**Project**

**Poisson Equation**

**Rutvik Mehta**

**Scientific Computing for Mechanical Engineers (MECE 5397)**

# 

# Abstract

The primary task of this report is to showcase the results obtained by solving the Poisson Equation. The report includes a step by step process which goes through and discusses the mathematical equation with discretization, numerical methods, results obtained and the validation process. The purpose of this project is to demonstrate how to solve a PDE equation using numerical methods on a computer program to obtain accurate results by using simple discretization formulas. This was done by two methods, the Gauss-Seidel and the SOR (Successive Over Relaxation) methods. Gauss Seidel was chosen as one of the methods since it is a relatively straightforward yet powerful method to implement. The second is the SOR, a more efficient method then the Gauss-Seidel method since it involves a factor, , which increases the rate of convergence.

# Discretization of Mathematical Problem

The mathematical PDE I have been assigned is the Poisson Equation. The equation is shown below:

The above equation models the problem for various situations lending its usage to many engineering problems. Due to its wide utilization, it is crucial to understand how to discretize the equation for using in computational purposes. A possible discretization choice, of a second order error, is:

In this case, h is the same for both x and y, which leads to the simplification to:

= 0

Now solve for ui,j which comes out to be:

Therefore, the above discretization can be used during numerical computations and the same was for the simulations in this report.

# Description of Numerical Method

As mentioned above, the Gauss-Seidel and the SOR method were both utilized for the Poisson Equation. A brief description of each can be found below.

The Gauss-Seidel is an upgrade from the conventional Jacobi Method, where updated values from the current time step are taken for the ui-1,j and ui,j-1 terms as opposed to the taking them from the previous time step. This reduces the time of convergence since half the values are updated.

The SOR, similarly, builds on the Gauss Seidel method, where a coefficient is introduced in the form of ω, which acts as an multiplier, which leads to a faster convergence than the Gauss Seidel. To achieve this, the Gauss Seidel pseudocode is multiplied by ω and then add (1-ω)\*(ui,j). The choice of ω is between 1 and 2 since it is over-relaxation.

The pseudocode below is for Gauss Seidel.

%% USING GAUSS SEIDEL METHOD

% PART A

clear all,clc

Ax = -pi;

Bx = pi;

Ay = -pi;

By = pi;

epsilon = 0.1;

N = 100; % Number of points; N+1 discretizations

h = (Bx-Ax)/(N+2); % Since the discretization for x and y are equal, h is used.

x = zeros(1,(N+2));

y = zeros(1,(N+2));

x(1,1) = Ax;

y(1,1) = Ay;

for a = 2:(N+2)

x(1,a) = Ax+(a\*h); % Grid being made in the for loop.

y(1,a) = Ay+(a\*h);

end

counter = 1;

Error = 10;

U = zeros(N+2,N+2); % This is the initial guess matrix.

C = zeros(N+2,N+2); % This is the matrix that will copy the previous time frame matrix

for i = 1:(N+2)

C(1,i) = cos(pi\*(x(i)-Ax))\*cosh(Bx-x(i));

%C(1,i) = 1+x(i)^2+2\*y(i)^2;

C((N+2),i) = ((x(i)-Ax))^2\*sin(pi\*(x(i)-Ax)/(2\*(Bx-Ax)));

%C(N+2,i) = 1+x(i)^2+2\*y(i)^2;

end

while Error>epsilon

for i = 2:(N+1)

for j = 2:(N+1)

F(i,j) = cos(pi/2.\*(2.\*((x(j)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i)-Ay)/(By-Ay)));

%F(i,j) = -6;

C(i,j) = 0.25\*(C(i-1,j)+C(i,j-1)+U(i,j+1)+U(i+1,j)-(F(i,j)\*h^2));

end

F(i,1) = cos(pi/2.\*(2.\*((x(1)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i))-Ay)/(By-Ay));

%F(i,1) = -6;

C(i,1) = 0.25\*(C(i-1,1)+C(i,2)+U(i,2)+U(i+1,1)-(F(i,1)\*h^2));

F(i,N+2) = cos(pi/2.\*(2.\*((x(N+2)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i))-Ay)/(By-Ay));

%F(i,N+2)=-6;

C(i,(N+2)) = 0.25\*(C(i-1,N+2)+C(i,N+1)+U(i,N+1)+U(i+1,N+2)-(F(i,N+2)\*h^2));

end

Err = max((abs(C-U))./max(U));

Error = max(Err);

U = C;

counter = counter+1;

end

figure(1)

surf(x,y,U)

title(['Solution of Poisson Equation Using Gauss-Seidel Method - Iteration ',num2str(counter)]);

xlabel('X','Fontsize',14)

ylabel('Y','Fontsize',14)

zlabel('U','Fontsize',14)

% PART B - F = 0

for i = 1:(N+2)

C1(1,i) = cos(pi\*(x(i)-Ax))\*cosh(Bx-x(i));

C1((N+2),i) = ((x(i)-Ax))^2\*sin(pi\*(x(i)-Ax)/(2\*(Bx-Ax)));

end

Error1 = 10;

counter1 = 0;

U1 = zeros(N+2,N+2);

while Error1>epsilon

for i = 2:(N+1)

for j = 2:(N+1)

C1(i,j) = 0.25\*(C1(i-1,j)+C1(i,j-1)+U1(i,j+1)+U1(i+1,j));

end

C1(i,1) = 0.25\*(C1(i-1,1)+C1(i,2)+U1(i,2)+U1(i+1,1));

C1(i,(N+2)) = 0.25\*(C1(i-1,N+2)+C1(i,N+1)+U1(i,N+1)+U1(i+1,N+2));

end

Err1 = max(abs(C1-U1))./max(U1);

Error1 = max(Err1);

U1 = C1;

counter1 = counter1+1;

end

figure(2)

surf(x,y,U1)

title(['Solution of Poisson Equation Using Gauss-Seidel Method (F=0) - Iteration ',num2str(counter1)]);

xlabel('X','Fontsize',14)

ylabel('Y','Fontsize',14)

zlabel('U','Fontsize',14)

The pseudocode below is for SOR:

%% SOR

clear all,clc

Ax = -pi;

Bx = pi;

Ay = -pi;

By = pi;

N = 15; % Number of points; N+1 discretizations % ONLY WORKS FOR UP TO 4.

h = (Bx-Ax)/(N+2); % Since the discretization for x and y are equal, h is used.

w = 1.3;

x = zeros(1,(N+2));

y = zeros(1,(N+2));

x(1,1) = Ax;

y(1,1) = Ay;

for a = 2:(N+2)

x(1,a) = Ax+(a\*h); % Grid being made in the for loop.

y(1,a) = Ay+(a\*h);

end

counter = 1;

epsilon = 0.1;

Error = 10;

U = ones(N+2,N+2); % This is the initial guess matrix.

C = ones(N+2,N+2); % This is the matrix that will copy the previous time frame matrix

for i = 1:(N+2)

C(1,i) = cos(pi\*(x(i)-Ax))\*cosh(Bx-x(i));

C((N+2),i) = (x(i)-Ax)^2\*sin(pi\*(x(i)-Ax)/(2\*(Bx-Ax)));

end

while Error>epsilon

for i = 2:(N+1)

for j = 2:(N+1)

F(i,j) = cos(pi/2.\*(2.\*((x(j)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i)-Ay)/(By-Ay)));

C(i,j) = 0.25\*w\*(C(i-1,j)+C(i,j-1)+U(i,j+1)+U(i+1,j)-(F(i,j)\*h^2))+((1-w)\*U(i,j));

end

F(i,1) = cos(pi/2.\*(2.\*((x(1)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i))-Ay)/(By-Ay));

C(i,1) = 0.25\*w\*(C(i-1,1)+C(i,2)+U(i,2)+U(i+1,1)-(F(i,1)\*h^2))+((1-w)\*U(i,j));

F(i,N+2) = cos(pi/2.\*(2.\*((x(N+2)-Ax)/(Bx-Ax))+1)).\*sin(pi.\*((y(i))-Ay)/(By-Ay));

C(i,(N+2)) = 0.25\*w\*(C(i-1,N+2)+C(i,N+1)+U(i,N+1)+U(i+1,N+2)-(h^2\*F(i,N+2)))+((1-w)\*U(i,j));

end

Err =max(abs(C-U))./max(U);

Error = max(Err);

U = C;

counter = counter+1;

end

figure(1)

surf(x,y,U)

title(['Solution of Poisson Equation Using SOR - Iteration ',num2str(counter)]);

xlabel('X','Fontsize',14)

ylabel('Y','Fontsize',14)

zlabel('U','Fontsize',14)

# Technical Specification for Computer Used

The computer used is a MacBook Pro 15 in with 2.5 GHz Processor Intel Core i7. RAM Memory of 16GB with1600MHz DDR3, with 4 cores and L2 and L3 cache of 256KB and 6MB respectively. Graphics Card of AMD Radeon R9 M370X 2048 MB and Intel Iris Pro 1536 MB. Currently running MacOS High Sierra, Version 10.13.4.

# Results

In the Gauss Seidel pseudocode, the two dominant parameters used were i and j, both being indices of the x and y positions respectively. These parameters were important to know where the current position of the code is as it is running and due to discretization. The rest of the parameters are shown below:

* N: Number of points
* h: discretization step, dx=dy
* U: initial guess matrix input (previous timestep)
* C: current timestep matrix
* F: force function
* Counter: the number of iterations needed for convergence
* Error: the difference in values for each u value between the previous and present time step
* Epsilon: the maximum error allowed for successful convergence

In the SOR method the same parameters, as the one mentioned above, are present with an additional parameter, ω, in the form of a coefficient which acts for faster convergence.

NUMBER OF POINTS USED FOR DISCRETIZATION

1. Gauss-Seidel: The number of points used has a tremendous effect since the accuracy has increased. In Figures 1 and 2 one can clearly see that just doubling the mesh made a tremendous effect on the accuracy from 20 to 40 nodes.

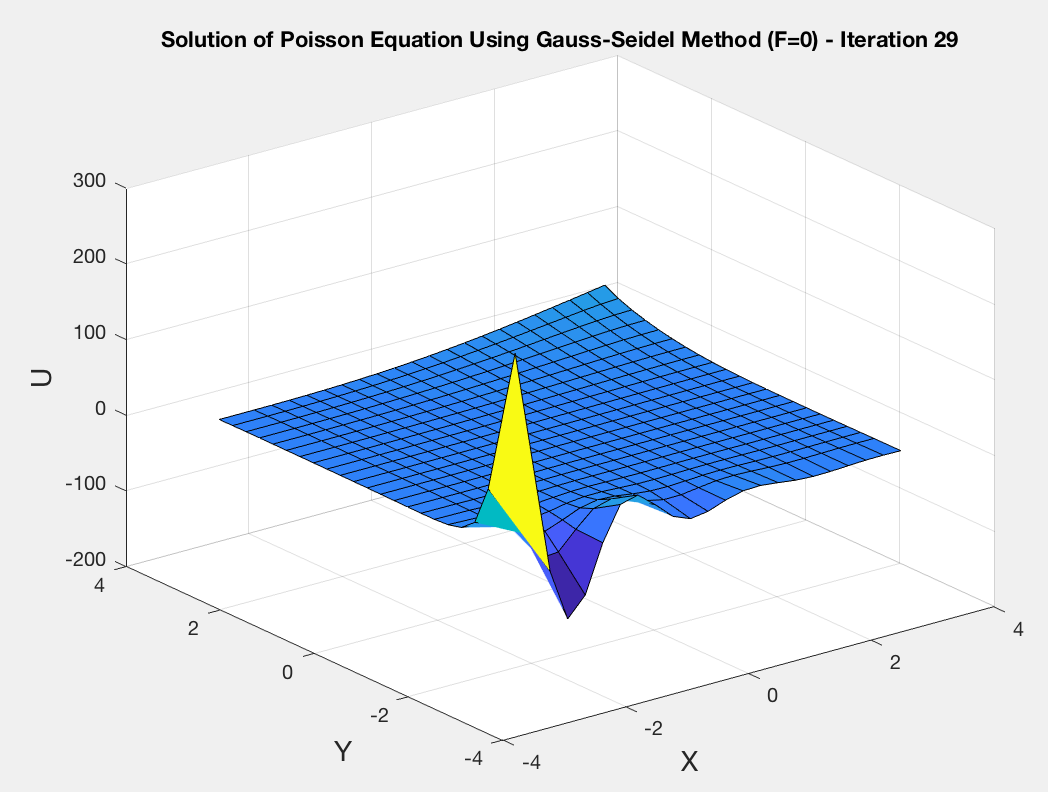


Figure 1: POISSON SOLUtion with gauss seidel (N=20)

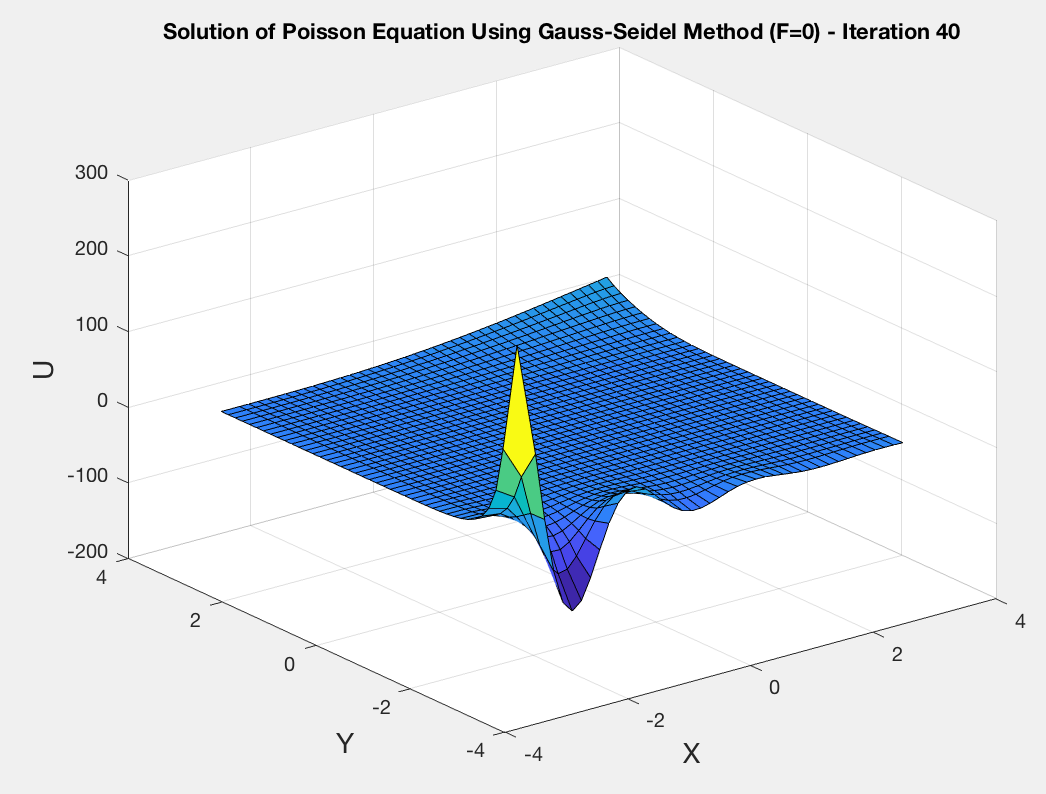


Figure 2:POISSON SOLUTION WITH GAUSS SEIDEL (N=40)

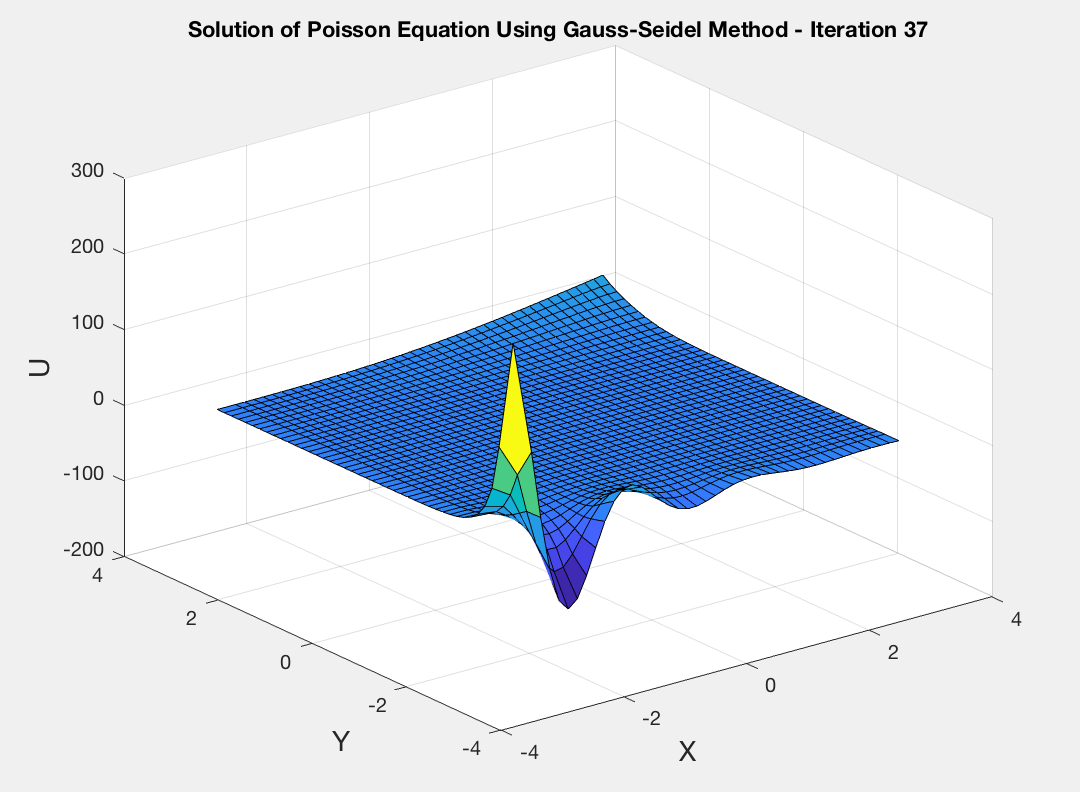


Figure 3: POISSON SOLUTION WITH GAUSS SEIDEL (N=40)

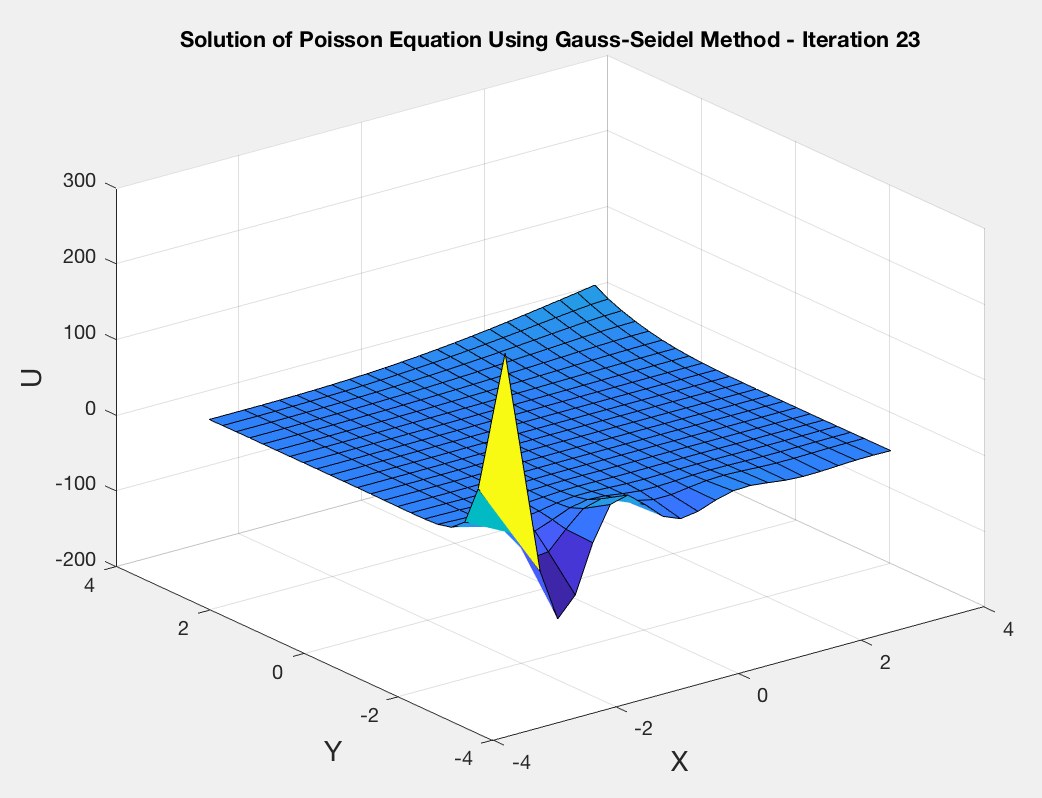


Figure 4:POISSON SOLUTION WITH GAUSS SEIDEl (N=20)

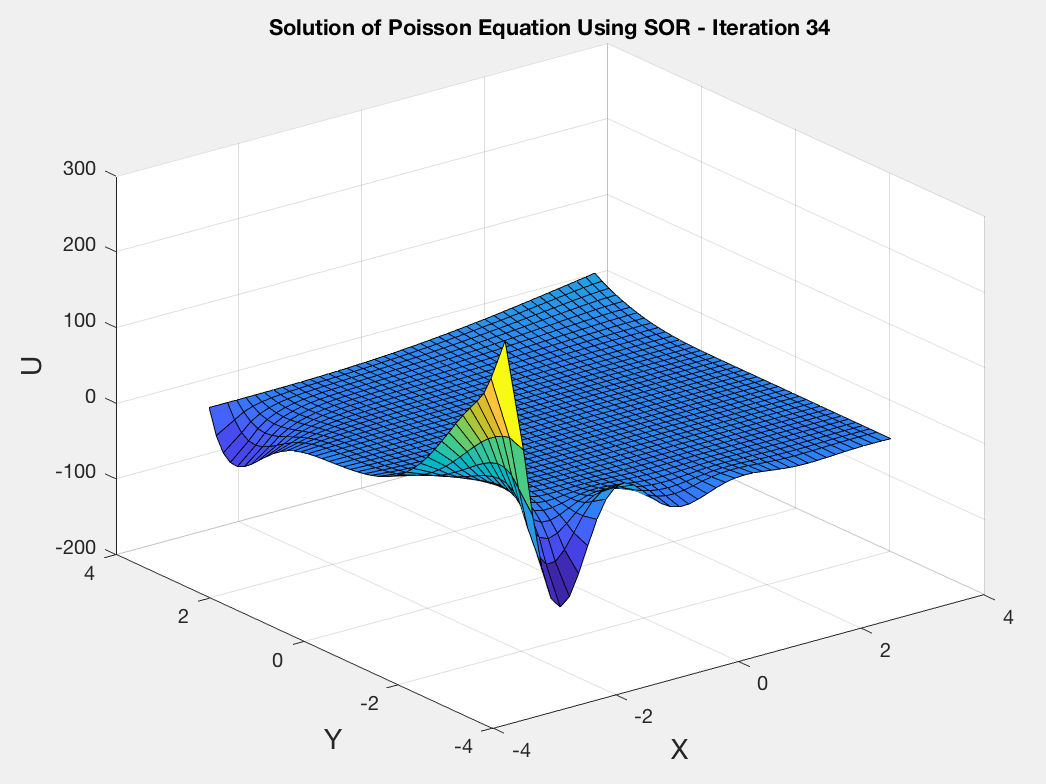


Figure 5:POISSON SOLUTION WITH sor (N=40)

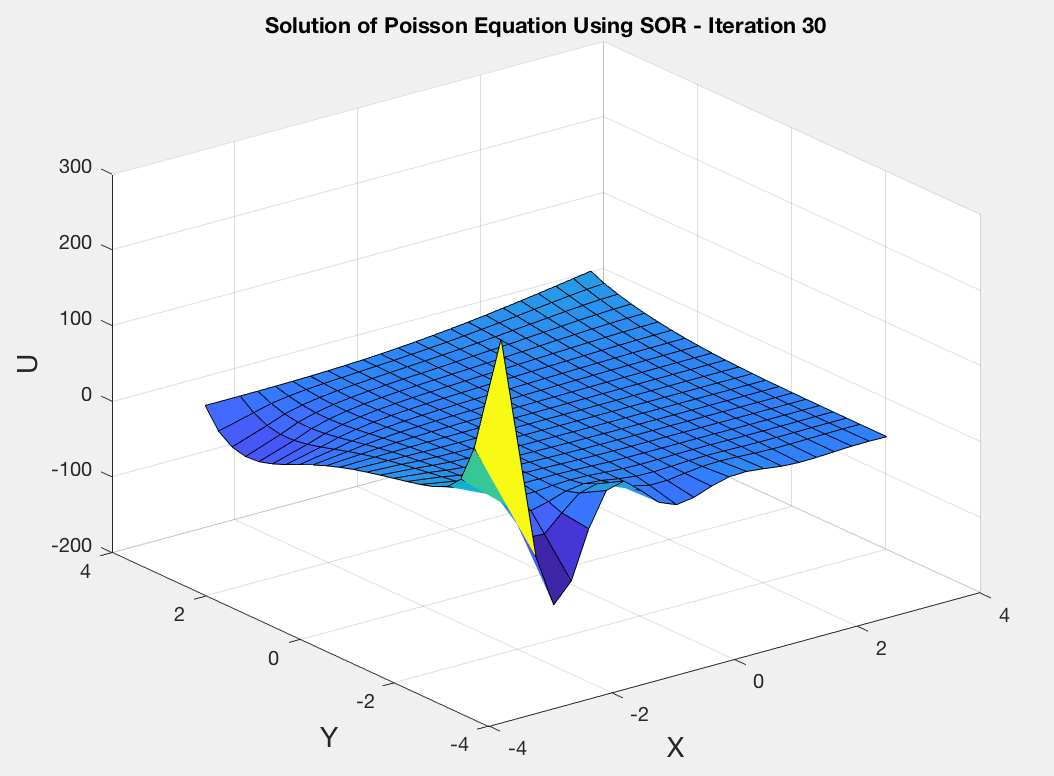


Figure 6: POISSON SOLUTION WITH SOR (N=20)