ID5130 Parallel Scientific Computing

Assignment 2

Ruthwik Chivukula

ME21B166

Contents

1 Programming Question 1

Obtaining numerical solutions for the 1-D advective flow equation using first-order upwind scheme and third-order QUICK scheme. Comparing the results with the analytical solution and drawing insights. We will be implementing both parallel and serial versions of both numerical methods.

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{1}$$

$$u_o(x) = \begin{cases} \sin(4\pi x) & \text{if } 0 < x \leq 0.5 \\ 0 & \text{if } 0.5 < x < 2 \end{cases}$$

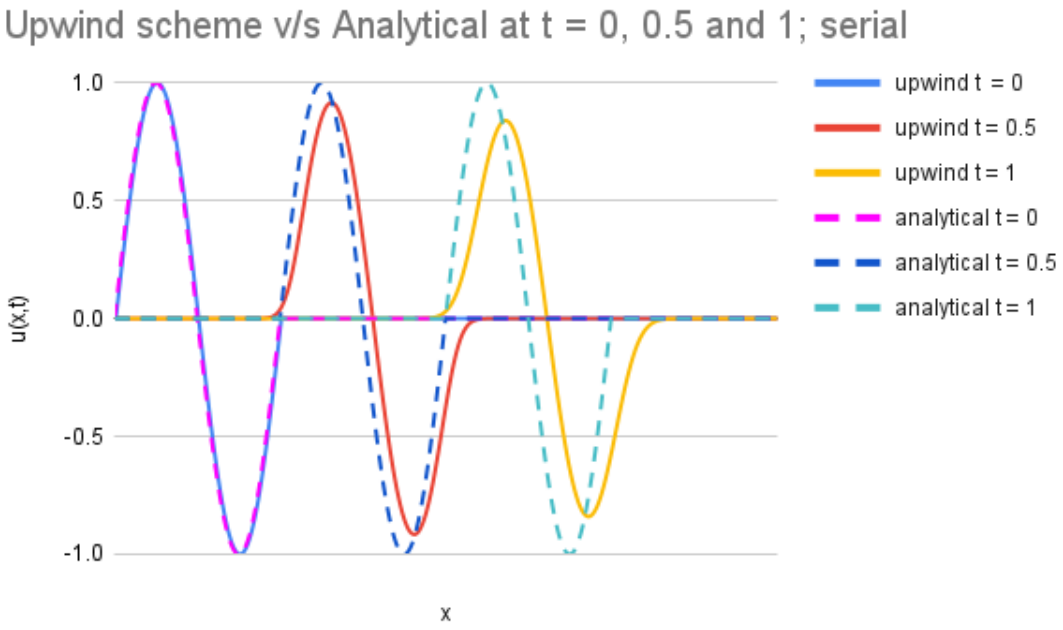
1.1 1 (a)

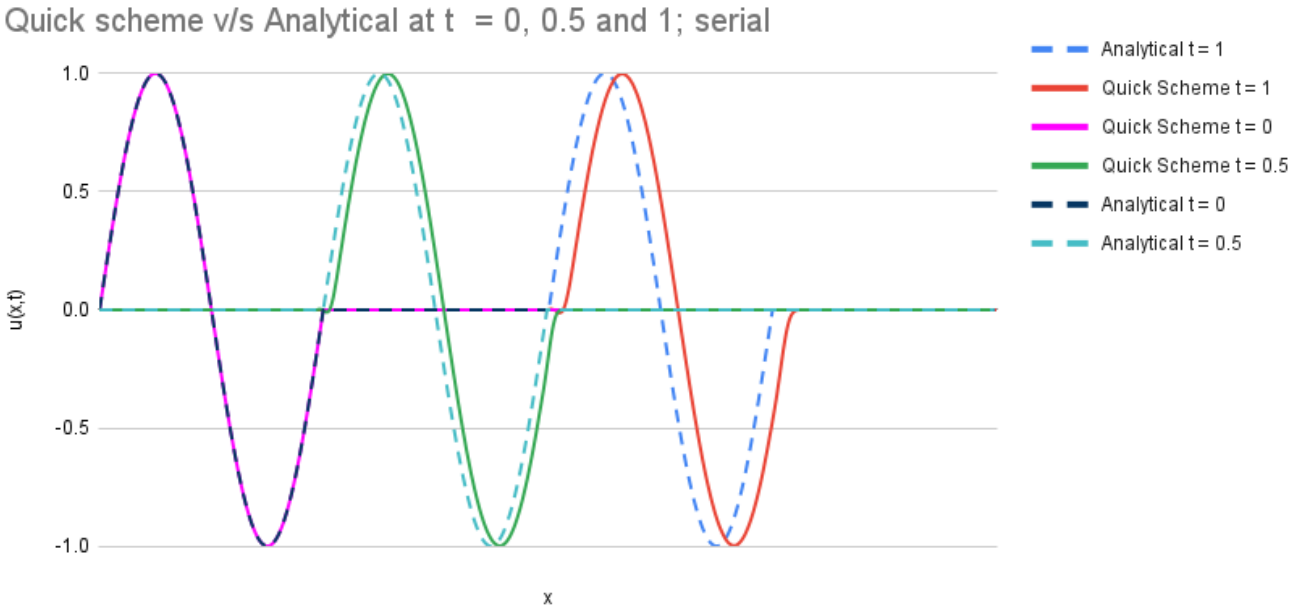
Plots for serial version of upwind scheme and QUICK scheme for time steps t = 0, t = 0.5 and t = 1. Analytical solutions are also plotted on the same graph for comparison purpose. The discretized equations of upwind and QUICK scheme are as follows:

$$\frac{\partial u}{\partial x} \approx \frac{u_i^n - u_{i-1}^n}{\Delta x} \tag{2}$$

$$\frac{\partial u}{\partial x} \approx \frac{3/8u_i^n - 7/8u_{i-1}^n + 1/8u_{i-2}^n + 3/8u_{i+1}^n}{\Delta x} \tag{3}$$

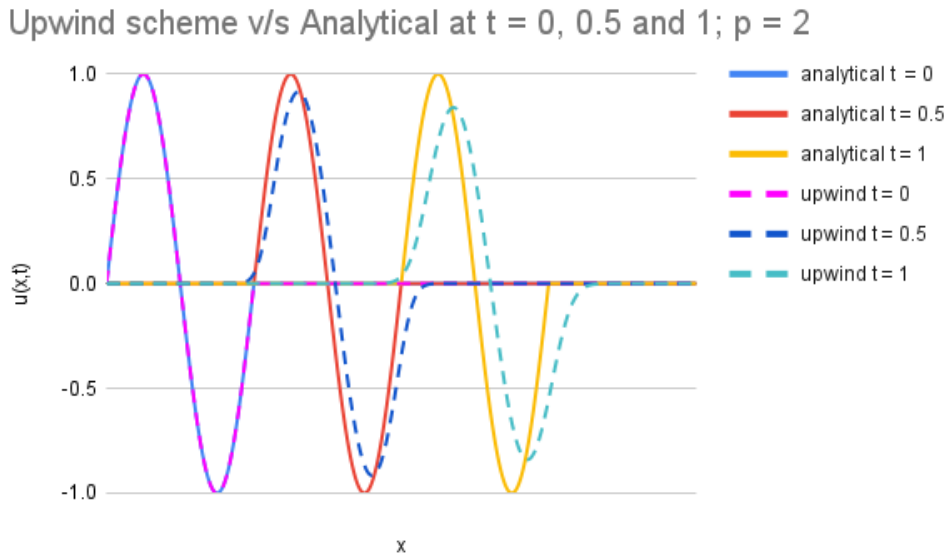
Ruthwik Chivukula
ME21B166

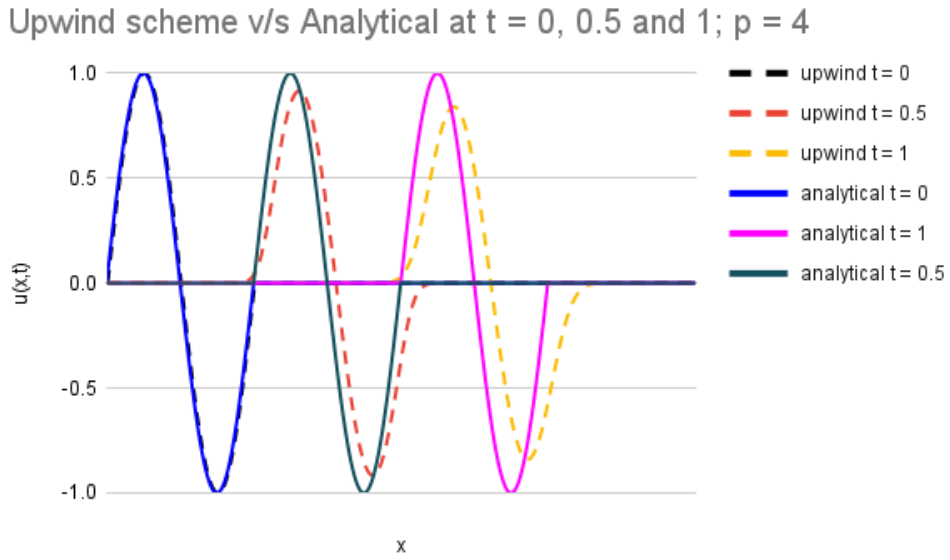




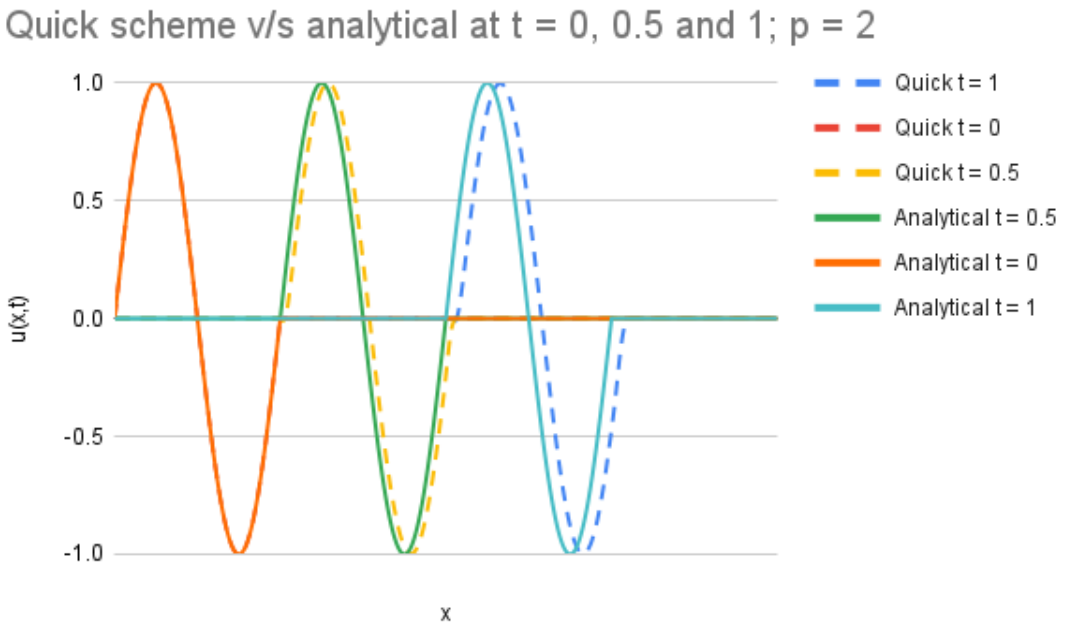
1.2 1(b)

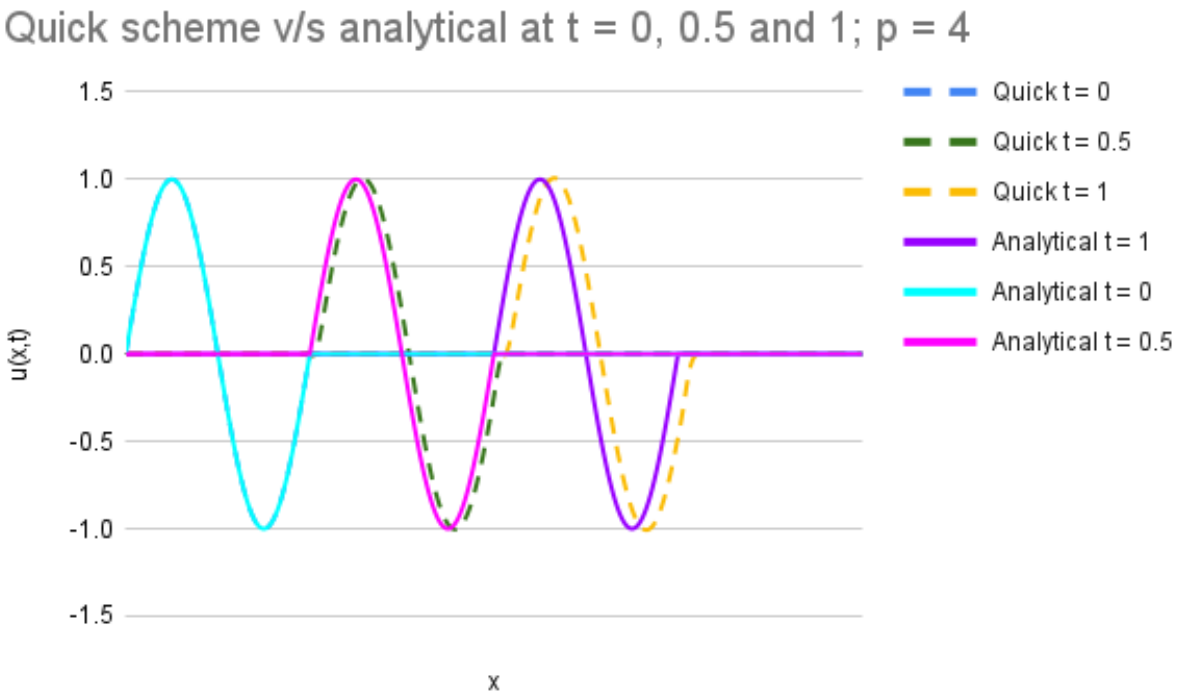
Plots for the parallel version of the above-mentioned numerical methods at time steps t = 0, 0.5, and 1. Used p = 2 and p = 4 as the number of processors to parallelize.





Ruthwik Chivukula
ME21B166





1.3 1(c)

It can be observed that for the upwind scheme, the amplitude gradually drops with time. But in the case of the QUICK scheme, the amplitude remains fixed. Both solutions are initially in phase with the analytical solution, but with time, they get ahead of it. So we can comment that the accuracy of the numerical solution is decreasing as we march into future steps.

2 Programming Question 2

$$\nabla^2\phi = -q; q = x^2 + y^2 \tag{4}$$

$$\phi^{(k+1)}(i,j) = \frac{\phi^{(k)}_{i+1,j} + \phi^{(k,k+1)}_{i-1,j} + \phi^{(k)}_{i,j+1} + \phi^{(k,k+1)}_{i,j-1} + \Delta^2 q_{i,j}}{4} \tag{5}$$

$$\phi_i = \frac{4\phi_{i-1} - \phi_{i-2}}{3} \tag{6}$$

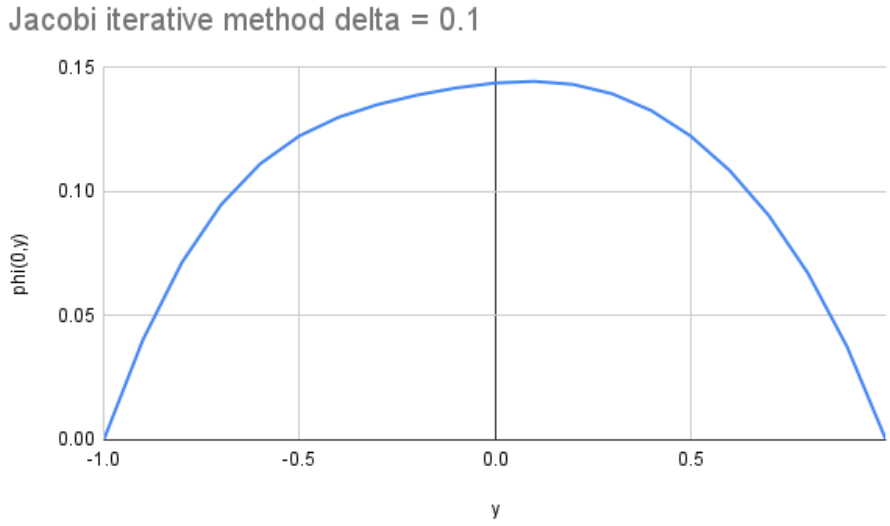
The problem involves imposing a Neumann boundary condition at x = 1. $\frac{\partial\phi}{\partial x} = 0$ It has been asked to obtain the numerical solution using the Jacobi iterative method as well as the Red-Black method using both serial and parallel implementations followed by speed-up analysis.

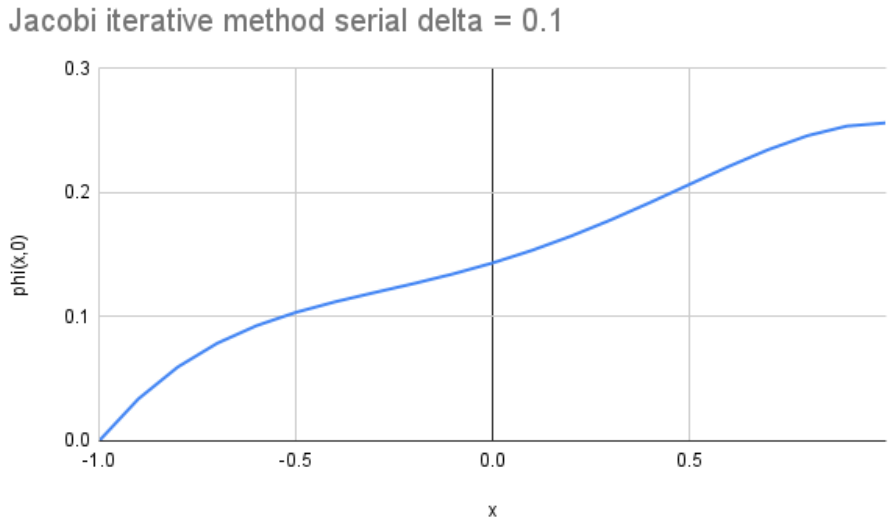
The initial values of $\phi_{i,j}$ are set to zero everywhere. $\Delta = 0.1$, $\Delta = 0.01$ and $\Delta = 0.005$ have been tested

2.1 2(a)

Serial code implementation for Jacobi iterative method with $\Delta = 0.1$. Used **L2 norm** as the metric for convergence with the tolerance being 0.0001

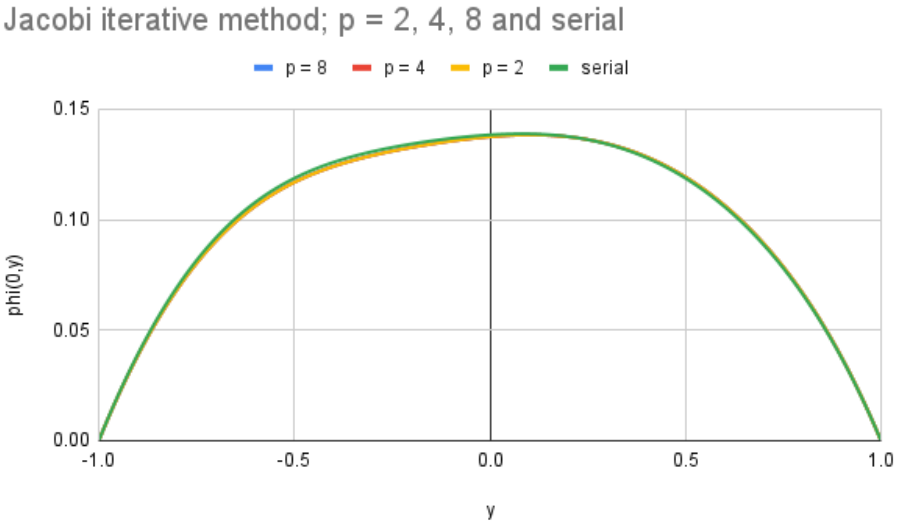
Number of iterations ($\Delta = 0.1$) taken for convergence: **688**.

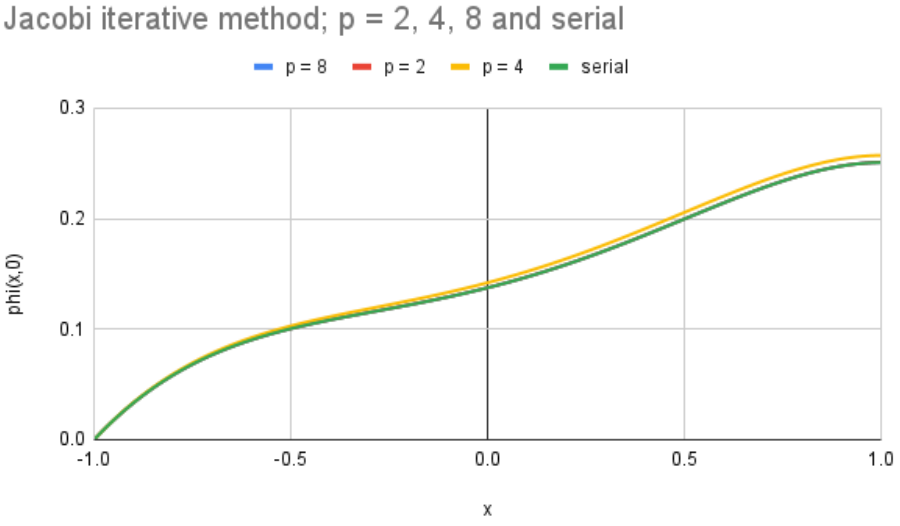




2.2 2(b)

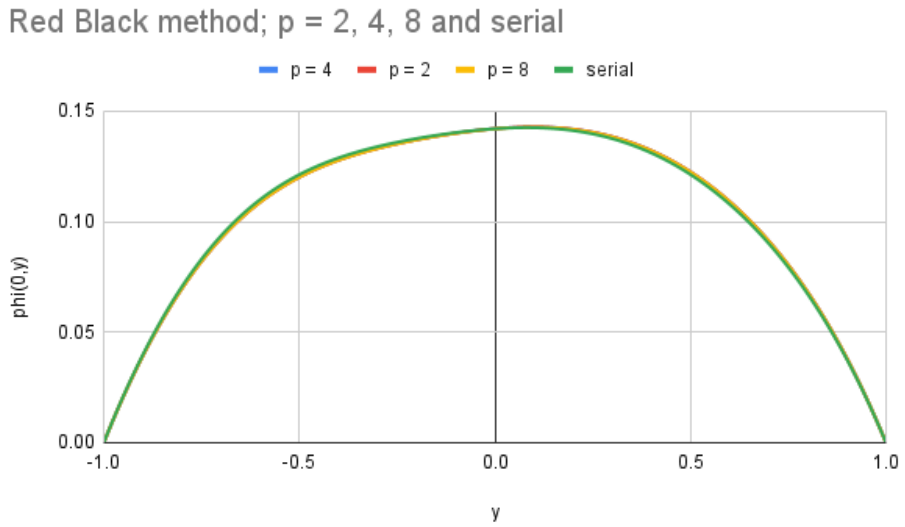
Jacobi implementation using MPI. I have done row-wise decomposition, and below are the plots for p = 2, 4, and 8 processors. The serial code has also been plotted to compare the results. $\Delta = 0.01$ is considered here, so the number of grid points increases to 201.

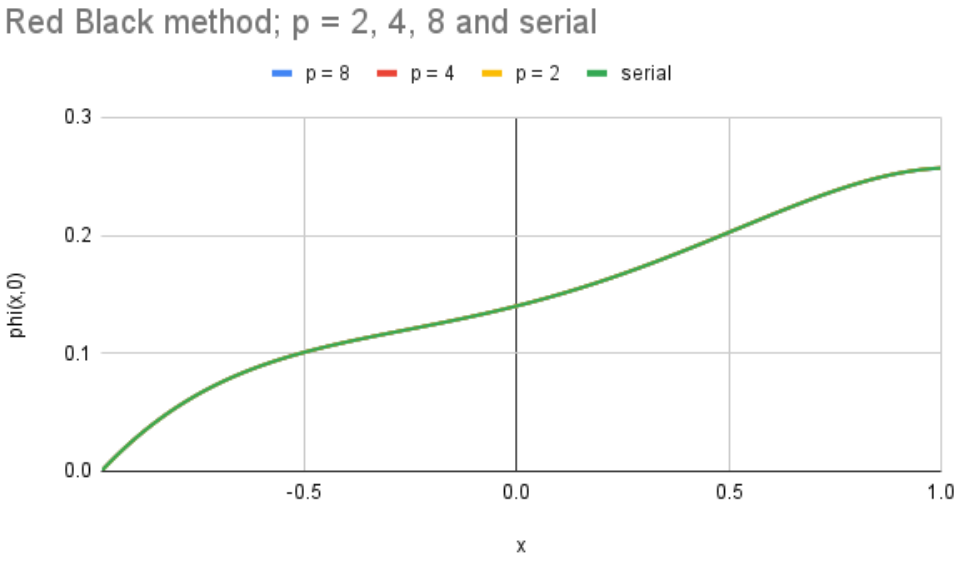




2.3 2(c)

Red Black implementation using MPI. I have done row-wise decomposition, and below are the plots for p = 2, 4, and 8 processors. The serial code has also been plotted to compare the results. $\Delta = 0.01$ is considered here, so the number of grid points increases to 201.



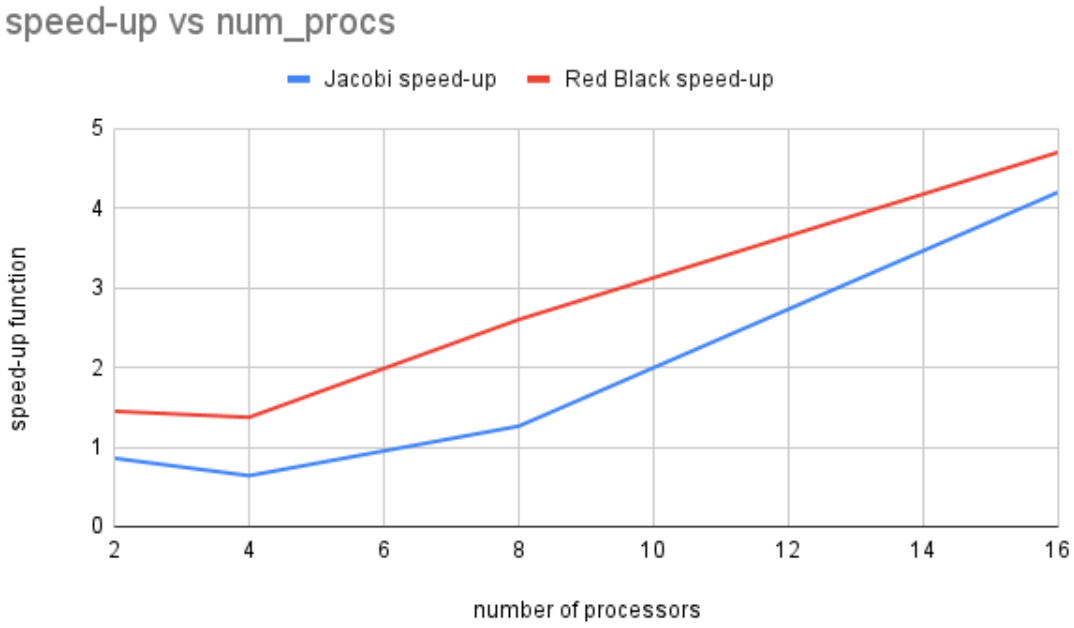


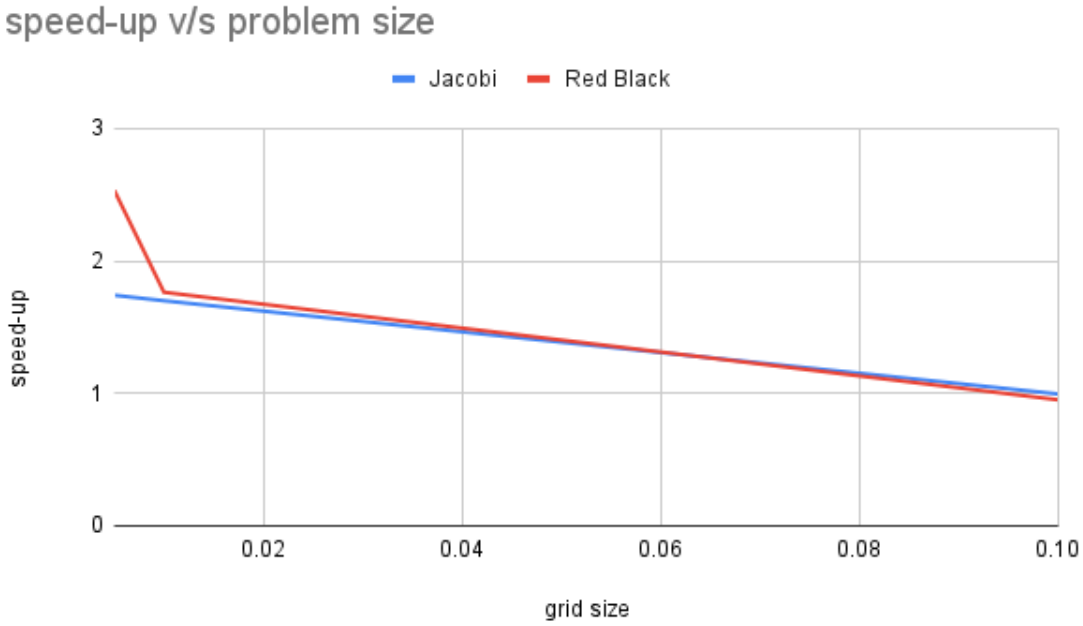
Scheme Used	Grid size	Iterations	p = 2	p = 4	p = 8	serial
Jacobi	0.01	38889	23.533 sec	26.604 sec	14.238 sec	33.911 sec
Red-Black	0.01	24032	15.468 sec	16.550 sec	8.592 sec	21.917 sec

The Gauss-Seidel red-black approach takes only 24000 iterations in comparison to Jacobi, which takes about 39000 iterations. This makes sense as well, given the fact that Gauss-Seidel uses updated values while predicting the next step. It can be observed that the time taken is slightly more for p = 4 processors in comparison to p = 2 processors. This may be attributed to the overhead involved in parallelization.

2.4 2(d)

Let us plot the speed-up as a function of processors as well as problem size for both Jacobi and Gauss-Seidel red-black, which can help us infer the advantages gained by parallelization.





Observing the $\psi(n,p)$ value as a function number of p (keeping n fixed), we can see that Gauss-Seidel red-black has a better advantage. Also, it has to be noted that as processors increase, the speed-up also increases. For Jacobi, $p = 2$ speed-up is less than one, so it is undesirable to parallelize in that case. We wish to have $\psi(n,p)$ as high as possible because it is defined as the ratio of sequential runtime to parallel runtime.

In the case of observing the trend with problem size, it can be clearly seen that speed-up increases as the grid size decreases. In other words, as the number of grid points increases, the advantage obtained from parallelization also increases. For smaller problem sets, the overhead associated will outweigh the advantage obtained from parallelization. Hence, it is preferred to adopt parallel implementation techniques as the refinement increases.