Jared Jones 1076497

Methods for Solving the 2-D Helmholtz Equation

Abstract

There are many numerical methods for approximating and solving partial differential equations. This report covers the use of the Gauss Seidel and Successive Over Relaxation (SOR) methods to solve a 2-D Poisson/Helmholtz equation. The Gauss Seidel method is an iterative method that approximates the solution by solving the next iteration's values using the previous solutions. Successive Over Relaxation is similar to Gauss Seidel except that it uses a correction factor to converge faster.

Computer Specifications:

Processor: Intel(R) Core(TM) i7-3770S CPU @ 3.10 GHz

RAM: 8.00 GB DDR3 1333/1600

Processor Graphics ‡ Intel® HD Graphics 4000

Cache: 8 MB Smart Cache

# of Cores 4

# of Threads 8

System Type: 64 Bit OS

Jared Jones 1076497

The problem is a 2-D Helmholtz PDE defined by,

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

Where,

$$F(x, y) = \sin\left(\pi \frac{x - ax}{bx - ax}\right) \cos\left[\frac{\pi}{2}\left(2\frac{y - ay}{by - ay} + 1\right)\right]$$

and the domain ranges from ax = ay = - π, bx = by = π, and λ = 0. The given boundary conditions are of both the Dirichlet and Neuman type,

$$u(x, y = by) = \left(bx - x(k)\right)^2 * \cos\left(\pi * \frac{x(k)}{bx}\right)$$

$$u(x, y = ay) = x(k) * \left(bx - x(k)\right)^2$$

$$u(x = ax, y) = (bx - ax)^2 * \cos\left(\pi * \frac{ax}{bx}\right) + \left(\frac{y(k) - ay}{by - ay}\right) * \left(ax * (bx - ax)^2 - (bx - ax)^2 * \cos\left(\pi * \frac{ax}{bx}\right)\right)$$

$$\frac{\partial u}{\partial x}\Big|_{x=bx} = 0$$

Discretizing the second order derivatives of the project equation, we get two similar equations.

$$\frac{\partial^2 u}{\partial x^2} = \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{\Delta x^2}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{U_{i,j-1} - 2U_{ij} + U_{i,j+1}}{\Delta y^2}$$

These equations are approximations of the second order partial derivative with second order error in x and y.

The Gauss Seidel method is an iterative process that approximates the solution by solving the next iteration's values using the previous solutions. In Matlab, the pseudo code is as follows,

Jared Jones 1076497

```
for jj = 2:N+1
    for ii = 2:N+1
        U(ii,jj) = (1/(4-(lambda*h^2))*(U(ii-1,jj) + U(ii+1,jj)
+ U(ii,jj-1) + U(ii,jj+1) - (h^2*F(ii,jj))));
    end
end
```

In this method, all of the values are updated before starting the next iteration. The function uses the U previously defined with the initial boundary conditions and, as it iterates, grabs the values solved from previous iterations. This allows the solution to converge faster than using the Jacobi Method and is the most basic iterative methods to use for solving linear systems. Using this method, the two dimensional Helmholtz equation reaches convergence to a tolerance of 10^-6 within the set tolerance limit. As the number of nodes, N, increased, the solution converged more and more to the exact solution. The figures below represent how the solutions converged with different values of N.

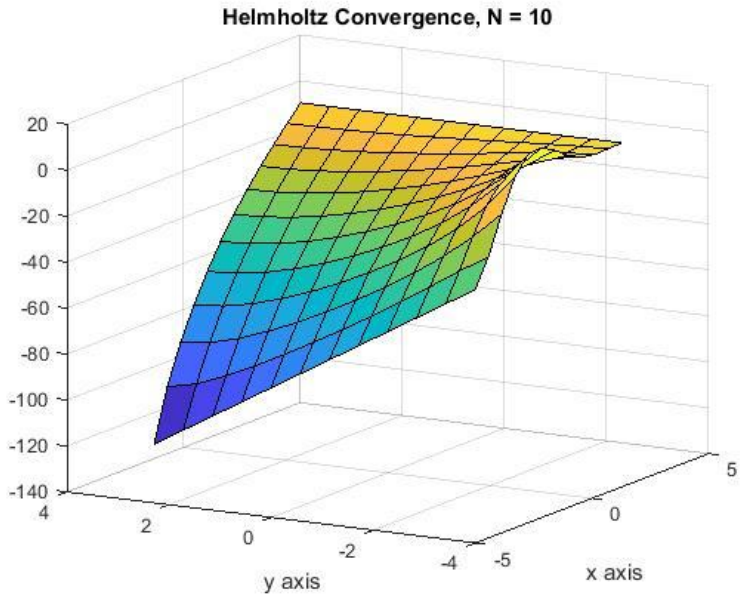

Figure 1: Gauss Seidel N = 10

In Figure 1, convergence was achieved within 360 iterations and 0.011 seconds.

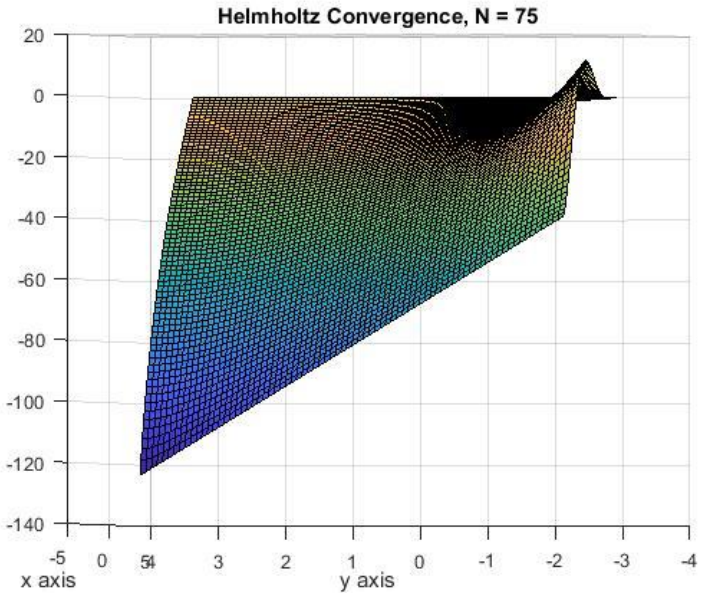

Figure 2: Gauss Seidel N = 75

Figure 2 shows convergence achieved within 16857 iterations and 4.117 seconds.

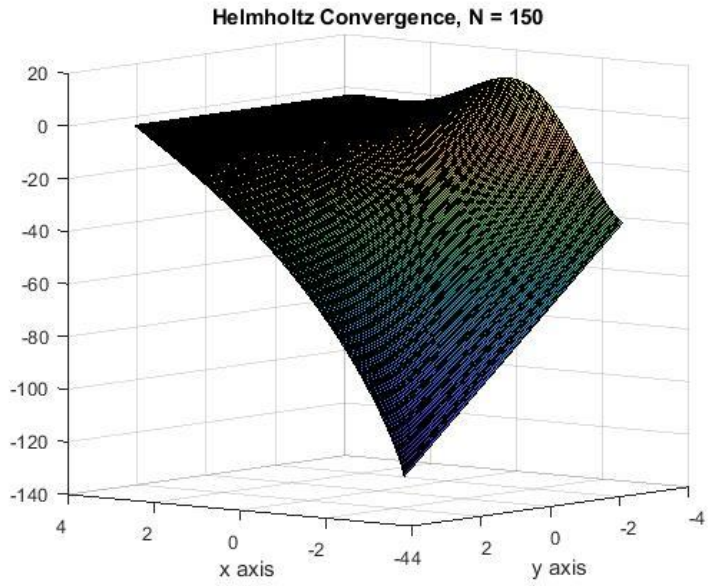

Figure 3: Gauss Seidel N = 150

Figure 3 shows convergence achieved within 70720 iterations and 66.122 seconds

While the Gauss Seidel is faster than the Jacobi Method, it is not as quick to solve linear systems as the Successive Over Relaxation Method. The principle of this method is much of the

same as for Gauss Seidel, except that it uses at correction factor to help it converge faster, assuming that each correction has the same sign.

```
for jj = 2:N+1
    for ii = 2:N+1
        U(ii,jj) = w*(1/(4-(lambda*h^2))*(U(ii-1,jj) +
U(ii+1,jj) + U(ii,jj-1) + U(ii,jj+1) - (h^2*F(ii,jj))));
    end
end
```

The "w" is the correction factor and in most cases ranges between 1< w <2. Where w is 1, it is evident that this is the same as the Gauss Seidel Method as shown above. In this project, for a tolerance of 1*10^-6, it was found that the most efficient correction factor for converging to the solution is about 1.7. Any more and the code never reaches tolerance before the defined maximum iterations which scales with N^2 by a factor of 10. The following figures show the SOR convergence at different N.

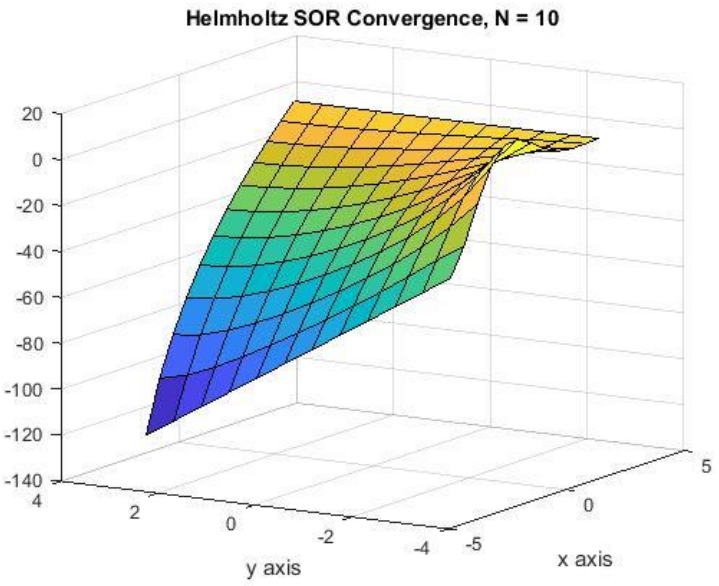

Figure 4:SOR convergence N=10

Figure 4 reaches convergence within 180 iterations and .008 seconds. Comparing this to Figure 1, it is clear that the SOR is superior and more efficient with a faster convergence.

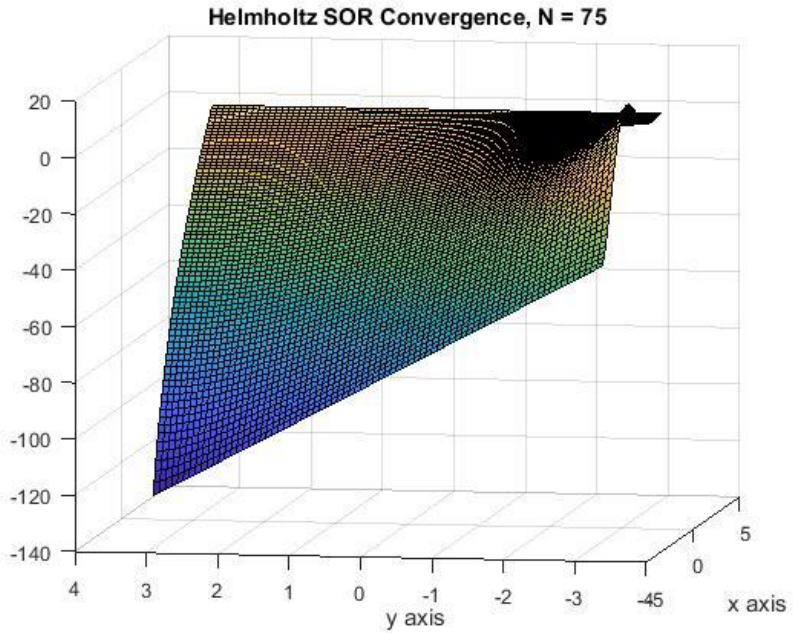

Figure 5: SOR Convergence N=75

Figure 5 shows that convergence was achieved in 6820 iterations and 1.635 seconds. This also shows a faster convergence than Figure 2.

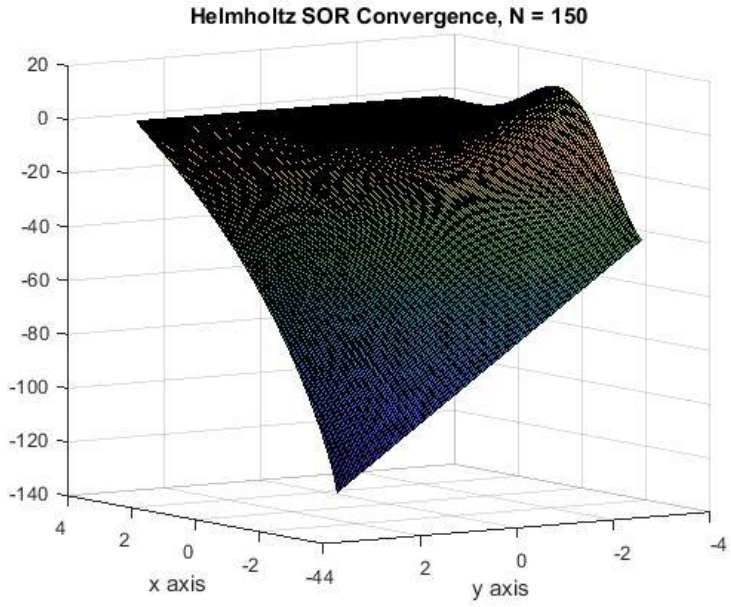

Figure 6: SOR convergence N = 150

Figure 6 shows that convergence was reached at the max iterations of 10*N^2 and 209.467 seconds. This is interesting because it doesn't support the concept that SOR converges faster and within fewer iterations. This is likely due to grid convergence where the solution begins to take longer outside of a certain range of N.

The SOR method begins to 'blow up' at a correction factor of 1.78 causing an oscillatory effect along the boundary and worsens as w approaches 2.

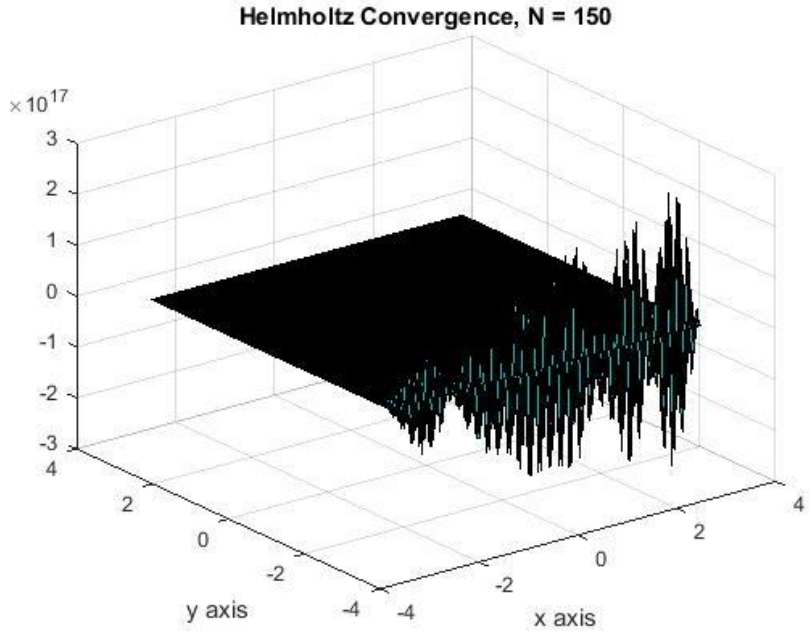

Figure 7: SOR Convergence for w = 1.78, N = 150

In conclusion, iterative methods are useful for approximating Helmholtz/Poisson elliptical equations provided that the size of N isn't too small or too large. Successive Over Relaxation converges faster than Gauss Seidel. As N increases and the mesh becomes more fine, so too does the time and number of iterations required for convergence. We can see this is true as the graph remains the same as N increases. However, it is important to be careful of 'blowing up' the solution while iterating SOR at certain correction factors.