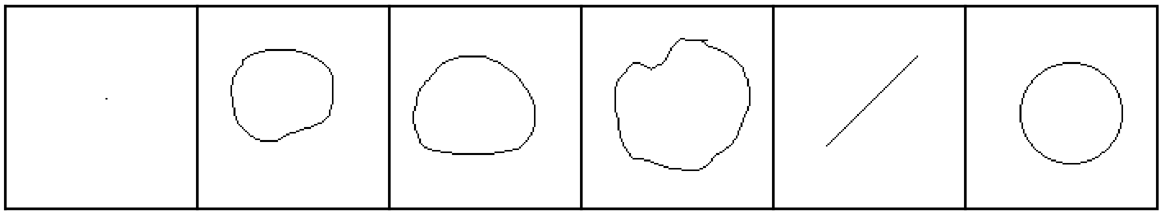
Assignment 3
2021 Fall EECS205002 Linear Algebra

(1) Circle plots of resolution 100 x 100 pixels

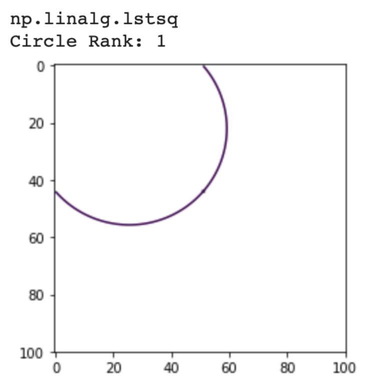
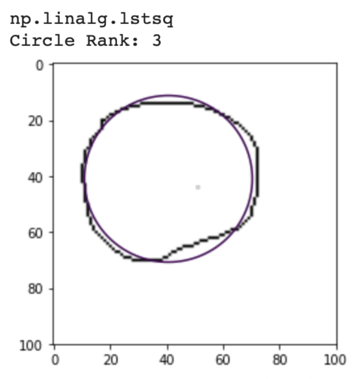
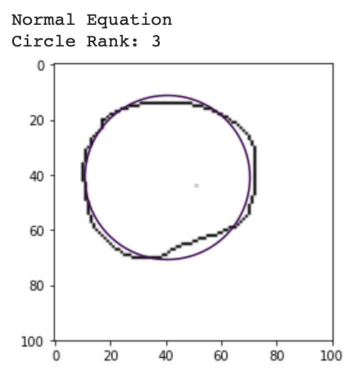


1.1 Circle plot 1 to 6

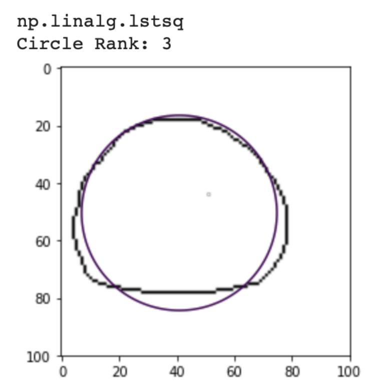
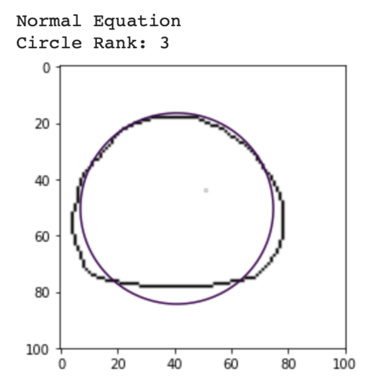
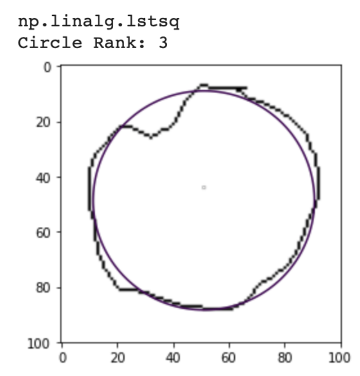
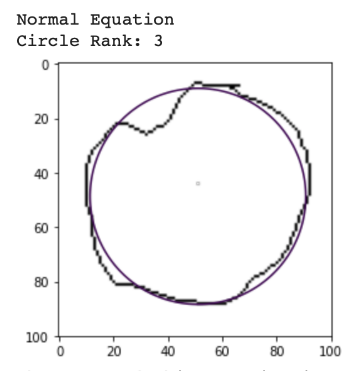
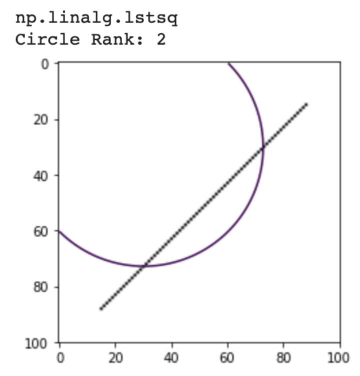
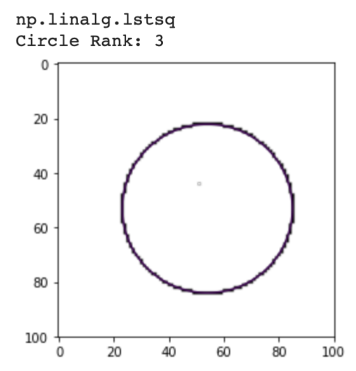
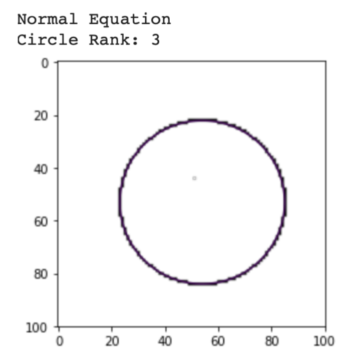
(2)

a) The result of the least squares problem when using `numpy.linalg.lstsq` with those when implementing normal equations is the same. But the normal equation may not work if the matrix n x m is singular. Therefore the result is identical. Because 2 of my images have rank lower than 3. The determinant is 0 and therefore they are singular. So, it will raise a LinAlgError Singular matrix. In my circle plots, they are circle plot 1 and circle plot 5.

Comparison of circle plots using np.linalg.lstsq vs normal equation

	Singular matrix because Rank < 3, Normal equation does not work.
	

Assignment 3
2021 Fall EECS205002 Linear Algebra

	
	
	Singular matrix because Rank < 3, Normal equation does not work.
	

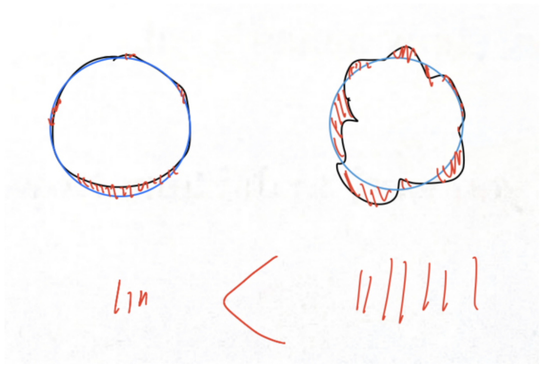
b) To measure the closeness of the drawing to a perfect circle, we can look at the value of r after implementing `numpy.linalg.lstsq`. The smaller the r is, the closer the figure is to a circle. To obtain this measurement from the returned value of `numpy.linalg.lstsq`, we can take the value of r from :

```
[x, r, rank] = np.linalg.lstsq(A, b, rcond=None)[0:3]
```

From the normal equation, we can obtain it by:

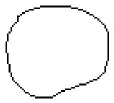
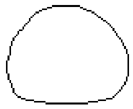
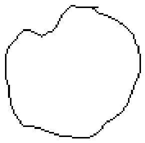
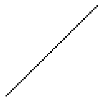
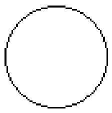
```
def normal_eq(A,b):
    AT = A.T
    AT_A = np.dot(AT,A)
    AT_A_inv = np.linalg.inv(AT_A)
    AT_b = np.dot(AT,b)
    x = np.dot(AT_A_inv,AT_b)
    r_mtrx = b - np.dot(A,x)
    r = 0
    for i in range(len(r_matrix)):
      r += r_mtrx[i]*r_mtrx[i]
    rank = np.linalg.matrix_rank(A)
    return x,r,rank
```

The picture below shows the sum of the residuals, or r of the drawing in Figure 1(a) and 1(c) respectively. The red area is the residuals, it can be seen that in Figure 1(a), the total amount of residuals are less than Figure 1(c) as the red shaded region at the figure is less. Therefore the sum of residuals is also less. Which concludes that Figure 1(a) is more similar to a circle than Figure 1(c). r



The smaller the residual is, the circular the drawing becomes.

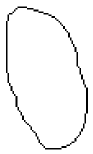
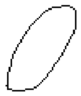
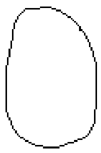
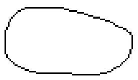
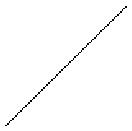
c) One of the return values of `numpy.linalg.lstsq` is the rank, which is the rank for the matrix's dimension. An image that outputs rank less than 3 is a dot, which has rank=1. And a line, which has rank=2. Whereas a circle itself would make a full rank of the matrix when utilizing the `numpy.linalg.lstsq` for perfecting the circle. When we use the images with rank lower than 3, to be implemented using the normal equation, it will break because the lower rank makes them a singular matrix. And a singular matrix has no inverse. Therefore the normal equation does not work.

					
Rank = 1	Rank = 3	Rank = 3	Rank = 3	Rank = 2	Rank = 3

(3) I implemented the equation $c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y = 1$ for the ellipse. Which is a conic section formula that can solve any curve from the conic section. The equation has a total of 5 unknowns. With this formula, we can solve any conic section problem. I implemented this equation by making a linear equation with an n x 5 matrix called A, where n is all the points in the corresponding image. A will then be multiplied to a vector which has all the values of c1 to c5. For the `def ellipse(sp)` function, I assign the values of the Matrix A into `A[i, :] = [x*x, x*y, y*y, x, y]`
And used it also to implement the `def draw_ellipse(x, h, w)` function. Where in the latter part of the code, I assign the value of C (the ellipse) into

```
C = x[0]*a*a + x[1]*a*b + x[2]*b*b + x[3]*a + x[4]*b - 1
```

This linear system can be solved with the least square algorithm that takes the minimum value for each of the constants. Ellipse plots of resolution 100 x 100 pixels:

					
Rank = 5	Rank = 5	Rank = 1	Rank = 5	Rank = 5	Rank = 3

(4) Sampling every single point is redundant and time consuming. However it will produce the most precise drawing that is closest to the image. To make the program faster, I reduce the number of points for the input of least squares, by choosing only 15 points

```
def dataSampling(x):
    chosen=[]
    l = len(x)
    chosen.append(x[0])
    for i in range(1,15):
        chosen.append(x[int(l/15*i)])
    return chosen
```

In the dataSampling function, I chose 15 points, which means that the time should be 15/len(x) of its original speed of O(n), as there is less data. This makes the code more efficient as it is faster. And the score and accuracy of the drawing did not change that significantly.

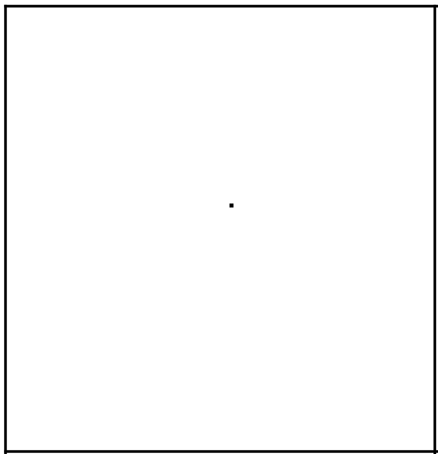
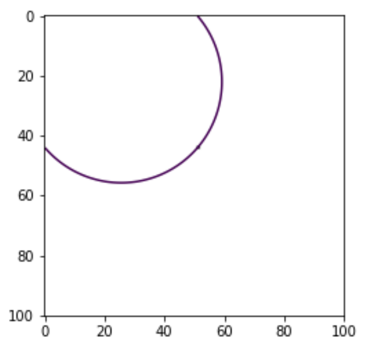
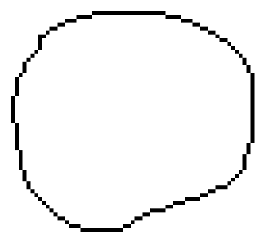
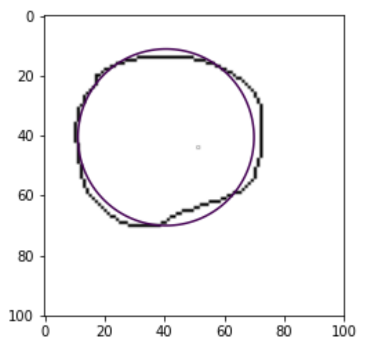
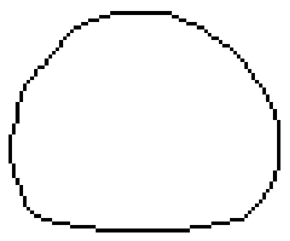
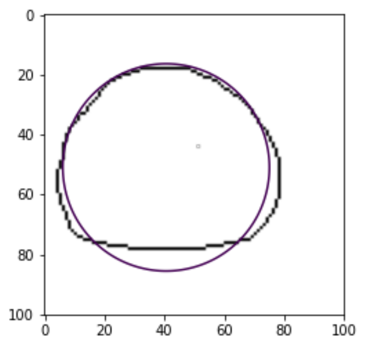
(5) Voronoi diagram is a diagram consisting of coordinates known as sites with lines in between them: edges. These edges divide the diagram into several regions. And each edge corresponds to a pair of sites where it serves as the perpendicular bisector of the two sites. When there are more than 2 sites, the voronoi diagram takes the odd polygon shapes and there would be intersections where the lines cross. This diagram can find the closest site of a certain coordinate. The voronoi diagram focuses on the closest points, and the opposite of it would be where the edges and coordinates correspond to the farthest site, called the farthest point voronoi diagram.

The farthest point voronoi diagram will be able to draw a circle known as the smallest enclosing circle that can fit all the given points. There is a property that the center of the minimum enclosing circle will be located at the vertex of the farthest point voronoi diagram or will coincide with a vertice in the diagram. So by enumerating all the possible centers and getting the one with the minimum radius we can get the optimal point.

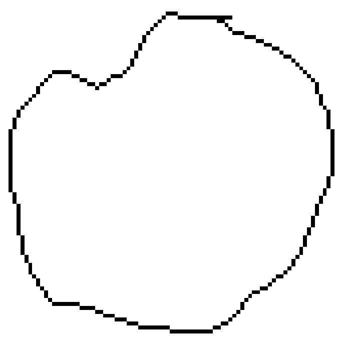
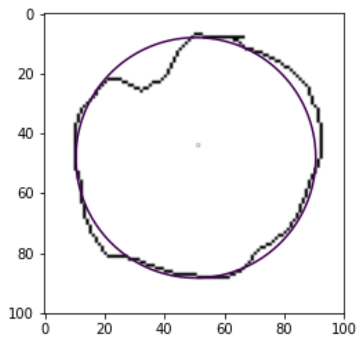
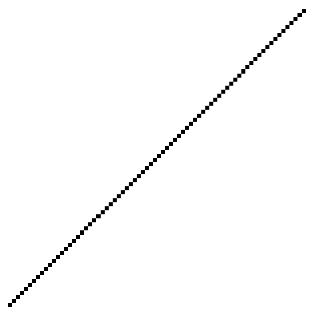
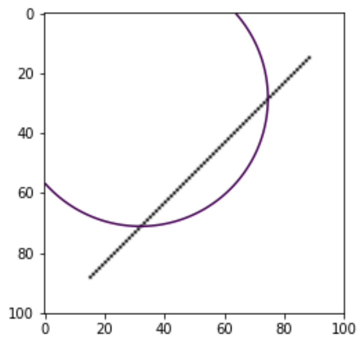
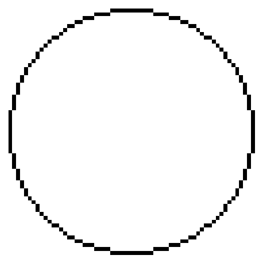
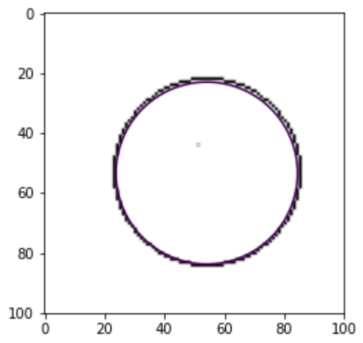
Assignment 3
2021 Fall EECS205002 Linear Algebra

Figures of hand drawn circles, ellipse and their fitted results

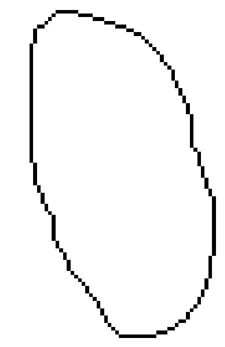
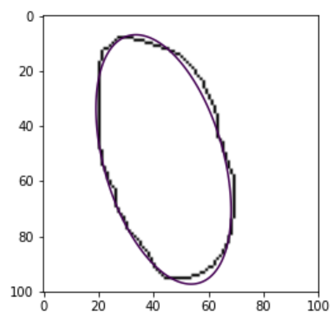
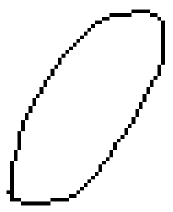
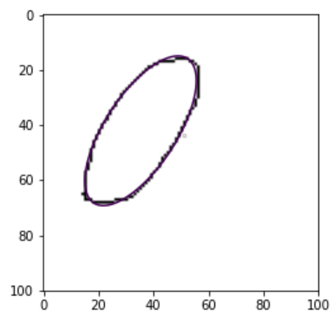
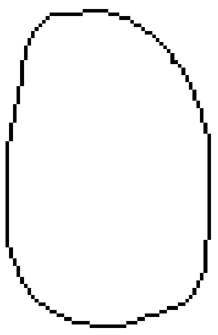
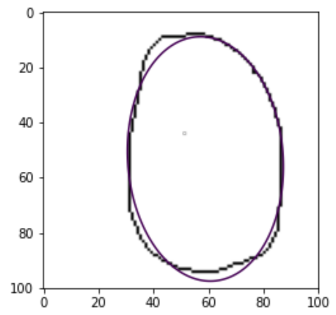
1. Circle

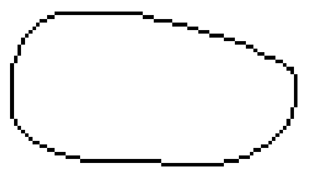
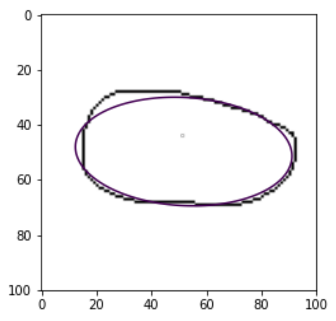
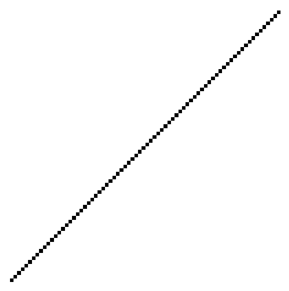
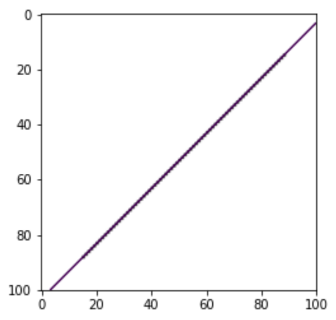
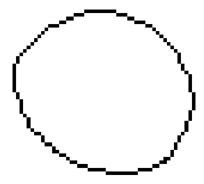
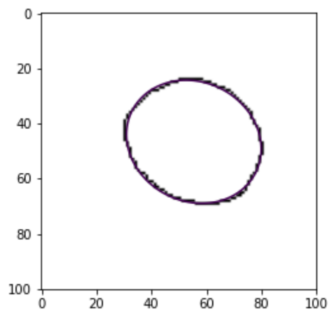
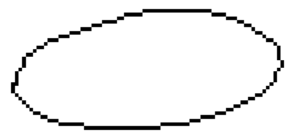
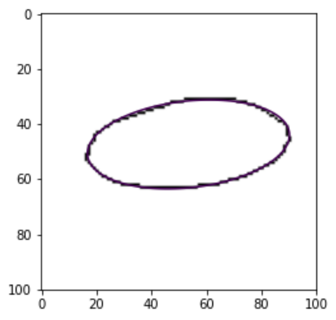
Assignment 3
2021 Fall EECS205002 Linear Algebra

2. Ellipse

Assignment 3
2021 Fall EECS205002 Linear Algebra

	 0 20 40 60 80 100
	 0 20 40 60 80 100
	 0 20 40 60 80 100
	 0 20 40 60 80 100