HPC Assignment B

Y3879643

February 3, 2023

1 Part A

The Laplacian for cylindrical polar coordinates is shown below.

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial z^2} + \frac{1}{r}\frac{\partial\Phi}{\partial r} + \frac{\partial^2\Phi}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2\Phi}{\partial\theta^2}$$

Applying the centred-difference formulae to each term in the Laplacian is shown below.

$$\frac{\partial\Phi}{\partial r} \approx \frac{\Phi\left(r+h\right) - \Phi\left(r-h\right)}{2h}$$

$$\frac{1}{r}\frac{\partial\Phi}{\partial r} \approx \frac{\Phi\left(r+h\right) - \Phi\left(r-h\right)}{2rh}$$

$$\frac{\partial^2\Phi}{\partial r^2} \approx \frac{\Phi\left(r+h\right) - 2\Phi + \Phi\left(r-h\right)}{h^2}$$

$$\frac{\partial^2\Phi}{\partial z^2} \approx \frac{\Phi\left(z+h\right) - 2\Phi + \Phi\left(z-h\right)}{h^2}$$

Bringing these together into the Laplacian, as well as $\theta = 0$ is shown below.

$$\nabla^2\Phi \approx \frac{\Phi\left(z+h\right) - 2\Phi + \Phi\left(z-h\right)}{h^2} + \frac{\Phi\left(r+h\right) - \Phi\left(r-h\right)}{2rh} + \frac{\Phi\left(r+h\right) - 2\Phi + \Phi\left(r-h\right)}{h^2}$$

Applying $z = i * h$ and $r = k * h$ and simplifying.

$$\approx \frac{\Phi_{i,k+1} + \Phi_{i-1,k} + \Phi_{i,k+1}\Phi_{i,k-1} - 4\Phi_{i,k}}{h^2} + \frac{\Phi_{i+1,k} - \Phi_{i-1,k}}{2k}$$

Laplacian is equal to 0 and $h = 1$. So,

$$0 = \Phi_{i,k+1} + \Phi_{i-1,k} + \Phi_{i,k+1} + \Phi_{i,k-1} - 4\Phi_{i,k} + \frac{1}{8k}\left(\Phi_{i,k+1} - \Phi_{i,k-1}\right)$$

$$4\Phi_{i,k} = \Phi_{i,k+1} + \Phi_{i-1,k} + \Phi_{i,k+1} + \Phi_{i,k-1} + \frac{1}{2k}\left(\Phi_{i,k+1} - \Phi_{i,k-1}\right)$$

$$\Phi_{i,k} = \frac{1}{4}\left(\Phi_{i+1,k} + \Phi_{i-1,k} + \Phi_{i,k+1} + \Phi_{i,k-1}\right) + \frac{1}{8k}\left(\Phi_{i,k+1} - \Phi_{i,k-1}\right)$$

For the case where $r = 0$, an axis formula is required. This can be found by instead using the Cartesian Laplacian.

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2}$$

$$\approx \frac{\Phi\left(x+h\right) - 2\Phi + \Phi\left(x-h\right)}{h^2} + \frac{\Phi\left(y+h\right) - 2\Phi + \Phi\left(y-h\right)}{h^2} + \frac{\Phi\left(z+h\right) - 2\Phi + \Phi\left(z-h\right)}{h^2}$$

Since $x = rcos\theta$ and $y = rsin\theta$.

$$\approx \frac{\Phi\left(rcos\theta+h\right) - 2\Phi + \Phi\left(rcos\theta-h\right)}{h^2} + \frac{\Phi\left(rsin\theta+h\right) - 2\Phi + \Phi\left(rsin\theta-h\right)}{h^2} + \frac{\Phi\left(z+h\right) - 2\Phi + \Phi\left(z-h\right)}{h^2}$$

Apply $\theta = 0$ and $r = 0$

$$\approx \frac{\Phi\left(h\right) - 2\Phi + \Phi\left(-h\right)}{h^2} + \frac{\Phi\left(h\right) - 2\Phi + \Phi\left(-h\right)}{h^2} + \frac{\Phi\left(z+h\right) - 2\Phi + \Phi\left(z-h\right)}{h^2}$$

System is rotationally symmetric so $h = -h$

$$\approx \frac{4\Phi\left(h\right) + \Phi\left(z+h\right) + \Phi\left(z-h\right) - 6\Phi}{h^2}$$

Applying $h = 1$, $z = i*h$, $r = k*h$ and that the Laplacian is equal to 0.

$$0 = 4\Phi_{i,1} + \Phi_{i+1,0} + \Phi_{i-1,0} - 6\Phi_{i,0}$$

$$6\Phi_{i,0} = 4\Phi_{i,1} + \Phi_{i+1,0} + \Phi_{i-1,0}$$

$$\Phi_{i,0} = \frac{4}{6}\Phi_{i,1} + \frac{1}{6}\left(\Phi_{i+1,0} + \Phi_{i-1,0}\right)$$

$$\Phi_{i,0} = \frac{2}{3}\Phi_{i,1} + \frac{1}{6}\left(\Phi_{i+1,0} + \Phi_{i-1,0}\right)$$

2 Part B

An MPI-based program was written to simulate the problem at hand. The program was compiled and ran using the Viking super cluster, allowing the program to make full use of several processors to increase computational power and speed.

The bash script to submit the job is shown below.

```
#!/bin/bash
#SBATCH --job-name=assignment_b
#SBATCH --ntasks=5
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=5
#SBATCH --cpus-per-task=1
#SBATCH --time=01:00:00
#SBATCH --mem-per-cpu=10gb

module load toolchain/foss/2022a

#mpifort -pg -g -O0 -o mpi_main mpi_main.f90
mpifort -o mpi_main mpi_main.f90

mpirun -np 5 ./mpi_main
```

The first $mpifort$ line is if you want to profile the code, and the second line is a simple compilation. The number of processors and nodes can be changed using the $SBATCH$ options at the beginning of the script. The script can then be submitted using the sbatch system.

In the main program, $mpi\_main.f90$, the variables $scaleFactor$ was first set to 10, w in the $UpdatedPotential$ function was set to 1.0, the convergence value was set to -3 and the number of processors used 1. This would give a base set of results from which to compare to. The ouput is shown below.

```
==============================================================
 Master processor (0) speaking
 Number of processors:           1
 Number of elements per processor:        20000
 ==============================================================
 Buffers allocate of size       20000 for processor           0
 Data scattered to processor           0
 ==============================================================
 Processor           0 took   16.022596312000001      seconds and       28273 steps
 Processor           0 done
 Point A:   417.56483328935633
 Point B:   123.87930566697430
 Point C:   47.461563799585285
```

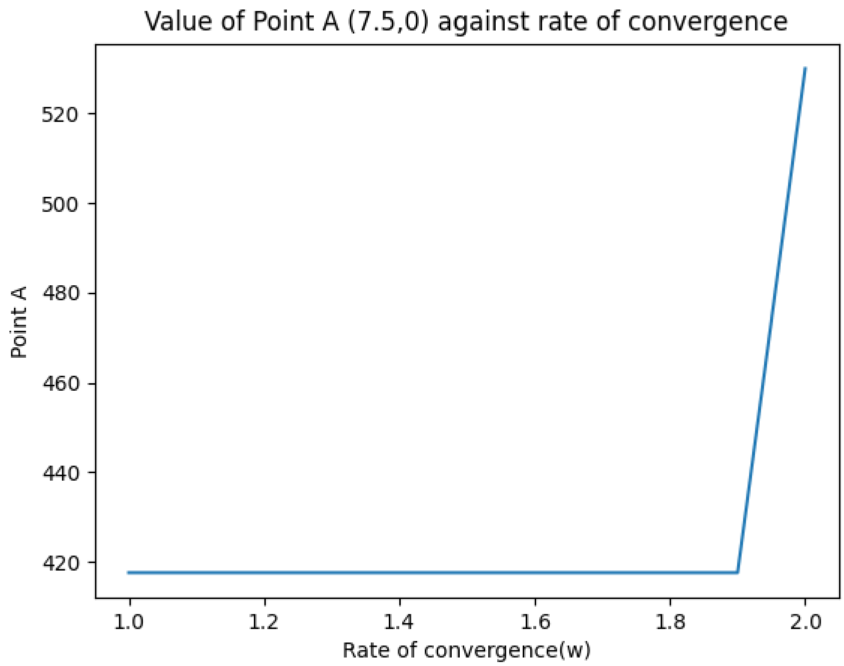

Figure 1: Plot showing how the rate of convergence value effects the measured voltage at point A

The value of w was then varied between 1.0 and 2.0, and plotted against the found values of point A. The plot is shown below in figure 1.

In the program $mpi\_main.f90$, the variable $scaleFactor$ was set to 20, with the rate of convergence set to 1.9 as this gives speed with a stable result correct to 3 decimal places. The number of processors was initially set to 2 and the following results were found.

```
==============================================================
 Master processor (0) speaking
 Number of processors:           2
 Number of elements per processor:        40000
 ==============================================================
 Buffers allocate of size       40000 for processor           0
 Data scattered to processor           1
 ==============================================================
 Data scattered to processor           0
 ==============================================================
 Processor           1 took   8.5642300789999997      seconds and        7623 steps
 Processor           1 done
```

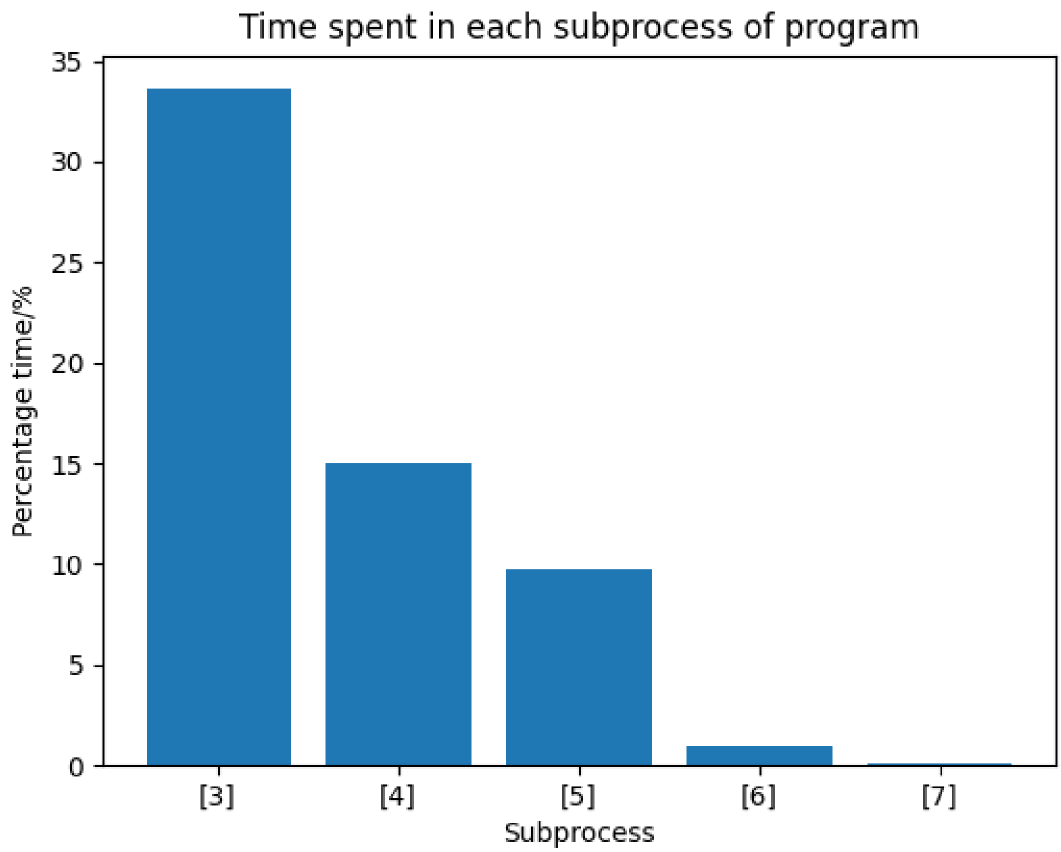

Figure 2: Bar chart showing tiem spent in eahc subprocess

```
Processor           0 took   8.5642306080000008      seconds and        7623 steps
Processor           0 done
Point A:   417.63284370166451
Point B:   122.82182316244787
Point C:   46.873932344894207
```

These results were not correct to 3 significant figure because the rate of convergence is now too high for this more refined grid, however there is a speedup as more processors are used.

When profiling the code, the `FindNeighbours` subroutine was spent in the longest (not including the `main` subprogram). The profiling for the program where the number of processors was 2, and the size of the system was 20.

3 Part C

The results for a system of size 20, w=1.6, and two processors are shown below.

```
Buffers allocate of size       40000 for processor           1
 ==============================================================
 Master processor (0) speaking
 Number of processors:           2
 Number of elements per processor:        40000
 ==============================================================
 Buffers allocate of size       40000 for processor           0
 Data scattered to processor          0
 ==============================================================
 Data scattered to processor          1
 ==============================================================
 Processor           0 took   172.74799756400000      seconds and       60204 steps
 Processor           1 took   172.74805327999999      seconds and       60204 steps
 Processor           1 done
 Processor           0 done
 Point A:   417.63285744830984
 Point B:   122.82182775830651
 Point C:   46.873941423533168
```