Project A- Helmholtz Equation Report

MECE 5397

By: Freddy Dominguez

May 4, 2017

Code: AHc2-5

Problem Statement:

The problem at hand revolves around finding a solution or set of solutions of the 2 dimensional Helmholtz equation inside a square, the square itself is constrained with both Newman and Dirichlet boundary conditions at all its four sides. Two methods for finding a solution will be implemented, first will be the use of the Gauss-Seidel method of linear equations, the second method will the use of the Gauss-Seidel method but with Successive Over Relaxation or SOR. Per each test case that will be ran through this code, special attention will be given to the number of iterations each method requires in order to reach the lowest possible error value.

$$\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \Lambda u = F(x, y)$$

The domain of interest is $-\pi < x < \pi$ and $-\pi < y < \pi$

The boundary conditions are the following:

$$u(x = -\pi, y) = \cos\left(\pi\left(y - a_y\right)\right)\cosh\left(b_y - y\right)$$

$$u(x = \pi, y) = \left(y - a_y\right)^2 \sin\left(\frac{\pi\left(y - a_y\right)}{b_y - a_y}\right)$$

$$\frac{\partial u}{\partial y}_{y=a_y} = 0 \qquad \frac{\partial u}{\partial y}_{y=b_y} = 0$$

Discretization of the Helmholtz equation

Given the equation for the Helmholtz equation

$$\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \Lambda u = F(x, y)$$

We discretize the second order derivatives of the equation by replacing them with the following expressions for a 2 dimensional system

$$\frac{U_{i,j-1} - 2U_{ij} + U_{i,j+1}}{\Delta y^2} + \frac{U_{i-1,j} - 2U_{ij} + U_{i+1,j}}{\Delta x^2} + \Lambda U_{ij} = F_{ij}$$

Where $\Delta y^2 = \Delta x^2 = \Delta h^2$

Rearranging and combining:

$$U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{ij} + \Lambda U_{ij}\Delta h^2 = F_{ij}\Delta h^2$$

Solving for U_{ij} we have the expression:

Let $B = (U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1})$ Then $U_{ij} = \frac{B - F_{ij}\Delta h^2}{(4 - \Lambda \Delta h^2)}$

Pseudocodes

-Pseudocode for solving any system of linear equation, using either Gauss-Seidel or SOR

```
function [ x, err, iter, flag ] = SOR_trial2(A, x, b, w, max_it, tol)
flag=('system converges');
iter=0;
norma2_b=norm(b);
if (norma2_b==0.0)
        norma2_b=1.0;
end
r=b-A*x;
err=norm(r) / norma2_b;
if (err < tol)
        return
end
[M,N,b]= matsep(A,b,w);
for iter=1:max_it
        x_1=x;
        x=M \ (N*x+b);
        err=norm(x-x_1)/norm(x);
        if (err<=tol)
          break
        end
end
 b=b/w;
if (err>tol)
        flag=('no convergence')
end
end
function [M, N, b]= matsep( A, b, w)
b=w*b;
M= w*tril(A,-1)+diag(diag(A));
N= -w* triu(A,1)+(1.0-w)*diag(diag(A));
end
```

The pseudocode shown previously is able to solve a system of equations of the form Ax=b, where A is a square matrix, x is the initial value of the unknowns of the equations of the system and b are the constants of the system. The pseudocode was written to focus on the SOR scheme, but knowing that if the relaxation value is set to 1, the code will solve for a system of equations simply using the Gauss-Seidel process.

-Pseudocode for the setup of the Helmholtz equation

```
function[Solution,Error_estimate,Number_of_iterations,flag]=Helmholtz(N,w,C1,C2)
n=N;
ax=-pi; ay=-pi; bx=pi; by=pi;
if C1==1
   gamma=-pi;
elseif C1==0
   gamma=0;
end
x=linspace(ax,bx,n); y=linspace(ay,by,n);
phiab=cos(pi*(y-ay)).*cosh(by-y);
psiab=((y-ay).^2).*sin((pi*(y-ay))/(2*(by-ay)));
u(:,1)=phiab; u(:,n)=psiab
F = cos((pi/2)*(2*((x-ax)./(bx-ax))+1)).*sin(pi*((y-ay)./(by-ay)));
if C2==1
   F=F;
elseif C2==0
   F=0;
end
F1=F';
h=(bx-ax)/n;
for j=2:n-1
   for i=2:n-1
      F(i,j) = cos((pi/2)*(2*((x(i)-ax)./(bx-ax))+1)).*sin(pi*((y(j)-ay)./(by-ay)));
      u(i,j)= (1/(4-(gamma*h^2)))*((u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1))-F(i,j));
   end
end
u(1,:)=u(2,2); u(n,:)=u(n-1,n-1);
U=u;
mesh(U)
S0= eye(n);
S1=diag(ones(n-4,1),4);
S2=diag(ones(n-5,1),5) ;
S3=diag(ones(n-6,1),6) ;
S4=diag(ones(n-10,1),10);
S=S0+S1+S2+S3+S4;
U_solve=S.*U;
[Solution,Error_estimate,Number_of_iterations,flag]=SOR_trial2(U_solve,zeros(n,1),F1,w,1e4,0.01);
end
```

The previous pseudocode shown allows for the setup of the Helmholtz equation as a system of equations, with both Dirichlet and Newman boundary conditions as stated in the original problem. Having done this, the code makes use of the SOR code in order to find the solutions to the system and know the error and number of iterations per case.

Technical Specifications of computer used

MacBook Pro (13 in Mid 2009)

OS X El Capitan, Version 10.11.6

Processor: 2.26 GHz Intel Core 2 Duo

Memory: 8 GB 1067 MHz DDR3

Graphics: NVIDIA GeForce 9400M 256MB

Results

The following are some of the results collected from the various tests ran through the Helmholtz solving code

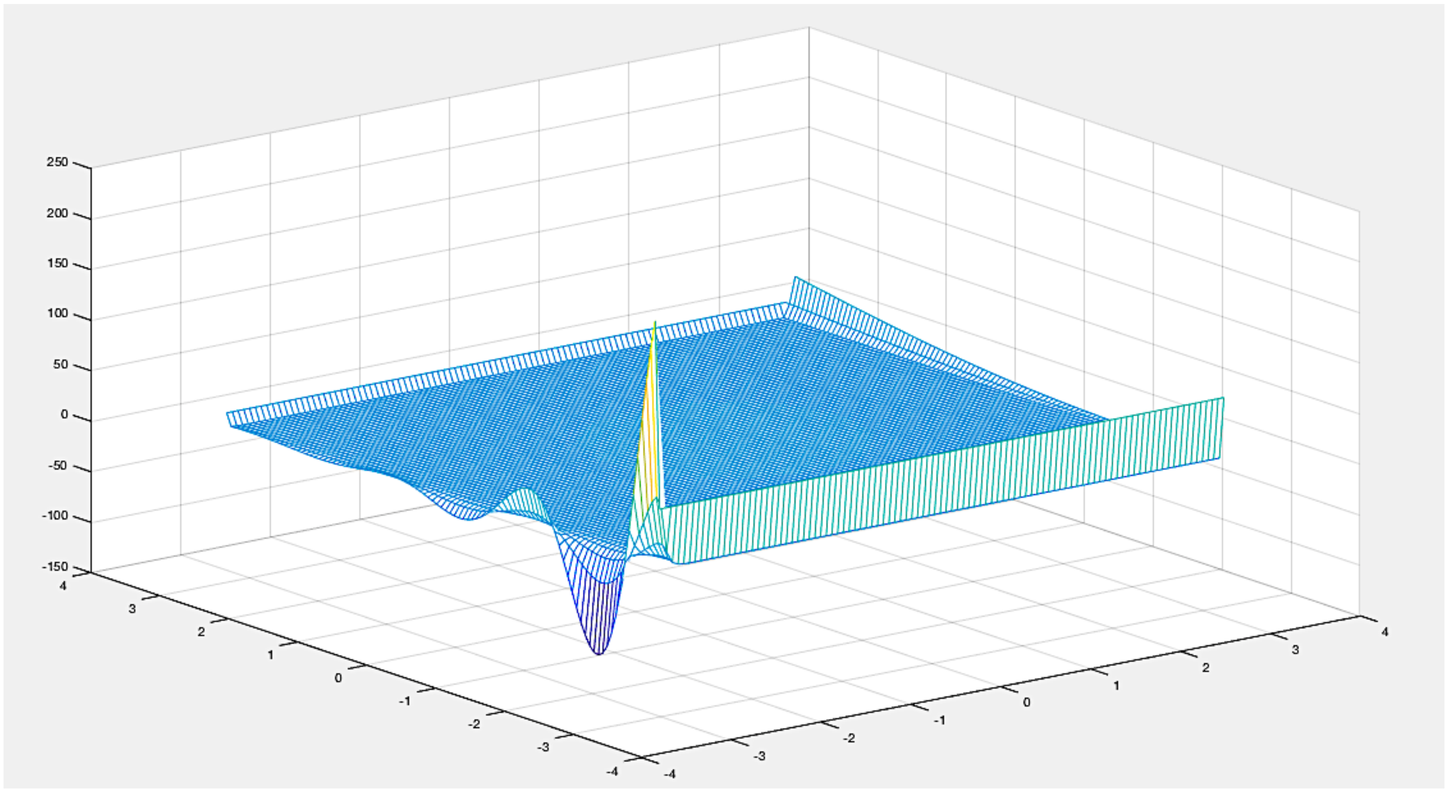

Figure 1- Mesh plot of values within the square region

The following tables show the tabulated results of the number of iterations and error amount for various runs of the Helmholtz solving code (N=number of points)

	Gauss-Seidel		SOR	
N	Iterations	Error	Iterations	Error
10	3	0.0063	10	0.0074
15	5	0	13	0.0086
20	6	0	17	0.0065
25	7	0	19	0.0071
30	8	0	20	0.007
100	24	0	59	0.0088

Table 1- Results of various code runs

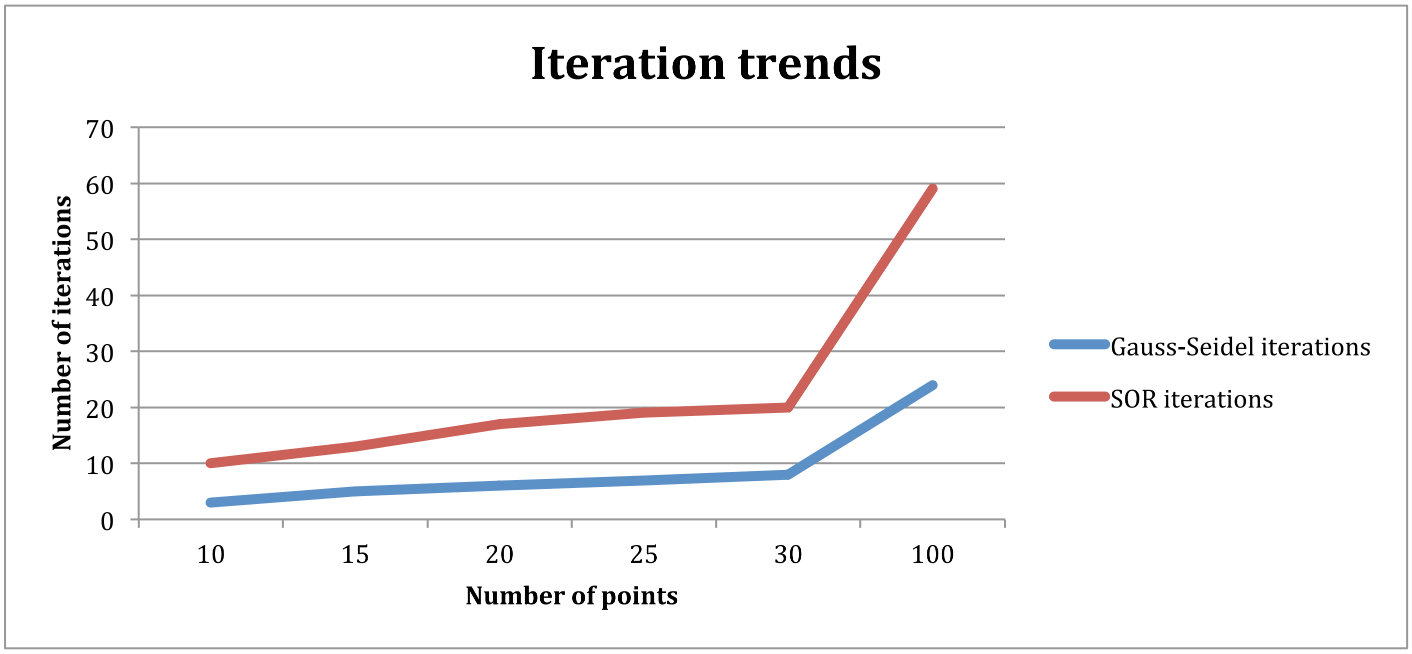

Figure 2- Iteration trends of both Gauss-Seidel and SOR

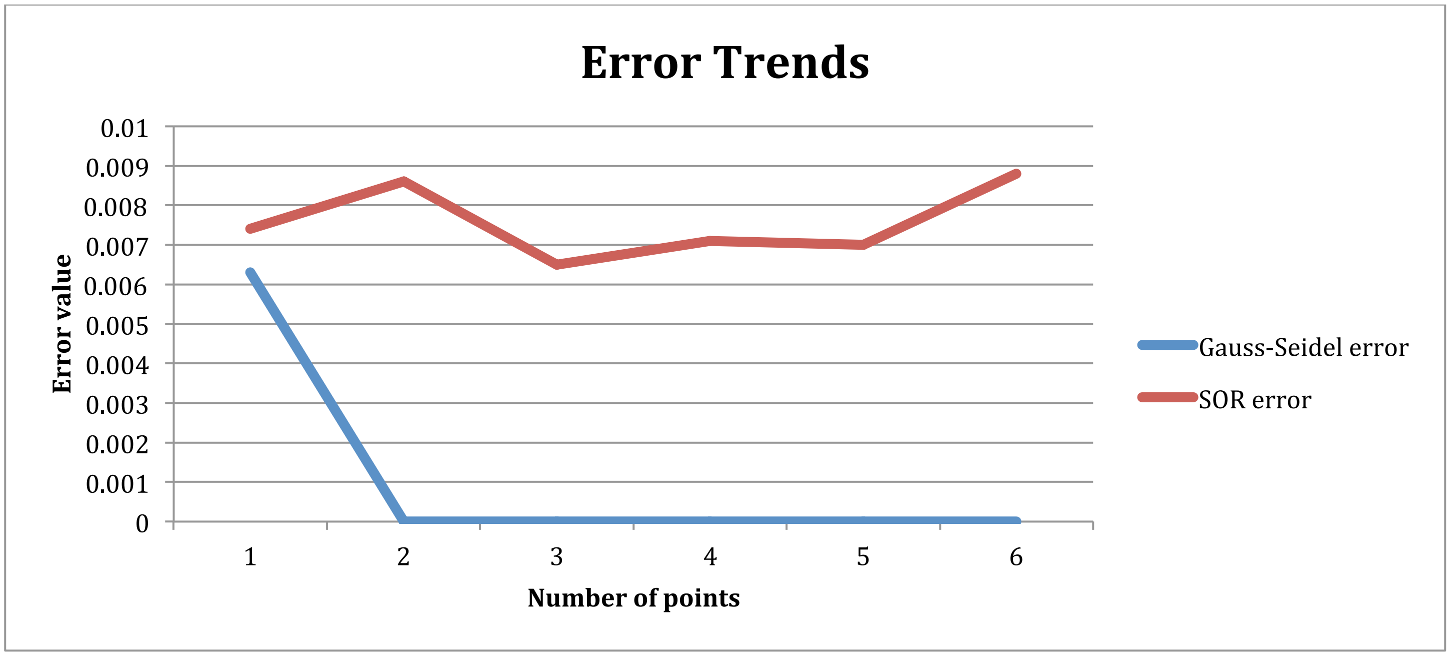

Figure 3- Error trends of both Gauss-Seidel and SOR

Some conclusions that can be inferred from looking at this particular set of data is that the Gauss-Seidel method exhibits both lower values of iterations to reach convergence as well as converging to a threshold error value than with the use of SOR, it is of course important to not that as the relaxation constant W is increased closer to 1, these trends will begin to parallel each other even more.

Special case, Λ=0 and F=0

-For all instances that F was set to 0, the solutions, error values and iteration count for both Gauss-Seidel and SOR were all 0, this should not be the case, but due to the limitations of the codes used this is the result.

-For all instances that Λ=0, the results are shown bellow

	Gauss-Seidel		SOR	
N	Iterations	Error	Iterations	Error
10	3	0.0063	10	0.0076
15	5	0	13	0.0064
20	6	0	17	0.0064
25	7	0	19	0.0089
30	8	4.71E-04	20	0.0097
100	24	0.0024	59	0.0088

Table 2- Results for when Λ=0

One thing to now from the data of this special case is that the number of iterations per number of points remains unchanged from before only the error values change significantly

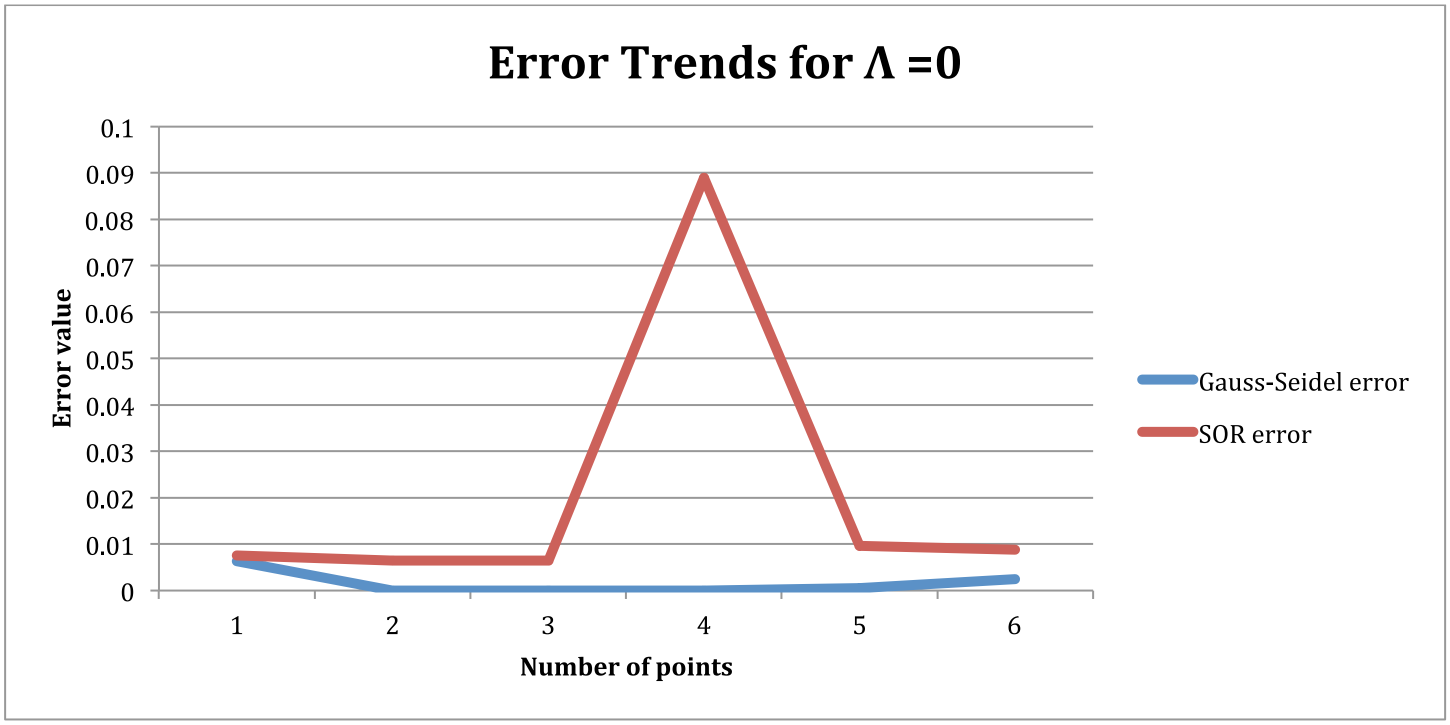

Figure 4- Error trends for special case