*MECE 5397*

*Scientific Computing for Mechanical Engineers*

**Final Project**

**Exploration of Several Iterative Finite Difference Methods for Solving the 2D Helmholtz Equation**

Michael Greer

*University of Houston*

**Several methods for solving the 2 dimensional Helmholtz Partial Differential Equation with the given boundary conditions and forcing function are presented and explored. The methods discussed here are the Jacobi Method, the Gauss-Seidel Method, and the Method of Successive Over Relaxation (SOR). Each method is implemented to solve the Helmholtz PDE using MATLAB. The methods presented here are related to each other in that each one is a numerical method for solving linear equations without requiring the construction and classical Gaussian resolution of large matrices. The methods are designed to find solutions through iteration; the solutions yielded are not exact solutions to the linear systems but approximations found from conducting repeated operations on all points on the solution grid. The inexact nature of these methods yield solutions with some amount of associated error. The error in these solutions are minimized through large numbers of iterations and grid convergence.**

1. **Introduction**

Iterative Methods for solving Partial Differential Equations are extremely convenient because of the ease with which they can solve through large data sets without creating a huge strain on the dynamic memory of the computer. Because the accuracy of the method largely depends on the number of iterations the computation goes through, higher accuracy can be achieved by simply allowing the method to compute for larger amount of time.

To solve the Helmholtz Equation given in this problem, several iterative methods are used and compared for accuracy and speed of convergence. To accomplish this, MATLAB functions are written that implements each method and are called in a series of benchmark scripts. The scripts generate plots that depict how the accuracy of the error changes with grid size and MATLAB timing functions are called to evaluate the performance of the method functions. Some optimization techniques are used to increase the speed of each method.

1. **Problem Statement**

The mathematical statement of the problem for this project is as follows:

Write a computer code to solve the two-dimensional Helmholtz Equation

On a rectangle defined by the boundaries

Using the boundary conditions

Where

Subject to the forcing function

1. **Problem Discretization**

The first step in computing any differential equation using numerical methods to choose a discretization for the differential terms of the equation. To find these discretizations, the Taylor series is employed and the derivate term is solved for to find an approximate solution for the derivate. The 2D Helmholtz Equation has two second derivate terms. The centered difference approximation for second derivates is used to discretize these terms because of its accuracy and high order of error (O(h2)).

The full discretization of the 2D Helmholtz Equation is

Solving for uij

Equation (12) is the form of the discretized equation used in the iterative methods to compute the solution at all points i, j.

1. **Discussion of Methods**

The methods used to solve the given problem are the Jacobi Method, the Gauss-Seidel Method, and the Method of Successive Over Relaxation (SOR). These methods were developed from the classic Gaussian Elimination Method for solving linear systems. In particular, these methods were developed to solve systems where only the central diagonal elements are populated such as tri- or penta-diagonal matrices.

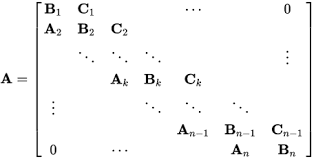


Figure : A Tridiagonal Matrix

In the solution of linear systems that take the form of a tridiagonal, pentadiagonal, or other diagonal matrix, there is never any need to define the actual system as a matrix. Instead, the solution of the system is determined by solving a set of algebraic equations that represent the rows of the diagonal matrix.

In the context of this problem, the mathematical descriptions of these methods are in the form of the discretized equation of the 2D Helmholtz equation (Eq. 12). Each method uses this equation to compute the value of u at point (i , j). This value is computed using the values surrounding this point; namely, using the points to the north, east, south, and west of point (i , j). This means that each point is dependent on its neighboring points, so points that start out at zero and are surrounded by more zeroes will remain zero until one of the surrounding points become nonzero. This will gradually happen as the nonzero boundaries work to populate the plain initially at zero. This initial population of values takes a number of iterations to complete, hence it being an “iterative method”. If the discretization in stable, the value of the points on the grid will converge on the solution for the original partial differential equation.

Each method was written as MATLAB functions for easy use in script files. The overall code implementation is mostly the same for each method. For example, the boundary conditions are

1. **The Jacobi Method**

The Jacobi method uses Equation (12) by computing each point on the grid using the surrounding values from the previous iteration. Once each point is computed in this way, it starts over, using the grid points it just finished computing to compute the next set of grid points. This method can be represented by the equation

Where k is the current iteration, and k – 1 is the previous iteration.

1. **The Gauss-Seidel Method**

The Gauss-Seidel Method is very similar to the Jacobi Method. The difference between the two methods is the points used in the computation of the current point. Where the Jacobi method uses only the grid points from the previous iteration for the computation, the Gauss-Seidel Method uses any previously computed point. This means that the computation can utilize a point that was computed during the current iteration. This method can be represented using the equation

As shown in the equation, the points to the west and south of the current point were computed during iteration k. This small change can lead to considerable increases in computational performance.

1. **The Method of Successive Over Relaxation**

The SOR method is a way to increase the speed of convergence by adding the current iteration points to the previous iteration points multiplied by a “relaxation factor”. This method can be used as an extension of either of the previous methods by adding one more step in the form of the equation

Where λ is the overrelaxation factor.

1. **Computer Specifications**

The software written to compute the solution to the given problem using the described methods was written to be run on MATLAB version R2016b on Windows OS. The computations were done on two different computers. The specifications for each computer is listed below:

|  |  |  |
| --- | --- | --- |
|  | System 1 | System 2 |
| Model | ASUS N501VW Laptop |  |
| CPU Model | Intel Core i7-6700HQ |  |
| Number of Cores | 4 |  |
| Number of Threads | 8 |  |
| Max CPU Clock Speed | 2.60 GHz |  |
| L1 Cache | 256 KB |  |
| L2 Cache | 1024 KB |  |
| L3 Cache | 6144 KB |  |
| DRAM Size | 16 GB |  |

Each computer can be used simultaneously to run or write code using Git version controlling. Git repositories were created on both computers and connected via a Github server repository. Files were then pushed and pulled to and from this server to quickly update the repositories on both systems.

1. **Results**

Reslults

1. **Conclusion**

Conclusion