1.

1.

quadratic equation: $a'x^2+2b'xy+c'y^2+d'x+e'y=-f'$

$$\Rightarrow \left(-\frac{a'}{f'}\right)x^2+2\left(\frac{-b'}{f'}\right)xy+\left(\frac{c'}{-f'}\right)y^2+\left(\frac{d'}{-f'}\right)x+\left(\frac{e'}{-f'}\right)y=1$$

let $a=-\frac{a'}{f'}$, $b=-\frac{2b'}{f'}$, $c=-\frac{c'}{f'}$, $d=-\frac{d'}{f'}$, $e=-\frac{e'}{f'}$

rewrite the equation $\Longrightarrow$ $ax^2+bxy+cy^2+dx+ey=1$

and we have $ax_1^2+bx_1x_2+cx_2^2+dx_1+ex_2=1$

we can also rewrite it in the form:

$$[x_1 \ \ x_2]\begin{bmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+[d \ \ e]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=1$$

$$A=\begin{bmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{bmatrix}, \quad B=[d \ \ e], \quad \vec{x}=\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\Rightarrow \vec{x}^T A\vec{x}+B\vec{x}=1$$

Circle: $\frac{1}{r^2}x_1^2+\frac{1}{r^2}x_2^2=1 \Rightarrow a=c=\frac{1}{r^2}, b=0$

$\det(A)=\frac{1}{r^4}>0$

Ellipse: $\frac{1}{\alpha^2}x_1^2+\frac{1}{\beta^2}x_2^2=1 \Rightarrow a\neq c, b=0$

$\det(A)=\frac{1}{\alpha^2\beta^2}>0$, $\alpha\neq\beta$

Parabola: $x_2^2=4\alpha x_1$, or $x_1^2=4\alpha x_2 \Rightarrow a$ or $c=0, b=0$

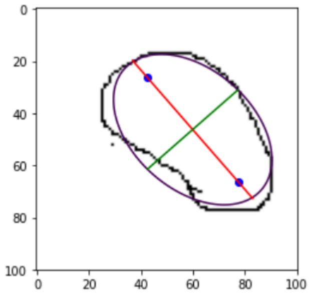
$\det(A)=0$,

Hyperbola: $\frac{x_1^2}{\alpha^2}-\frac{x_2^2}{\beta^2}=1 \Rightarrow a=\frac{1}{\alpha^2}>0, c=\frac{-1}{\beta^2}<0, b=0$

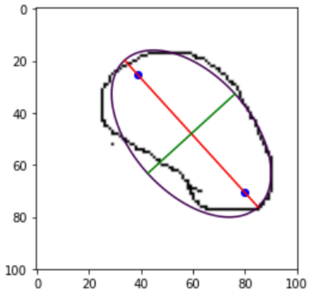
$\det(A)<0$

4.

```
Algorithm 1
============================
[46.22515698 59.92766807] 23.033165766517318 35.16844663129809
Focal points: ( 77.41343417776497 , 66.23867483545345 ),( 42.441901962535674 , 26.211639129051644 )
Major axis: ( 0.7530624772910545 , -0.6579490142072255 )
Minor axis: ( 0.6579490142072255 , 0.7530624772910545 )
```



```
Algorithm 2
============================
[47.96618357 59.4057971 ] 22.735897733827592 38.18966336108739
Focal points: ( 38.753849042646486 , 25.271990854127377 ),( 80.05774516025207 , 70.66037629563107 )
Major axis: ( -0.7396015440125272 , 0.6730449881651939 )
Minor axis: ( -0.6730449881651939 , -0.7396015440125272 )
```



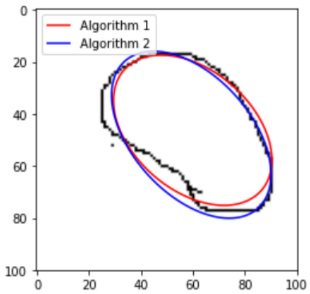
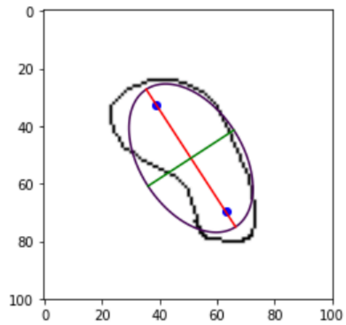


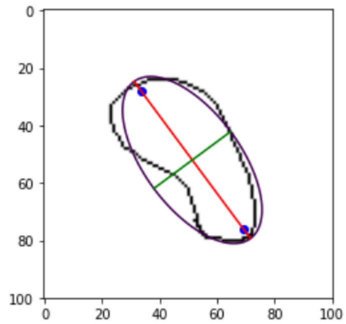
image1

From the above image , it is obviously that the major axis calculated by algorithm1 is shorter than the one calculated by algorithm2, and the longer one is the one we prefer to.
It seems like that there is little difference between two algorithms in the above instance.
But apparently, the accuracy of the second one is greater than the first one!
There is another example to prove that the above statement is clearly true.

```
Algorithm 1
=============================
[51.09495952 50.90355191] 28.526866769857435 17.82952898729437
Focal points: ( 38.75876001352157 , 32.42964265790356 ),( 63.048343808493954 , 69.76027638213492 )
Major axis: ( -0.8381903976558998 , -0.5453777198212673 )
Minor axis: ( 0.5453777198212673 , -0.8381903976558998 )
```



```
Algorithm 2
=============================
[52.00632911 51.37974684] 34.0224918892241 16.652403301354646
Focal points: ( 69.04820696805942 , 75.84019021915621 ),( 33.71128670282666 , 28.17246800869189 )
Major axis: ( 0.8033354975548971 , 0.5955267234711016 )
Minor axis: ( -0.5955267234711016 , 0.8033354975548971 )
```



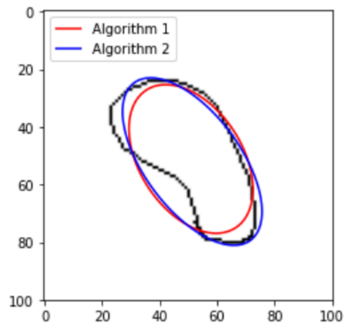


image2

In image2, the major axis draw by algorithm1 is too short ,which isn't what we want. Just as the result in image1 , the major axis of the eclipse we want is longer than what created by algorithm1.

On the other hand, the one drawn by algorithm2 is correspond to the longest line that connect two points in the original graph.

Hence, it is evidently that algorithm2 is more accurate.

5.

The amount of nonzero singular values equals to the rank of the matrix, but the magnitude of the nonzero singular values provide a measure of how close the matrix is to a matrix of lower rank, and it's the reason why numpy.linalg.lstsq return singular value.

Besides, singular values can also help us to find out eigenvalues, and we can use it to find out the other things that we can't find when we only have the rank.