MECE 5397- Scientific Computing for Mechanical Engineers

Project A – Helmholtz Equation (AHc2-4)

Bryan Soto Salinas

UH ID: 1279821

Class time: MW @ 1-2:30pm

Date: May 9th, 2018

Abstract:

This project focuses on solving a two-dimensional Helmholtz equation given specific boundary conditions (both Neumann and Dirichlet) and rectangle domain. In order to complete with the requirements of the problem statement, it was crucial to create a MATLAB code that would be able to simulate all the conditions needed to approximate the solution numerically. The simulation takes into account two different adaptive iterative methods for linear systems; the Gauss-seidel method and the successive over-relation method (SOR). Both methods follow a similar derivation, especially since the SOR method is a more generalized than the Gauss-seidel method. However, each differ in some respects such as the arithmetic derivation, and also the time each method takes to convergence to the solution. Such aspects serve as a way to analyze the properties of each method’s results through the use of visualizations and graphs for a better comprehension of the code.

The main body of this report is divided into several steps taken to achieve the project’s plan development. The mathematical statement of the problem section provides with information about the its scope and problem definition, the mathematical derivation of the discretized equations used for the boundary value (Neumann) and the finite difference approximation (central) of the inner grid points, and a succinct description of the iteration methods used both the Gauss-seidel and SOR. Another part of the body introduces the technical specifications of the computer where the code was executed, to offer some insight of the computer’s performance, memory information, and processing power.

After running and debugging the script that solves the project assignment, the results are presented in the shape of graphs and other visualizations. The code written has comments to allow a better comprehension of the code and the output allowed to make a mesh convergence and independence study. The results will be analyzed for both methods and compared to each other in terms of the number of iterations taken to convergence, the difference in approximated values, error analysis and the convergence solution.

Mathematical statement of problem:

Given a fixed frequency and by applying the Fourier transformation in time, the Maxwell equations are reduced to a set of stationary equations in the frequency domain, which in turn can be compacted into a single differential equation of the second order known as the Helmholtz equation. This type of applications is usually modeled in domains whose geometries are very complex, which makes it unlikely to obtain a closed or analytical expression for the solution of said equation, so its numerical approximation is fundamental. The Helmholtz equation is a time-independent partial differential equation that conforms the category of elliptic PDEs, and it aids in the study of physical problems in both space and time. Figure 1 shows the particular Helmholtz equation for the specified boundary conditions, domain conditions, and given constant values such as the wavenumber (Λ) and the domain values ax, ay, bx, and by.

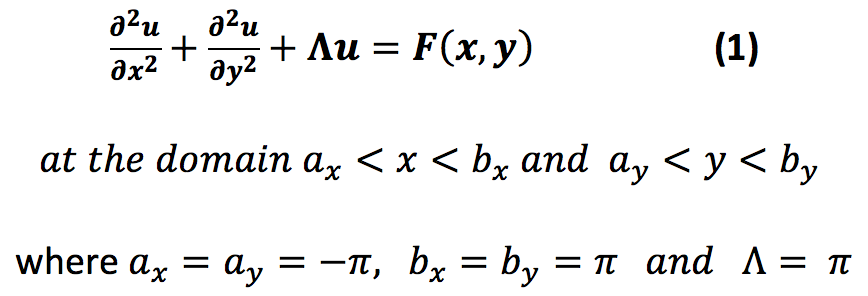


Figure 1- Helmholtz Equation with specified conditions for chosen assignment

Equation 1 is a generalized form of the Laplace’s equation (also elliptic), which is derived by setting the vector function F=0 and Λ=0. The Laplace’s case will also be simulated as the final part of the assignment and results will be presented in the latter part of the report. It is also important to note that the case where F=0 and Λ<0, the equation becomes the spatial part of the parabolic diffusion equation, which is significant for the understanding of the wave propagation property of the equation in matter.

The stated problem requires boundary conditions in order to be solvable. The boundary conditions specified in the problem statement are particular to a rectangle domain and are different for every side of the proposed rectangle grid of n by m points, as shown in Figure 2.

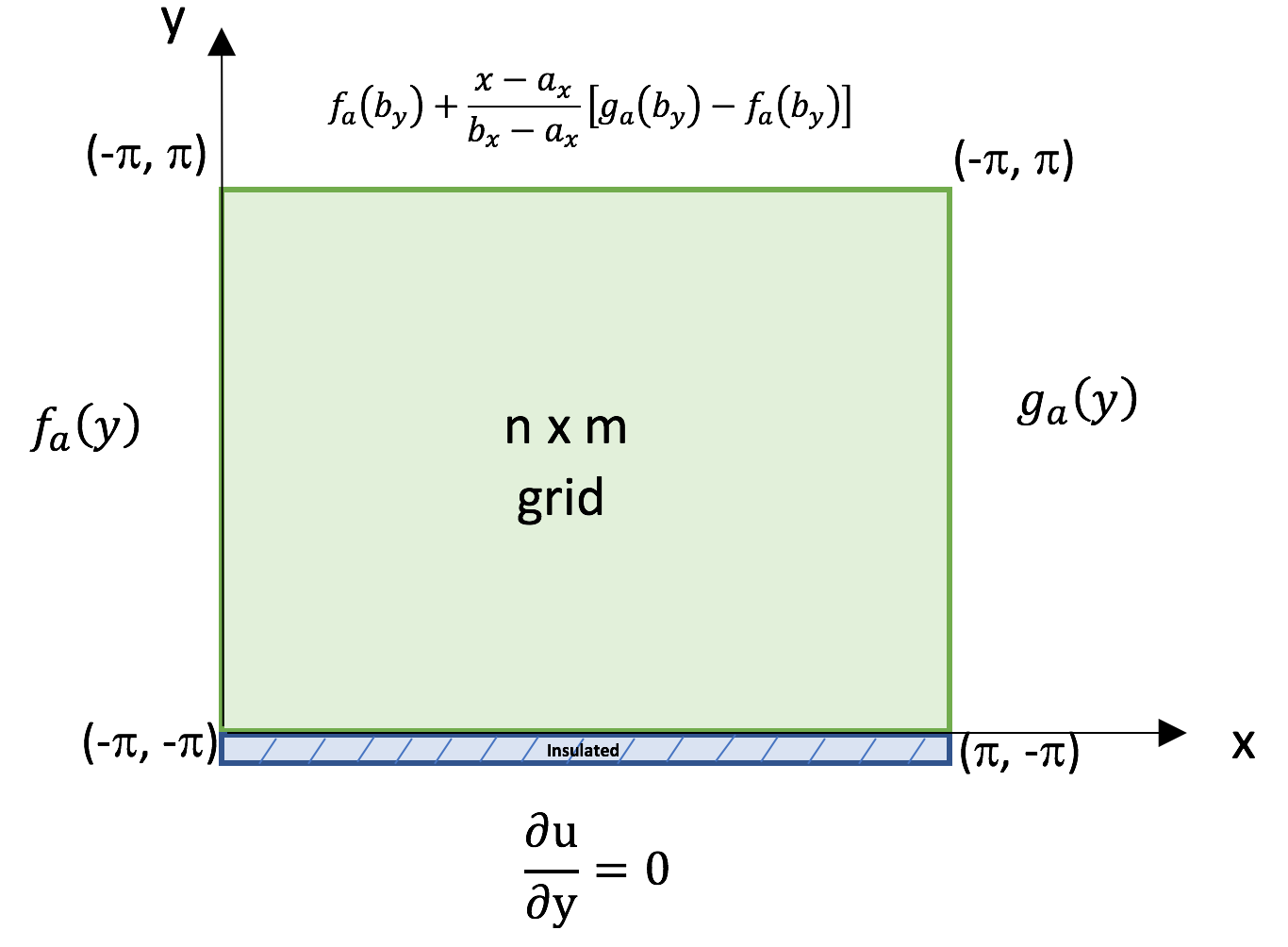


Figure 2- Representation of rectangle grid sample with boundary conditions

The values for each boundary are representative of each side of the rectangle given by different functions that ultimately affect the behavior of the boundary. The upper, left and right sides of the rectangle each represent a different function but are all Dirichlet conditions because the boundary value is known. Because they are all functions, however, the values along each boundary differs according to the x and/or y-position of the nodes within the sides of domain. This type of boundary condition is easy to apply because the solution of the boundary condition is fixed throughout the edge. The boundary on the bottom edge of the rectangle is imposed as a partial derivative, meaning that the solution to the boundary value is implied and needs to be derived. If we were talking about temperature being the u, the Neumann condition would represent the flux of heat through the edge. Because the partial derivative equation is homogenous, it is equivalent to the boundary being insulated. Figure 3 shows the given boundary functions as well as the governing equation functions in more detail.

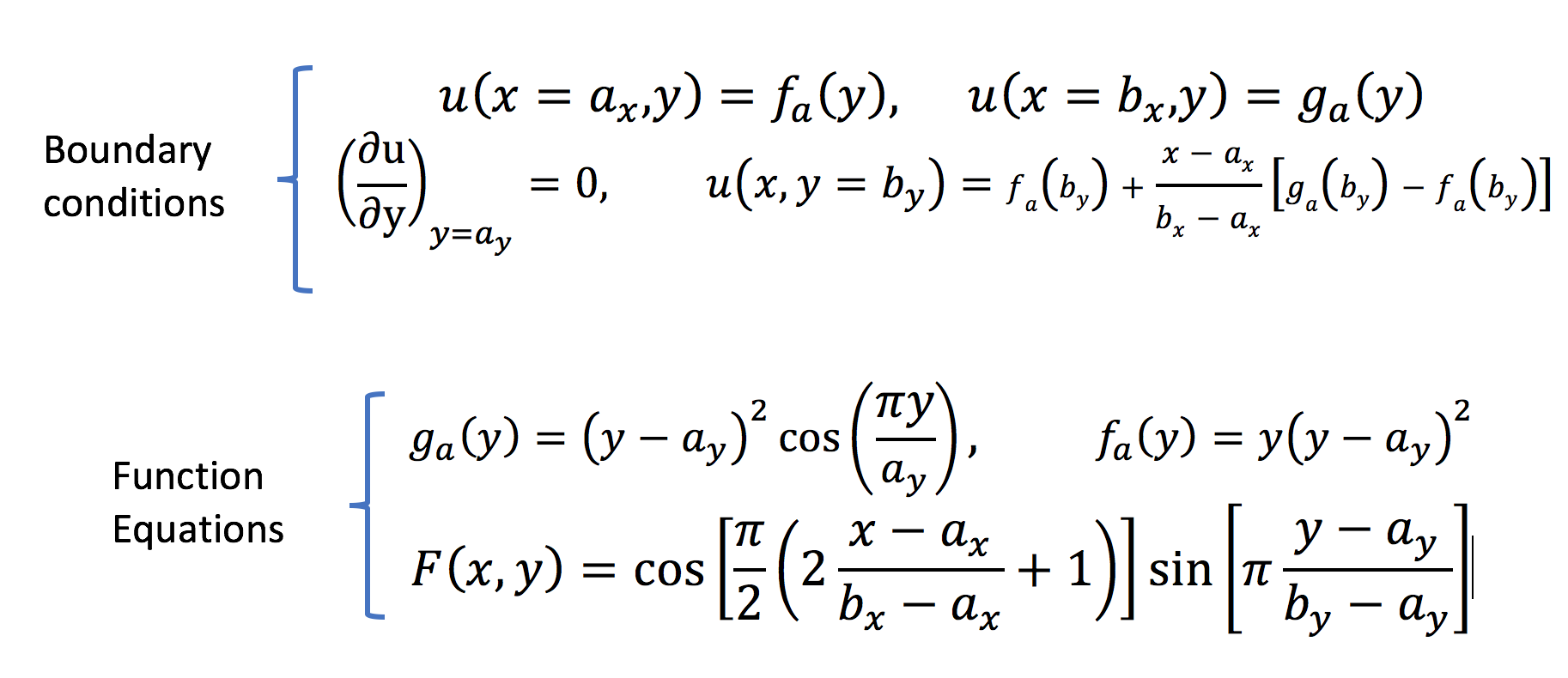


Figure 3- Boundary conditions (left, right, bottom and top respectively) and the equations used to calculate them

The equations from Figure 3 are all dependent of the x and y values in the n by m grid, n being the number of inner nodes in the x direction and m being the nodes in the y direction. The vector function F represents the values at every point in the grid, located on right hand side of Equation 1. This will be even more useful for the discretization part of the Helmholtz equation. The complexity of the problem comes from the only Neumann boundary condition, which is approximated by discretization.

Discretized version of equations:

In order to make the Helmholtz equation suitable for numerical evaluation it is important to begin by discretizing the Equation 1 from Figure 1. There are some steps to be taken to correctly discretize it and make it practical for approximation in MatLab, by using the iterative methods discussed previously. Figure 4 shows the steps taken to discretize the Helmholtz equation.

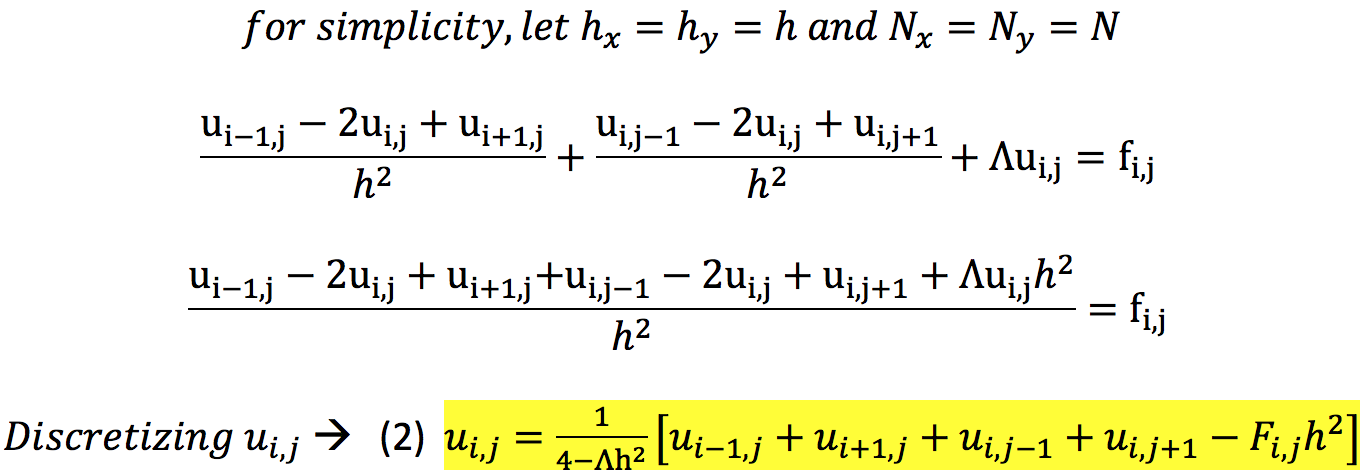
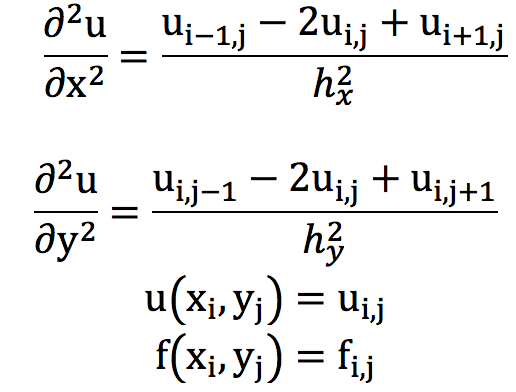


Figure - Approximation of partial derivatives to be able to discretize Helmholtz equation

Figure 4 shows how to break down the PDE into different parts, which is useful to display the different approximations. In this case, the centered difference approximation scheme was used to approximate the partial derivatives of the left side of the equation.

The central difference is chosen over the backwards and forwards difference schemes because the central difference scheme is second order in *h* for a smooth *f* while the backward and forwards schemes are both first order in *h,* they only use an adjacent point for approximation. The benefit of using central difference is that the error of the approximation is reduced by a factor of 4, given that *O(h2)* different from the *O(h)* of the other two. This explains why the reasonably smaller the value of h is, the better the central difference will execute. Further explanation will be provided in the results section, but to give an idea, the bigger the value of nodes N for a grid, the smaller the *h* value thus the more accurate approximation. Another thing to note from Figure 4, is why *hx =hy*. The number of nodes in the x-direction are assumed to be equal to the number of nodes in the y-direction, mainly because the problem statement established equal domains in both directions. For this reason, it is convenient to allow h to be equal since the distance between nodes is the same. The most important part is the highlighted discretized equation in Figure 4. Ui,j is conforms all the values of the inner points of the grid, while the outer values are defined by the boundary conditions From Figure 3. The discretization of Ui,j will help us understand the Gauss-seidel and SOR methods and their application.

The complex part of the project is the implementation of the Neumann condition into the boundaries of the grid. Because the Neumann conditions are not given, it is possible to find the behavior of the boundary by adding “ghost nodes”, that will help us approximate the boundary values as well. Figure 5 shows the discretization of the Neumann boundary on the lower part of the rectangle domain.

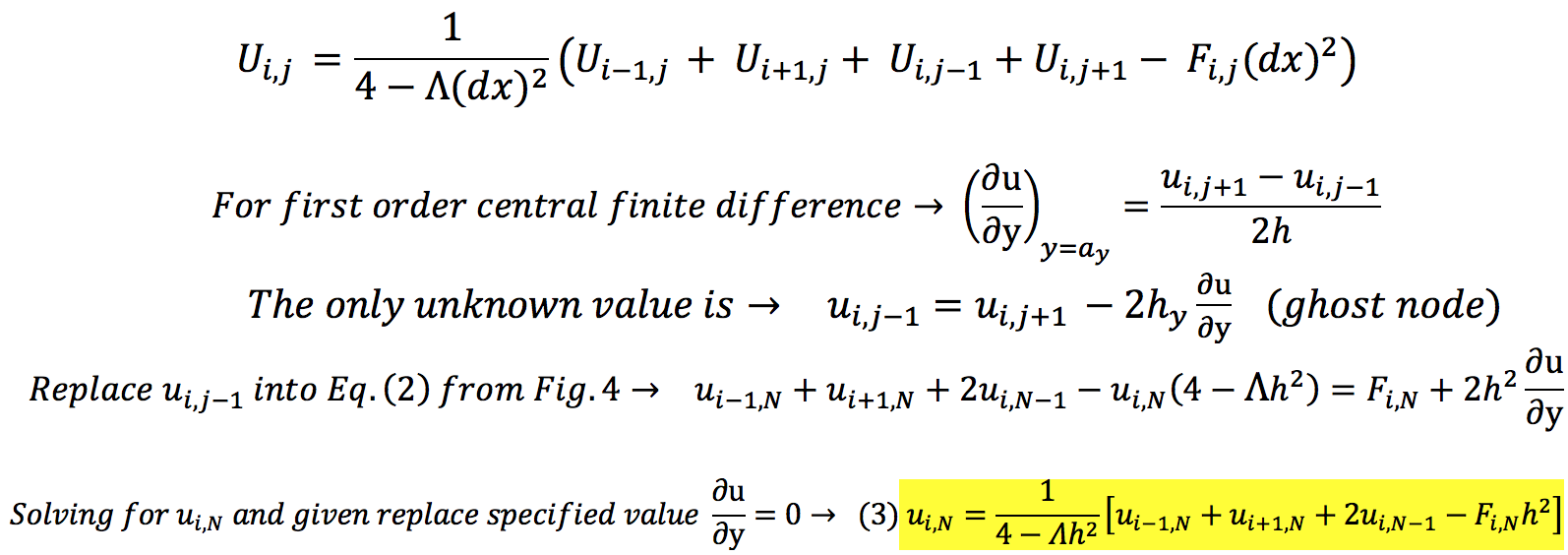


Figure - Discretization for the Neumann boundary condition.

Description of Numerical Methods:

An iterative method is a method that progressively calculates approximations to the solution of a problem. In an iterative method, the same improvement process is repeated over to approximate a solution. After the first iteration, it is expected that the obtained solution is more approximated than the initial one. The process is repeated on this new solution until the most recent result satisfies certain requirements. Unlike the direct methods, in which the process must be finished to have the answer, in the iterative methods the process can be suspended at the end of any iteration, where an approximation to the solution is obtained.

The two numerical methods selected are Gauss-seidel and SOR iteration methods that are used to converge an approximation for the inner nodes values of the prescribed rectangle domain. Gauss-seidel method is generally preferred over the more laborious Jacobi method, which is a reason why it was selected. While Jacobi uses unknown values calculated after the first iteration, Gauss-seidel uses an initial guess to calculate the first iteration and continue like from that determined approximation. The Successive Over-Relaxation method is quite similar to the Gauss-seidel in that it is based on the same principle but differs slightly in arithmetic conditions and also has a better rate of convergence.

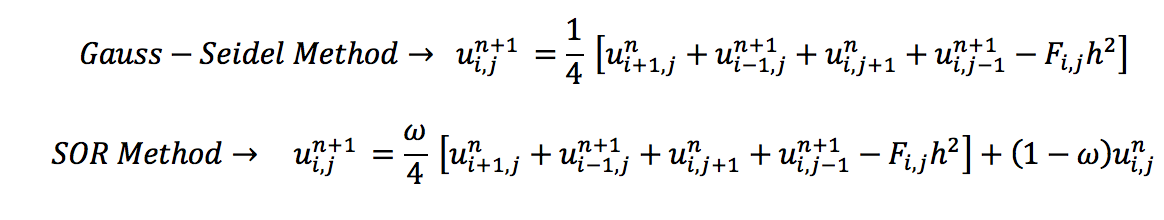


Figure - Two iterative methods selected used accelerate convergence in our discretization of the Helmholtz equation

As seen in Figure 6, the Gauss-seidel method relies on an initial guess to anticipate the value of the next iteration un+1. For the project, Gauss-seidel was applied by creating two different matrices; on that represents the initial guess grid, while the other is the new grid. Once the values for the new grid are approximated from the discretized Equation 2 from Figure 4, the new grid become the initial guess grid used to calculate a new grid and so on. The iterative process ends when the error is small enough to meet the tolerance standards needed. For the purpose of this code, tolerance was set within ranges of 1E-9 and 1E-3, anything smaller than that would just add to the iterations thus time steps needed to converge. Figure 7 shows the small pseudocode applied in both cases, with optimization in mind. Of course some modifications have to be done like adding the ω parameter to the SOR discretization method.

% Do while loop for error convergence

for j = 1:ny-1 % indexing applied for performance optimization

for i = 1:nx-1

% enter discretization code for inner nodes u\_i,j

end

% enter discretization code for the Neumann condition u\_i,N (j in

% terms of rows and columns in MATLAB

end

% END loop

Figure - Pseudocode that fits both Gauss-Seidel and SOR Methods

The SOR method provides a faster solution than its counterpart, but everything depends on the omega ω parameter or over-relaxation factor. The idea behind this over-relaxation method is that if the approximation is a good one for a specific value within the inner grid points, the approximation should move in that good direction for the right solution by a factor of ω. The range that omega lies within has to do with the derivation of the constant. However, it is good enough to choose a value within the range 1<ω<2, usually 1.5, because anything bigger than 2 would make the approximation diverge, while anything smaller than 1 leads to under-relaxation. To correctly evaluate the parameter ω, it is also possible to calculate the optimal value for it by using Equation 4 below.

Technical Specifications

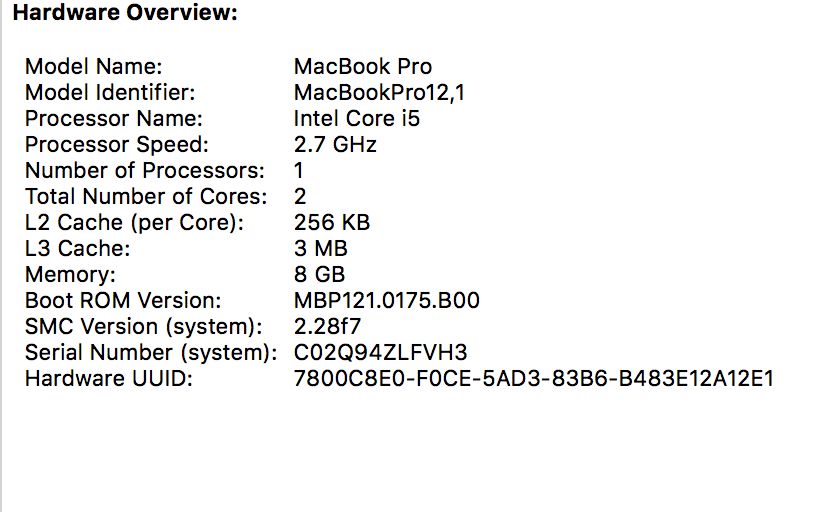
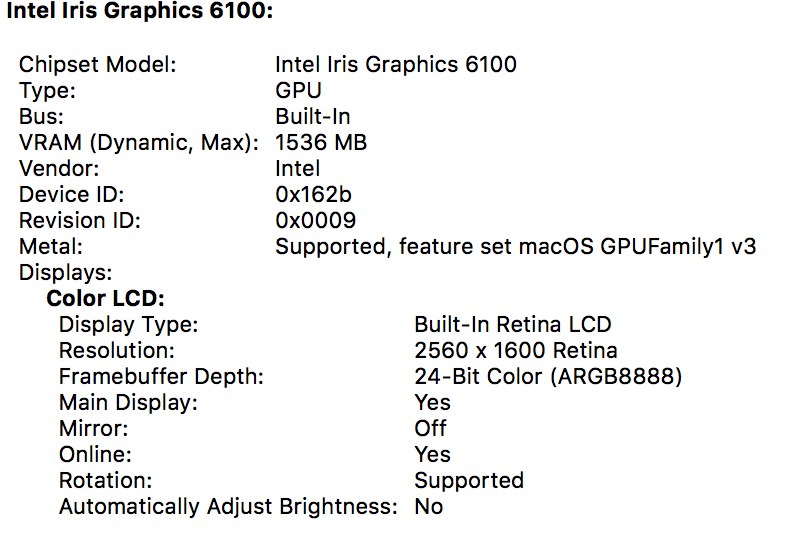
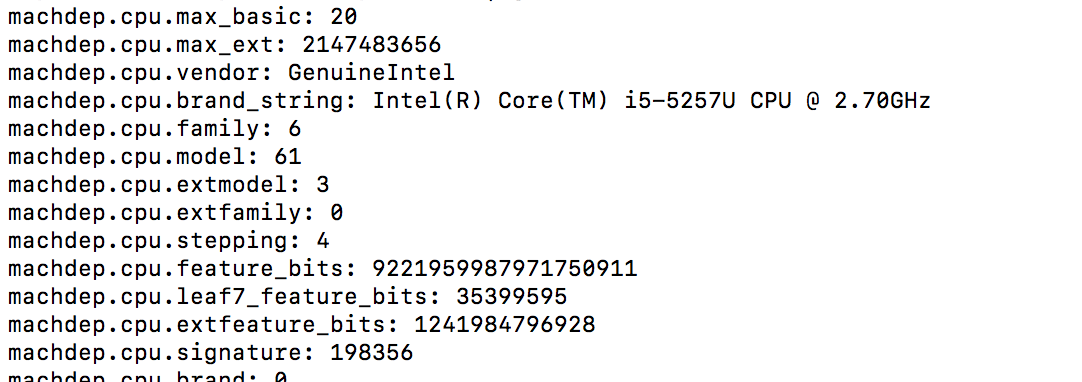


Figure 8- Basic System information obtained through settings

The computer used for the execution of the code is a MacBook Pro 2015 model. Because the OS for this computer is not as detailed when it comes to CPU, the code script **sysctl -a | grep machdep.cpu** provided the necessary specs.



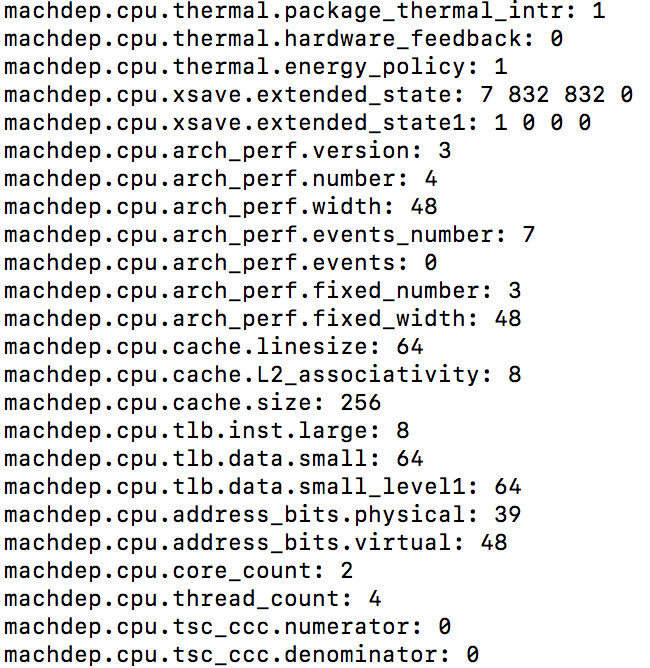


Figure 9- Detailed information on the CPU running in computer

The computer has an Intel Core i5 CPU clocked at 2.70Ghz for a total of 2 cores, that is reliable enough to run multitasking and coding for purposes like this one. From Figure 9, we can tell the cache memory sizes for L1 cache is 4x32KB 8-way set associative and data instruction caches, for L2 unified cache size is 4x256KB also an 8-way set associative caches per core, and L3 unified cache size is 3MB. The ram memory installed in the computer consists of 2 4GB DDR3 RAM memories installed for each core. The GPU is an Intel Iris Graphis 6100, very common for a MacBook pro, with a built in Bus and with a memory size of 1536MB (VRAM).

Results

The final results from the study of the Helmholtz equation is provided below. The model presented in Figure 10 represent specific conditions when F = F(x,y) (the function shown in Figure 3) and Λ=0, essentially the Poisson’s equation. For accuracy of approximations, a medium number of nodes (N=80) was selected to present a smoother 3D Plot. Because the order of accuracy for the second order equation is *O(h2),* the error is dependent and directly proportional to step size *h*, thus also rely on the number of nodes. This error converged into a tolerance of 1E-9 in order to make the approximations as accurate as possible. This however increased the amount of computer processing time, which can be time consuming.

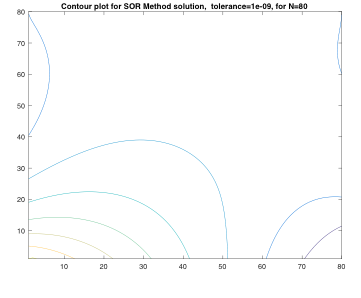
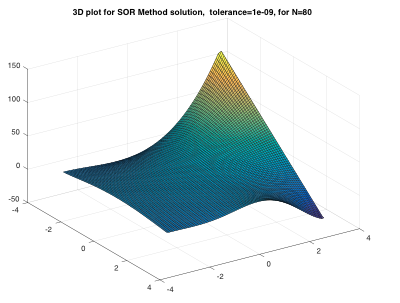
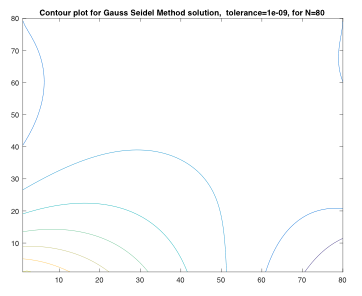
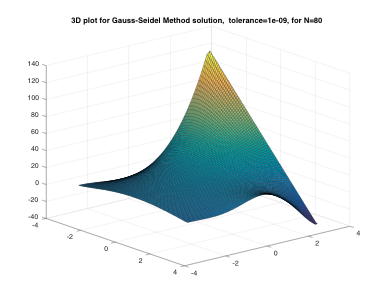


Figure - 3D and contour plot solutions for Helmholtz Equation using Gauss-Seidel and SOR methods when F =F(x,y) and Λ=0 (Poisson’s Equation)

The plotted results in Figure 10 show the similarity between the Gauss-seidel and the SOR methods not only in the way we approach to each discretization scheme, but also on the results they display. Because SOR is based on Gauss-Seidel, the solution for both are really similar. This particular case shows the implementation of the boundary conditions on the sides, while the contour plot also serves to picture the values of u(x,y) in the grid created. Both boundary conditions and contour plot displayed share similarities in both Gauss-seidel and SOR plots because they are both solutions converging from the same boundary conditions. The peak u values are on the corner of the boundary conditions of the top and left side of the grid domain, which is why the simulation peaks at these points given the boundary condition’s linear and parabolic behavior of the functions. The same can be seen in the contour plots where the peak values of u are on the lower-left corner in yellow. The Neumann conditions behavior is also better represented in the surface plots. The edge that doesn’t curve much is the Neumann condition and it’s behavior comes from the discetization, as shown in Figure 5. Because the partial derivative is equal to 0, the values of the boundary do not change much with each iteration. In fact, they are close to zero most of the time.

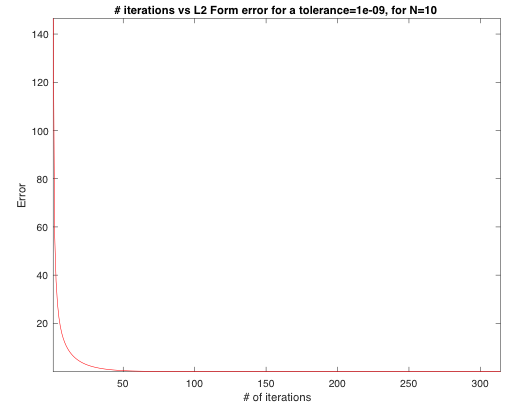


Figure 11- Plot of iterations vs Error propagated using Gauss-Seidel Method

Figure 9 serves as verification of the code, using the L2 form of error per iteration. For convenience purposes, the Gauss-Seidel method is provided only, but it gives us an idea of the behavior of the error. The expectation of the iterative methods is that the error occurring consistently throughout the iterative process must be used as a means of understanding whether the approximation is valid or not. The plot in Figure 11 shows how the L2 error is decreasing rapidly as the number of iterations increase, representing a function decay of 1/x. As the error converges to 0 (or 1E-9 to be exact), the number the curve now becomes a constant line and the approximation gets closer to the desired value.

Table 1-Grid independence Study for GS and SOR schemes @ tolerance=1E-9, F=F(x,y) and Λ=0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gauss-Seidel Method | | SOR Method | | |
| Number of Nodes (N) | # of iterations taken | U Avg for GS | # of iterations taken | U Avg for SOR |
| 10 | 314 | 16.4307 | 307 | 16.4307 |
| 20 | 1365 | 16.5963 | 1123 | 16.5963 |
| 40 | 5658 | 16.6878 | 2769 | 16.6878 |
| 80 | 22867 | 16.7366 | 6046 | 16.7366 |
| 160 | 91226 | 16.7619 | 12647 | 16.7619 |

The other part of the grid independent study was to focus on the iterations per node instead of the error propagation. From Table 1, it is possible to note the time it took for each method to compute an approximated solution for the problem. The Gauss Seidel method is clearly taking more time to finish the approximations. Even though the number of iterations at N=10 is pretty similar, only 7 iterations apart, the difference between both methods becomes more apparent as we increase the number of N. A plot with the information from Table 1 is presented below.

Figure 12- Plot of n vs time steps

Figure 12 is essentially showing how the number of iterations for the Gauss-seidel is increasing at a factor of 2 in relation to the number of iterations of the SOR method, as the number of iterations rise. While the SOR plot is relatively consistent and linear, the Gauss-seidel curves upwards showing that the SOR method is a lot more reliable in terms of performance for larger grid sizes, taking less time steps while still being accurate.

As a final task to the project, the problem statement asks to also output results for the case where F=0 and Λ=0. Figure 13 shows a 3D plot and contour plot of the specified case with N=80 and toleration=1E-9, the same conditions from Figure 10 for comparison purposes.

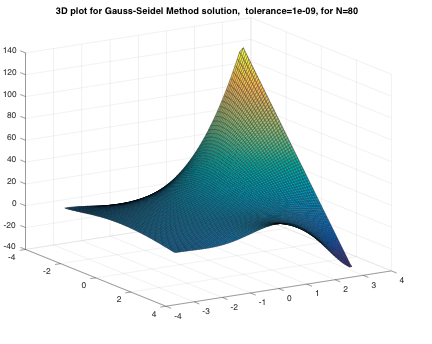
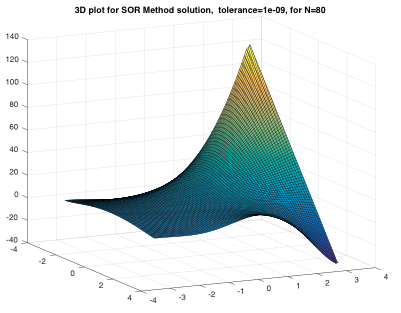
Both 3D plots are consistent with the plots from Figure 10, the only difference now being that the value of F throughout the grid is zero, therefore it does not affect the previous plot’s shape.

Figure - 3D and contour plot solutions for Helmholtz Equation using Gauss-Seidel and SOR methods when F =0 and Λ=0 (Laplace’s Equation)

References:

<https://math.stackexchange.com/questions/2120946/why-is-central-difference-preferred-over-backward-and-forward-difference-in-conv>

http://www.math.umbc.edu/~kogan/technical\_papers/2007/Yang\_Gobbert.pdf

**Note: The code will be uploaded to blackboard for grading but can also be access through the github website for any other concerns at https://github.com/30s30.**