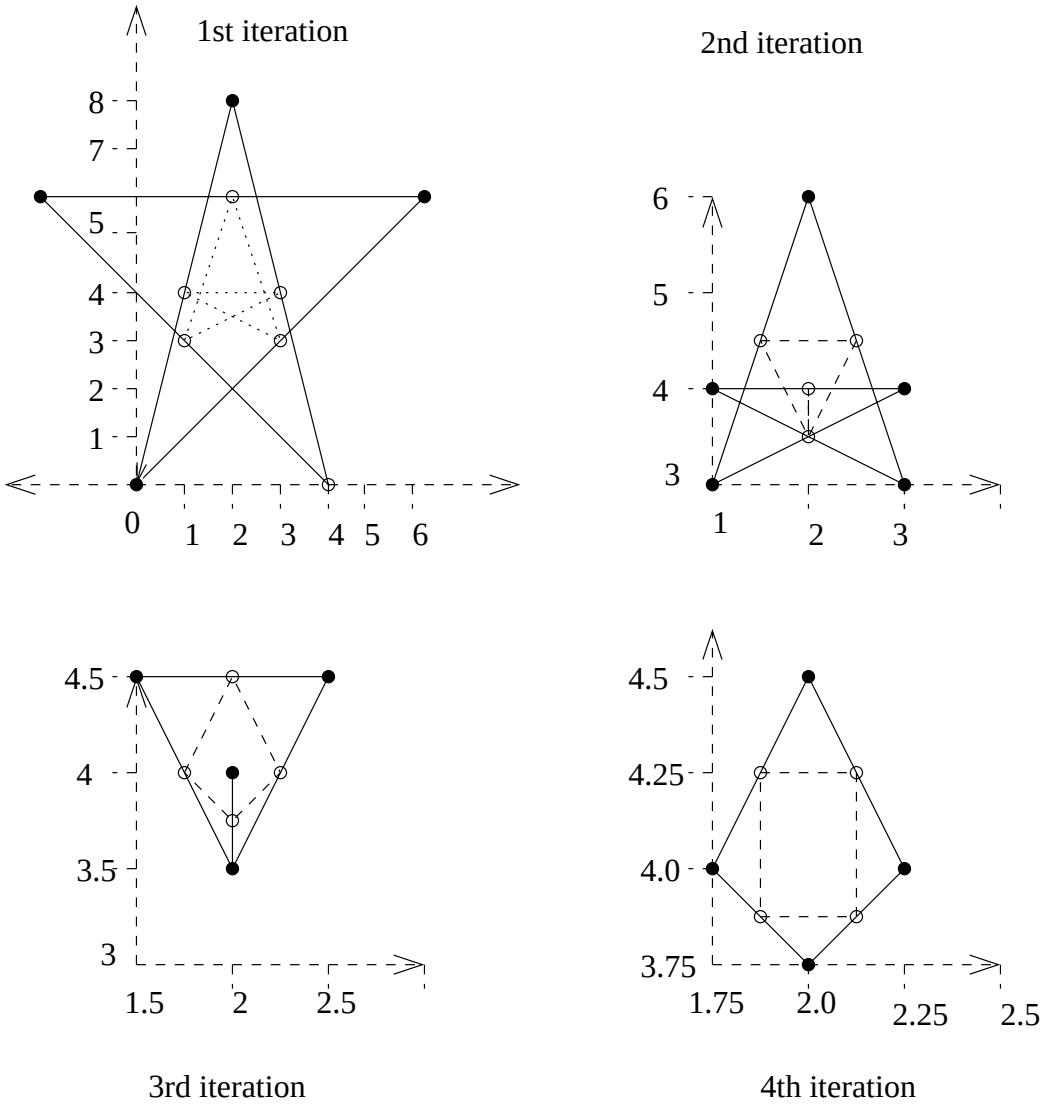
$$Q = \begin{bmatrix} 1 & 3 & 1 & 2 & 3 \\ 4 & 4 & 3 & 6 & 3 \end{bmatrix}$$

Each column of `Q` is constructed by taking the average of two consecutive columns of `P` and dividing by 2; e.g. `Q[:,1] = 1/2(P[:,1]+P[:,2])`. The last column of `Q` is the average of the last and first column of `P`; i.e. `Q[:,1] = 1/2(P[:,5]+P[:,1])`.

Your code should of course work for polygons with any number of points, not just polygons with 5 points.

B. Write a MATLAB function `ConvergingPolygons(P,N)` which takes as input a polygon `P` and a number `N` and draws pictures of the first N polygons in this sequence, starting with `P`. Let MATLAB adjust the scale on each successive picture, or the picture will soon become too small to see. Also, as always with geometric drawings in Matlab, call `axis equal` to make sure that the x and y axes have the same scale.

(Your pictures, produced by MATLAB, will not look exactly like the picture on the next page, which was drawn with a line-drawing program.)



Problem 2

The process that you considered in problem 1, is invertible if there are an odd number of points, though not if there are an even number of points (see
Midpoint Problem) That is, if k 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

$\vec{u}_1$ is the midpoint of $\vec{v}_1$ and $\vec{v}_2$.
$\vec{u}_2$ is the midpoint of $\vec{v}_2$ and $\vec{v}_3$.
. . .
$\vec{u}_{k-1}$ is the midpoint of $\vec{v}_{k-1}$ and $\vec{v}_k$.
$\vec{u}_k$ is the midpoint of $\vec{v}_k$ and $\vec{v}_1$.

Write a MATLAB function `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

$$\texttt{Q} = \begin{bmatrix} 1 & 3 & 1 & 2 & 3 \\ 4 & 4 & 3 & 6 & 3 \end{bmatrix}$$

then `InvertMidpoint(Q)` should return

$$\texttt{P} = \begin{bmatrix} 0 & 2 & 4 & -2 & 6 \\ 0 & 8 & 0 & 6 & 6 \end{bmatrix}$$

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

$x_1 = (a_1 + a_2)/2.$
$x_2 = (a_2 + a_3)/2.$
. . .
$x_{k-1} = (a_{k-1} + a_k)/2.$
$x_k = (a_k + a_1)/2.$

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.)