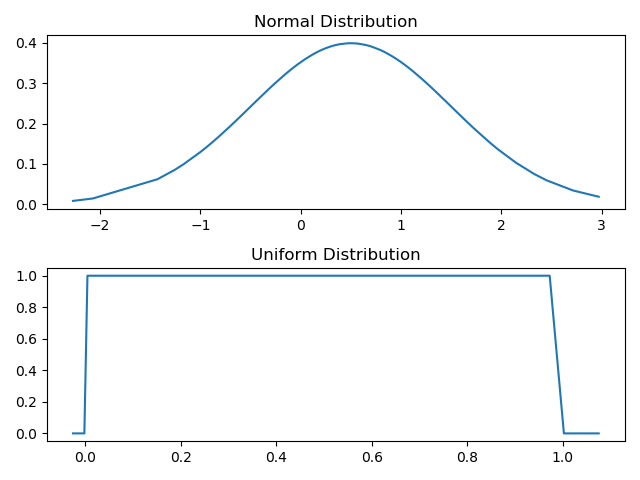
Stochastic gradient descent

SGD uses iterative method to find the objective solution for current input.

For example, we create a normal distribution sequences and a uniform distribution for Task1 and Task2:

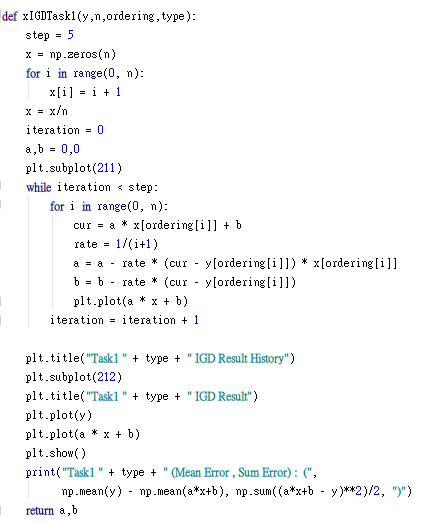


and then we sort both sequence:

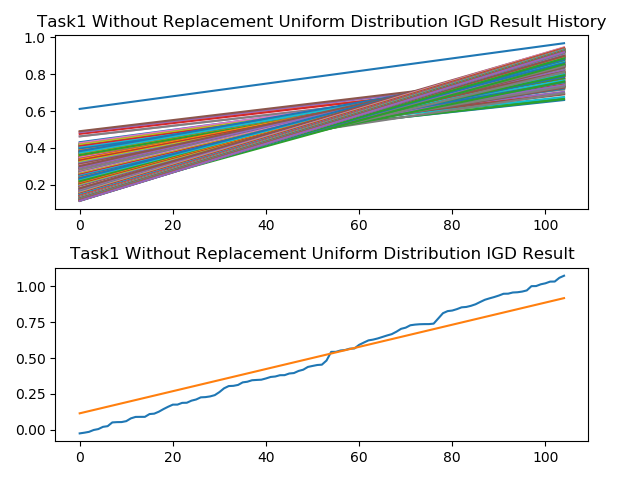
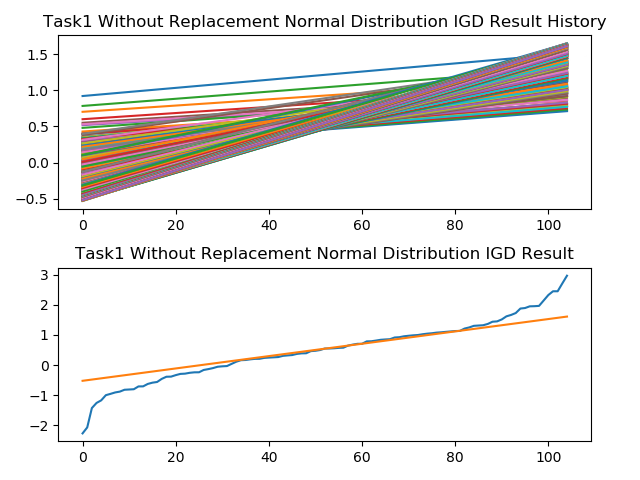
Our mission is to find a minimum 1D solution for both sequences.

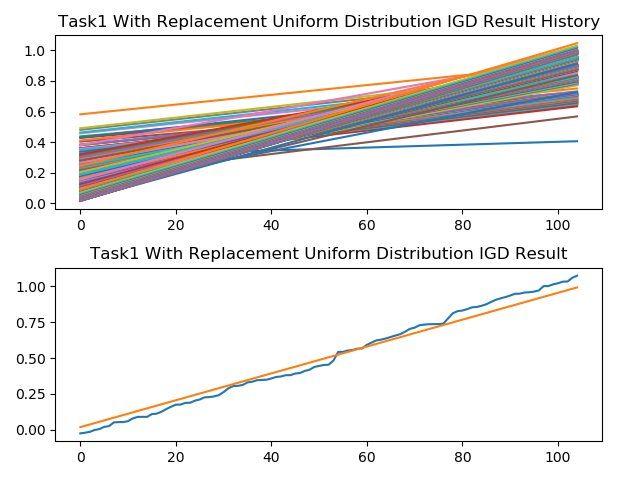
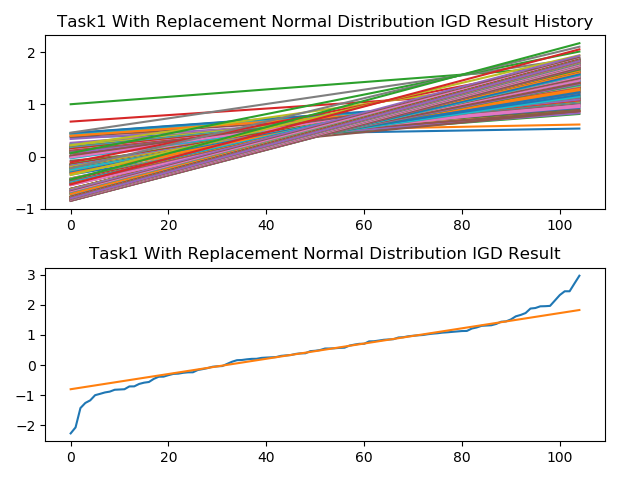
**Task1:**

In task1, our learning rate is r=1/(k+1)×1/n, where k is iteration time and n is sequence size. It means that when k is small, the step length is higher. Therefore, based on SGD, we update our results every iteration.



Result:



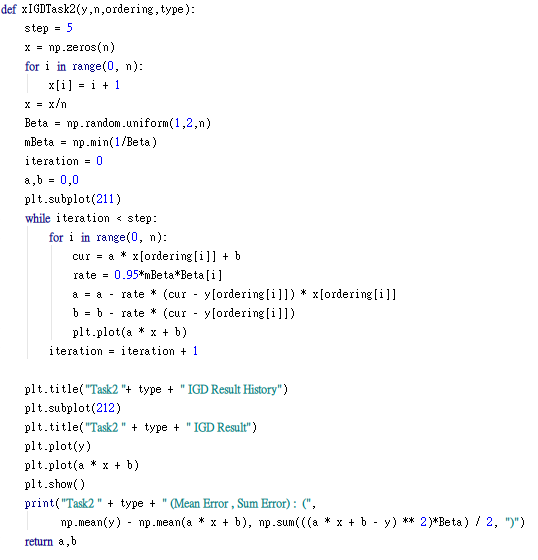


**Task2:**

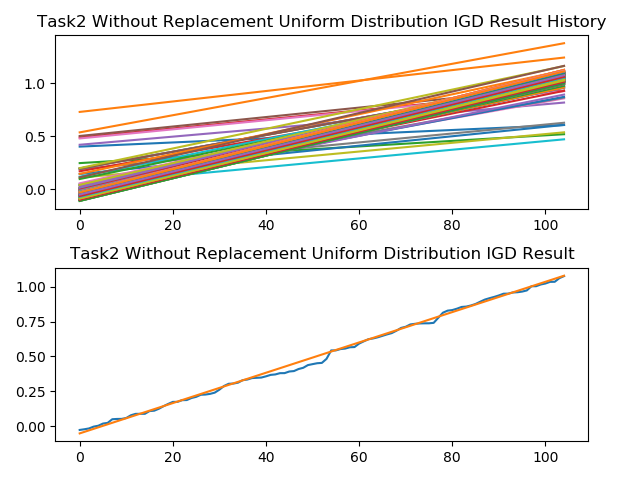
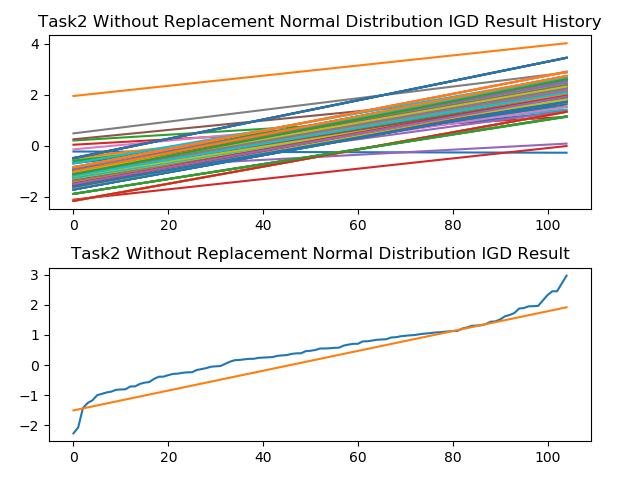
Similar to Task1, but we change the learning rate:

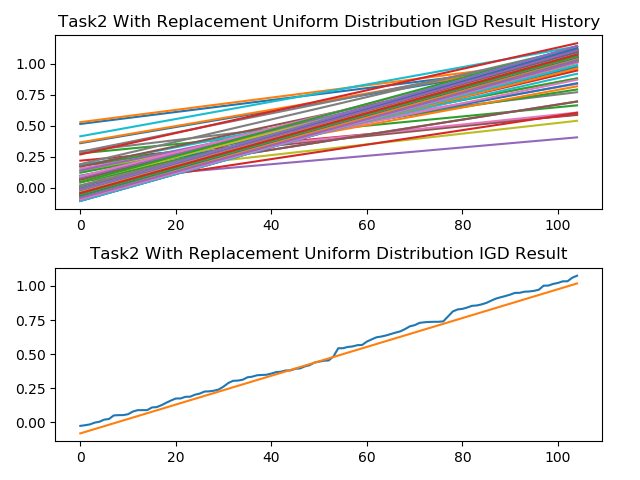
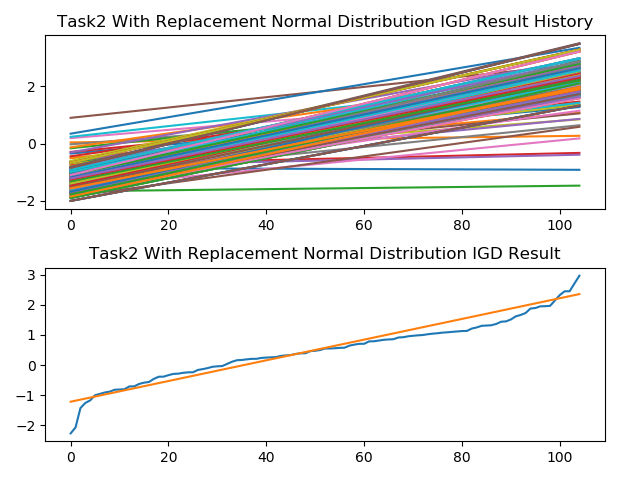
r=0.95×min⁡ (Beta^(-1) )×Beta×1/n,

where Beta is constrained to [1 2]. It means that the learning rate would be fixed in a range [0.95/2n 0.95/n]).



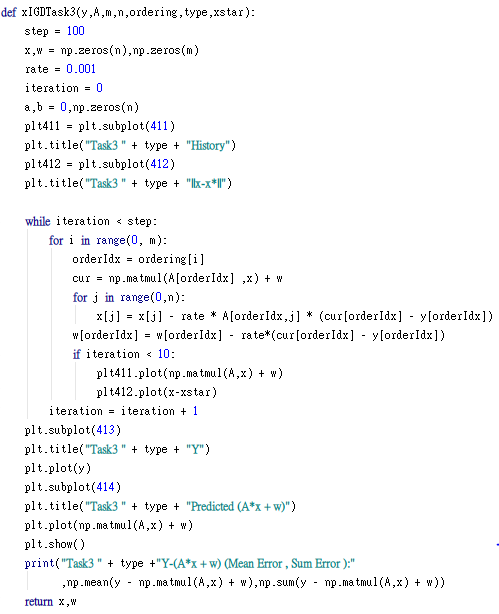
Result:



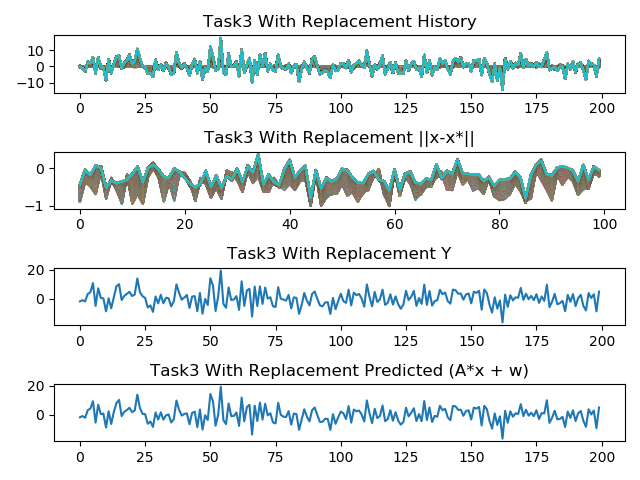
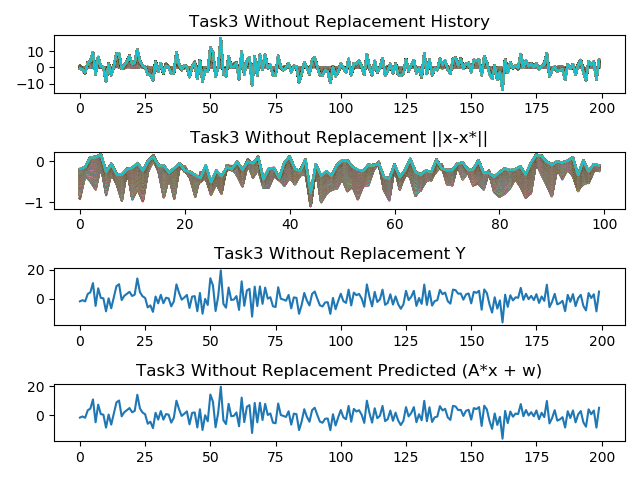


**Task3 :**

Different from Task1 and Task2, Task3 need to solve the high level dimension problem:



Result:



Conclude:

We can observe that if we increase the iteration times, the without replacement ordering would get the better solution compared with the with replacement ordering. It is because that the without replacement would traverse more training point.

Tool : pycharm or colab

Lib : numpy, scipy, matplotlib

Copy and Paste main.py code to **colab** or **pycharm**

Execution main.py in python environment like pycharm