ID5130: Parallel Scientific Computing

Assignment 1 - OpenMP

Date: March 4, 2024 Abhijeet Nair (AE20B101)

Question 1

The given system is:

$$\begin{aligned} 3x_1 - x_2 &= 2 \quad (1)\\ -x_1 + 3x_2 - x_3 &= 1 \quad (2)\\ -x_2 + 3x_3 - x_4 &= 1 \quad (3)\\ -x_3 + 3x_4 &= 2 \quad (4) \end{aligned}$$

Substituting x_1 from Eqn. 1 to Eqn. 2, x_2 from Eqn. 2 to Eqn. 1 and 3, x_3 from Eqn. 3 to Eqn. 2 and 4 and x_4 from Eqn. 4 to Eqn. 3, we have:

$$\begin{aligned} 3x_1 - \left(\frac{1+x_1+x_3}{3}\right) &= 2\\ \implies \frac{8}{3}x_1 - \frac{1}{3}x_3 &= \frac{7}{3}\\ -\left(\frac{2+x_2}{3}\right) + 3x_2 - \left(\frac{1+x_2+x_4}{3}\right) &= 1\\ \implies \frac{7}{3}x_2 - \frac{1}{3}x_4 &= 2\\ -\left(\frac{1+x_1+x_3}{3}\right) + 3x_3 - \left(\frac{2+x_3}{3}\right) &= 1\\ \implies -\frac{1}{3}x_1 + \frac{7}{3}x_3 &= 2\\ -\left(\frac{1+x_2+x_4}{3}\right) + 3x_4 &= 2\\ \implies -\frac{1}{3}x_2 + \frac{8}{3}x_4 &= \frac{7}{3} \end{aligned}$$

Removing 3 from the demoninators, we have:

$$\begin{aligned} 8x_1 - x_3 &= 7 \quad (5)\\ 7x_2 - x_4 &= 6 \quad (6)\\ -x_1 + 7x_3 &= 6 \quad (7)\\ -x_2 + 8x_4 &= 7 \quad (8) \end{aligned}$$

Substituting x_3 from Eqn. 7 to Eqn. 5, x_4 from Eqn. 8 to Eqn. 6, x_1 from Eqn. 5 to Eqn. 7 and x_2 from Eqn. 6 to Eqn. 8, we have:

$$\begin{aligned} 8x_1 - \left(\frac{6+x_1}{7}\right) &= 7\\ \implies \frac{55}{7}x_1 &= \frac{55}{7}\\ 7x_2 - \left(\frac{7+x_2}{8}\right) &= 6 \end{aligned}$$

$$\implies \frac{55}{8}x_2 = \frac{55}{8}$$

$$-\left(\frac{7+x_3}{8}\right) + 7x_3 = 6$$

$$\implies \frac{55}{8}x_3 = \frac{55}{8}$$

$$-\left(\frac{6+x_4}{7}\right) + 8x_4 = 7$$

$$\implies \frac{55}{7}x_4 = \frac{55}{7}$$

Thus, the solution is:

$$\boxed{x_1 = 1,\ x_2 = 1,\ x_3 = 1,\ x_4 = 1} \tag{9}$$

Question 2

LU Decomposition:

Fig. 1 shows the analytical and numerical values for the derivative of the given function computed using the LU decomposition method, with number of grid points, $n = 25$.

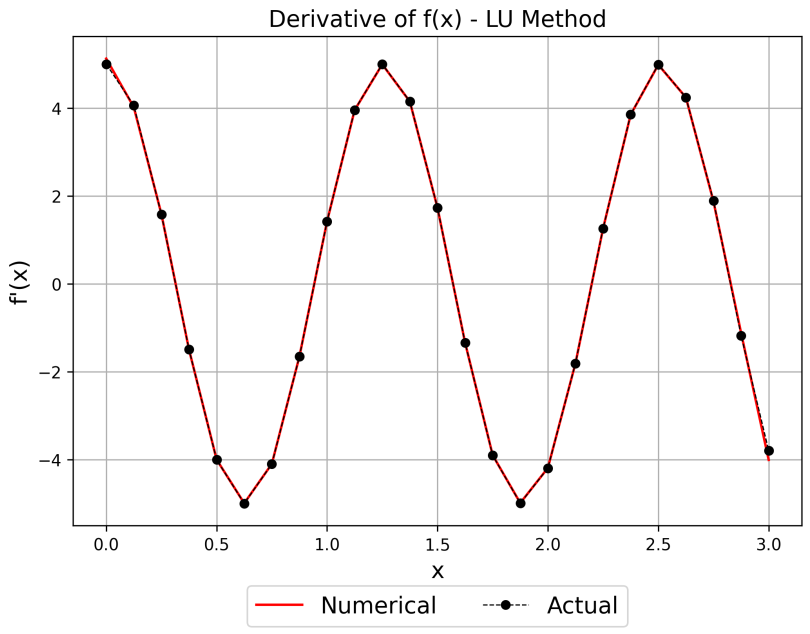

Figure 1: Comparison of analytical and numerical solution

We observe that the solution given by LU decomposition closely follows the analytical solution, with the most error being at the the end points, which is also quite small. The norm of the error is 0.2542.

Recursive-doubling algorithm (RDA):

Fig. 2 shows the analytical and numerical values for the derivative of the function computed using the Recursive-doubling algorithm, with $n = 100$. The parallelized code was run using number of threads, $p = 2$. We can see that solution given by the RDA is almost same as the analytical solution, with the norm of the error being around 0.0025.

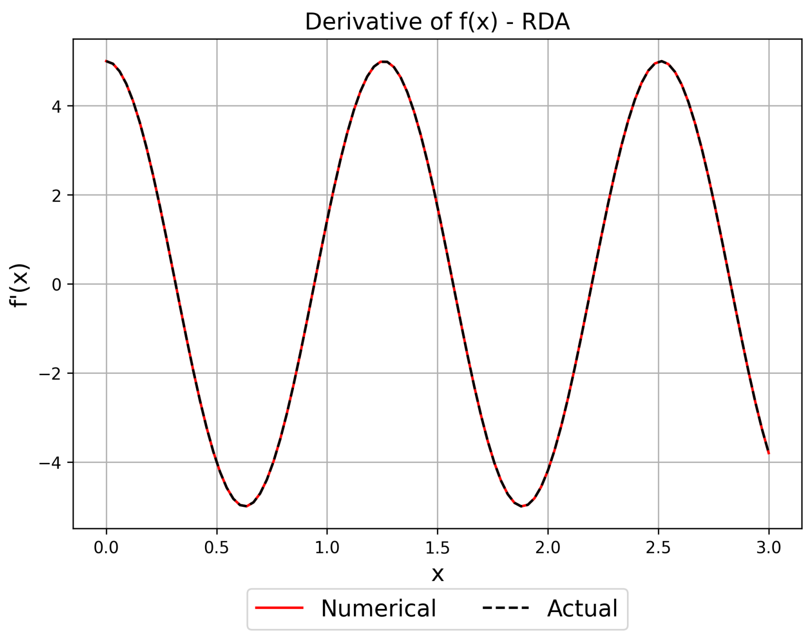

Figure 2: Comparison of analytical and numerical solution

Now, to see the advantage in time that is achievable due to the parallelization of RDA, we plot the time taken for different number of threads. Fig. 3 shows this.

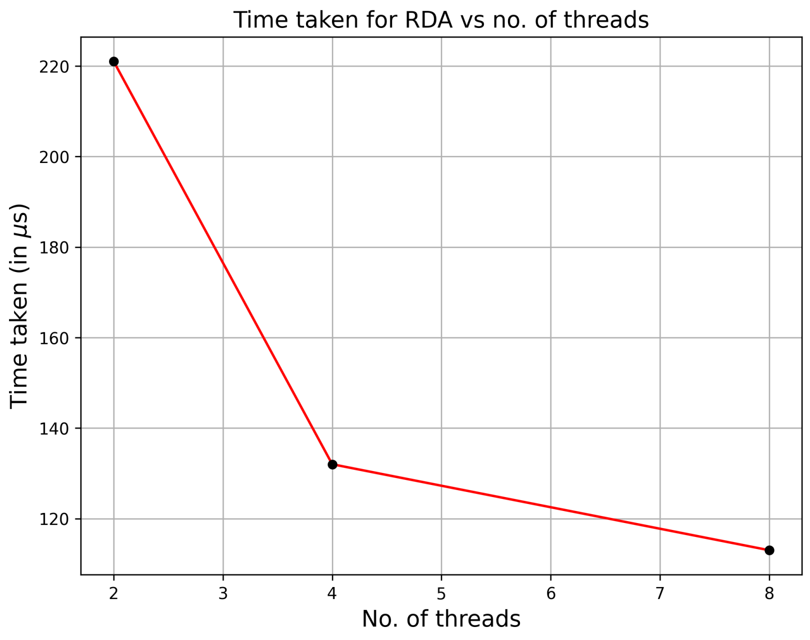

Figure 3: Comparison of time-taken by RDA solver for different thread counts

The time taken decreases almost by half when the number of threads is increased from 2 to 4. But, the improvement from 4 to 8 is not that high. The one possible explanation is that the device the code was run on has 8 threads, and using all of the threads available made it slower.

Question 3

Serial Gauss-Seidel Method:

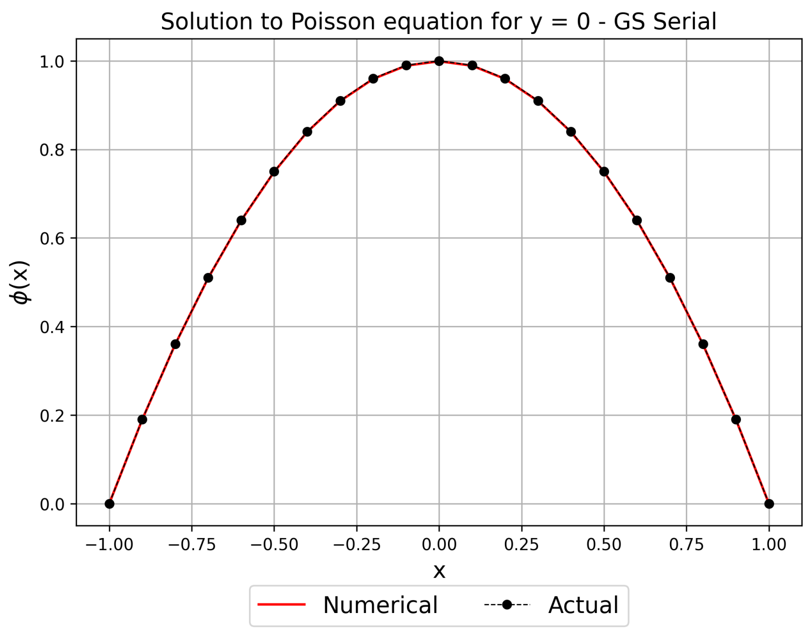

Figure 4: Comparison of analytical and numerical solution of ϕ vs x for $y = 0.5$

Fig. 4 shows the comparison between analytical and numerical solutions of ϕ with respect of x for a constant $y = 0.5$. This solution satisfies the given Poisson equation. The grid size was $\Delta = \Delta x = \Delta y = 0.1$. Given the domain of interest is $-1 \leq x \leq 1$, $-1 \leq y \leq 1$. Which means that there were 21 points along each direction.

The number of iterations taken to achieve a 1% error between the numerical and analytical solution was **283**. We can also observe from the plot that the solution has converged to the actual solution.

Parallel Gauss-Seidel Method:

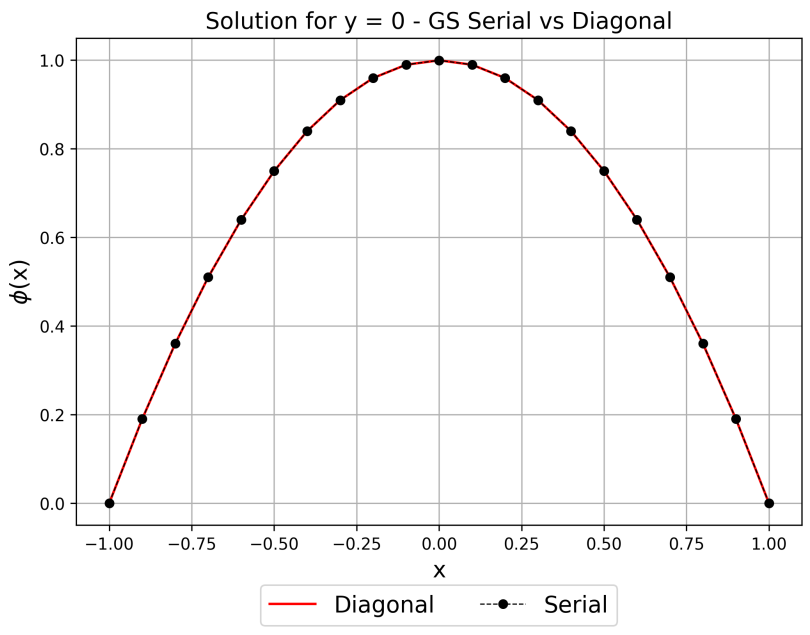

Figure 5: Comparison of analytical and numerical solution of ϕ vs x for $y = 0.5$

Fig. 5 shows the comparison between the solutions from serial GS method and parallelized diagonal GS method. We see that the solutions are matching. Fig. 6 shows the comparison between the solutions from serial GS method and parallelized Red-Black Coloring (RBC) method. We again see that the solutions are matching. Thus, both the approaches to parallelize GS method is validated.

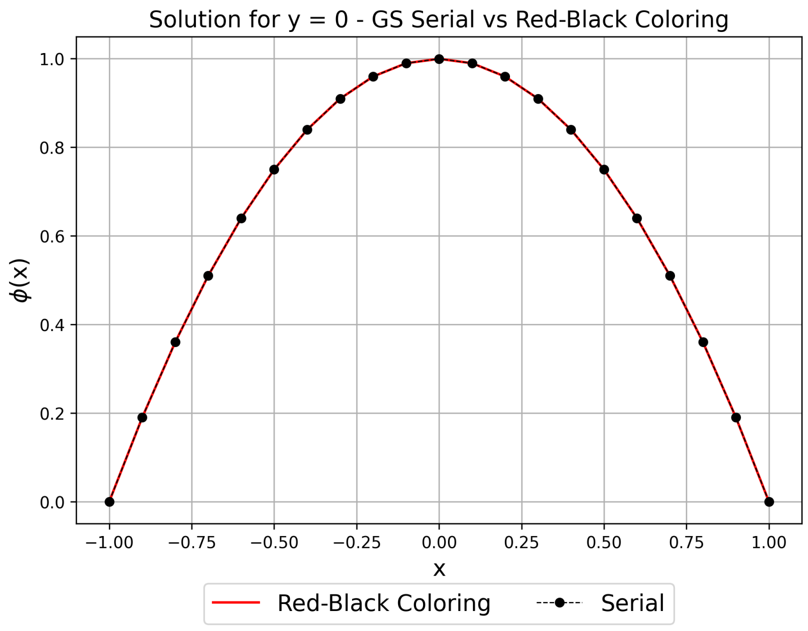

Figure 6: Comparison of analytical and numerical solution of ϕ vs x for $y = 0.5$

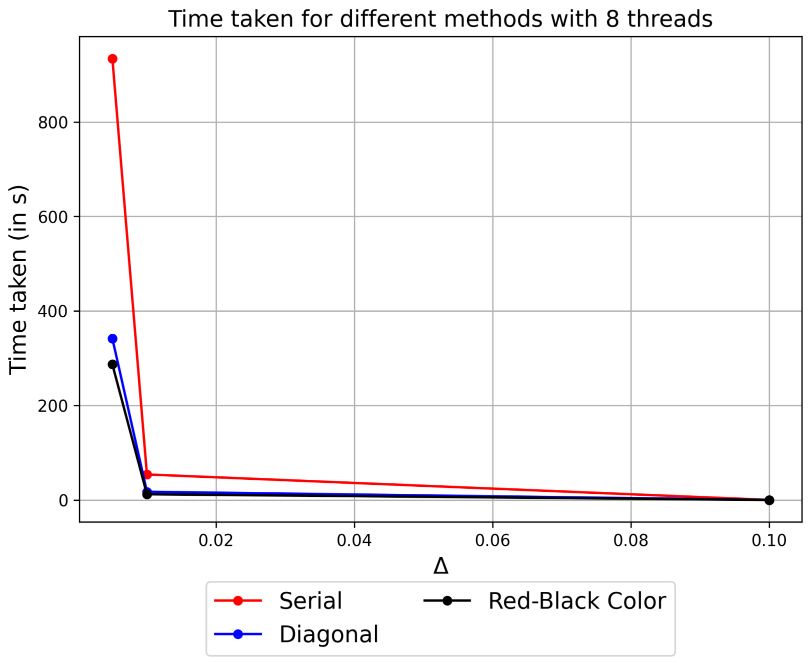

Figure 7: Comparison of time taken for serial and parallel GS programs

Now, we run the serial and parallel programs for grid size $\Delta = 0.1, 0.01$ and 0.005. Fig. 7 shows this comparison. We see that for small grid size, which corresponds to smaller solution matrices, all three programs (serial, diagonal and RBC) take similar amount of time. But as we reduce grid size and increase the number of elements in solution matrix, the serial program takes a lot of time to converge to a solution. But the increase in time for parallel programs is significantly small. For $\Delta = 0.01$, the diagonal and RBC seem to take similar amount of time, but $\Delta = 0.005$, the time taken for RBC method is less than the diagonal method. Thus, we can conclude that the RBC method is better than the diagonal method in terms of time performance.

Note that, all of the codes take the same number of iterations to converge (283 for $\Delta = 0.1$, 37584 for $\Delta = 0.01$ and 161572 for $\Delta = 0.005$). The parallel programs execute the tasks faster and are able to save time. Fig. 8 shows the same plot, but in a log-log scale. We see that the time plot are now more or less linear in log scale, which implies they are exponential in reality. Also, the diagonal method takes more time than serial for $\Delta = 0.1$, which may be due to the overhead required to fork the threads, which is a significant amount for the small problem.

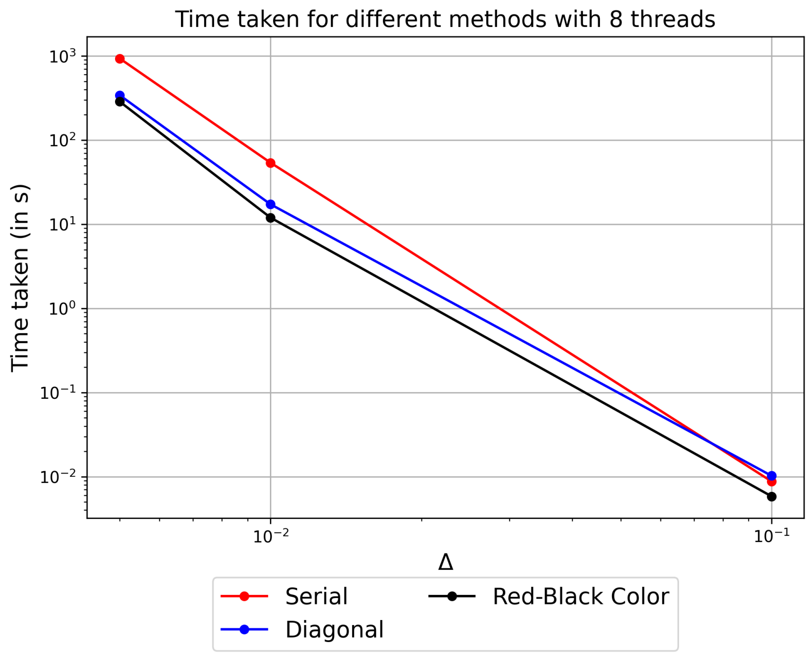

Figure 8: Log-log plot of Fig. 7

Fig. 9 shows the time taken by the parallel programs for $\Delta = 0.005$ for different number of threads (I did not run for 16 threads as my laptop only has 8 threads). We see that the RBC method is able to converge in faster time in comparison with the diagonal method.

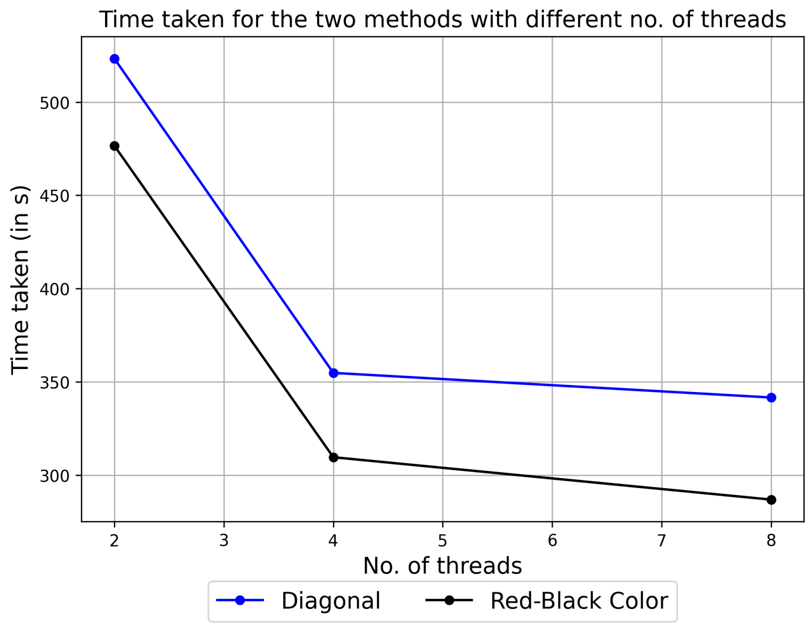

Figure 9: Comparison of time taken for parallel GS programs for different p