McGill University

Computational Aerodynamics

MECH 539

Project 2

Name: *Student ID:*

Doug Shi-Dong 260466662

February 18, 2016

Question 1 & 2

The hand-written derivation is given on the last page. The truncation error of the second-order finite-difference discretization is $\mathcal{O}((\Delta x)^2, (\Delta y)^2\ )$ as expected. The modified equation in this case is simply the PDE on the LHS and the truncation error on the RHS. The truncation error only contains even terms, therefore the error is purely dissipative.

Question 3

Figure 1 shows the solution for a 400×400 grid.

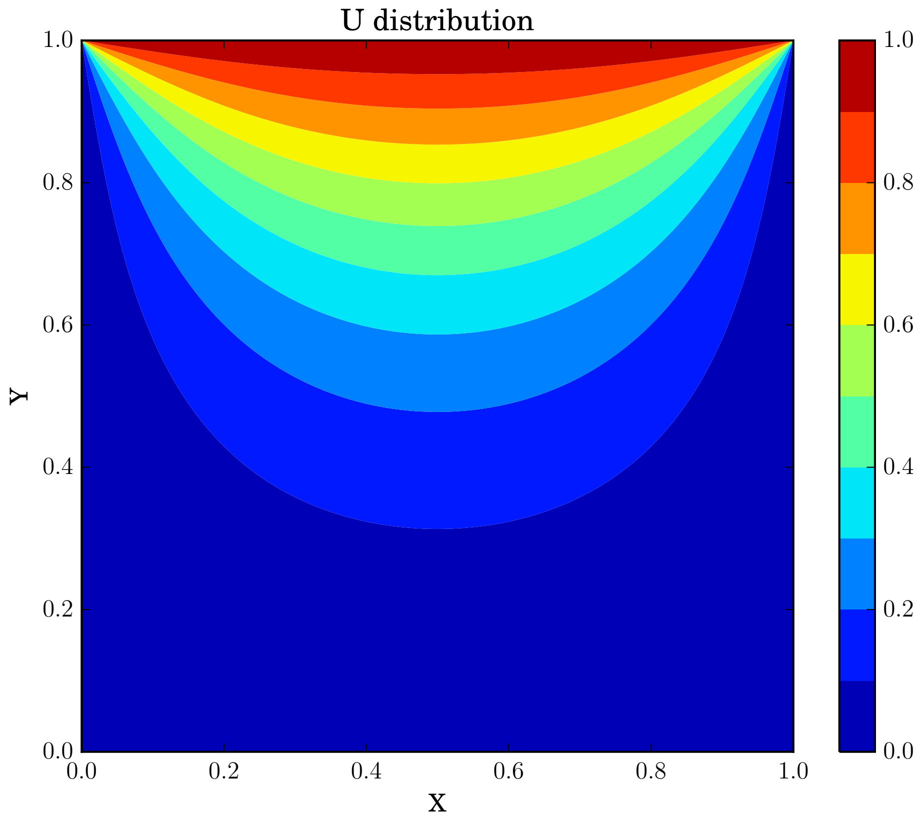

Figure 1: Numerical Solution $nx = 400$

Question 4

The convergence plots over the iterations are shown in Figure 2-4. The convergence criterion is the maximum change in the solution $max(\Delta u) < 1E\text{-}6$. The relaxation factor of the Successive Over-Relaxation (SOR) method is 1.5.

From the figures, it is obvious that Jacobi requires the most iterations, followed by Gauss-Seidel and then SOR. As the grid size increases, more iterations are required in order to converge to the same tolerance.

The Forsythe-Moler method is used to approximate the condition number of the A matrix since direct inversion would be too costly. Both the numerical solution of the Laplace equation and the linear system from the Forsythe-Moler are evaluated until $max(\Delta u) < 1E\text{-}15$ is reached. SOR is used to due its fast convergence. The condition number is compiled in Table 1. The condition number increases with grid size. It is consistent with the fact that bigger grid sizes require more iterations to converge.

Grid Size	100	200	400
Condition Number	1968	7905	29028

Table 1: Condition Number for Various Grid Sizes

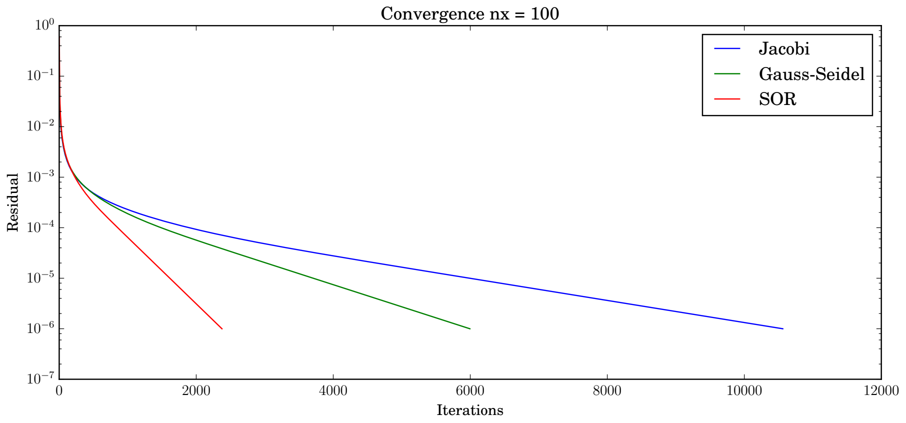

Figure 2: Convergence over Iterations $nx = 100$

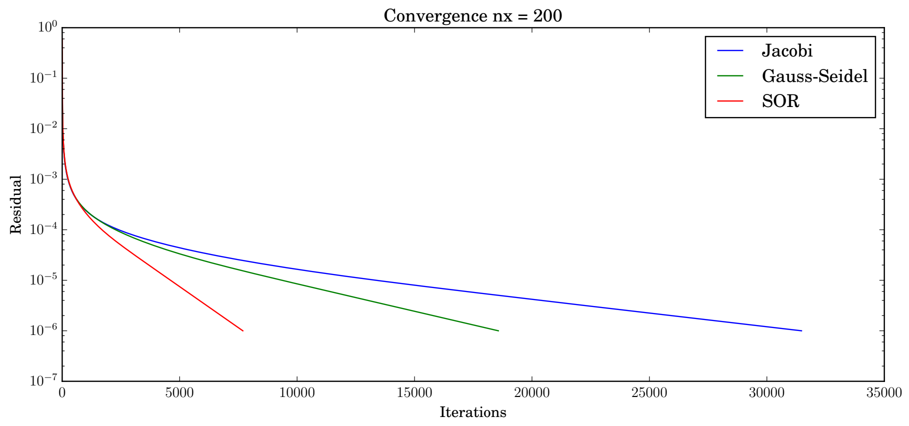

Figure 3: Convergence over Iterations $nx = 200$

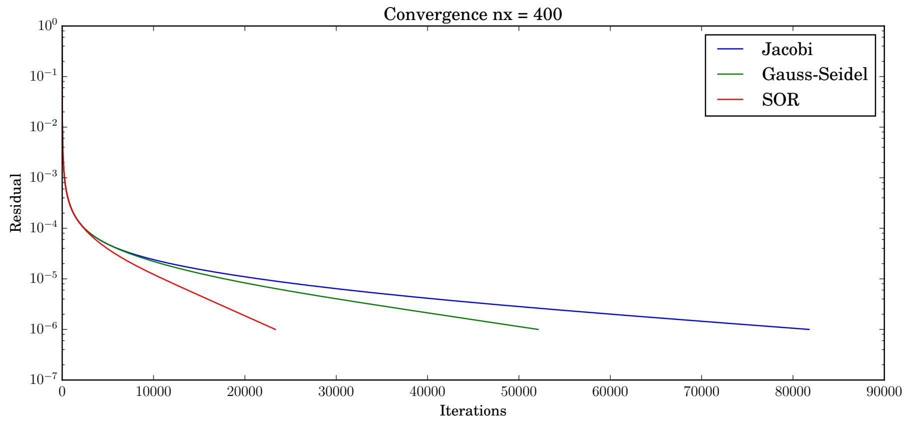

Figure 4: Convergence over Iterations $nx = 400$

Question 5

The convergence of the residual versus CPU time is shown in Figure 5-7. Note that the computation has been done in FORTRAN with optimization level -O3.

SOR still dominates the other two methods in terms of performance. However, the difference between Gauss-Seidel and Jacobi has reduced considerably. Although Jacobi takes almost twice as many iterations, the CPU time is close to Gauss-Seidel. Therefore, each iteration of Jacobi is less costly.

Table 2 compiles results to compare time and iterations for the different schemes for $nx = 400$ with the debugging flag (-g) and with optimization flag (-O3). Jacobi keeps the previous solution k in memory while it is computing $k+1$, therefore it requires twice as much memory as Gauss-Seidel or SOR. Although it is more costly in terms of memory, it allows for a few improvements in terms of speed. First, the dependence of the previous iteration allows the problem to be divided into parallel problems, allowing parallel computation. Second, the compiler is aware that the $k + 1$ step is only dependent on the k step, therefore, it is able to better optimize the process in terms of accessing memory.

It can be seen that in debug mode, the iteration of each method is approximately the same cost in terms of CPU time. However, Jacobi is much faster than the other two methods when -O3 is enabled due to the simplicity of memory access. SOR is the slowest per iteration since it requires more complicated memory accesses.

	Jacobi	Gauss-Seidel	SOR
Iterations	81787	52109	23340
Time (O3)	48.96	45.68	28.23
Time/Iterations (O3)	0.599	0.877	1.210
Time (g)	429.09	265.84	125.07
Time/Iterations(g)	5.24	5.10	5.36

Table 2: Time and Iterations for Different Methods $nx = 400$

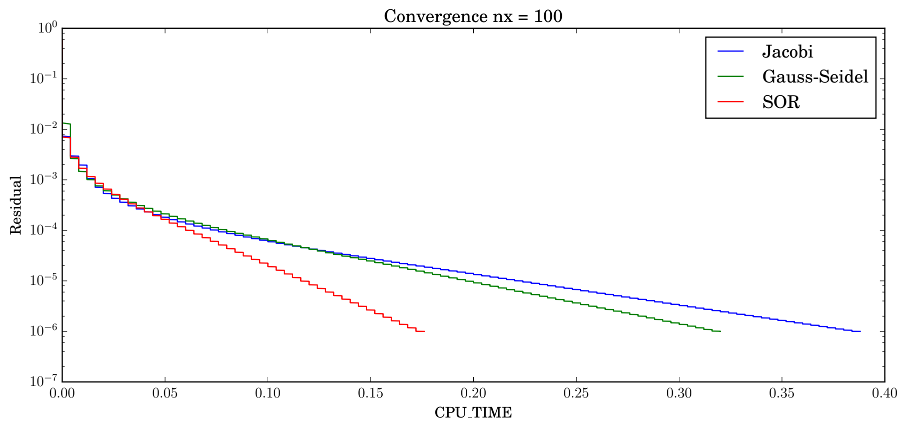

Figure 5: Convergence over Time $nx = 100$

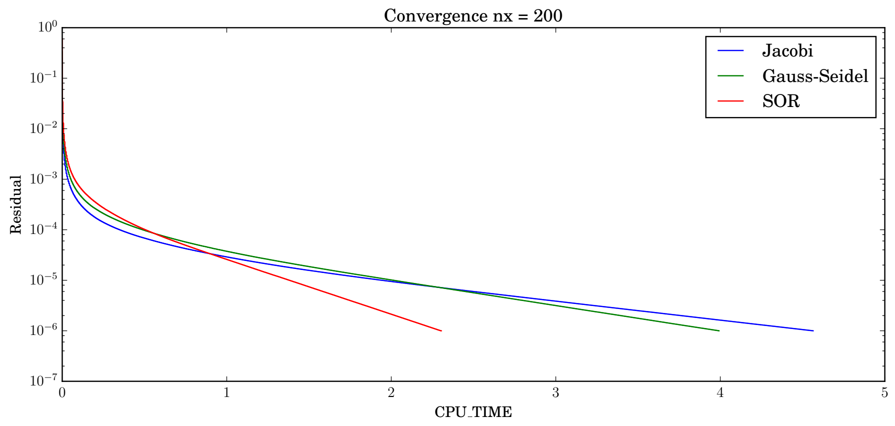

Figure 6: Convergence over Time $nx = 200$

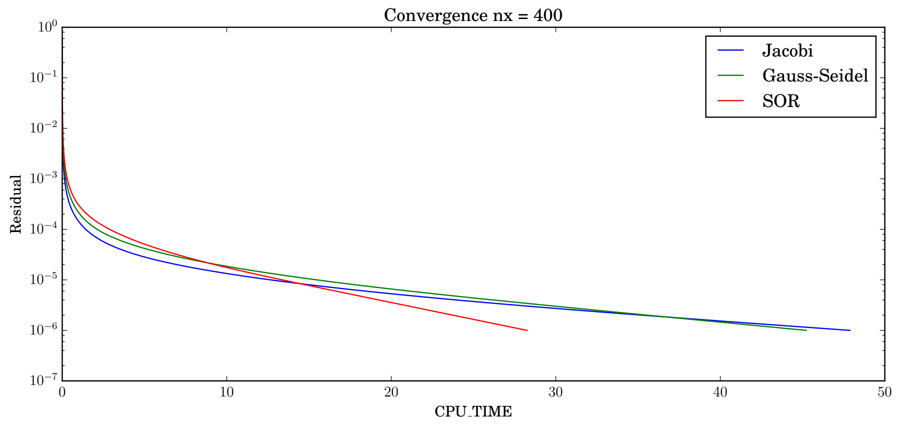

Figure 7: Convergence over Time $nx = 400$

Question 6

The exact solution is given by Equation 1. Numerically, 3600 terms have been used to evaluate the exact solution. However, the exact solution sum cannot resolve the straight line boundary accurately. Only 3 to 4 digits are recovered. Therefore, the exact solution itself contains some error near the boundaries.

$$\begin{aligned} U(x,y) &= \sum_{n=1}^{\infty} c_n sinh(n\pi y) sin(n\pi x) \\ c_n &= 2\int_0^1 sin(n\pi x)dx / sinh(n\pi) \end{aligned} \tag{1}$$

The absolute error of the numerical solution versus the approximated analytical solution is show in Figure 8. Along For this reason, the relative error is calculated using the L1-norm due to its robustness against outliers. Figure 9 shows the error versus the grid size. First, all iterative methods give the same error as it would be expected since the same linear system is being solved. Second, the logarithmic slope is nearly 2, meaning that the error is proportial to $(\Delta x)^2$, which corresponds to second-order accuracy.

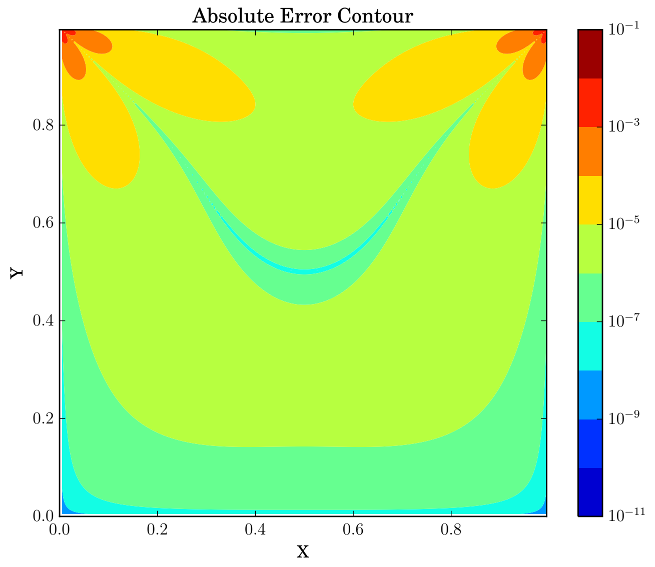

Figure 8: Contour Plot of Absolute Error

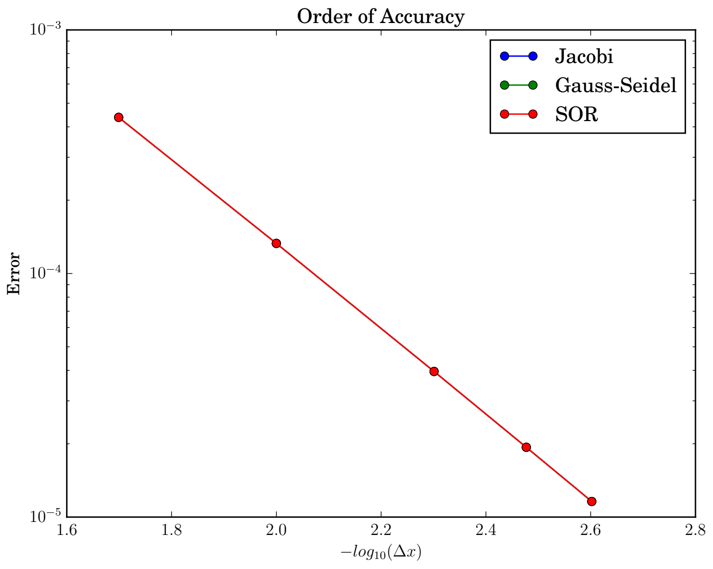

Figure 9: Order of Accuracy

Question 7

Figure 10-12 show the convergence of the SOR method for different relaxation parameters and grid sizes. When the relaxation parameter is less than 1, it takes more iterations than GS. When it is exactly 1, the exact GS is retrieved. For values above 1, increasing the relaxation parameter also increases convergence rates.. However, if the relaxation parameter is greater or equal to 2, the iterative method diverges. Even values near 2 such as 1.99 start being dangerously close to making the iterative solving unstable.

The optimal relaxation parameter will depend on how close it is allowed to be 2 without sacrificing stability. For all grid sizes, the optimal parameter will be between 1.90 and 1.99.

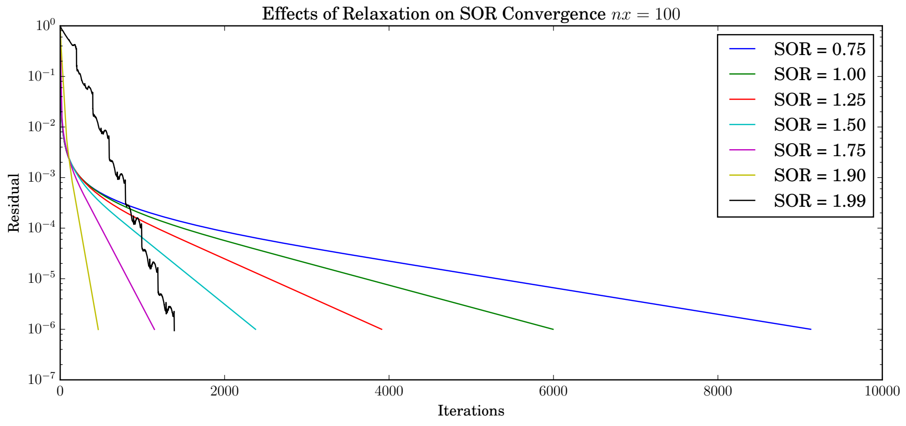

Figure 10: Relaxation Parameter Study $nx = 100$

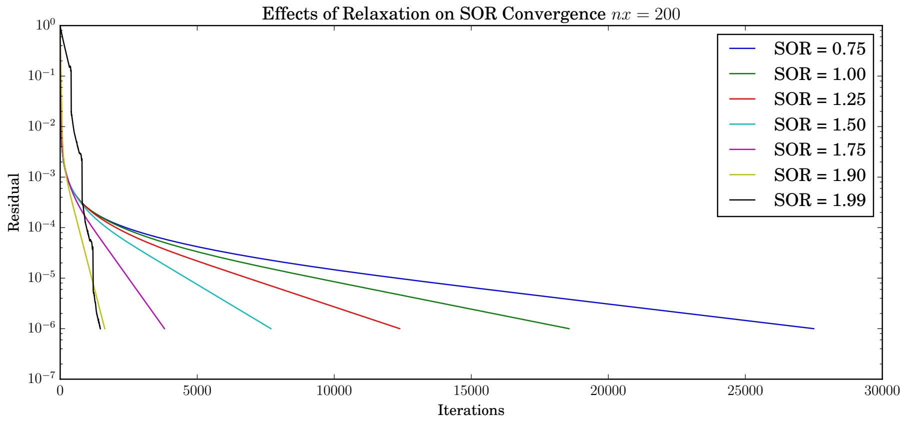

Figure 11: Relaxation Parameter Study $nx = 200$

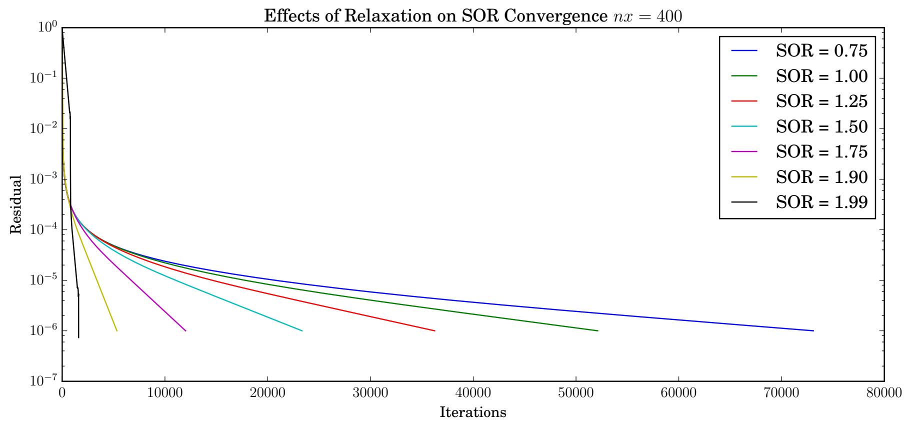

Figure 12: Relaxation Parameter Study $nx = 400$

Bonus

The bonus questions are appended at the end of the document.

Codes

Code has been written in FORTRAN. Default arithmetic operations are in double precision and optimization level -O3 unless specified otherwise.

All codes are available on my GitHub:

`https://github.com/dougshidong/mech539/tree/master/a2`

Truncation error 2nd order FD of Laplace Equation

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x,y) - 2u(x,y) + u(x-\Delta x,y)}{(\Delta x)^2}$$

$$u(x+\Delta x,y) = u(x,y) + \frac{\partial u}{\partial x}(x,y)\,\Delta x + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(x,y)(\Delta x)^2 + \frac{1}{6}\frac{\partial^3 u}{\partial x^3}(x,y)(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 u}{\partial x^4}(x,y)(\Delta x)^4 + \dots$$

$$u(x-\Delta x,y) = u(x,y) - \frac{\partial u}{\partial x}(x,y)\,\Delta x + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(x,y)(\Delta x)^2 - \frac{1}{6}\frac{\partial^3 u}{\partial x^3}(x,y)(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 u}{\partial x^4}(x,y)(\Delta x)^4 + \dots$$

$$u(x+\Delta x,y) + u(x-\Delta x,y) = 2u(x,y) + \frac{\partial^2 u}{\partial x^2}(x,y)(\Delta x)^2 + \frac{1}{24}\frac{\partial^4 u}{\partial x^4}(x,y)(\Delta x)^4 + \text{EVEN TERMS}$$

$$\frac{\partial^2 u}{\partial x^2}(x,y) = \frac{u(x+\Delta x,y) + u(x-\Delta x,y) - 2u(x,y) - \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(x,y)(\Delta x)^4 - \dots}{(\Delta x)^2}$$

$$TE = -\frac{2}{4!}\frac{\partial^4 u}{\partial x^4}(x,y)(\Delta x)^2 - \frac{2}{6!}\frac{\partial^6 u}{\partial x^6}(x,y)(\Delta x)^4 - \frac{2}{8!}\frac{\partial^8 u}{\partial x^8}(x,y)(\Delta x)^6 - \dots$$

$$\boxed{TE \text{ is } O[(\Delta x)^2, (\Delta y)^2]}$$

The modified equation is

$$\boxed{\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2 - \frac{1}{360}\frac{\partial^6 u}{\partial x^6}(\Delta x)^4 - \text{EVEN TERMS}}$$