MECH 309: Numerical Methods in Mechanical Engineering Department of Mechanical Engineering.

McGill University Final Project: Solution to the Two-Dimensional Heat Equation

Presented by:

Muzammil Fazal: 260610299

Abdul Khan: 260621639

Saad Malik: 260559335

Contents

Introduction

In this report we are asked to numerically approximate the temperature field T (x, y) solution to the following partial differential equation:-

$$-\mathrm{k}(\nabla^2)\mathrm{T} = \mathrm{f} \text{ on } \Omega$$

Where: -

∇ = stands for the gradient differential operator
k= for the thermal conductivity
f = for the heat generated inside the body.

Both scalar k and f are functions of space, i.e. $k(x_i, y_j)$ and $f(x_i, y_j)$ in a system of Cartesian coordinates.

The solution to this two-dimensional heat equation is obtained through extensive use of MATLAB. This problem was broken down into 8 parts. The methodology used to solve each part is explained in detail in this report. All relevant MATLAB codes and graphs are also attached where applicable.

We started off by using Finite-Difference scheme in two dimensions, to suggest an approximation of the quantity $k(\nabla^2)T$ at point (x_i, y_j). To solve the system in MATLAB we were asked to consider $f(x, y) = \exp(-x^2 - y^2)$, $k(x, y) = 2 + \cos(x + y)$ and assume T=0 on the boundary of the domain. A system of grid size 50x50, 100x100 and 200x200 was solved through use of Jacobian, Gauss-Seidel and SOR approaches. For each approach the following graphs were plotted:-

1. Temperature Field (T) in the plate
2. Temperature Field along the center line of the domain (between (0,1/2) and (1,1/2))
3. The convergence of the log of the residual as a function of the number of iterations and CPU Time.

Furthermore, we estimated the order of convergence through the use of (decide Jacobi or Gauss-Seidel) approach and finally solved a new system with radiation by using an appropriate method.

$$-\mathrm{k}(\nabla^2)\mathrm{T} + \mathrm{T}^4 = \mathrm{f} \text{ on } \Omega$$

1. Provide the expression of ∇ in Cartesian coordinates

In 2-D Cartesian coordinate system R^2 with coordinates (x, y) and standard basis or unit vectors of axes $\{e_x, e_y\}$, it is expressed as:

$$\nabla = e_x \frac{\partial}{\partial x} + e_y \frac{\partial}{\partial y}$$

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$$

2. Using a Finite-Difference scheme in two-dimensions, suggest an approximation of the quantity k∇^2T at point $(\boldsymbol{x_i}, \boldsymbol{y_i})$

To approximate the quantity we will expand the equation and write the differentials in terms of finite difference.

$$\mathrm{k}\nabla^2\mathrm{T} = -\mathrm{f}$$

$$-k(x_i, y_j)\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right] = -\mathrm{f}(x_i, y_j)$$

$$k(x_i, y_j)\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right] = \mathrm{f}(x_i, y_j)$$

$$-k(x_i, y_j)\left[\frac{T(x_{i+1}, y_j) - 2T(x_i, y_j) + T(x_{i-1}, y_j)}{[\Delta \mathrm{x}]^2} + \frac{T(x_i, y_{j+1}) - 2T(x_i, y_j) + T(x_i, y_{j-1})}{[\Delta \mathrm{y}]^2} = \mathrm{f}(x_i, y_j)\right.$$

$$[\text{For } \Delta\mathrm{x} = \Delta\mathrm{y}]$$

$$k(x_i, y_j)\left[\frac{T(x_{i+1}, y_j) - 4T(x_i, y_j) + T(x_{i-1}, y_j) + T(x_i, y_{j+1}) + T(x_i, y_{j-1})}{[\Delta \mathrm{x}]^2}\right]$$

3. Function set-up

To formulate an algorithm, the individual nodes will have to be indexed in terms of i, j and n and these indexes are developed by assigning custom values for i, j and n. From a practical point of view, this is a bit more complicated than in the 1D case, since we have to deal with "book-keeping" issues, i.e. the mapping of $T_{i,j}$ to the entries of a temperature vector T(k) (as opposed to the more intuitive matrix T(i,j) we could use for the explicit scheme). If a 2D temperature field is to be solved for with an equivalent vector T, the nodes have to be numbered continuously. The derivative versus x-direction is then:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta \mathrm{x}]^2}\left[T_{i+1,j} - 2T_{i,j} + T_{i-1,j}\right]$$

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta \mathrm{x}]^2}\left[T_{34} - 2T_{33} + T_{32}\right]$$

and the derivative versus y-direction is given by:

$$\left.\frac{\partial^2 T}{\partial y^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta \mathrm{y}]^2}\left[T_{i,j+1} - 2T_{i,j} + T_{i,j-1}\right] \ and\ \left.\frac{\partial^2 T}{\partial y^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta \mathrm{y}]^2}\left[T_{43} - 2T_{33} + T_{23}\right]$$

If n_x are the number of grid points in x-direction and n_Y the number of points in y-direction, we can write the equations in a more general way as:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta x]^2}\ \left[T_{i+1+n(j-1)} - 2T_{i+n(j-1)} + T_{i-1+n(j-1)}\right]$$
$$\left.\frac{\partial^2 T}{\partial y^2}\right|_{i=3\ j=4} = \frac{1}{[\Delta y]^2}\ \left[T_{i+nj} - 2T_{i+n(j-1)} + T_{i+n(j-2)}\right]$$

4. Solving the system using Jacobi Method

We will solve the system of equations generated in previous part by first using the Jacobi method. The Jacobi method is an algorithm for determining the solutions of the diagonally dominant system of linear equations. The algorithm solves each diagonal element and this process is then iterated until we reach convergence. The algorithm was used to obtain the temperature field as well as the centre line temperature for three different grid sizes of 50x50, 100x100 and 200x200. The centre line temperature for the three grids are superimposed on the same graph for easy comparison. Finally a graph for the log of the residual vs CPU time and number of iterations was also generated.

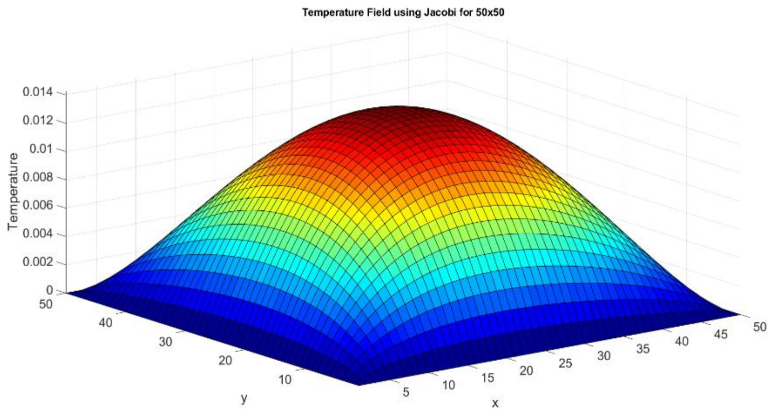

Figure 1: Temperature Field using Jacobi for 50x50

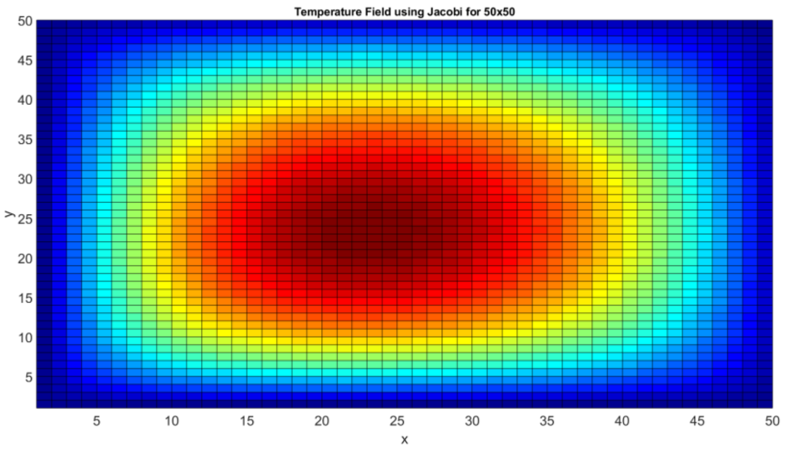

Figure 2: Top of View Temperature Field using Jacobi for 50x50

Figures 1 and 2 are a visualization of the temperature distribution for a 50x50 grid size plate. As can be seen by the color scheme of the contour plot, temperature at the boundaries of the plate is zero (as defined by the project guidelines) and gradually increases to reach the maximum temperature towards the centre region of the unit plate.

On the other hand, Figure 3 represents the temperature distribution over the centerline of the plate. From the trend shown in the graph, we can see that the highest temperature across the centerline is recorded for the 50x50 grid size, whereas the 200x200 grid plots a relatively lower temperature distribution. The reason for this trend can be explained by the change in resolution or step size. However when running the system using Gauss-Seidel SOR as discussed later we see that that the distance between the centre-lines for each grid size does not exhibit that wide a gap as is observed here. This depicts the limitations of the Jacobi method compared to the other two methods where the most accurate representation is observed by solving the system using SOR.

Furthermore, we can also ascertain from the graph that on a unit length plate, the maximum temperature is achieved at 2/5th the distance of the centreline. This can be seen by the fact that all the 3 plots peak at the value of 0.4 on the x-axis. Since the trend is uniform, it invokes confidence in the validity of our findings.

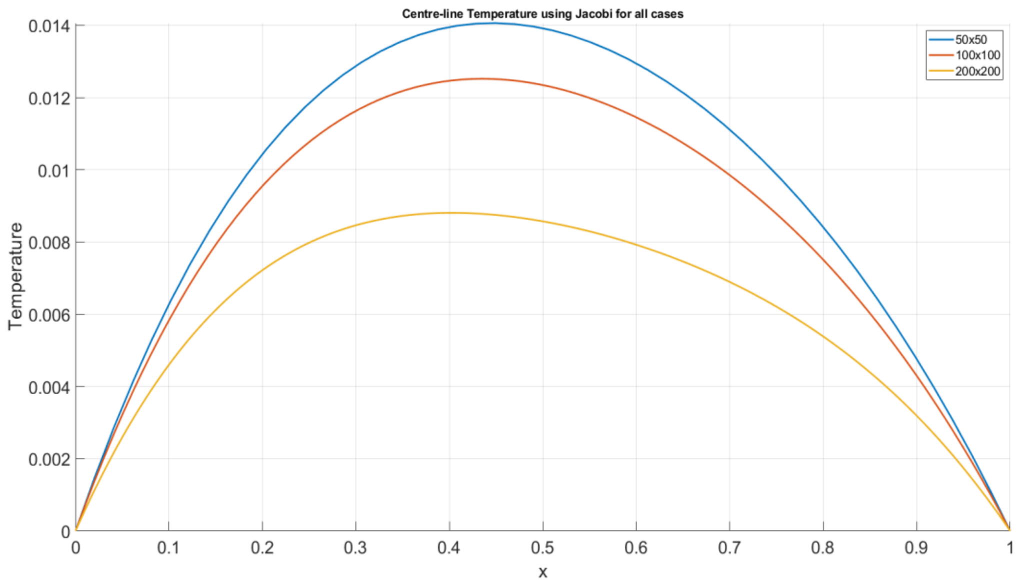

Figure 3: Temperature Field (T) through plate centre-line for all cases grid sizing

5. Plot of Log of Residual vs. Iterations and CPU Time for Jacobi Method

In this section, we plotted the log of Residual against the number of iterations it takes to solve the system using the Jacobi method.

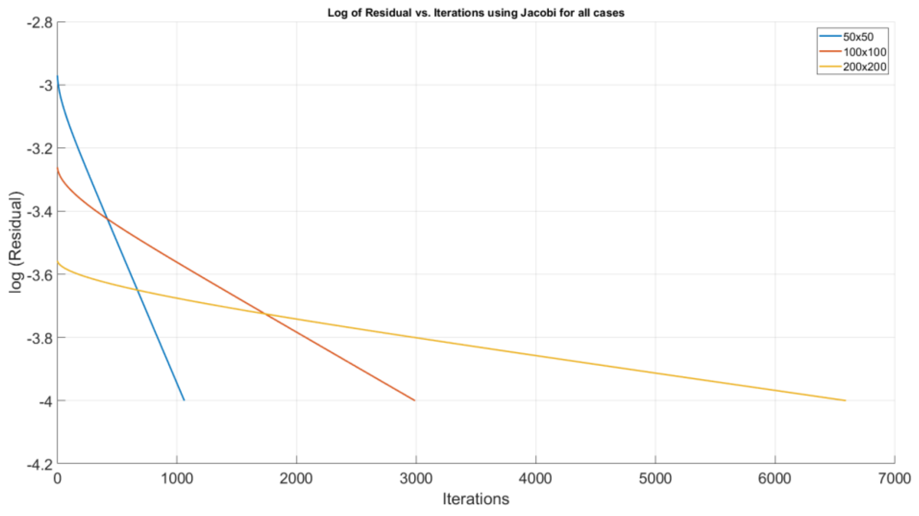

Figure 4: Log of Residual vs Iterations using Jacobi for all cases

As we can see in Figure 4, the 50x50 grid took the least number of iterations followed by the 100x100 and then finally, 200x200 has the most number of iterations. The trend can be validated by th e fact that since the 50x50 has the least number of nodes, therefore it takes the least number of iterations e.g. if *n* (number of nodes) = 50 then the number of operations performed are n^2. Thus, as n increases to 100 and 200, the subsequent number of operations are increased and thus, the overall number of iterations also increases.

Figure 5, meanwhile, supports the trend already seen in the previous graph i.e. as the grid size, and therefore, the number of iterations are increased the CPU also takes more time to process the operations. An interesting trend to observe is that between the grid sizes of 50 to 100, there is only a 4 second gap. However, between sizes 50 and 100, there is a substantial 35 second gap.

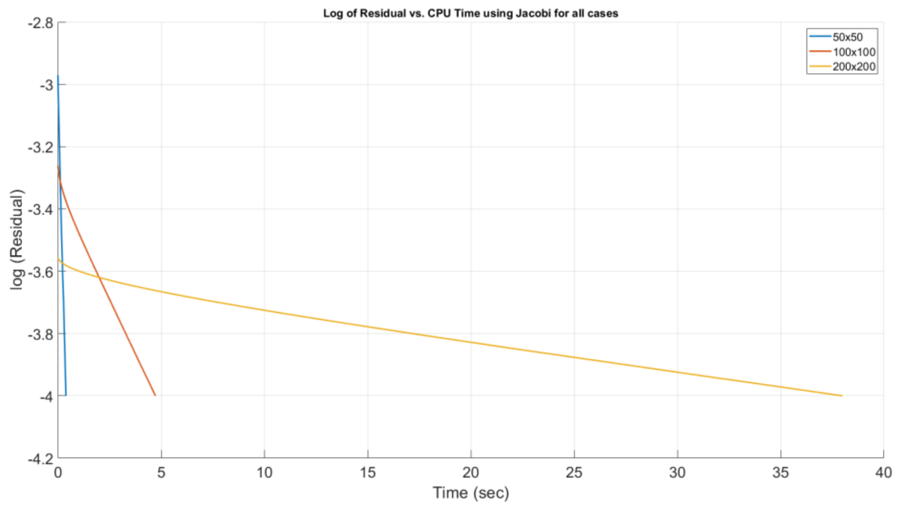

Figure 5: Log of residual vs CPU Time using Jacobi for all cases

6. Solving the system using Gauss-Seidel and SOR

We are then asked to solve the same system through the use of Gauss-Seidel and SOR approach. The only difference between Jacobi and Gauss-Seidel is that in the latter, the most recently updated values of the unknowns are used at each iteration. The successive over relaxation (SOR) approach takes the Gauss-Seidel direction towards the solution and attempts to speed convergence by multiplying the latest iteration by a weighted value "*w*".

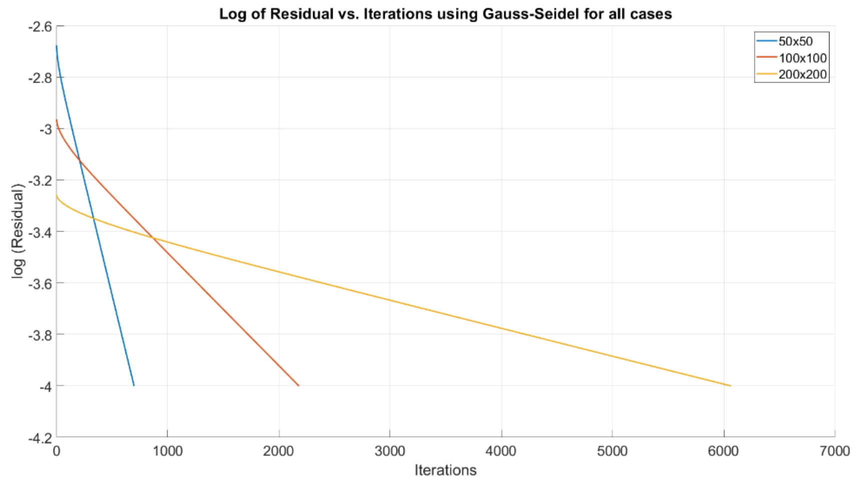

Figure 6: Log of residual vs Iterations using Gauss-Seidel for all cases

The trend for Gauss Siedel is similar to the one for Jacobi i.e. the number of iterations and subsequent CPU processing times are the highest for 200x200 gridsize and decrease as grid size decreases. However, what differentiates Gauss Siedel from Jacobi is that Gauss Siedel is considerably faster. It uses lesser iterations to achieve the final value, in a lesser overall time. This trend is in agreement with the theory that the Gauss-Seidel method is superior to the Jacobi method in terms of accuracy and processing time because, Gauss-Seidel uses the most recently updated values of the unknowns at each iteration.

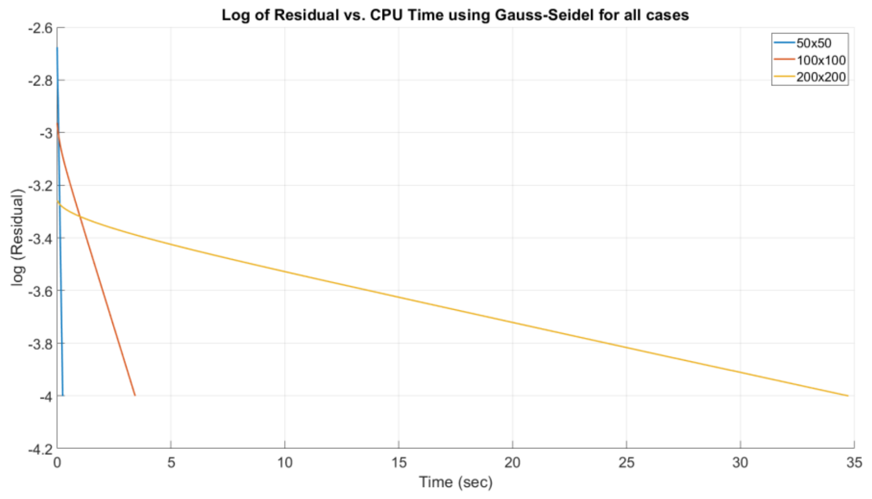

Figure 7: Log of residual vs CPU time using Gauss-Seidel for all cases

Now, we plot the graph for the residual of log against the number of iterations for all 3 iterative methods (Jacobi, GS and SOR) for the 100x100 grid size. Gauss-Seidel and SOR methods are essentially the same, with the exception of a relaxation perimeter (w) being multiplied to the SOR. For the purpose of our project we calculated the value for w by using the following formula obtained from our textbook, which related "w" to the grid size "n" that we are using:

$$w = \frac{4}{2 + \sqrt{4 - \left[2\cos\left(\frac{\pi}{n}\right)\right]^2}}$$

In our result, we can clearly see that not only is the SOR is a much faster method overall but that it also uses the least amount of iterations by a huge margin, when compared to Jacobi and GS (Figure 8 and 9). This result is completely validated by theory since SOR essentially takes Gauss-Seidel direction towards the solution and attempts to speed convergence, which is exactly what happened in this case.

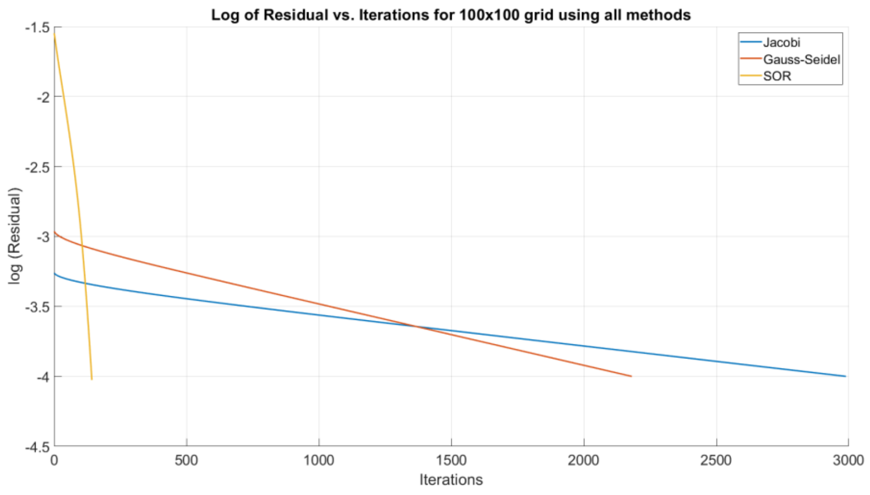

Figure 8: Log of residual vs Iterations using all methods for a 100x100 grid

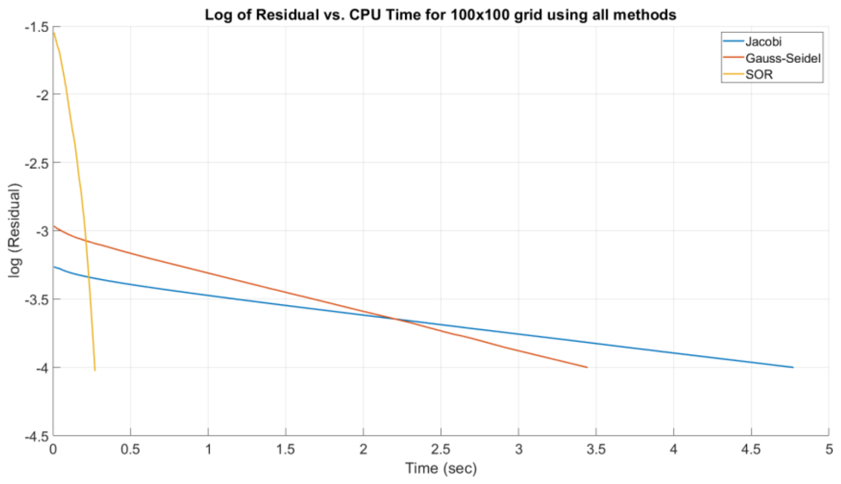

Figure 9: Log of residual vs CPU time using all methods for a 100x100 grid

7. Estimate the order of convergence of the developed scheme using either the Jacobi or Gauss-Seidel approaches.

For this section the Jacobi method was used and the order of convergence of each of our previous grids (50,100,150,200) was compared to our exact solution which we assumed to be a 250x250 grid. From the graph obtained below we can see that the slope of the curve is approximately 2 which means that the order of convergence is quadratic when implementing the Jacobi method.

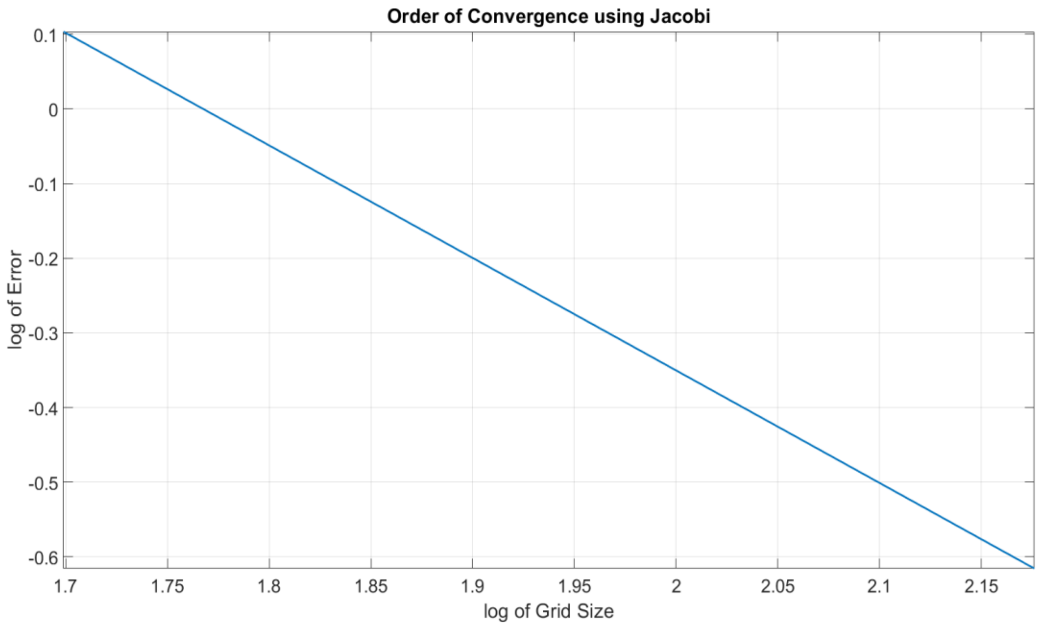

Figure 10: Order of Convergence using Jacobi method

8. Introduce radiation in the problem and solve using an appropriate method.

Our method of choice for this part of the problem where we had to solve for a non-linear equation was the Newton's method. We then compare the norm of the temperature difference vs. Iterations and CPU Time for all 3 grids against the Jacobi Method which we discussed earlier.

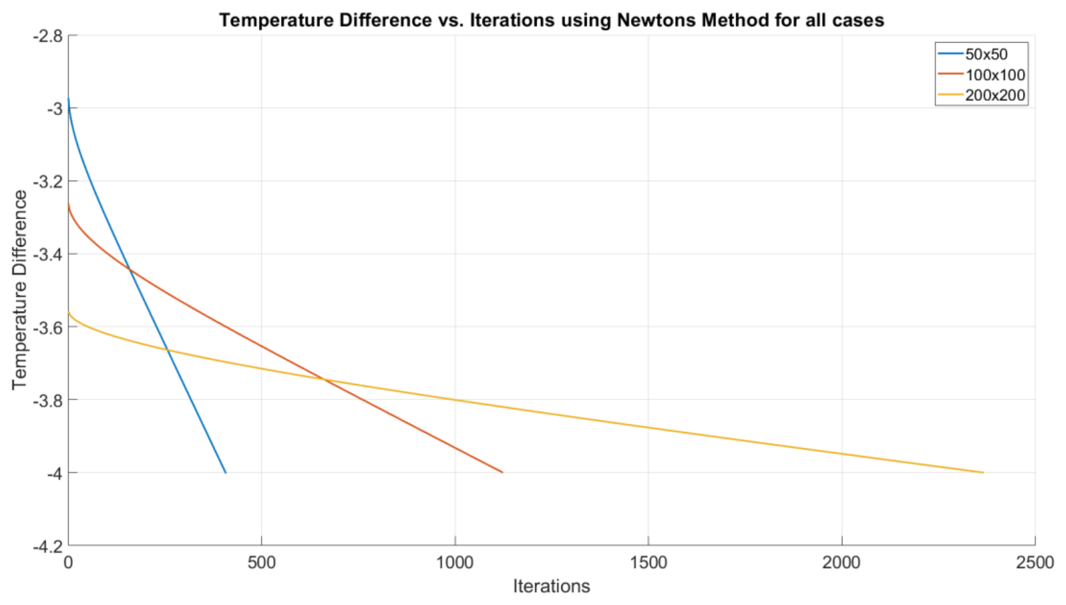

Figure 11: Convergence of Temperature difference vs Iterations using Newton's method

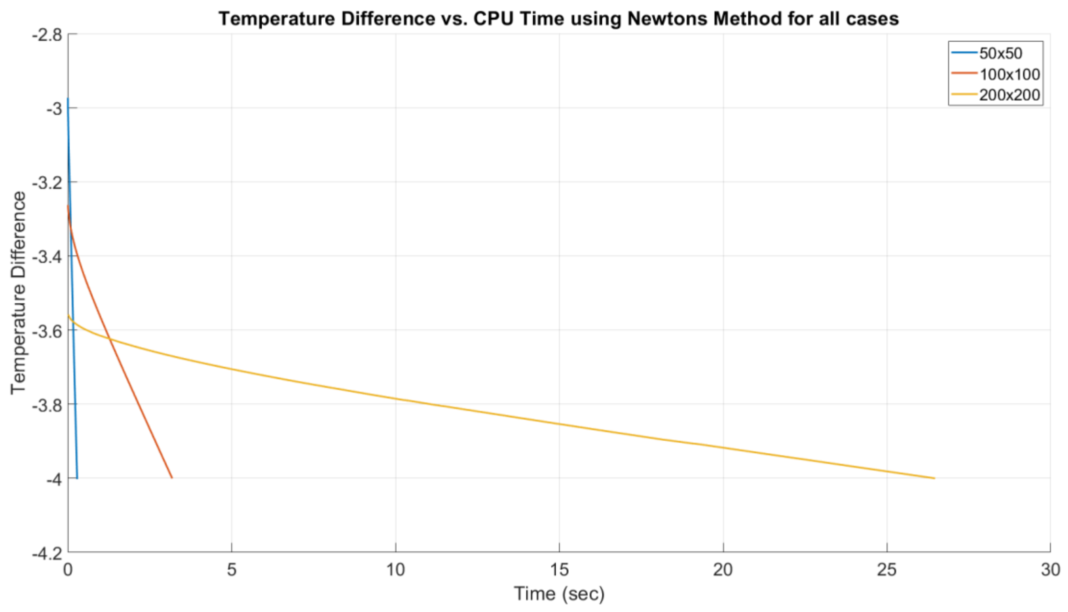

Figure 12: Convergence of Temperature difference vs CPU Time grid using Newton's method

Based on these results we can observe that upon introducing radiation into the problem, the problem becomes non-linear and therefore the Newton's Method is used for solving the system by slightly modifying the code we used for the Jacobi method. In both the cases for iterations and CPU Time we can observe that the Newton's Method is clearly faster than the Jacobi since the max grid of 200x200 is solved under 26 seconds using 2300 iterations whereas the Jacobi uses 6600 iterations to solve the same grid in 37 seconds.

9. Conclusion

Overall, the findings presented in this report via the use of various charts and graphs are in agreement with the different theoretical concepts concerning the systems of linear and non-linear equations. Based on our results it was observed that, for systems of linear equations using various grid sizes, the SOR method performed the fastest and used the least number of iterations followed by Gauss Seidel and then Jacobi methods respectively. Furthermore, it was also observed that for a non-linear system, the Newton's method is not only quite effective but was also relatively easy to implement by slightly modifying the existing Jacobi code to incorporate an additional element in the matrix diagonal i.e. the radiation aspect that was introduced for the problem. To conclude, through the various systems that have been solved in this report, not only did we get to practically visualize various concepts taught within the course (via a real-life practical problem) but were also able to validate the theory behind those concepts. This has left us with a greater appreciation for numerical methods and their vital role in solving complex engineering problems.