**A Comparison of Solving the Poisson Equation Using Different Numerical Methods in MATLAB**

**MECE 5397- Scientific Computing**

**Project A – Poisson Equation AP02-1**

Luis Espinoza, 1226327

Friday May 5th, 2017

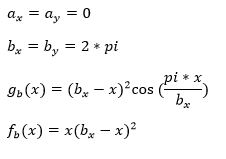
**Abstract**

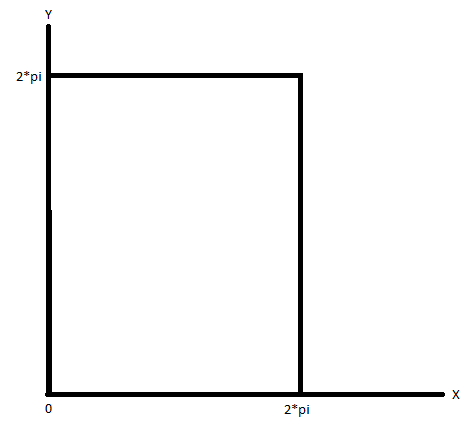
Systems of linear equations resulting from partial differential equations arise frequently in many phenomena such as heat, sound, and fluid flow. We apply the finite difference method to the Poisson equation with 3 homogeneous Dirichlet boundary conditions, 1 nonhomogeneous Neumann boundary condition, and a right side forcing function. We use the center-difference approximation to discretize the Poisson equation. Then, we use the Gauss Seidel and Successive Over-Relaxation (SOR) methods to obtain a numerical solution to the Poisson Equation. The resulting graph, performance statistics, and grid convergence study are all presented in this report for each numerical method. After comparing the performance statistics for the two numerical methods, the SOR method is concluded to be the fastest numerical method to solve the Poisson equation.

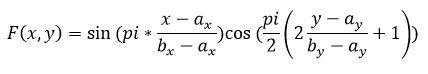
**Mathematical Statement of Problem**

In this report, we will test various numerical methods to solve the Poisson Equation. The Poisson equation is an elliptic Partial Differential Equation (PDE) that is linear and has constant coefficients. The Poisson equation is used to model phenomena such as the potential field caused by a given charge or mass density distribution.

We will test the numerical methods on a Poisson equation with 3 nonhomogeneous, Dirichlet boundary conditions, 1 homogenous, Neumann boundary condition, and a right-hand side function F(x,y).







**Discretized Version of the Equation**

The partial derivatives in the PDE are approximated by linear combinations of function values at the grid points. The second-order center difference approximation is applied to both the x and y second derivatives at all points in the mesh.



As a result, the approximated x and y second derivatives of u at a given mesh point (i, j) are given below through the evaluation of u at (i+1, j), (i-1, j), (i, j+1), and (i, j-1).

After rearranging the above equation, the discretized Poisson equation is,

Where ∆x=∆y=h is the grid spacing between nodes. This equation will be rearranged for ui,j=… later depending on the numerical method.

A quick note must be made about the discretization of the Neumann boundary condition.

Using the second-order center difference approximation to approximate the x partial derivative, the following discretization is achieved.

Simplify

This is substituted in for the Neumann boundary condition in the numerical method.

**Description of Numerical Methods Analyzed**

***Gauss-Seidel Method***

The Gauss-Seidel method is a classical iterative method. It will be used to solve the linear system of equations presented in the previous section. The pseudocode for the method is shown below.

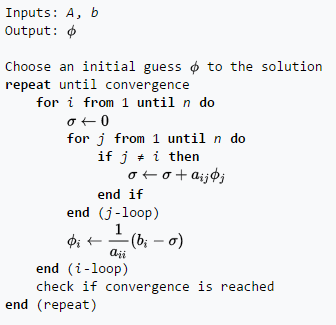


Figure 1: Gauss-Seidel Method Algorithm, Wikipedia

The method starts with a guess for the solution. In the case of our Poisson Equation project, we assumed that all interior nodes (2:Nx-1 and 2:Ny-1) are initially zero. Then, the linear system of equations is solved using the initial guess and the error in relation to the previous value of the solution is calculated.

The process continues until convergence is obtained by the numerical method. Convergence is obtained when the maximum error is less than the user-input tolerance (typically 1e-06).

To solve for ui,j in MatLAB, the discretized equation is rearranged to the following form.

***Successive Over-Relaxation Method***

The SOR method is another classical iterative method quite similar to the Gauss-Seidel method but expected to result in faster convergence. It will be used to solve the linear system of equations presented in the previous section. The pseudocode for the method is shown below.

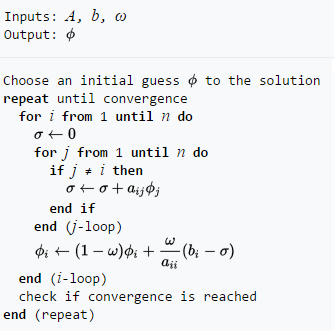


Figure 2: SOR Method Algorithm, Wikipedia

The method starts with a guess for the solution. In the case of our Poisson Equation project, we assumed that all interior nodes (2:Nx-1 and 2:Ny-1) are initially zero. Then, the linear system of equations is solved using the initial guess. The difference with the Gauss-Seidel method is that a SOR coefficient (B) must be included. The coefficient is bounded by 1<B<2. For the case of our Poisson Equation, a value of B=1.5 was used. Finally, the error in relation to the previous value of the solution is calculated.

The process continues until convergence is obtained by the numerical method. Convergence is obtained when the maximum error is less than the user-input tolerance (typically 1e-06). To solve for ui,j in MatLAB, the discretized equation is rearranged to the following form.

**Technical Specifications of Computer Used**

The machine used for this project is located at the University of Houston Engineering Computing Center. The machine is an Intel ® Xeon ® CPU E5620 @ 2.40GHz with 1 core/CPU and a current CPU clock frequency of 2394 MHz (max CPU clock frequency of 2660 MHz). The machine has 64 memory channels, a DRAM total width of 32 bits, and a total DRAM per CPU of 16384 MB.

* 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**

***Gauss Seidel***

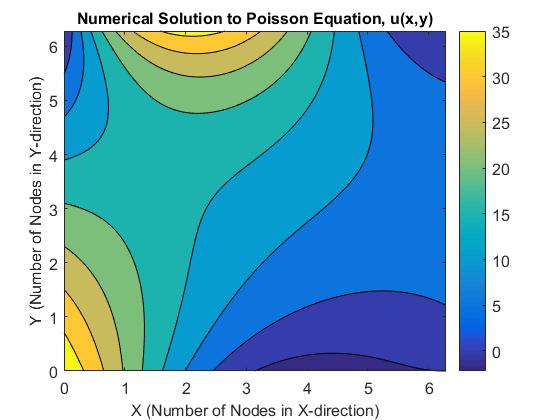
******

Figure 3: Contour Plot of Solution, Gauss-Seidel

Figure 3 shows the contour plot of the numerical solution to the Poisson equation obtained via the Gauss-Seidel method with a mesh size of N=100. By inspecting the boundaries of the plot, it is clear that they match the boundary conditions given by the problem.

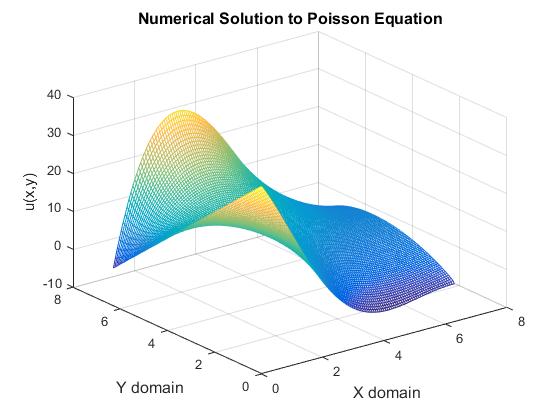
******

Figure 4: Surface Plot of Solution, Gauss-Seidel

Figure 4 shows the surface plot of the numerical solution to the Poisson equation obtained via the Gauss-Seidel method with a mesh size of N=100. The surface plot perfectly matches the information provided by the contour plot. The table below will display the code’s performance based on number of iterations and running time to reach convergence.

Table 1: Performance results for Gauss-Seidel Method

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Number of Iterations | Running Time |
| 4 | 22 | 0.022970 |
| 8 | 102 | 0.082787 |
| 16 | 432 | 0.315920 |
| 32 | 1,956 | 2.568503 |
| 64 | 7,890 | 13.475713 |
| 128 | 35,745 | 70.703948 |
| 256 | 123,659 | 575.944671 |
| 512 | - | Excessive Time required |

The following parameters are used in the simulation:

* Tolerance= 1e-06
* Checkpointing frenquency= 10 iterations

By analyzing Table 1, we notice that the number of iterations and running time to obtain a solution increase as the mesh size increases. It can be concluded that the size of the mesh or the number of discretization points has an impact on the performance of the numerical method in terms of convergence speed. It is positive to notice that the numerical method does converge to a unique solution which validates the MATLAB code written and presented in the /src/ directory.

Table 2: Grid Convergence Study for Gauss Seidel Method

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Average Value of Solution | Difference to Previous Value |
| 10-N | 13.9550 | 0.0 |
| 20-2N | 13.6847 | 0.2703 |
| 40-4N | 13.5723 | 0.1124 |
| 80-8N | 13.5217 | 0.0506 |
| 160-16N | 13.4976 | 0.0241 |

A grid convergence study is performed to determine if more nodes are needed to obtain an accurate numerical solution to the studied PDE. The simulation is ran for increasing mesh sizes (N, 2N, 4N, 8N, 16N, 32N, 64N, 128N) and the average value of the solution at each mesh size is determined via the mean MATLAB function. Then, the average solution value is compared to the previous average solution and when the average solution value doesn’t differ much between neighboring mesh sizes then there are sufficient nodes in the simulation. For the grid convergence study shown in Table 2 for a Gauss-Seidel method, the difference approaches zero as the value of N increases. N=160 nodes seems to be the optimal mesh size. The fact that the difference goes to zero as N increases is another validation that the written code works.

***Successive Over-Relaxation***

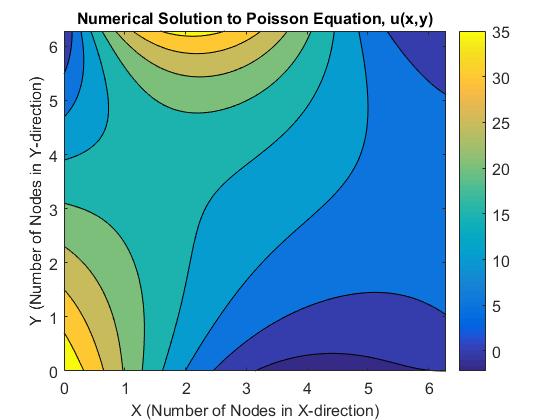
******

Figure 4: Contour Plot of Solution, SOR

Figure 4 shows the contour plot of the numerical solution to the Poisson equation obtained via the SOR method with a mesh size of N=100. By inspecting the boundaries of the plot, it is clear that they match the boundary conditions given by the problem.

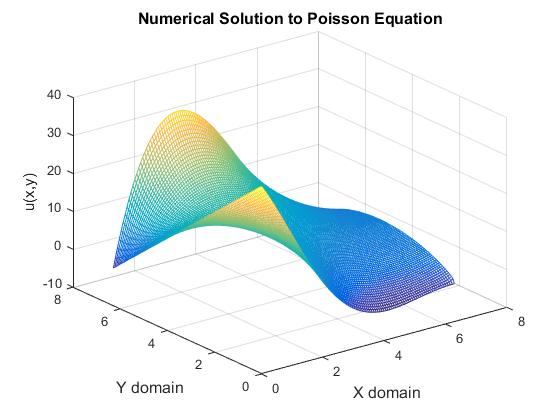
******

Figure 5: Surface Plot of Solution, SOR

Figure 5 shows the surface plot of the numerical solution to the Poisson equation obtained via the SOR method with a mesh size of N=100. The surface plot perfectly matches the information provided by the contour plot. The table below will display the code’s performance based on number of iterations and running time to reach convergence.

Table 3: Performance results for SOR Method

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Number of Iterations | Running Time |
| 4 | 30 | 0.040106 |
| 8 | 45 | 0.037680 |
| 16 | 165 | 0.267341 |
| 32 | 684 | 0.439422 |
| 64 | 3,737 | 3.538062 |
| 128 | 11,161 | 17.567156 |
| 256 | 48,387 | 241.784734 |
| 512 | - | Excessive Time required |

The following parameters are used in the simulation:

* Tolerance= 1e-06
* Checkpointing frenquency= 10 iterations

By analyzing Table 3, we notice that the number of iterations and running time to obtain a solution increase as the mesh size increases. It can be concluded that the size of the mesh or the number of discretization points has an impact on the performance of the numerical method in terms of convergence speed. It is positive to notice that the numerical method does converge to a unique solution which validates the MATLAB code written and presented in the /src/ directory.

In comparison to the Gauss-Seidel method, the SOR method converges much quicker. For example, it takes the Gauss-Seidel method 123,659 iterations and 576 seconds to converge with a mesh size of N=256 while it takes the SOR method 48,387 iterations and 242 seconds.

Table 4: Grid Convergence Study for SOR Method

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Average Value of Solution | Difference to Previous Value |
| 10-N | 13.9832 | 0 |
| 20-2N | 13.6499 | 0.333 |
| 40-4N | 13.5168 | 0.1331 |
| 80-8N | 13.4582 | 0.0586 |
| 160-16N | 13.4308 | 0.0274 |

A grid convergence study is performed to determine if more nodes are needed to obtain an accurate numerical solution to the studied PDE. The simulation is ran for increasing mesh sizes (N, 2N, 4N, 8N, 16N, 32N, 64N, 128N) and the average value of the solution at each mesh size is determined via the mean MATLAB function. Then, the average solution value is compared to the previous average solution and when the average solution value doesn’t differ much between neighboring mesh sizes then there are sufficient nodes in the simulation. For the grid convergence study shown in Table 4 for a SOR method, the difference approaches zero as the value of N increases. N=160 nodes seems to be the optimal mesh size. The fact that the difference goes to zero as N increases is another validation that the written code works.

As a final requirement for the assignment, I run the simulation for F=0 using the Gauss-Seidel and obtained the following results.

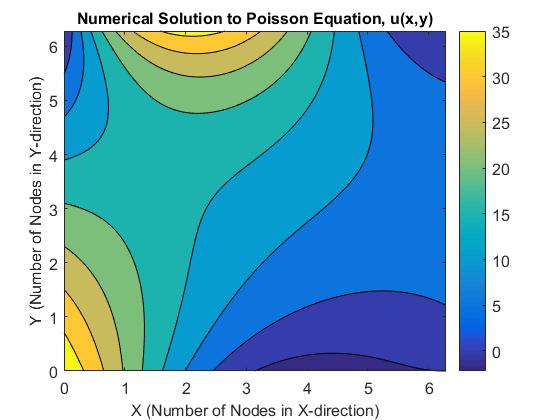
******

Figure 6: Contour Plot of Solution, Gauss-Seidel with F=0

Figure 6 shows the contour plot of the numerical solution to the Poisson equation obtained via the Gauss-Seidel method with a mesh size of N=100 and F=0. The plot is similar to one obtained for F=function(x,y).

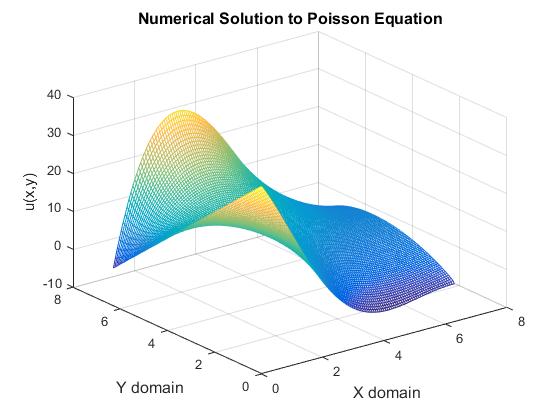
******

Figure 7: Surface Plot of Solution, Gauss-Seidel and F=0

Figure 7 shows the surface plot of the numerical solution to the Poisson equation obtained via the Gauss-Seidel method with a mesh size of N=100 and F=0. The plot is similar to one obtained for F=function(x,y).

Table 5:: Performance results for Gauss-Seidel Method and F=0

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Number of Iterations | Running Time |
| 4 | 22 | 0.037059 |
| 8 | 102 | 0.075814 |
| 16 | 432 | 0.477242 |
| 32 | 1,837 | 3.217593 |
| 64 | 7,693 | 8.791824 |
| 128 | 37,159 | 73.651444 |
| 256 | 165,330 | 750.122054 |
| 512 | - | Excessive Time required |