MAE 6225 Project 1: Solving the 2-D Poisson Equation with Jacobi and Successive Over-Relaxation Methods

Luis Martinez

March 16, 2016

1 Introduction

This paper presents solutions to two test cases of the 2-D Poisson Equation using different iterative solvers. The iterative solvers studied are the Jacobi, Gauss-Sidel, and Successive Over-Relaxation (SOR) techniques. The Poisson equation was discretized using a second-order central finite difference scheme. The grid employed is a cell-centered node approach (see figure 1).

The test cases observed correspond to different source term functions. These are:

- Case 1: Homogeneous Dirichlet boundary conditions, and $f(x, y) = -8sin(2\pi nx)sin(2\pi ny)$
- Case 2: Homogeneous Neumann boundary conditions, and $f(x, y) = -8cos(2\pi nx)cos(2\pi ny)$

In these cases, n represents frequency. We examine solutions for different frequencies and how solution parameters need to be modified based on these frequencies. Section 2 provides details of the derivation of the numerical scheme; Section 3 contains a discussion of boundary conditions, Section 4 provides results for the 1^{st} case; and 2^{nd} case; and Section 5 summarizes how the SOR solution is affected by the relaxation parameter ω , frequency, number of grid points, and boundary conditions.

2 Discretization of the Poisson Equation and Iterative Methods

The 2-D Poisson equation is discretized using a second-order finite difference scheme consistent with the cell-centered grid system for computation. We begin the process of discretization by stating the 2-D equation, which looks like this:

$$f(x,y) = \frac{\partial^2\phi(x,y)}{\partial x^2} + \frac{\partial^2\phi(x,y)}{\partial y^2} \tag{1}$$

Now, using a second-order approximation, we can express the equation as such:

$$f_{i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} \tag{2}$$

The objective here is to find an expression for the unknown variable $\phi_{i,j}$, and with some algebraic manipulations we can find this expression.

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

$$\phi_{i,j} = \frac{\Big(\phi_{i+1,j} + \phi_{i-1,j}\Big)\Delta y^2 + \Big(\phi_{i,j+1} + \phi_{i,j-1}\Big)\Delta x^2}{2(\Delta x^2 + \Delta y^2)} - \frac{\Delta x^2\Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)} \tag{3}$$

Equation (3) above is the discretized form that we will use and modify to solve using the different iterative techniques. Let's observe how the different methods approach the solution.

2.1 Jacobi method

With the Jacobi method as with all other methods considered in this paper, we initialize the field with an initial guess. This initial guess is then used to initiate the iterative process and find the computed solution at the next iterative cylce. Then, equation (3) can be modified to obtain the solution at the next cycle:

$$\phi_{i,j}^{k+1} = \frac{\left(\phi_{i+1,j}^{k} + \phi_{i-1,j}^{k}\right)\Delta y^2 + \left(\phi_{i,j+1}^{k} + \phi_{i,j-1}^{k}\right)\Delta x^2}{2(\Delta x^2 + \Delta y^2)} - \frac{\Delta x^2 \Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)} \tag{4}$$

Note that in equation (4) the subscript k corresponds to the solution at the previous iteration cycle and $k + 1$ represents the solution at the current cycle.

2.2 Gauss-Sidel Method

The Gauss-Sidel (GS) method improves the Jacobi method by using updated values as soon as they are obtained. Figure 1 shows the sweep direction for the GS method used in this paper. Using the sweep direction shown in this figure, we can modify the Jacobi method, and the new solution is expressed as:

$$\phi_{i,j}^{k+1} = \frac{\left(\phi_{i+1,j}^{k} + \phi_{i-1,j}^{k+1}\right)\Delta y^2 + \left(\phi_{i,j+1}^{k} + \phi_{i,j-1}^{k+1}\right)\Delta x^2}{2(\Delta x^2 + \Delta y^2)} - \frac{\Delta x^2 \Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)} \tag{5}$$

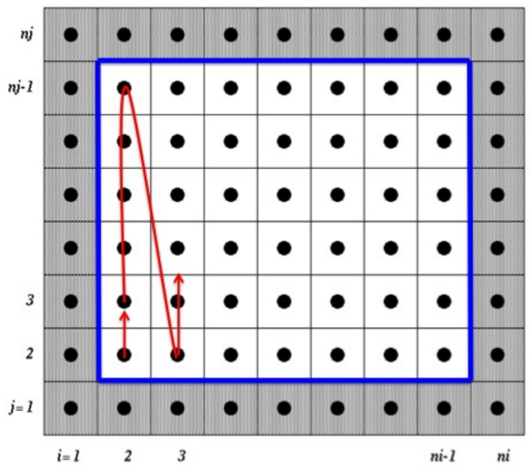


Figure 1: Direction (red arrow) of Gauss-Sidel sweep in the algorithm. The shaded region represents ghost cells, while the blue represents the boundary.

2.3 Successive Over-Relaxation (SOR) Method

Yet another approach to solve these cases is to use the SOR method. In this case, a relaxation parameter (ω) is introduced to improve on the guesses determined by the Gauss-Sidel method. The SOR method can be expressed as:

$$\phi_{i,j}^{k+1} = (1-\omega)\phi_{i,j}^{k} + \omega\frac{\left(\phi_{i+1,j}^{k} + \phi_{i-1,j}^{k+1}\right)\Delta y^2 + \left(\phi_{i,j+1}^{k} + \phi_{i,j-1}^{k+1}\right)\Delta x^2}{2(\Delta x^2 + \Delta y^2)} - \omega\frac{\Delta x^2 \Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)} \tag{6}$$

Note that equation (6) tells us that when $\omega = 1$, we return to the Gauss-Sidel method. When $\omega < 1$, the solution is "under-relaxed", and when $\omega > 1$, the solution is over-relaxed. As we will see below, we have to be careful when choosing these relaxation parameters.

3 Boundary Conditions

For case 1, the boundary conditions would be straightforward if we were using nodes on the boundary. However, because we have the cell-centered system, we need to make sure we derive the proper equations. Similarly, for case 2, we also need to derive the proper equations at the edge nodes. Let's look at case 1 first.

Using the lower edge nodes as shown by figure (1), we can derive the adequate Poisson equation at these nodes. Figure (2) shows the stencil used for the lower edge nodes.

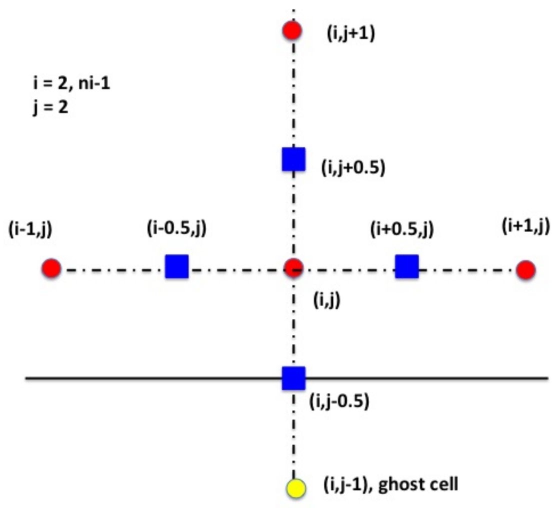


Figure 2: Nodes used to derive expressions for $\phi_{i,j}$ at the lower edge nodes. A similar process is performed for the other three boundaries and four corners.

3.1 Case 1 - Homogeneous Dirichlet Boundaries

Using figure (2), for case 1 we can express the second derivatives at these edge nodes as follows:

$$\begin{aligned}
\frac{\partial^2\phi(i,j)}{\partial y^2} &= \frac{1}{y_{i,j+0.5}-y_{i,j-0.5}}\left(\frac{\partial\phi_{i,j+0.5}}{\partial y}-\frac{\partial\phi_{i,j-0.5}}{\partial y}\right)\\
\frac{\partial^2\phi(i,j)}{\partial y^2} &= \frac{1}{y_{i,j+0.5}-y_{i,j-0.5}}\left(\frac{\phi_{i,j+1}-\phi_{i,j}}{y_{i,j+1}-y_{i,j}}-\frac{\phi_{i,j}-\phi_{i,j-0.5}}{y_{i,j}-y_{i,j-0.5}}\right)\\
\frac{\partial^2\phi(i,j)}{\partial y^2} &= \frac{1}{\Delta y}\left(\frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y}-\frac{\phi_{i,j}-\phi_{i,j-0.5}}{0.5\Delta y}\right)\\
\frac{\partial^2\phi(i,j)}{\partial y^2} &= \frac{\phi_{i,j+1}-3\phi_{i,j}+2\phi_{i,j-0.5}}{\Delta y^2}
\end{aligned}$$

We know that for case 1, $\phi_{i,j-0.5}$ is zero because of the specified homogeneous dirichlet conditions, so we can simplify our expression to:

$$\frac{\partial^2\phi(i,j)}{\partial y^2} = \frac{\phi_{i,j+1}-3\phi_{i,j}}{\Delta y^2}$$

For the x second derivative, we can simply use the neighboring cell-center nodes to obtain:

$$\frac{\partial^2\phi(i,j)}{\partial x^2} = \frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{\Delta y^2}$$

Then, combining the derivatives and solving for $\phi_{i,j}$:

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1})+\Delta y^2(\phi_{i+1,j}+\phi_{i-1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta x^2+2\Delta y^2} \quad i=2,ni-1, \quad j=2 \tag{7}$$

Repeating this process for all boundaries and corners, we can complete the equations at the boundaries for case 1:

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1})+\Delta y^2(\phi_{i+1,j}+\phi_{i-1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta x^2+2\Delta y^2} \quad i=2,ni-1, \quad j=nj-1 \tag{8}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1}+\phi_{i,j-1})+\Delta y^2(\phi_{i+1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+2\Delta x^2} \quad j=2,nj-1, \quad i=2 \tag{9}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1}+\phi_{i,j-1})+\Delta y^2(\phi_{i-1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+2\Delta x^2} \quad j=2,nj-1, \quad i=ni-1 \tag{10}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1})+\Delta y^2(\phi_{i+1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+3\Delta x^2} \quad i=2 \quad j=nj-1 \tag{11}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1})+\Delta y^2(\phi_{i+1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+3\Delta x^2} \quad i=2 \quad j=2 \tag{12}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1})+\Delta y^2(\phi_{i-1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+3\Delta x^2} \quad i=ni-1 \quad j=nj-1 \tag{13}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1})+\Delta y^2(\phi_{i-1,j})-\Delta x^2\Delta y^2 f_{i,j}}{3\Delta y^2+3\Delta x^2} \quad i=ni-1 \quad j=2 \tag{14}$$

Equations (7) through (14) are the complete set of equations for the boundary conditions at the edge cell-centered nodes. Now let's look at case 2.

3.2 Case 2 - Homogeneous Neumann Boundaries

Following a similar process as described for case 1, the objective is to approximate the second derivatives with values we can use in the flow-field. In this case, a useful approximation is to state that the edge nodes are approximately equal to the ghost cells due to the imposed boundary condition. Then, using the lower boundary again as an example, we can say that $\phi_{i,j} \approx \phi_{i,j-1}$. Then, for case 2 we can express the complete set of boundary equations:

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1}) + \Delta y^2(\phi_{i+1,j} + \phi_{i-1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{2\Delta y^2 + \Delta x^2} \quad i = 2, ni-1 \quad j = 2 \tag{15}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1}) + \Delta y^2(\phi_{i+1,j} + \phi_{i-1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{2\Delta y^2 + \Delta x^2} \quad i = 2, ni-1 \quad j = nj-1 \tag{16}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1} + \phi_{i,j-1}) + \Delta y^2(\phi_{i+1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + 2\Delta x^2} \quad i = 2 \quad j = 2, nj-1 \tag{17}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1} + \phi_{i,j-1}) + \Delta y^2(\phi_{i-1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + 2\Delta x^2} \quad i = ni-1 \quad j = 2, nj-1 \tag{18}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1}) + \Delta y^2(\phi_{i+1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + \Delta x^2} \quad i = 2 \quad j = 2 \tag{19}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1}) + \Delta y^2(\phi_{i+1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + \Delta x^2} \quad i = 2 \quad j = nj-1 \tag{20}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j-1}) + \Delta y^2(\phi_{i-1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + \Delta x^2} \quad i = ni-1 \quad j = nj-1 \tag{21}$$

$$\phi_{i,j} = \frac{\Delta x^2(\phi_{i,j+1}) + \Delta y^2(\phi_{i-1,j}) - \Delta x^2 \Delta y^2 f_{i,j}}{\Delta y^2 + \Delta x^2} \quad i = ni-1 \quad j = 2 \tag{22}$$

Equations (15) through (22) are the complete set of equations for the boundary conditions at the edge cell-centered nodes for case 2.

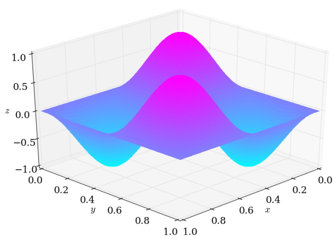
4 Test Cases

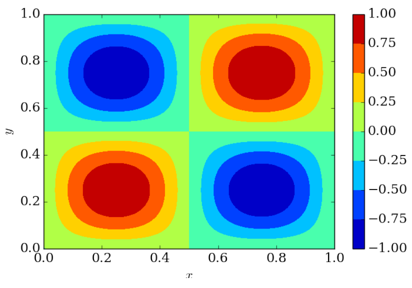
In this section we will loot at two different cases using different source terms and boundary conditions. We will perform iteration and grid convergence analysis for both cases.

4.1 Case 1: Source Term with Homogeneous Dirichlet Boundary Conditions

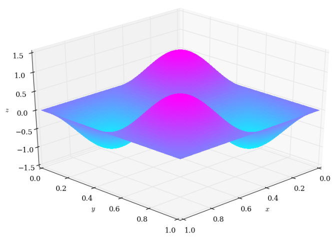
In this section we'll look at results obtained from the three different iterative methods, as well as convergence and grid refinements studies. We will also examine the effect of varying the source term frequency on our convergence and errors. To start, let's look at the simplest case: the solution for a frequency of $n = 1$. The solution is initiated by assigning random numbers from 0 to 1 in the entire flow field. This initial guess is then passed into the iterative solvers.

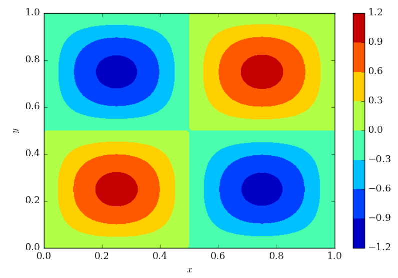
Figure (3) shows how the analytical solution looks compared to the computed solutions on a 3-D topographical representation. Figure (3) visually show that the general shapes of the analytical and computed solutions *appear* to be similar. If you look closely the computed solutions all appear to exceed analytical maximum and minimum (values of [1,-1]). The maximum and minimum values for the different methods did not vary significantly for our analysis, and all were within 99.9 percent of the analytical minimum and maximum.


(a)

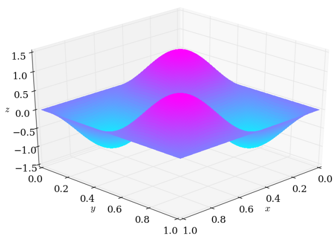


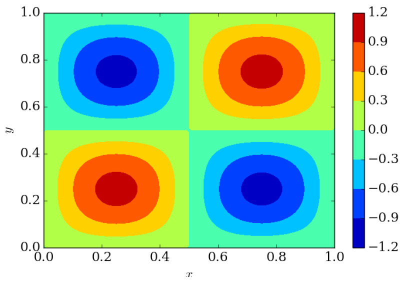
(b)


(c)

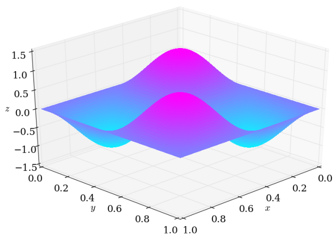


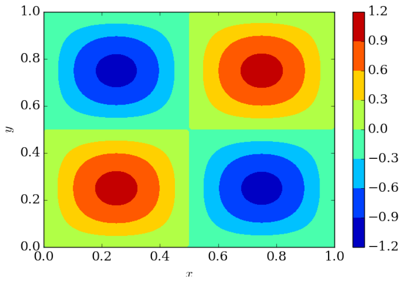
(d)


(e)



(f)


(g)



(h)

Figure 3: Visual comparison of analytical and computed solutions. (a-b) analytical solution, (b-c) Jacobi, (d-e) Gauss-Sidel, (f-g) SOR

Now, lets observe how these methods converged to the target error. Figure (4) shows the iterations required by each method to converge to the l2 norm of the error. We can clearly see the improvements made by the Gauss-Sidel method in comparison to the Jacobi method, and we can also see that the SOR ($\omega = 1.9$) required less than one-fourth of the iterations needed by the Gauss-Sidel Method. We'll discuss the choice of relaxation parameter below.

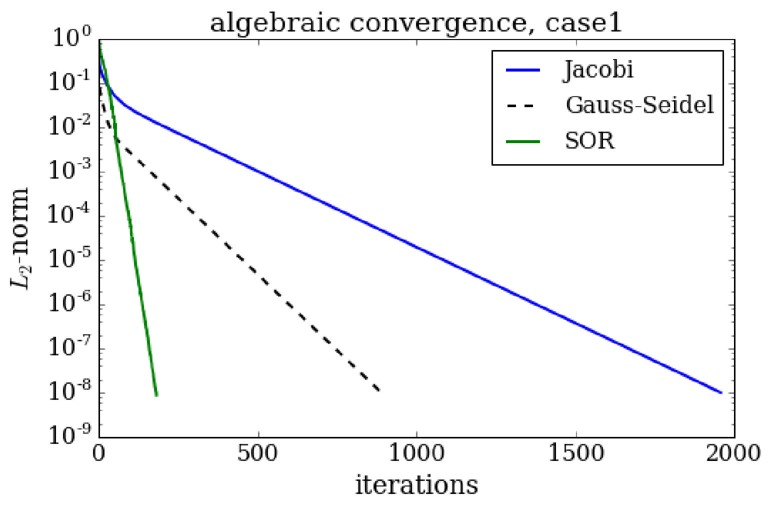


Figure 4: Convergence of the iterative methods for case 1 with a frequency of n=1.

Now we are poised to examine how these methods compared to the analytical solution by looking at the l2 norm of the error and by using different grid sizes. For this analysis, I chose an array of grid points with x and y grid points being the same. The array of grid points used for this analysis is described as:

$$n_x = n_y = [25, 50, 100, 150, 200]$$

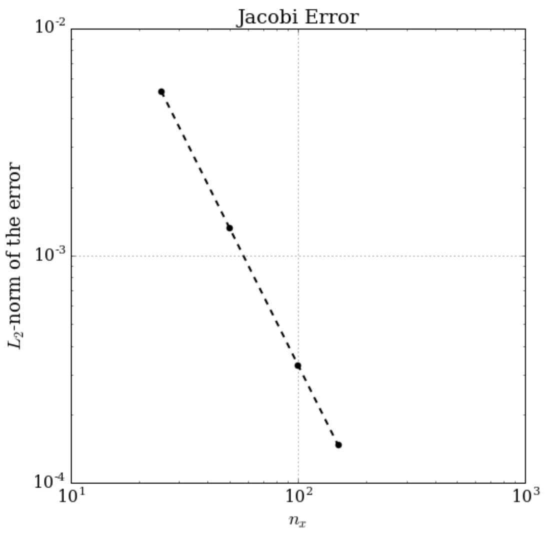
Figure (5) on the next page shows that the error looks like a second-order error, which is consistent with our discretization scheme. The error decreases as we refine the grid.

At this point, we still need to understand what happens to the computed solution with increasing frequency. Also, we have to take a closer look at the impact of different relaxation parameters for the SOR method.

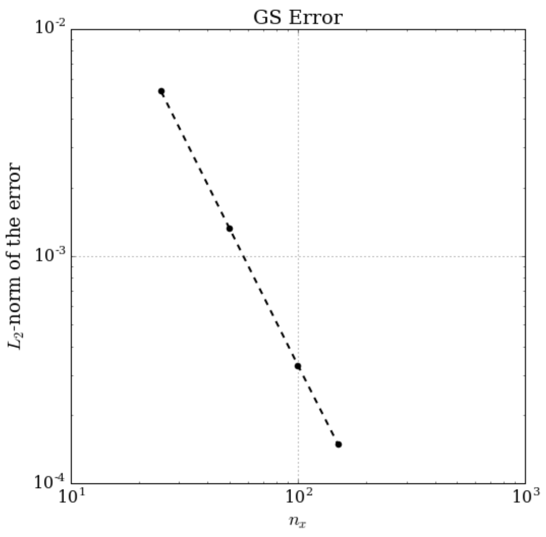
Building on the simple case with ($n = 1$), we can look at how the SOR method converges based on different relaxation parameters. We can perform a convergence test to our solution with the following relaxation parameters:

$$\omega = [1.2, 1.4, 1.6, 1.8, 1.9]$$

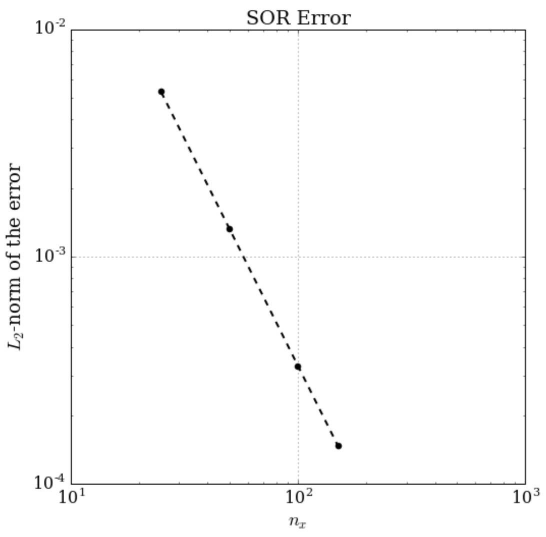
Figure (6) shows convergence rates by increasing omega . It is also interesting to note that the solution converges the fastest with $\omega = 1.8$.



(a)



(b)



(c)

Figure 5: Grid Refinement

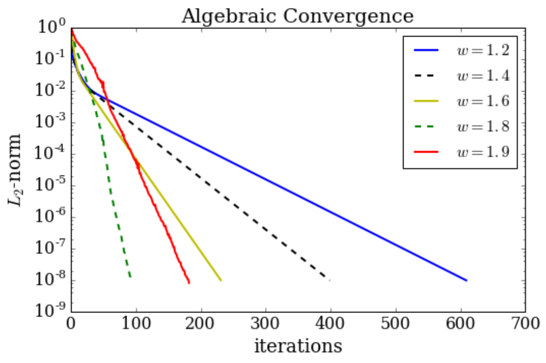


Figure 6: Convergence of the SOR method for case 1 with a frequency of n=1. (Grid is 25 by 25)

If we solve the same problem using a 100 by 100 grid instead of a 25 by 25 grid, we see that the solution requires more iterations, but more importantly, the overall error is reduced (from about 0.00528 to 0.00033). Additionally we see that $\omega = 1.9$ is more desirable in this case.

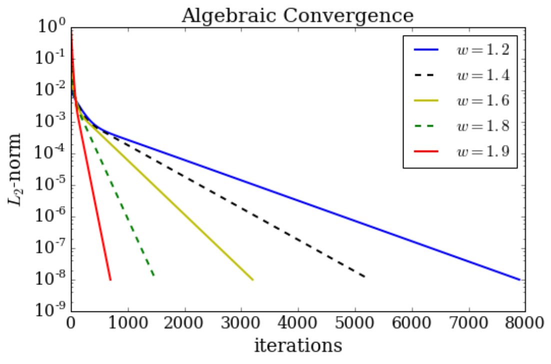


Figure 7: Convergence of the SOR method for case 1 with a frequency of n=1. (Grid is 100 by 100)

We should make sure that by improving the convergence cycles, we are not also increasing the error. Ideally we would want the error to remain the same if we only vary the relaxation parameter. In other words, for a given frequency, we want to increase convergence speed and also maintain accuracy. Continuing with our simple case of n=1, we see (Figure 8) that the error of the computed solution when compared to the analytical solution has not changed as the relaxation parameter is varied(remains at about error = 0.0053 on a25 by 25 node grid), which is good news. Our error is stable.

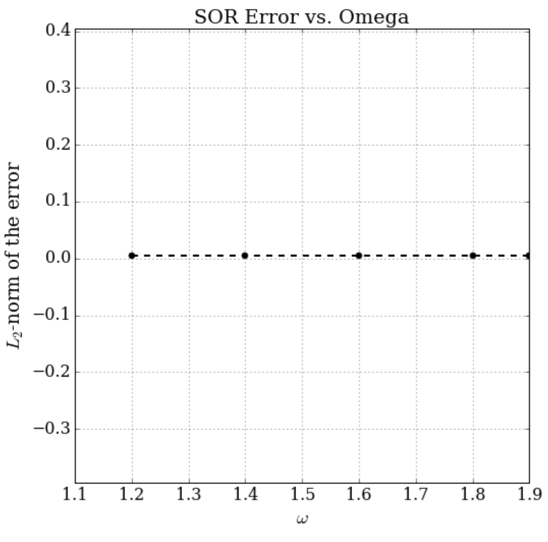


Figure 8: Analytical error as a function of relaxation parameter for the SOR method

Now, we can proceed to look into another outstanding issue:

What is the impact of frequency on the requirements for grid resolution and convergence of the methods?

To better understand, we can examine solutions for a range of frequencies. Generally, we can also choose a range of frequencies for each grid size by the following relationship:

$$1 \leq n_{frequency} \leq \frac{(min(n_x, n_y))}{2}$$

In order to keep the computational resources at a minimum, let's look at the lower end those frequencies. We will proceed with the following values:

$$f = [1.0, 2.0, 4.0, 8.0, 12.0]$$

Let's solve these frequencies on our 25 by 25 grid first. Figure (9) tells us that if we increase the frequency of the source term and solve on a single grid, the error of the solution (defined as the l2 norm of analytical vs computed solutions) then our error will also increase. This means that if we wish to solve the Poisson equation with source terms with high frequencies, we better make sure that we refine the grid to keep the error acceptably low. In the case shown in figure (9) the error went from 0.0053 at a frequency of 1, to an error of 1.28 at a frequency of 12 - that's a huge increase of more than two-hundred-fold!

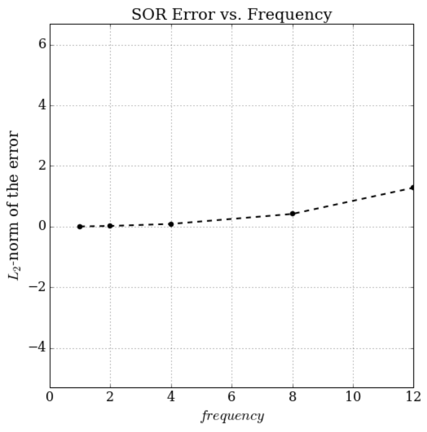


Figure 9: Error as a function of frequency

Now let's say we wanted to solve this case with a source term frequency of $n_f = 8.0$ and $n_f = 12.0$. The objective here is to refine the grid such that we keep the error down. Figure (10) shows that the grid was refined from a 25 by 25 grid, with the final grid being 200 by 200 nodes. The error was reduced from 0.42 to 0.0053.

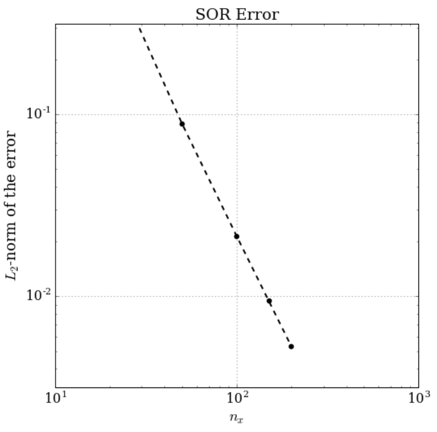


Figure 10: Error is reduced as the grid is refined for a frequency of 8.0

When we perform a similar analysis for a frequency of 12, we see that a grid of 25 by 25 nodes yields an l2 norm of the error of 1.28. When the grid is refined to 200 by 200 nodes, the error is reduced to 0.012. So, in order to reduce the error to levels comparable to the previous case, we must further refine the grid. Figure (11) shows the reduction in error due to grid refinement for a source term with a frequency of 12.

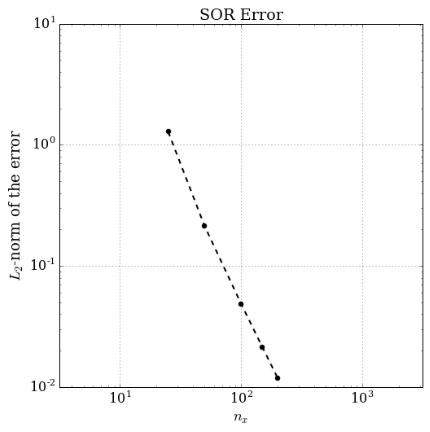


Figure 11: Error is reduced as the grid is refined for a frequency of 12.0

4.2 Case 2: Homogeneous Neumann Boundary Conditions

Similar to Case 1, we will analyze the computed solution by performing error analyses. We will, again, look at the simplest case: the solution for a frequency of $n = 1$. The number of grid points in x and y directions are both 25 nodes. The maximum and minimum values for the different methods are all within 99.9 percent of the analytical values. Additionally, figure (12) clearly shows the advantage of using the SOR method. It took the SOR method ($\omega = 1.9$) about 6 times less number of iterations to converge compared to the Jacobi method, and about 3 times less iterations than the Gauss-Sidel method.

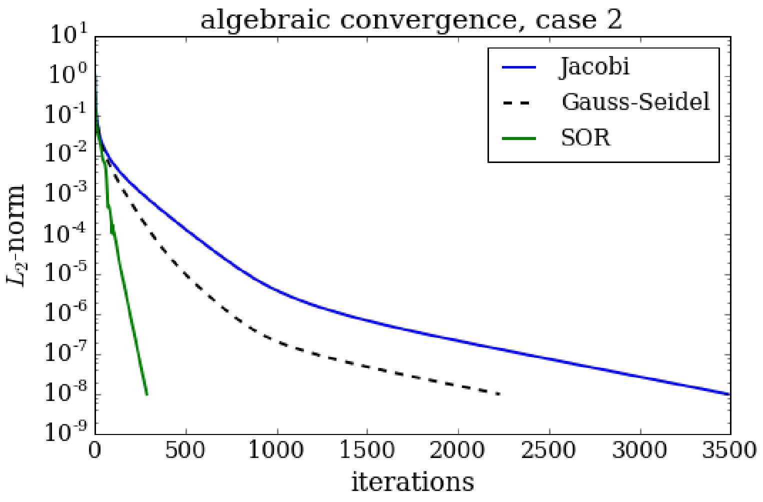
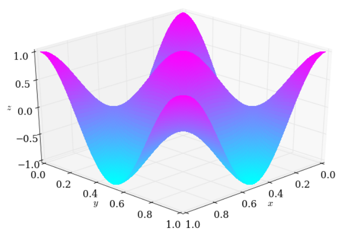
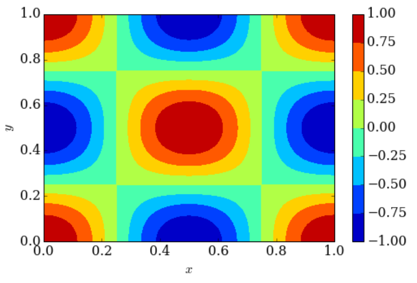


Figure 12: Convergence of the iterative methods for case 2 with a frequency of n=1.

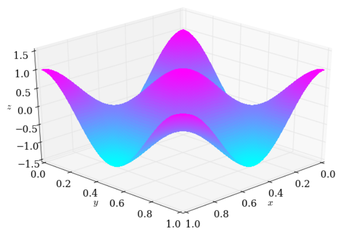
Figure (13) shows the 3- and 2-D graphs of the analytical and computed function.



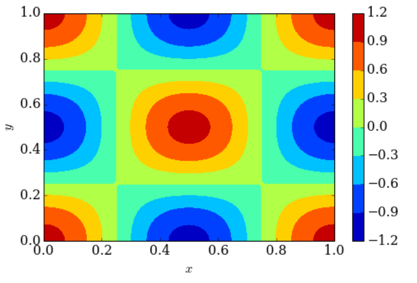
(a)



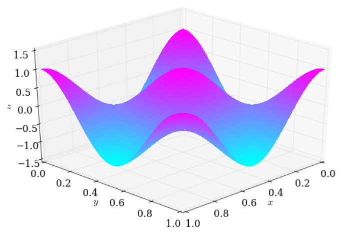
(b)



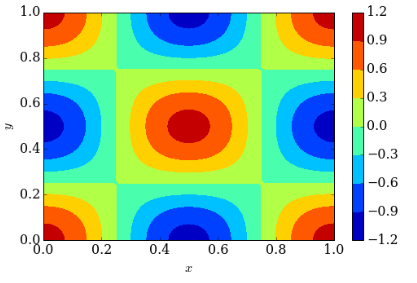
(c)



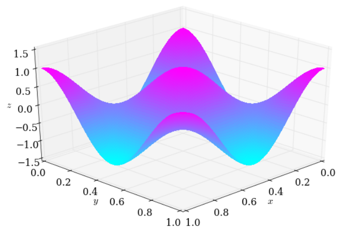
(d)



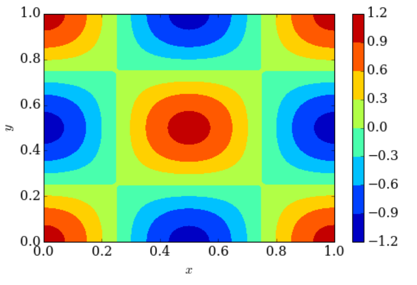
(e)



(f)



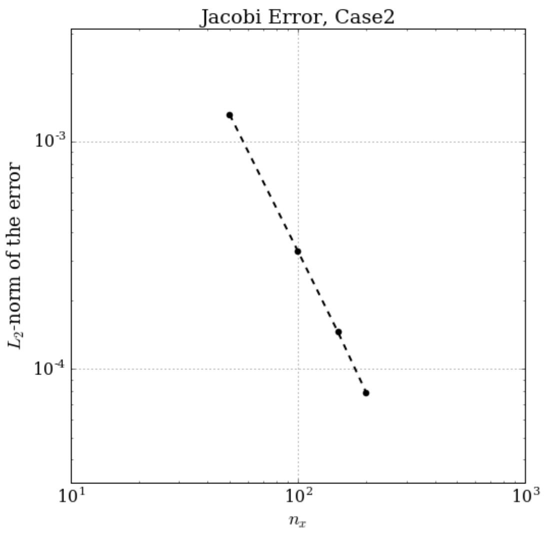
(g)



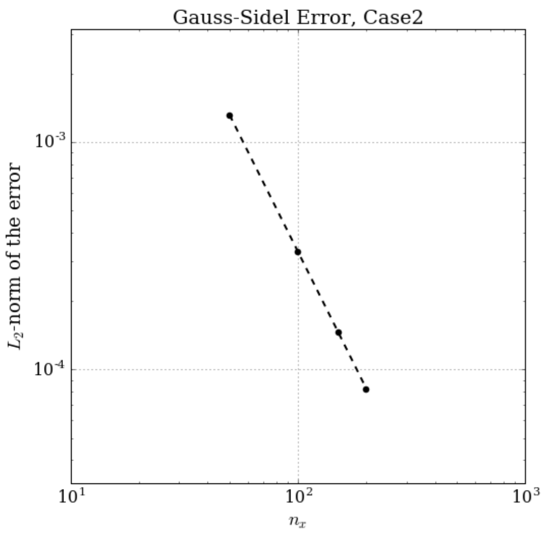
(h)

Figure 13: Visual comparison of analytical and computed solutions. (a-b) analytical solution, (b-c) Jacobi, (d-e) Gauss-Sidel, (f-g) SOR

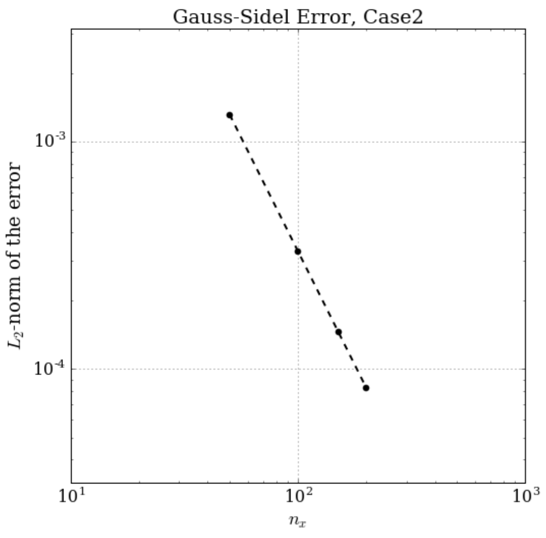
Now let's perform a grid refinement on the solution using the same array of nodes as part 1. Figure (14) shows that the error is reduced as nodes are increased and the rate of error decrease is consistent with our second-order discretization scheme.



(a)



(b)



(c)

Figure 14: Grid Refinement

Now let's look at how the SOR method converged using different relaxation parameters. Figure (15) shows that for the case of the 25 by 25 grid with a frequency of one, a relaxation parameter of 1.8 was optimal.

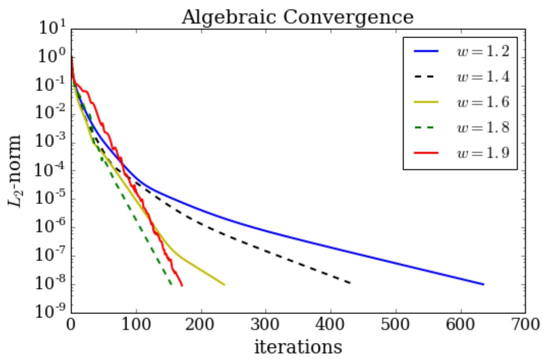


Figure 15: Convergence of the SOR method for case 2 with a frequency of n=1. (Grid is 25 by 25)

Similarly to case 1, if we solve the same problem using a 100 by 100 grid instead of a 25 by 25 grid, we see that the solution requires more iterations, but the overall error is reduced (from about 0.00528 to 0.00033). We also see that $\omega = 1.9$ is more desirable in this case.

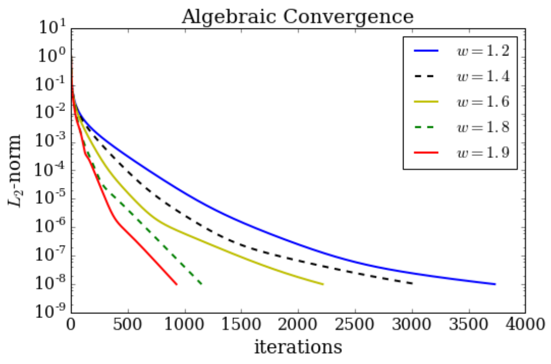


Figure 16: Convergence of the SOR method for case 1 with a frequency of n=1. (Grid is 100 by 100)

We can also verify that the error remains constant given that omega is the only parameter allowed to change. This means that given that we keep our error low, there is substantial benefit in over-relaxation.

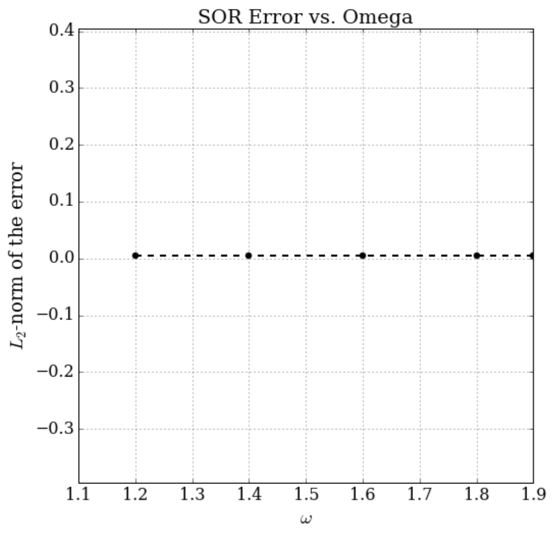


Figure 17: Analytical error as a function of relaxation parameter for the SOR method

Similar to part 1, we solve for a range of frequencies on the same grid to examine how the error changes. Figure (18) shows that the error increases with higher frequencies. Again, we can say that if we wish to solve the Poisson equation for source terms with higher frequencies, we need to make sure that we employ a fine-enough mesh so that the error is reduced. In this chart, for a 25 by 25 grid, the error goes from 0.0053 (frequency=1) to 1.28 (frequency=12).

By performing grid refinement fo the solutions of the Poisson equation with source terms that contain frequencies of 8 and 12, we see that the error decreases and the grid is refined. For the solution with a frequency of 8, we see that the error goes from 0.418 on a 25 by 25 grid to 0.00528 on a 200 by 200 grid. For a frequency of 12, we see that the error goes from 1.28 to 0.012 using the same grids. Again, this suggests that solutions with higher frequencies require more nodes to keep the error at the same level.

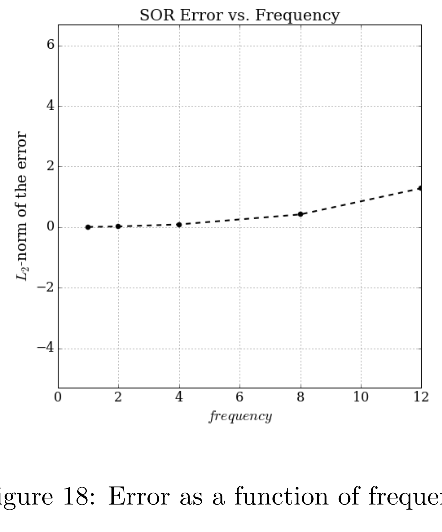


Figure 18: Error as a function of frequency

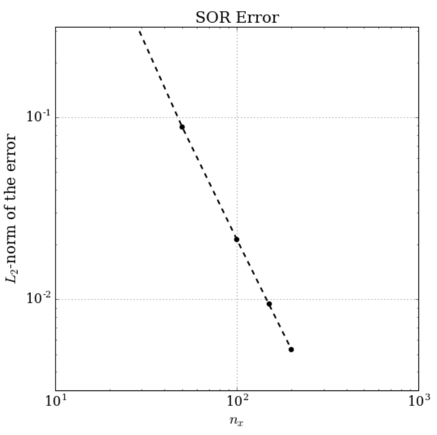


Figure 19: Error is reduced as the grid is refined for a frequency of 8.0

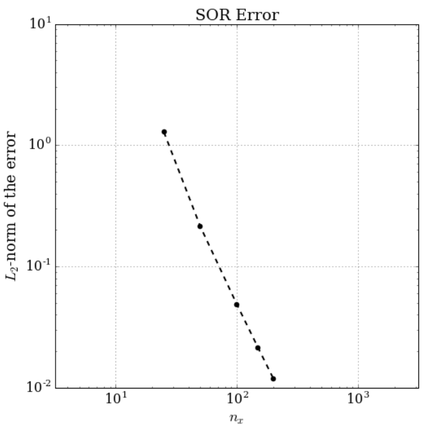


Figure 20: Error is reduced as the grid is refined for a frequency of 12.0

5 Conclusions

When we started this analysis, we wanted to be able to get a sense of how the relaxation parameter is affected by the frequency, grid size, and boundary conditions. After developing the code and looking at some results on the lower end of frequencies (in order to keep the grid size small due to computational restrictions), one of the most important aspects about developing an accurate solution is to make sure that the computed results closely resemble the analytical results. It is often the case that analytical results are not available, so experimental results can be helpful in validating the CFD models. If analytical solutions are available then ideally we are looking to keep the error as small as possible. It is in the context of maintaining a standard of accuracy that it is most productive to talk about what the optimal relaxation parameter is. Because, after all, if the error is large, it does not matter what relaxation parameter converges the fastest.

Looking at figures 6 and 7 for case 1, and figures 15 and 16 for case 2, I did notice that as we reduce the error, we increase the grid size, and it is in this range of very low errors that it is most optimal to use higher relaxation parameters. We don't necessarily want to increase the computational expense if not necessary, but if the errors are large enough it will be necessary to increase the grid. The results here show that an SOR method with a high relaxation parameter sort of "compensates" for the extra computations required to keep the error low.

Another issue that I have not discussed at all, is the sensitivity of these iterative methods to initial conditions. For example, for case 1, I initiated the computational domain with random numbers from zero to one. For case 2, initializing the field in this way often led to very long computational times for convergence, or even to higher errors than expected. So for case 2, I had to initialize the flow field with all zero values, and this allowed the solver to converge faster with expected results.

Finally, I have also derived and coded the model with a placement of nodes on cell boundaries (as opposed to cell centers) and the general trends discussed in this paper also hold for that model.