ID5130 ASSIGNMENT-3 REPORT

Name: Keerthiharan A Roll no: CH22B026

May 13, 2024

Question 1: Parallelizing Thomas Algorithm

Given function is

$$f(x) = sin(5x) \quad for \quad 0 \leq x \leq 3. \tag{1}$$

The fourth-order accurate Padé scheme for the interior points is given by:

$$f'_{j+1} + 4f'_j + f'_{j-1} = \frac{3}{h}(fj + 1 - fj - 1) \tag{2}$$

The third-order accurate Padé scheme for the exterior points is given by:

$$f'_0 + 2f'_1 = \frac{1}{h}(-\frac{5}{2}f_0 + 2f_1 + \frac{1}{2}f_2) \tag{3}$$

$$f'_n + 2f'_{n-1} = \frac{1}{h}(\frac{5}{2}f_n - 2f_{n-1} - \frac{1}{2}f_{n-2}) \tag{4}$$

where h is the grid spacing and n is the number of grid points in the x direction. Exact solution is given by :

$$f(x) = 5cos(5x) \quad for \quad 0 \leq x \leq 3. \tag{5}$$

Upon Numerically solving this by tridiagonal LU Decomposition/Thomas Algorithm Taking number of grid points as 100, h = 3/100

The result obtained and time taken are plotted below

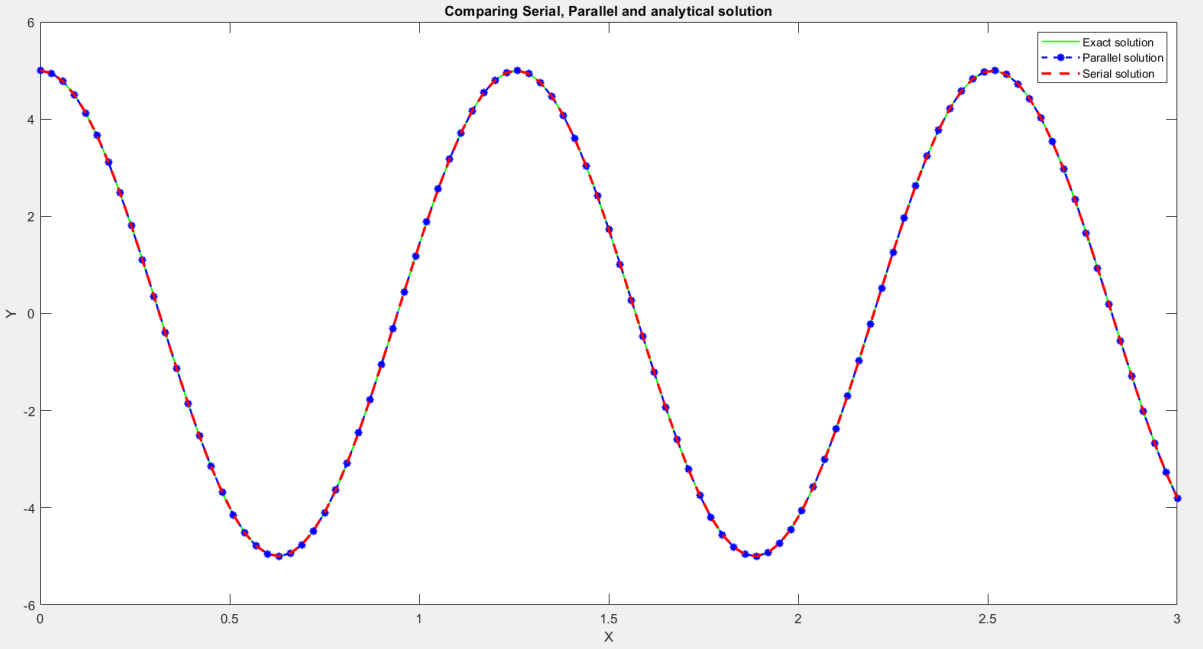

Figure 1: Analytical vs Serial and Parallel LU Numerical Solutions for $n = 100$

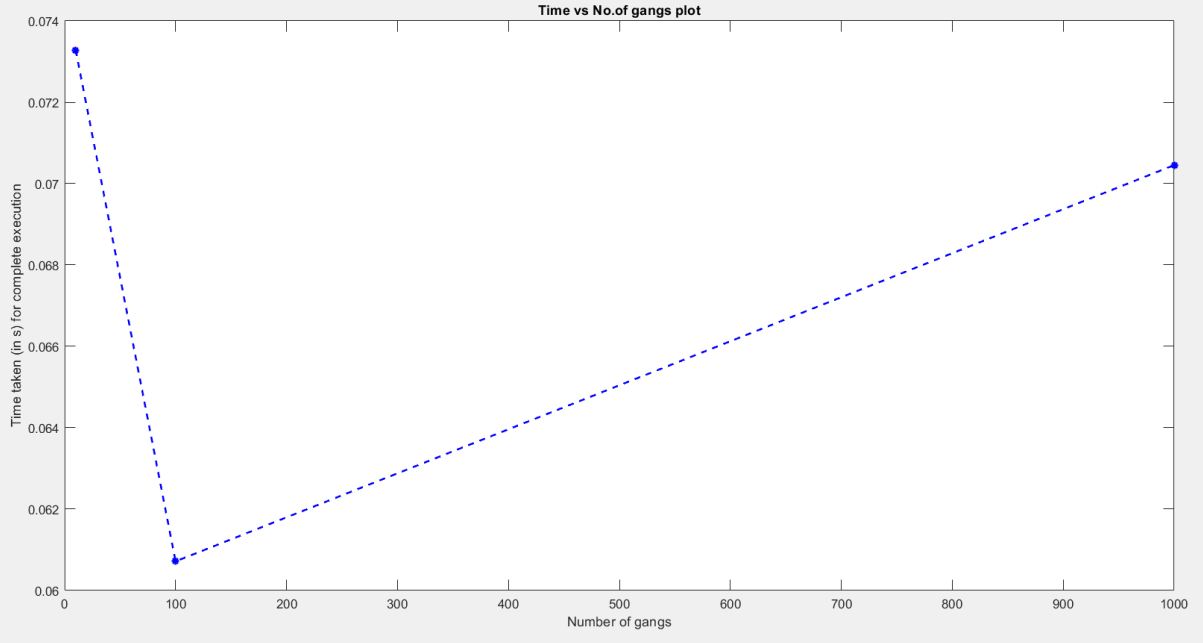

Figure 2: Time taken (in s) vs No.of gangs

The time calculated includes the printing time as well. As the number of gangs available in the system was 128, the performance was best around 100 and drops steadily no.of gangs goes beyond it. As we know there isn't much scope for parallelization in Thomas algorithm / LU Decomposition for tri-diagonal system, the runtime isn't greatly improved.

Question 2 Cholesky Decomposition

Cholesky decomposition is a method to decompose a symmetric positive definite matrix into the product of a lower triangular matrix and its transpose. It is often used in numerical analysis and statistics for solving linear equations and simulating multivariate distributions efficiently.

Suppose we have a symmetric positive definite matrix A. Cholesky decomposition of A gives us:

$$A = LL^T \tag{6}$$

where L is a lower triangular matrix.

The Cholesky decomposition algorithm proceeds as follows:

1. For $i = 1$ to n:
   (a) $l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$
   (b) For $j = i + 1$ to n:
      i. $l_{ji} = \frac{1}{l_{ii}}(a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik})$

This process fills in the lower triangular matrix L such that $LL^T = A$. Note that Cholesky decomposition is only applicable for symmetric positive definite matrices. This form of equation can be used to parallelize the inner j and k loop.
It was ensured that the solution obtained in parallel and serial solutions obtained matches.
For N=10:

1. L matrix obtained from Serial Cholesky decomposition is:

$$L = \begin{bmatrix}
1.000000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.010000 & 0.999950 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.020000 & 0.029801 & 0.999356 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.030000 & 0.039702 & 0.048248 & 0.997595 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.040000 & 0.049602 & 0.057759 & 0.064198 & 0.994224 & 0 & 0 & 0 & 0 & 0 \\
0.050000 & 0.059503 & 0.067270 & 0.073068 & 0.076916 & 0.989029 & 0 & 0 & 0 & 0 \\
0.060000 & 0.069403 & 0.076781 & 0.081937 & 0.084953 & 0.086129 & 0.982008 & 0 & 0 & 0 \\
0.070000 & 0.079304 & 0.086292 & 0.090806 & 0.092990 & 0.093211 & 0.091957 & 0.973307 & 0 & 0 \\
0.080000 & 0.089204 & 0.095803 & 0.099676 & 0.101026 & 0.100294 & 0.098029 & 0.094780 & 0.963157 & 0 \\
0.090000 & 0.099105 & 0.105314 & 0.108545 & 0.109063 & 0.107377 & 0.104101 & 0.099838 & 0.095099 & 0.951805
\end{bmatrix}$$

2. L obtained from Parallel Cholesky decomposition is:

$$L = \begin{bmatrix}
1.000000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.010000 & 0.999950 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.020000 & 0.029801 & 0.999356 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.030000 & 0.039702 & 0.048248 & 0.997595 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.040000 & 0.049602 & 0.057759 & 0.064198 & 0.994224 & 0 & 0 & 0 & 0 & 0 \\
0.050000 & 0.059503 & 0.067270 & 0.073068 & 0.076916 & 0.989029 & 0 & 0 & 0 & 0 \\
0.060000 & 0.069403 & 0.076781 & 0.081937 & 0.084953 & 0.086129 & 0.982008 & 0 & 0 & 0 \\
0.070000 & 0.079304 & 0.086292 & 0.090806 & 0.092990 & 0.093211 & 0.091957 & 0.973307 & 0 & 0 \\
0.080000 & 0.089204 & 0.095803 & 0.099676 & 0.101026 & 0.100294 & 0.098029 & 0.094780 & 0.963157 & 0 \\
0.090000 & 0.099105 & 0.105314 & 0.108545 & 0.109063 & 0.107377 & 0.104101 & 0.099838 & 0.095099 & 0.951805
\end{bmatrix}$$

It can be seen that both serial and parallel gives the same L matrix.

On comparing the time taken for both the methods for different N we get,

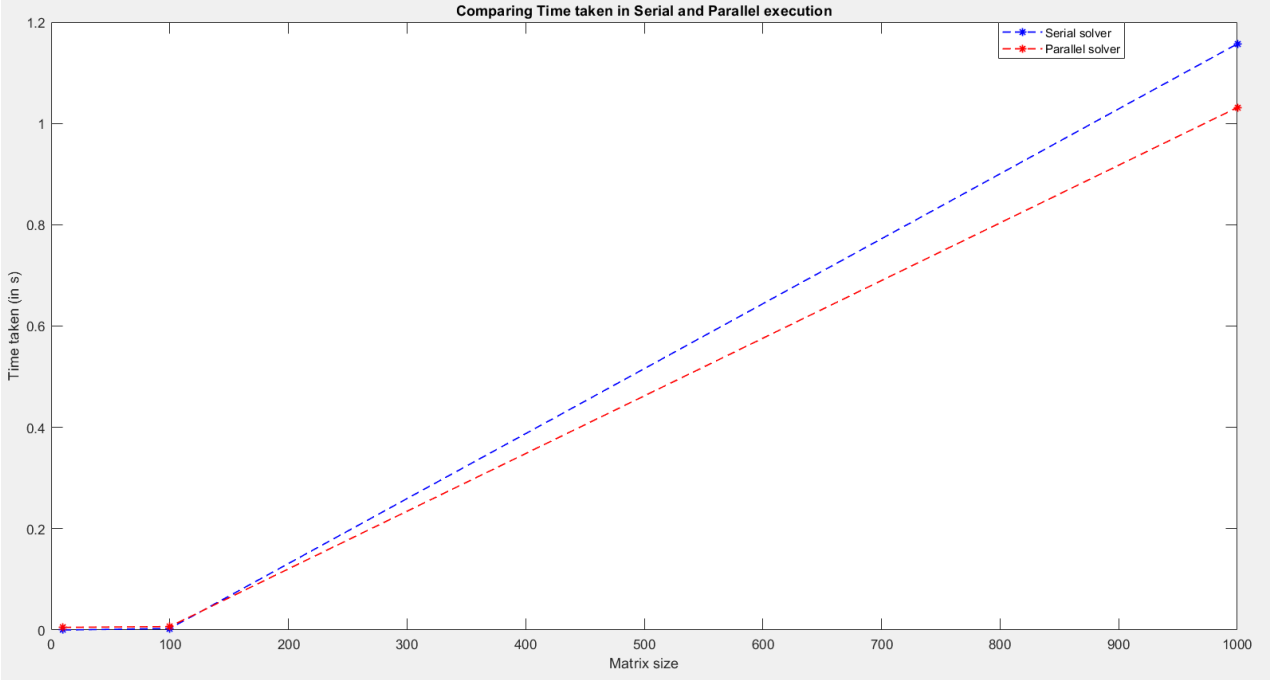

Figure 3: Execution time for parallel and serial solvers vs Size of matrix

From the equation and graph we can see there isn't much scope for parallelism here especially for relatively small matrix sizes, which is reflected on the execution time of parallel code, this is mainly due to the inherent loop carried data dependency of the main loop and data transfer overhead. But as the matrix size increases we tend to get a better speedup which is quite expected. Here, I have used the default number of gangs available (which was 128).