# 6CS005 Learning Journal - Semester 1 2019/20

## Bishrut Neupane 1928726

## **Table of Contents**

	Table of Contents
<u>1</u>	<u>CUDA</u>
<u>1.1</u>	Password Cracking
<u>1.2</u>	Image Processing
<u>1.3</u>	Linear Regression
<u>2</u>	Verbose Repository Log

6CS005 Portfolio, Bishrut Neupane 1928726

#### 1 CUDA

### 1.1 Password Cracking

```
    #include <stdio.h>

2. #include <cuda runtime api.h>
3. #include <time.h>
4.
5.
6. __device__ int is_a_match(char *attempt) {
7.
        char mypassword1[] = "AV4567";
      char mypassword2[] = "FG7868";
8.
9.
        char mypassword3[] = "HJ7654";
10. char mypassword4[] = "DE6789";
11.
12.
        char *b = attempt;
13.
14. char *i = attempt;
15. char *s = attempt;
16. char *h = attempt;
17. char *p1 = mypassword1;
18. char *p2 = mypassword2;
19. char *p3 = mypassword2;
20. char *p4 = mypassword4;
21.
22.
        while(*b == *p1) {
             if(*b == '\0')
23.
24.
25.
                 printf("Password: %s\n",mypassword1);
26.
                 break;
27.
             }
28.
29.
             b++;
30.
             p1++;
31.
32.
```

```
while(*i == *p2) {
33.
            if(*i == '\0')
34.
35.
36.
                printf("Password: %s\n",mypassword2);
37.
                break;
38.
            }
39.
40.
            i++;
41.
            p2++;
42.
43.
        while(*s == *p3) {
44.
            if(*s == '\0')
45.
46.
47.
                printf("Password: %s\n",mypassword3);
48.
                break;
49.
            }
50.
51.
            s++;
52.
            p3++;
53.
        }
54.
        while(*h == *p4) {
55.
            if(*h == '\0')
56.
57.
            {
58.
                printf("Password: %s\n",mypassword4);
59.
                return 1;
60.
61.
62.
            h++;
            p4++;
63.
64.
65.
        return 0;
66.
67.}
68.
69. __global__ void kernel() {
70. char i1,i2,i3,i4;
71.
72.
        char password[7];
73.
        password[6] = '\0';
74.
75.
        int i = blockIdx.x+65;
76.
        int j = threadIdx.x+65;
77.
        char firstMatch = i;
78.
        char secondMatch = j;
79.
80.
        password[0] = firstMatch;
81.
        password[1] = secondMatch;
        for(i1='0'; i1<='9'; i1++){
82.
83.
            for(i2='0'; i2<='9'; i2++){</pre>
84.
                for(i3='0'; i3<='9'; i3++){</pre>
85.
                    for(i4='0'; i4<='9'; i4++){</pre>
86.
                        password[2] = i1;
87.
                        password[3] = i2;
88.
                        password[4] = i3;
89.
                         password[5] = i4;
90.
                        if(is a match(password)) {
91.
                        }
92.
                        else {
                        //printf("tried: %s\n", password);
93.
```

```
94.
95.
96.
97.
98.
99.}
100.
101.
           int time_difference(struct timespec *start,
               struct timespec *finish,
102.
               long long int *difference) {
103.
               long long int ds = finish->tv_sec - start->tv_sec;
104.
               long long int dn = finish->tv_nsec - start->tv_nsec;
105.
106.
               if(dn < 0 ) {
                   ds--;
107.
108.
                   dn += 1000000000;
109.
               *difference = ds * 1000000000 + dn;
110.
               return !(*difference > 0);
111.
112.
113.
114.
115.
           int main() {
116.
117.
               struct timespec start, finish;
               long long int time_elapsed;
118.
119.
               clock_gettime(CLOCK_MONOTONIC, &start);
120.
121.
               kernel <<<26,26>>>();
122.
               cudaThreadSynchronize();
123.
124.
               clock_gettime(CLOCK_MONOTONIC, &finish);
125.
               time_difference(&start, &finish, &time_elapsed);
126.
               printf("Time elapsed was %1ldns or %0.9lfs\n", time_elapsed, (time_elapsed/1
    .0e9));
127.
128.
               return 0;
129.
           }
```

#### 6CS005 Portfolio, Bishrut Neupane 1928726

Insert a table that shows running times for the original and CUDA versions.

S.N.	Nano seconds	CUDA Version	
1	57739815	0.057739851	
2	56786470	0.056786470	
3	58123375	0.058123375	
4	55750196	0.055750196	
5	54850015	0.054850015	
6	51335547	0.051335547	
7	54674328	0.054674328	
8	55536384	0.055536384	
9	55536384	0.055113559	
10	55134101	0.055134101	
AVERAGE	555.3644563 second	0.055134262 second	

#### Write a short analysis of the results

## 6CS005 Portfolio, put your name and student number here

## 1.2 Image Processing

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <GL/glut.h>
5. #include <GL/gl.h>
6. #include <malloc.h>
7. #include <signal.h>
8. #include <cuda_runtime_api.h>
9.
10. #define width 100
11. #define height 72
12.
13. unsigned char results[width * height];
15.
 16.
 17.
 18.
 19.
 20.
 21.
 22.
 23.
 24.
 25.
 26.
 27.
 28.
 29.
 30.
 31.
 32.
 33.
 34.
 35.
 36.
 37.
 38.
 39.
 40.
 41.
 42.
 43.
 44.
 45.
 46.
 47.
 48.
 49.
 50.
 51.
 52.
 53.
 54.
 55.
```

```
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
 100.
 101.
102.
 103.
 104.
 105.
 106.
 107.
108.
 109.
 110.
111.
 112.
 113.
 255,0,255,255,0,255,255,0,255,255,0,0,255,0,255,0,255,0,
114.
 115.
116.
```

```
117.
 118.
119.
 120.
 121.
122.
 123.
 124.
 125.
126.
 127.
 128.
 129.
 130.
 131.
132.
 133.
134.
 135.
136.
 137.
138.
 139.
 140.
 141.
 142.
143.
 144.
145.
 146.
147.
 148.
149.
 150.
 151.
152.
 0,255,0,255,255,255,255,0,0,255,0,0,255,0,0,255,0,0,
 153.
154.
 155.
 156.
 157.
 158.
 159.
 160.
 161.
 162.
163.
 164.
 165.
 166.
 167.
 168.
169.
 170.
 171.
 172.
 173.
 174.
175.
 176.
 177.
```

```
178.
 179.
 180.
 181.
 182.
 183.
 184.
 185.
 186.
187.
 188.
 189.
 190.
 191.
 192.
 193.
 194.
 195.
 196.
 197.
 198.
 199.
 200.
 201.
 202.
 203.
 204.
 205.
 206.
207.
 208.
 209.
 210.
 211.
 212.
 213.
 214.
 215.
 216.
 217.
 218.
 219.
 220.
 221.
 222.
 223.
 224.
 225.
 226.
 227.
 228.
 229.
 230.
 231.
 232.
 233.
234.
 235.
 236.
 237.
 238.
```

```
239.
 240.
 241.
 242.
 243.
 244.
 245.
 246.
 247.
 248.
 249.
 250.
 251.
 252.
253.
 254.
 255.
 256.
 257.
 258.
 259.
 260.
 261.
 262.
 263.
 264.
 265.
 266.
 267.
 268.
269.
 270.
 271.
 272.
 273.
 274.
 275.
 276.
 277.
 278.
 279.
 280.
 281.
 282.
 283.
 284.
 285.
 286.
 287.
 288.
 289.
 290.
 291.
 292.
 293.
 294.
 295.
 296.
 297.
 298.
 299.
```

```
300.
 301.
 302.
 303.
 304.
 305.
 306.
 307.
 308.
 309.
 310.
 311.
 312.
 313.
 314.
 315.
 316.
 317.
 318.
 319.
 320.
 321.
 322.
 323.
 324.
 325.
 326.
 327.
 328.
 329.
 330.
 331.
 332.
 333.
 334.
 335.
 336.
 337.
 338.
 339.
 340.
 341.
 342.
 343.
 344.
 345.
 346.
 347.
 348.
 349.
 350.
 351.
 352.
 353.
 354.
 355.
 356.
 357.
 358.
 359.
 360.
```

```
361.
     362.
     363.
     364.
     365.
     366.
     367.
     368.
     369.
     370.
     371.
     372.
     373.
     374.
     375.
     376.
     377.
     378.
     379.
     380.
     381.
     382.
     383.
     384.
     385.
     386.
     387.
     388.
     389.
     390.
     391.
     392.
     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
393.
     _global___ void detect_edges(unsigned char *in, unsigned char *out) {
394.
     int i = (blockIdx.x * 72) + threadIdx.x;
395.
     int y, w; // the pixel of interest
396.
397.
     int p, a, n, m; // the pixels adjacent to x,y used for the calculation
398.
     int o; // the result of calculate
399.
400.
      w = i / width;
401.
      y = i - (width * w);
402.
403.
      if (y == 0 || w == 0 || y == width - 1 || w == height - 1) {
404.
       out[i] = 0;
405.
      } else {
406.
       p = i + width;
407.
       a = i - 1;
408.
       n = i + 1;
409.
       m = i - width;
410.
411.
       o = (in[i] * 4) + (in[p] * -1) + (in[a] * -1) + (in[n] * -1)
412.
         + (in[m] * -1);
413.
414.
       if (o > 0) { // if the result is positive this is an edge pixel
415.
        out[i] = 255;
416.
       } else {
417.
        out[i] = 0;
418.
       }
419.
      }
420.
     }
421.
```

```
422.
423.
           void tidy_and_exit() {
424.
             exit(0);
425.
           }
426.
427.
           void sigint_callback(int signal_number){
428.
             printf("\nInterrupt from keyboard\n");
             tidy_and_exit();
429.
430.
           }
431.
           static void display() {
432.
433.
             glClear(GL_COLOR_BUFFER_BIT);
434.
             glRasterPos4i(-1, -1, 0, 1);
435.
             glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, image);
436.
             glRasterPos4i(0, -1, 0, 1);
437.
             glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, results);
438.
             glFlush();
439.
440.
441.
           static void key_pressed(unsigned char key, int x, int y) {
442.
             switch(key){
443.
               case 27:
444.
                 tidy_and_exit();
445.
                 break;
446.
               default:
447.
                 printf("\nPress escape to exit\n");
448.
                 break;
449.
             }
450.
           }
451.
452.
           int time_difference(struct timespec *start, struct timespec *finish,
453.
                                long long int *difference) {
454.
             long long int ds = finish->tv_sec - start->tv_sec;
             long long int dn = finish->tv_nsec - start->tv_nsec;
455.
456.
457.
             if(dn < 0 ) {
458.
               ds--;
459.
               dn += 1000000000;
460.
             *difference = ds * 1000000000 + dn;
461.
462.
             return !(*difference > 0);
463.
464.
           int main(int argc, char **argv) {
465.
466.
             unsigned char *d results;
467.
468.
             unsigned char *d_image;
469.
470.
             cudaMalloc((void**)&d_image, sizeof(unsigned char) * (width * height));
471.
472.
             cudaMalloc((void**)&d_results, sizeof(unsigned char) * (width * height));
473.
474.
             cudaMemcpy(d_image, &image, sizeof(unsigned char) * (width * height), cudaMemc
   pyHostToDevice);
475.
              signal(SIGINT, sigint callback);
476.
477.
478.
             struct timespec start, finish;
479.
             long long int time elapsed;
480.
             clock_gettime(CLOCK_MONOTONIC, &start);
481.
```

```
482.
             detect_edges<<<100,72>>>(d_image, d_results);
483.
             cudaThreadSynchronize();
484.
485.
            cudaMemcpy(&results, d_results, sizeof(unsigned char) * (width * height), cudaM
    emcpyDeviceToHost);
486.
487.
488.
             clock_gettime(CLOCK_MONOTONIC, &finish);
489.
490.
             time_difference(&start, &finish, &time_elapsed);
             printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
491.
492.
                     (time_elapsed/1.0e9));
493.
            cudaFree(&d_image);
494.
             cudaFree(&d_results);
495.
496.
             glutInit(&argc, argv);
497.
             glutInitWindowSize(width * 2,height);
             glutInitDisplayMode(GLUT_SINGLE | GLUT_LUMINANCE);
498.
499.
500.
             glutCreateWindow("6CS005 Image Progessing Courework");
501.
             glutDisplayFunc(display);
502.
             glutKeyboardFunc(key_pressed);
503.
             glClearColor(0.0, 1.0, 0.0, 1.0);
504.
505.
             glutMainLoop();
506.
507.
             tidy_and_exit();
508.
509.
             return 0;
510.
           }
```

### 6CS005 Portfolio, put your name and student number here

Insert a table that shows running times for the original and CUDA versions.

S.N.	Original Version		CUDA Version
	L	0.000128093	0.000210587
	2	0.000184618	0.000178902
3	3	0.000387604	0.000250891
4	1	0.000407642	0.000187704
	5	0.000961456	0.000194665
(	5	0.000759508	0.000378856
-	7	0.000248477	0.000238169
8	3	0.000453902	0.000186494
9	9	0.000178767	0.000180056
10	)	0.000301390	0.000193863
AVERAGI	0.00	0154084 second	0.000216523 second

#### 6CS005 Portfolio Bishrut Neupane 1928726

### 1.3 Linear Regression

```
1. #include <stdio.h>
2. #include <math.h>
3. #include <time.h>
4. #include <unistd.h>
5. #include <cuda runtime api.h>
6. #include <errno.h>
7. #include <unistd.h>
8.
9.
10.
11. typedef struct point_t {
12. double x;
13.
      double g;
14. } point_t;
15.
16. int n_data = 1000;
17. __device__ int d_n_data = 1000;
18.
19.
20. point_t data[] = {
21.
      {82.73,128.67},{79.53,133.54},{66.86,124.65},{69.21,135.74},
22.
      {82.20,122.07},{84.32,120.46},{71.12,93.14},{85.64,121.42},
23.
      {69.22,116.28},{83.12,137.30},{84.31,113.18},{75.60,121.42},
24.
      {69.04,91.83},{85.41,131.06},{17.44,58.69},{68.92,119.86},
25.
      {69.95,110.05},{ 0.15, 5.39},{73.96,118.70},{27.70,64.64},
26.
      {97.97,158.15},{56.21,100.99},{30.27,48.32},{37.47,89.65},
27.
      {98.98,144.03},{92.61,133.89},{ 4.72,32.88},{19.51,57.43},
28.
      {94.74,145.50},{31.66,71.27},{94.76,134.53},{32.73,59.95},
29.
      {32.64,54.53},{38.78,69.06},{91.47,150.49},{77.99,119.35},
30.
      {33.38,65.87},{79.28,123.62},{39.69,72.53},{95.47,140.97},
31.
      {82.64,137.69},{25.53,51.33},{68.58,85.98},{92.25,132.34},
32.
      {74.79,101.30},{ 1.32,18.87},{53.85,95.13},{78.75,128.26},
33.
      { 2.91,21.77},{90.68,128.55},{11.44,35.27},{30.72,56.54},
34.
      {49.06,74.08},{49.09,83.45},{62.54,104.58},{38.83,72.26},
35.
      {78.43,130.83},{69.49,122.49},{27.27,56.35},{80.06,131.95},
36.
      { 5.73,39.00},{80.21,140.42},{ 8.47,36.12},{86.98,152.43},
37.
      {64.26,108.56},{95.74,133.36},{15.06,48.67},{31.96,72.31},
38.
      {95.27,141.34},{61.10,89.26},{27.51,68.47},{26.48,60.30},
39.
      {92.61,128.38},{ 8.25,47.51},{90.69,118.91},{45.40,79.96},
40.
      {23.59,53.12},{46.71,68.27},{21.15,50.29},{27.99,76.29},
41.
      { 7.75,43.57},{13.70,43.56},{74.85,97.83},{50.93,103.11},
42.
      {33.80,64.85},{80.99,125.37},{92.41,126.27},{92.61,134.36},
43.
      {34.70,55.32},{35.07,55.04},{86.87,157.26},{41.99,90.46},
44.
      {16.27,44.43},{36.31,83.84},{22.35,73.11},{89.11,127.19},
45.
      {56.11,77.28},{51.90,75.07},{35.74,94.18},{10.66,29.60},
46.
      {61.27,114.15},{77.55,117.04},{61.17,99.68},{15.54,55.33},
47.
      {91.99,143.18},{12.91,21.82},{48.52,89.94},{54.88,90.86},
48.
      {73.59,131.33},{5.49,13.95},{92.31,147.29},{48.50,89.49},
49.
      {40.02,58.26},{48.22,81.96},{17.08,52.59},{34.27,66.17},
50.
      {59.06,94.26},{92.71,134.53},{37.70,65.30},{77.11,111.38},
51.
      {43.27,74.12},{79.71,123.45},{ 0.86,38.69},{ 3.00,17.76},
52.
      {56.03,80.33},{17.66,43.27},{18.39,47.08},{31.08,83.84},
53.
      {32.64,77.85},{51.68,84.57},{78.46,134.18},{9.57,40.28},
54.
      {68.38,98.26},{30.29,67.59},{86.15,131.86},{16.82,64.91},
55.
      { 3.35,20.88},{65.78,98.73},{50.70,90.92},{38.26,71.11},
56.
      {85.52,132.23}, {44.06,83.02}, {44.09,86.42}, {81.86,114.30},
57.
      {33.98,69.09},{93.80,147.73},{59.58,103.07},{98.75,154.73},
```

```
{88.98,120.59},{78.08,109.00},{82.77,133.94},{76.49,106.31},
59.
      {55.38,85.71},{46.56,79.57},{83.92,141.58},{81.38,133.52},
      { 4.88,35.01},{ 4.57,17.99},{57.96,90.07},{33.42,63.80},
60.
61.
      { 9.95,34.53},{47.14,92.75},{63.17,105.19},{95.01,163.93},
62.
      {30.36,57.81},{ 2.46,23.97},{69.75,115.88},{64.85,111.01},
63.
      {25.18,56.58},{69.84,104.78},{40.43,51.98},{75.61,107.05},
64.
      {36.75,69.37},{50.08,100.02},{64.97,103.68},{41.72,86.64},
65.
      { 1.70,47.26},{99.93,141.75},{24.57,64.51},{75.23,116.35},
      { 1.95,18.53},{78.84,102.70},{67.38,97.71},{55.35,82.37},
66.
67.
      {58.10,100.09},{53.10,96.07},{41.24,83.81},{68.86,111.98},
68.
      {87.36,86.88},{54.06,98.42},{64.12,90.56},{11.77,49.66},
69.
      {99.43,134.33},{55.24,99.18},{56.44,74.73},{39.47,62.99},
70.
      { 8.94,48.15},{92.91,130.45},{87.68,138.76},{80.37,116.69},
71.
      {56.72,108.65},{ 0.76,24.26},{26.98,75.13},{ 0.39,42.16},
72.
      {81.99,138.50},{88.32,117.16},{51.01,87.42},{21.38,55.45},
73.
      {72.66,122.82},{18.04,53.56},{11.22,49.73},{36.75,60.26},
74.
      {64.81,90.19},{72.72,121.14},{24.03,74.08},{41.38,81.38},
75.
      {62.79,98.75},{63.66,109.17},{91.12,143.91},{7.41,34.06},
76.
      {94.05,131.99},{53.12,90.28},{68.31,114.79},{25.33,67.23},
77.
      {42.34,86.91},{94.61,131.38},{43.78,73.28},{50.18,78.10},
78.
      {81.64,135.88},{11.27,44.45},{41.03,76.34},{21.25,57.54},
79.
      {29.23,57.27},{35.74,75.16},{ 0.91,14.33},{30.08,59.05},
80.
      {23.99,56.25},{90.79,120.98},{99.22,152.22},{94.21,143.09},
81.
      {19.35,30.03},{82.04,113.25},{79.22,113.69},{83.40,144.06},
82.
      {55.82,80.85},{42.49,48.94},{17.60,55.62},{35.65,81.91},
83.
      {82.50,135.41},{81.15,114.46},{53.47,78.67},{44.30,73.73},
84.
      {32.88,80.28},{99.26,147.55},{76.32,110.24},{78.97,110.27},
85.
      {18.08,47.48},{87.01,140.40},{56.25,83.61},{42.62,55.40},
86.
      {15.95,16.25},{47.85,106.69},{6.61,35.83},{66.38,116.30},
87.
      {94.97,122.56},{42.29,73.37},{31.48,67.15},{69.67,105.40},
88.
      {30.41,65.31},{ 2.98,19.40},{ 8.12,48.34},{80.41,127.03},
89.
      {63.68,112.61},{24.60,78.23},{77.61,123.49},{39.87,38.20},
      {77.80,109.59},{58.53,107.63},{23.97,62.36},{7.77,27.38},
90.
91.
      { 0.80,41.55},{ 6.45,32.91},{45.32,82.24},{35.56,59.56},
92.
      {65.05,97.68},{62.21,96.14},{86.61,121.99},{87.91,125.40},
93.
      {48.08,88.87},{ 2.41,40.02},{69.55,119.31},{22.07,61.86},
94.
      {61.87,121.40},{82.50,119.46},{26.97,38.40},{31.53,86.30},
95.
      { 1.81,38.57},{72.57,108.34},{88.88,139.23},{63.90,95.79},
96.
      {93.29,135.35},{86.26,143.55},{63.62,94.76},{20.24,38.84},
97.
      {16.23,48.64},{72.87,108.22},{16.26,51.25},{37.86,66.06},
98.
      {57.53,81.37},{61.66,97.20},{49.48,84.98},{95.20,142.45},
      {12.10,45.25},{47.79,84.80},{17.29,48.98},{47.11,87.23},
100.
             {85.74,119.95},{89.94,142.94},{97.68,155.27},{78.73,123.81},
101.
             {51.65,85.91},{52.82,96.05},{50.95,93.50},{16.14,37.21},
             {16.73,41.57},{57.25,95.50},{78.47,136.77},{42.35,75.64},
102.
             {93.24,135.04},{12.56,38.20},{21.40,62.92},{70.60,136.98},
103.
104.
             {44.04,83.57},{ 6.43,36.61},{12.01,50.32},{79.61,119.78},
             {43.05,69.07},{14.42,53.01},{51.68,83.82},{25.59,55.77},
105.
             { 9.14,31.58},{37.24,80.94},{15.73,69.21},{71.54,123.11},
106.
             { 1.26,25.72},{ 4.25,38.46},{21.42,39.99},{44.12,79.01},
107.
             {31.12,64.63},{85.27,143.62},{43.25,79.30},{77.27,104.30},
108.
             {47.34,83.76},{90.57,125.82},{17.35,36.40},{82.01,130.41},
109.
110.
             {81.58,124.10},{68.62,117.62},{47.48,79.29},{4.30,26.77},
111.
             { 6.94,32.22},{11.71,55.76},{22.62,54.74},{58.43,89.61},
112.
             {69.10,111.51},{56.77,101.10},{67.10,102.75},{93.20,144.51},
             {83.61,128.56},{71.97,116.09},{75.19,122.16},{48.03,79.67},
113.
             {97.95,143.80},{92.27,123.08},{23.88,63.39},{79.15,115.57},
114.
115.
             {24.42,51.27},{12.58,34.65},{46.58,78.16},{1.29,37.96},
116.
             {17.09,45.61},{12.45,40.77},{82.75,107.46},{52.15,75.34},
117.
             {39.51,68.51},{31.71,64.23},{39.36,72.00},{12.16,37.99},
             {83.13,127.76},{42.25,73.17},{45.32,77.14},{20.52,36.60},
118.
```

```
119.
             { 7.99,11.50},{23.34,55.47},{25.87,54.36},{78.73,112.49},
120.
             {55.60,94.90},{31.98,73.40},{85.93,137.12},{58.56,97.64},
             {88.16,120.43},{78.65,136.60},{25.93,43.32},{84.83,136.32},
121.
122.
             {68.09,102.12},{68.36,111.80},{39.80,69.69},{ 0.38,27.89},
123.
             { 4.49,27.85},{32.53,66.32},{54.23,97.63},{19.98,67.32},
124.
             {90.62,143.43},{18.31,67.91},{95.66,146.41},{95.41,149.68},
125.
             {71.64,111.15},{23.02,44.96},{97.06,154.54},{41.58,75.95},
             {79.80,130.01},{74.55,119.44},{72.19,113.27},{70.01,106.48},
126.
127.
             {75.24,94.18},{19.82,60.09},{96.31,137.91},{2.21,27.44},
128.
             {40.52,70.36},{ 2.40,29.12},{35.24,57.25},{26.38,71.34},
129.
             {26.02,59.48},{34.73,66.07},{45.15,78.23},{9.35,32.58},
             {19.37,57.18},{ 9.51,31.70},{15.03,49.81},{85.08,140.35},
130.
             { 3.23,13.46},{58.26,108.47},{ 4.84,31.78},{49.49,83.50},
131.
             {35.55,70.67},{26.51,55.44},{20.12,53.39},{72.73,119.37},
132.
133.
             {31.04,72.96},{30.66,58.35},{ 2.96,33.18},{18.68,31.50},
134.
             {91.41,138.24},{44.67,81.81},{81.57,135.26},{ 0.17,26.66},
135.
             {49.03,100.11},{54.47,102.27},{61.78,113.45},{22.67,59.51},
136.
             {89.80,143.05},{33.05,78.20},{67.76,108.19},{7.64,41.18},
137.
             {36.91,87.28},{95.44,147.27},{52.76,94.34},{ 3.52,29.51},
138.
             {87.39,118.48}, {41.48,64.71}, {1.44,14.21}, {95.04,136.99},
139.
             {71.77,115.75},{23.39,47.58},{62.66,115.03},{15.98,34.38},
140.
             {29.06,62.77},{ 2.94,28.25},{71.50,119.18},{65.24,119.14},
141.
             {30.65,82.39},{16.36,38.82},{ 0.98,48.82},{33.19,56.41},
142.
             {27.49,64.34},{53.69,102.47},{28.15,52.58},{40.21,66.07},
143.
             {50.56,86.39},{74.71,97.44},{24.72,46.29},{48.05,80.63},
144.
             {34.99,52.13},{66.75,115.96},{17.62,49.17},{98.99,157.80},
145.
             {37.96,72.18},{56.88,105.06},{48.27,97.04},{71.18,138.90},
146.
             {46.35,82.02},{10.43,44.65},{24.14,42.85},{82.21,144.13},
147.
             {96.85,148.15}, {93.68,126.32}, {33.02,61.55}, {66.73,108.51},
148.
             {83.89,136.35},{80.85,91.16},{79.21,128.88},{84.37,119.84},
149.
             {38.41,71.48},{47.49,85.53},{ 1.54,24.44},{68.32,106.44},
150.
             {22.82,54.16},{ 2.65,16.35},{19.91,53.53},{12.99,34.98},
151.
             {30.87,57.17},{44.10,83.88},{15.84,31.99},{36.46,59.74},
152.
             {26.25,79.73},{79.12,132.06},{86.26,132.45},{ 0.61,23.61},
153.
             {33.94,59.37},{99.92,145.88},{26.20,53.99},{69.77,115.40},
154.
             {69.07,107.00},{ 1.89,17.20},{38.25,81.40},{27.08,62.96},
155.
             {23.09,53.98},{55.56,86.93},{ 6.68,50.41},{22.86,49.26},
156.
             {17.25,50.25},{19.01,50.16},{35.07,85.09},{59.08,89.15},
             {87.02,128.83},{ 1.57,27.68},{97.76,148.25},{70.78,108.00},
157.
158.
             {38.01,65.83},{96.41,139.67},{ 2.86,22.44},{27.05,53.00},
159.
             {90.99,134.97},{86.60,145.27},{54.66,99.42},{67.61,107.07},
160.
             {85.16,137.50},{87.64,144.60},{14.69,36.65},{16.08,49.31},
161.
             {14.45,44.07},{65.91,98.39},{50.74,90.72},{ 6.98,31.11},
162.
             {52.76,83.96},{ 8.03,43.93},{17.58,52.58},{33.63,59.04},
163.
             {87.65,137.34},{77.97,142.54},{30.56,69.47},{59.61,114.61},
164.
             {14.05,53.07},{87.65,116.66},{33.19,75.96},{31.87,66.95},
165.
             {25.89,57.59},{48.60,75.67},{80.25,109.89},{6.61,24.27},
166.
             { 4.56,44.00},{40.17,62.33},{92.32,117.73},{75.07,112.71},
167.
             {17.10,35.12},{39.06,66.60},{4.26,34.01},{52.95,102.49},
168.
             {45.73,76.57},{ 4.72,29.94},{ 2.01,17.54},{39.08,88.44},
             {82.94,141.75},{44.51,90.97},{27.27,63.14},{60.16,95.38},
169.
170.
             {41.26,72.59},{66.50,104.49},{58.37,110.13},{62.11,96.01},
171.
             {70.30,90.15},{18.47,47.61},{24.80,51.82},{79.02,133.40},
172.
             {96.61,147.92},{18.14,33.27},{ 0.83,51.20},{99.67,143.65},
173.
             {34.07,67.38},{57.28,110.02},{35.92,59.90},{66.15,124.45},
174.
             {81.82,135.08},{ 2.97,28.49},{95.97,135.79},{51.17,80.95},
175.
             {91.47,142.00},{94.09,121.08},{57.70,82.98},{67.96,100.92},
176.
             {81.91,132.34},{11.55,39.74},{86.59,126.05},{5.36,41.72},
177.
             {90.86,144.15},{81.02,137.56},{35.87,81.76},{63.73,105.92},
178.
             {78.29,129.54},{96.72,150.04},{14.97,61.93},{45.76,77.17},
179.
             {82.69,123.95},{85.82,132.89},{85.95,127.24},{15.04,36.92},
```

```
180.
             {89.91,112.87},{30.86,58.13},{5.77,42.22},{75.24,108.41},
181.
             { 8.43,32.09},{90.70,147.99},{80.16,112.57},{42.81,73.54},
182.
             {82.47,123.41},{48.23,98.48},{77.48,143.96},{ 0.48,14.50},
             {29.75,63.12},{88.76,137.72},{33.59,70.61},{22.74,43.51},
183.
184.
             {82.15,116.11},{89.10,120.65},{26.56,68.17},{40.72,74.98},
185.
             {68.46,99.23},{34.82,66.71},{36.56,67.33},{72.32,114.23},
             {29.65,65.99},{44.39,64.83},{82.08,116.35},{99.73,139.12},
186.
187.
             {79.04,118.48},{20.78,42.05},{72.39,96.47},{90.62,147.11},
             {35.99,59.11},{50.65,83.23},{59.04,100.47},{87.01,145.78},
188.
189.
             {43.71,76.56},{95.61,151.81},{50.25,88.96},{69.64,122.07},
190.
             {40.07,79.38},{82.61,133.63},{20.84,39.75},{10.28,42.50},
             {47.43,70.82},{30.47,67.19},{69.16,100.10},{46.06,74.00},
191.
192.
             {93.78,152.76},{19.93,67.46},{79.61,130.88},{81.11,120.11},
193.
             {76.16,123.94},{75.84,111.70},{50.97,85.30},{47.35,90.59},
194.
             {93.21,115.44},{19.22,39.30},{11.67,29.58},{52.48,95.64},
195.
             {38.76,59.62},{ 2.74,-2.03},{18.99,63.67},{82.38,128.08},
196.
             {15.68,32.34},{39.19,83.38},{31.06,65.92},{28.91,73.05},
197.
             {19.01,59.69},{76.62,117.74},{36.82,91.33},{86.28,121.19},
198.
             {39.26,50.72},{41.45,70.26},{65.81,111.41},{77.09,117.88},
199.
             {78.96,128.48},{16.41,56.61},{39.54,64.11},{72.45,110.54},
200.
             {48.83,77.35},{27.61,51.82},{26.53,47.44},{83.06,111.09},
201.
             {97.06,127.57},{89.01,146.82},{89.44,141.17},{69.18,100.25},
202.
             { 1.11,11.60},{71.63,123.66},{92.93,151.73},{99.46,165.34},
203.
             {36.49,71.56},{95.48,153.13},{65.33,102.37},{15.28,35.93},
204.
             { 5.52,36.67},{ 0.78,42.47},{10.09,36.68},{ 5.75,37.39},
205.
             {52.34,89.11},{14.55,47.37},{67.92,113.35},{36.66,77.34},
206.
             {99.76,143.75},{26.67,58.72},{ 3.21,39.37},{87.70,124.12},
207.
             {90.03,131.24},{51.54,91.39},{62.86,98.04},{52.75,90.87},
208.
             {34.17,84.31},{62.00,89.08},{82.47,111.89},{61.38,123.48},
209.
             {47.17,84.64},{20.91,53.51},{96.96,131.54},{46.06,85.14},
210.
             {26.85,71.44},{91.67,138.51},{54.07,85.26},{51.63,89.63},
211.
              {94.04,140.80},{67.75,107.07},{29.24,76.71},{38.29,75.78},
212.
             {28.49,72.87},{60.51,102.28},{77.22,107.79},{99.25,145.86},
213.
             {33.11,52.32},{72.47,125.80},{21.97,59.23},{14.25,61.11},
214.
             {23.79,63.11},{77.78,109.13},{23.51,81.38},{66.92,110.89},
215.
             {79.81,109.80},{56.72,94.63},{59.60,110.57},{57.68,104.54},
216.
             {27.83,42.43},{47.80,87.68},{58.79,76.51},{78.33,126.71},
217.
             {85.14,128.99},{71.61,116.42},{58.09,96.85},{44.89,71.34},
218.
             {33.12,80.19},{98.79,130.09},{44.57,82.03},{88.63,142.61},
219.
             {61.96,98.55},{58.54,106.80},{19.17,61.00},{13.51,26.68},
220.
             {76.68,124.52},{82.62,138.53},{78.13,122.09},{37.10,60.33},
221.
             { 8.82,48.63},{71.64,105.27},{68.44,115.07},{ 7.66,61.91},
             {64.37,96.58},{54.90,88.28},{78.35,133.29},{79.84,129.58},
222.
             { 3.09,28.37},{48.62,76.00},{38.26,63.99},{42.05,102.17},
223.
224.
             {48.89,73.66},{54.38,100.05},{11.16,55.43},{63.24,110.69},
225.
             {68.17,114.15}, {68.68,109.15}, {53.43,90.23}, {67.45,106.67},
226.
             {10.60,34.41},{56.81,90.86},{11.42,27.08},{36.93,93.13},
227.
             {41.64,89.77},{69.74,103.98},{23.07,55.12},{44.98,83.65},
228.
             {35.75,72.65},{14.80,56.15},{72.19,115.53},{51.10,80.69},
229.
             {96.54,140.10},{15.04,62.30},{21.17,56.15},{46.42,79.63},
230.
             {22.35,52.01},{35.47,54.95},{4.27,21.33},{84.37,139.55},
             {43.95,93.24},{86.56,132.82},{44.35,83.36},{76.81,114.79},
231.
232.
             { 1.05,31.66},{32.76,76.15},{83.66,120.90},{12.14,42.52},
233.
             {25.85,55.83},{82.12,140.05},{75.33,126.93},{32.92,75.90},
234.
             { 7.52,24.51},{25.42,41.55},{42.57,67.15},{87.36,150.38},
             { 0.51,17.68}, {45.70,84.75}, {58.74,88.68}, {28.62,74.38},
235.
             {73.22,113.45},{78.64,114.25},{42.40,92.03},{84.22,132.25},
236.
237.
             {54.24,73.34},{ 2.71,30.27},{54.11,84.97},{66.74,112.66},
238.
             {28.80,57.88},{87.02,146.20},{32.02,63.03},{59.57,94.41},
239.
             {40.46,79.73},{23.74,49.78},{87.58,140.94},{84.15,113.32},
             {32.48,63.48},{ 4.59,25.85},{98.00,128.35},{12.23,37.43},
240.
```

```
241.
             {66.17,102.97},{50.73,93.82},{74.68,137.79},{43.72,92.85},
             {53.95,91.99},{54.47,105.25},{56.70,104.89},{16.59,46.52},
242.
243.
             {71.56,115.18},{80.62,99.79},{71.29,101.42},{16.81,56.15},
244.
             {48.88,84.93},{ 8.41,40.02},{93.98,147.39},{39.20,86.04},
245.
             {61.75,90.80},{ 1.06,32.69},{21.40,33.33},{ 8.60,28.69},
246.
             {38.80,61.88},{14.41,38.37},{40.14,70.01},{69.45,105.44},
247.
             {14.41,43.93},{51.20,93.11},{39.10,57.10},{21.04,39.51},
248.
             \{10.12,30.43\},\{70.13,93.88\},\{1.74,20.56\},\{12.23,34.33\},
249.
             {98.81,151.87},{50.48,92.07},{6.98,9.52},{24.08,69.94},
250.
             {15.72,40.89},{83.99,127.44},{47.21,90.46},{88.31,138.70},
251.
             {91.05,132.13},{45.22,62.24},{87.76,128.67},{99.37,168.24},
252.
             {94.38,140.24},{31.30,67.65},{40.85,84.03},{40.91,79.56},
253.
             {77.14,135.74},{50.92,80.52},{17.81,49.14},{90.30,135.15},
254.
             {28.44,64.60},{49.23,85.12},{81.63,141.58},{83.04,111.19},
255.
             {28.39,63.30},{ 8.61,44.11},{25.36,50.79},{51.35,93.32},
256.
             {64.49,80.42},{96.17,134.31},{96.10,144.32},{47.58,83.36},
257.
             {94.38,131.03},{41.97,69.05},{37.86,62.21},{26.97,65.30},
258.
             {37.57,88.95},{65.08,108.58},{17.68,39.80},{63.75,103.14},
259.
             {91.86,132.07},{76.35,121.19},{22.98,34.87},{96.46,140.54},
260.
             { 9.38,31.40},{42.97,82.09},{20.56,49.02},{13.73,41.31},
261.
             {37.35,63.18},{69.54,105.57},{38.17,83.30},{47.04,80.34},
262.
             {48.79,98.00},{39.34,61.59},{82.57,125.55},{40.77,82.18},
263.
             {13.62,53.38},{35.33,95.17},{95.36,148.79},{20.25,62.00},
264.
             {47.48,86.54},{30.22,61.07},{83.90,120.30},{85.81,123.25},
265.
             {84.29,130.44},{52.84,95.43},{96.72,140.32},{3.29,45.68},
266.
             {71.77,98.66},{ 8.52,42.40},{22.55,54.27},{15.08,47.10},
267.
             {91.29,130.23},{16.48,40.04},{44.84,72.14},{34.44,73.42},
268.
             {36.26,78.30},{58.45,115.51},{96.59,150.22},{63.80,98.30},
269.
             {85.92,120.14},{93.68,129.88},{74.09,119.30},{99.44,136.93},
             {88.39,131.55},{64.40,117.89},{13.87,47.30},{81.17,106.77}
270.
271.
272.
           double residual_error(double x, double g, double m, double c) {
             double e = (m * x) + c - g;
273.
             return e * e;
274.
275.
           }
276.
277.
             _device__ double d_residual_error(double x, double g, double m, double c) {
278.
             double e = (m * x) + c - g;
             return e * e;
279.
280.
281.
282.
           double rms error(double m, double c) {
283.
             int i:
284.
             double mean;
285.
             double error sum = 0;
286.
287.
             for(i=0; i<n data; i++) {</pre>
288.
               error_sum += residual_error(data[i].x, data[i].g, m, c);
289.
290.
291.
             mean = error_sum / n_data;
292.
293.
             return sqrt(mean);
294.
295.
296.
             global__ void d_rms_error(double *m, double *c, double *error_sum_arr, point_t
     *d data) {
297.
298.
               int i = threadIdx.x + blockIdx.x * blockDim.x;
299.
300.
             error_sum_arr[i] = d_residual_error(d_data[i].x, d_data[i].g, *m, *c);
```

```
301.
           }
302.
303.
           int time_difference(struct timespec *start, struct timespec *finish,
304.
                                          long long int *difference) {
305.
             long long int ds = finish->tv_sec - start->tv_sec;
306.
             long long int dn = finish->tv_nsec - start->tv_nsec;
307.
             if(dn < 0 ) {
308.
309.
               ds--;
310.
               dn += 1000000000;
311.
312.
             *difference = ds * 1000000000 + dn;
313.
             return !(*difference > 0);
314.
315.
316.
           int main() {
317.
             int i;
318.
             double bm = 1.3;
319.
             double bc = 10;
320.
             double be;
321.
             double dm[8];
322.
             double dc[8];
323.
             double e[8];
324.
             double step = 0.01;
             double best_error = 999999999;
325.
326.
             int best_error_i;
             int minimum_found = 0;
327.
328.
329.
             double om[] = \{0,1,1, 1, 0,-1,-1,-1\};
             double oc[] = {1,1,0,-1,-1,-1, 0, 1};
330.
331.
332.
               struct timespec start, finish;
333.
             long long int time_elapsed;
334.
335.
336.
             clock_gettime(CLOCK_MONOTONIC, &start);
337.
338.
               cudaError_t error;
339.
340.
             double *d dm;
341.
342.
             double *d dc;
               double *d_error_sum_arr;
343.
344.
               point t *d data;
345.
346.
             be = rms error(bm, bc);
347.
348.
349.
               error = cudaMalloc(&d_dm, (sizeof(double) * 8));
350.
               if(error){
               fprintf(stderr, "cudaMalloc on d_dm returned %d %s\n", error,
351.
352.
                   cudaGetErrorString(error));
353.
               exit(1);
354.
               }
355.
356.
357.
               error = cudaMalloc(&d dc, (sizeof(double) * 8));
358.
               if(error){
359.
               fprintf(stderr, "cudaMalloc on d dc returned %d %s\n", error,
360.
                 cudaGetErrorString(error));
361.
               exit(1);
```

```
362.
363.
364.
365.
               error = cudaMalloc(&d_error_sum_arr, (sizeof(double) * 1000));
366.
               if(error){
367.
               fprintf(stderr, "cudaMalloc on d_error_sum_arr returned %d %s\n", error,
368.
                 cudaGetErrorString(error));
369.
               exit(1);
370.
371.
372.
373.
               error = cudaMalloc(&d_data, sizeof(data));
374.
               if(error){
375.
               fprintf(stderr, "cudaMalloc on d_data returned %d %s\n", error,
376.
                 cudaGetErrorString(error));
377.
               exit(1);
378.
               }
379.
380.
             while(!minimum found) {
381.
               for(i=0;i<8;i++) {
382.
                 dm[i] = bm + (om[i] * step);
383.
                 dc[i] = bc + (oc[i] * step);
384.
385.
386.
387.
               error = cudaMemcpy(d_dm, dm, (sizeof(double) * 8), cudaMemcpyHostToDevice);
388.
               if(error){
                    fprintf(stderr, "cudaMemcpy to d_dm returned %d %s\n", error,
389.
                 cudaGetErrorString(error));
390.
391.
               }
392.
393.
               error = cudaMemcpy(d_dc, dc, (sizeof(double) * 8), cudaMemcpyHostToDevice);
394.
395.
               if(error){
396.
                    fprintf(stderr, "cudaMemcpy to d_dc returned %d %s\n", error,
397.
                 cudaGetErrorString(error));
398.
399.
400.
401.
               error = cudaMemcpy(d data, data, sizeof(data), cudaMemcpyHostToDevice);
402.
               if(error){
                    fprintf(stderr, "cudaMemcpy to d data returned %d %s\n", error,
403.
404.
                 cudaGetErrorString(error));
405.
               }
406.
407.
               for(i=0;i<8;i++) {</pre>
408.
409.
                        double h error sum arr[1000];
410.
                        double error sum total;
411.
                        double error sum mean;
412.
                        d_rms_error <<<100,10>>>(&d_dm[i], &d_dc[i], d_error_sum_arr, d_data
   );
413.
                        cudaThreadSynchronize();
                      error = cudaMemcpy(&h_error_sum_arr, d_error_sum_arr, (sizeof(double)
   * 1000), cudaMemcpyDeviceToHost);
415.
                      if(error){
416.
                    fprintf(stderr, "cudaMemcpy to error sum returned %d %s\n", error,
417.
                      cudaGetErrorString(error));
418.
```

```
419.
                        for(int j=0; j<n_data; j++) {</pre>
420.
                        error_sum_total += h_error_sum_arr[j];
421.
                    }
422.
423.
                        error_sum_mean = error_sum_total / n_data;
424.
                        e[i] = sqrt(error_sum_mean);
425.
426.
                  if(e[i] < best_error) {</pre>
427.
                    best error = e[i];
428.
                    best error i = i;
429.
                  }
430.
431.
                        error_sum_total = 0;
432.
433.
434.
435.
               if(best_error < be) {</pre>
436.
                 be = best_error;
437.
                  bm = dm[best error i];
438.
                 bc = dc[best_error_i];
439.
               } else {
440.
                 minimum_found = 1;
441.
442.
443.
444.
               error = cudaFree(d_dm);
445.
               if(error){
                    fprintf(stderr, "cudaFree on d_dm returned %d %s\n", error,
446.
447.
                    cudaGetErrorString(error));
448.
                    exit(1);
449.
               }
450.
451.
               error = cudaFree(d_dc);
               if(error){
452.
                    fprintf(stderr, "cudaFree on d dc returned %d %s\n", error,
453.
454.
                        cudaGetErrorString(error));
455.
                    exit(1);
456.
457.
458.
               error = cudaFree(d_data);
459.
               if(error){
460.
                    fprintf(stderr, "cudaFree on d_data returned %d %s\n", error,
461.
                    cudaGetErrorString(error));
462.
                    exit(1);
463.
               }
464.
465.
               error = cudaFree(d_error_sum_arr);
466.
               if(error){
                    fprintf(stderr, "cudaFree on d_error_sum_arr returned %d %s\n", error,
467.
468.
                    cudaGetErrorString(error));
469.
                    exit(1);
470.
               }
471.
472.
             printf("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);
473.
474.
               clock gettime(CLOCK MONOTONIC, &finish);
475.
476.
             time difference(&start, &finish, &time elapsed);
477.
478.
             printf("Time elapsed was %11dns or %0.91fs\n", time elapsed,
479.
                     (time_elapsed/1.0e9));
```

```
480.
481. return 0;
482. }

6CS005 Portfolio, Bishrut Neupane 1928726
```

Insert a table that shows running times for the original and CUDA versions.

SN	Original Version	CUDA Version
1	0.0681925	0.362978443
2	0.068495572	0.353358696
3	0.069165606	0.360965479
4	0.068734417	0.356041658
5	0.068627775	0.358192532
6	0.06858241	0.353261725
7	0.068617562	0.354974673
8	0.068765762	0.352310346
9	0.068644216	0.357385403
10	0.068706941	0.356378793
AVERAGE	0.068653276 second	0.352354312 second

Write a short analysis of the results

# 2 Verbose Repository Log

Paste your verbose format repository  $\log$  here. With subversion this can be achieved by the following:

svn update
svn -v log > log.txt
gedit log.txt
Then select, copy and paste the text here

6CS005 Portfolio, Bishrut Neupane 1928726

6CS005 Portfolio, Bishrut Neupane 1928726