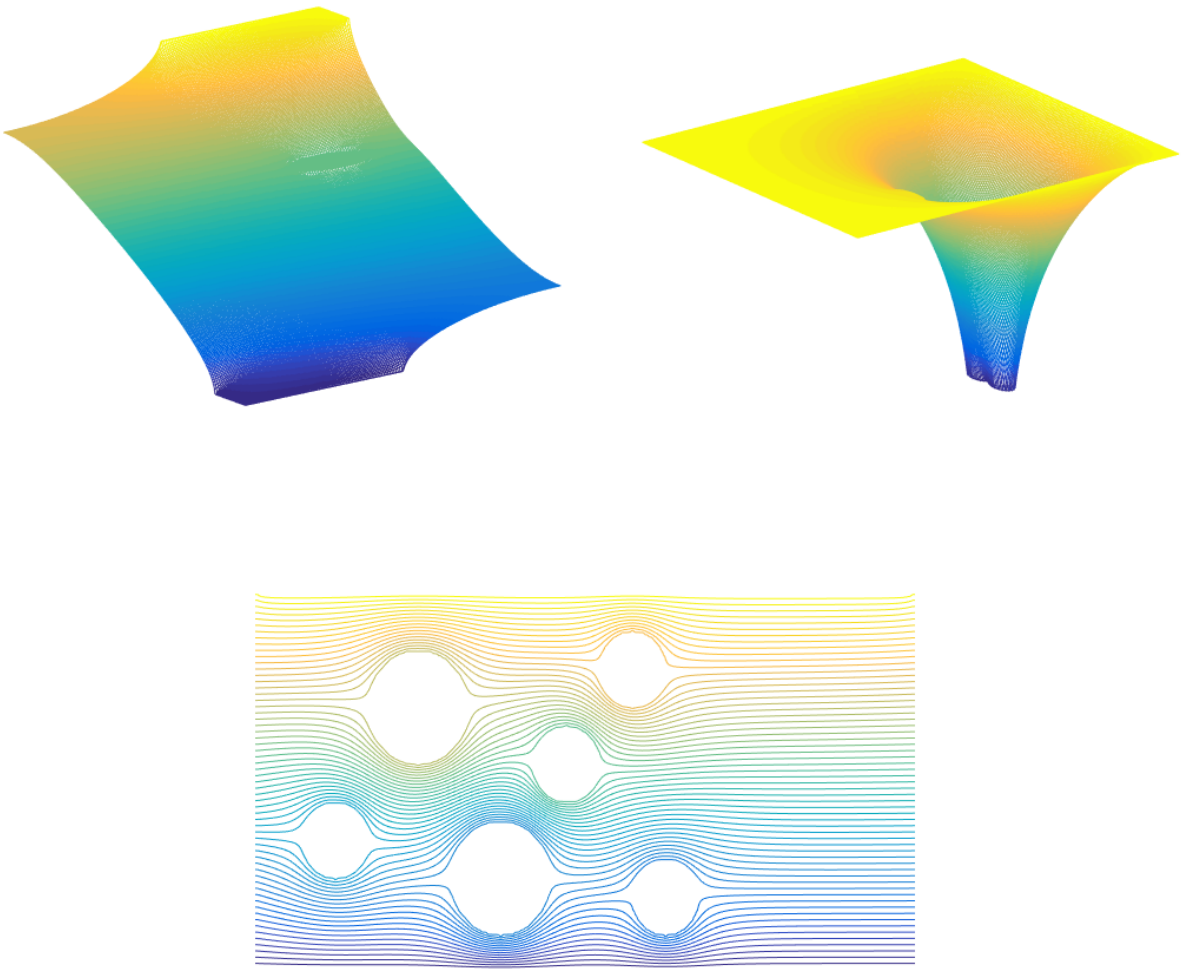
Computer-based Modeling 2

Dongdong Chen, Kyle Zhao, Thomas Meehan

December 8, 2017



1 Introduction

Modeling is a really powerful tool to analyze physical problems. In engineering, some system behavior is not feasible to be achieved by experiments or the experiment is high cost which is not worthy to conduct. Therefore, modeling is a good choice for researcher to gain the preliminarily approximate results.
Modeling is process to apply physical principles to corresponding systems to derive mathematical models. In this report, it demonstrates three different physical problems — Thermo-dynamics, Fluid-dynamics and Structural-dynamics. The methods and results of simulation for corresponding systems are shown in order to analyze the difference between methods.

2 Method

There are two methods applied to solve these three problems which are normal relaxation method(Jacobi's Method) and Successive Over-relaxation(SOR). The first method is chosen to model problem B and C which are fluid dynamics and structural dynamics respectively. The other one is for problem A, thermo-dynamics. This is, on the one hand, because SOR is a faster way compared with the Jacobi's method. On the other hand, problem A needs more cycles to reach the requirement($errc/erro > 1e^{-6}$) than that of problem B and C if Jacobi's method applied on all of them.

2.1 Multi-grid and Matrix replacing loops

In order to increase the running speed of basic program which included 'for' loop, 'while' loop and Jacobi's method, matrix has to be applied to replace the loops which are not necessary firstly. Therefore, function 'find' and 'meshgrid' are used to identify the positions of highly conducting fluid, cylinders and feet of three problems respectively so that values can be fast easily assigned by matrix. Besides, matrix also used to calculate iteration instead of 'for' loop. Secondly, simplified multi-grid method is also greatly helpful to accelerate program running speed. On these three problems, the plate is assumed to be consisted of a certain amount of grids. The calculation of Jacobi's method is based on a coarse grid firstly and then, the grid is expanded to be with doubled resolution.[1] This step is repeated three times until program has finest grid. Here, function 'interp2' is chosen to enlarge grids rather than function 'repelem' and 'repmat' due to its linear interpolation between every two elements.

Methods	**SOR with Multi-grid**	**SOR Without Multi-grid**
Iteration	1504	5685

Table 1: Effects of Multi-grid

This table above ;is based on two programs written with same maths formula and tolerance but one is with multi-grid method and the other one is not. It clearly shows that the one with multi-grid method has much less cycles compared with the other one.

2.2 Jacobi's Method and SOR

Apart from these, the SOR and Jacobi's method are compared. Jacobi's method is a basic way to solve Poisson equation and it continuously calculate new solution based on its adjacent elements till requirement is achieved. From the equation (1)[6] and figure 1, it is obvious that every new iteration is the average value of its four neighboring values and $f_{i,j}$. However, it cannot converge as quick as the SOR. Their formulas are shown by equation (1) and (2).
Relaxation method:

$$\phi_{i,j}^{m+1} = \frac{\phi_{i-1,j}^{m} + \phi_{i+1,j}^{m} + \beta^2(\phi_{i,j+1}^{m} + \phi_{i,j-1}^{m}) - \Delta x^2 f_{i,j}}{2 + 2\beta^2} \tag{1}$$

Successive Over-relaxation:

$$\phi_{i,j}^{m+1} = (1-\omega)\phi_{i,j}^{m} + \omega\frac{\phi_{i-1,j}^{m+1} + \phi_{i+1,j}^{m} + \beta^2(\phi_{i,j+1}^{m} + \phi_{i,j-1}^{m+1}) - \Delta x^2 f_{i,j}}{2 + 2\beta^2} \tag{2}$$

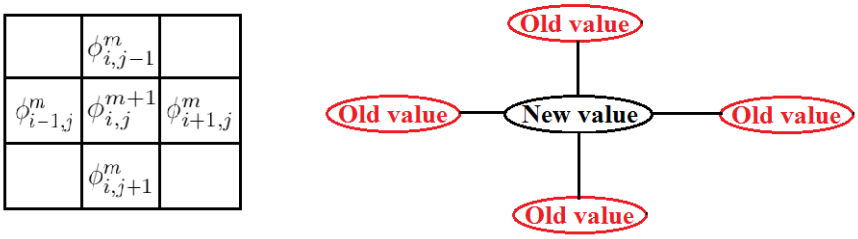


Figure 1: Relaxation method

where $\beta = \Delta x/\Delta y$ and ω is relaxation factor. $\phi^{m+1}_{i,j}$ means that relaxation for $\phi_{i,j}$ has been processed m+1-th times.
SOR is generalized from the Gauss-Seidel Method and is is same as Gauss-Seidel algorithm as ω equals 1[3]. The SOR is similar to Jacobi's method since it calculates every element iteratively and its linear combination of the surrounding elements. Nevertheless, there are two improvements of the SOR with comparison of Jacobi's method. The first one is to evaluate new solutions by its adjacent updated values[2]. From figure 2, it is clear that updated values of $\phi^m_{i-1,j}$ and $\phi^m_{i,j-1}$ have been available to gain the value of $\phi^{m+1}_{i,j}$ since these two updated values have been obtained from earlier calculation in this iteration.

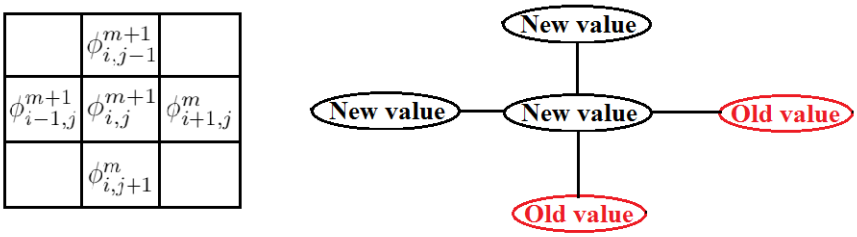


Figure 2: Successive Over-relaxation method

As shown in figure 1 and 2, one of the differences between SOR and Jacobi's method is the application of available updated value. However, the update order cannot be directly parallelized since the every new value calculation is determined by the previously relatively updated values. Therefore, iteration is conducted according to the Red-Black order which is shown in figure 3.

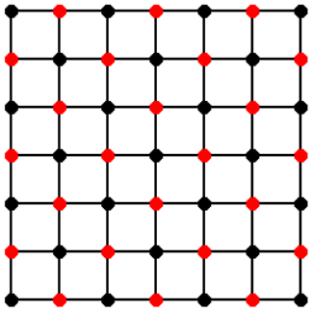


Figure 3: Red-Black Order of grid points
[6]

It is demonstrated that red points are surrounded by the black points and black points also have only red

points as neighbors. Therefore, the updating process can be divided into two parts, one is for red points updating and the other one is for black points. These two are parallel steps which can be implemented. Nevertheless, this way cannot significantly accelerate the convergence. Hence, the second improvement is provided, called Successive Over-relaxation. This method is to solve linear system of equations Ax = b which is derived with basis of the Gauss-Seidel method.
This idea is to correct the old values of grid by the most up-to-date value of ϕ which is shown in equation (3).

$$\phi_{i,j}^{m+1} = \phi_{i,j}^{m} + \boldsymbol{\omega}\left(\frac{\boldsymbol{\phi_{i-1,j}^{m+1}} + \boldsymbol{\phi_{i+1,j}^{m}} + \boldsymbol{\beta^2}(\boldsymbol{\phi_{i,j+1}^{m}} + \boldsymbol{\phi_{i,j-1}^{m+1}}) - \boldsymbol{\Delta x^2 f_{i,j}}}{\boldsymbol{2 + 2\beta^2}} - \boldsymbol{\phi_{i,j}^{m}}\right) \tag{3}$$

The bold part on right hand of equation above is the correction of old value and it can improve the accuracy of solution.[5]
As demonstrated above, the SOR method can be transformed into the Gauss-Seidel method as ω is equivalent with 1. If ω is beyond the scope from 0 to 2[4], the convergence of the SOR method will fails. Additionally, when ω is smaller than 1, it produces a under-relaxation which not needed. Besides, it is generally impossible to identify the value of ω which can maximize the speed of convergence of SOR. However, some estimate can be referred, such as $\omega = 2 - O(h)$ and h is the grid resolution of domain.

3 Code Running and results

3.1 Graphical Results

To increase user friendliness, a GUI is created as shown in figure4. The top of the interface is where the plots are displaced and variables can be inserted at the bottom panels. Three sets of panels are integrated into same side, which makes the GUI clean and tidy. Position of circle, cylinders, rip and gymnast can be changed by inserting different x and y coordinates in corresponding boxes of button panels. Figure 5 and figure 6 are the other two graphical results, the right side are the convergence plots. For each convergence plot, the range of y-axes is too large so that user is hardly to observe any change. Hence, the logarithmic scale is used to narrow the distribution of y-axes, the trend of the curve is clearly shown in Gui.

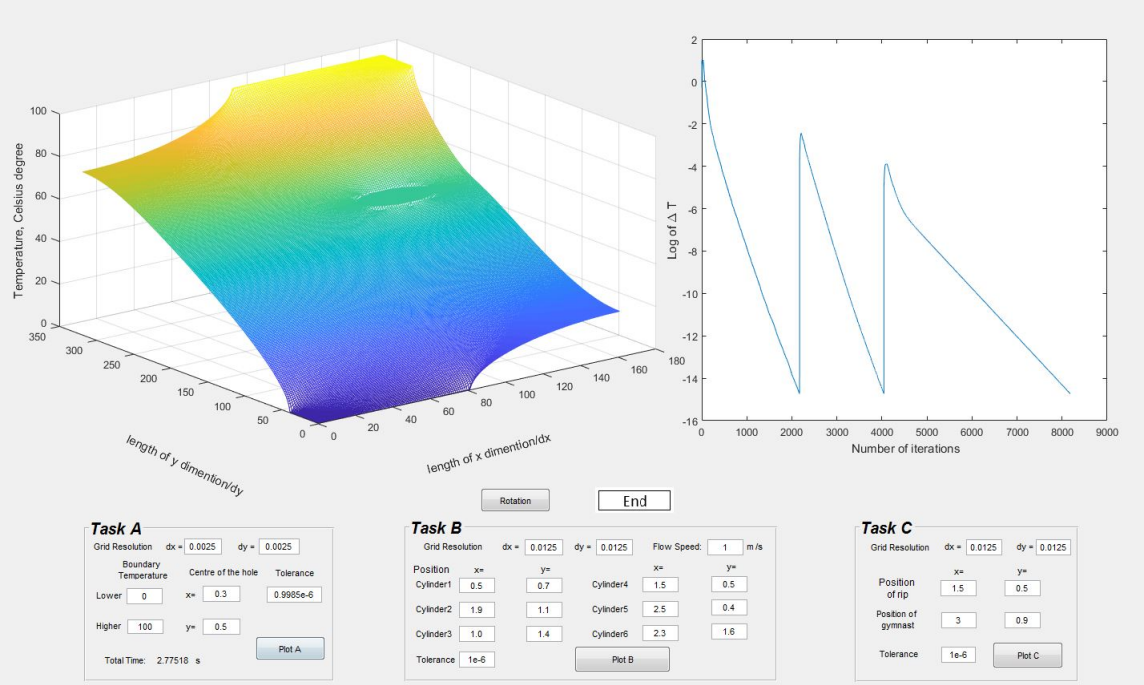


Figure 4: Graphical User Interface and result of task A

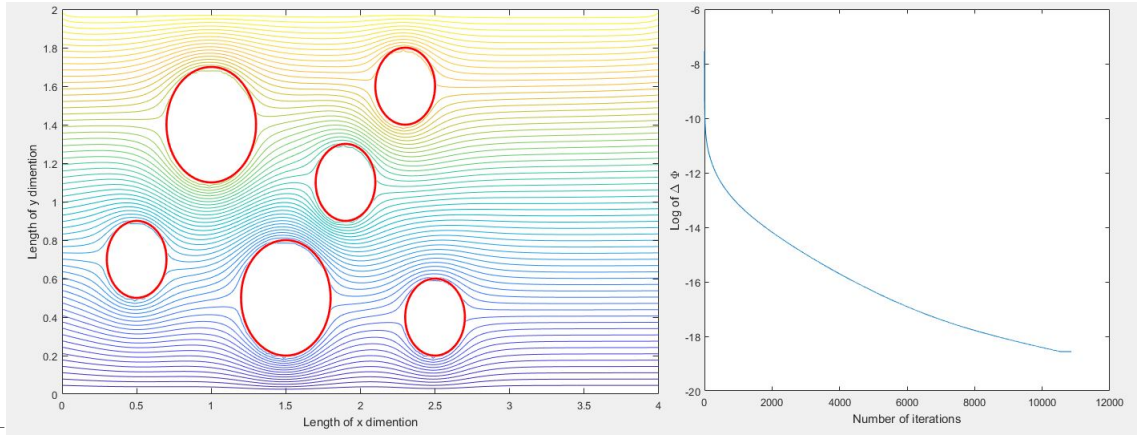


Figure 5: Graphical results of task B

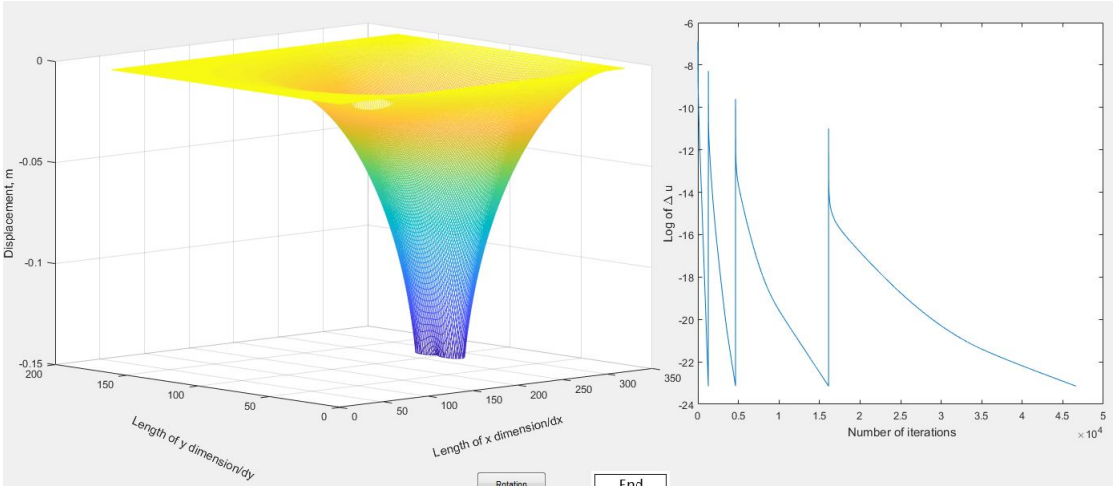
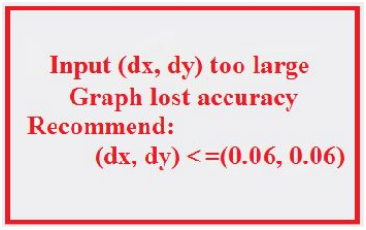


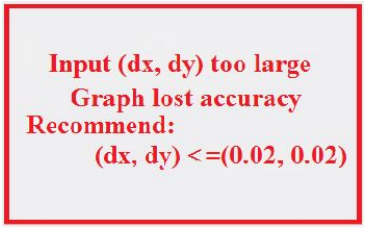
Figure 6: Graphical results of task C

3.2 GUI(User Friendliness)

After inserting all the values, bottoms can be pushed and plots will be displayed. Resolutions of each task are named 'dx' and 'dy' in the panels respectively. However, due to the nature of the code, the resolutions can not be too large, code will not run successfully with large input. Hence, a code is designed so that warning as shown in figure7a will displayed when there is large resolution input. Figure7a is the resolution warning for task A and task B, figure7b is the warning for task C. User is able to see the status of the program, there is an independent region for that. In figure4, the status will be displayed when code is finished and figure7c indicates that the code is still running.
To be able to see the characters such as circle or rip more clearly in task A and task C, user could 3D rotate the plots by pressing 'rotation' button shown in figure7c. Also, a timer is coded for the first task, the total time value will display as shown in figure7d when code is finished. In task A, if user insert a larger temperature value in first box as shown in figure7e, a warning will display in plotting area as shown in figure7f. These details could help run the code more efficiently and further improve the user friendliness.



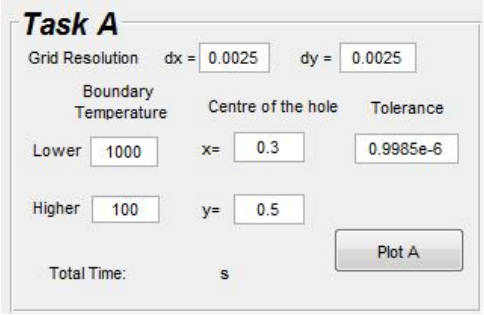
(a) Waring when resolution is too large



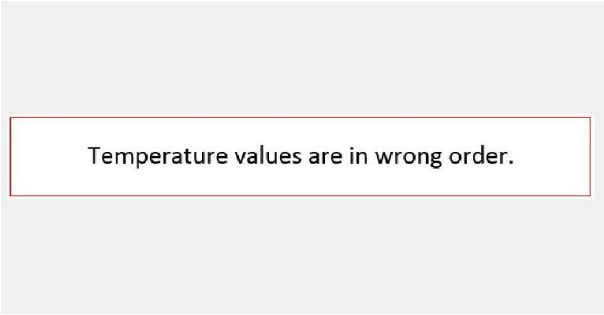
(b) Waring when resolution is too large

(c) Code running status

(d) Total time displayed in the panel



(e) Wrong input of temperature in task A



(f) Warning of incorrect temperature input

Figure 7: Detailed designs of GUI

4 Discussion

4.1 Convergence of Jacobi's method and SOR

As demonstrated above, there are two main methods applied for modeling. Their iteration times have been recorded in the table 1. Besides, it also includes iteration of Gauss-Seidel Method.

Methods	**Jacobi's Method**	**Gauss-Seidel Method**	**Successive Over-relaxation Method**
Iteration	186641	150254	7470

Table 2: Iteration times corresponding to different methods

This table illustrates that the number of cycles of Jacobi's Method for modeling is 186641 which is slightly more than that of Gauss-Seidel Method—-150254 times. It means that the rate of convergence of Jacobi's Method is slightly slower than that of Gauss-Seidel. With respect of SOR Method, it can enormously decrease the number of iterations compared with Jacobi's and Gauss-Seidel Method which is approximately half of theirs. This indicates that SOR rapidly reduces the running time since its rate of convergence is much more with the comparison of the other two methods. However, there is a shortcoming of SOR which slightly higher inaccuracy of graph. The consequential graphs of thermo-dynamics run by Jacobi's Method and fluid-dynamics run by SOR Method are shown in figure below.

Compared with the results of thermo-dynamics and fluid-dynamics above, there are some subtle differences. For example, the area around highly conducting liquid is more smooth and compact in figure 8. This indicates that the temperature values are more coherent in the area surrounding the circle in the graph. Furthermore, curve around cylinder in figure 9 is not as smooth as the first graph in figure 5. Therefore, modeling of SOR Method has much higher convergence than that of Jacobi's Method

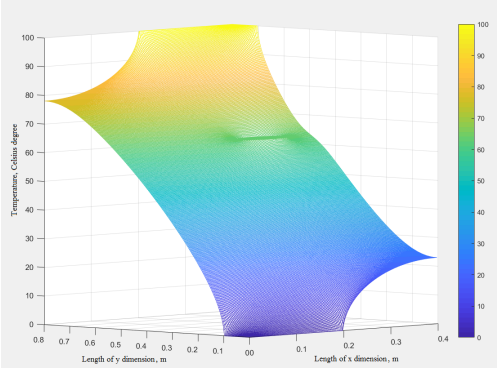


Figure 8: Jacobi's Method for thermo-dynamics

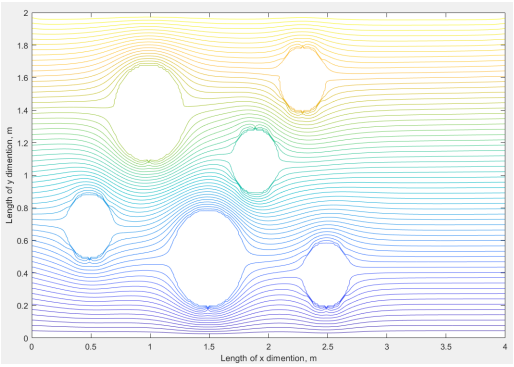


Figure 9: SOR Method for fluid-dynamics

although it is slightly poorer in graph quality.

4.2 Effects of relaxation factor

The relaxation factor scope is from 1 to 2 for over-relaxation. In order to decrease the time-consumption, the best value of relaxation factor was chosen by a number of attempts. The result is demonstrated in figure below.

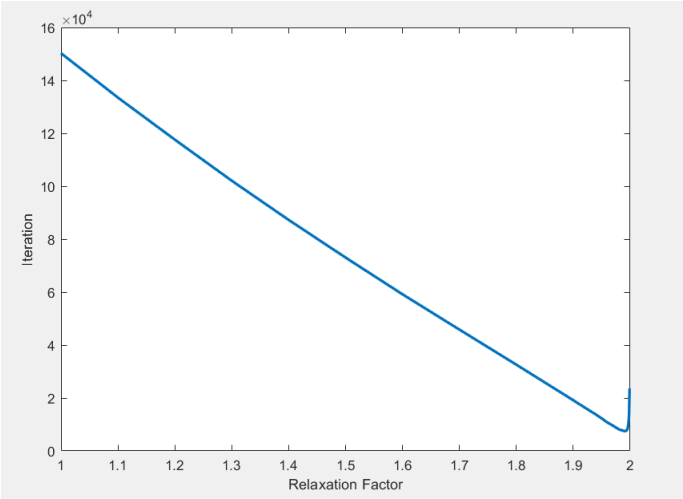


Figure 10: Relaxation factor vs Iteration

This curve illustrates that the number of iteration reaches the peak as relaxation factor is between 1 and 2. Then, iteration count is reduced until it reaches minimum point. The minimum iteration is 7470 as relaxation factor is 1.99. After that, the count is increased. Therefore, 1.995 is chosen to be relaxation factor value to balance accuracy and time-consumption. Furthermore, this graph can clearly illustrate the effects of relaxation factor.

4.3 GUI

When running the Gui, if the plot button of task A is pressed twice while keeping all inputs as constant, the grids of plot become much smaller and longer time period is taken. It turns out that the resolution in the program is become smaller after first running. At the second time, the program uses this small

resolution value to get a even smaller values. Hence, the value of resolution 'dx', for example, will be assigned to a variable before running the code. After code is finished, the value will be assigned back to the 'dx', which makes the resolution as same as original even if it is different during the program.

4.4 Tolerance of Thermo-dynamics

It is clear that tolerance of requirements is 1e-6. However, the original tolerance cannot be directly satisfied since there are ghost cells around boundary involved in calculation of error whereas those ghost cells are abandoned before meshing graph. Therefore, in order to satisfy requirement, the tolerance applied in program is reduced to 0.9985e-6 after multiple attempts for identifying tolerance magnitude.

5 Conclusions

In this project, three engineering scenarios are modeled by corresponding Matlab program. In order to optimize the problems, several solutions are adapted to reduce the time consumed. Successive Over-relaxation Method is applied to thermodynamics problem. Jacobi's Method is applied to fluid and structural dynamics scenarios. These methods reduce the 'loop' to a minimum level and all the calculations are carried out within the matrix. Hence, the total time is dramatically decreased. For example, in thermodynamics problem, the running time was 6 minutes at the beginning. Nevertheless, after modification, it only takes about 3 seconds. After coding these fundamental programs, a Gui is created based on them. User could easily change the parameter with the Gui, it provides almost all the graphical information that user needs. Also, it could provide appropriate warning information for user to prevent program from collapsing. These features improve user friendliness to maximum level.

References

[1] W. L. Briggs. *A multigrid tutorial*. Society for Industrial and Applied Mathematics, 1987.

[2] A. Grégoire and M. K. Sidi. *Numerical Linear Algebra*. Springer, 2008.

[3] T. Grétar(2010). Computational fluid dynamics i. [online]$https://www3.nd.edu/\ gtryggva/CFD-Course2010/2010-Lecture-11.pdf$ [Accessed 7 Dec. 2017].

[4] W. Kahan. *Gauss-Seidel methods of solving large systems of linear equations.* Toronto, Canada., 1st edition, 1958.

[5] V. Kalman(2014). Discrete poisson equation. [online]Available at: $https://sites.google.com/site/varga1kalman/teaching/physics-257-computational-physics/lectures/lecture-5/discrete-poisson-equation$[Accessed 7 Dec. 2017].

[6] J. Paul(2008). Iterative method for elliptic pdes. [online] Maths.manchester.ac.uk. Available at: $http://www.maths.manchester.ac.uk/\ pjohnson/CFD/Lectures/elliptic_iteration_handout.pdf$[Accessed 7 Dec. 2017].