**Poisson’s Equation Project Report**

Project AP02-3

Chris Hixon

May 5th

MECE 5397 – Scientific Computing for Mechanical Engineers– Spring 2019

Dr. Andrea Prosperetti & Dr. Amit Amritkar

**Abstract**

For the semester project I chose to implement Poisson’s equation in two dimensions for a rectangle. This equation is a partial differential equation that is used for finding electric potentials or gravitational fields. This equation is also commonly used by mechanical engineers and physicists. This project was to create a MATLAB code that solves Poisson’s equation using different numerical methods to create an approximate solution. The project also required the use of source control, checkpoints, and visualization to represent the solution. The version of Poisson equation that this project is solving is for the diffusion in a rectangle. This rectangle consists of four boundary conditions that surround the entire rectangle and consist of mixed conditions. The boundary conditions govern the solution of the problem and are composed of three Dirichlet boundary conditions and one Neuman boundary condition. The Neuman condition occurs at the top of the rectangle when y was at the maximum distance 2π and required a ghost node to solve for the values at the boundary. The boundaries at surrounding the left, bottom, and right side of the rectangle are Dirichlet boundaries and were governed by functions that changed as your position changed along the boundary changes. The equation was discretized and solved by using two different numerical methods, Gauss-Seidel and Successive Over Relaxation. These two methods allowed for the grid to converge to an approximate solution for the partial differential equation.

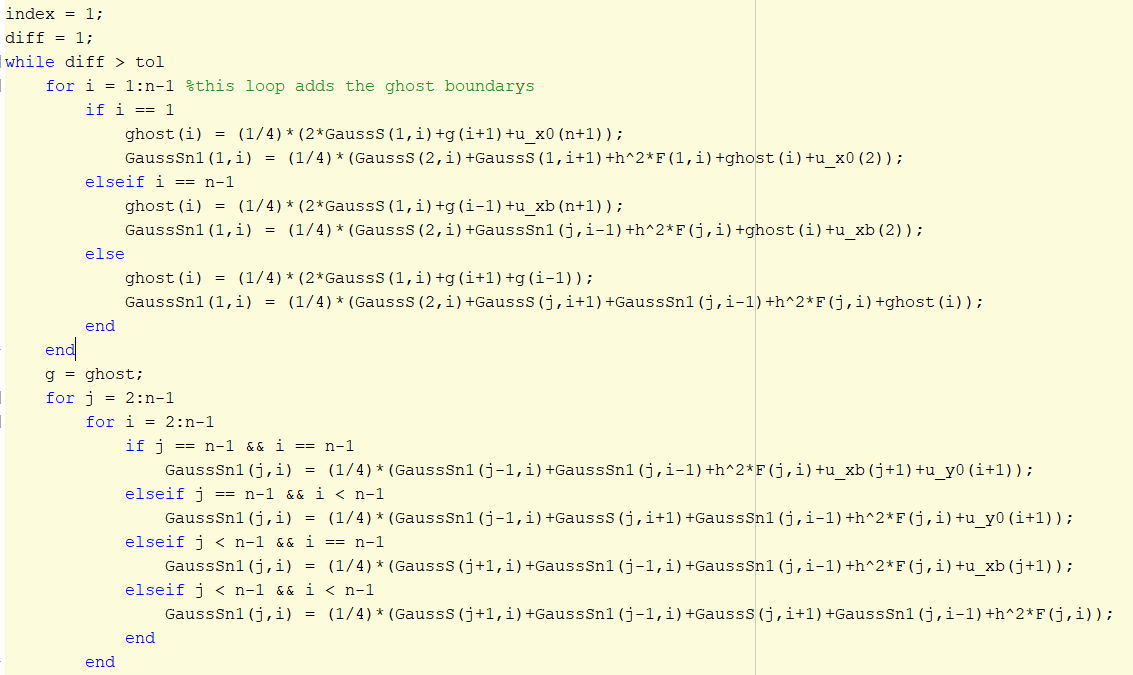
**Mathematical Statement of the Problem**

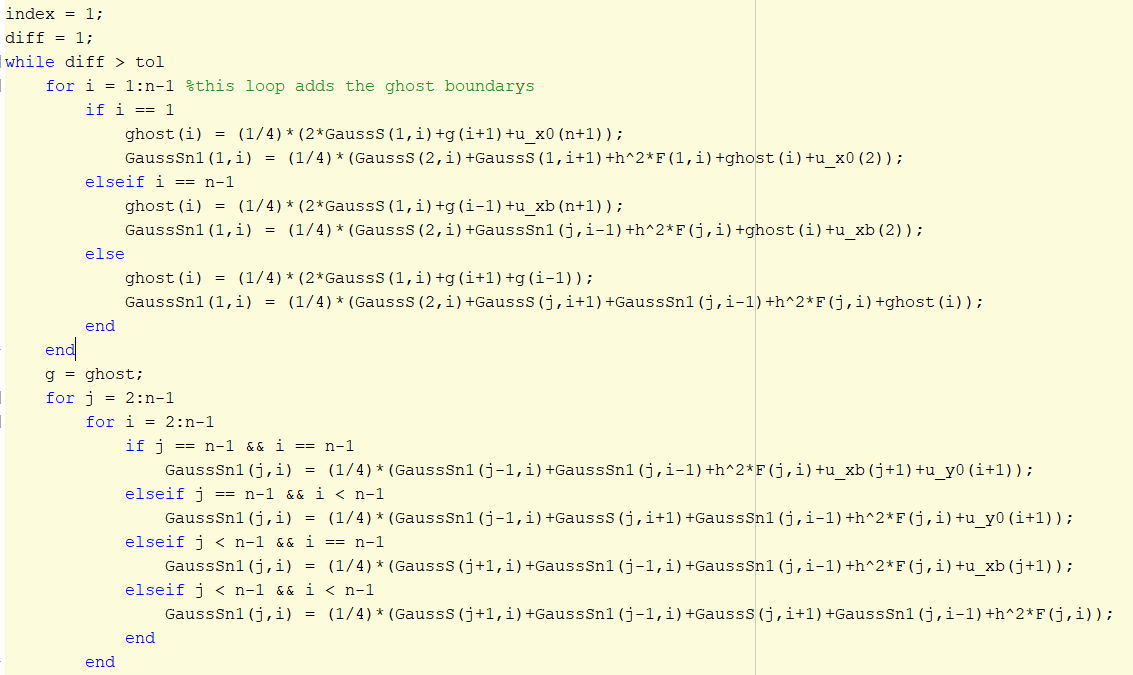
The problem I solved with my code was the two-dimensional Poisson equation with the form . The function on the right is The domain for the problem is and . The boundary conditions for the problem on the y domain are , and , and the boundary conditions for the x domain are , and . The two additional functions in this problem that govern the boundary conditions, and , are and .

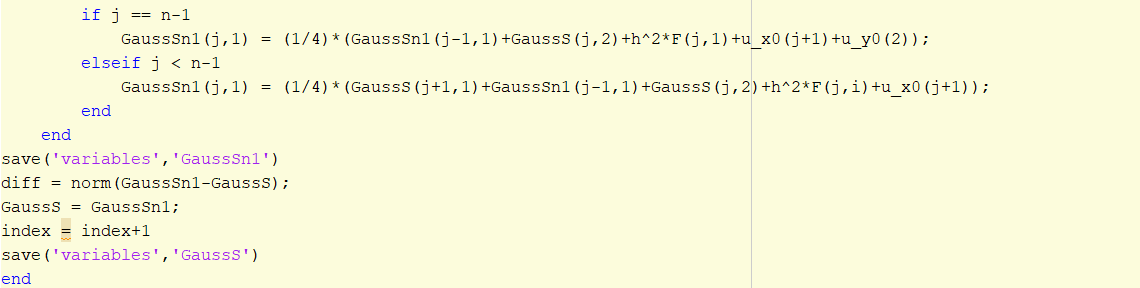
**Discretized Version of the Problem**

To successfully compute this problem using Poisson’s equation the problem needed to be discretized so that a numerical method could be applied to solve. The discretized equation for the problem is . Since x and y both have the same domain, and , we can say that where and n is equal to the number of steps from 0 to. Based on these conditions we can simplify the discretized equation even further to be .

**Numerical Methods**

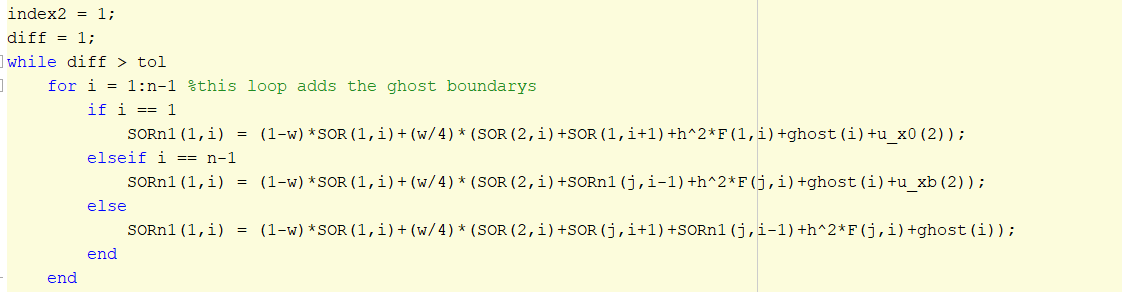
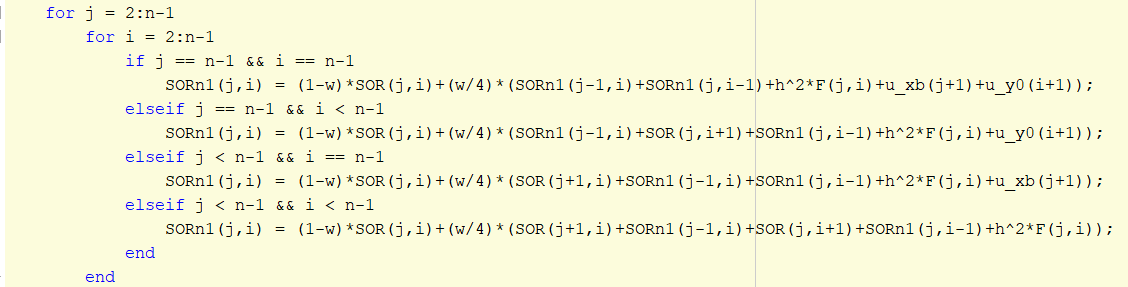
In order to solve the newly discretized problem we must implement a numerical method that will approximate the solution of the partial differential equation. The first method that was used to solve this problem was the Gauss-Seidel method. Gauss-Seidel method is an iterative method that solves for the solution of Poisson’s equation, or any other partial differential equation, by taking the boundary conditions and the values of the previous iteration to form a new iteration with more accurate numbers. This method solves for the next iteration at a given point by following the equation . For this equation n+1 represents values for the next iteration and n represents the current iteration. As you can see this method improves efficiency by taking the new values previously generated to help with finding values for the next point. The pseudocode for my solution is as follows.

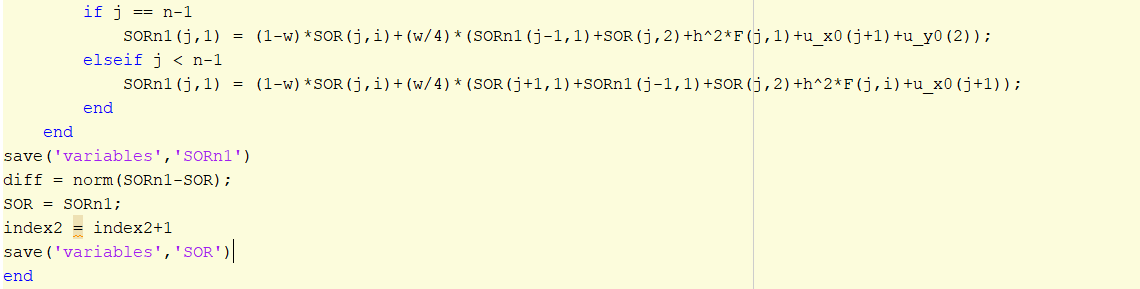




For this code the loop starts by solving for the ghost node boundary conditions and filling in the first row of the grid. After the ghost nodes and first row are made the code will then start to solve for the values of the rest of the rows excluding the first column. The first column is solved outside of the i loop to improve performance.

The next numerical method used to solve this problem is Successive Over Relaxation. This method is very similar to Gauss-Seidel method however it uses a new parameter ω to speed up the rate of convergence. The SOR method uses the follow equation to solve for the next iteration of a point . The pseudo code used to solve for this method is like the Gauss-Seidel method and is as follows.





The loop operates the exact same with the addition of the new term .

**Computer Specs**

* Processor: Intel® Core™ i7-5500U CPU @ 2.40 GHZ
* Motherboard: Inspiron 7558
* RAM: 8.00 GB
* System type: 64-bit operating system
* Graphics Card: Intel® HD Graphics 5500
* Cores: 2
* Sockets: 1
* L1 Cache: 128 KB
* L2 Cache: 512 KB
* L3 Cache: 4.0 Mb

**Results**

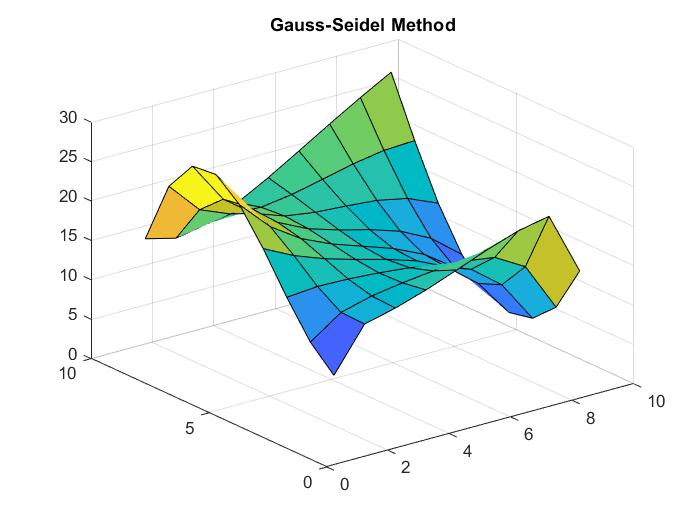
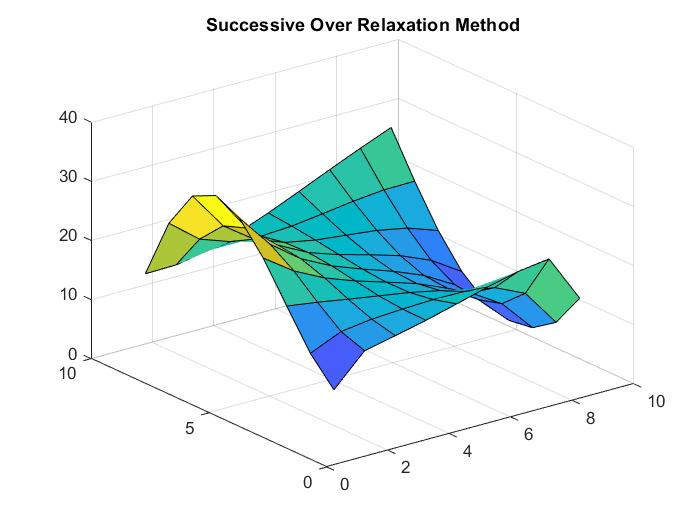
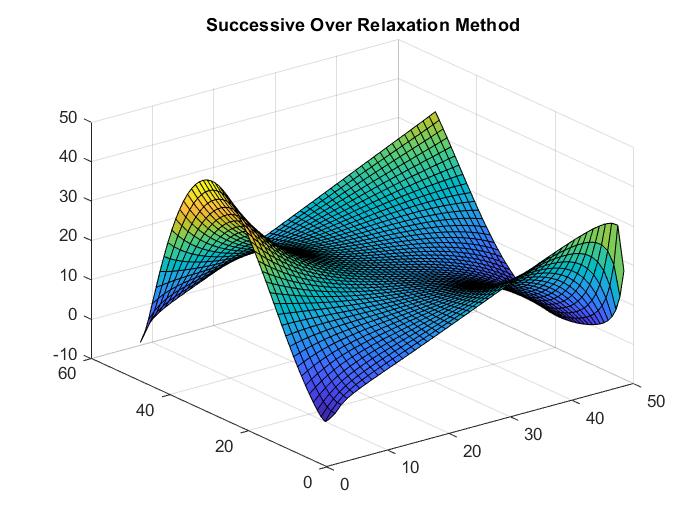
To do a proper simulation of the diffusion that occurs from Poisson’s equation a few additional parameters are necessary. First, we have the given domain for the problem. The x domain for the equation is and the y domain is . The boundary conditions are already defined for the problem as well. For my code the values for the initial iteration are all set to 0 and change as the diffusion occurs. Another parameter of the problem is the step size of the problem. The step size, h, is determined by the number of steps selected to reach the end of the domains. The number of steps needed can be set to any value, however by increasing the number of steps you will be achieving a more accurate solution. Below are some graphs that represent how increasing the number of steps change the results.

Figure 2: SOR with 10 steps

Figure 1: Gauss-Seidel with 10 steps





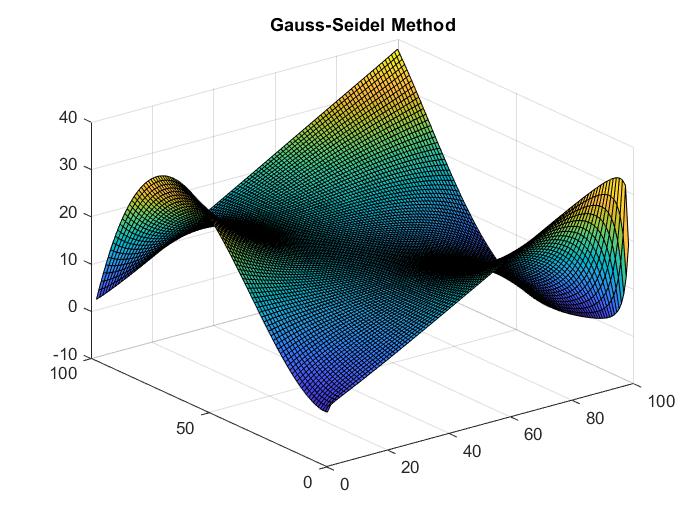
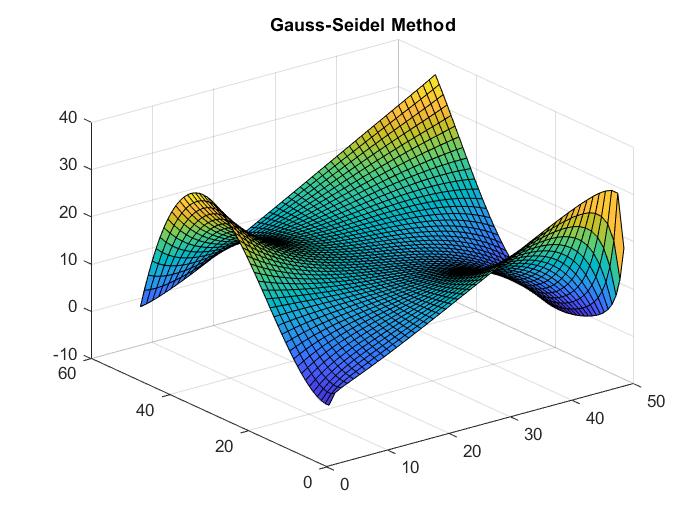
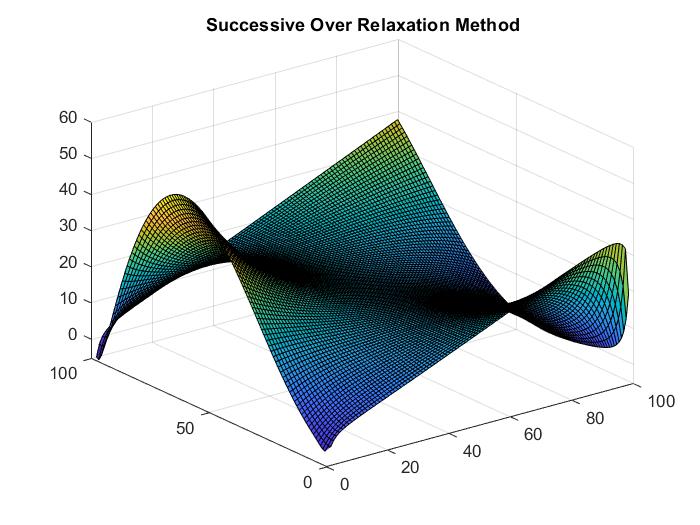


Figure 6: SOR with 100 steps

Figure 5: Gauss-Seidel with 100 steps

Figure 4: SOR with 50 steps

Figure 3: Gauss-Seidel with 50 steps

From these graphs you can see that as the number of steps increase the mesh used to approximate the method gets finer and finer. A finer mesh allows for the simulation to be more accurate and allows for the model to become smoother, creating a better representation of the real solution. To see the results of these changes a grid convergence study is often conducted on the solutions. For a successful grid convergence study, you will need at least 3 different meshes to compare results. For the grid convergence I checked these three meshes for the value at the center of the mesh, , and checked the results for convergence. The results of these tests are shown on the tables on the following page.

Table 1: Grid Convergence Data for Gauss-Seidel Method

|  |  |
| --- | --- |
| Value for U | Size of mesh, n |
| 15.8612 | 10 |
| 15.5053 | 50 |
| 15.4553 | 100 |

Table 2: Grid Convergence Data for SOR Method

|  |  |
| --- | --- |
| Value for U | Size of mesh, n |
| 17.2845 | 10 |
| 18.0302 | 50 |
| 18.1904 | 100 |

From these two tables you can see that as the size of the mesh increases the difference between the approximations begins to shrink significantly. The Gauss-Seidel method converges much faster then the SOR method, but the method takes much longer to solve for. The results of the two methods follow what is expected for Poisson’s equation, however the SOR method produce results that are slightly different than Gauss-Seidel due to the different numerical solutions. Both solutions converge to the assigned tolerance of which should provide sufficient enough of an approximation.