MECE 5397 Scientific Computing for Mechanical Engineers

Poisson Equation: Project A – APc1-3

Seth Reyes

1223631

Class session: M,W 1:00pm - 2:30 pm

Poisson Equation: Project A – APc1-3

The objective of the final project is to discretize the Poisson Equation given varying boundary conditions and varying domains of interest from person to person using two methods. The boundary conditions are a mixture of derelict and Neumann boundary conditions. The problem statement assigned is as follows.

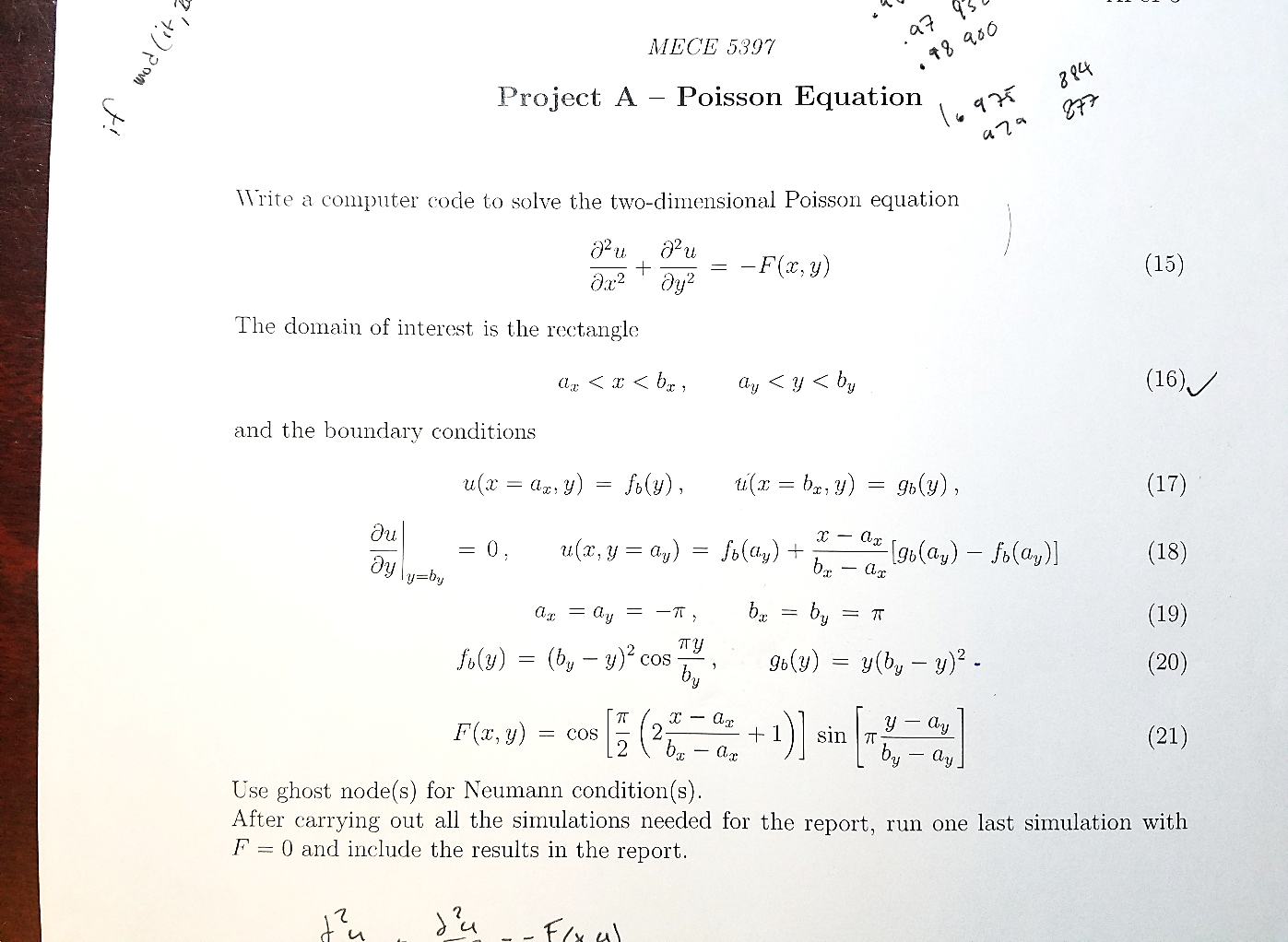


Figure 1: Mathematical Problem Statement

As provided in Figure 1, the domain of interest lines within a square starting on the X-axis at -π and ending at π, and starting on the Y-axis at -π and ending at π. The three derelict boundary conditions are as follows: the left side of the square where x = -π, the boundary condition will be governed by the function which is a function of only y, the right side of the square where x = π, the boundary condition will be governed by the function which is also a function of only y, and finally, the bottom boundary condition is given to be a function of x and y, where y held constant, for the value of y is -π. The fourth boundary condition is a Neumann boundary condition which must be solved iteratively in conjunction with the middle of the domain.

The first step to solving the Poisson Equation is to discretize the second order differential equations. Then, the similar terms are combined to make the discretized equation neater. Finally, was solved for, and the order of error for this discretization is .

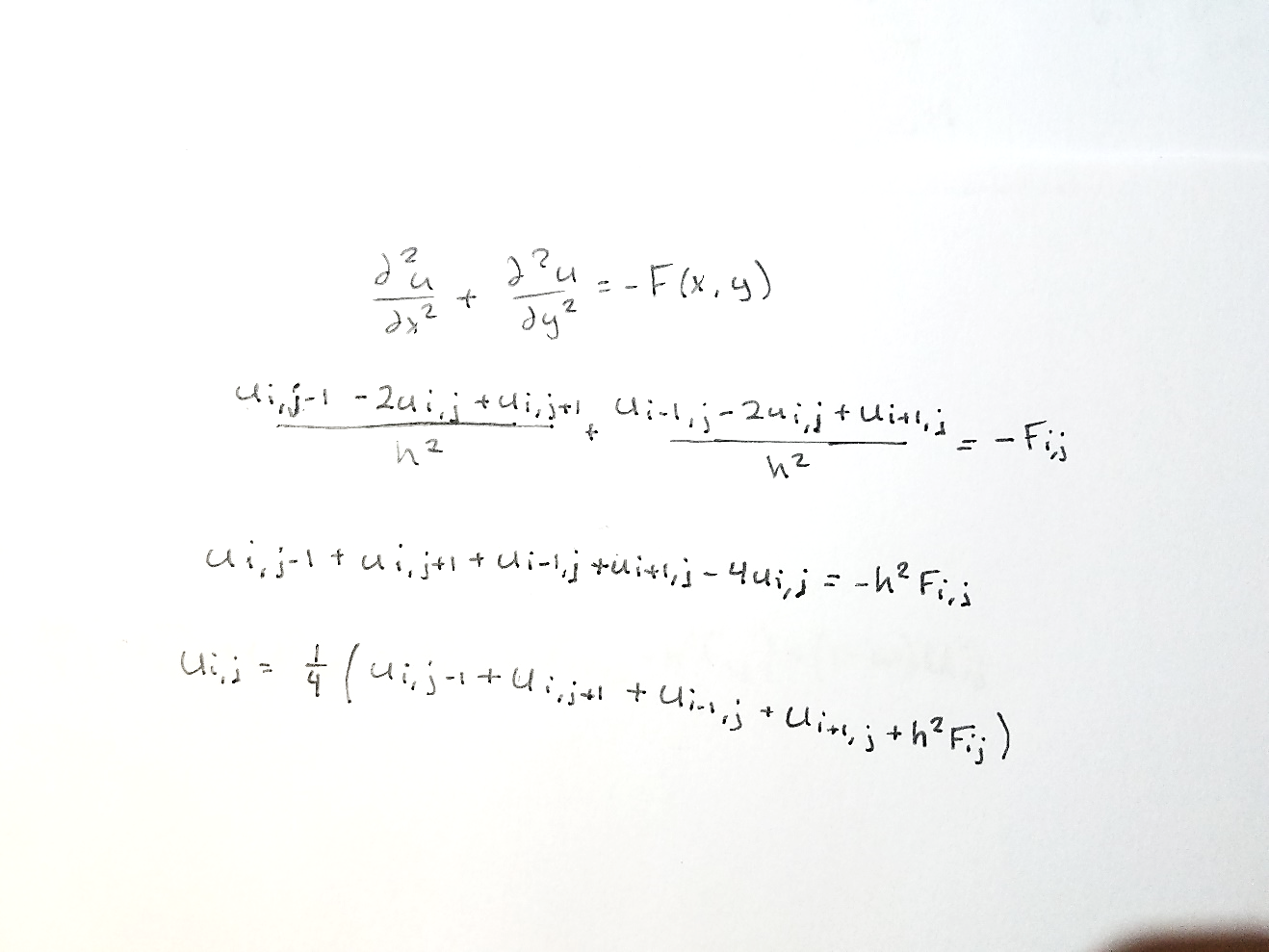


Figure 2: Discretization of the Poisson Equation

This is not enough to plug into the code to solve the Poisson Equation. Neumann boundary condition needs to be solved using a “Ghost Node”, or a node that does not exist within the domain.

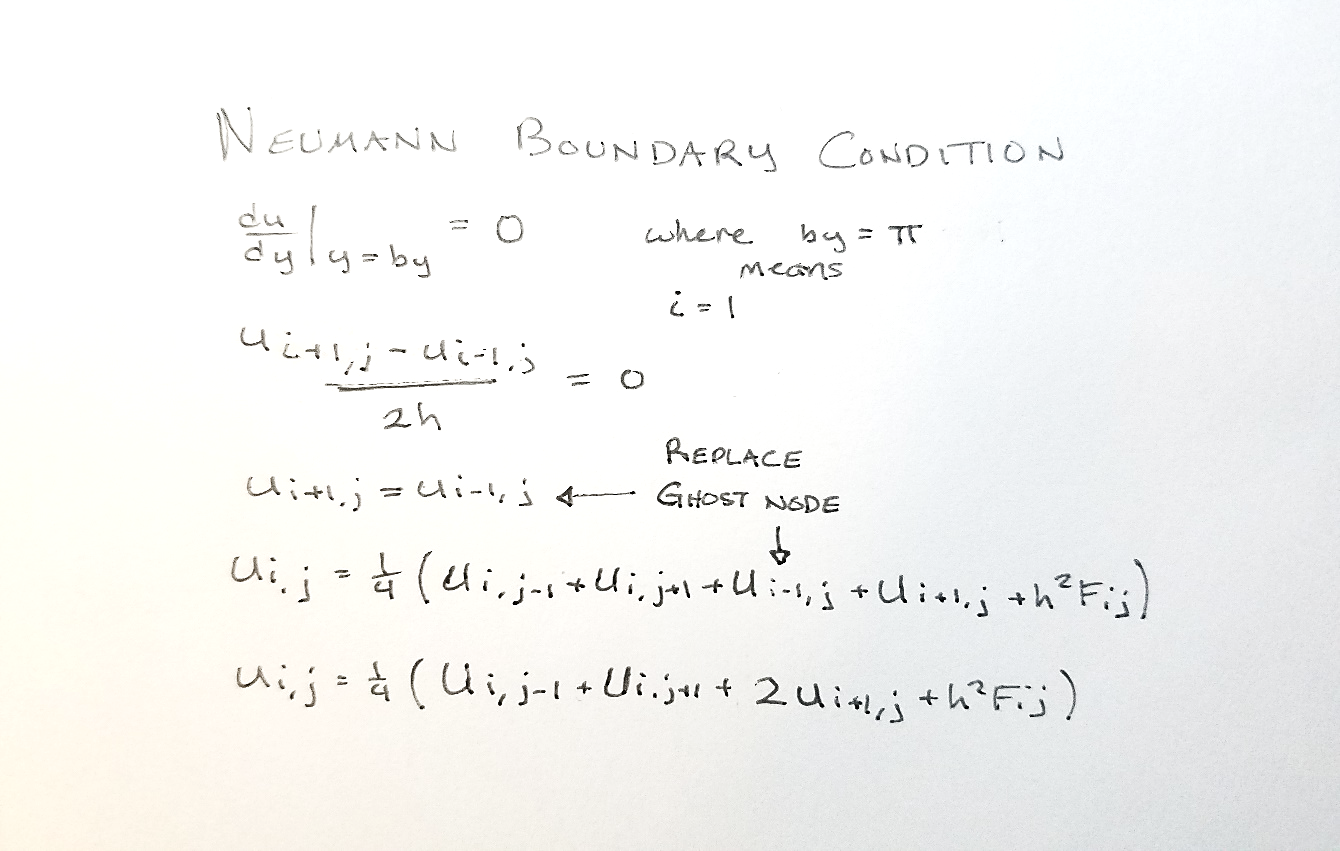


Figure 3: Discretization of Neumann Boundary Condition

In this problem, the Neumann boundary condition is at the top of the square domain where y = π. To solve the Neumann boundary condition, it first must be discretized. Once discretized, the “Ghost Node” can be solved for since the right side of the equation is known. Going back to our discretized Poisson Equation, we can plug in the equivalent to our ghost node into the discretized Poisson Equation. After that, we get a special discretization in our domain that only occurs at the top boundary.

Finally, we can modify our discretization of the Poisson Equation to find a solution. The first method chosen to find a solution is the Gauss-Seidel method. This method is already very similar to discretization of the Poisson Equation that we found with a few exceptions. As we are solving the equation iteratively from the top left of the domain and working our way to the bottom right of the domain, we find that for points within the boundary nodes, we have already calculated the node to the left and the node to the top of our subject node.

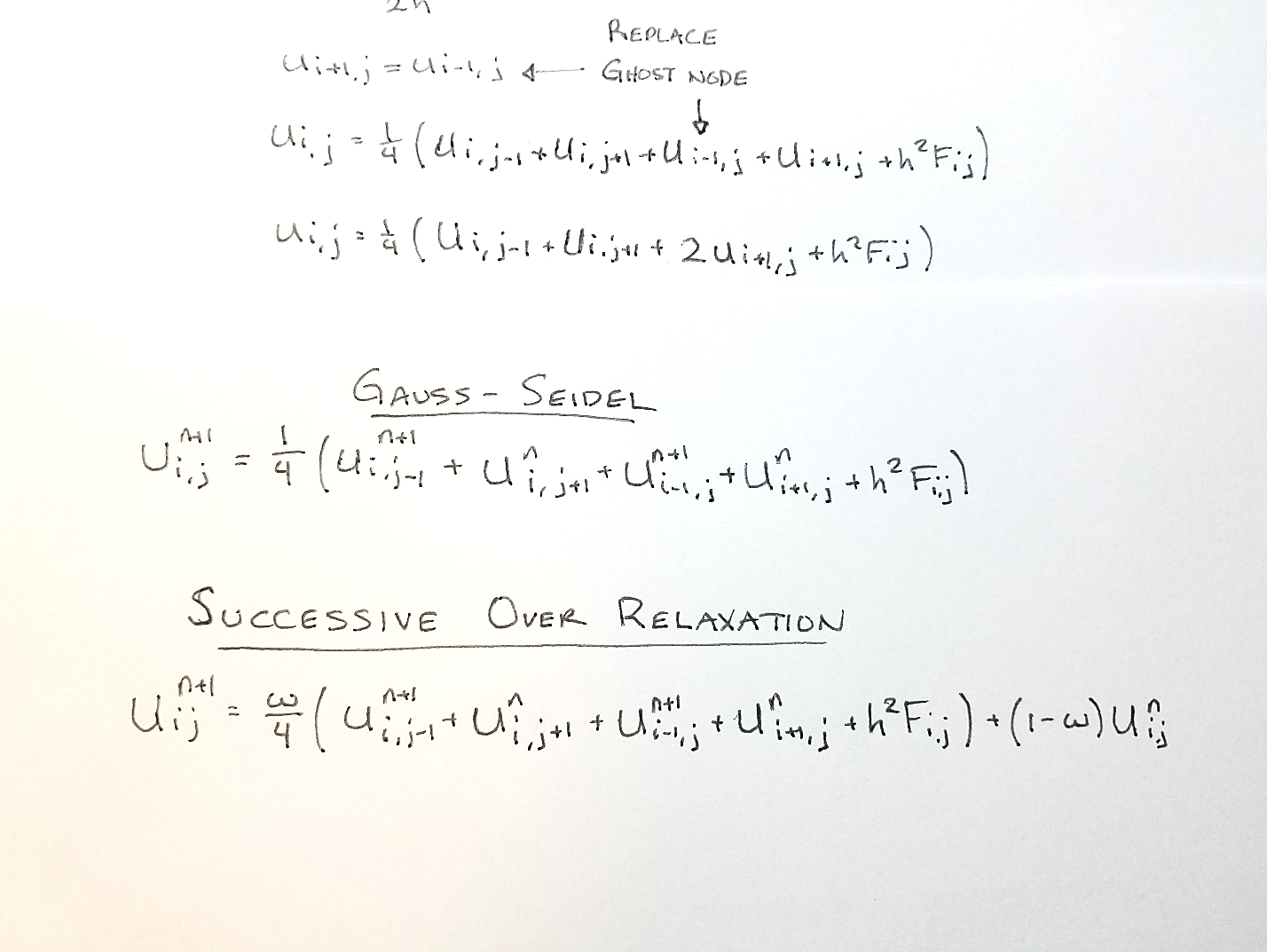


Figure 4: Gauss-Seidel Iterative Method

As seen in Figure 4, the new value for uses the new value above it, the new value to the left of it, previous value to the right of it, and previous value under it. The previous values are used because the new values at the nodes have not been calculated yet since we are calculating notes from top left to bottom right.

The second method of choice is a modified version of the Gauss-Seidel. This method is called the Successive Over Relaxation, or SOR for short.

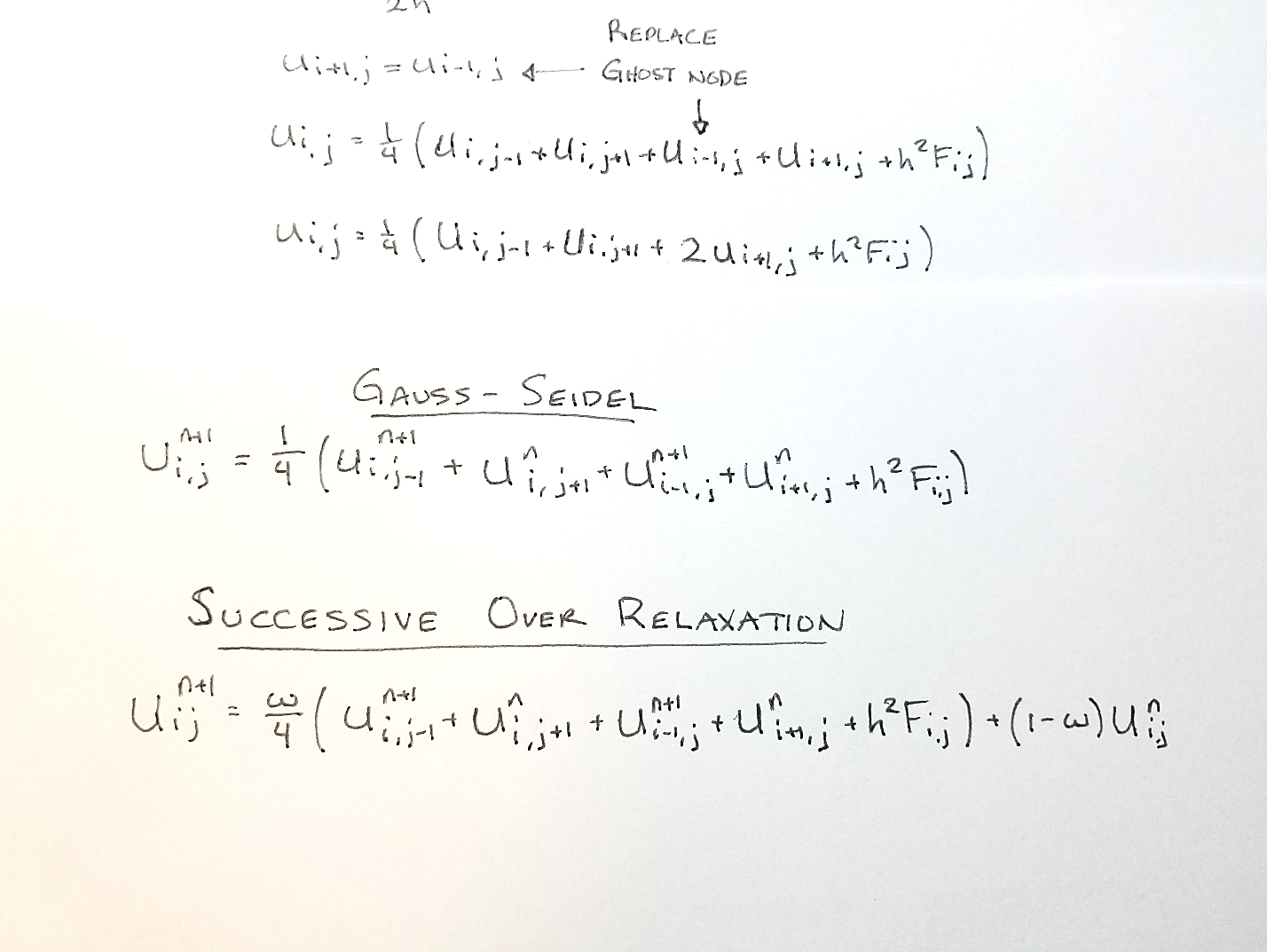


Figure 5: Successive Over Relaxation Method

It works very similarly to the Gauss-Seidel method except it has a relaxation term ‘w’ which improves the iterative solution for . Because of that, SOR converges faster than Gauss-Seidel. Notice if w=1, the equation simplifies the Gauss-Seidel method. W is carefully chosen between 1 and 2, and the lowest number of iterations do not correspond for a constant w term. For each different number of nodes, there Is a different “best” value for w.

Generally, the w selected in our code that gives wonderful results that allows the SOR method converge much faster than the Gauss-Seidel method is as follows.

For both, Gauss-Seidel and SOR method, they converge and it is verified in the code. Also in the code it has check pointing, it times each method, and it counts the number of iterations it has. Also included in the code is the capability to watch the grid converge.

The specifications of the computer used to run the code:

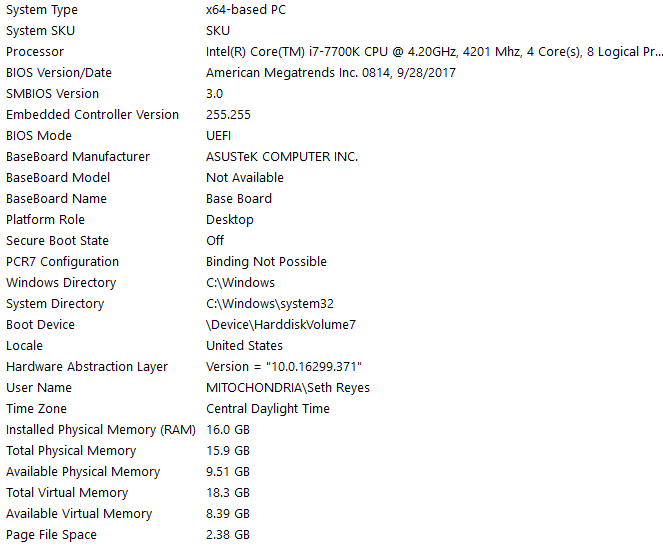


Figure 6: System Information

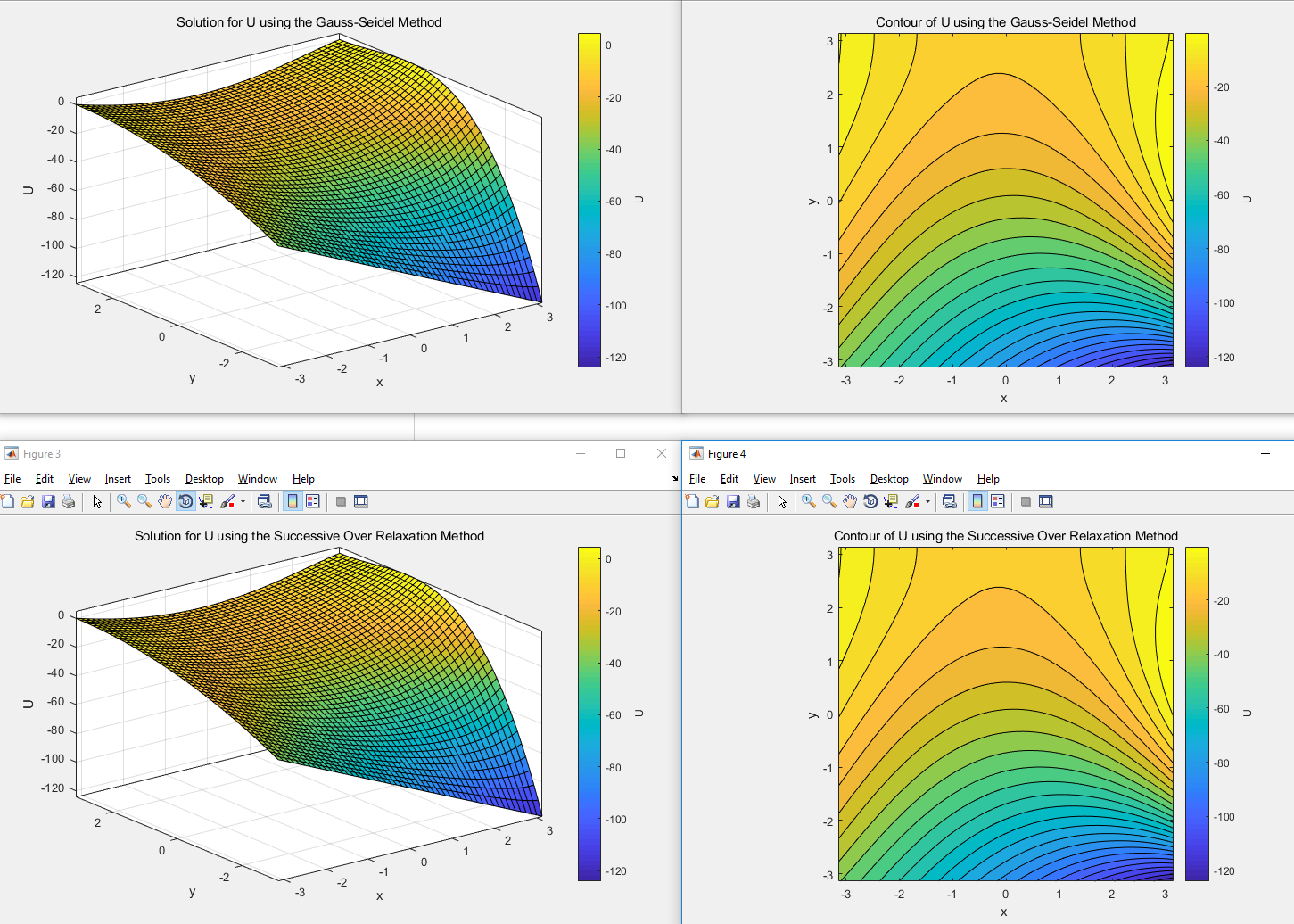
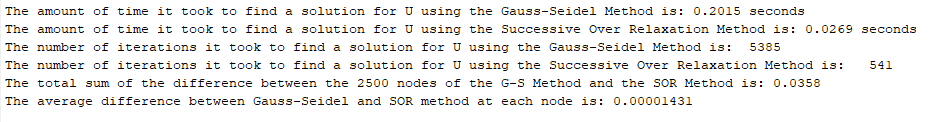


Figure 8: Grid Convergence and Contours for N=50 and Convergence Error = 10^-7



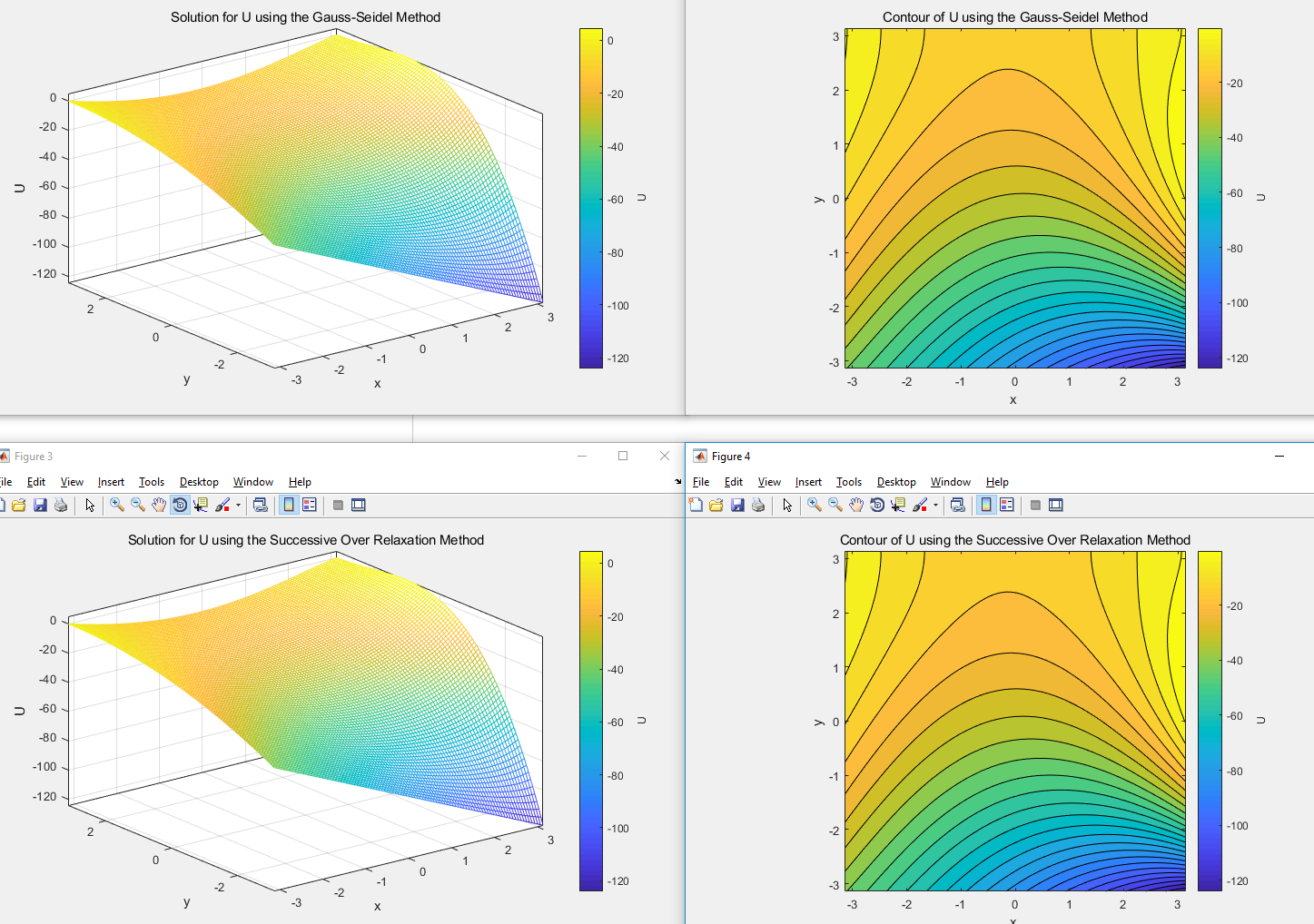
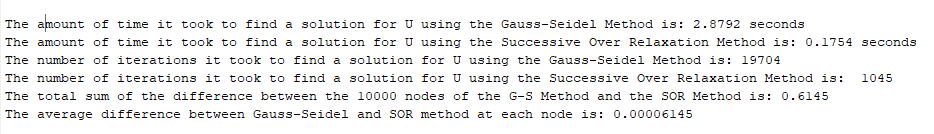


Figure 8: Grid Convergence and Contours for N=100 and Error = 10^-7



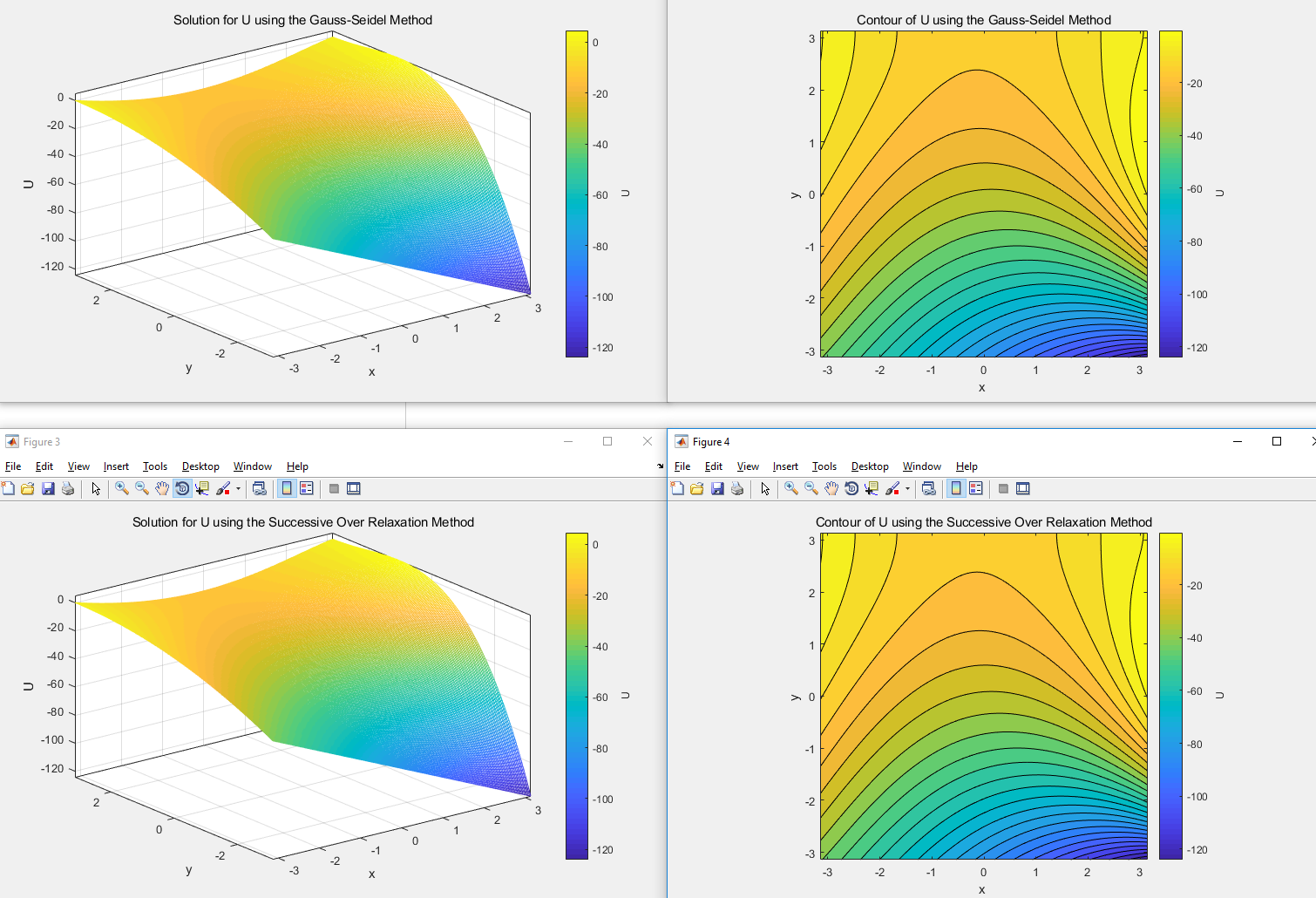


Figure 9: Grid Convergence and Contours for N=200 and Error = 10^-7

