# Code Used:

As per requested on the assignment sheet, the following are the changes for assign\_weights.m file.



Code Snapshot : assign\_weights.m Addition

In order to duplicate run file for 100 times, a master loop with counter control is added to the file as well as recorders for recording results of trip cost and iteration cost: (Modifications has been highlighted with cyan color.)



Code Snapshot : hopfield\_TSP modifications

Compute\_trip\_cost has been modified so that we can keep track of result validity. We do that by checking if we have visited 10 cities.



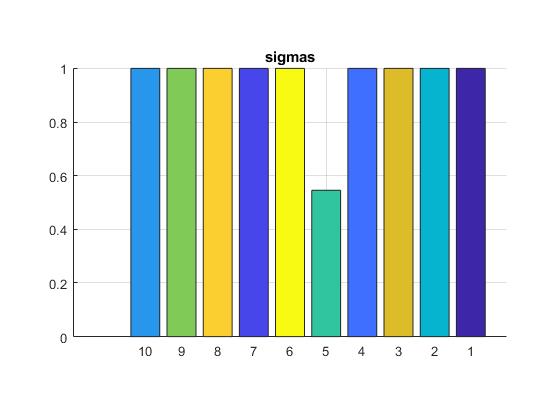
Code Snapshot : compute\_trip\_cost additions

# Try Excluding u\_dot(X,ix) line:

I would say that the code:



Was mandatory for the net to work smoothly. Without the code, the network behaves wabbly. With the u\_dot code, the network is more decided, with a relatively definitive answer. **This code is used for making sure the result is an integral.** However, I do notice that due to some inherent bug, the network thinks it finished when the last city didn’t grow all the way up to 1. Below on the left is result run with the line of the code, and on the right is the code run without the line of the code.

A picture containing text

Description automatically generated

Figure : run result comparison

I also notice the speed of convergence is drastically different too. With the integral filtering, the code converges faster than without this line of code.

# Statistics for performance of your neural-net computation

After 100 run funnies:

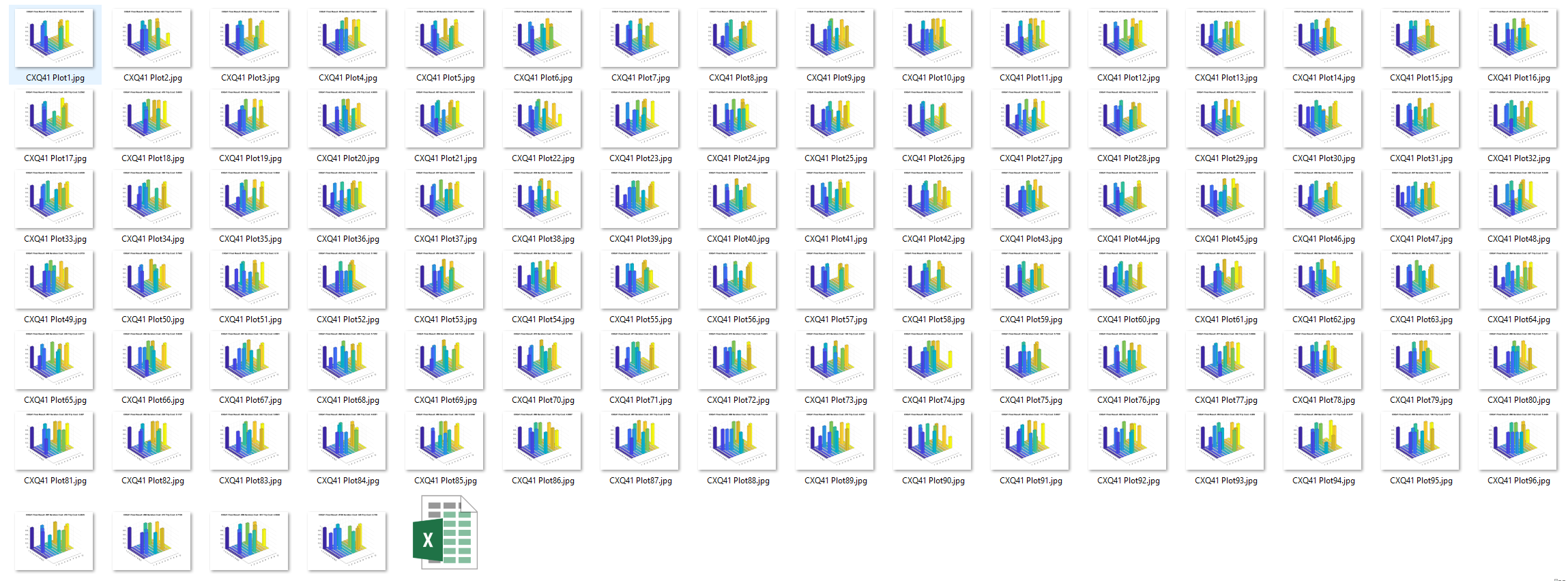


Figure : 4 Hours of Computation!

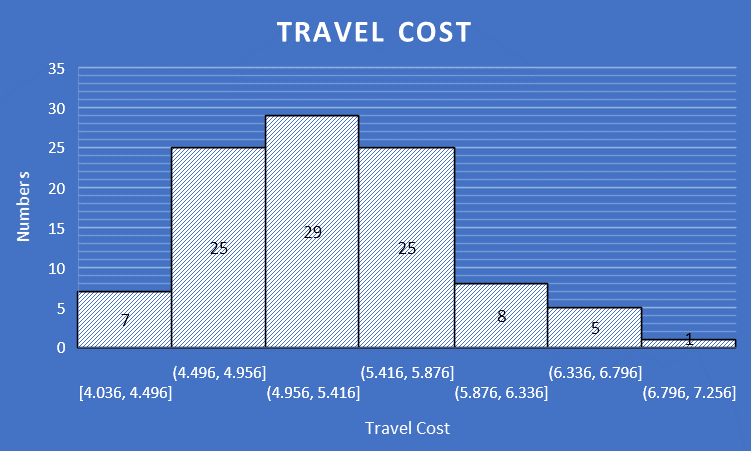


Figure : Travel Cost histogram

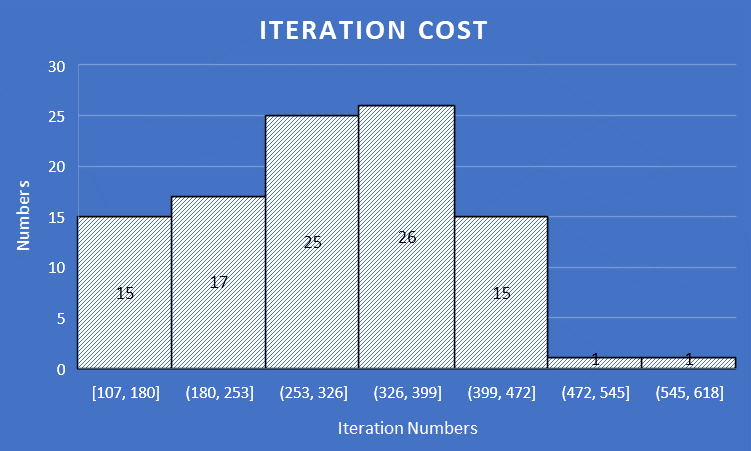


Figure : Iteration Cost histogram



Table : Calcuated Statistics Given Parameter vs provided

As we can observed, since our std value is smaller than provided, I can say my data are precise (stable), comparing to the mean value I get and min and max, I would not call my data too accurate considering the fact the test run was only 100. I do want to point out that due to this is a neural network, sometimes we will get outliers just like students have different performance level of academia.

# Changing Network Parameter

**The following result are run for 10 times only due to time constraints**

I have tried turn down the D value since it will not affect decision making that much, following are the result:



Table : Modified D = 0 Result Comparison

As you can see the travel cost have increased for about 1 which is bad, and since data size are reduced, standard deviation value has been decreased. What’s more, for iteration cost, we can see that the net takes fewer run to converge to a final result. So if we are looking for pure speed, tuning down D is a good option

Since I kind of like the current speed, I will try up tune the parameter to see what I can get out of.



Table : Modified A = 1000 Result Comparison

From above observation we can see that when we up tune the A value, our 10-run mean is similar to the 100 ones with A = 500; but our iteration cost has been decreased. In fact, our mean iteration cost has decreased about 50 times, which is about 20% or previous one, so I think based on current indication, up tuning A value can help.

For comparison, I have tuned down A value to 100, following are the observations:

It does not converge… So I guess 500 is somewhat optimal value…