Name: Monish Kumar V Roll.no: CE18B118

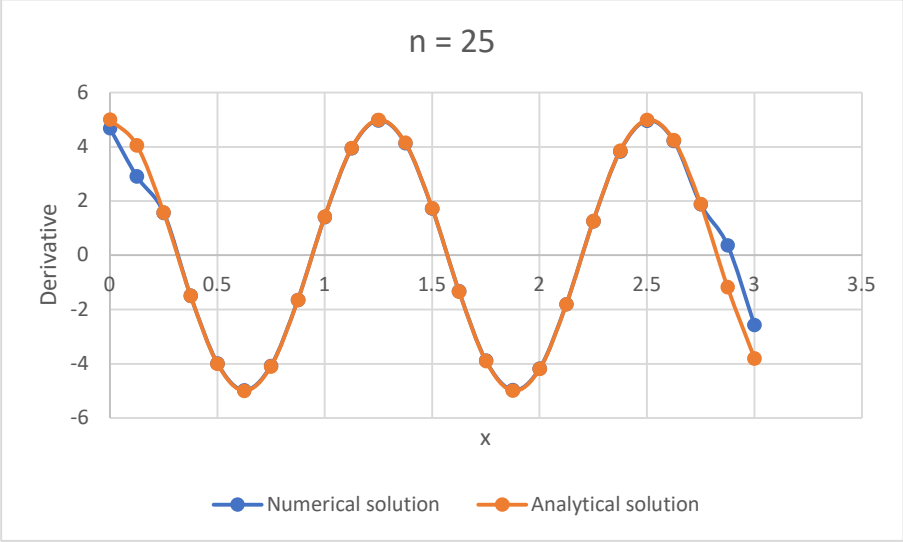
ID5130 - Parallel Scientific Computing
Assignment - 2

1.a)

Status of code: runs-and-gives-correct-result

The following plot represents the obtained numerical and analytical solution (in a serial manner) as a function of x using fourth-order accurate central difference formula for the interior nodes and one-sided forward/backward finite-difference formula for the boundary and near-boundary points.

Both the numerical and analytical solution plots are overlapping one another (therefore, the obtained derivative is correct) except boundary and near boundary points because accuracy of forward/backward finite-difference formula is less when compared with fourth-order accurate central difference formula.

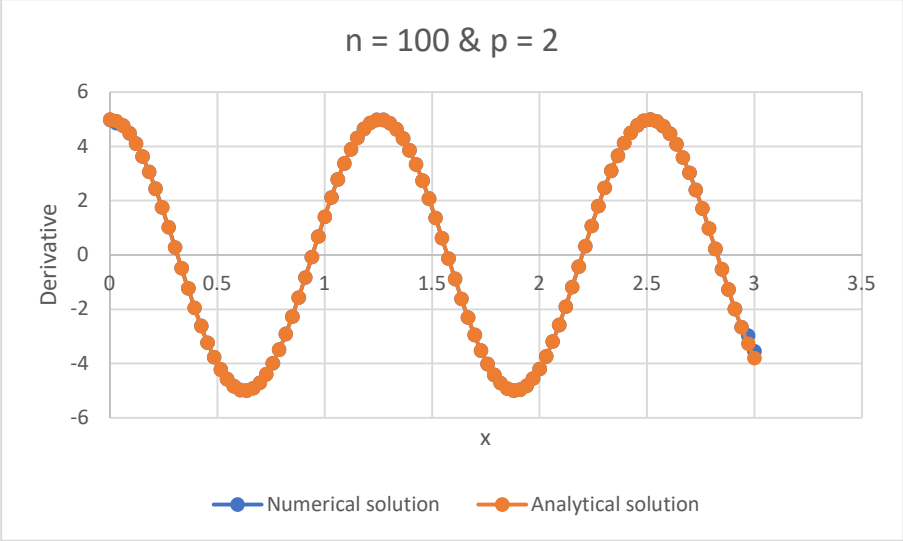


1.b)

Status of code: runs-and-gives-correct-result

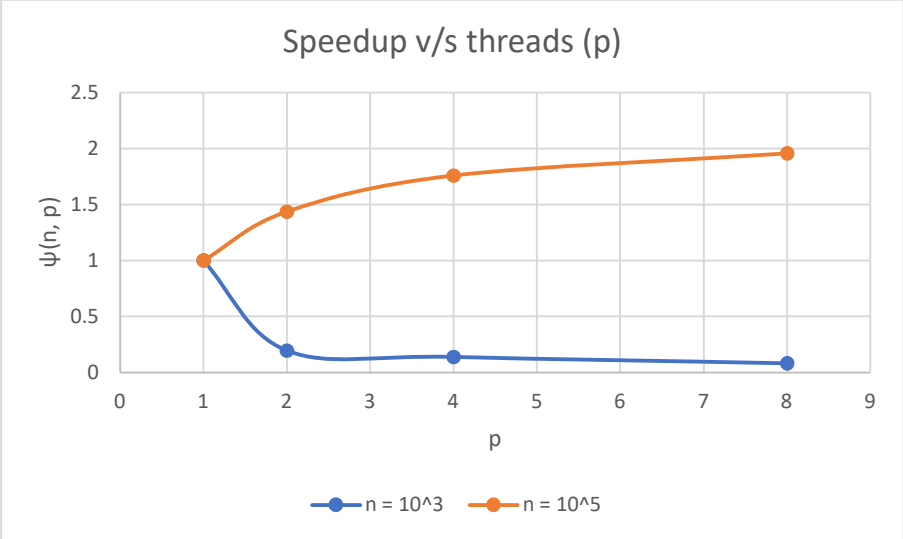
The following plot represents the obtained numerical and analytical solution (in a parallel manner using 2 threads) as a function of x using fourth-order accurate central difference formula for the interior nodes and one-sided forward/backward finite-difference formula for the boundary and near-boundary points.

Both the numerical and analytical solution plots are overlapping one another (therefore, the obtained derivative is correct) except boundary and near boundary points because accuracy of forward/backward finite-difference formula is less when compared with fourth-order accurate central difference formula.



The following table represents the execution time in seconds.

	Serial	p = 2	p = 4	p = 8
$n = 10^3$	0.000081	0.000416	0.000587	0.00098
$\psi(n, p)$	1	0.194712	0.13799	0.082653
$n = 10^5$	0.007211	0.005018	0.004101	0.003684
$\psi(n, p)$	1	1.437027	1.758352	1.957383



Observations from the above plot:

For n = 10^5, as expected, the speedup increases as no. of threads increase. But for n = 10^3, the speedup decreases as no. of threads increase and the speedup is always less than one. This maybe because the parallel overhead time greater than the sequential execution time and this increases as p increases.

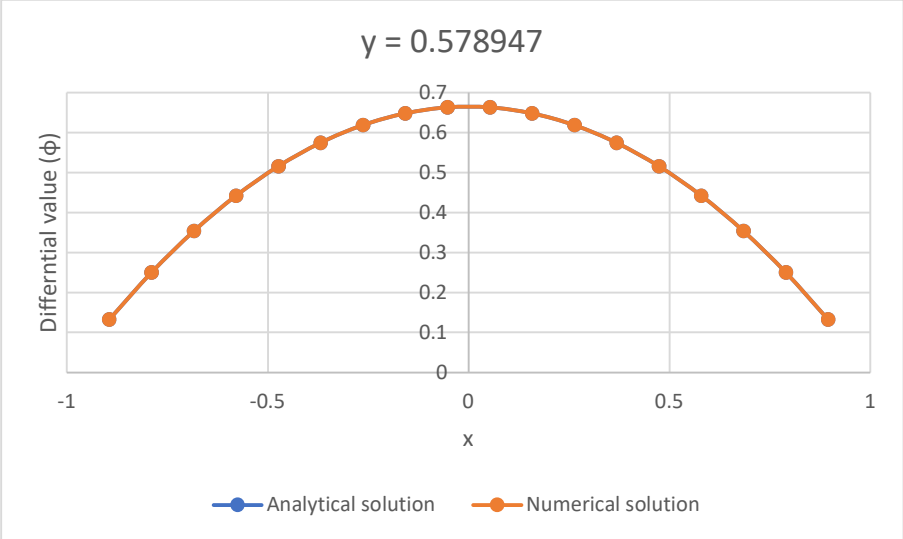
2.a)

Status of code: runs-and-gives-correct-result

For ease of calculation purposes, n = 20 was taken instead of 21 (so that n is divisible by p = 2). Therefore, Δ = Δx = Δy = 0.105263 (approx.).

The following plot represents the obtained numerical and analytical solution (in a parallel manner using 2 threads) as a function of x for y = 0.578947 (instead of y = 0.5) using Jacobi iterative method. Both the numerical and analytical solution plots are overlapping one another. Therefore, the obtained derivative is correct.

Th no. of iterations required to bring the numerical solution to within 1% of the exact solution is ***751***.



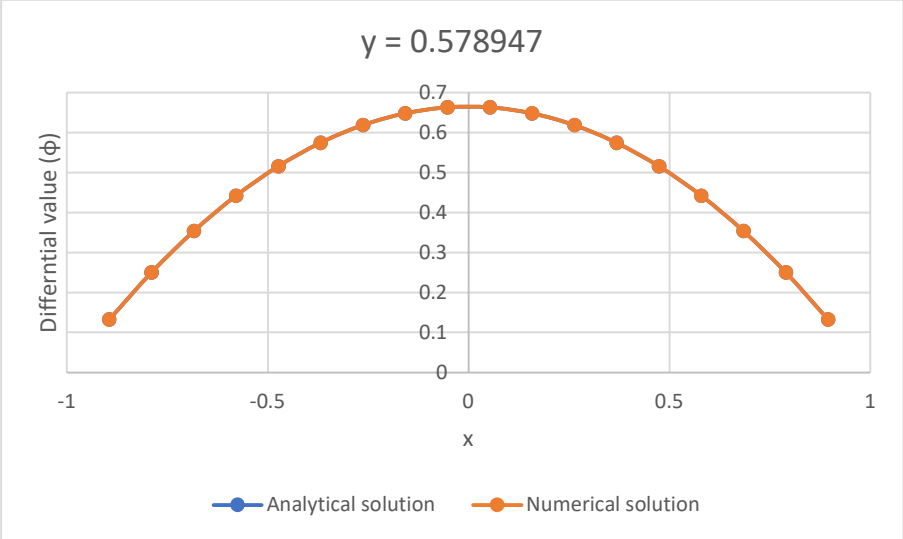
2.b)

Status of code: runs-and-gives-correct-result

For ease of calculation purposes, n = 20 was taken instead of 21 (so that n is divisible by p = 2). Therefore, $\Delta = \Delta x = \Delta y = 0.105263$ (approx.).

The following plot represents the obtained numerical and analytical solution (in a parallel manner using 2 threads) as a function of x for y = 0.578947 (instead of y = 0.5) using Red-black coloring approach. Both the numerical and analytical solution plots are overlapping one another. Therefore, the obtained derivative is correct.

Th no. of iterations required to bring the numerical solution to within 1% of the exact solution is ***396*** which is way less than what we obtained using Jacobi iterative method (751). Therefore, is Red-black coloring approach faster than Jacobi iterative method.



2.c)

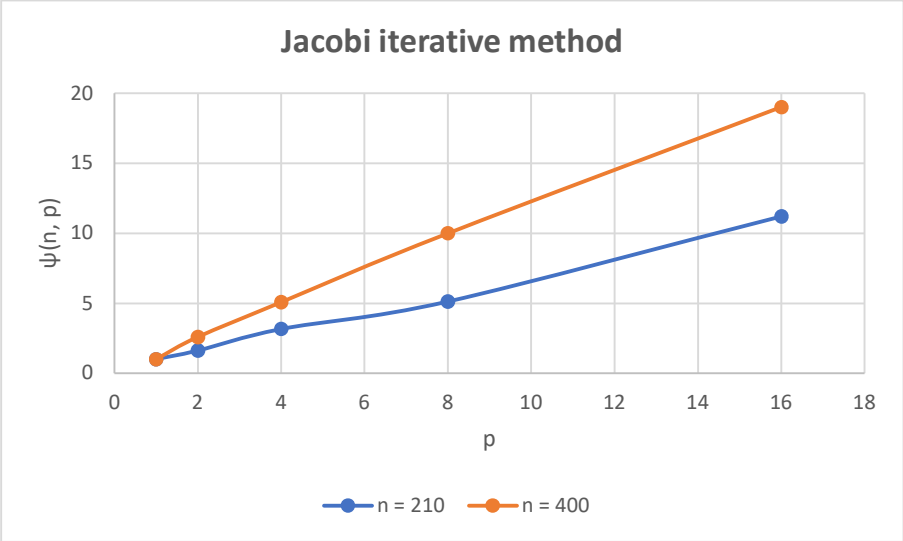
The following n values include the 2 boundary conditions and the units of execution time is in seconds.

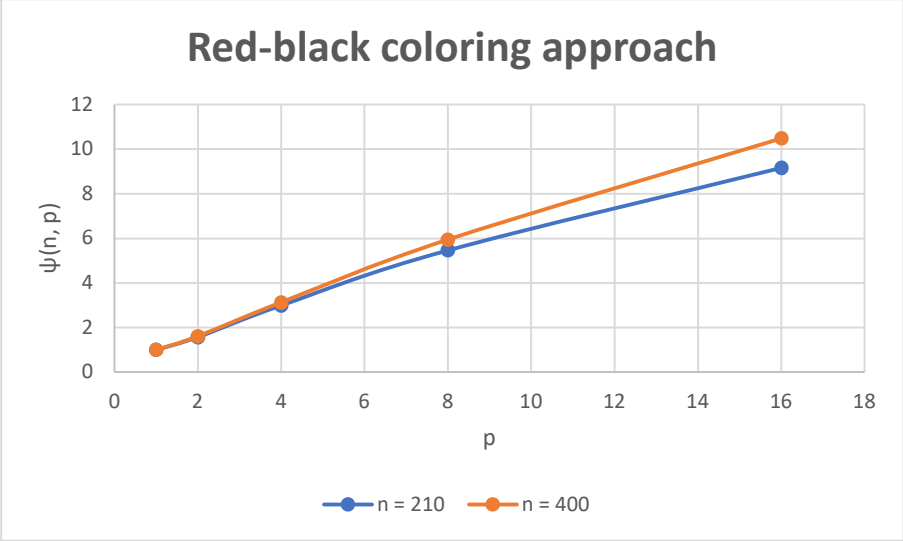
Jacobi iterative method:

	Serial	p = 2	p = 4	p = 8	p = 16
n = 210	1303.253364	808.285803	411.657521	253.982133	116.237668
ψ(n, p)	1	1.612367	3.165868	5.13128	11.21197
n = 400	26320.275279	10173.647912	5189.872882	2631.802139	1383.987322
ψ(n, p)	1	2.587103	5.071468	10.00086	19.01771

Red-black coloring approach:

	serial	p = 2	p = 4	p = 8	p = 16
n = 210	594.137095	378.629228	198.671239	108.721982	64.821985
ψ(n, p)	1	1.569179	2.990554	5.464738	9.165673
n = 400	8310.727920	5160.124287	2657.098922	1397.981257	792.669203
ψ(n, p)	1	1.610567	3.127745	5.944806	10.48448





From the above plot and tables, it is evident that Red-black coloring approach is way faster than Jacobi iterative method. Also, there is a significant improvement in performance because speedup increases as no. of threads increases.