Rehan Tariq
22i-0965
CS-6A

Report for Question 2

OpenMP Case Study: Drawing a Circle with Parallel Processing

1. Introduction

This report covers the implementation of a circle-drawing program that leverages OpenMP for parallel processing in C++. The program computes the points of a circle by varying an angle and calculating the corresponding x and y coordinates. Two methods are used:

1. **Using the math library** for sine and cosine.
2. **Using Taylor series** approximations for sine and cosine.

The objectives are to:

- Compare serial vs. parallel performance.
- Demonstrate parallelization of both the circle point computation and the Taylor series terms.
- Draw the circle using the SDL2 graphics library.

2. Approach

2.1 Circle Computation

- A loop iterates over a range of angles, computing the x and y positions for each point on the circle.

- In the **serial** approach, a single thread processes all angles sequentially.
- In the **parallel** approach, OpenMP divides the angles among multiple threads.

2.2 Taylor Series Approximation

- Custom functions approximate sine and cosine through iterative summations.
- The summation of terms is also parallelized with OpenMP, distributing the work among available threads.

2.3 Rendering

- SDL2 is used to create a window and a renderer.
- The computed points are plotted on the screen, producing a smooth circle.
- Different colors are used to distinguish between serial vs. parallel methods (and math library vs. Taylor series).

2.4 Timing and Speedup

- Execution times are recorded using `omp_get_wtime()`.
- Speedup is calculated by dividing the serial time by the parallel time for each method.

3. Implementation Details

1. **Data Structures**:
    - Vectors store the computed (x, y) points for both the math library and Taylor series methods.
    - Each vector holds the results of either the serial or parallel approach.

2. **Parallel Directives**:
    - `#pragma omp parallel for` is applied to loops that compute circle points.
    - The Taylor series summation uses a parallel loop, often combined with a reduction clause to safely accumulate partial sums.
3. **SDL2 Integration**:
    - The program initializes SDL2, creates a window and renderer, and clears the screen.
    - Each point is drawn onto the renderer. Finally, the updated screen is displayed to confirm a smooth circle.
4. **Parameters**:
    - **Number of Points**: The angle range (0 to 360 degrees) is split into many discrete steps (e.g., 1000 or more).
    - **Taylor Series Terms**: Tests are run with 1500, 15000, and 150000 terms to gauge how performance scales.

4. Results and Performance Analysis

4.1 Visual Output

In all tests, the program produces a circle in the SDL window. The circle remains smooth and centered, confirming the correctness of the coordinate calculations.

4.2 Timing Results

The table below summarizes the measured times (in milliseconds) for three different runs using **1500**, **15000**, and **150000** Taylor series terms. Each run compares:

- Serial Math Library vs. Parallel Math Library
- Serial Taylor Series vs. Parallel Taylor Series

TAYLOR_TERMS	Serial Math Library (ms)	Parallel Math Library (ms)	Speedup (Math Library)	Serial Taylor (ms)	Parallel Taylor (ms)	Speedup (Taylor Series)
1500	0.029493	0.124322	0.237213x	10.0246	4.53553	2.21023x
15000	0.030022	0.106094	0.29815x	86.244	25.8555	3.33562x
150000	0.029757	0.128775	0.231077x	718.71	214.315	3.33522x

Observations

1. **Math Library Approach**
   - The parallel version sometimes runs slower or only slightly faster than the serial version. This can happen when the overhead of creating threads outweighs the minimal computation needed per loop iteration.
   - The small time measurements (well under 1 ms) make overhead more significant compared to actual work.
2. **Taylor Series Approach**
   - As the number of terms increases, the parallel approach consistently outperforms the serial approach by a factor of roughly 2–3 times.
   - The larger workload in the Taylor series summation benefits more from parallelization.
3. **Thread Usage**
   - The program detects 4 available threads in these runs, distributing the workload accordingly.

5. Discussion

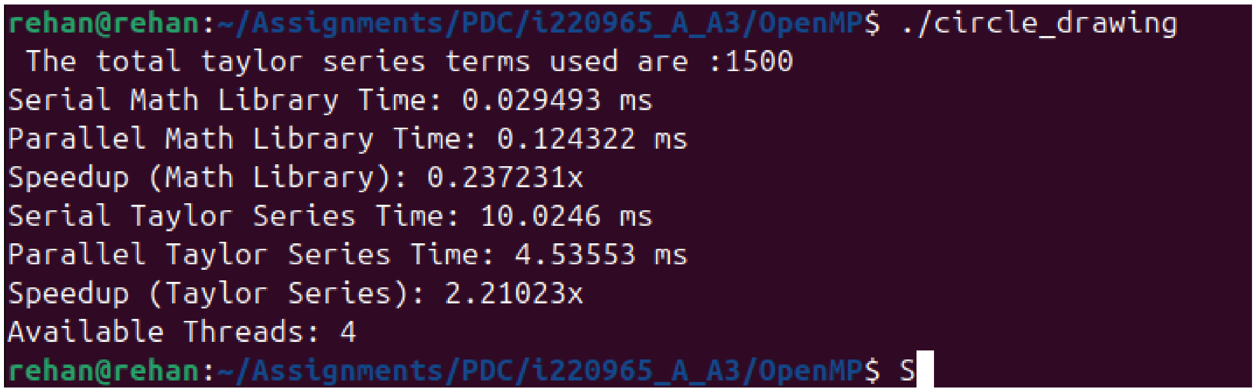
- **Correctness**:
  The circle is rendered correctly, indicating that both math library and Taylor series computations produce valid results for sine and cosine.

- **Performance**:
    - For simple tasks (like the math library calls), parallelization overhead may limit speedup.
    - For more computationally demanding tasks (like the Taylor series), parallelization provides clear benefits.
- **Scalability**:
  Increasing the number of Taylor series terms significantly increases the workload, making parallelization more beneficial.
- **Potential Improvements**:
    - Adaptive scheduling in OpenMP could be tested to balance workload more efficiently.
    - Reducing the number of Taylor series terms for small angles could further optimize runtime without harming accuracy significantly.

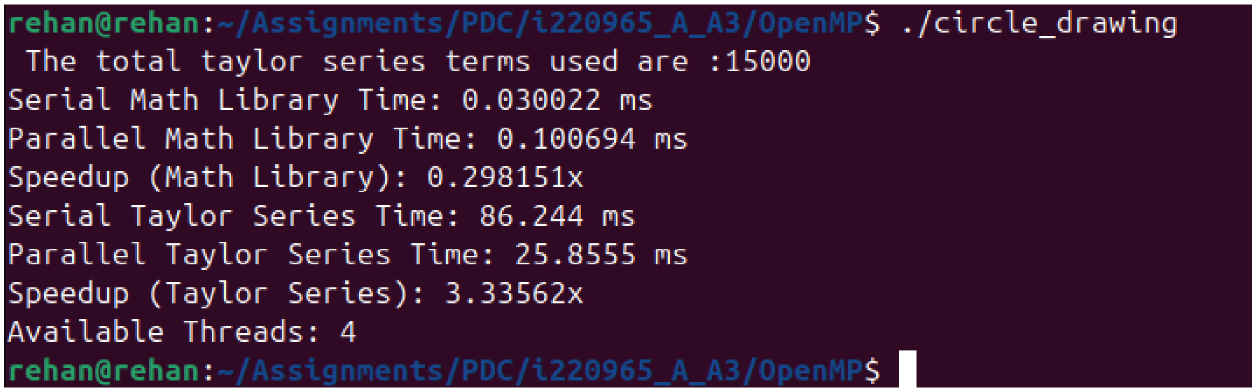
6. Conclusion

This project demonstrates the advantages of using OpenMP to parallelize both the coordinate computation for drawing a circle and the Taylor series approximations for sine and cosine. The final outputs confirm the correctness of the circle drawing, while the timing results show that parallelization yields notable speedups for more intensive tasks. Although the math library approach may not always benefit greatly from parallelization (due to overhead), the Taylor series computations exhibit consistent performance gains as the computational load increases.

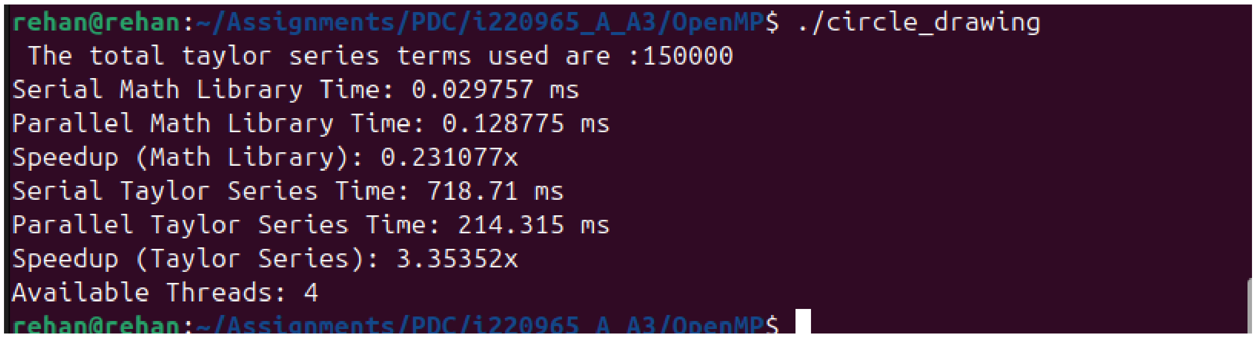
Screenshots:



```
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$ ./circle_drawing
 The total taylor series terms used are :1500
Serial Math Library Time: 0.029493 ms
Parallel Math Library Time: 0.124322 ms
Speedup (Math Library): 0.237231x
Serial Taylor Series Time: 10.0246 ms
Parallel Taylor Series Time: 4.53553 ms
Speedup (Taylor Series): 2.21023x
Available Threads: 4
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$ S
```



```
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$ ./circle_drawing
 The total taylor series terms used are :15000
Serial Math Library Time: 0.030022 ms
Parallel Math Library Time: 0.100694 ms
Speedup (Math Library): 0.298151x
Serial Taylor Series Time: 86.244 ms
Parallel Taylor Series Time: 25.8555 ms
Speedup (Taylor Series): 3.33562x
Available Threads: 4
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$
```



```
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$ ./circle_drawing
 The total taylor series terms used are :150000
Serial Math Library Time: 0.029757 ms
Parallel Math Library Time: 0.128775 ms
Speedup (Math Library): 0.231077x
Serial Taylor Series Time: 718.71 ms
Parallel Taylor Series Time: 214.315 ms
Speedup (Taylor Series): 3.35352x
Available Threads: 4
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenMP$
```