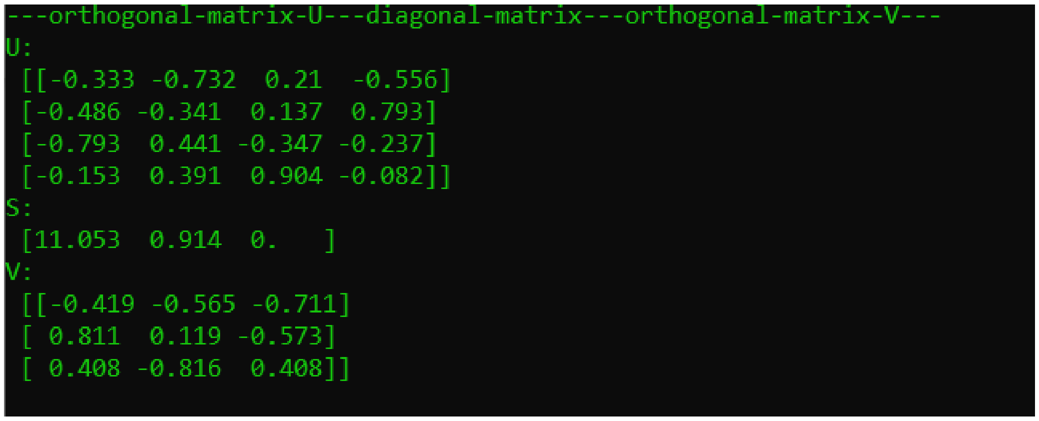
Question 1:

The program just simply uses the numpy library to calculate the SVD of the given matrix.

Here is the result from the q1.py:



```
---orthogonal-matrix-U---diagonal-matrix---orthogonal-matrix-V---
U:
 [[-0.333 -0.732  0.21   -0.556]
 [-0.486 -0.341  0.137  0.793]
 [-0.793  0.441 -0.347 -0.237]
 [-0.153  0.391  0.904 -0.082]]
S:
 [11.053  0.914  0.    ]
V:
 [[-0.419 -0.565 -0.711]
 [ 0.811  0.119 -0.573]
 [ 0.408 -0.816  0.408]]
```

Question 2:

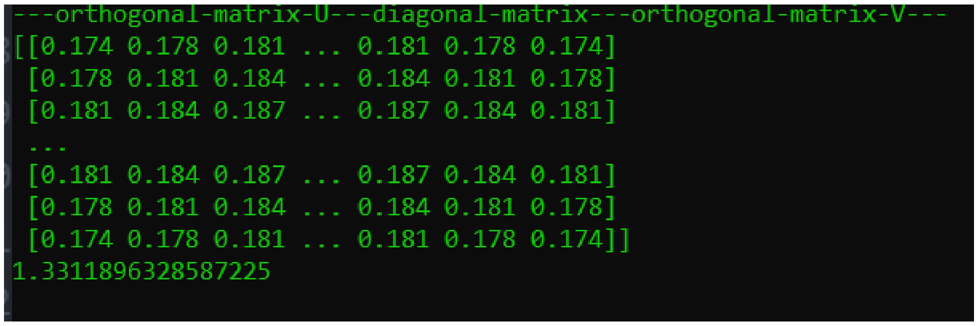
The program first creates a matrix 1401 x 1401 to record the value of z where $z = \sqrt{1 - x^2 - y^2}\ where\ x_i = y_i = 0.07 + \Delta(i-1), \Delta = 0.001\ and\ 1 \leq i,j \leq 1401.$

Then the program calculate the SVD of the matrix, then apply the following algorithm to calculate the best rank(2) matrix:

$$A_2 = \sum_i^2 S^i\, U[i]Vt[i]$$

Then the program calculates the: $||A - A_2|| \approx 1.33119$

Here is the result from the q2.py:



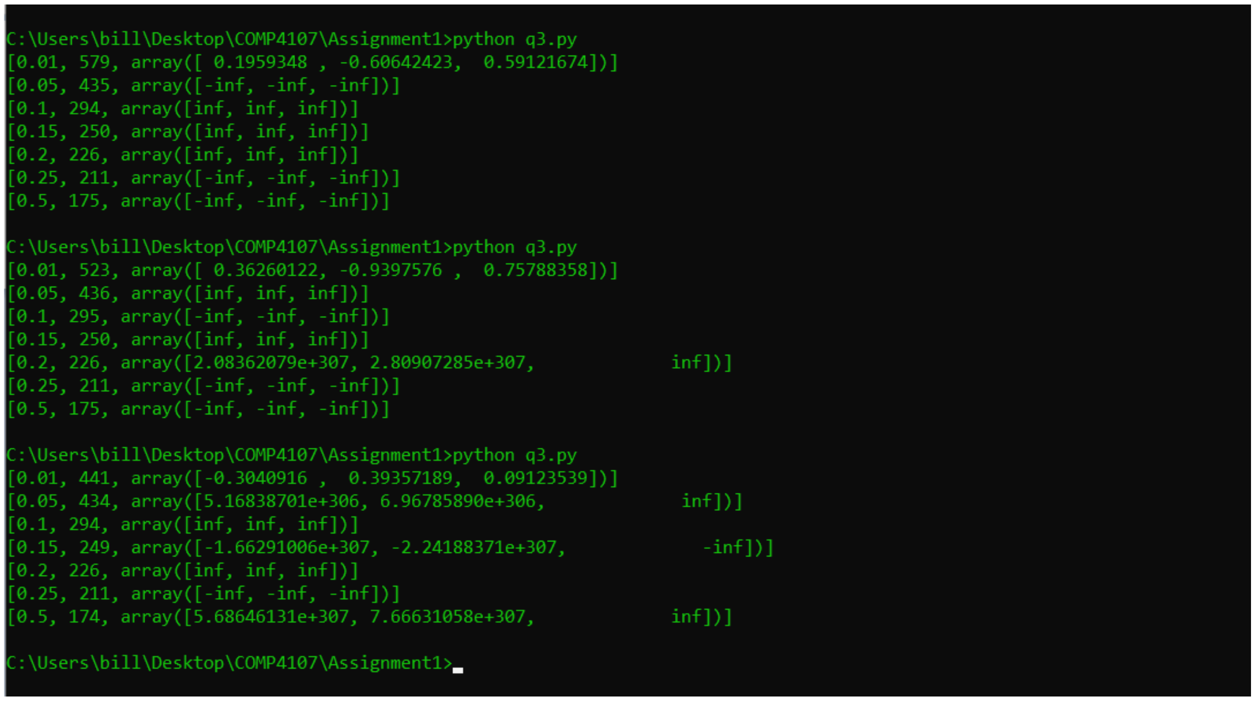
```
---orthogonal-matrix-U---diagonal-matrix---orthogonal-matrix-V---
[[0.174 0.178 0.181 ... 0.181 0.178 0.174]
 [0.178 0.181 0.184 ... 0.184 0.181 0.178]
 [0.181 0.184 0.187 ... 0.187 0.184 0.181]
 ...
 [0.181 0.184 0.187 ... 0.187 0.184 0.181]
 [0.178 0.181 0.184 ... 0.184 0.181 0.178]
 [0.174 0.178 0.181 ... 0.181 0.178 0.174]]
1.3311896328587225
```

Question 3:

To approach this problem, I randomly choose a vector, x, in R^3 in range of (-2,2). Then using the General Gradient Descent to compute x' :

$$x' = x - \varepsilon\Delta_x f(x) = x - A^T(Ax - b) = A^T Ax - A^T b$$

Here is the output of q3.py:



```
C:\Users\bill\Desktop\COMP4107\Assignment1>python q3.py
[0.01, 579, array([ 0.1959348 , -0.60642423,  0.59121674])]
[0.05, 435, array([-inf, -inf, -inf])]
[0.1, 294, array([inf, inf, inf])]
[0.15, 250, array([inf, inf, inf])]
[0.2, 226, array([inf, inf, inf])]
[0.25, 211, array([-inf, -inf, -inf])]
[0.5, 175, array([-inf, -inf, -inf])]

C:\Users\bill\Desktop\COMP4107\Assignment1>python q3.py
[0.01, 523, array([ 0.36260122, -0.9397576 ,  0.75788358])]
[0.05, 436, array([inf, inf, inf])]
[0.1, 295, array([-inf, -inf, -inf])]
[0.15, 250, array([inf, inf, inf])]
[0.2, 226, array([2.08362079e+307, 2.80907285e+307,             inf])]
[0.25, 211, array([-inf, -inf, -inf])]
[0.5, 175, array([-inf, -inf, -inf])]

C:\Users\bill\Desktop\COMP4107\Assignment1>python q3.py
[0.01, 441, array([-0.3040916 ,  0.39357189,  0.09123539])]
[0.05, 434, array([5.16838701e+306, 6.96785890e+306,             inf])]
[0.1, 294, array([inf, inf, inf])]
[0.15, 249, array([-1.66291006e+307, -2.24188371e+307,            -inf])]
[0.2, 226, array([inf, inf, inf])]
[0.25, 211, array([-inf, -inf, -inf])]
[0.5, 174, array([5.68646131e+307, 7.66631058e+307,             inf])]

C:\Users\bill\Desktop\COMP4107\Assignment1>
```

Then, I applied the gradient descent method to find the min $||Ax - b||$. After running the program several times, I recognize that the value of learning rate is important. Because if the learning rate is chosen incorrectly, then the goal could not be approached as the result of the step of moving is incorrect. The moving step could be too small or too big, that lead to unexpected results.

For example: if the rate of learning is small, then the change of misleading to the goal is small. So that at the learning rate 0.01, we likely to get the valid vector x to find the min $||Ax - b||$. But since the learning rate is bigger, the misleading usually happened as the result, the vector x are mostly defined as a positive or negative infinite vector.

Through the question, I have learnt that to be aware of choosing a suitable learning rate and how is it important when applying it to learn something.

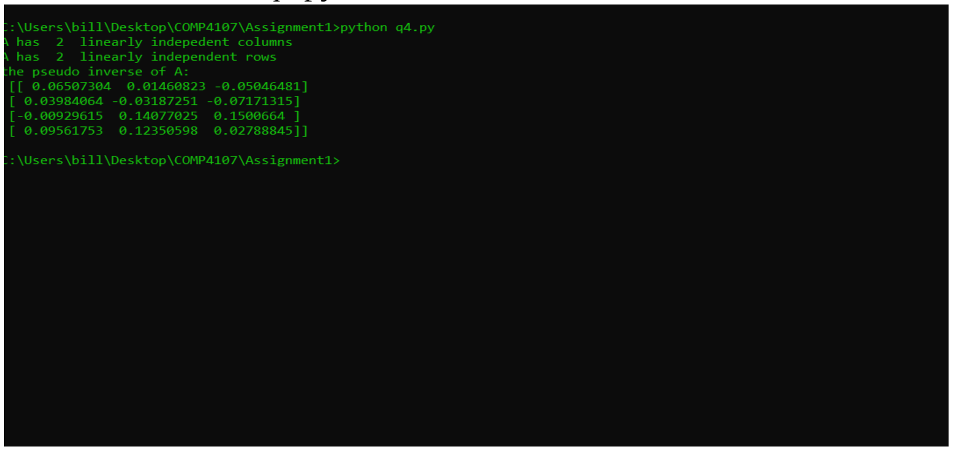
Question 4:

To figure the number of linear independent columns in matrix A. I calculated the rank of A, because the number of independent columns and the number of independent rows in a matrix is its rank. The result is 2.

To make sure the result above is correct. I calculated the rank of the transpose of A and the result is the same which is 2.

Since the matrix is not a square, therefore, I can’t calculate the inverse of matrix A, however, I can calculate the pseudo inverse of the matrix A using the numpy library.

Here is the result of q4.py:



```
:\Users\bill\Desktop\COMP4107\Assignment1>python q4.py
 has  2  linearly indepedent columns
 has  2  linearly independent rows
he pseudo inverse of A:
[[ 0.06507304  0.01460823 -0.05046481]
 [ 0.03984064 -0.03187251 -0.07171315]
 [-0.00929615  0.14077025  0.1500664 ]
 [ 0.09561753  0.12350598  0.02788845]]

:\Users\bill\Desktop\COMP4107\Assignment1>
```

Question 5:

The program just follows the instruction of the journal articles.

First the program reads the input and store the input of one digit into a matrix $m^2\ x\ n$, where n is the amount of data for one single digit and m is the number of columns of one image (all columns are stacked to form up m^2 x 1). All of the data will be store in another matrix such that:

A = 10 x ($m^2 x\ n$). Hence, A[0] will return the matrix $m^2\ x\ n$ of the digit 0.

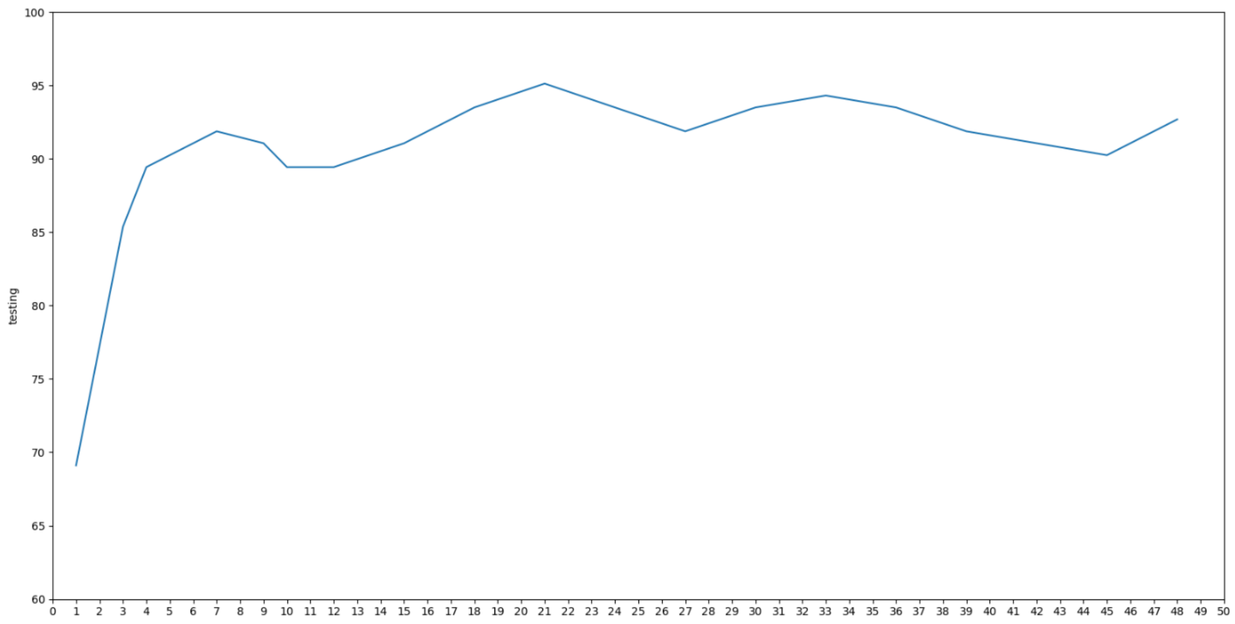
Then the program randomly generates two matrixes once for training and once for testing purpose. The training sample takes randomly 567 units and the testing sample takes randomly 123 units (123 test cases).

The program next calculate & stores the sub-result $U_k U_k^T$, k from range (0, …, 50), of SVD in a table which will be used to calculate the given residual: $||(I - U_k U_k^T)z||_2$ where z is a random digit.

Finally, the program will find the smallest element $||(I - U_k U_k^T)z||_2$ of a specific digit z to compare with the digit in the test case. If both are equals, then the number of correctness increased by one.

Finally, the program output the results.

Here is the output of the q5.py program



Question 6:

The program firstly read the data and store the table into a table has form $A_{i,j} = rating$, where i is the user id and j is the movie id.

Then the program calculates the average of each users, which will be used for predicted data and calculating MAE.

The program randomly chooses train and test data based on the test data ratio x (0.2; 0.5; 0.8) and stores them into a table. As the lecture notes, each train data in the matrix (the train matrix) will be added its (the user average) average for the prediction purpose.

Then, the SVD computation process of the training data will be applied fold_in process from basis 600 to 943 with rank 14. For each basis, the program calculates the table $U_K\sqrt{S_kT}.\sqrt{S_k}VT_k + R_i(average)$ to compute MAE. In particular, MAE will be calculated as

$\frac{|P-(U_K\sqrt{S_kT}\,(i).\sqrt{S_k}VT_k(j)+R_i)|}{N}$, N is the total number of time calculation for one basis test.

Along with that the time of computing SVD and predictions will be recorded.

Below are two screenshots of the program’s outputs.

Figure 1:

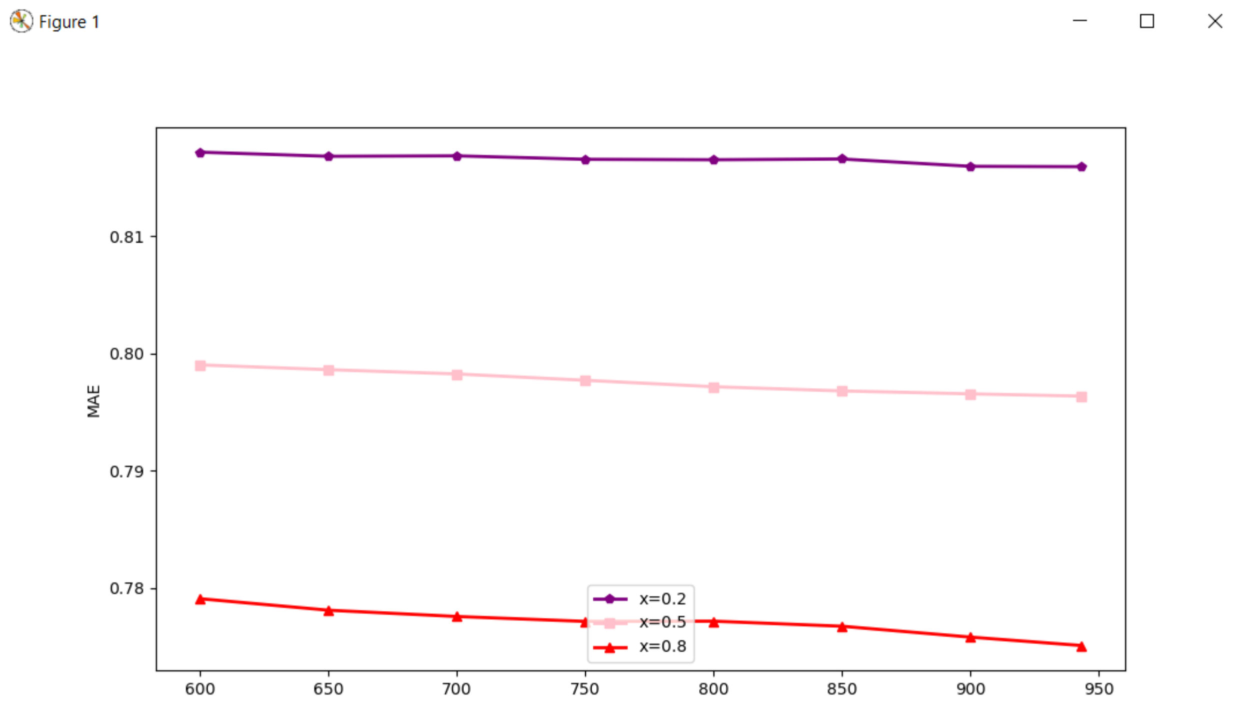
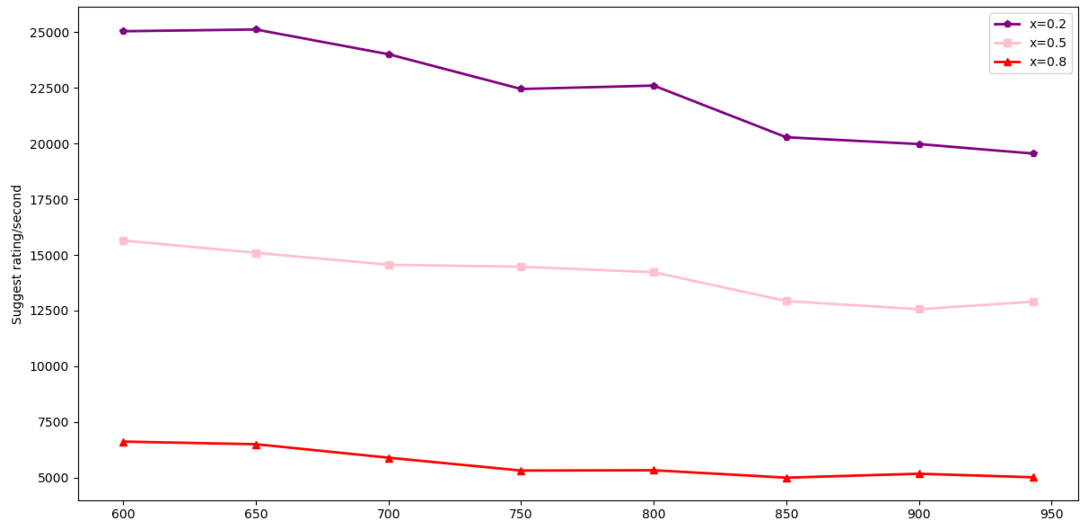


Figure 1
MAE
0.81
0.80
0.79
0.78
600
650
700
750
800
850
900
950
x=0.2
x=0.5
x=0.8

Figure 2:


Suggest rating/second
25000
22500
20000
17500
15000
12500
10000
7500
5000
600
650
700
750
800
850
900
950
x=0.2
x=0.5
x=0.8