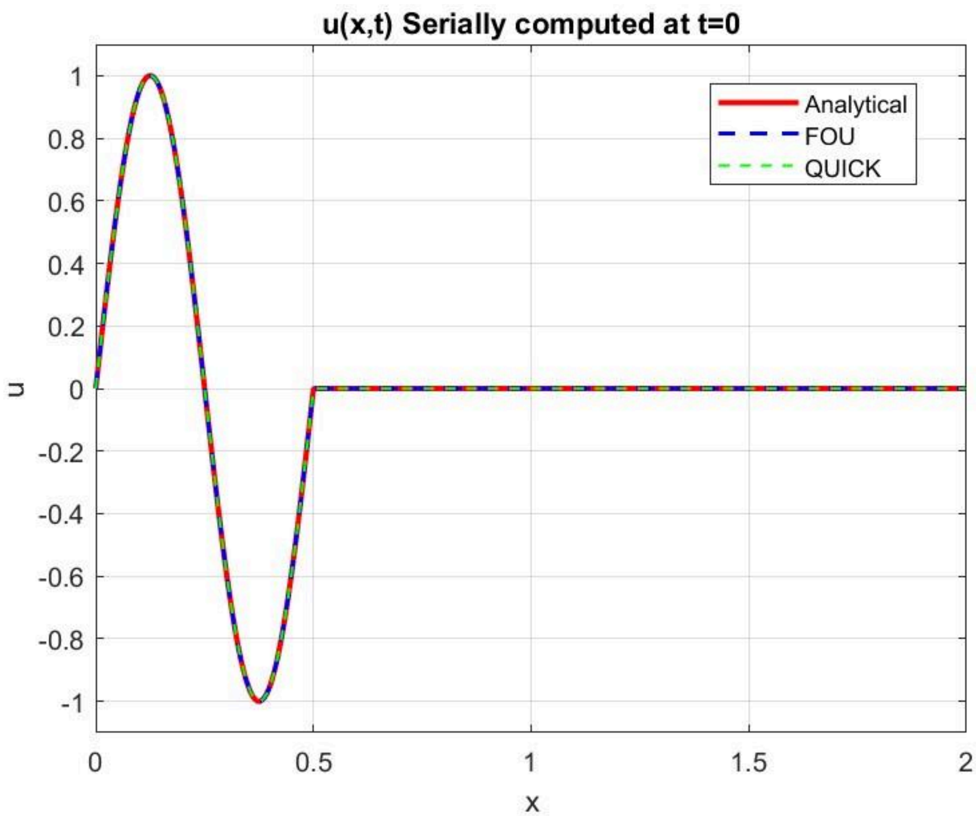
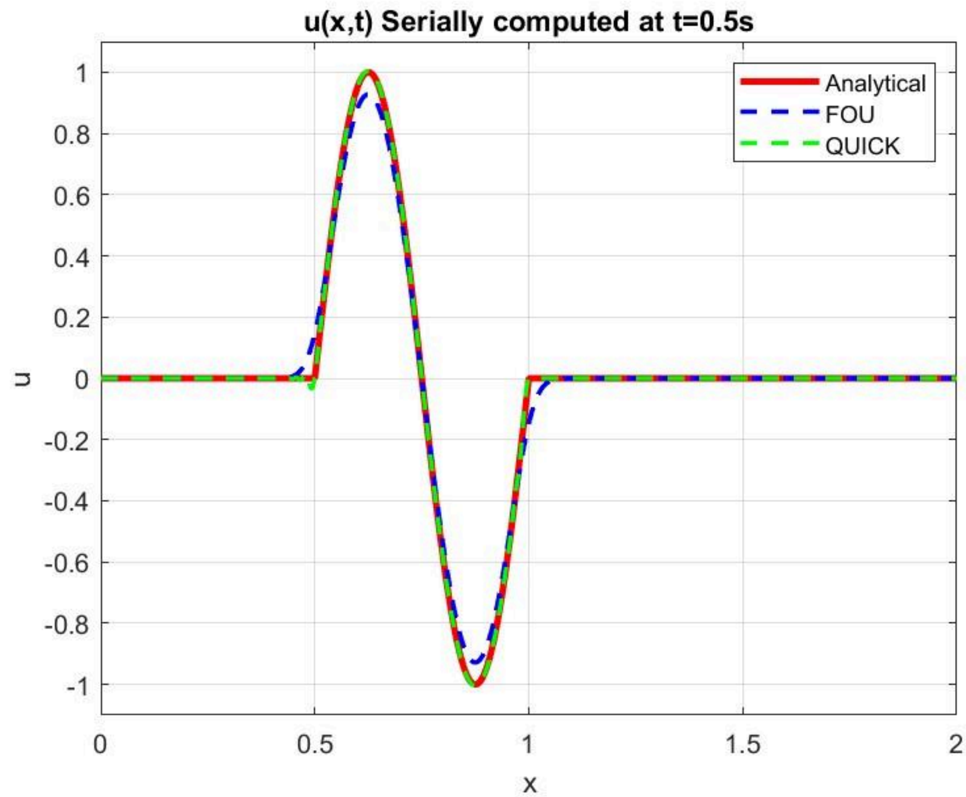
ID5130 - Assignment 2

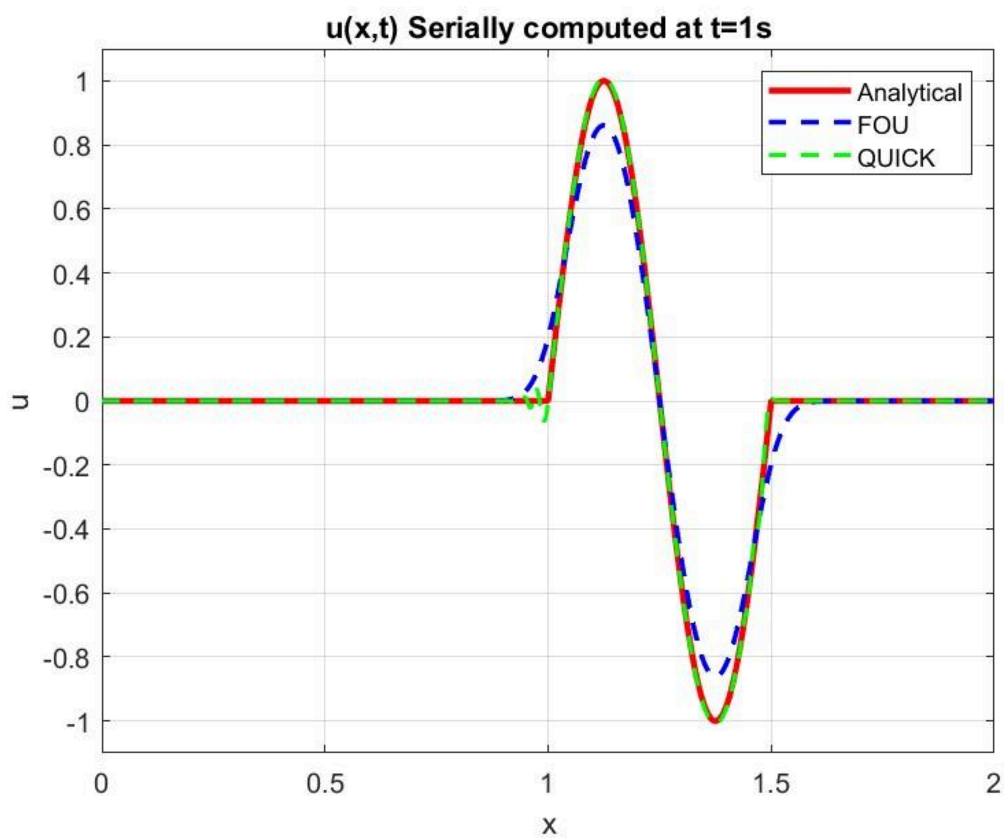
Karthik Sriram
MM20B032

Q1.Computationally solving the first-order traveling wave equation using first-order upwind scheme and QUICK scheme.

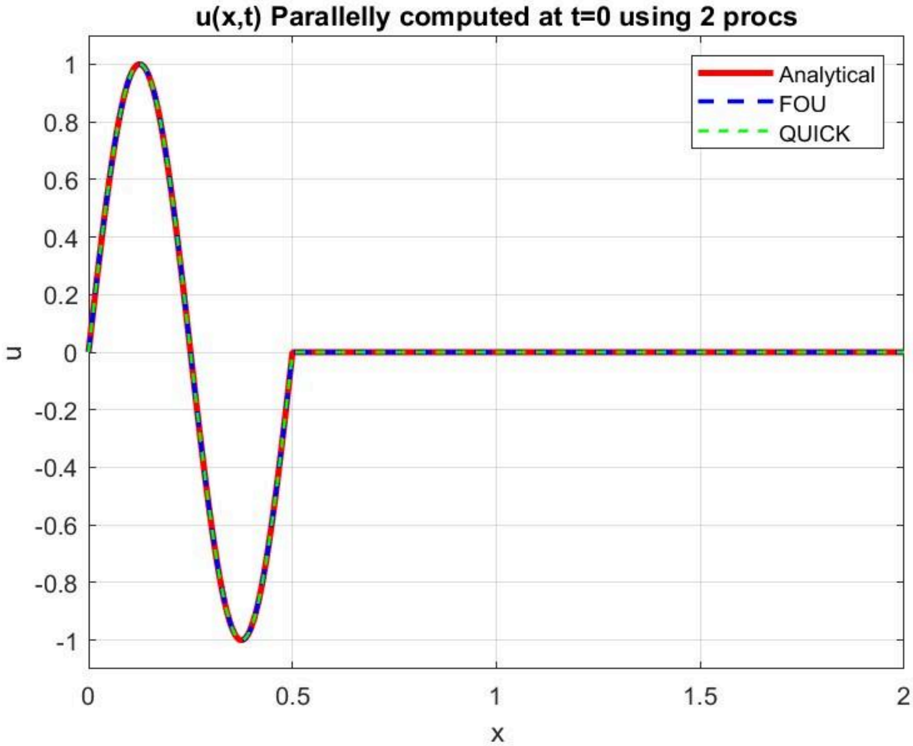
a)



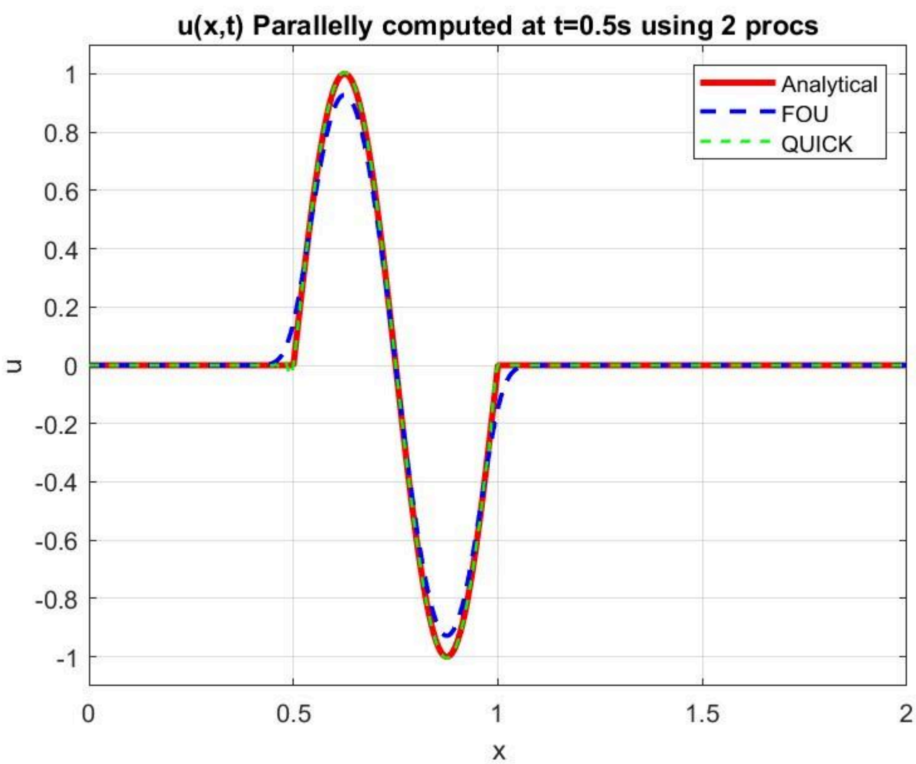

u(x,t) Serially computed at t=0.5s
Analytical
FOU
QUICK
u
x
0
0.5
1
1.5
2
1
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
-1


u(x,t) Serially computed at t=1s
Analytical
FOU
QUICK
u
x
0
0.5
1
1.5
2
1
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
-1

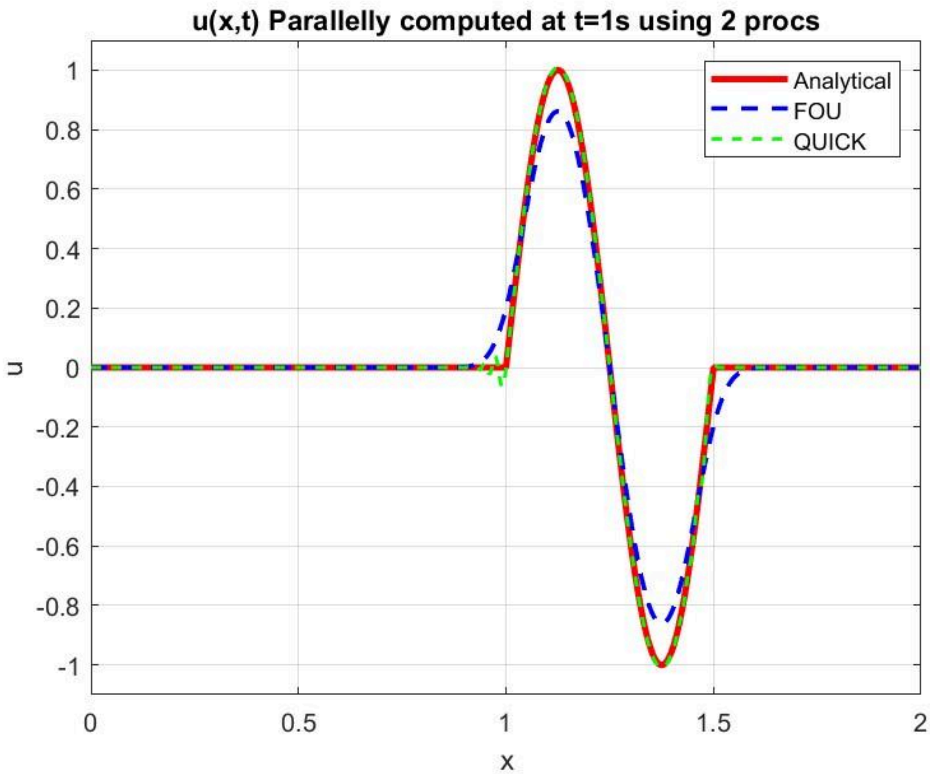
b)

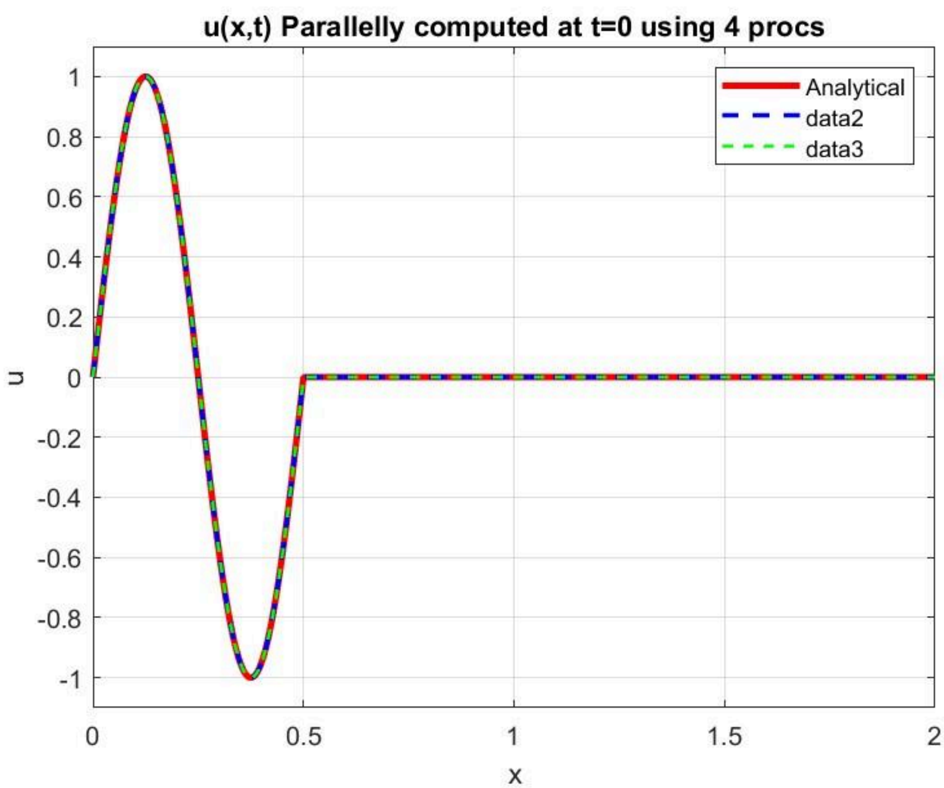


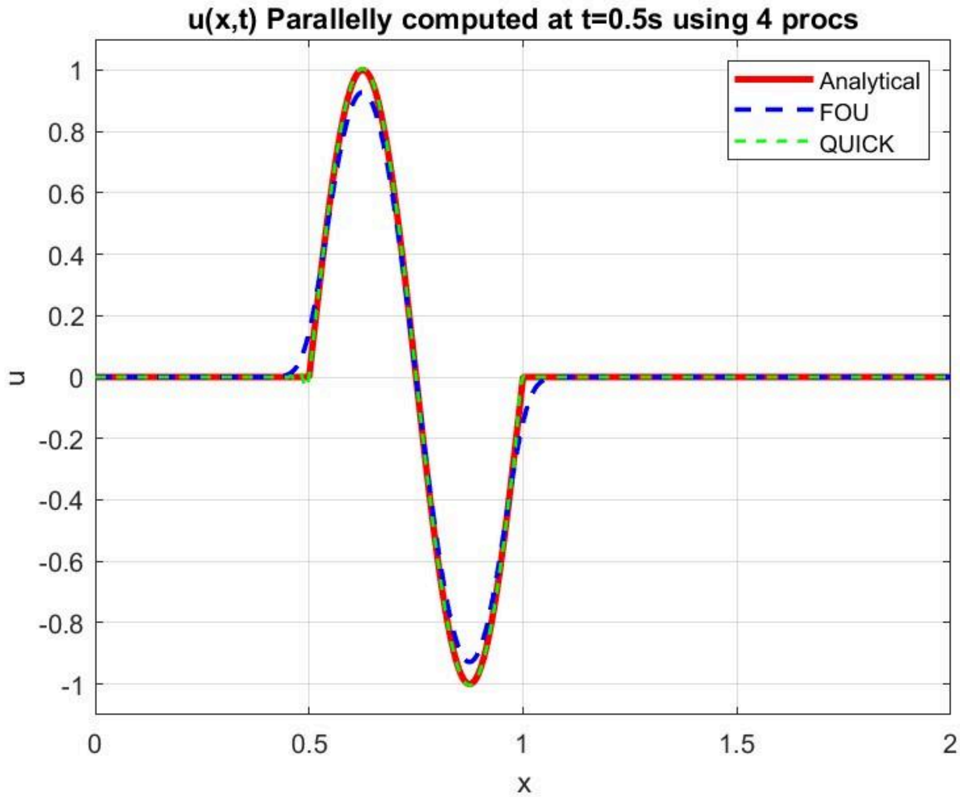
u(x,t) Parallelly computed at t=0 using 2 procs
Analytical
FOU
QUICK
u
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2

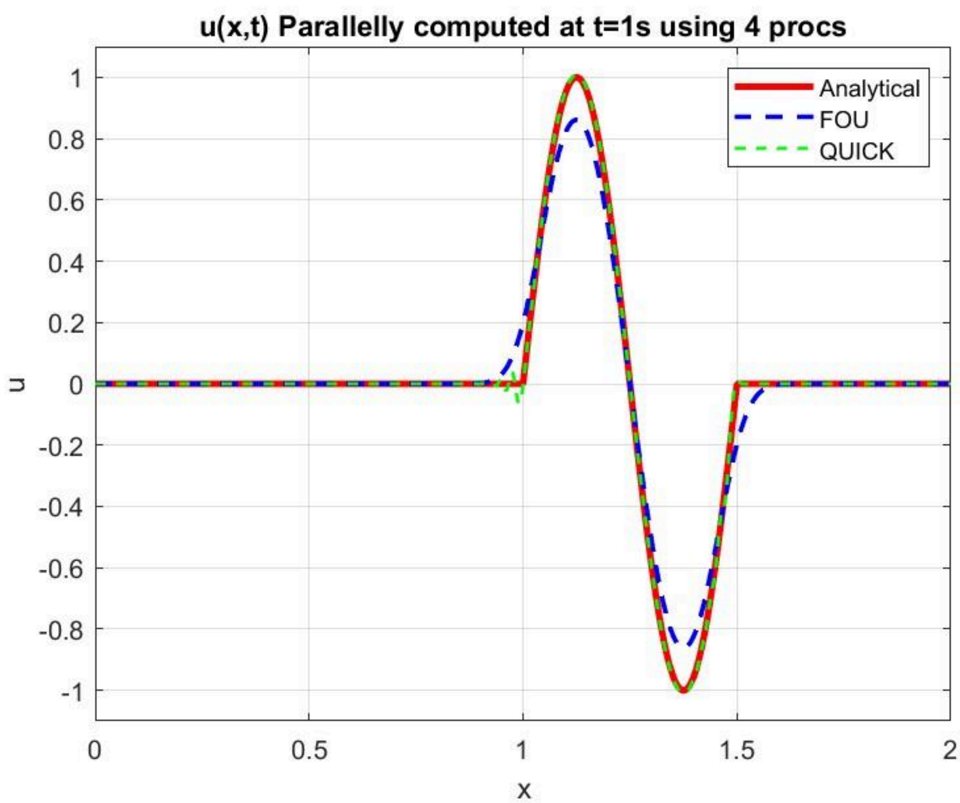


u(x,t) Parallelly computed at t=0.5s using 2 procs
Analytical
FOU
QUICK
u
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2


u(x,t) Parallelly computed at t=1s using 2 procs
Analytical
FOU
QUICK
u
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2


u(x,t) Parallelly computed at t=0 using 4 procs
Analytical
data2
data3
u
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2

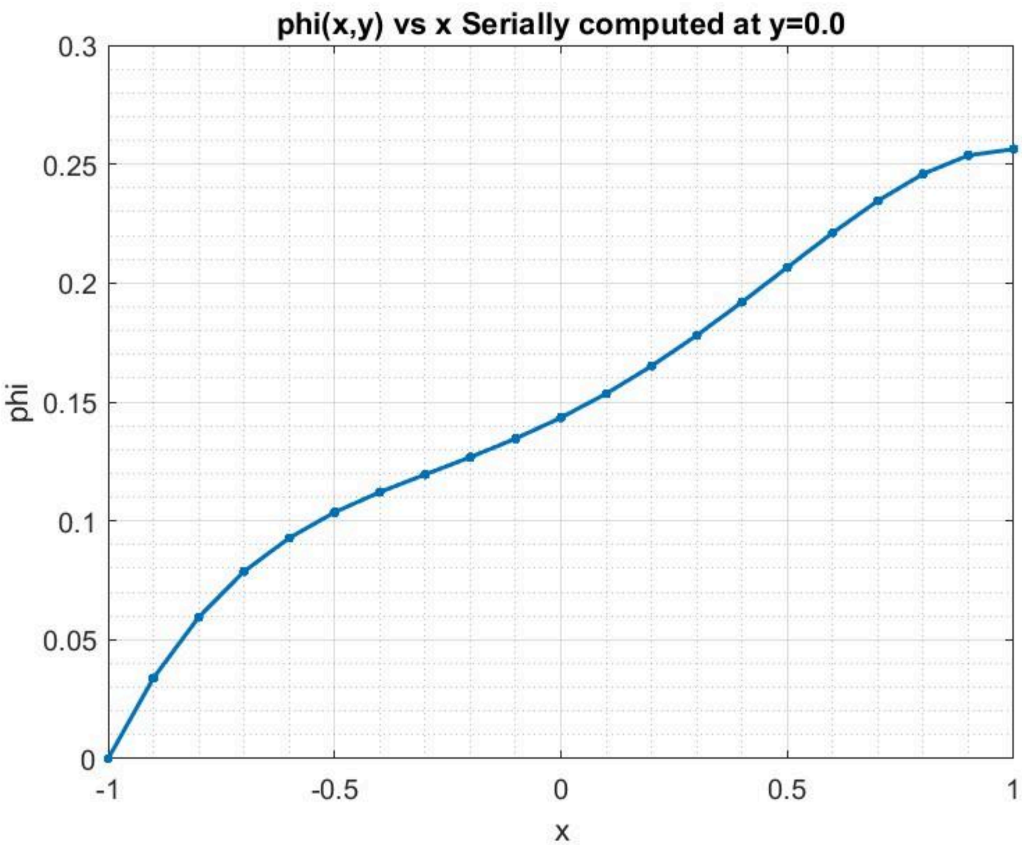

u(x,t) Parallelly computed at t=0.5s using 4 procs
Analytical
FOU
QUICK
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
u
0
0.5
1
1.5
2
x

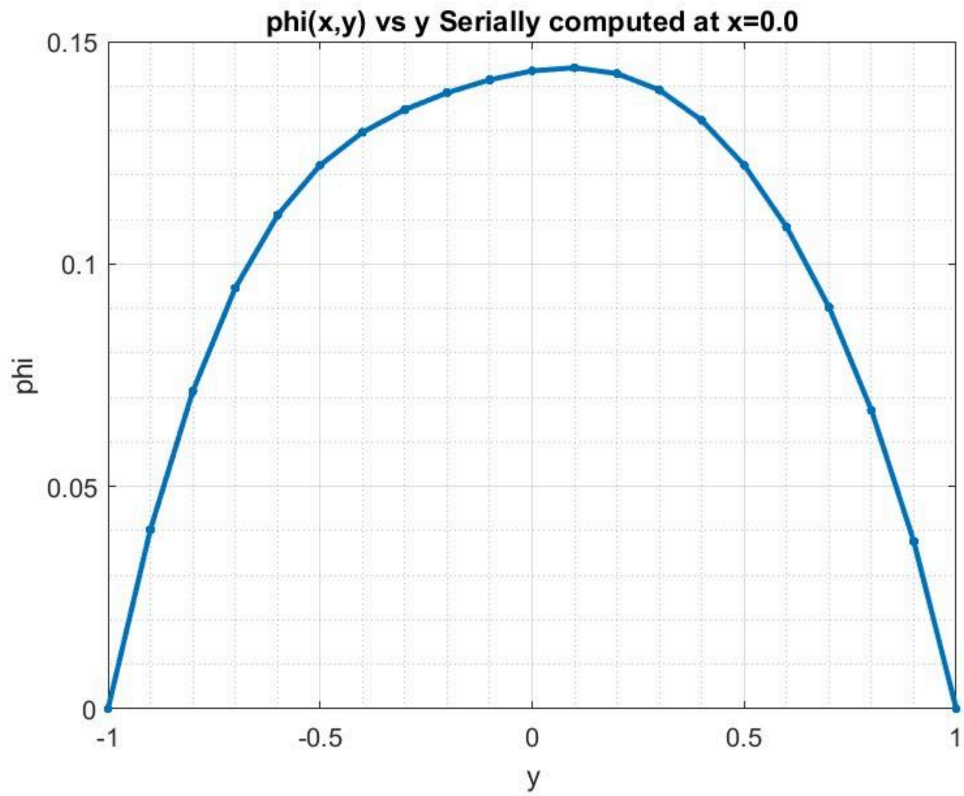

u(x,t) Parallelly computed at t=1s using 4 procs
Analytical
FOU
QUICK
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
u
0
0.5
1
1.5
2
x

c) The QUICK scheme is a closer approximation to the analytical solution as opposed to the first-order upwind scheme. The error for the FOU scheme seems to grow over time.

Q2

a)





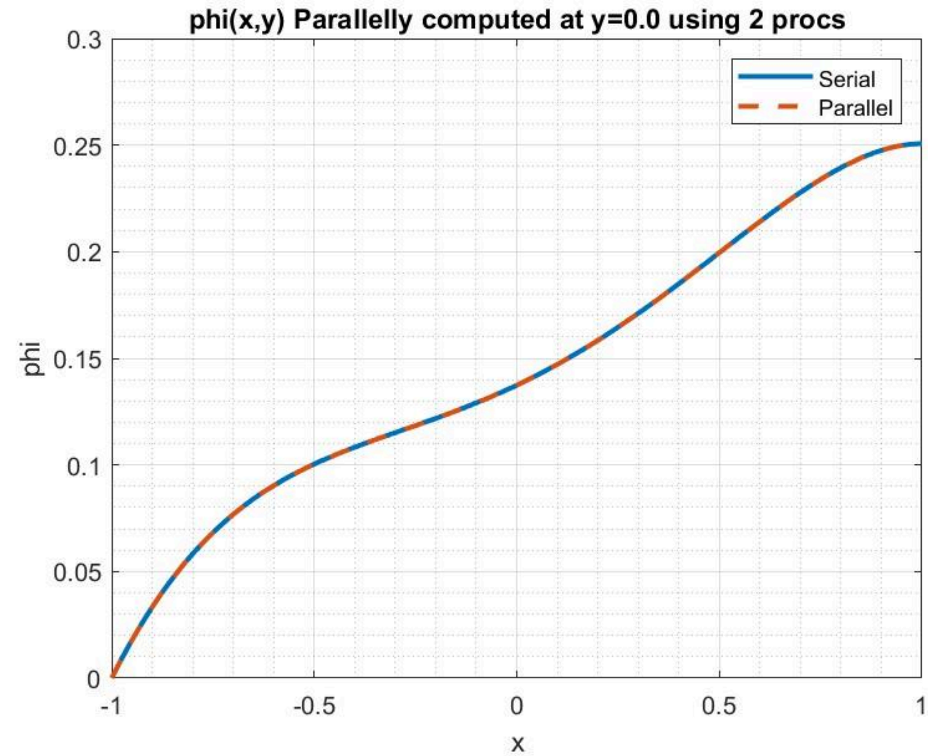
Solution for the Poisson equation using $\Delta = \Delta x = \Delta y = 0.1$ - ϕ vs x at y=0.0 and ϕ vs y at x=0.0

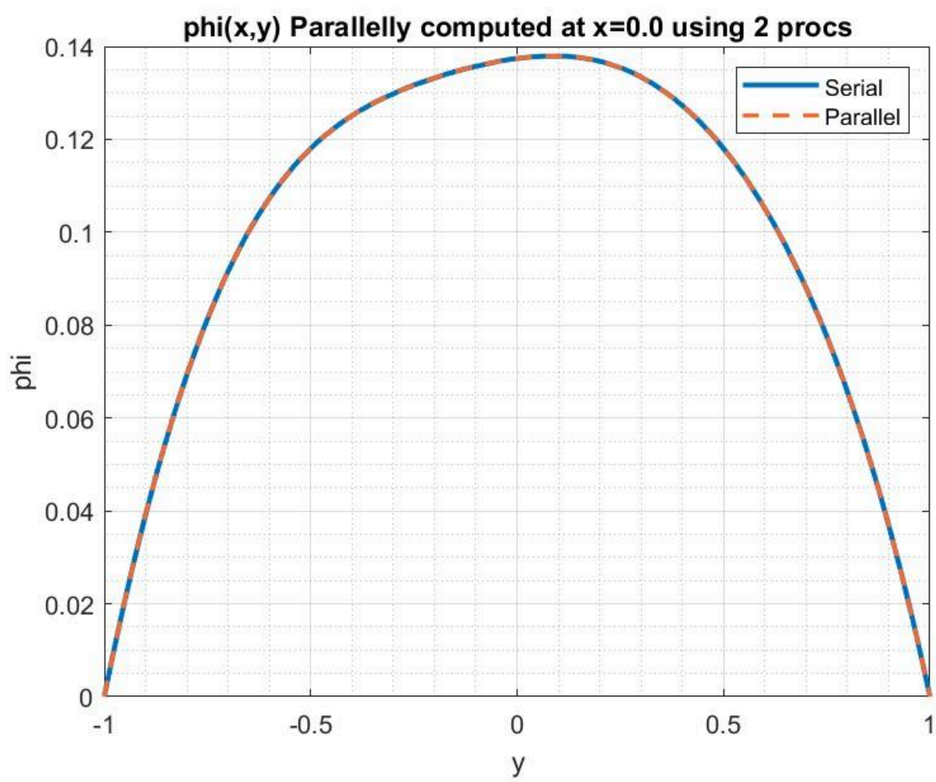
The method takes 688 iterations to converge within a norm error tolerance of 10^{-4}

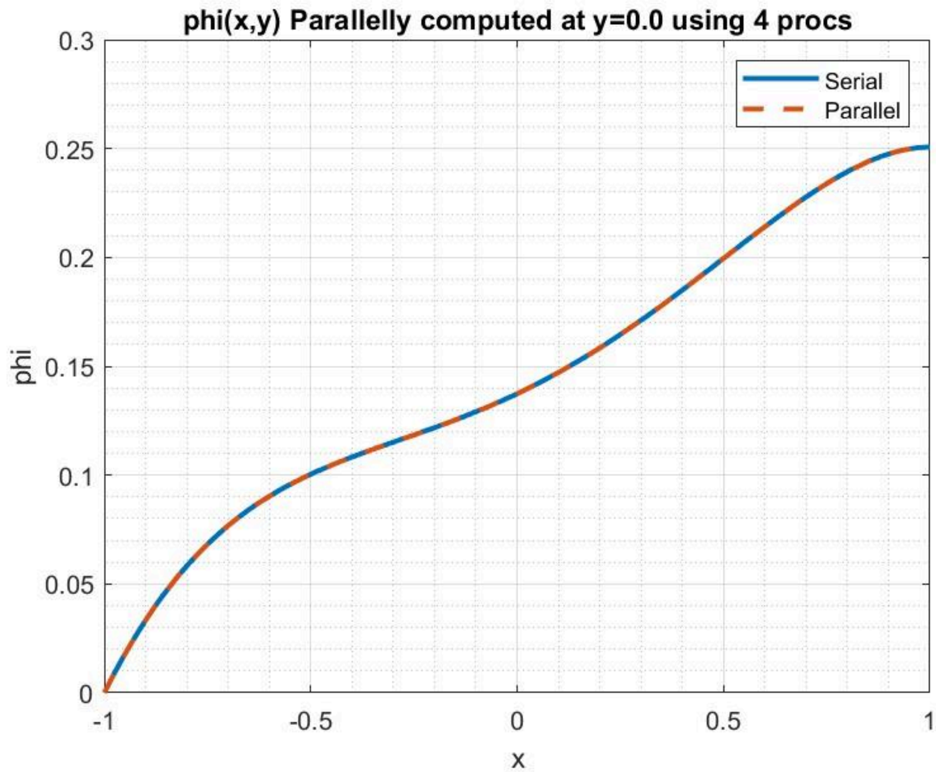
b)

Jacobi Iterative Scheme:

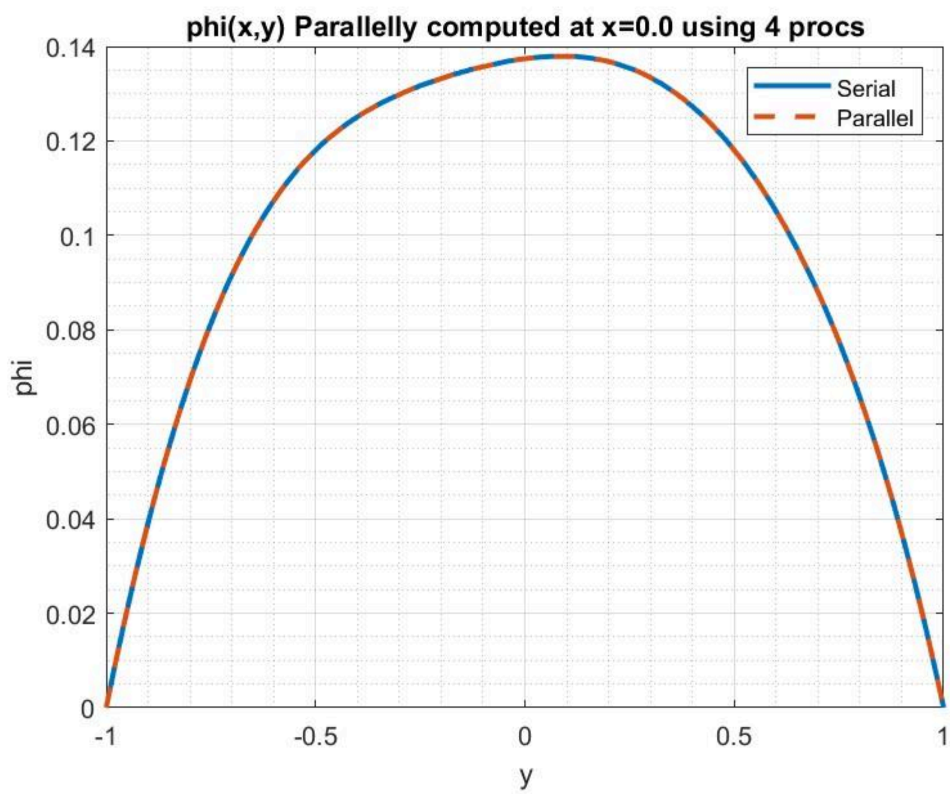
Solution for the Poisson equation using $\Delta = \Delta x = \Delta y = 0.01$


phi(x,y) Parallelly computed at y=0.0 using 2 procs
Serial
Parallel
phi
x
0.3
0.25
0.2
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1

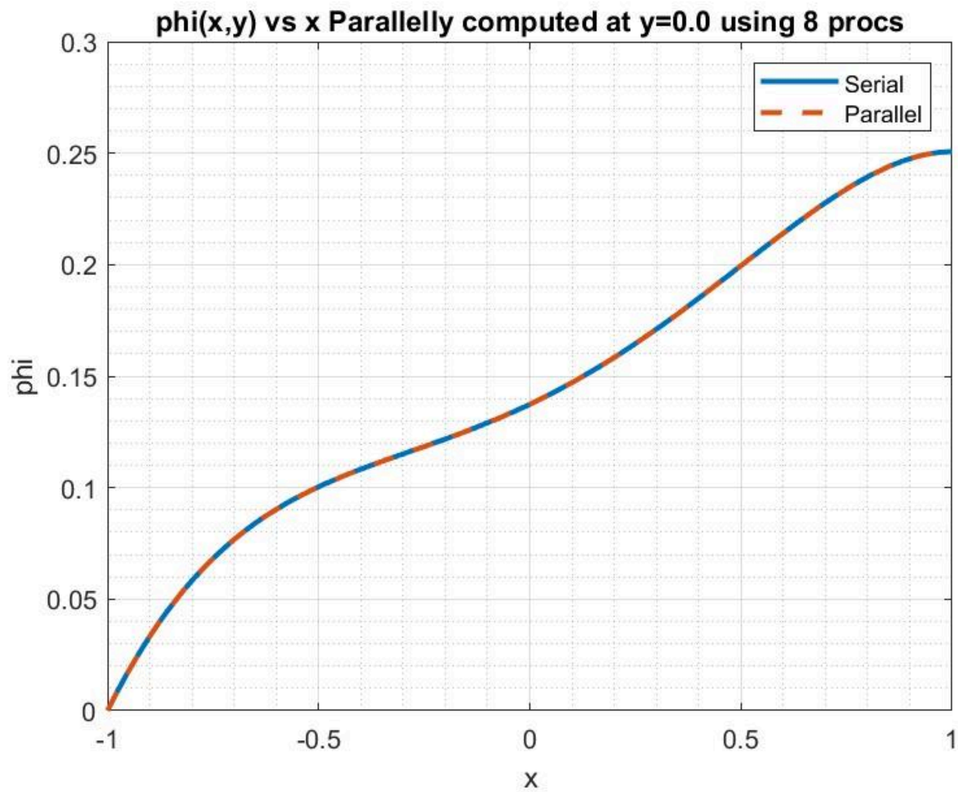

phi(x,y) Parallelly computed at x=0.0 using 2 procs
Serial
Parallel
phi
y
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-1
-0.5
0
0.5
1

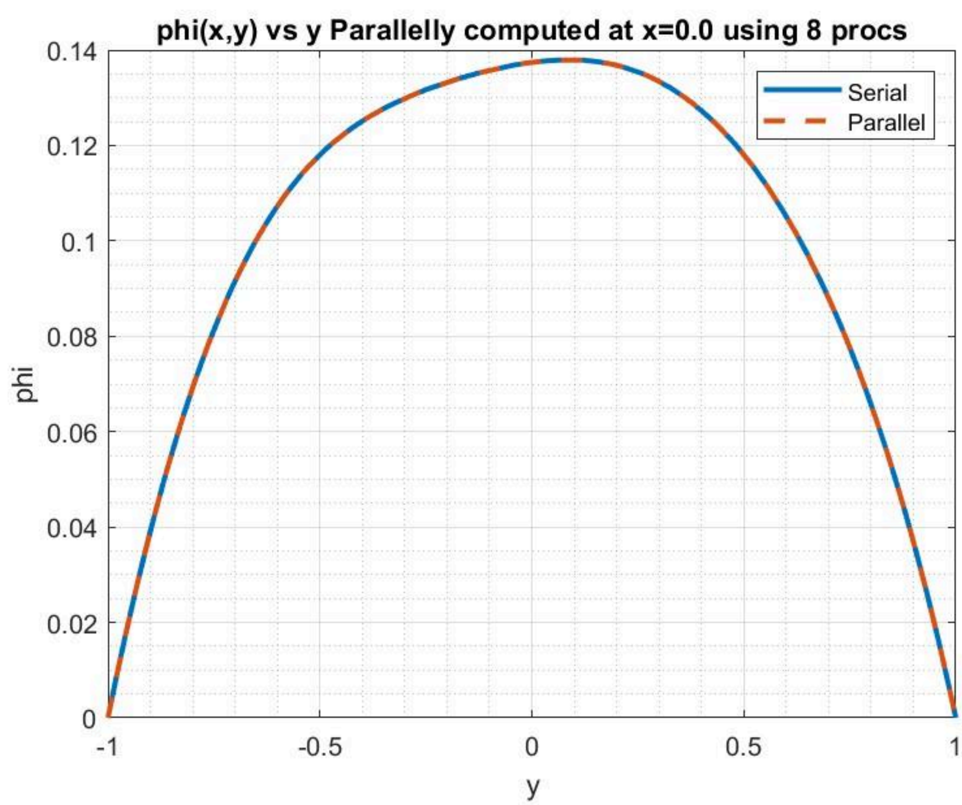


phi(x,y) Parallelly computed at y=0.0 using 4 procs
Serial
Parallel
phi
x
0.3
0.25
0.2
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1

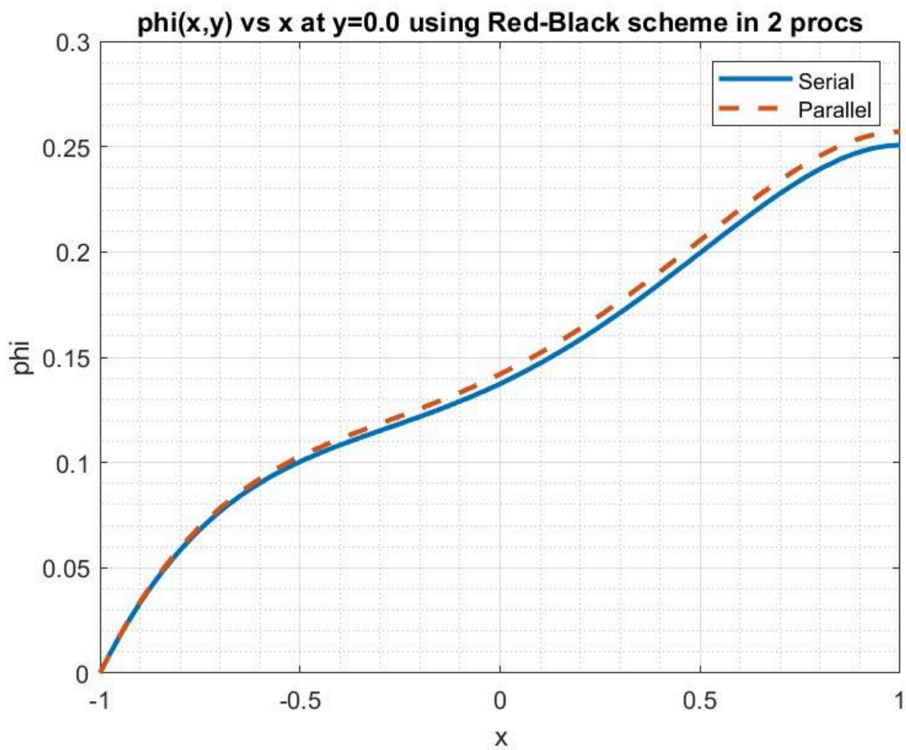


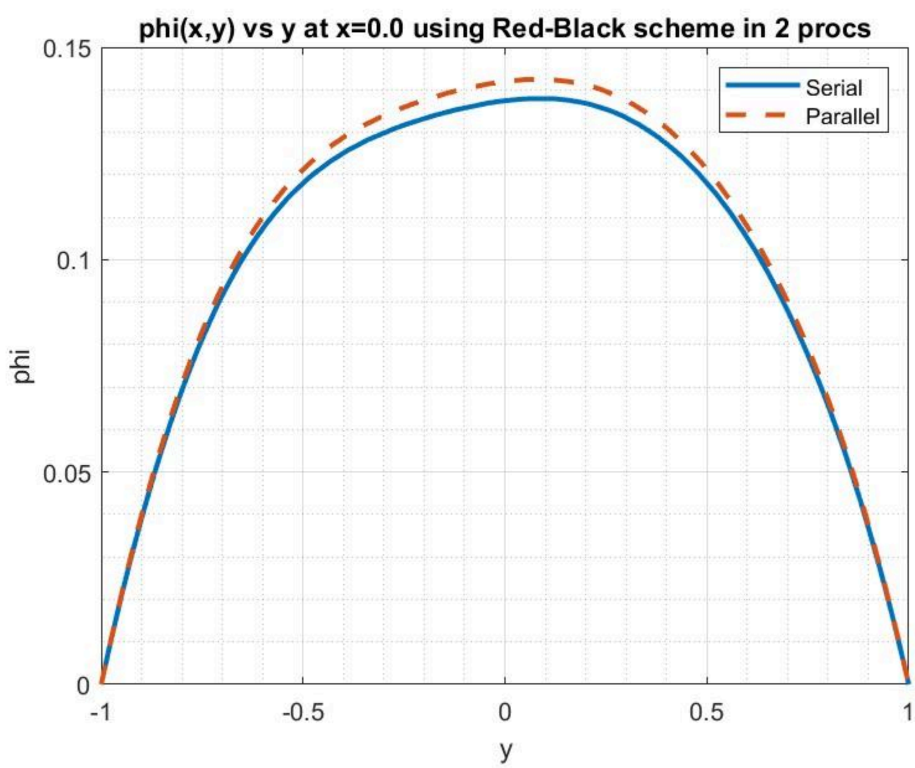
phi(x,y) Parallelly computed at x=0.0 using 4 procs
Serial
Parallel
phi
y
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-1
-0.5
0
0.5
1

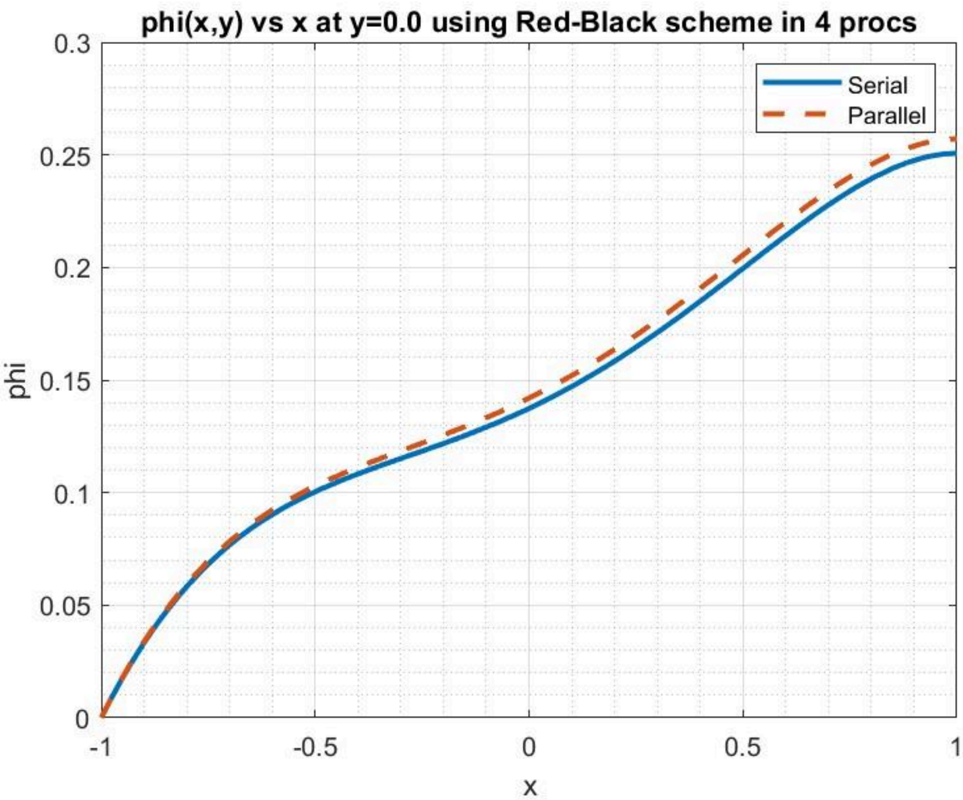

phi(x,y) vs x Parallelly computed at y=0.0 using 8 procs
Serial
Parallel
phi
x
0.3
0.25
0.2
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1

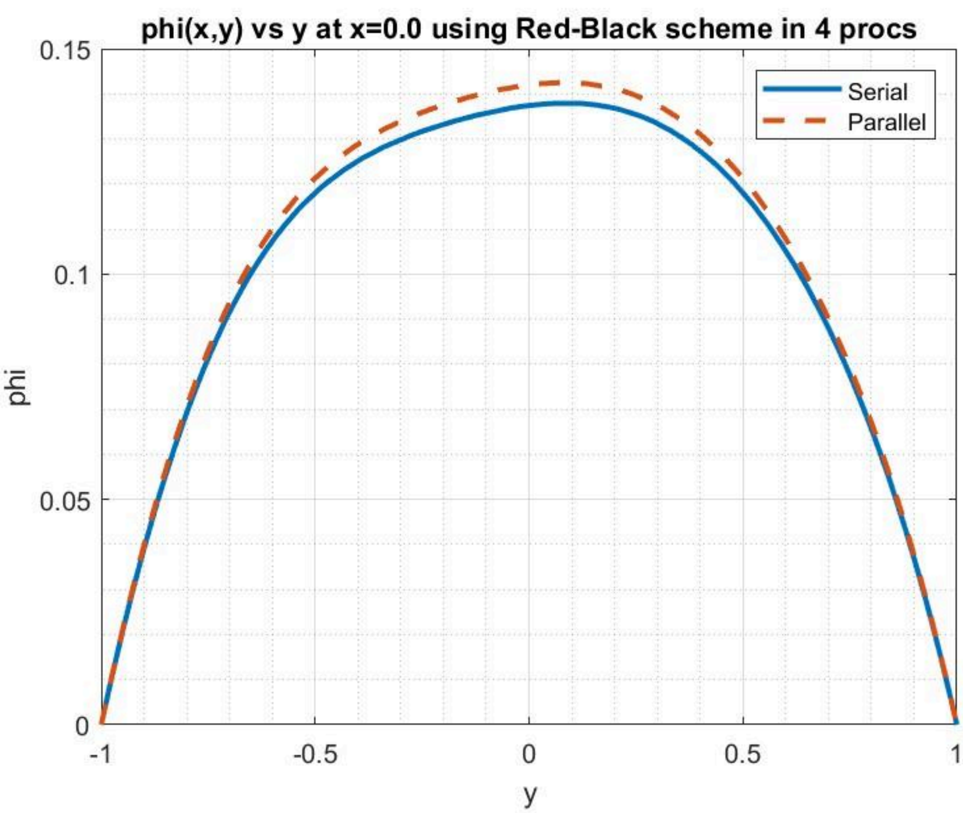

phi(x,y) vs y Parallelly computed at x=0.0 using 8 procs
Serial
Parallel
phi
y
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-1
-0.5
0
0.5
1

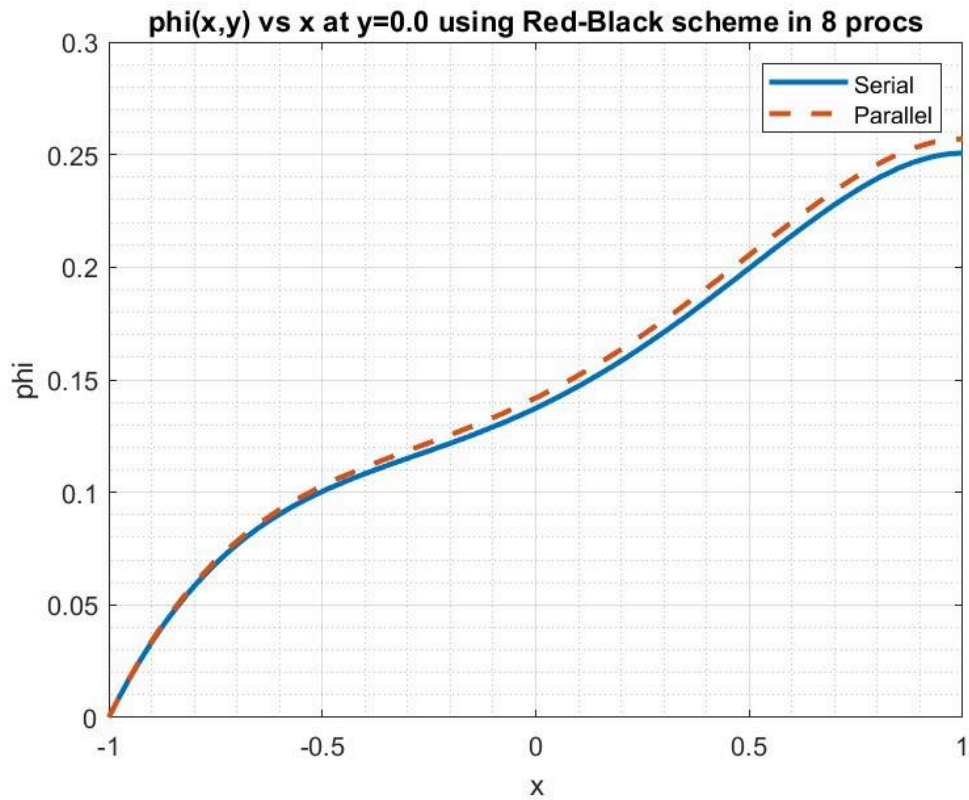
c) **Gauss-Seidel Red-black Iterative Scheme:**
Solution for the Poisson equation using Δ = Δx = Δy = 0.01

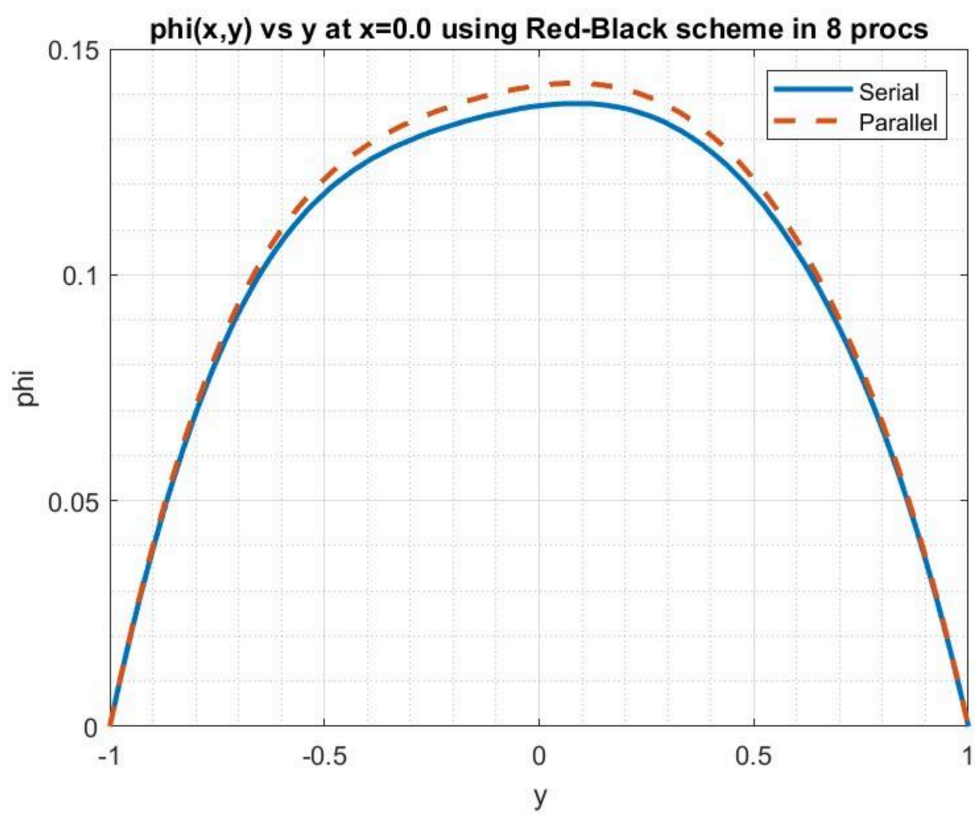





phi(x,y) vs x at y=0.0 using Red-Black scheme in 4 procs
Serial
Parallel
phi
x
0.3
0.25
0.2
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1


phi(x,y) vs y at x=0.0 using Red-Black scheme in 4 procs
Serial
Parallel
phi
y
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1


phi(x,y) vs x at y=0.0 using Red-Black scheme in 8 procs
Serial
Parallel
phi
x
0.3
0.25
0.2
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1


phi(x,y) vs y at x=0.0 using Red-Black scheme in 8 procs
Serial
Parallel
phi
y
0.15
0.1
0.05
0
-1
-0.5
0
0.5
1

For $\Delta = \Delta x = \Delta y = 0.01$, the number of iterations required is given below:

Number of iterations		
n_procs	**Jacobi**	**GS Red-Black**
1	38886	24031
2	38886	24072
4	38886	24045
8	38886	24330

Clearly from the above data, we can see that the Gauss-Seidel scheme has around **38% lesser iterations** required as opposed to the Jacobi scheme.

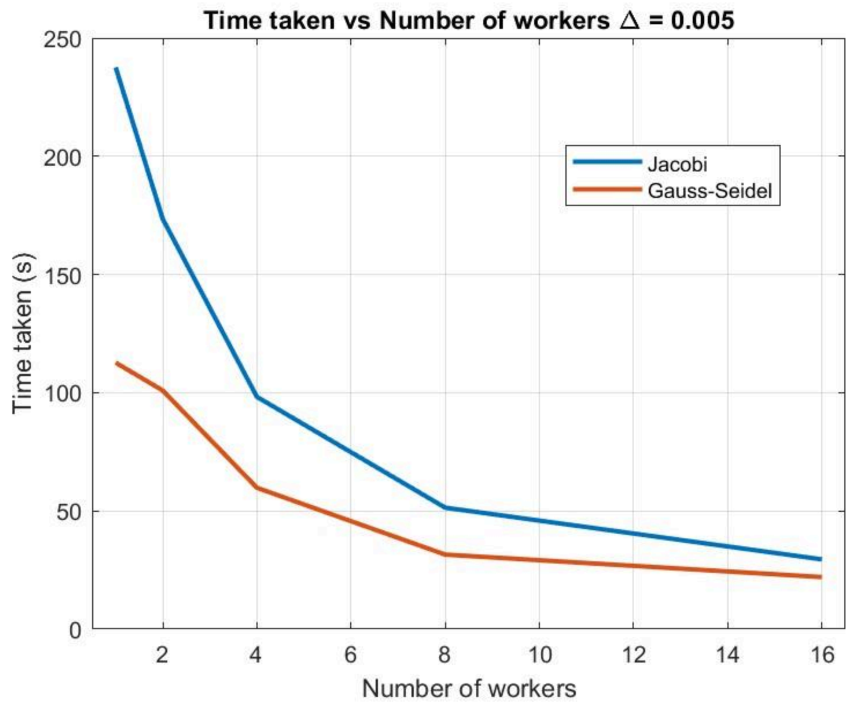
d)

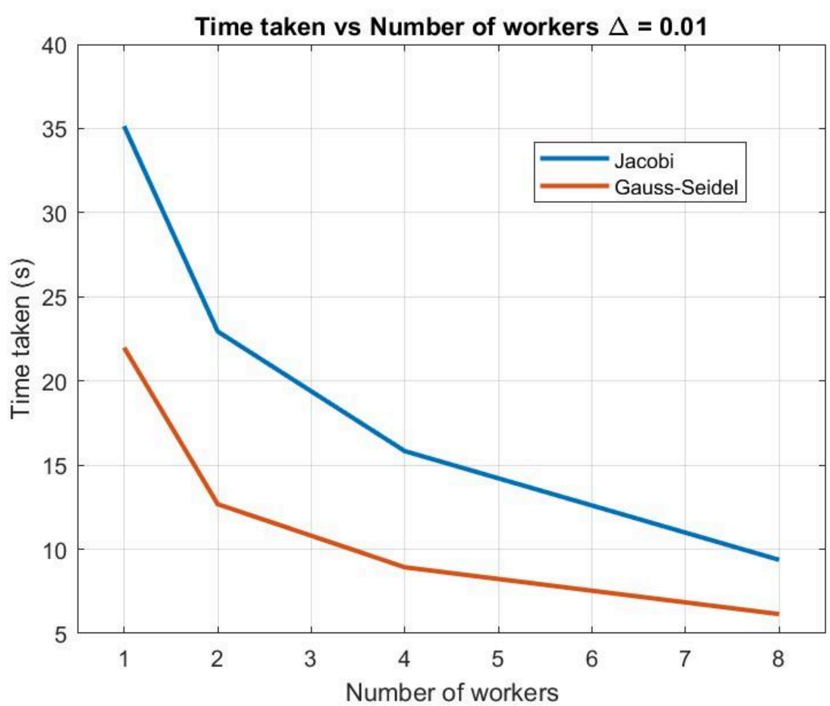
Time taken for each of the methods vs the number of processors is given below:

Time taken (for $\Delta = \Delta x = \Delta y = 0.005$)		
n_procs	**Jacobi**	**GS Red-Black**
1 (serial program)	237.630	112.710
2	173.476	100.931
4	98.244	59.788
8	51.367	31.526
16	29.508	22.022

Time taken (for $\Delta = \Delta x = \Delta y = 0.01$)		
n_procs	**Jacobi**	**GS Red-Black**
1 (serial program)	35.138	21.982
2	22.9399	12.6796

4	15.830	8.928
8	9.369	6.145





It is clear that the time taken by the **Gauss-Seidel Red Black method is better than the Jacobi Scheme** - both in terms of number of iterations and time taken for convergence. On increasing the number of processors, the two methods approach a similar time taken for computation.

Speed-up (ψ(n,p)) is given by (Serial Time / Parallel Time (p procs))

Speed-up (ψ(n,p)) *(for $\Delta = \Delta x = \Delta y = 0.005$)*		
n_procs	**Jacobi**	**GS Red-Black**
1	-	-
2	1.3698	1.1167
4	2.4188	1.8852
8	4.6261	3.5751
16	8.0531	5.1181

Speed-up (ψ(n,p)) *(for $\Delta = \Delta x = \Delta y = 0.01$)*		
n_procs	**Jacobi**	**GS Red-Black**
1	-	-
2	1.5317	1.7337
4	2.2197	2.4621
8	3.7505	3.5772

With respect to Speed-up, it is clear that the Jacobi method speeds up much better than the Gauss Seidel method. Both methods have almost a linear speed-up performing better when increasing the number of processors.

On increasing the problem size, there is no significant change in the speedup of Gauss Seidel scheme. On the other hand, the Jacobi method speeds up better when increasing the problem size.