Exercise 2: Parallelization of a numerical integration using OpenMP Run Report

Number of Threads	Number of Internals	Sequential Run	Parallel Run 1	Parallel Run 2	Parallel Run 3
2	1000000	2874	1547	1524	1432
	2000000	5656	2872	3052	2882
	3000000	8133	4191	4352	4166
	3700000	11334	5898	6418	5989
	4200000	12418	6292	6667	6072
3	1000000	3061	1056	1140	995
	2000000	6057	2276	2951	2611
	3000000	8531	2908	3228	2910
	3700000	10460	3625	3879	3663
	4200000	12527	4186	4484	4345
4	1000000	2975	997	914	817
	2000000	5905	1517	1619	1499
	3000000	8860	2419	2649	2702
	3700000	11506	3033	3454	2813
	4200000	12539	3187	3322	3137

Why OpenMP reduction does not compute exactly the same result as the sequential code?

➔ We know, In OpenMP the nature of parallelized loop code results in the sum additions being performed in an unfixed order and “Float value” operations are not commutive by nature, Due to this causes a slightly unpredictable value in decimal.1