Parallel and Distributed Computing

Assignment 4

102303654

Exercise 1:

1. Problem Statement

Write an MPI program in a ring topology where Process 0 starts with value 100 and sends it to the next process. Each process adds its own rank to the received value before passing it to the next process. The last process sends the value back to Process 0.

2. Output Obtained

```
Process 0 starts with value 100
Process 1 received 100
Process 1 added rank, now 101
Process 2 received 101
Process 2 added rank, now 103
Process 3 received 103
Process 3 added rank, now 106
Process 0 received FINAL value: 106
```

3. Observations

- The message successfully travelled in a ring
- Each process correctly added its rank (0+1+2+3 = 6).
- Final value = 100 + 6 = 106, which matches the expected result.
- Point-to-point communication (MPI_Send and MPI_Recv) works reliably with proper use of modulo for next/previous rank.

4. Conclusion

This exercise helped understand cyclic communication and how to handle wrap around in distributed processes using modular arithmetic.

Exercise 2:

1. Problem Statement

Create an array of numbers 1 to 100 in Process 0. Divide the array equally among all processes using MPI_Scatter, compute local sums, and combine them using MPI_Reduce to get the global sum. Also calculate the average.

2. Output Obtained

```
Process 0 local sum = 325
Process 1 local sum = 950
Process 2 local sum = 1575
Process 3 local sum = 2200
Global Sum = 5050 (Correct: 5050)

Average = 50.5
```

3. Observations

- The array was perfectly divided (25 elements per process).
- Local sums added up exactly to 5050, which is the correct sum of first 100 natural numbers.
- MPI_Scatter and MPI_Reduce worked efficiently for data distribution and aggregation.
- Bonus average calculation was also correct (5050/100 = 50.5).

4. Conclusion

This exercise demonstrated how collective communication primitives can be used to parallelize simple reduction operations like summation, which is very useful in real scientific computing.

Exercise 3:

1. Problem Statement

Each process generates 10 random numbers (0–1000), finds its local maximum and minimum, and then uses MPI_Reduce with MPI_MAXLOC and MPI_MINLOC to find the global maximum and minimum along with the process ID where they occurred.

2. Output Obtained

```
Process 0 → Local Max: 813 | Local Min: 11
Process 2 → Local Max: 980 | Local Min: 113
Process 1 → Local Max: 939 | Local Min: 144
Process 3 → Local Max: 883 | Local Min: 54
FINAL RESULT:-
Global Maximum = 980 (from Process 2)
Global Minimum = 11 (from Process 0)
```

3. Observations

- All processes generated different random numbers (due to different seeds).
- MPI_MAXLOC and MPI_MINLOC correctly returned both the extreme value and the rank of the process.
- Print order is jumbled because processes run in parallel and print at slightly different times — this is normal behaviour in MPI.
- Global results are accurate.

4. Conclusion

This exercise showed the power of special reduction operations (MAXLOC/MINLOC) to not only find the value but also identify which process produced it.

Exercise 4:

1. Problem Statement

Compute the dot product of two vectors A = [1,2,3,4,5,6,7,8] and B = [8,7,6,5,4,3,2,1] in parallel. Scatter both vectors, compute partial dot products in each process, and combine them using MPI_Reduce.

2. Output Obtained

```
Process 1 partial dot = 38
Process 2 partial dot = 38
Process 0 partial dot = 22
Process 3 partial dot = 22
Final Dot Product = 120 (Correct: 120)
```

3. Observations

- Vectors were correctly divided among 4 processes (2 elements each).
- Partial results: 22 + 38 + 38 + 22 = 120 (exact expected value).
- MPI_Scatter and MPI_Reduce handled data distribution and summation efficiently.
- Print order was jumbled due to parallel execution.

4. Conclusion

This exercise demonstrated a practical application of parallel matrix/vector operations, which are widely used in machine learning and scientific simulations.