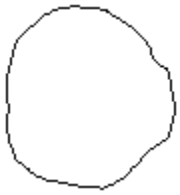
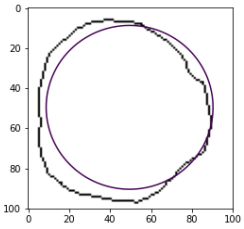
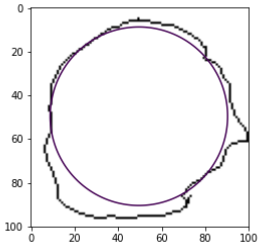
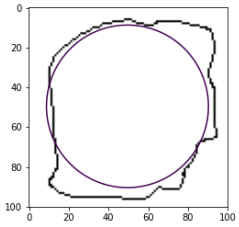
1) **Hand draw at least 3 “circle” plots of resolution 100 × 100 pixels, and use them as inputs for the given program.**





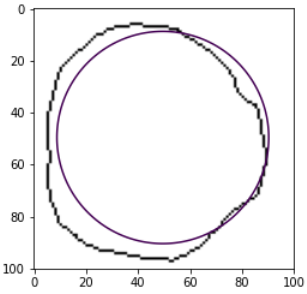


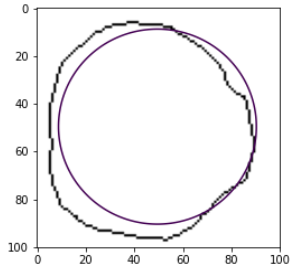


2) **The given program uses numpy.linalg.lstsq to solve the least square problem**

   (a) **Use your own drawing to compare the results with those by numpy.linalg.lstsq.**

Least square method vs normal method on my drawing number 1.





When compared to each other, I can see a clear difference. To my eyes, the least square method is more centered than the normal method and the edges are more precise.

(b) How to measure the closeness of your drawings to a perfect circle?

To measure the closeness of your drawings to a perfect circle is the radius which is the distance from the center point of any endpoint on the circle and for this reason Figure 1(a) is better than that of Figure(c). Whereas the radius of Figure(a) is more consistent than Figure(c). However, in our case, we first have to find the slope of our circle. In other words, let us be given the values of c_1 and c_2 of the circular formula $(x - c_1)^2 + (y - c_2)^2 = r^2$, then we could find solve for radius r and measure the closeness of our drawings to a perfect circle.

How to obtain this measurement from the returned values of numpy.linalg.lstsq?

To obtain the radius, it is found in the "draw circle" function $radius = math.sqrt(x[2] + x[0] * x[0] + x[1] * x[1])$. However, to obtain this measurement from the returned values of numpy.linalg.lstsq, we first have to find the c_1 and c_2. Thus, to get the values of these two we can implement the code to $c_1, c_2 = np.linalg.lstsq(A, y, rcond = None)[0]$ where we easily calculate the radius and get closer to drawing the perfect circle.

How to obtain it from the normal equation you implemented?

From the normal equation that I implemented you can use

$c_1 = np.mat(xArray)$ and $c_2 = np.mat(yArray)$ where the $xArray$ and $yArray$ are arrays that correspond to the equation $Ax = b$ which is also similar to $y = c_1 x + c_2$

(c) What are the ranks of your examples?

The rank of my examples and base test case are all 3.

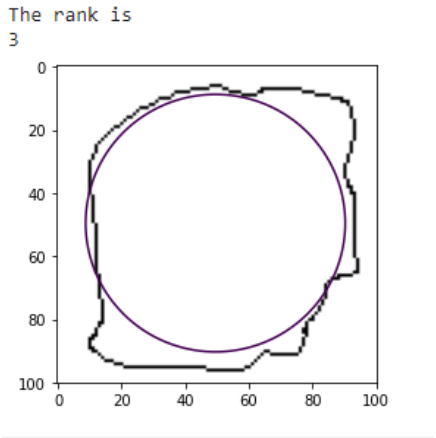
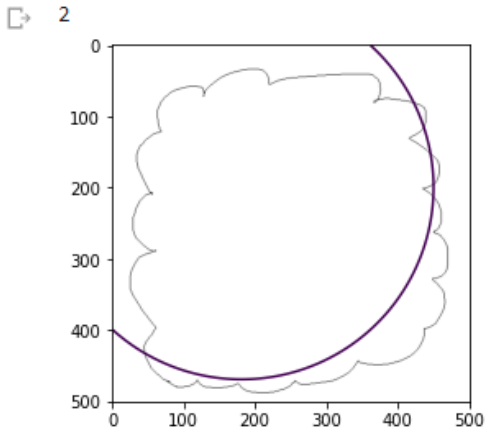
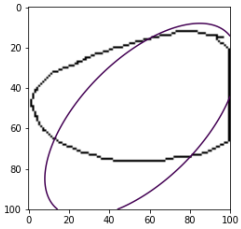
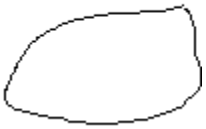


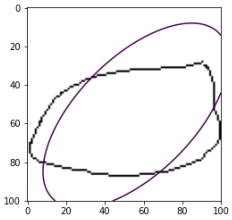
Image whose rank output by numpy.linag.lstsq is less than 3.

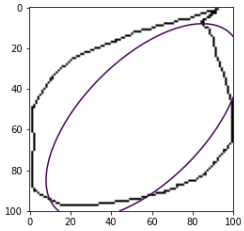


3) Design and implement an algorithm to find the best fitting ellipse using least square algorithm. Test the program with three hand drawing ellipses.









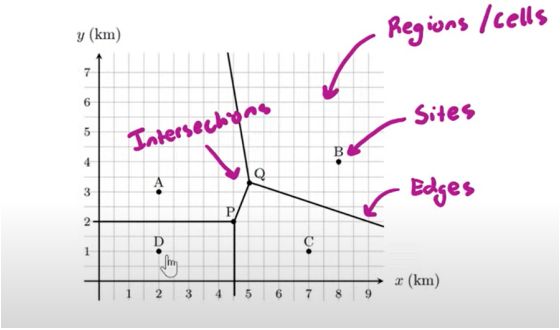
4)

(a)Which is the time complexity of your algorithm.

To start, the time complexity of the algorithm is $O(C^2N)$. Namely to explain the time complexity of the equation $A^TAx = A^Tb$, alone the transpose multiplication is $O(C^2N)$ and thus A^Tx is $O(CN)$, so total would be $O(C^3)$, but since $O(C^2N)$ dominates $O(CN)$, the total time complexity for the normal equation is $O(C^2N)$.

5)Look up what Voronoi diagram and Farthest Point Voronoi Diagram are, and how to use them to find the minimum enclosing circle for the given point?

In mathematics, a Voronoi diagram is a plane which is divided into regions close to each other depending on the set of objects. In a way we could say that the Voronoi diagram is a map such that we can find a way to a certain direction. The main point of a Voronoi diagram is to find the closest site or site given the point and which region you are currently in.



The farthest point Voronoi diagram is when a set of n points the $(n-1)^{th}$-order in a Voronoi diagram. Hence it could be used to identify Euclidean distance and may structure a topological tree for the Voronoi diagram.

To find the minimum enclosing circle for the given point, given a set of points on the plane, you mark a center point in the plane which will approach the solution by hill climbing. The way of hill climbing is to calculate the distance from the center point to each given point and after finding the farthest point move along the distance towards the point by a measure of d such that it is big enough and near 1 to change the center of the circle. Hence, repeat the process for all the points until you get the smallest diameter.