ID5130 ASSIGNMENT-2 REPORT

Name: Keerthiharan A Roll no: CH22B026

April 20, 2024

Question 1

(a) Serial program for Upwind and Quick Scheme

First order accurate Upwind scheme for the solving Hyperbolic equations:

$$u_i^{n+1} = u_i^n - c\frac{\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) \tag{1}$$

Third order accurate Quick scheme for the interior points is given by:

$$u_i^{n+1} = u_i^n - c\frac{\Delta t}{\Delta x}(\frac{3}{8}u_i^n - \frac{7}{8}u_{i-1}^n + \frac{1}{8}u_{i-2}^n + \frac{3}{8}u_{i+1}^n) \tag{2}$$

For boundary points:

$$u_i^{n+1} = u_i^n - c\frac{\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) \tag{3}$$

In this problem c = 1.0, δx = 0.002, δt = 0.001, L = 2.0

Boundary conditions u(x=0,t) = 0 and u(x=L,t) = 0 Plots of analytical vs quick and upwind numerical solutions obtained at t=0, t=0.5 and t=1 as follows

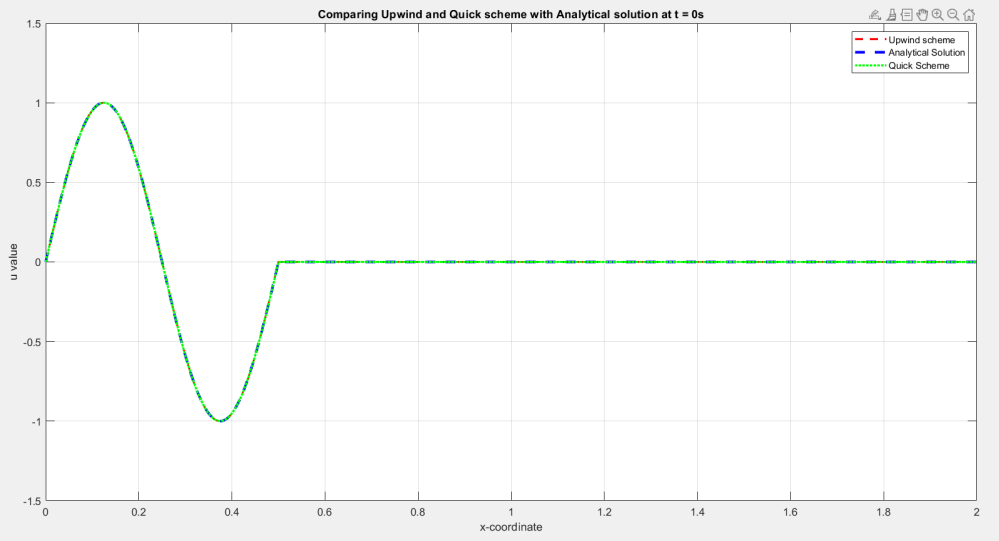


Figure 1: u(x,t=0) vs x

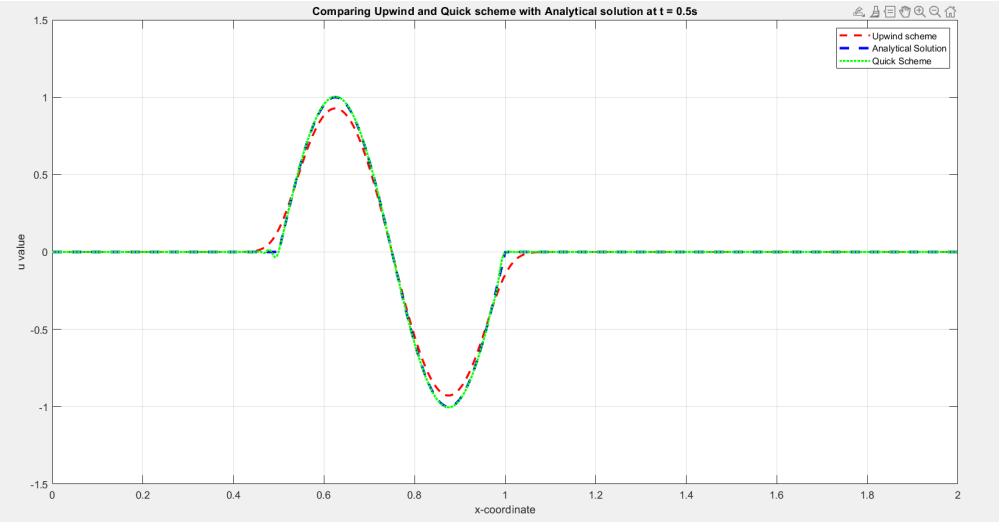


Figure 2: u(x,t=0.5) vs x

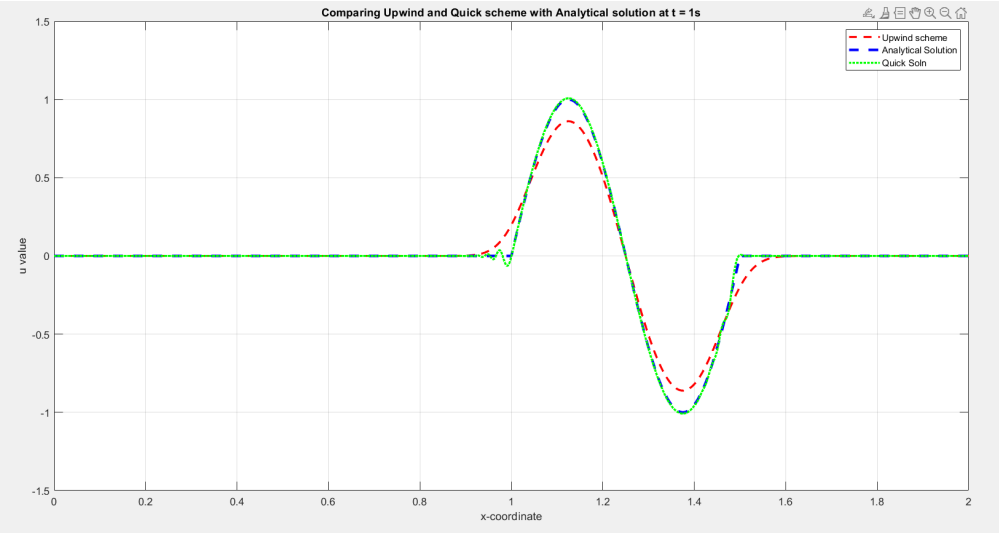


Figure 3: u(x,t=1) vs x

(b) MPI program for Numerical Scheme

A MPI Program was developed for the Numerical Schemes using Halo points for communication between the processors and the results are plotted as below,

Comparing MPI Parallel Upwind scheme solution with 2 and 4 threads with Analytical Solution at various time instants

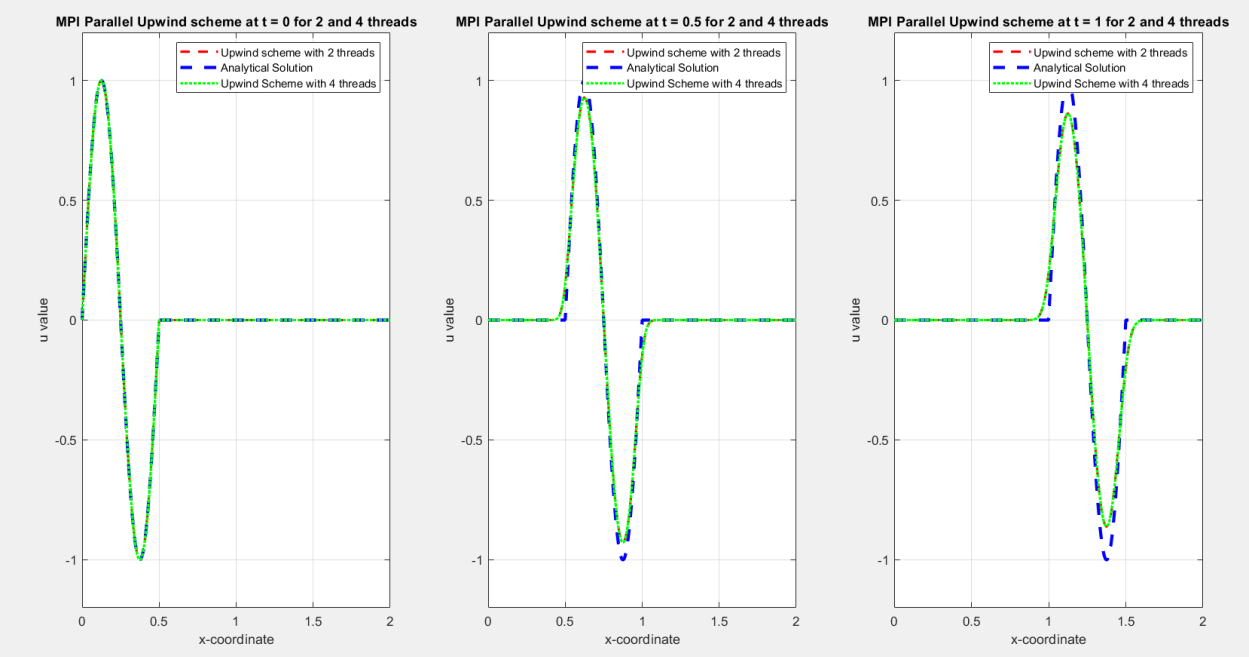


Figure 4: Parallelized Upwind Scheme Solution at t = 0, 0.5 and 1.0

Comparing MPI Parallel Quick scheme solution with 2 and 4 threads with Analytical Solution at various time instants

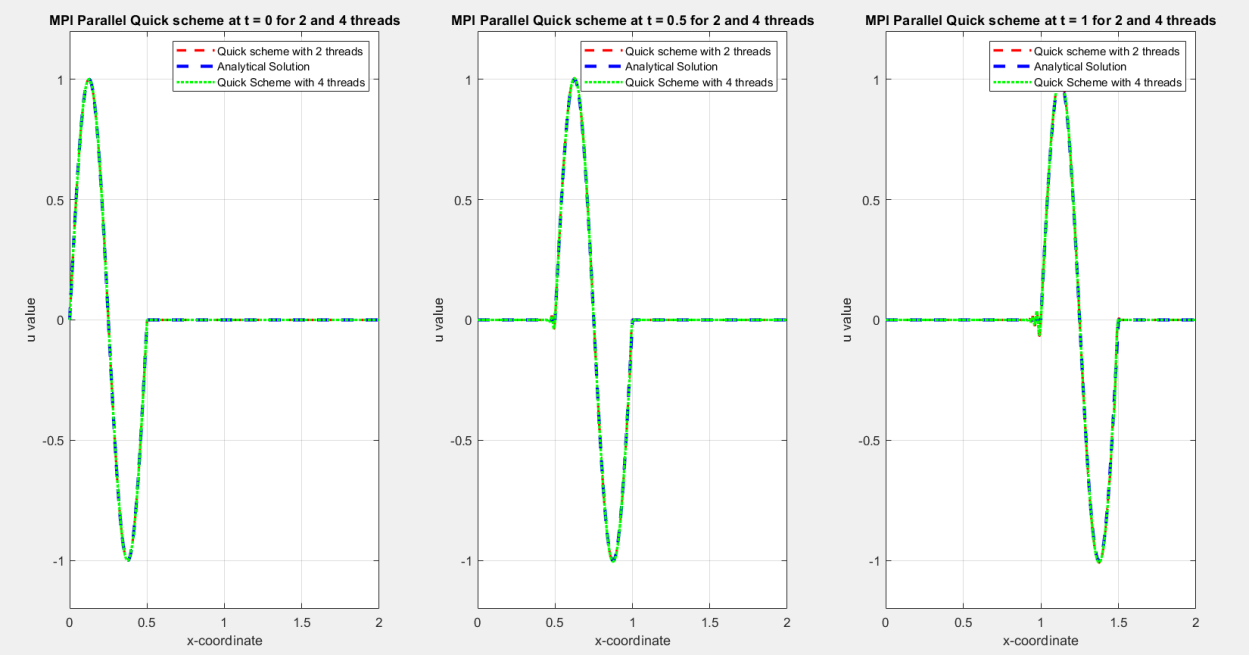


Figure 5: Parallelized Quick Scheme Solution at t = 0, 0.5 and 1.0

(c) Observed Difference between Upwind and Quick Scheme

From the graph, the Quick Scheme is accurate than the upwind scheme, this is evident due to the fact that Upwind scheme is 1st order in both time and space whereas Quick scheme is 3rd order in space (in interior points and 1st order at the boundaries) and 1st order in time, as both the schemes are 1st order in space the error increases along the time direction uniformly but at a given time instant Quick scheme is more accurate that the Upwind due to its superior space approximation.

Question 2

Consider the Poisson equation given by:

$$\nabla^2\phi = -q; \quad q = x^2 + y^2; \quad \phi(\pm 1, y) = 0; \quad \phi(x, -1) = 0; \tag{4}$$

We have $\Delta = \Delta x = \Delta y$.

Initial guess: $\phi(x, y) = 0$ everywhere. Here I have used infinity norm of residual between successive approximations to be less than 1e-4 as criteria for convergence. Which is given by,

$$\|\mathbf{x}\|_\infty = \max_i |x_i| \tag{5}$$

(a) Serial Program for Jacobi Iterative method

Using $\Delta = 0.1$, we get 21 points along x and y direction, using the Jacobi iterative method we have;

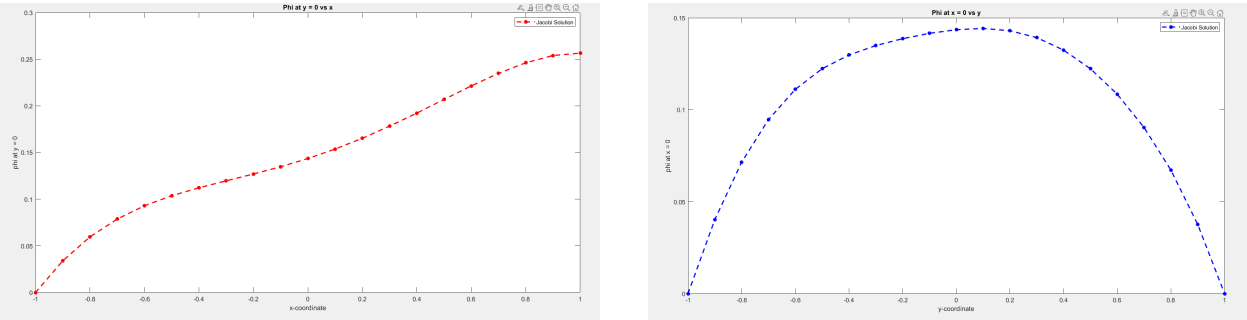
$$\phi_{i,j}^{(k+1)} = \frac{1}{4}(\phi_{i+1,j}^{(k)} + \phi_{i-1,j}^{(k)} + \phi_{i,j+1}^{(k)} + \phi_{i,j-1}^{(k)}) + \frac{\Delta^2}{4} q_{i,j} \tag{6}$$

where i = 1,2...20 and j = 1,2,..20, for i = 21 (and j = 1,2,..20) we have,

$$\phi_{i,j}^{(k+1)} = (4\phi_{i-1,j}^{(k)} - \phi_{i-2,j}^{(k)})/3 \tag{7}$$

It took **716** iterations to meet the specified convergence criteria (Final error norm was **9.93947e-05**)

The following plot shows the obtained solution of ϕ vs x for y = 0 and ϕ vs y for x = 0



(a) Serial Jacobi Solution for ϕ vs x at $y = 0$

(b) Serial Jacobi Solution for ϕ vs y at $x = 0$

Figure 6: Serial Jacobi Solutions

(b) MPI Program for Jacobi Iterative method

Using $\Delta = 0.01$, we get 201 points along x and y direction.

I have used 1-D Row wise domain decomposition and communicated between the processors by using hallo points.

This method took **42,105** iterations to converge.

The following plot shows the obtained solution of ϕ vs x for y = 0 and ϕ vs y for x = 0 for various number of threads

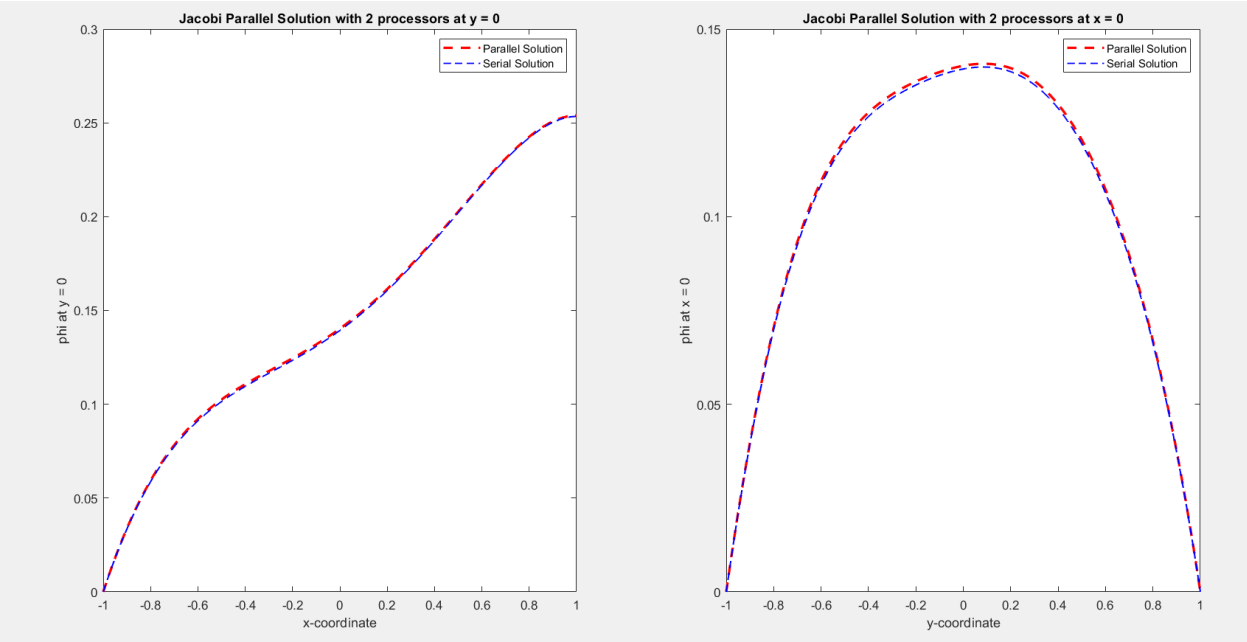


Figure 7: Parallelized Jacobi Iterative Solution for p = 2

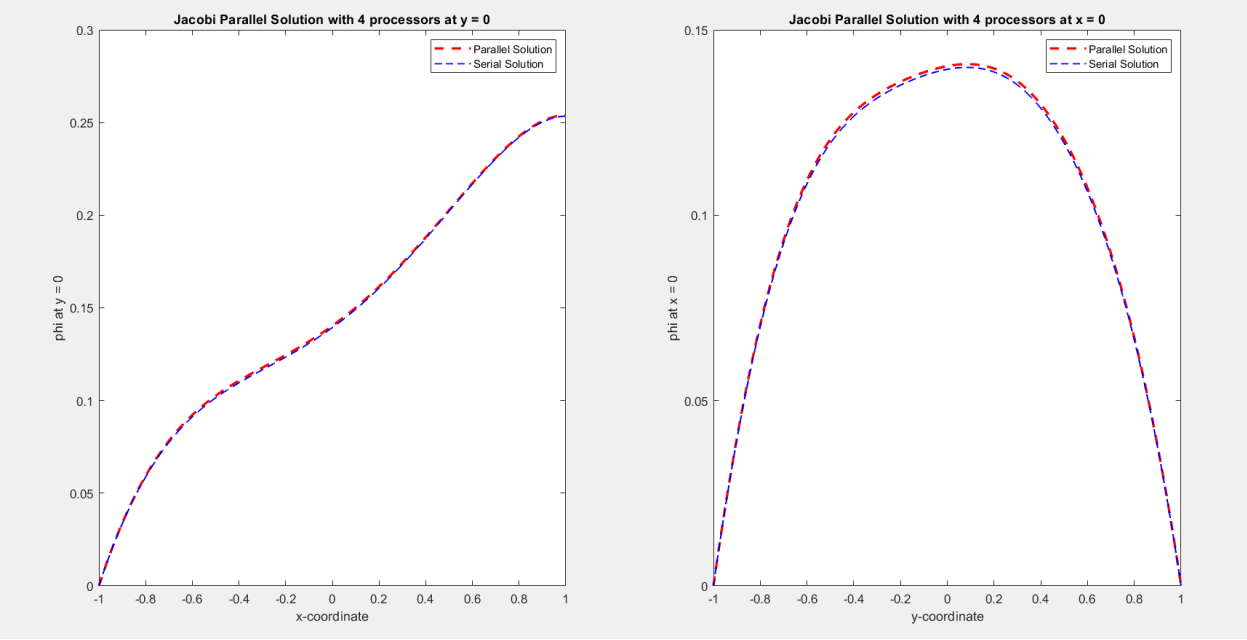


Figure 8: Parallelized Jacobi Iterative Solution for p = 4

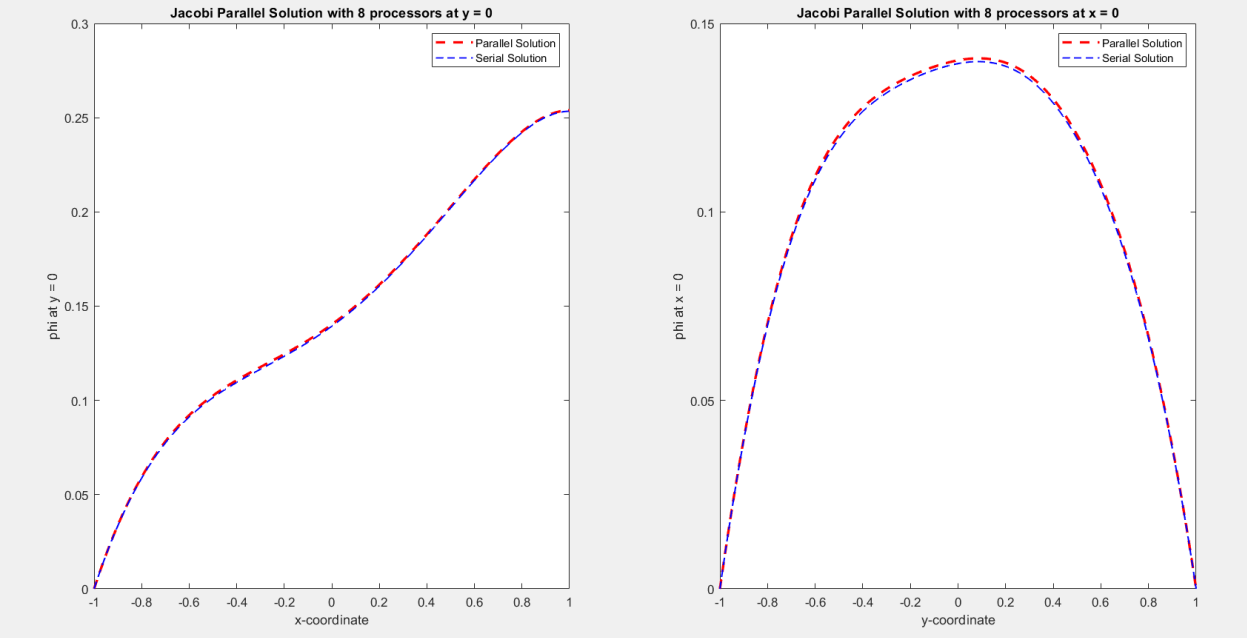


Figure 9: Parallelized Jacobi Iterative Solution for p = 8

The figures shows the solution obtained were consistent even when we use different number of processors as we take infinity norm by communicating the error between the processors.

(c) MPI Program for Gauss Seidel Red-Black Coloring Approach

The same procedure as (b) was carried out and it took **25,642** iterations to converge. The results were as plotted,

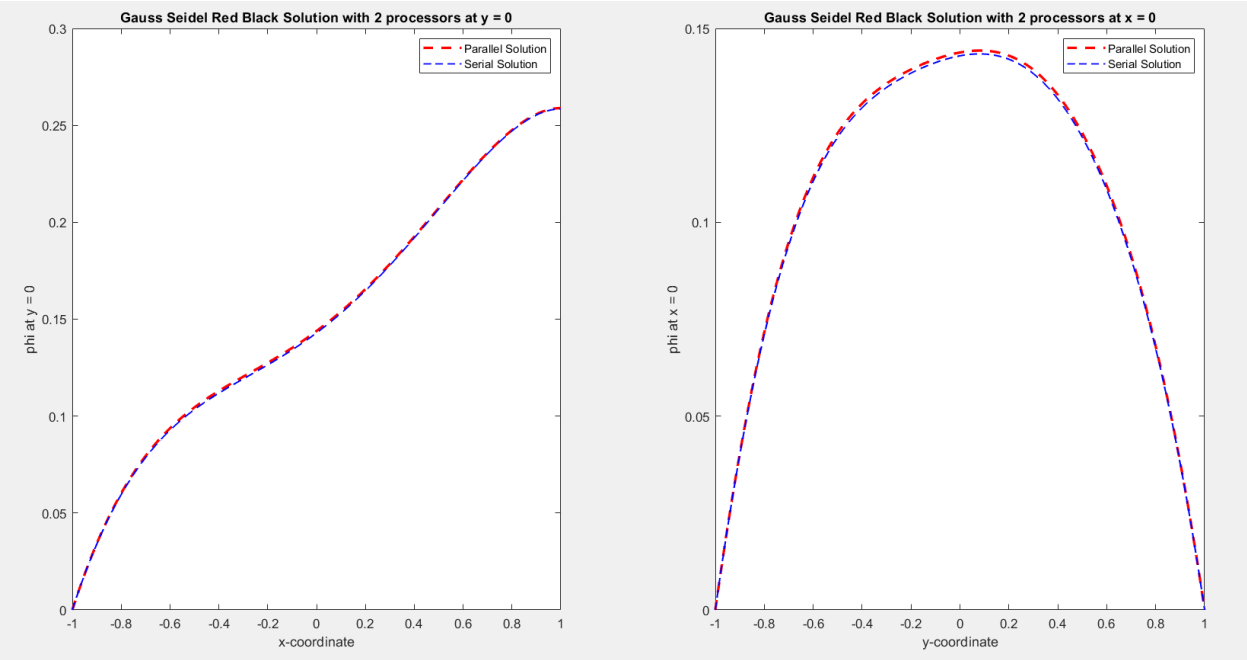


Figure 10: Parallelized Gauss Seidel Red-Black Solution for p = 2

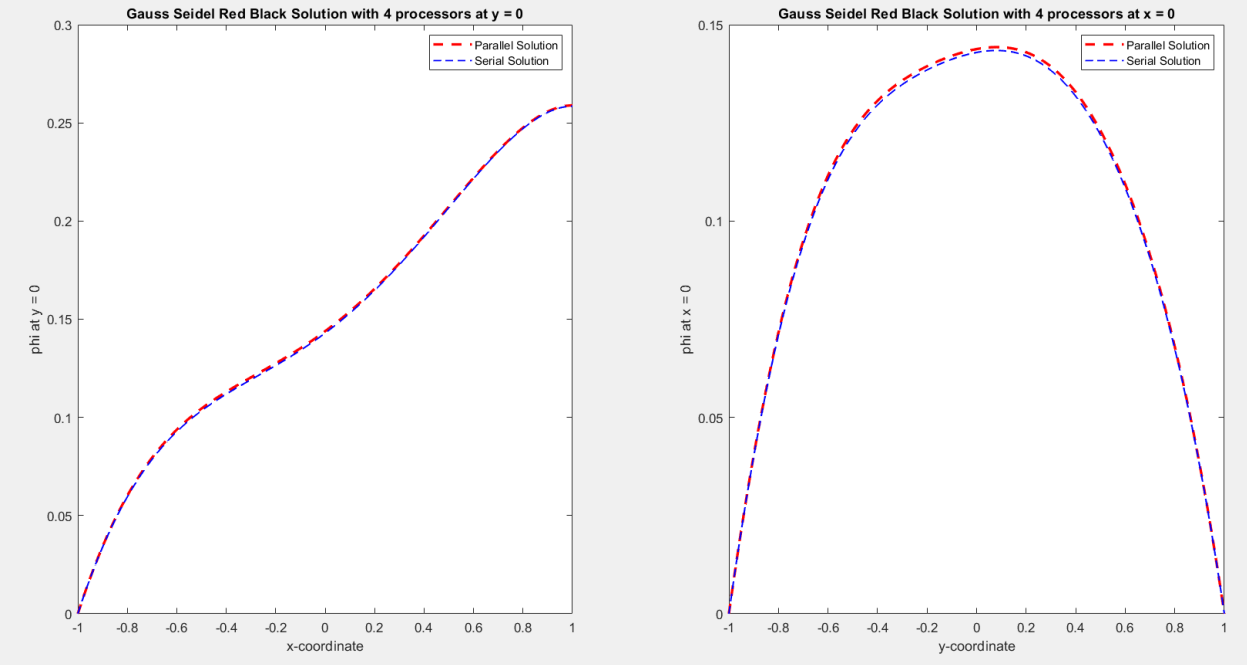


Figure 11: Parallelized Gauss Seidel Red-Black Solution for p = 4

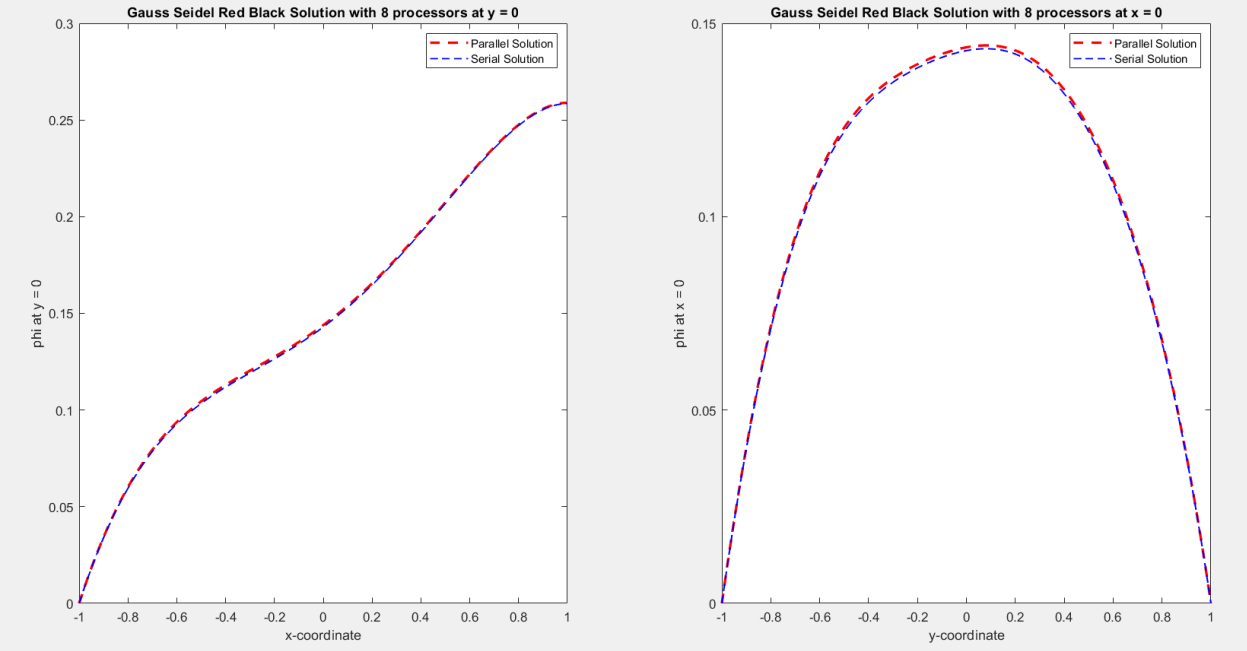


Figure 12: Parallelized Gauss Seidel Red-Black Solution for p = 8

Again the figures shows the solution obtained were consistent even when we use different number of processors as we take infinity norm by communicating the error between the processors.

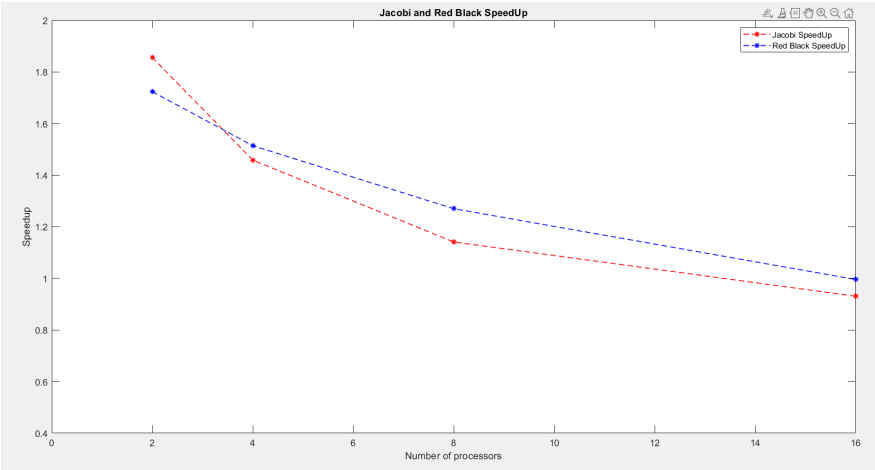
On comparing the number of iterations it took to converge, as expected Gauss Seidel took much less number of iterations **(roughly half)**, when compared to the Jacobi method. This is evident due to the fact that Gauss Seidel is kind of an impilcit iterative scheme and uses already updated value while Jacobi doesn't.

(d) Speedup in the parallel solutions

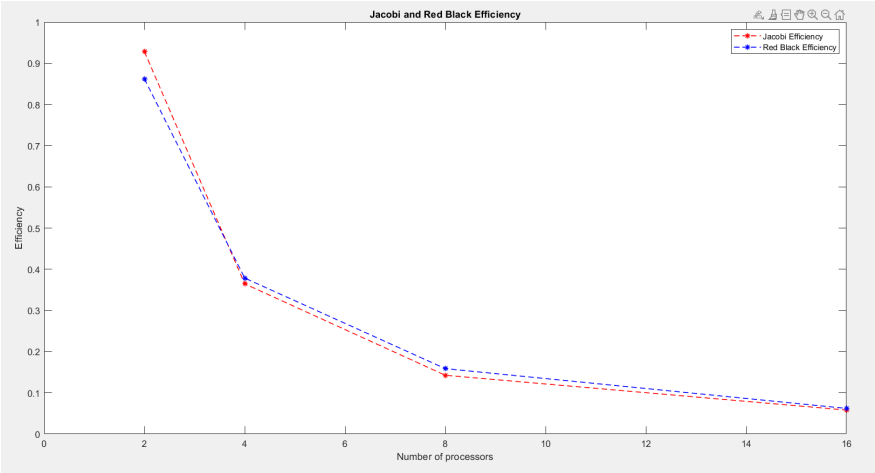
Speedup is a measure of how much faster a parallel algorithm performs compared to a sequential algorithm. It is calculated as,

$$\text{Speedup} = \frac{T_{\text{sequential}}}{T_{\text{parallel}}} \qquad \text{Efficiency} = \frac{Speedup}{No : of processors} \tag{8}$$

Here, we have considered speedup for $\Delta = 0.005$



(a) Speedup



(b) Efficiency

Figure 13: Speedup and Efficiency as a function of number of processors for a given problem size

As we can see the speedup is declining as number of processors increases contradicting the Amdahl's law and ideal speedup, this is highly likely due to the **resource constraint** in my Laptop as it supports only 2 cores, as I oversubscribe for more processors the performance drops due to the associated communication overhead and inefficiency, the same is observed with efficiency too.

The Jacobi method has slight advantage over Guass Siedel Red-Black for relatively smaller number of processors, but as the number of processors increase Red-Black shows better performance due to it's better scope for parallelization. Based on the number of iterations and their performance **Gauss Seidel Red-Black** is superior to Jacobi Iterative method.