MECE 5397: Scientific Computing for Mechanical Engineers

Dr. Andrea Prosperetti

Dr. Amit Amritkar

Solving two-dimensional Poisson's Equation on a Rectangle using Gauss-Seidel and Successive Over-Relaxation Methods

Project ID: APc1-3

Nidal Kiwai Chaban

1278641

Table of Contents

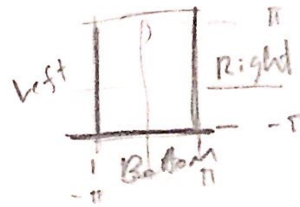
Table of Figures

Table of Tables

Abstract

Poisson’s equation in 2D is a very popular elliptic partial differential equation used very widely in mechanical engineering and theoretical physics. It describes the potential field caused by a given density distribution when the potential field is known. As part of this project, iterative approaches as Gauss-Seidel and Successive Over-Relaxation were used to solve this equation with set boundary conditions, several codes were created; these codes are optimized (as much as possible to the author’s knowledge) and commented thoroughly. Also, the codes include a checkpoint functionality and each has a counterpart code with a restart functionality in the case that the original code is stopped meanwhile it is running. Also, a separate code for verification and grid independence study was created using the Successive Over-Relaxation method. The project results show a steady and converging function for different mesh sizes and iterations with different orders of convergence as well as depiction of the discrepancies in the use of coarser vs finer meshes in approximation.

Mathematical Statement of the Problem



APc1-3

MECE 5397

Project A – Poisson Equation

Write a computer code to solve the two-dimensional Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -F(x, y) \tag{15}$$

The domain of interest is the rectangle

$$a_x < x < b_x\,, \qquad a_y < y < b_y \tag{16}$$

and the boundary conditions

$$u(x = a_x, y) = f_b(y)\,, \qquad u(x = b_x, y) = g_b(y)\,, \tag{17}$$

$$\left.\frac{\partial u}{\partial y}\right|_{y=b_y} = 0\,, \qquad u(x, y = a_y) = f_b(a_y) + \frac{x - a_x}{b_x - a_x}\left[g_b(a_y) - f_b(a_y)\right] \tag{18}$$

$$a_x = a_y = -\pi\,, \qquad b_x = b_y = \pi \tag{19}$$

$$f_b(y) = (b_y - y)^2 \cos\frac{\pi y}{b_y}\,, \qquad g_b(y) = y(b_y - y)^2 \tag{20}$$

$$F(x, y) = \cos\left[\frac{\pi}{2}\left(2\frac{x - a_x}{b_x - a_x} + 1\right)\right] \sin\left[\pi\frac{y - a_y}{b_y - a_y}\right] \tag{21}$$

Use ghost node(s) for Neumann condition(s).
After carrying out all the simulations needed for the report, run one last simulation with $F = 0$ and include the results in the report.

Discretization of the Equations

Discretizing Poisson Equation

$$\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = -F(x,y)$$

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = -F_{i,j}$$

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = -h^2F_{i,j}$$

Rearranging for $u_{i,j}$

$$u_{i,j} = \frac{1}{4}[u_{i+1.j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + h^2F_{i,j}]$$

For Gauss-Seidel Method

$$u_{i,j} = \frac{1}{4}[u^n_{i+1.j} + u^{n+1}_{i-1,j} + u^n_{i,j+1} + u^{n+1}_{i,j-1} + h^2F_{i,j}]$$

For Successive Over Relaxation Method

$$u_{i,j} = \frac{\beta}{4}\left[u^n_{i+1.j} + u^{n+1}_{i-1,j} + u^n_{i,j+1} + u^{n+1}_{i,j-1} + h^2F_{i,j}\right] + (1-\beta)u^{n-1}_{i,j}$$

Description of Numerical Methods Used

Gauss-Seidel Method

The Gauss-Seidel Method is an iterative linear technique for solving square systems. It is defined by the form

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

where the procedure is generally continued until the changes between x and the next iteration of x are below a set tolerance.

Convergence for Gauss-Seidel method are dependent on the matrix A. It converges if A is symmetric positive-definite or A is strictly diagonally dominant. Even if conditions are not satisfied, Gauss-Seidel converges sometimes.

```
Inputs: A, b
Output: ϕ

Choose an initial guess ϕ to the solution
repeat until convergence
    for i from 1 until n do
        σ ← 0
        for j from 1 until n do
            if j ≠ i then
                σ ← σ + a_ij ϕ_j
            end if
        end (j-loop)
        ϕ_i ← (1/a_ii)(b_i − σ)
    end (i-loop)
    check if convergence is reached
end (repeat)
```

Successive Over Relaxation

The successive over-relaxation (SOR) is a variant of the Gauss-Seidel method for solving a linear system of equations, resulting in faster convergence.

$(D+\omega L)\mathbf{x} = \omega \mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}$ where w is the relaxation factor, thus

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$

Where w is the choice for relaxation, and depends of the coefficient matrix. Usually, w is greater than 0 and smaller than 2.

```
Inputs: A, b, ω
Output: ϕ
```

```
Choose an initial guess ϕ to the solution
repeat until convergence
  for i from 1 until n do
    σ ← 0
    for j from 1 until n do
      if j ≠ i then
        σ ← σ + a_ij ϕ_j
      end if
    end (j-loop)
    ϕ_i ← (1 − ω)ϕ_i + (ω / a_ii)(b_i − σ)
  end (i-loop)
  check if convergence is reached
end (repeat)
```

Technical Specifications of Computer Used

- Processor – Intel Core i7-3770S CPU @ 3.10GHz
- RAM - 8 GB
- Hard Drive - 500 GB
- Graphics Card - any with DisplayPort/HDMI or DVI support - desktop only
- Monitor – Dell OptiPlex widescreen LCD with DisplayPort/HDMI or DVI support

Results

Graphs

For the Project given, we obtained a contour plot as shown

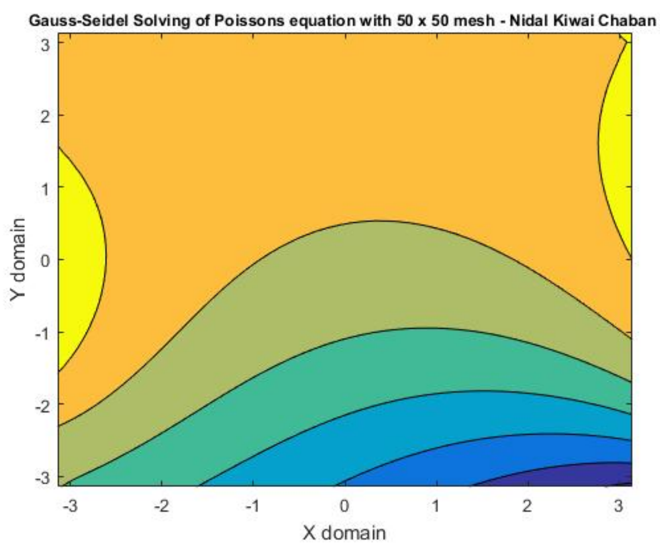


Figure 1-Contour Plot

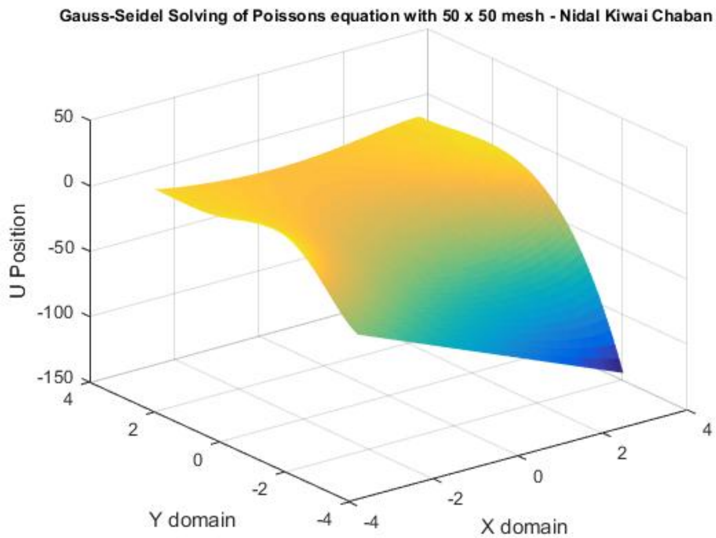


Figure 2-Surface Plot

Parameters used in simulations

- Input Parameters
  - Number of nodes n for n x n mesh size
  - ax = ay = -π
  - bx = by = π
  - Boundary Conditions as stated
- Output Parameters
  - Number of iterations
  - Elapsed time
  - Mean of u (for grid independence)

Effect of Number of Points Used For Discretization

Table 1-Gauss-Seidel Method

Gauss-Seidel Method		
Mesh size	Iterations for tol=1e-6	Elapsed time (seconds)
10x10	34	.037481
20x20	769	.5911
50x50	4992	5.66963
100x100	20976	34.6258
200x200	Excessive time consumed	
1000x1000		
5000x5000		

Table 2-Successive Over-Relaxation Method

Successive Over-Relaxation Method		
Mesh size	Iterations for tol=1e-6	Elapsed time (seconds)
10x10	8	.008049
20x20	269	.158379
50x50	1811	1.503082
100x100	7547	12.2652
200x200	33060	164.638
1000x1000	Excessive time consumed	
5000x5000		

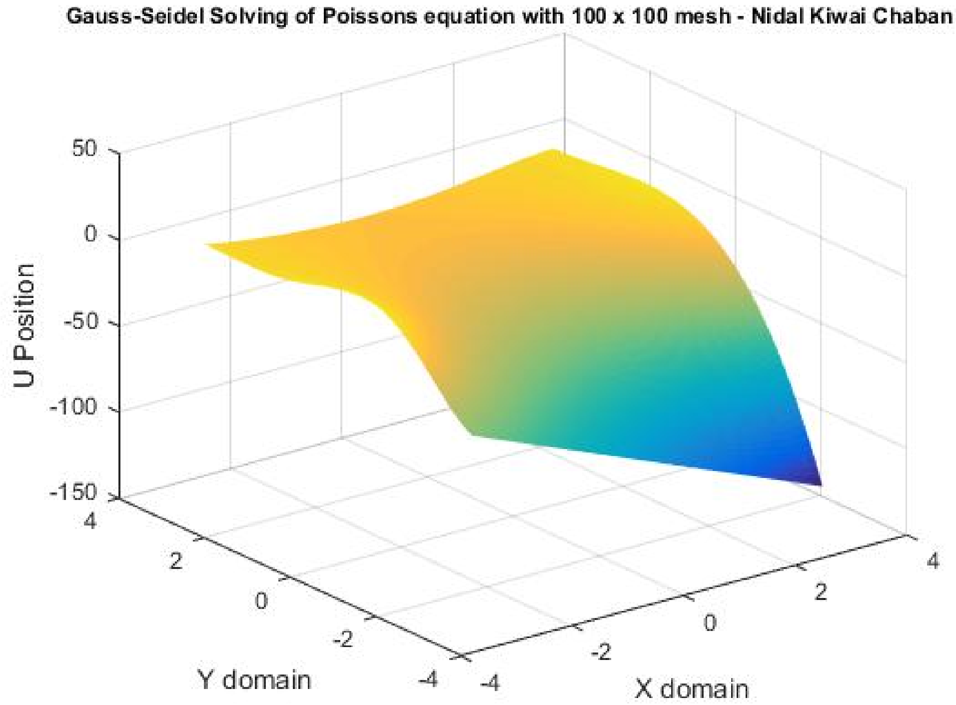


Figure 3-Surface plot of 100x100 mesh for Gauss-Seidel

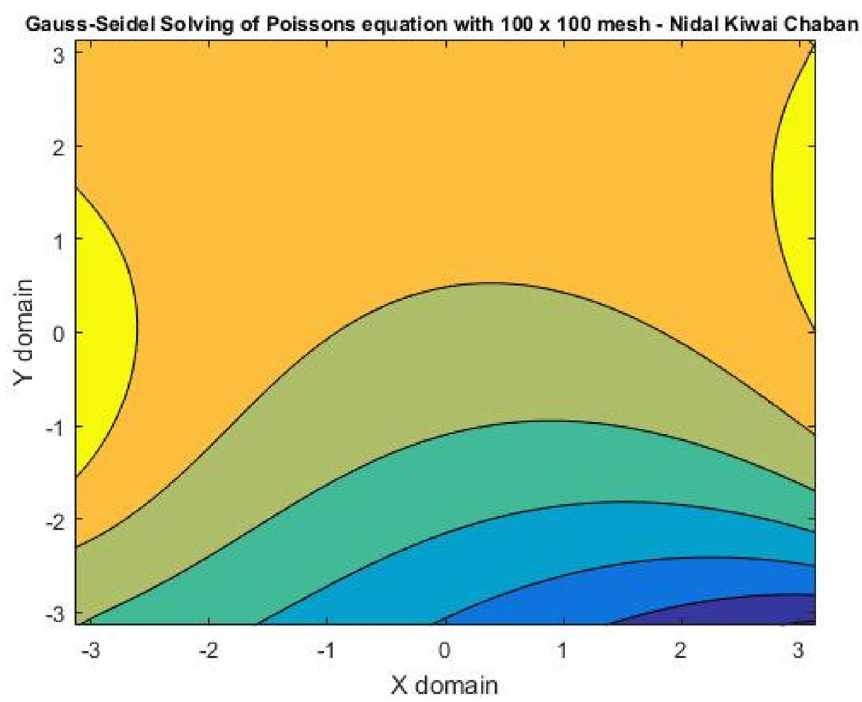


Figure 4-Contour plot of 100x100 mesh for Gauss-Seidel

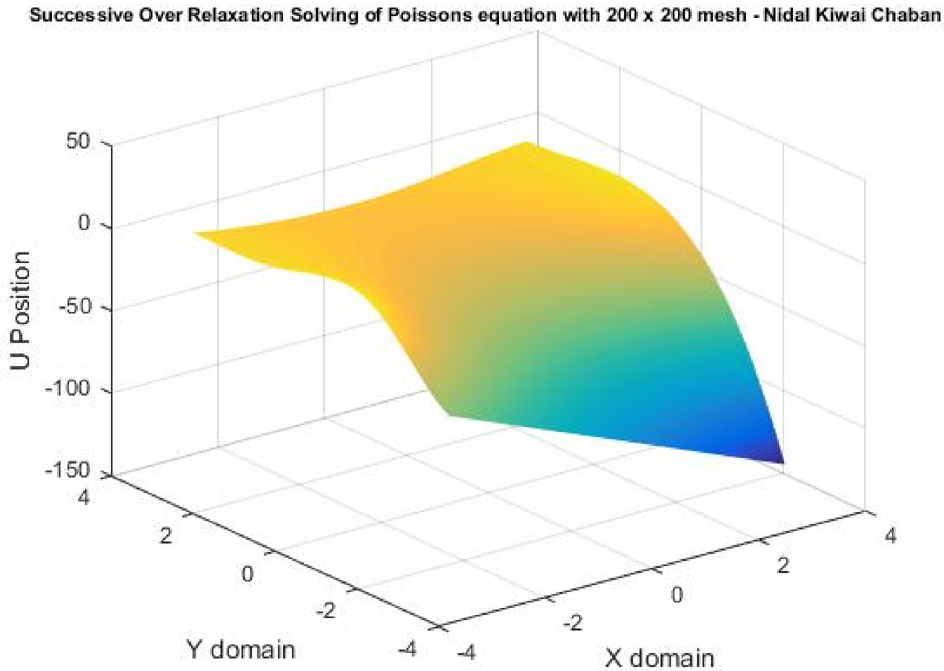


Figure 5-Surface Plot of 200 x 200 mesh of SOR

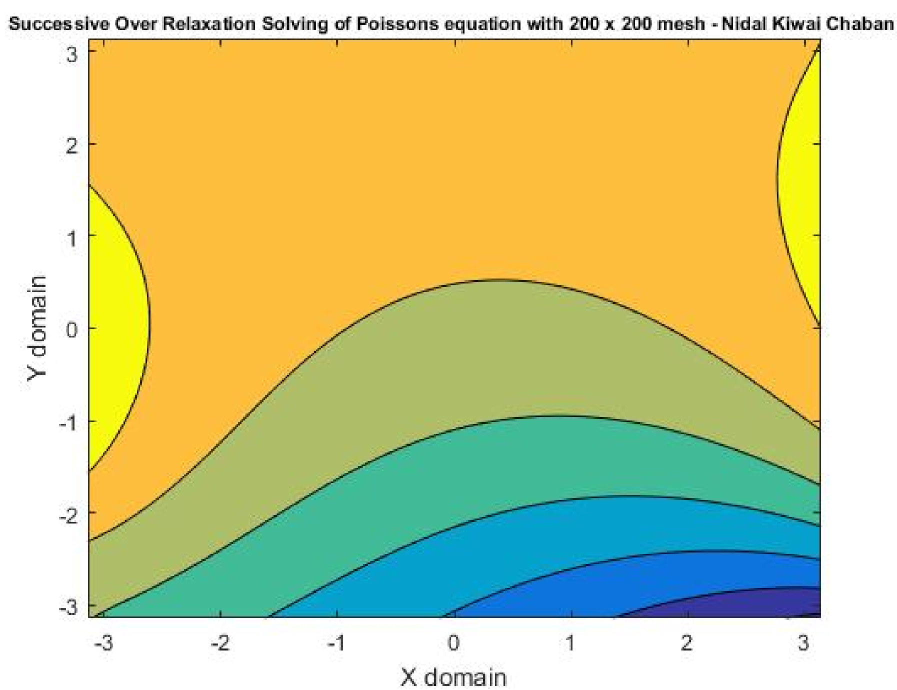


Figure 6-Contour Plot of 200x200 mesh for SOR

As it is shown, the finer the mesh is, the more accurate and the more iterations are required to come to convergence. This is due to the need for every node to come within tolerance to be considered a satisfactory simulation run.

Verification

In order to verify that the approximations are correct, a comparison of the error between results is done with a fixed iteration method, this method will demonstrate that as iteration for specific size of meshes increase, the accuracy in results will increase as well. For this part, Successive Over-Relaxation method will be used since it converges faster and a wider range of iterations are able to be used.

Mesh Size: 30		
Iterations	**Elapsed Time**	**Error of u(i) and u(i-1)**
10	.007264	5.4605
100	.013694	9.6395e-01
1000	.085238	8.6302e-05
5000	.405092	-0.000

Mesh Size: 70		
Iterations	**Elapsed Time**	**Error of u(i) and u(i-1)**
10	.010761	2.2038e01
100	.054540	3.9100e00
1000	.487224	1.5703e01
5000	2.424058	3.9048e-04
10000	4.695324	7.4827e-07

Mesh Size: 200		
Iterations	**Elapsed Time**	**Error of u(i) and u(i-1)**
100	.402067	3.0773e01
1000	4.010405	2.6782e02
10000	40.235330	3.6238e-01
20000	79.6598	5.8127e-03

As expected, for each mesh size, as iteration number increases, the biggest error between u and the previous u value becomes smaller and smaller. Although having a coarser mesh make error smaller for larger numbers of iterations, it doesn't represent the entire equation as good as having a larger mesh. The following graphs show 10000 iterations for a 20x20 mesh and a 60x60 mesh to show this discrepancy. For the first table, it shows that curves are steeper for smaller mesh sizes; while for the second table, color gradients are smoother for greater mesh sizes.

Table 3-Comparison of Contour Plots of 20x20 and 60x60 mesh

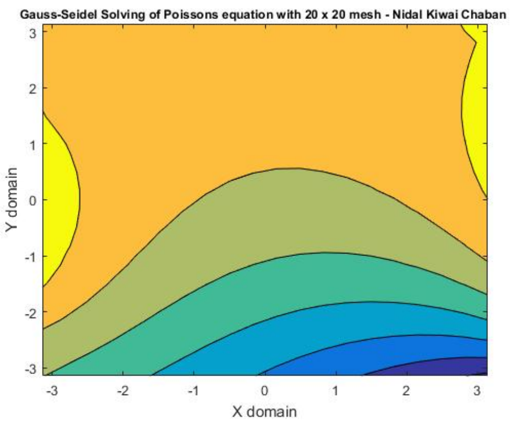
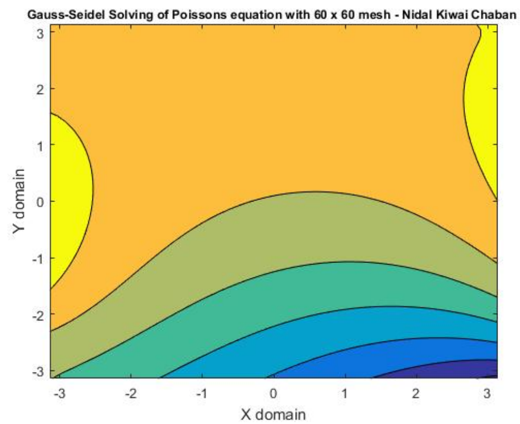
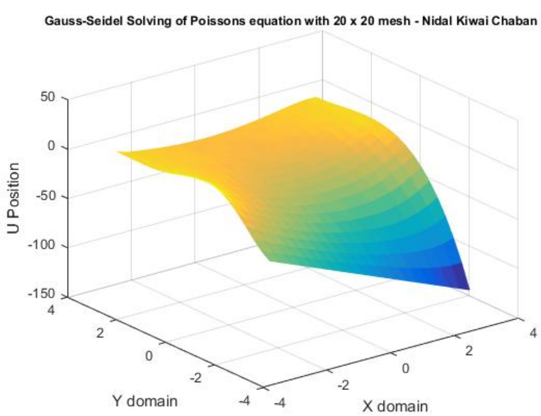
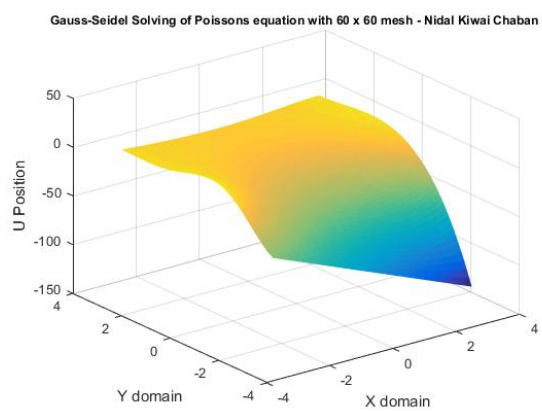
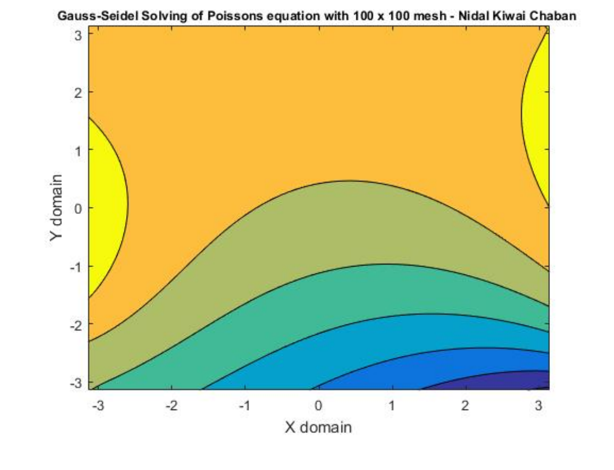
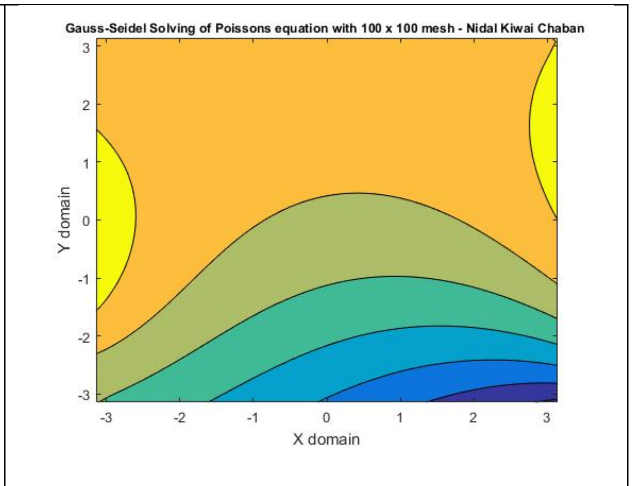
	
20x20 mesh	60x60 mesh

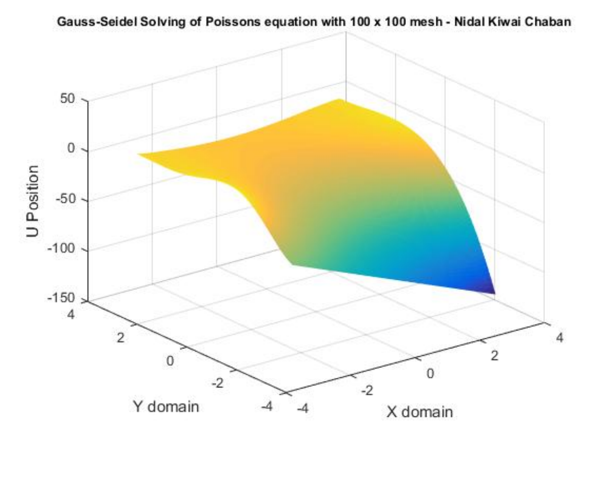
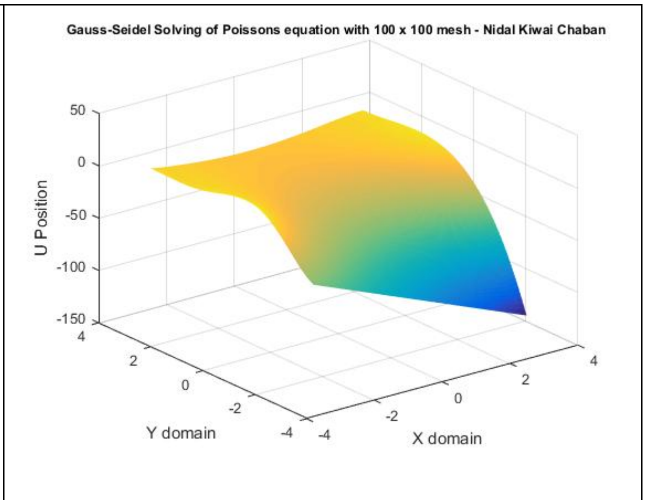
Table 4-Comparison of Surface Plots of 20x20 and 60x60 mesh

	
20x20 mesh	60x60 mesh

Comparing the function to a forcing function of F=0

Changing the forcing function transforms Poisson’s equation into Laplace’s equation. Here is a comparison of 100x100 nodes between Poisson’s and Laplace equations using Gauss-Seidel

	
Laplace	Poisson

	
Laplace	Poisson

Gauss-Seidel Method for Poissons Equation with Forcing Function of 0 (Laplace)

Mesh size: 100 x 100

Optimal Iterations: 21839

Elapsed time: 42.463649

Biggest error value between u(i) and u(i-1): 9.9995e-07

Mean of u: -2.5346e+01>>

Gauss-Seidel Method for Poissons Equation with a Forcing Function

Mesh size: 100 x 100

Optimal Iterations: 21839

Elapsed time: 44.397125

Biggest error value between u(i) and u(i-1): 9.9995e-07

Mean of u: -2.5346e+01>>

The only change between Laplace and Poisson’s for this case is a slightly faster convergence for the Laplace form of the equation. Graphs and data for u is very similar for both. Also, if a fixed iteration number is simulated between the two, using SOR, it is shown that the error for small iterations is bigger when there is a forcing function present.

Gauss-Seidel Method for Poissons Equation with a Function of 0 (Laplace)

Mesh size: 50 x 50

Optimal Iterations: 0

Fixed Iterations: 100

Elapsed time: 0.030500

Biggest error value between u(i) and u(i-1): 3.9455e+00

Mean of u: -1.4008e+01>>

Gauss-Seidel Method for Poissons Equation with a Forcing Function

Mesh size: 50 x 50

Optimal Iterations: 0

Fixed Iterations: 100

Elapsed time: 0.030176

Biggest error value between u(i) and u(i-1): 1.0996e+01

Mean of u: -1.4075e+01>>