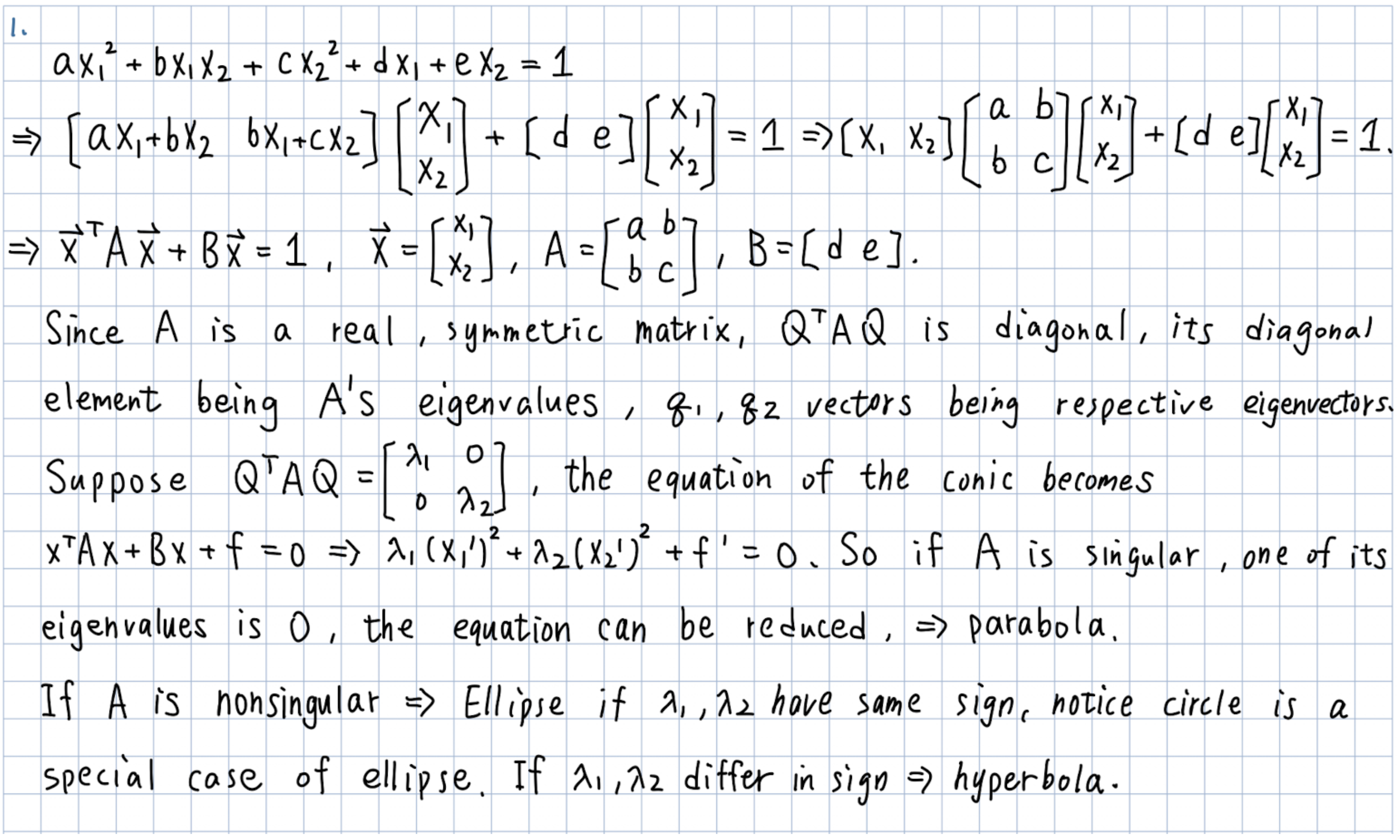
Rectifying your ellipse

學號: 109062129　　　　　　　　姓名: 林奕廷

1.



1.

$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2 = 1$

$\Rightarrow \begin{bmatrix} ax_1+bx_2 & bx_1+cx_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} d & e \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1 \Rightarrow \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} d & e \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1.$

$\Rightarrow \vec{x}^T A \vec{x} + B\vec{x} = 1$, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$, $B = \begin{bmatrix} d & e \end{bmatrix}$.

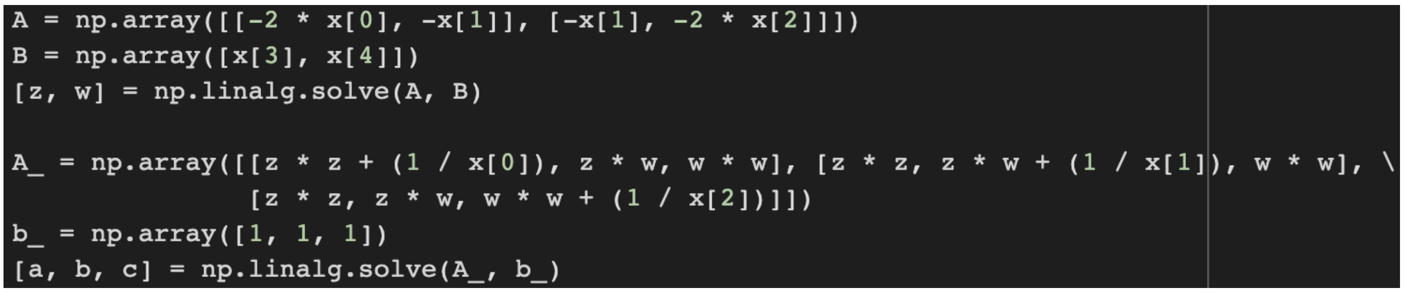
Since A is a real, symmetric matrix, Q^TAQ is diagonal, its diagonal element being A's eigenvalues, q_1, q_2 vectors being respective eigenvectors.

Suppose $Q^TAQ = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$, the equation of the conic becomes $x^TAx + Bx + f = 0 \Rightarrow \lambda_1(x_1')^2 + \lambda_2(x_2')^2 + f' = 0$. So if A is singular, one of its eigenvalues is 0, the equation can be reduced, $\Rightarrow$ parabola.

If A is nonsingular $\Rightarrow$ Ellipse if λ_1, λ_2 have same sign, notice circle is a special case of ellipse. If λ_1, λ_2 differ in sign $\Rightarrow$ hyperbola.

2.

def translation(x) :



```
A = np.array([[-2 * x[0], -x[1]], [-x[1], -2 * x[2]]])
B = np.array([x[3], x[4]])
[z, w] = np.linalg.solve(A, B)

A_ = np.array([[z * z + (1 / x[0]), z * w, w * w], [z * z, z * w + (1 / x[1]), w * w], \
               [z * z, z * w, w * w + (1 / x[2])]])
b_ = np.array([1, 1, 1])
[a, b, c] = np.linalg.solve(A_, b_)
```

A and A_ matrix are as provided in the spec pdf. By solving them, general equation can be transformed to semi-standard model: a(x-z)(x-z) + b(x-z)(y-w) + c(y-w)(y-w) = 1. Finally this function returns the coefficients.

def rotation(xp) :

```
S = np.array([[xp[0], xp[1] / 2], [xp[1] / 2, xp[2]]])
w, Q = np.linalg.eig(S)
U = np.transpose(Q)

alpha = math.sqrt(1 / w[1])
beta = math.sqrt(1 / w[0])
```

The semi-standard model can be written as x^T*A*x, according to textbook. Let S matrix be A. Then compute the eigenvalues and eigenvectors. Let U be a matrix whose columns are eigenvectors of S. w[0]^2*x^2 + w[1]^2*y^2 = 1 = (x/alpha)^2 + (y/beta)^2. The value of alpha and beta can be found.

3.

def PCA(points) :

```
# TODO: find the rotation matrix U and replace it with the current values
w, U = np.linalg.eig(S)
#U = np.array([[-0.73960154, -0.67304499], [ 0.67304499, -0.73960154]])
YR = np.dot(U, Y.T)
```

Rotation matrix U can be found by forming a matrix whose columns are eigenvectors of S.

def standard_to_general(means, U, alpha, beta) :

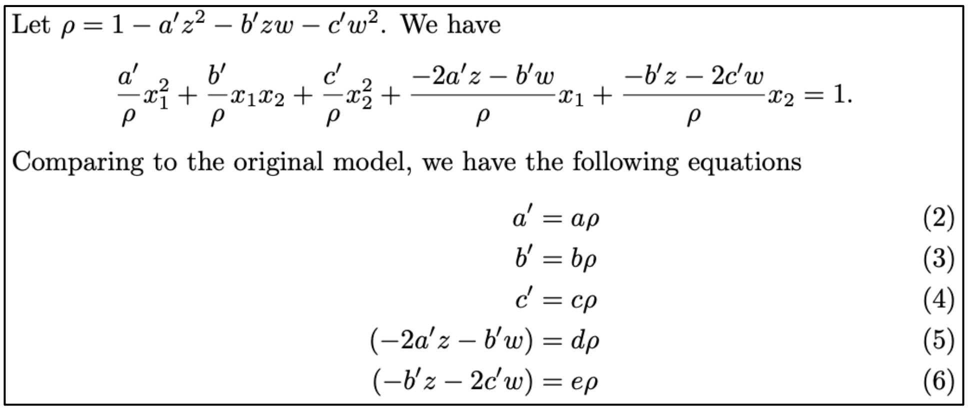
```
A = np.array([[pow(alpha, -2), 0], [0, pow(beta, -2)]])
A = np.dot(np.dot(np.transpose(U), A), U)
```

$$A = \begin{bmatrix} \alpha^{-2} & 0 \\ 0 & \beta^{-2} \end{bmatrix}$$

Then A = U^T*A*U.

```
a = A[0][0]
b = A[0][1] * 2
c = A[1][1]
[x_bar, y_bar] = means
d = -2 * a * x_bar - b * y_bar
e = -b * x_bar - 2 * c * y_bar
p = 1 - a * x_bar * x_bar - b * x_bar * y_bar - c * y_bar * y_bar
```

The above code finds the coef a, b, c, d and e of the general equation. The equations are in the spec pdf:



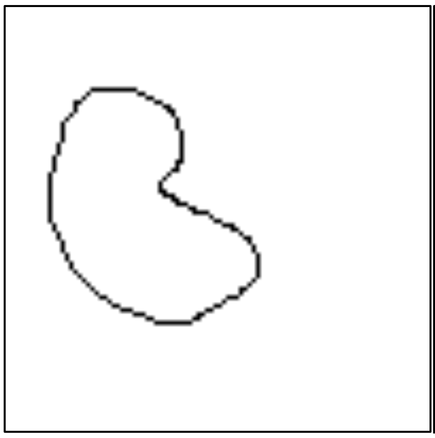
Let $\rho = 1 - a'z^2 - b'zw - c'w^2$. We have

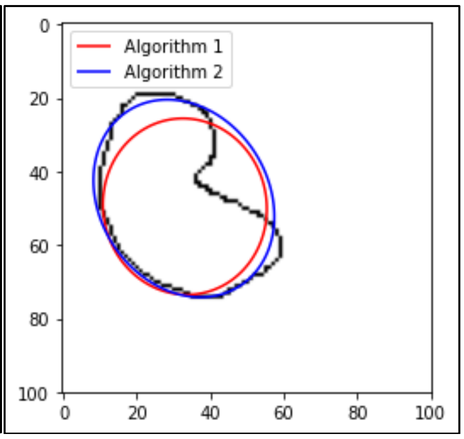
$$\frac{a'}{\rho}x_1^2 + \frac{b'}{\rho}x_1x_2 + \frac{c'}{\rho}x_2^2 + \frac{-2a'z - b'w}{\rho}x_1 + \frac{-b'z - 2c'w}{\rho}x_2 = 1.$$

Comparing to the original model, we have the following equations

$$a' = a\rho \tag{2}$$
$$b' = b\rho \tag{3}$$
$$c' = c\rho \tag{4}$$
$$(-2a'z - b'w) = d\rho \tag{5}$$
$$(-b'z - 2c'w) = e\rho \tag{6}$$

4.





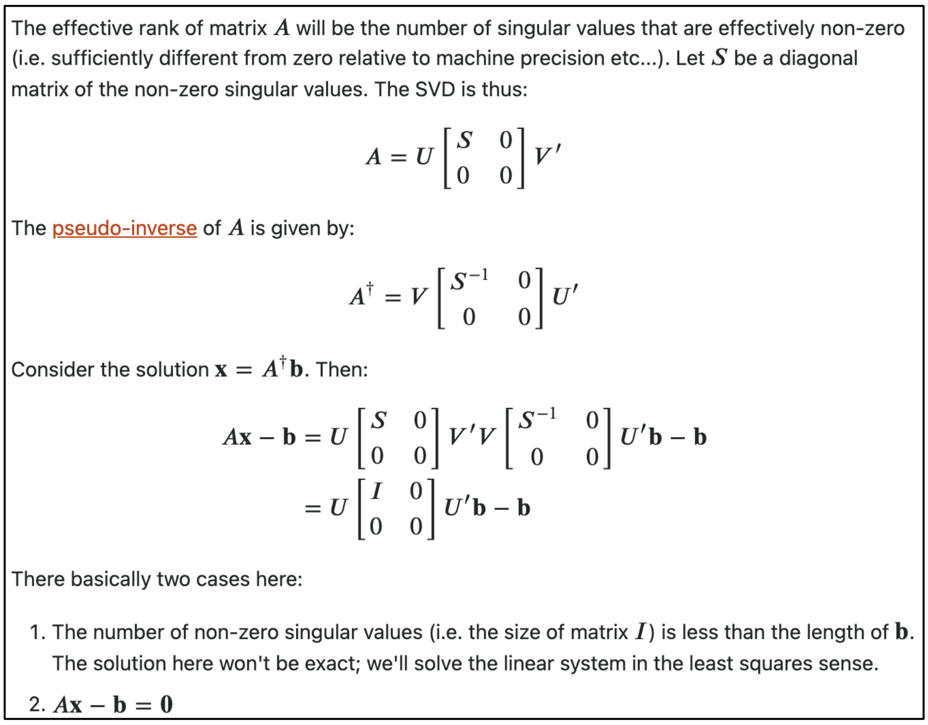
Take this drawing for example. It is clear that algorithm 2 is more accurate. Algorithm 1 takes the least square approach, while algorithm 2 uses PCA. Firstly, PCA takes means of all points’ x and y value as center point of the ellipse, translate points to the center and rotate them to fit the center. This method should be accurate if the given points are discrete. The least square method, on the other hand, could result in a ellipse that is much smaller or bigger in size than anticipated, as the result shown. It is because that it only takes the sum of distance between points of drawing and points of resulting ellipse into consideration, and the center is not guaranteed to be “centered” as well.

5.

Explain the relation between rank and singular value(s):

The rank of a square matrix represents the number of eigenvectors, which is the same as the number of nonzero singular values(singular value = sqrt(eigenvalue)).

Suppose the system Ax = b is undertermined, lstsq solves x by obtaining the SVD form of A.



The effective rank of matrix A will be the number of singular values that are effectively non-zero (i.e. sufficiently different from zero relative to machine precision etc...). Let S be a diagonal matrix of the non-zero singular values. The SVD is thus:

$$A = U \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} V'$$

The pseudo-inverse of A is given by:

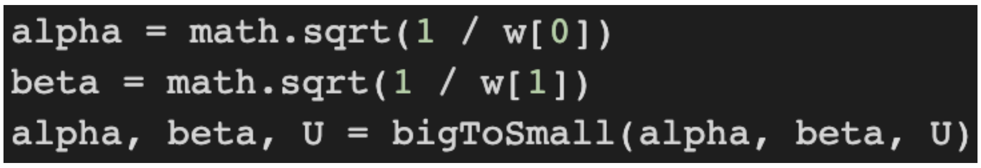
$$A^{\dagger} = V \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} U'$$

Consider the solution $\mathbf{x} = A^{\dagger}\mathbf{b}$. Then:

$$\begin{aligned} A\mathbf{x} - \mathbf{b} &= U \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} V'V \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} U'\mathbf{b} - \mathbf{b} \\ &= U \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U'\mathbf{b} - \mathbf{b} \end{aligned}$$

There basically two cases here:

1. The number of non-zero singular values (i.e. the size of matrix I) is less than the length of $\mathbf{b}$. The solution here won't be exact; we'll solve the linear system in the least squares sense.
2. $A\mathbf{x} - \mathbf{b} = \mathbf{0}$

(Source: https://stats.stackexchange.com/questions/240573/how-does-numpy-solve-least-squares-for-underdetermined-systems)
6.


```
alpha = math.sqrt(1 / w[0])
beta = math.sqrt(1 / w[1])
alpha, beta, U = bigToSmall(alpha, beta, U)
```

(in def rotation(xp))
原本 PCA 遇到的旋轉軸問題在更新助教提供的 bigToSmall function 有解決，但是 algorithm 1 在 ellipse-2.png 仍然會有問題。後來我將 bigToSmall 移到 algorithm 1 的 rotation function 前並使用，有得到解決。會有這樣的問題是因為原本的 bigToSmall 沒有考慮到 alpha < beta 的狀況，導致 U matrix 在 alpha < beta 時會出錯，造成長短軸歪掉。