Homework 8 Report

Jesse Harper

5/29/2018

Introduction 1.1:

This problem is an exercise in the use of hashing functions and data organization. The set of N randomly generated tuples are organized into numbered bins. The sum of the bins make up the entire domain and range of values generated. However the algorithm used to sort, which is described in the following sections, has a trouble sorting out at least one fringe case.

Models and Methods 1.2:

This program uses matlab's built in random number generator, set to default, to create two sets of randomly generated x and y coordinates. The Program sets the number of points based on the value N and the range is constrained by x and y max. These coordinates are then organized using the hashing function described by **equation 1.2.1**. The program keeps track of the bins, and which coordinates that contain them, using a cell array. Once all the coordinates are organized, a plot is printed which shows both the location of each coordinate as well as the location of the average point in each bin.

$$binNum_k = \left(\left[\frac{x_k}{\Delta x}\right] - 1\right) N_y + \left[\frac{y_{max} - y_k}{\Delta y}\right] \qquad \textbf{eqn. 1.2.1}$$

Calculations and Results 1.3:

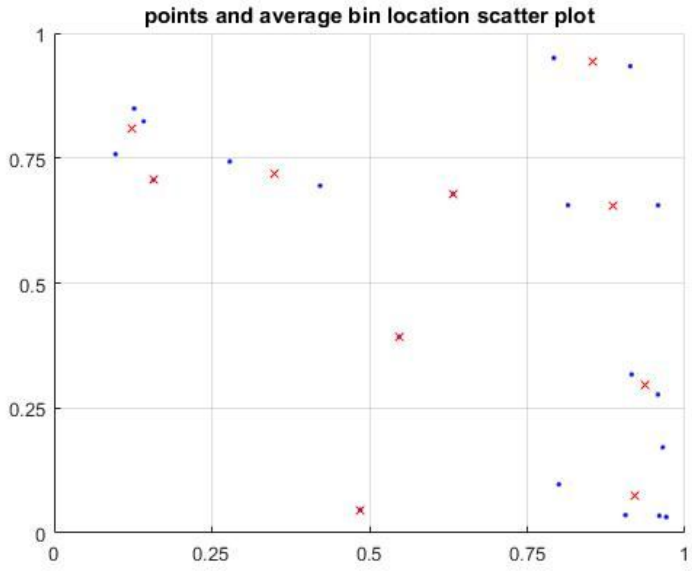

Figure 1.3.1: scatter plot for N = 20

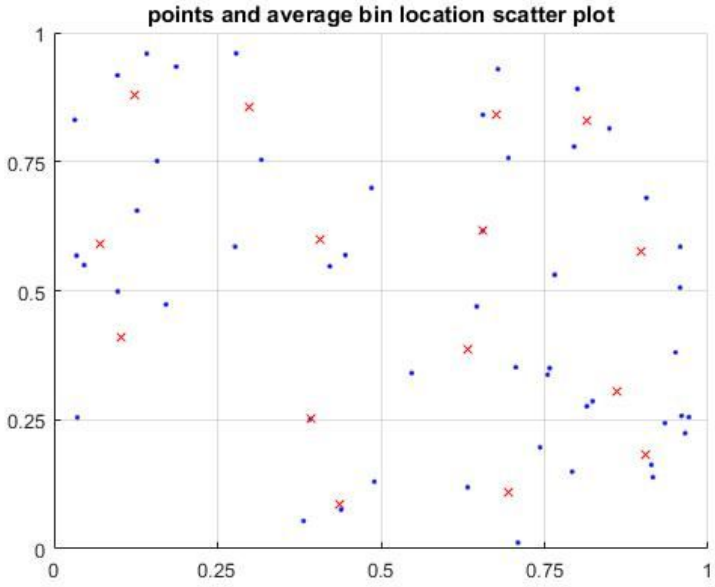

Figure 1.3.2: scatter plot for N = 50

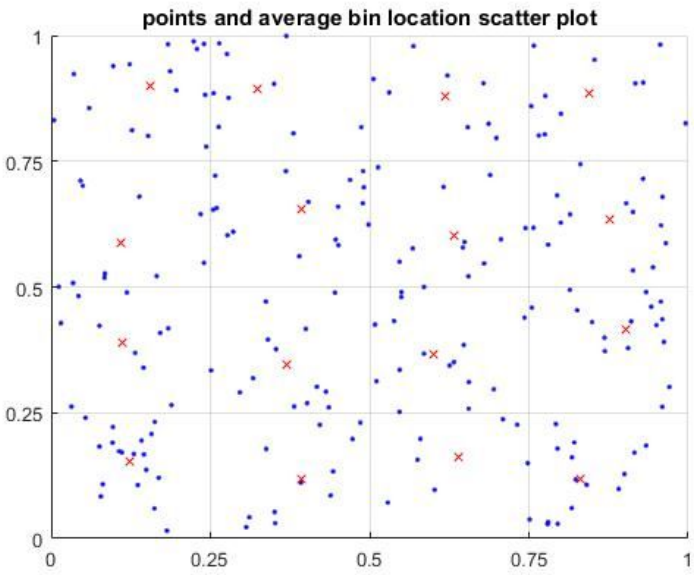

Figure 1.3.3: scatter plot for N = 200

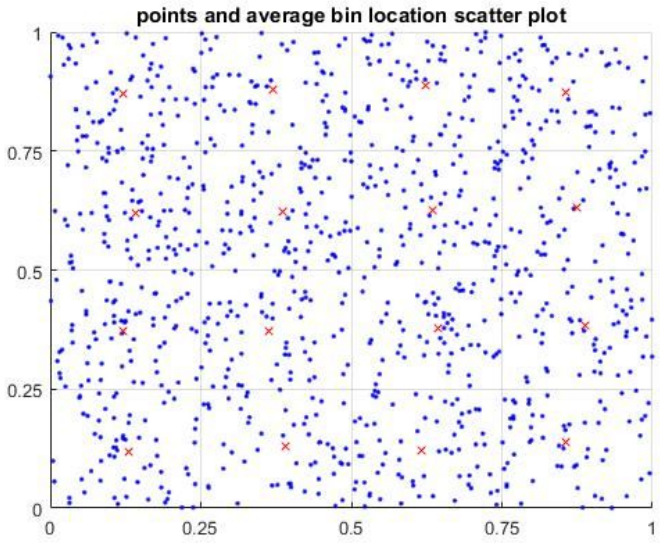

Figure 1.3.4: scatter plot for N = 1000

The following is a sample of the console output for the program running with N value equal to 20 and h equal to 0.25.

```
Bin 1:  3 6 16
Bin 2:  11
Bin 3:  []
Bin 4:  []
Bin 5:  []
Bin 6:  7 17
Bin 7:  []
Bin 8:  14
Bin 9:  []
Bin 10:  5
Bin 11:  8
Bin 12:  []
Bin 13:  4 19
Bin 14:  1 9
Bin 15:  13 18
Bin 16:  2 10 12 15 20
```

Discussion 1.4:

This program shows some interesting effects when N is increased to infinity. **Figure 1.3.4** shows that as N becomes increasingly large, the location of the average coordinate location tends closer to the center of the bins. This is a kind of analog to the idea that the average of large set of unweighted randomly distributed 1 dimensional array of values will also tend to the middle of the range of values. The instructed approach to spatial hashing has a problem handling the fringe case that x_k = 0 and y_k = y_max. The use of equation **1.2.1** yields bin numbers that are negative which cause errors in the program. The error is that array indexes are numbered by bin number which can not be negative. This problem could be resolved if this particular case is pre tested before use of the hash function. If h is the maximum interaction radius and we are only concerned with particles in bin 1, then we need 3 bins to identify all particles within the zone of influence of the particles in bin 1.

Introduction 2.1:

This problem required a construction of a function which could iteratively use Newton's method to solve for function zeros. This function was written, and implemented in a driver function which found the zeros of a function f(x). The function f(x) is described in section 2.2. Likewise the results of the program are given in section 2.3.

Models and Methods 2.2:

This program utilizes Newton's algorithm for iteratively solving the zero of function. A function, named *Newton*, was created to accepted a function handle, first guess, error, and max iteration, and returns the domain input for function zero as well as the number of iterations required to obtain it. **Equation 2.2.1** shows the formula which describes Newton's method.

$$X_c = X_0 - \frac{F(x)}{F'(x)} \qquad \textbf{Eqn. 2.2.1}$$

The helper function, *Newton*, has its own local defined function which calculates the derivative of F(x) at a location x. This function, f_prime, uses **equation 2.2.2** to calculate the local derivative.

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \qquad \textbf{Eqn. 2.2.2}$$

here , h is set to be 1e-6 as specified by the problem statement.

The function which this program will be working on is given below

$$f(x) = 816X^3 - 3835x^2 + 6000x - 3125 \qquad \textbf{Eqn 2.2.3}$$

Calculations and Results 2.3:

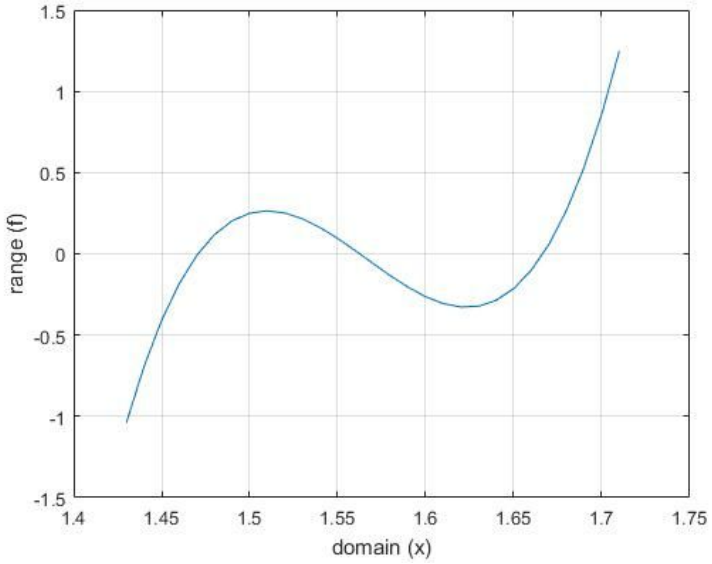

Figure 2.3.1: a plot of f with x in the set [1.43, 1.71]

The following is a sample of console output with delta = 1e-3 and max iterations set to 15.

```
x0 = 1.43, evals =  3, xc = 1.470523
 x0 = 1.44, evals =  3, xc = 1.470574
 x0 = 1.45, evals =  3, xc = 1.470587
 x0 = 1.46, evals =  2, xc = 1.470557
 x0 = 1.47, evals =  1, xc = 1.470583
 x0 = 1.48, evals =  2, xc = 1.470536
 x0 = 1.49, evals =  3, xc = 1.470543
 x0 = 1.50, evals =  5, xc = 1.470587
 x0 = 1.51, evals = 15, xc = 1.670379
```

The following table shows the 3 zeros of the function f(x)

Table 2.3.1: zeros

Zero number	X location
1	1.47058
2	1.5625
3	1.6667

Discussion 2.4:

The results of the program show, as well as the plot in **Figure 2.3.1**, that the zeros of the function f(x) are as described by **Table 2.3.1**. After some experimentation, delta was found to control how precise the values of the zeros were given: that is how many significant figures the solutions held. The number of iteration also affected this to some degree, but not directly. The number of iterations and max number of iterations controlled another piece of data which was not the purpose of the program. The range of x0 in the set [1.61,1.62] show that number of iterations to find a solution tend to peek and may not find the closest zero either. This is explained by the fact that local minimum is found near this range of values. This result shows that the closer a guess made to a point where f_prime of x0 is zero, you will require more and more iterations to get to a solution.