In the name of God

Project 2 — Parallel Programming Course

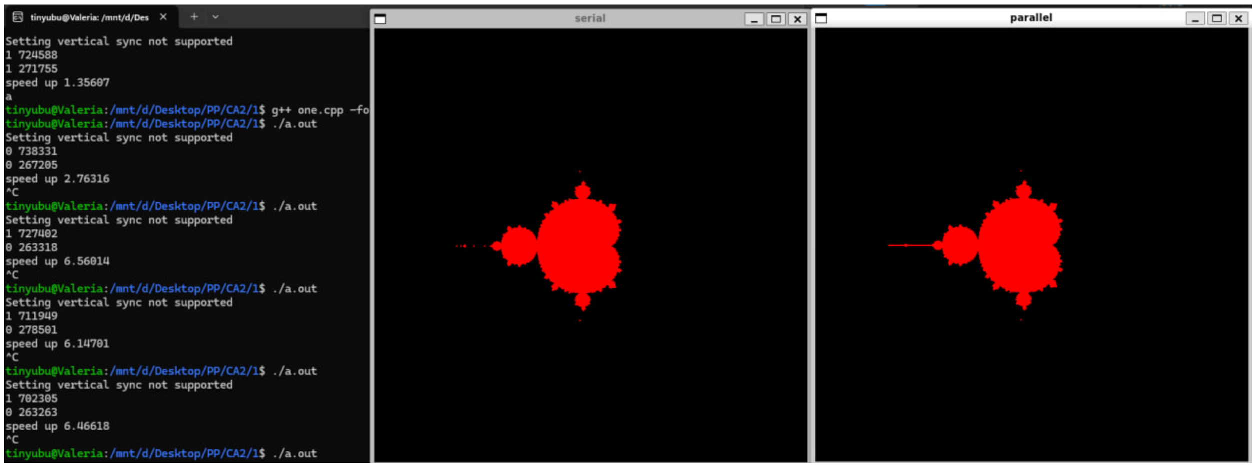
Instructor: **Dr. Safari**

Students:

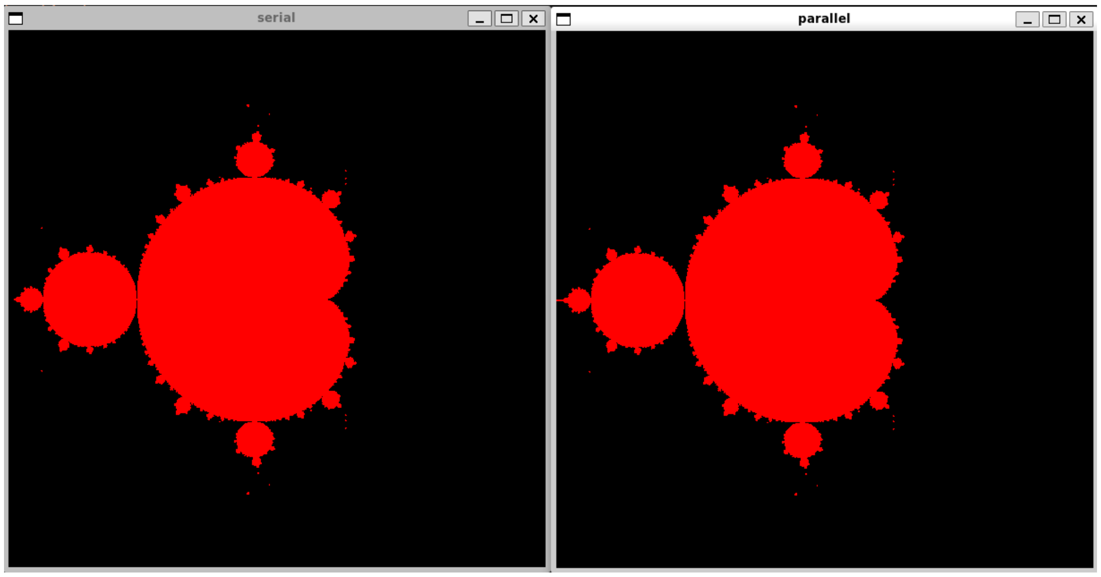
Mohamad Yahyapour

Mohammad Moien Joneidi Jaafari

Part 1

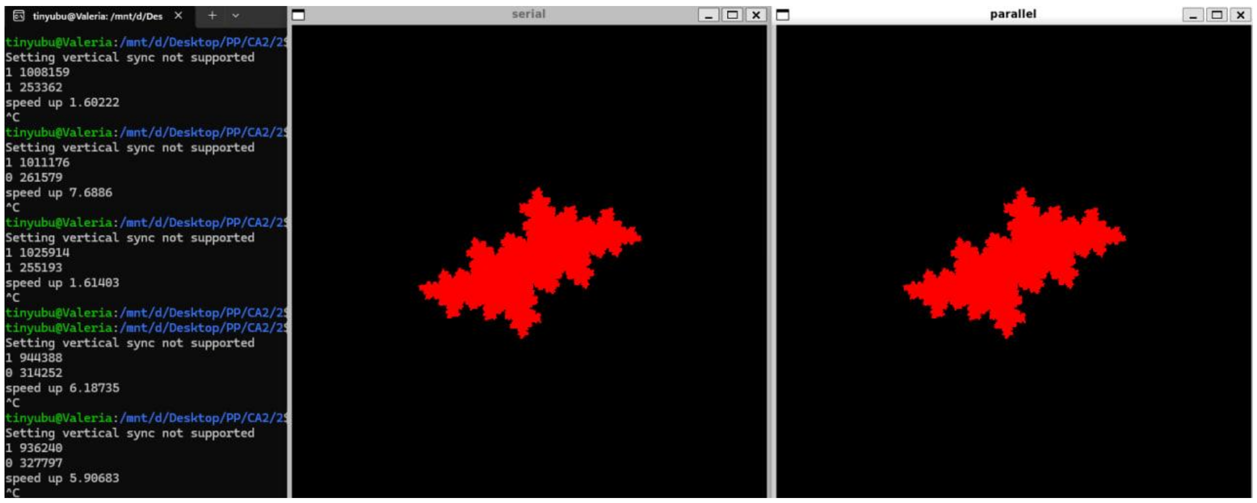


```
Setting vertical sync not supported
1 724588
1 271755
speed up 1.35607
a
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ g++ one.cpp -fo
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ ./a.out
Setting vertical sync not supported
0 738331
0 267205
speed up 2.76316
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ ./a.out
Setting vertical sync not supported
1 727402
0 263318
speed up 6.56014
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ ./a.out
Setting vertical sync not supported
1 711949
0 278501
speed up 6.14701
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ ./a.out
Setting vertical sync not supported
1 702305
0 263263
speed up 6.46618
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/1$ ./a.out
```

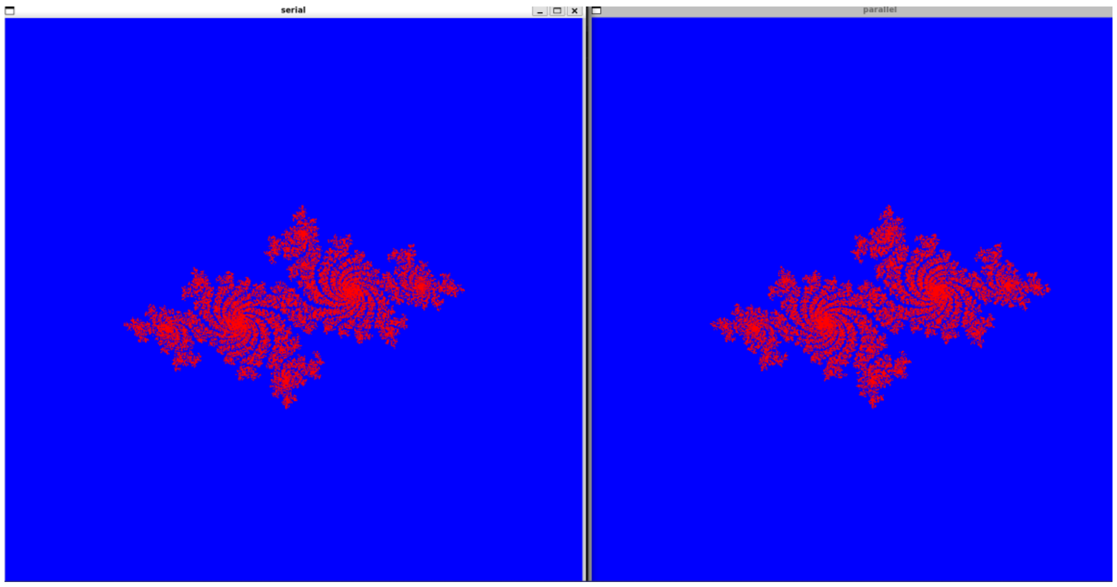


The **speedup** achieved was approximately **6**, which was obtained using **8 hyper-threaded cores**. The reason it is less than 8 is that **drawing on the screen**, performed using the **SFML library**, is in a **critical section**, so the cores could not simultaneously add a point found as a convergence point to the screen. In the second image, as shown, the image is **zoomed in**.

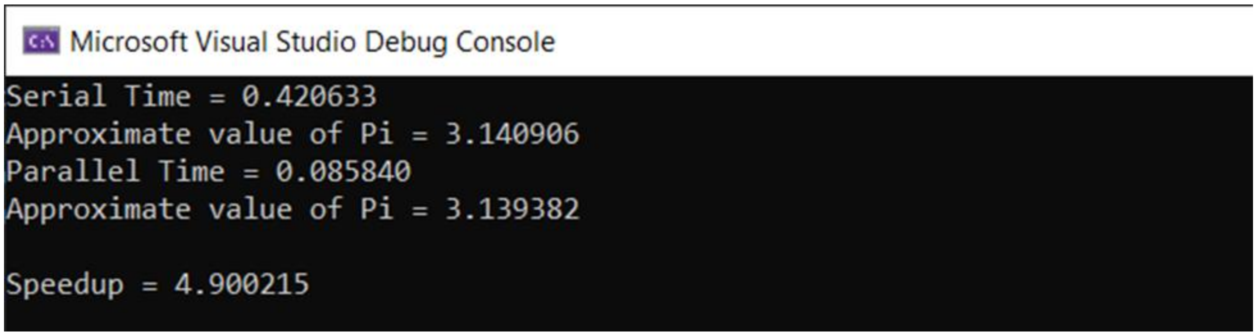
Part 2



```
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
Setting vertical sync not supported
1 1008159
1 253362
speed up 1.60222
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
Setting vertical sync not supported
1 1011176
0 261579
speed up 7.6886
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
Setting vertical sync not supported
1 1025914
1 255193
speed up 1.61403
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
Setting vertical sync not supported
1 944388
0 314252
speed up 6.18735
^C
tinyubu@Valeria:/mnt/d/Desktop/PP/CA2/2
Setting vertical sync not supported
1 936240
0 327797
speed up 5.90683
^C
```

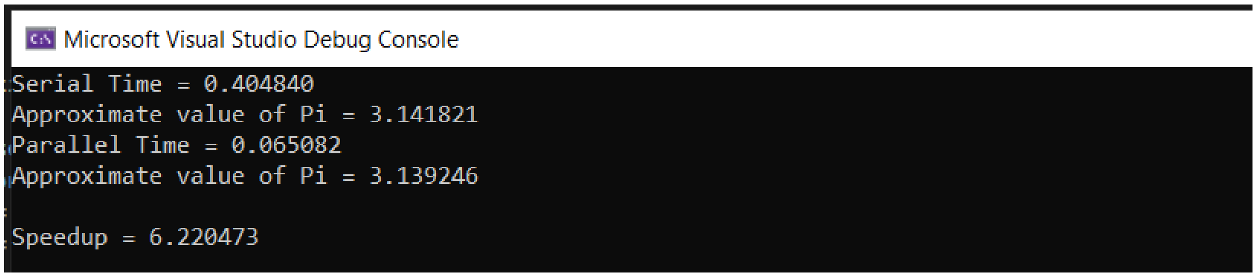


This section is similar to the previous one, with the only difference being that the value of **c** is always **1**, while the initial value of **z** determines whether the point converges or not. The **speedup** in this section is also approximately **6.** Due to the similarity of the code with the previous section and the presence of a **critical section**, the speedup falls short of the maximum possible value of 8.



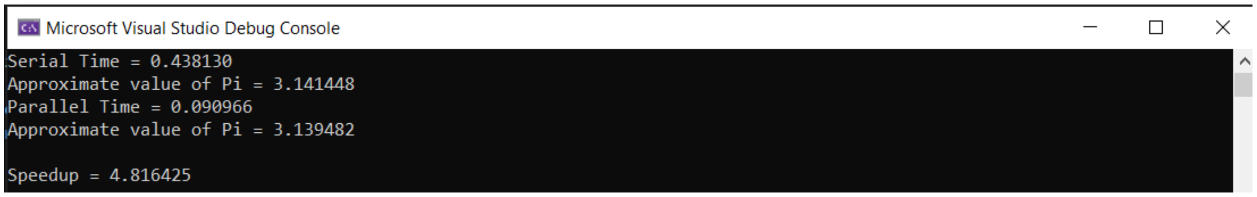
```
Serial Time = 0.420633
Approximate value of Pi = 3.140906
Parallel Time = 0.085840
Approximate value of Pi = 3.139382

Speedup = 4.900215
```



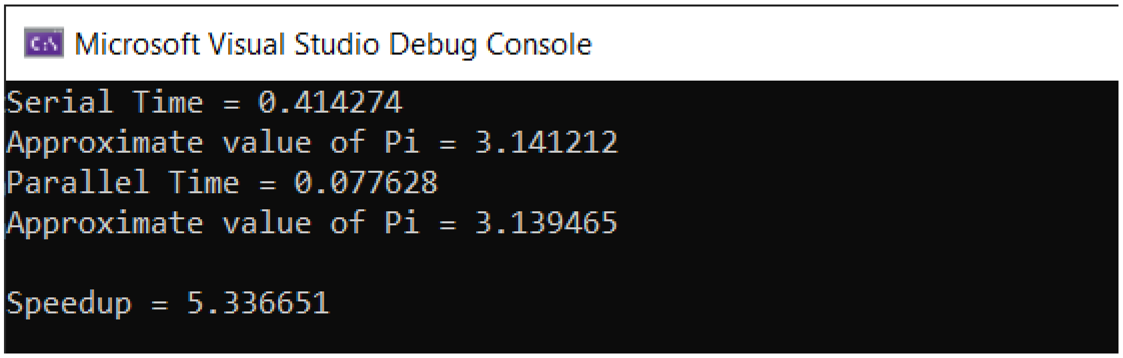
```
Serial Time = 0.404840
Approximate value of Pi = 3.141821
Parallel Time = 0.065082
Approximate value of Pi = 3.139246

Speedup = 6.220473
```



```
Serial Time = 0.438130
Approximate value of Pi = 3.141448
Parallel Time = 0.090966
Approximate value of Pi = 3.139482

Speedup = 4.816425
```



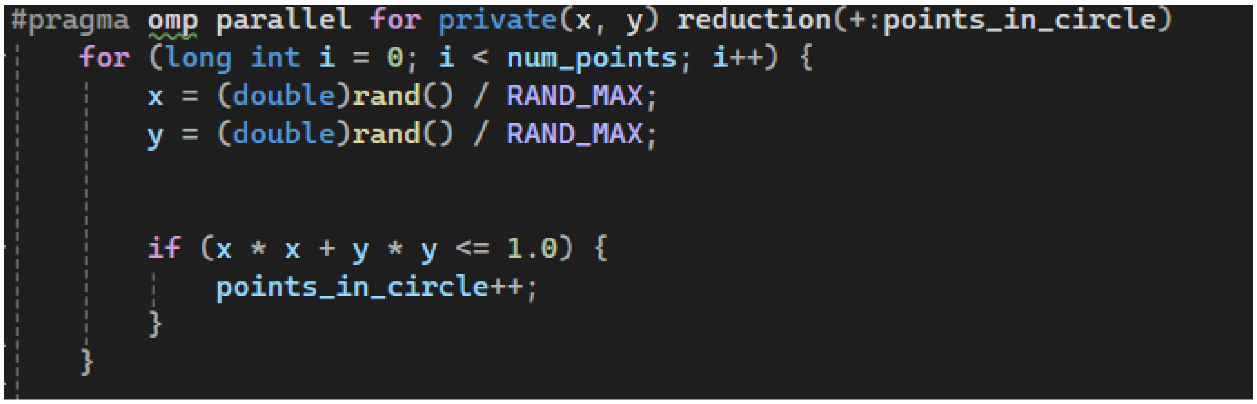
```
Serial Time = 0.414274
Approximate value of Pi = 3.141212
Parallel Time = 0.077628
Approximate value of Pi = 3.139465

Speedup = 5.336651
```

Part 3

The **average speedup** is approximately **5.5**.

In the **Monte Carlo method**, we consider a **circle with a radius of 1**, and around it, we inscribe a **square with side length 2**.



```
#pragma omp parallel for private(x, y) reduction(+:points_in_circle)
    for (long int i = 0; i < num_points; i++) {
        x = (double)rand() / RAND_MAX;
        y = (double)rand() / RAND_MAX;


        if (x * x + y * y <= 1.0) {
            points_in_circle++;
        }
    }
```

Then, **random points** are placed inside this square. The probability of a point falling inside the circle is equal to the **area of the circle divided by the area of the square**.

Therefore, a **large number of points** are placed, and if a point falls inside the circle, the variable **points_in_circle** is incremented by one. Finally, the **ratio of points inside the circle to the total points** is multiplied by the area of the square, which is 4, and we expect to obtain the value of **π** as the output.

The **parallelized section** is the same as the serial section, with the only difference being that a **parallelization directive** is called before the loop.