In [52]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from sklearn.metrics import mean_squared_error
#1.read in data
data = pd.read_csv('x06Simple.csv')
k = 1
y = data.to_numpy()
#2 randomize the data
np.random.seed(0)
np.random.shuffle(y)

size = np.size(y[:,1])

#3 Select the first 2/3 of the data for training and the remaining for testing
x = math.ceil(size*(2/3))

Yan = np.array(y[0:x,[3]]) #2/3 train data
yanTest = np.array(y[x:,[3]]) #1/3 test data
#4. Standardize the data (except for the last column of course) using the training data
mattrain = np.array(y[0:x,0:2])
mattest = np.array(y[x:,0:2])

mean2 = np.mean(mattrain ,axis=0)
std2 = np.std(mattrain, axis=0,ddof=1)

mattrain = (mattrain-mean2)/std2
mattest = (mattest-mean2)/std2

size = np.shape(mattrain)[0]
size1 = np.shape(mattest)[0]

one = np.ones((size,1))
mattrain = np.concatenate((one, mattrain), axis=1)
one1 = np.ones((size1,1))
mattest = np.concatenate((one1, mattest), axis=1)


error = []
error_root = []
#5. Then for each testing sample
```

```
for j in range(np.size(mattest[:,1])):
    d = []
    for i in range(np.size(mattrain[:,1])):
        #(a) Compute the necessary distance matrices relative to the training data in order to compute a local m
        d.append(np.sum(np.square(mattest[j]-mattrain[i])))
    #create the diagnal arrea
    dsize = np.shape(d)[0]
    d = np.array(d)

    d1 = np.zeros((dsize,dsize))
    np.fill_diagonal(d1, np.exp((-1*d)/k**2))

    d = d1
    #(b) Evaluate the testing sample using the local model.
    theta = np.linalg.inv(mattrain.T @ (d@mattrain)) @ mattrain.T @ (d@Yan)

    #(c) Compute the squared error of the testing sample.
    y = mattest[j,:]@theta
    error.append(mean_squared_error(yanTest[j],y))
error = np.array(error)
#6. Computes the root mean squared error
error = np.sqrt(error)
error = np.average(error)

print(error)
```

```
121.8923540656184
```

In [ ]:

In [ ]: