This project entails the representation of the Helmholtz Equation in two dimensions using numerical methods in MATLAB. Boundary conditions, domain limits, and constant parameters are prescribed in the assignment. The Gauss-Seidel/Liebmann method with and without relaxation is used to represent the equation.

MATHEMATICAL FORMULATION

The Helmholtz equation is an elliptic partial differential equation of the second order with respect to the spatial variables. It is similar to the Poisson equation but includes an additional output term that is not differentiated. The equation is

The parameter is given as 1. The domain is a rectangle, given as

,

Where

The boundary conditions for y are given as

,

The boundary conditions for x are given as

The forcing function F is given as

DISCRETIZATION

For the discretization of this PDE, the centered difference formula is used, where

The spatial terms are set such that where is the area of the squares used to discretize the domain. Substituting the approximate terms into the original equation yields

Solving for ,

This is the equation used to implement the Gauss-Seidel method. The insulated/Neumann boundary condition on the right side of the rectangle is accounted for through the use of a ghost node. The ghost node uses the centered difference formula as such:

Exploiting this fact for the right side of the domain results in

allowing for the use of the discretized formula at the edge of the domain.

Additionally, the technique known as Successive Overrelaxation (SOR) is employed. Overrelaxation is the process where the previous and current iterations are weighted to speed up convergence. Mathematically, this is represented as

where a of 1 would be equivalent to no relaxation, would be underrelaxation, and would be overrelaxation. The term will be used to prevent any confusion with the term in the problem statement.

PSEUDOCODE

The following is the core loop for the implementation of the Gauss-Seidel method. It passes through a full column of the y coordinates before moving to the next x value in order to exploit the column-major ordering of Matlab. The loop termination condition is based upon the infinity norm of the matrix, continuing to iterate until the maximum difference between two iterations is less than 1%.

Helmholtz

while error > 0.01

uprev = u

for i = 2:length(x)- 1

for j = 2:length(y) - 1

u(i,j) = [u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1)

* Δ^2 \* F(i,j)] / (4 - Δ^2 \* Λ)

end

u(i,end) = [2 \* u(i,end-1) + u(i+1,end) + u(i-1,end)

- Δ^2 \* F(i,end)] / (4 - Δ^2 \* Λ)

end

error =

end

The pseudocode for the SOR method is very similar and includes only a couple of additional lines to implement it.

SORΛ = 1.2

while error > 0.01

uprev = u

for i = 2:length(x)- 1

for j = 2:length(y) - 1

u(i,j) = [u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1)

* Δ^2 \* F(i,j)] / (4 - Δ^2 \* Λ)

u(i,j) = SORΛ \* u(i,j) + (1-SORΛ) \* uprev(i,j)

end

u(i,end) = [2 \* u(i,end-1) + u(i+1,end) + u(i-1,end)

- Δ^2 \* F(i,end)] / (4 - Δ^2 \* Λ)

u(i,end) = SORΛ \* u(i,j) + (1-SORΛ) \* uprev(i,j)

end

error =

end

MACHINE TECHNICAL SPECIFICATIONS

OS: Windows 10 Home 64-bit

CPU:

* Intel Celeron N2830 2.16 GHz (Dual-core)
* L1 Cache: 2 x 24 KB
* L2 Cache 1024 KB

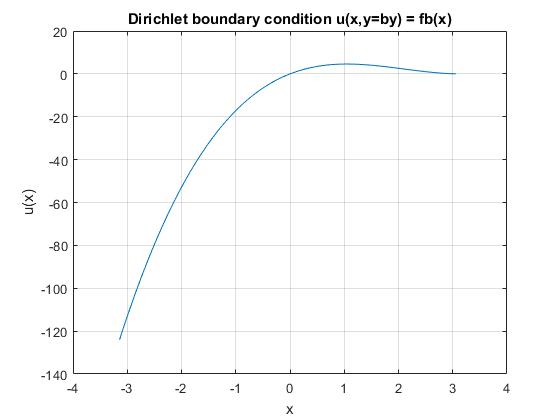
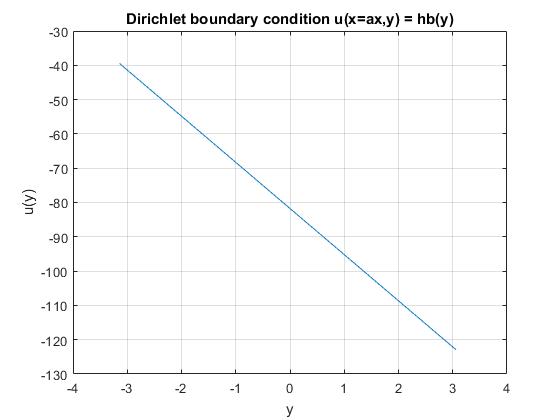
Memory: 4 GB DDR3 800 MHz

GPU: Intel HD Graphics 2GB VRAM

Disk: WD Blue 500GB 5400 RPM

RESULTS

The first step taken in the approach to this problem was to verify the implementation of the boundary conditions and the forcing function. The three Dirichlet boundary conditions were fairly straightforward to implement, and are shown in Figure 1.



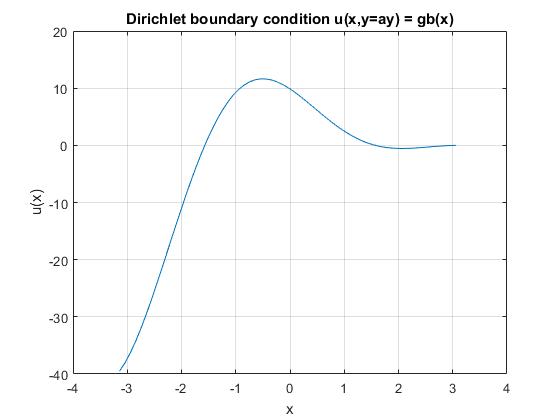


Figure . Dirichlet Boundary Conditions

As can be seen, these boundary conditions form three continuous sides of the domain, representing the left, bottom, and top respectively. The forcing function F(x,y) is shown in Figures 2 and 3.

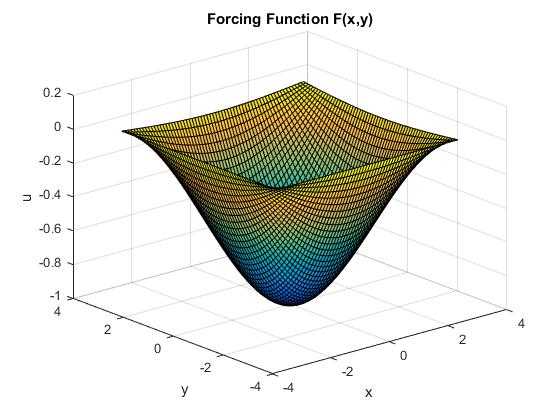


Figure . Surface of the forcing function F(x,y)

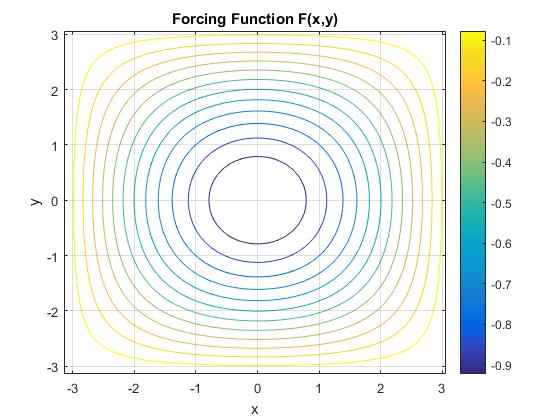


Figure . Contour of forcing function F(x,y)

The right side (x=bx) of the domain has a Neumann boundary condition, which showed some unusual behavior, shown in Figures 2 and 3.

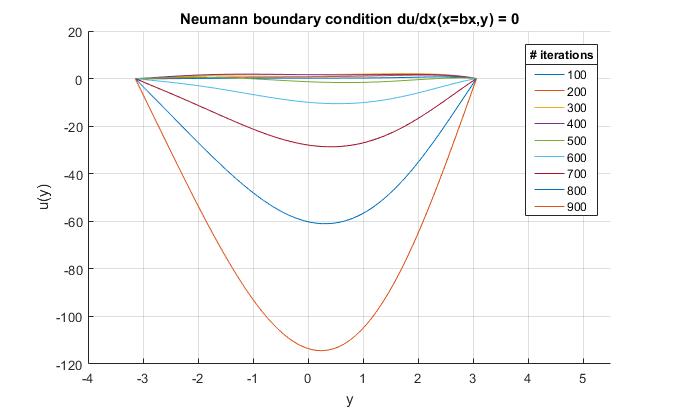


Figure . Neumann condition, delta = 0.1

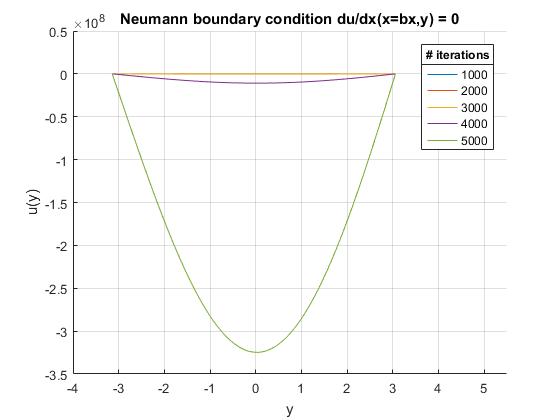


Figure . Neumann condition, delta = 0.1

The Neumann boundary condition was accounted for using the centered difference formula and ghost nodes, as shown in the Mathematical Formulation section. Because of this, the values for this boundary condition were part of the iterations of the Gauss-Seidel loop, and changed with each iteration. Figures 2 and 3 show the effect of increasing iterations on the boundary values for . At each of these instances the flux remains zero, satisfying the condition, but otherwise continues to diverge to (note the y-axis values on Figure 3). This behavior was considered unusual so simplifications of the Helmholtz equation, namely the Laplace and Poisson equation, were tested using the same boundary conditions to compare the behavior.

The Laplace equation in 2 dimensions is represented by

,

while the Poisson equation is represented by

These equations occur when setting the forcing function and in the Helmholtz equation to 0. Figure 6 shows the surface and contour plots of the Laplace equation, and Table 1 shows numerical results for several different grid sizes.

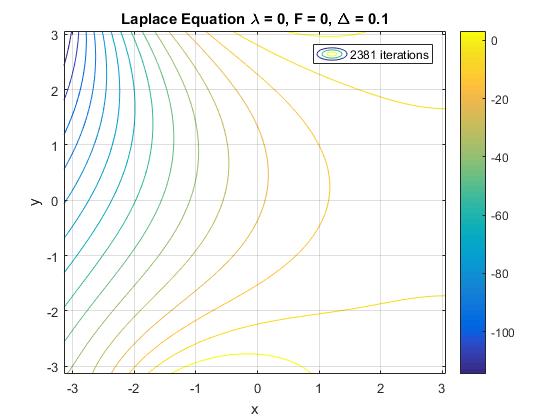
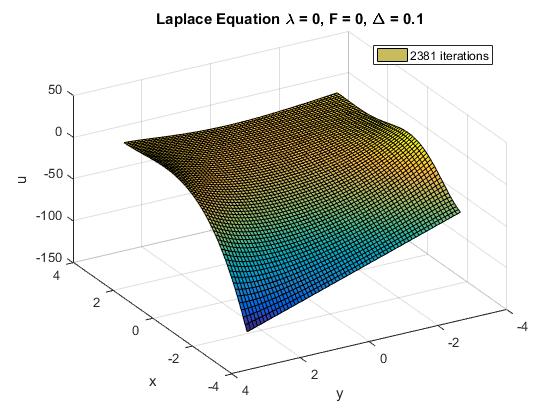


Figure . Surface and contour of Laplace function

Table . Results for Laplace equation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Laplace** | | Gauss-Seidel | | |  | SOR |  |
| Δ | # nodes | max | mean | min | max | mean | min |
| 0.25 | 632 | 11.5384 | -25.3143 | -123.579 | 11.5384 | -24.9498 | -123.579 |
| 0.1 | 3948 | 11.6244 | -25.044 | -122.906 | 11.6244 | -24.9609 | -122.906 |
| 0.025 | 63165 | 11.6369 | -24.8688 | -123.915 | 11.6369 | -24.496 | -123.915 |

The figures show a smooth and continuous response and maintain the given boundary conditions, and the numerical results show little change between differing grid sizes and relaxation methods. Appendix A contains additional plots and figures for the different grid sizes and relaxations for both the Laplace and Poisson equations. It is apparent that the numerical solver works well for the Laplace equation and displays grid independence. Increasing in complexity is the Poisson equation, results of the numerical implementation of which are shown in Figure 7 and Table 2. The results are nearly the same as with the Laplace equation, again showing little variation with changes in grid size and the implementation of relaxation.

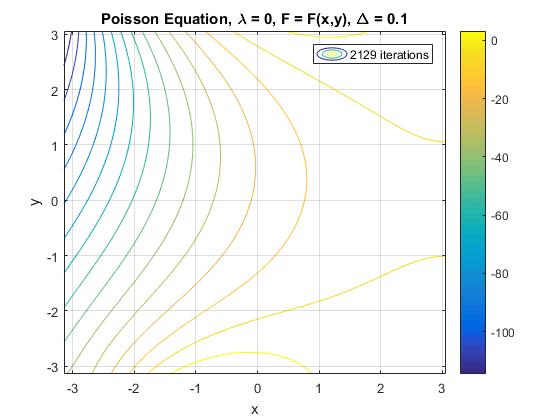
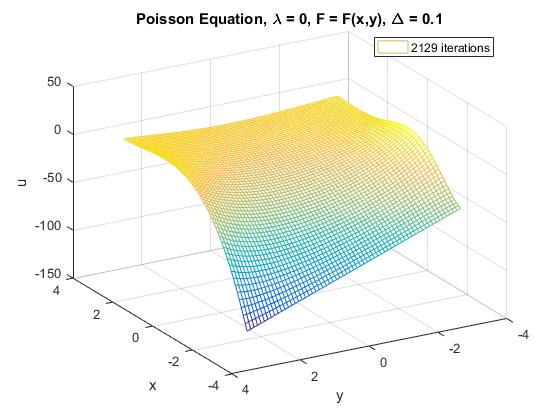
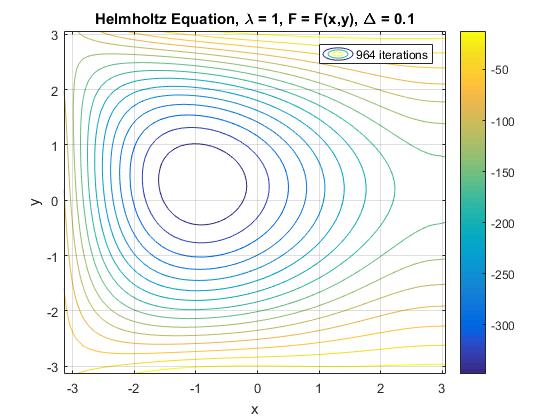
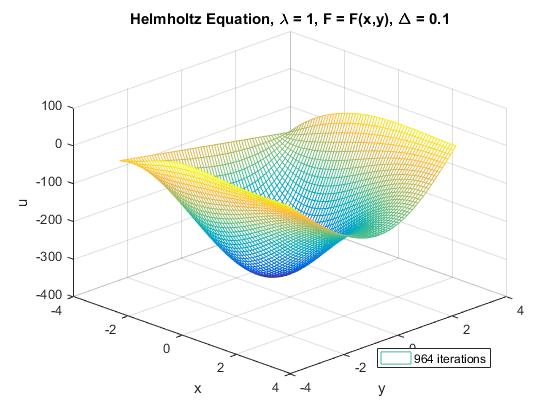


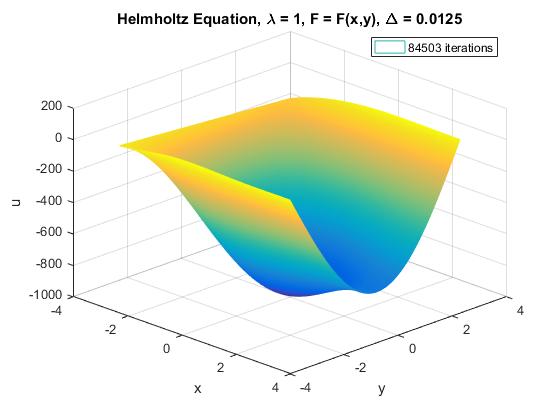
Figure . Surface and contour plots of the Poisson equation

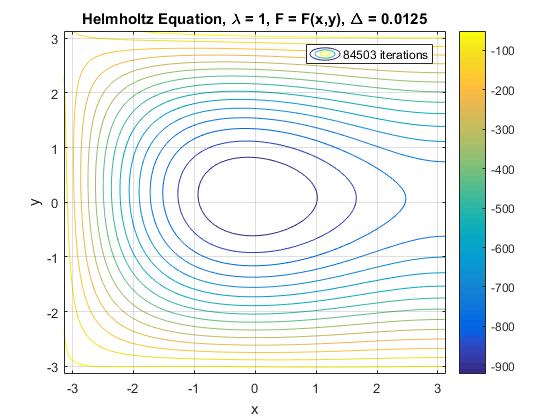
Table . Results for Poisson equation

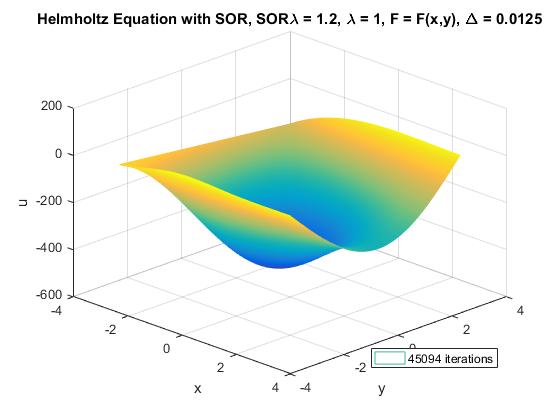
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Poisson** | | Gauss-Seidel | | |  | SOR |  |
| Δ | # nodes | max | mean | min | max | mean | min |
| 0.25 | 632 | 11.5384 | -24.2268 | -123.578 | 11.5384 | -24.9498 | -123.579 |
| 0.1 | 3948 | 11.6244 | -23.8074 | -122.906 | 11.6244 | -24.9609 | -122.906 |
| 0.025 | 63165 | 11.6369 | -23.6992 | -123.915 | 11.6369 | -24.496 | -123.915 |

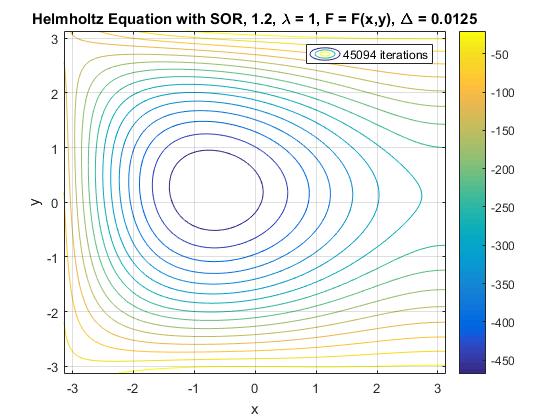
These simpler test cases were implemented to verify that the Gauss-Seidel solver could produce coherent results with and without relaxation before tackling the full Helmholtz equation.



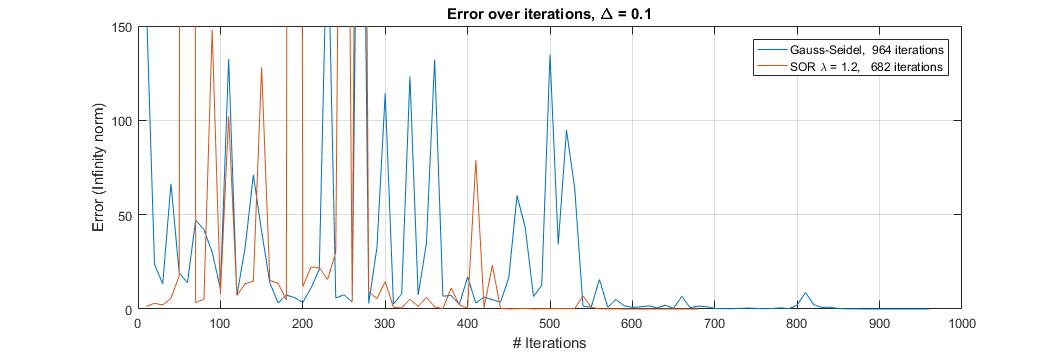






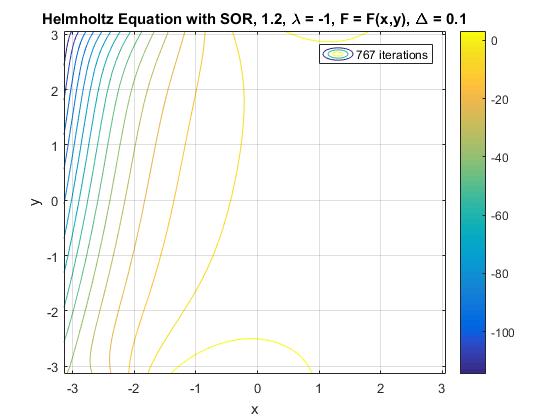


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Helmholtz** | | Gauss-Seidel | | |  | SOR |  |
| Δ | # nodes | max | mean | min | max | mean | min |
| 0.15 | 1755 | 11.6076 | -383.614 | -761.42 | 11.6076 | -2188.6 | -5195.4 |
| 0.1 | 3948 | 11.6244 | -170.721 | -372.087 | 11.6244 | -224.167 | -462.064 |
| 0.05 | 15791 | 11.6369 | -259.296 | -534.755 | 11.6369 | -276.696 | -560.354 |
| 0.025 | 63165 | 11.6369 | -514.55 | -1004.4 | 11.6369 | -526.96 | -1023.5 |
| 0.0125 | 252662 | 11.6378 | -503.868 | -982.956 | 11.6378 | -242.328 | -500.082 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Helmholtz** | | Gauss-Seidel | SOR (λ = 1.2) |
| Δ | # nodes | # iterations | # iterations |
| 0.1 | 3948 | 964 | 682 |
| 0.05 | 15791 | 4421 | 2923 |
| 0.025 | 63165 | 21210 | 13903 |
| 0.0125 | 252662 | 84503 | 45094 |

|  |  |
| --- | --- |
| Δ = 0.025 | |
| SOR λ | # iterations |
| 1 | 17183 |
| 1.1 | 14078 |
| 1.2 | 13903 |
| 1.3 | 11255 |
| 1.4 | 8985 |
| 1.5 | 7019 |
| 1.6 | 5299 |
| 1.7 | 3783 |
| 1.8 | 2441 |
| 1.9 | 1258 |
| 2 | 24280 |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Helmholtz (Λ = -1)** | | Gauss-Seidel | | |  | SOR |  |
| Δ | # nodes | max | mean | min | max | mean | min |
| 0.1 | 3948 | 11.6244 | -12.5682 | -122.906 | 11.6244 | -12.8613 | -122.906 |

APPENDIX A: ADDITIONAL PLOTS AND FIGURES

