MECE 5397

SCIENTIFIC COMPUTING FOR ENGINEERS

THE POISSON EQUATION

Code: APc2-2

SPRING 2017

THOMAS HILLMAN

Abstract

The intent of this project was to explore and understand the Gauss Seidel and Gauss Seidel with successive overrelaxation methods of approximation. These methods require iterative operations over the entire problem domain to come to an acceptable solution. To evaluate both methods, a Poisson Equation problem over a square domain is given with boundary conditions and both methods were used to solve the problem. Both methods yield a similar result, however the method using successive over relaxation proved to converge to the answer sooner. Such a speedy solution is prompted because each iterative solution is assumed to be better than the previous, so the values are pushed further in the direction which the solution trends toward; the solution values converge to the exact value sooner by requiring fewer calculations. Each method is qualified by analysis of the number of cycles to convergence for a given number of nodes. Necessary grid size is also analyzed by observing the solution at a point in each quadrant for different grid sizes. Plots of results are given for various grid sizes. The Gauss Seidel method utilizing successive over relaxation proved to be the superior method.

Table of Contents

Problem Statement

The two-dimensional region between -pi and pi in the x and y coordinates is subject to boundary conditions and a forcing function. See the below image for the domain parameters.

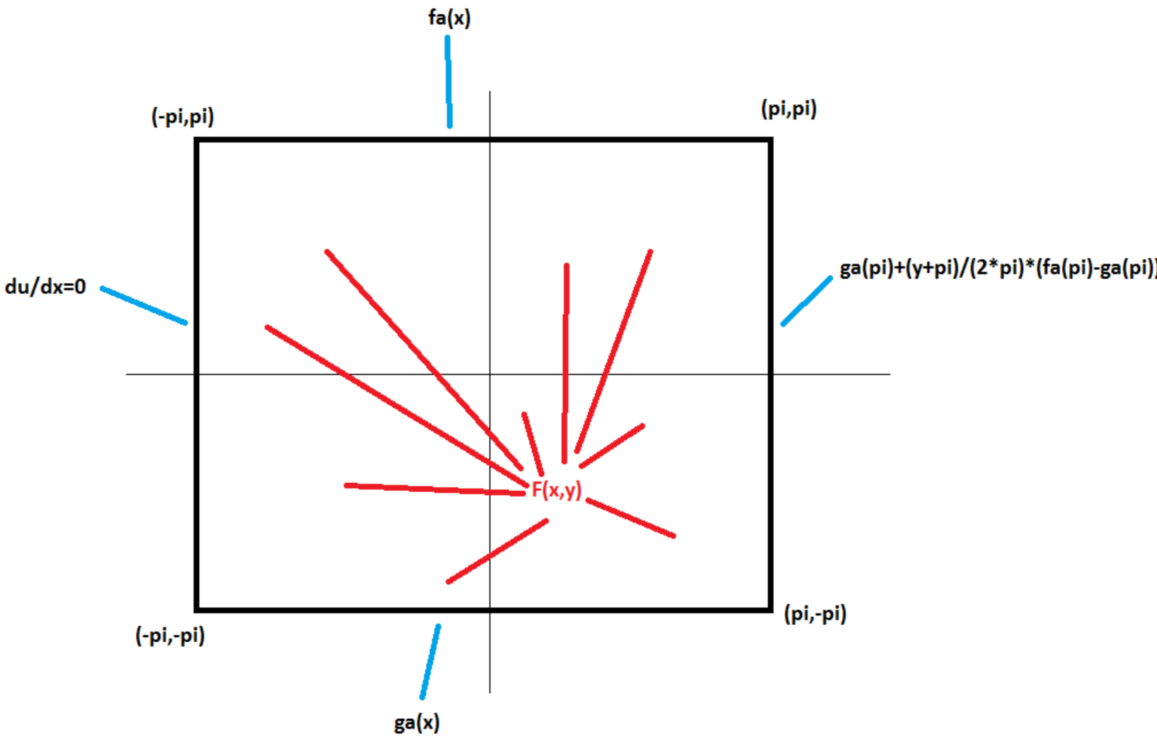


Figure 1 Domain Setup

Where $g_a(x) = (x + pi)^2 * \cos(-x)$ and $f_a(x) = x * (x + pi)^2$

And the forcing function $F(x, y) = \sin\left(pi * \frac{x+pi}{2*pi}\right) * \cos(\left(\frac{pi}{2}\right) * \left(2 * \frac{y+pi}{2*pi} + 1\right))$.

The Poisson Equation is $\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = -F(x, y)$.

Descritized Equations

The Taylor series expansion is used to approximate the second derivatives as

$$\frac{d^2u}{dx^2} = (u_{i-1,j} - 2 * u_{i,j} + u_{i+1,j})/\Delta x^2$$

And

$$\frac{d^2u}{dy^2} = (u_{i,j-1} - 2 * u_{i,j} + u_{i,j+1})/\Delta y^2$$

For this problem, the step sizes in x and in y are kept the same for simplicity. So the Poisson equation can be approximated as:

$$\frac{u_{i-1,j}-2*u_{i,j}+u_{i+1,j}}{h^2}+\frac{u_{i,j-1}-2*u_{i,j}+u_{i,j+1}}{h^2} = -F(x,y)$$

- ⇨ $u_{i-1,j} - 2 * u_{i,j} + u_{i+1,j} + u_{i,j-1} - 2 * u_{i,j} + u_{i,j+1} = -F(x,y) * h^2$
- ⇨ $u_{i-1,j} - 4 * u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} = -F(x,y) * h^2$

Solving for a node:

- ⇨ $u_{i,j} = F(x,y) * h^2 + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}$

Numerical Method

Two numerical methods were used to approximate the solution over the domain.

The first is the Gauss Seidel iterative method, which solves nodes based on values of adjacent nodes, and after the entire domain is solved the procedure is repeated over and over until the new values converge to the old ones within a given acceptable error.

The second method used to solve the domain is the Gauss Seidel using relaxation. The process is essentially identical to the Gauss Seidel Method except that each successive iteration is assumed to be closer to the exact solution than the previous iteration. Thus, a weighting factor is applied to the new solution based on the difference between the new and old solutions. The weighting factor Lambda as shown in the below equation was determined to be most effective at 1.4 by experimentation.

$$x_{new_i} = \lambda * x_{new_i} + (1 - \lambda) * x_{old_i}$$

After the solutions are obtained, the process is repeated using zero forcing function.

Process

The numerical methods used to solve for u inside the domain are the Gauss-Seidel and Gauss Seidel with successive over relaxation. The nodes are solved for iteratively until the error between the previous solutions and the current solutions over the domain reaches 1%.

The nodes are solved for in an order that reduces the necessary number of iterations to convergence. The boundary conditions serve as known values of u. And nodes are solved for in the following order.

The nodes along the boundary are solved first.

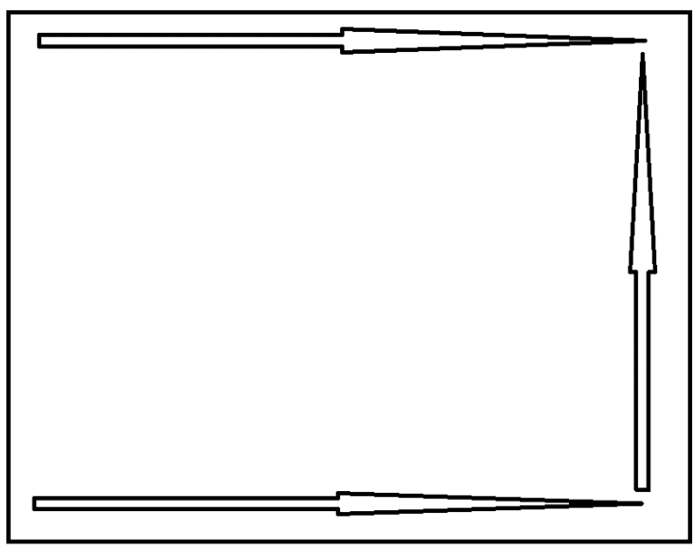


Figure 2 Node Solutions Step 1

Then the nodes just solved for act as a new smaller boundary for the next nodes to be solved. This process is repeated until the triangular region near the Neumann condition remain.

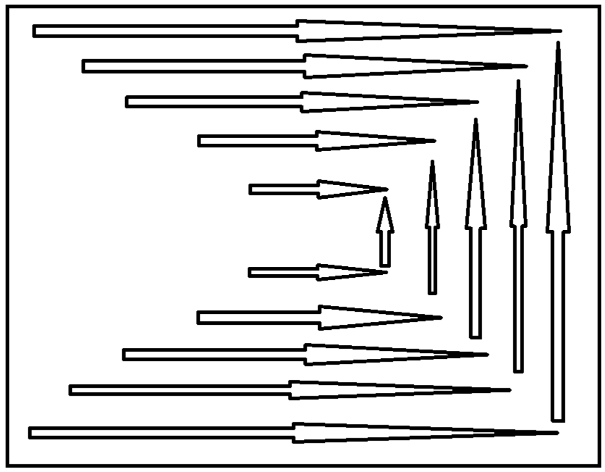


Figure 3 Node Solutions Step 2

Then the nodes in the center of the domain are solved using the previously solved nodes. Starting at the inside and working toward the boundary.

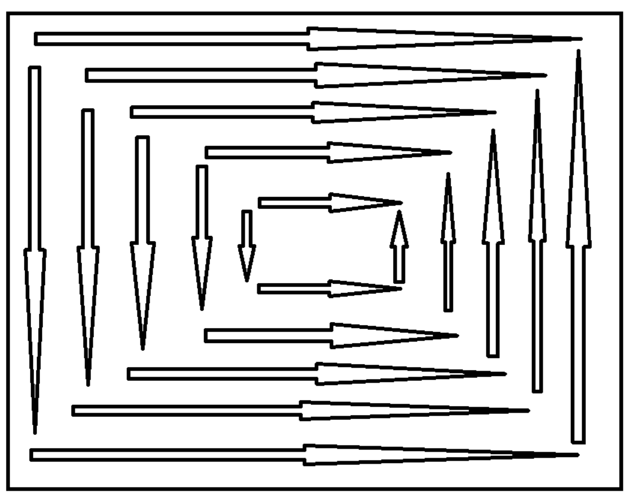


Figure 4 Node Solutions Step 3

The nodes on the left side of the domain are copied across the boundary as ghost nodes, artificially simulating the Neuman condition.

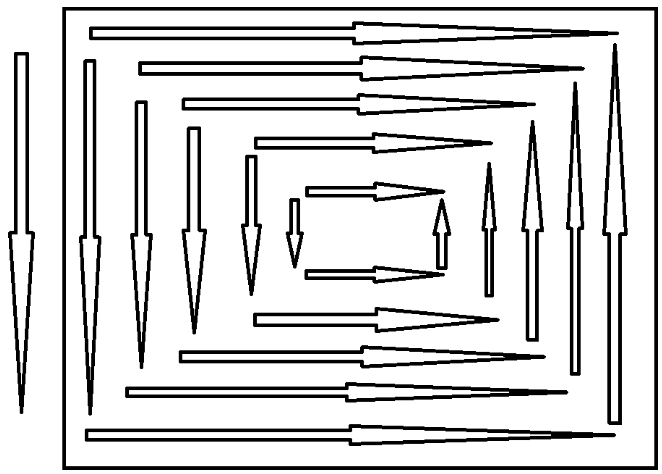
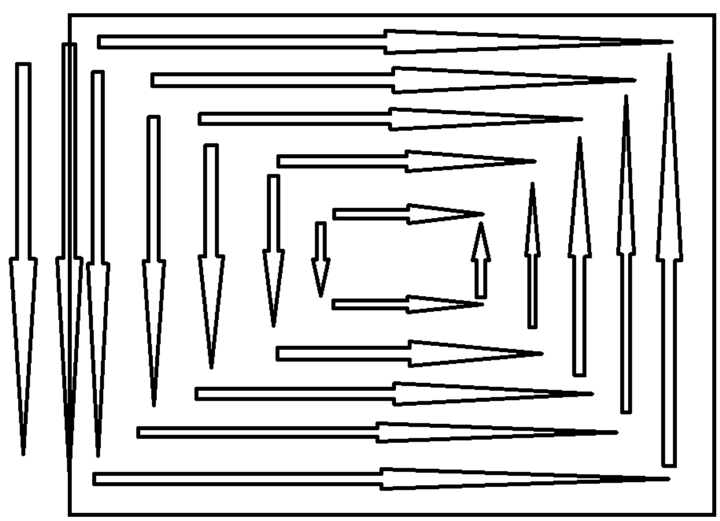


Figure 5 Node Solutions Step 4

Finally, the nodes along the left boundary are solved and the first Gauss Seidel solution is



complete.

Figure 6 Node Solutions Step 5

This process is repeated until the solutions converge to 1% discrepancy.

Pseudo Code

```
% Top B.C.'s
for k=2:N+1
    u(1,k)=x(k)*(x(k)+pi)^2;
end

% Bottom B.C.'s
for k=2:N+1
    u(N,k)=(x(k)+pi)^2*cos(pi*x(k)/-pi);
end

% Right B.C.'s
for k=1:N
    u(k,N+1)=((pi+pi)^2*cos(pi*pi/-pi))+...
        ((y(k)+pi)/(pi+pi))*...
        (pi*(pi+pi)^2-...
        ((pi+pi)^2*cos(pi*pi/-pi)));
end

errorval=100;
iterations=0;

while errorval>1
    iterations=iterations+1;
    u_old=u;
    counter=2;
    while counter<=N/2

        % Top Pyramid
        for j=      N-counter+2    :   -1  :    counter+1
        u( counter , j ) =
((sin(pi*(x(j)+pi)/(2*pi))*cos((pi/2)*(2*(y(counter)+pi)/(2*pi)+
1)))*h^2+...
             u(counter-1,j)+u(counter+1,j)+u(counter,j-
1)+u(counter,j+1))/4;
        end


        % Bottom Pyramid
        for j=      N-counter+2   :   -1  :     counter+1
            u( N-counter+1  , j ) =
((sin(pi*(x(j)+pi)/(2*pi))*cos((pi/2)*(2*(y(N-
counter+1)+pi)/(2*pi)+1)))*h^2+...
             u(N-counter+2,j)+u(N-counter,j)+u(N-counter+1,j-
1)+u(N-counter+1,j+1))/4;
        end
```

```
        % Right Pyramid
        for k=       counter+1       :        N-counter
            u( k , N-counter+2 ) = ((sin(pi*(x(N-
counter+2)+pi)/(2*pi))*cos((pi/2)*(2*(y(k)+pi)/(2*pi)+1)))*h^2+.
..
             u(k-1,N-counter+2)+u(k+1,N-counter+2)+u(k,N-
counter+1)+u(k,N-counter+3))/4;
        end
    counter=counter+1;

    end
        % Left Pyramid
    counter2=0;

    for j=       floor(N/2)+1 :   -1 :  3
        counter2=counter2+1;
        for k =  floor(N/2)-counter2+2    :
floor(N/2)+counter2-1
            u( k , j) =
((sin(pi*(x(j)+pi)/(2*pi))*cos((pi/2)*(2*(y(k)+pi)/(2*pi)+1)))*h
^2+...
               u(k-1,j)+u(k+1,j)+u(k,j-1)+u(k,j+1))/4;
        end

    end


    % ghost nodes
    u(:,1)=u(:,3);

    % left boundary nodes
    for k=2:N-1

u(k,2)=((sin(pi*(x(2)+pi)/(2*pi))*cos((pi/2)*(2*(y(k)+pi)/(2*pi)
+1)))*h^2+...
               u(k-1,2)+u(k+1,2)+u(k,1)+u(k,3))/4;
    end

       for k=1:N
           for j=1:N+1
   error(k,j)=abs((u(k,j)-u_old(k,j))/u(k,j))*100;
           end
       end
       % average error over domain
       errorval=mean(mean(error));
end
```

Analysis

The solution is plotted over the domain using different step sizes. For each solution using a different number nodes, the number of cycles throughout the domain to convergence are counted for comparison. The percent difference in the values using Gauss Seidel and Gauss Seidel with relaxation are calculated. The percent of improvement in number of cycles between the two methods is also calculated. Finally, the methods are checked for grid convergence by observing the values at a point in each quadrant, using different grid sizes.

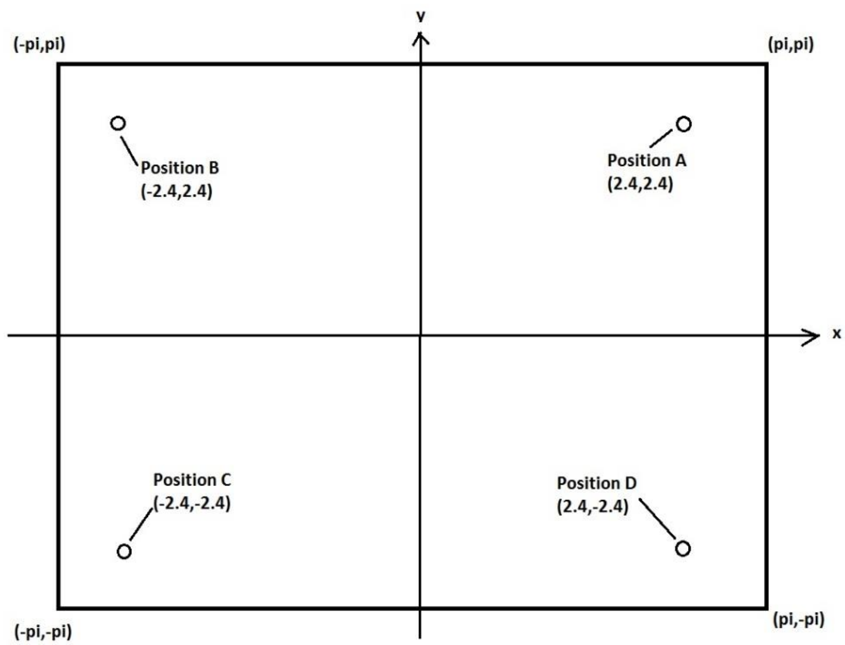


Figure 7 Position Values Used to Check Grid Convergence

Computer Specifications

Operating System: Windows 7 Professional

Processor: Intel® Core™2 Duo CPU T9900 @ 3.06GHz 3.07GHz

Installed RAM: 4.00 GB

System Type: 64-bit

Results

Tabulated Results

The below table shows the performance of the two methods. Each row represents results of the approximation analysis using different domain densities.

Table 1 Result Performances

Nodes	Step Size	GS Cycles	GSR Cycles	% Speed Improvement	GS no F Cycles	GSR no F Cycles	% Speed Improvement No F
10	0.6981	33	24	27.3	30	22	26.7
20	0.3307	93	67	28.0	84	60	28.6
50	0.1282	183	131	28.4	229	165	27.9
100	0.0635	205	147	28.3	220	163	25.9
250	0.0252	248	161	35.1	437	314	28.1
500	0.0126	239	167	30.1	837	602	28.1

Result Analysis

The below graphs illustrate the effect of increasing the number of nodes in the solution. It can be clearly seen that as domain density increases so do the number of cycles until the solution converges. Also, final values in the domain converge to the same result as the number of nodes in the domain become sufficiently large.

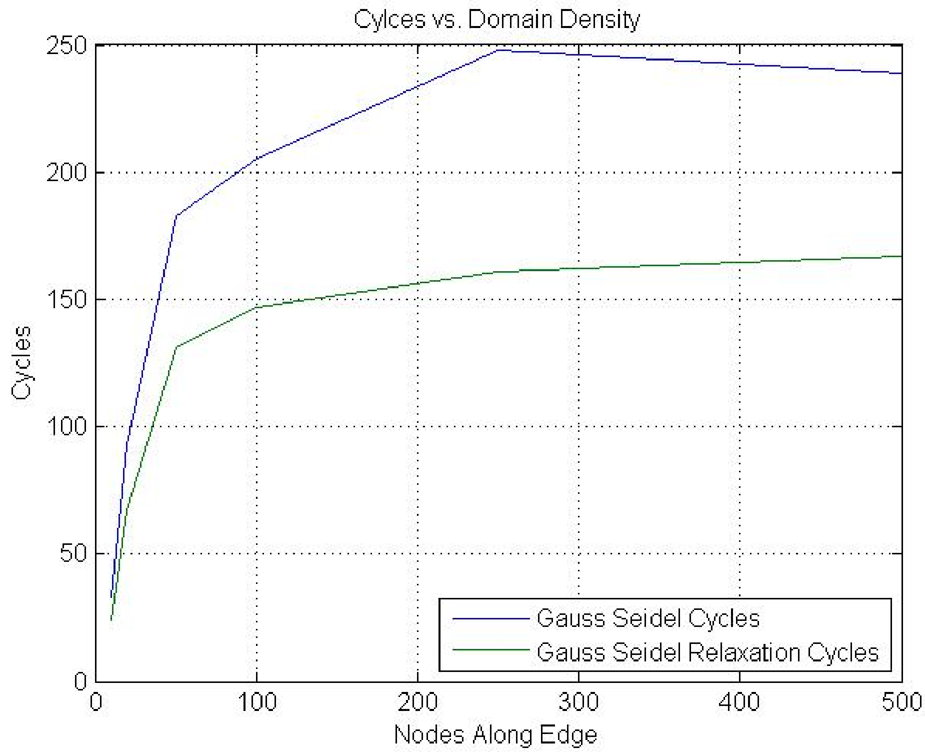


Figure 8 Cycle Analysis

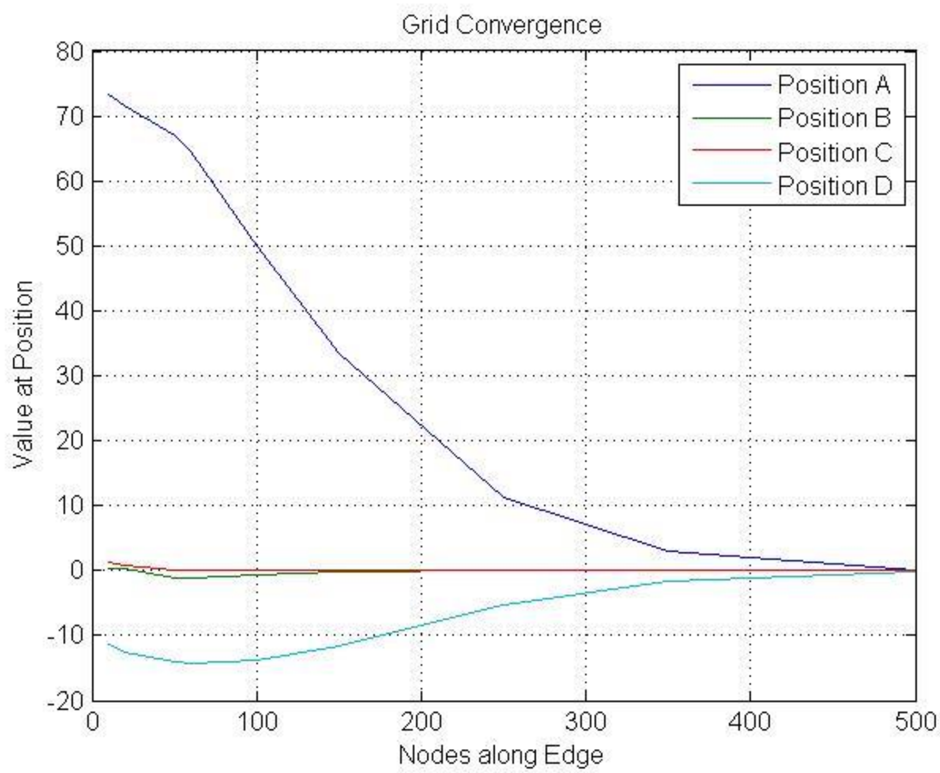


Figure 9 Grid Convergence Analysis

Gauss Seidel Results

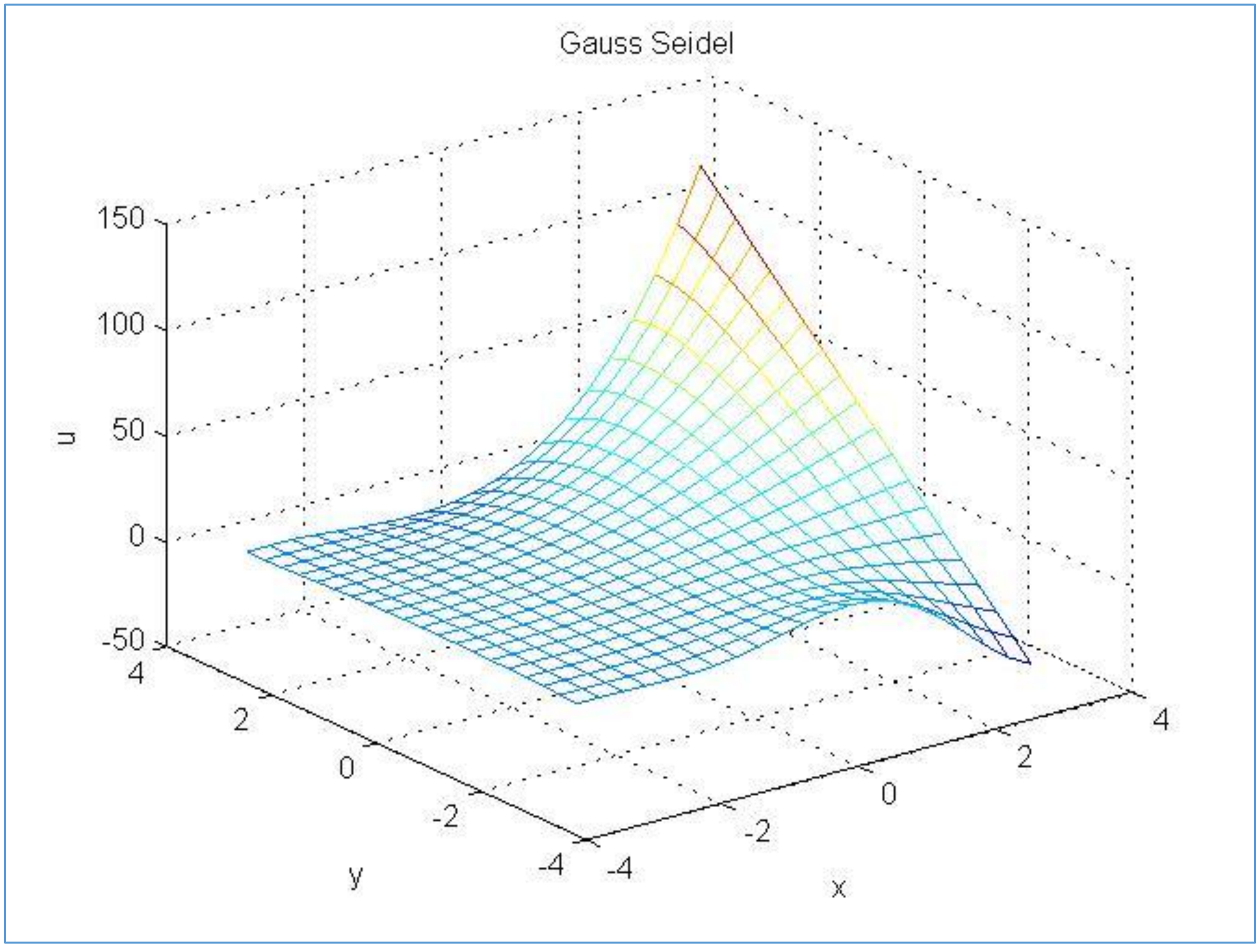


Figure 10 Gauss Seidel 20 x 20

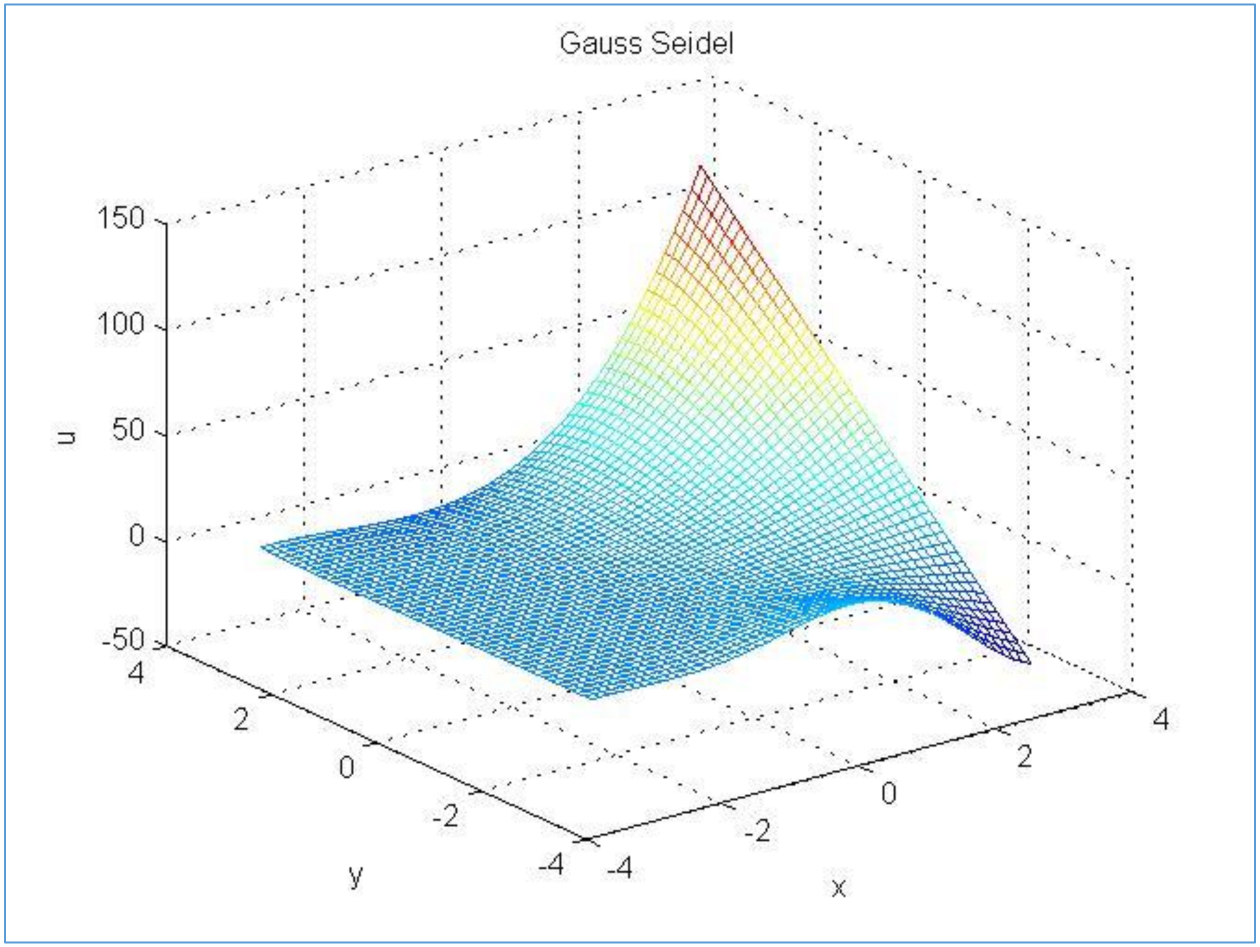


Figure 11 Gauss Seidel 50 x 50

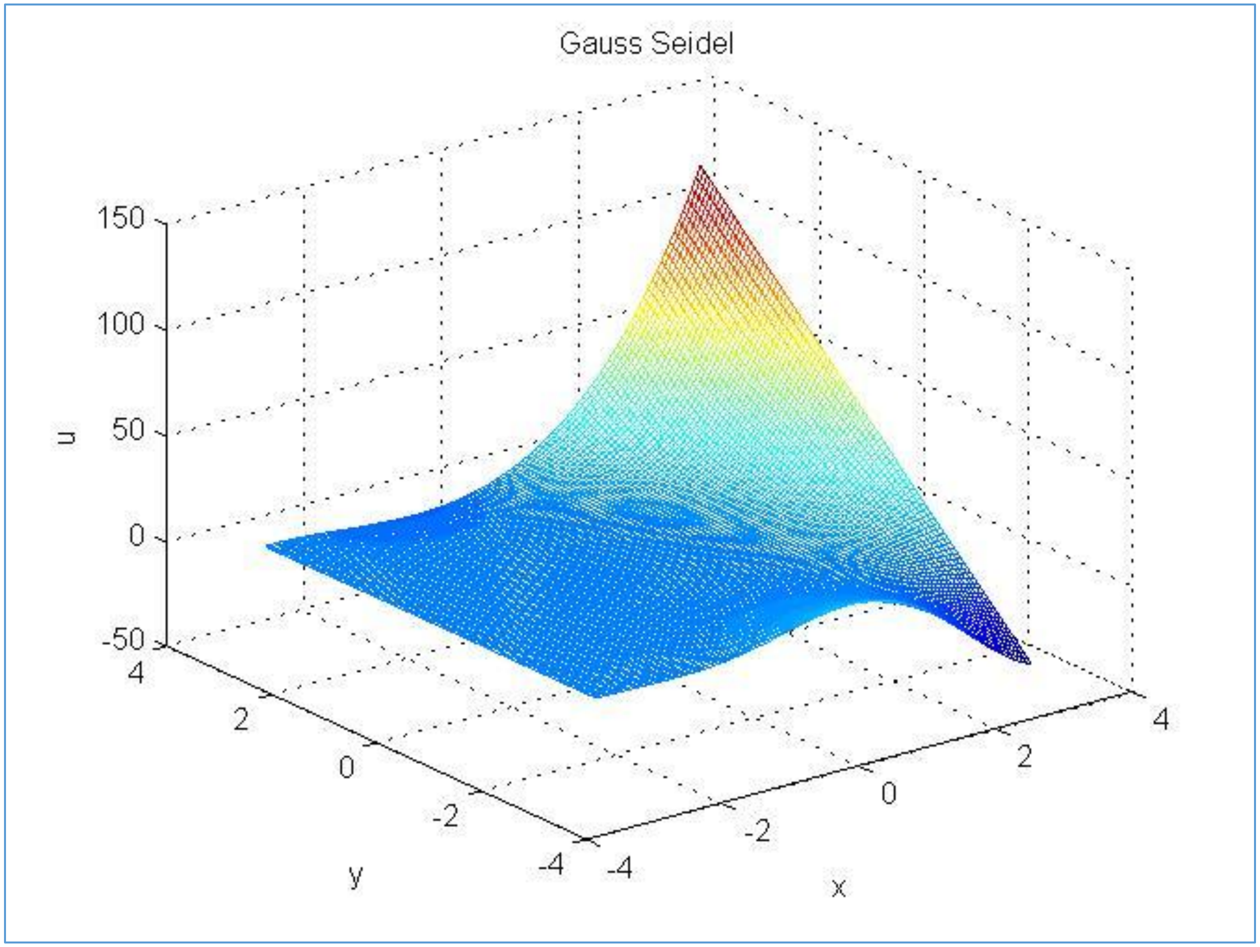


Figure 12 Gauss Seidel 100 x 100

Gauss Seidel Results with Relaxation

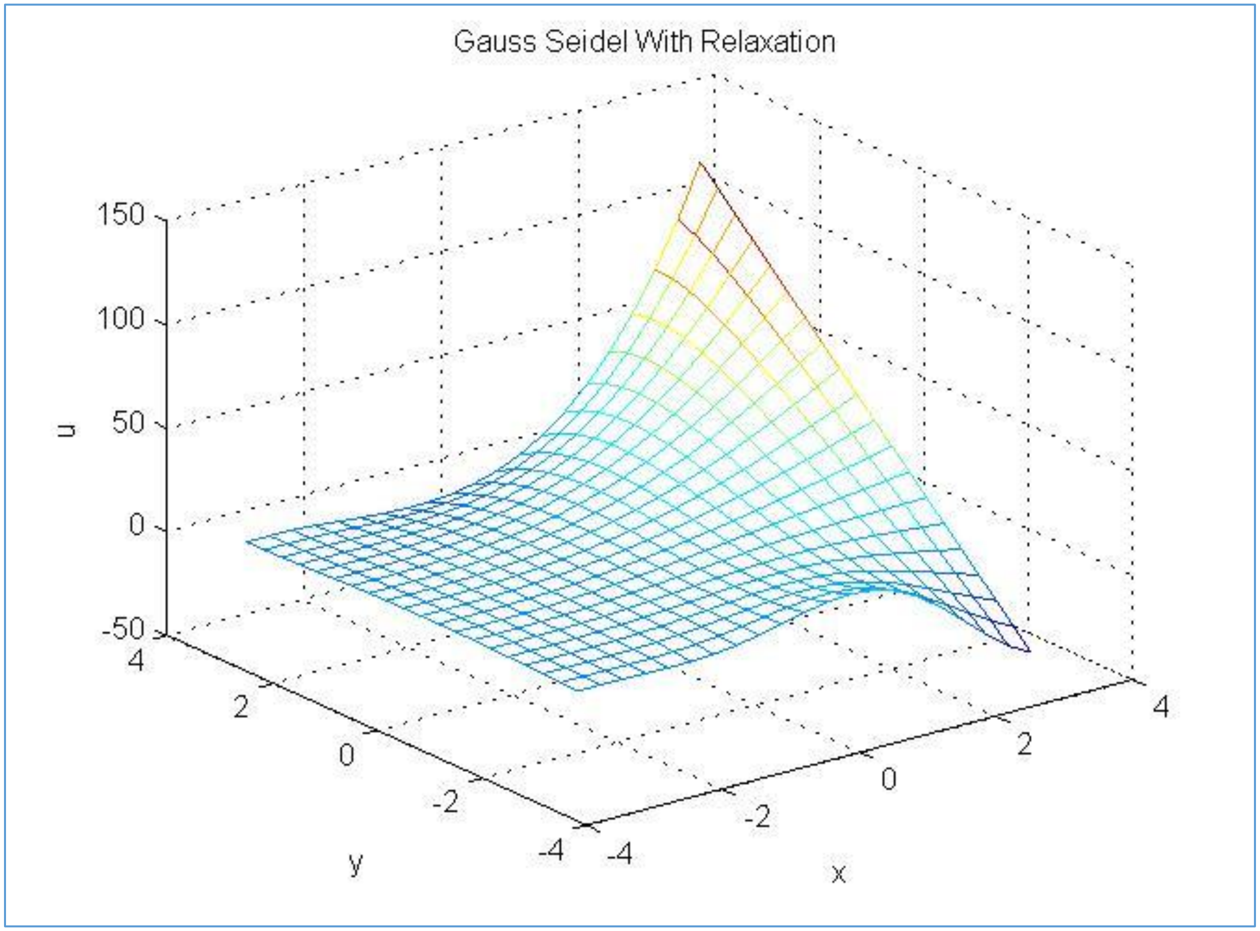


Figure 13 Gauss Seidel 20 x 20 Relaxed

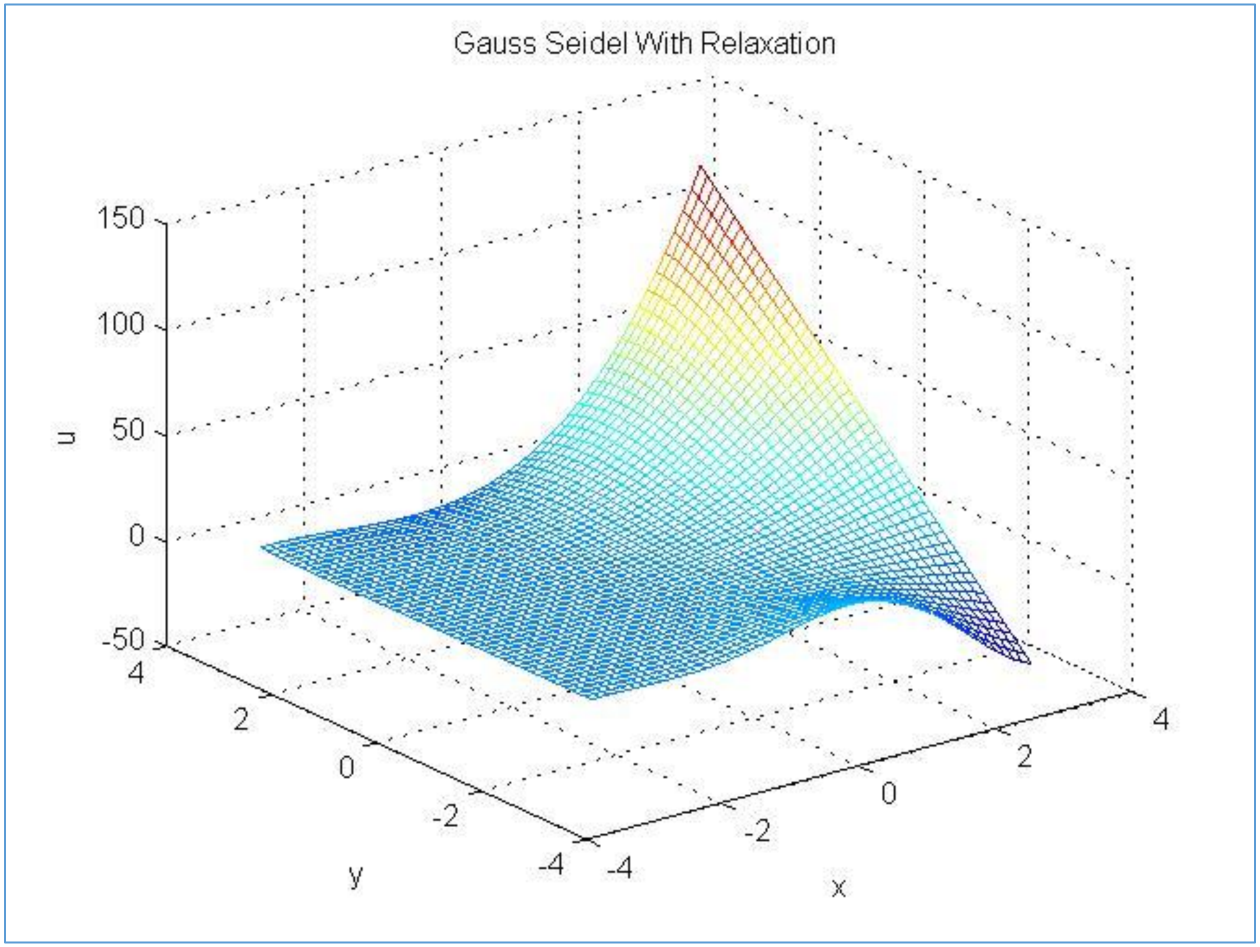


Figure 14 Gauss Seidel 50 x 50 Relaxed

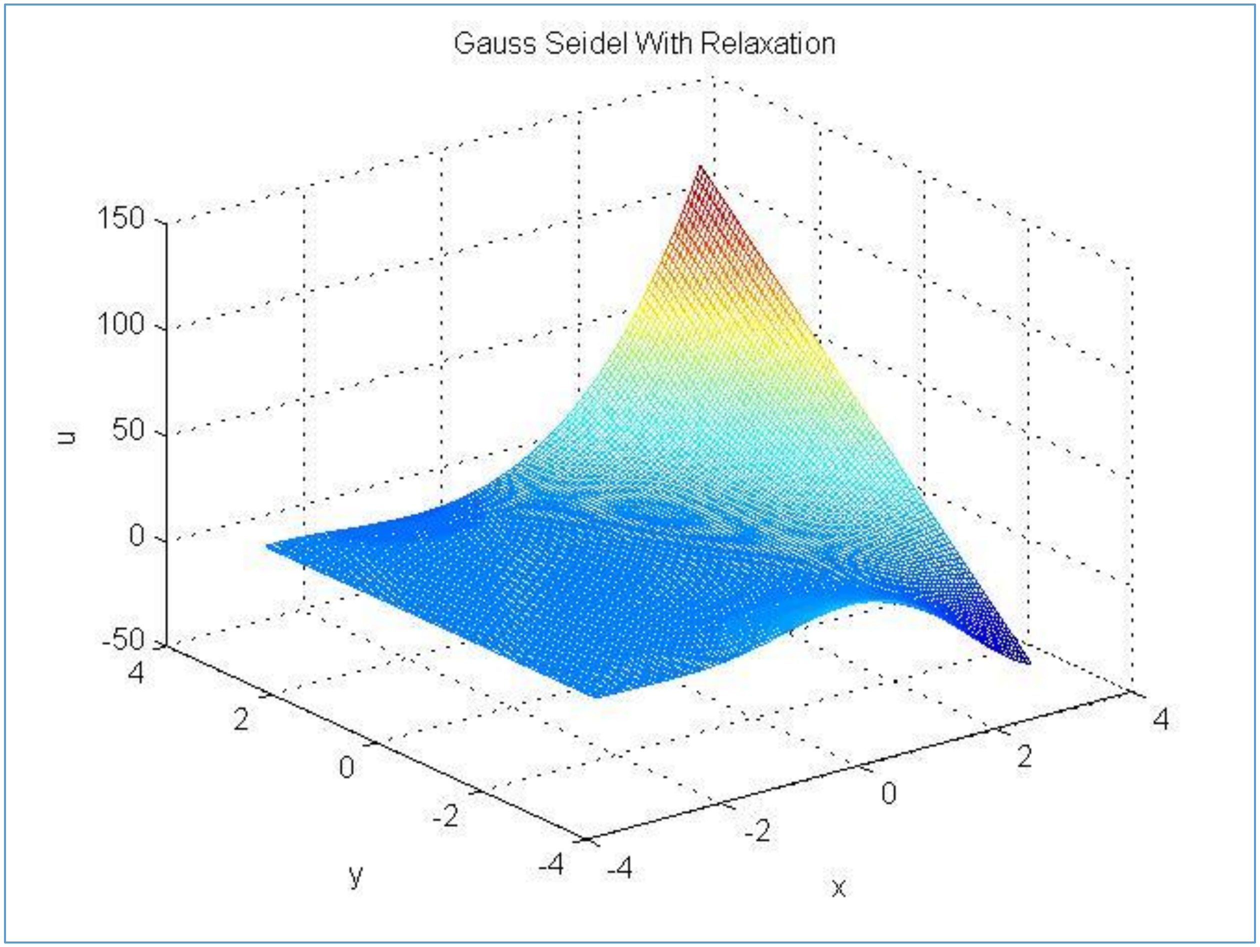


Figure 15 Gauss Seidel 100 x 100 Relaxed

Gauss Seidel No Forcing Results

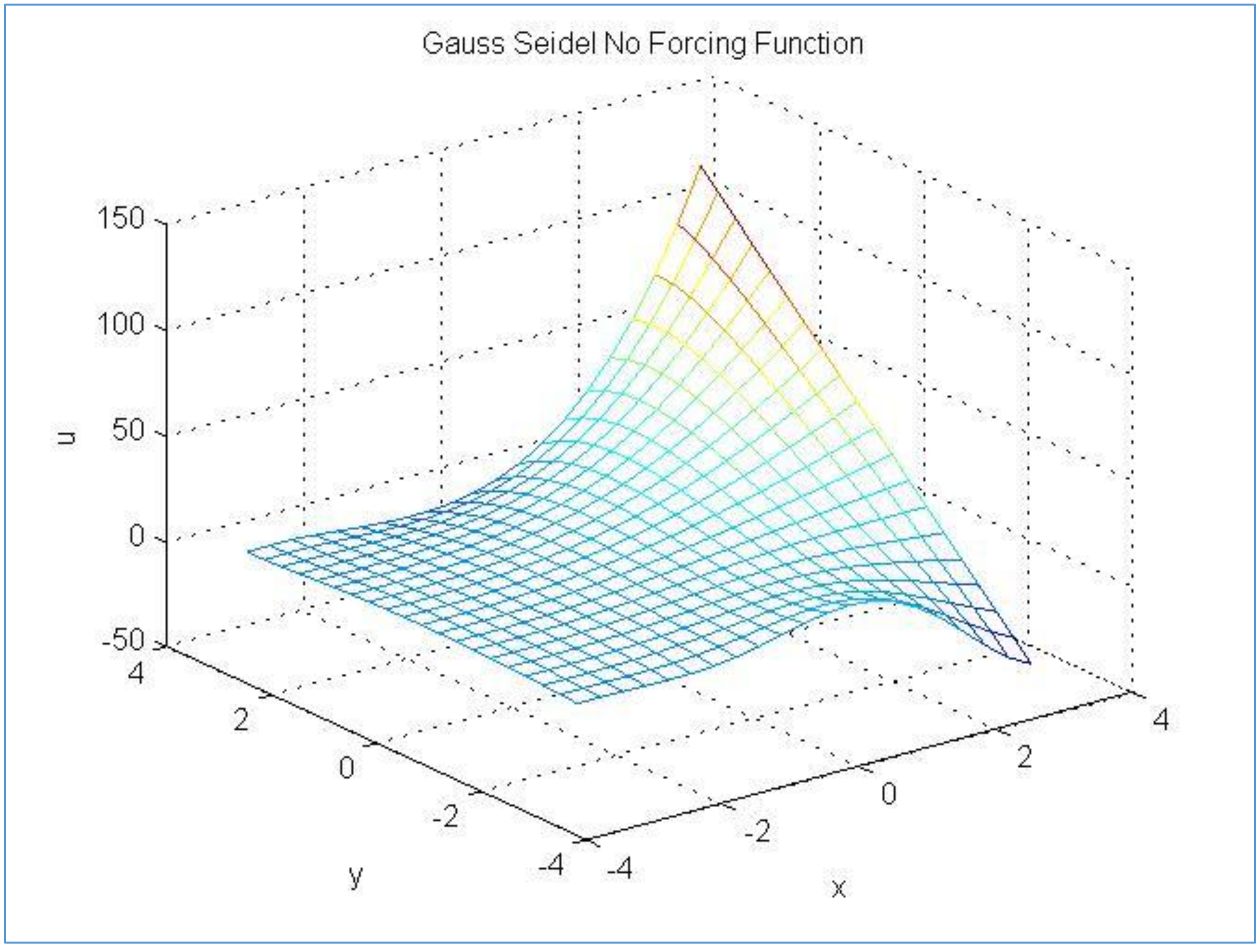


Figure 16 Gauss Seidel 20 x 20 No Forcing

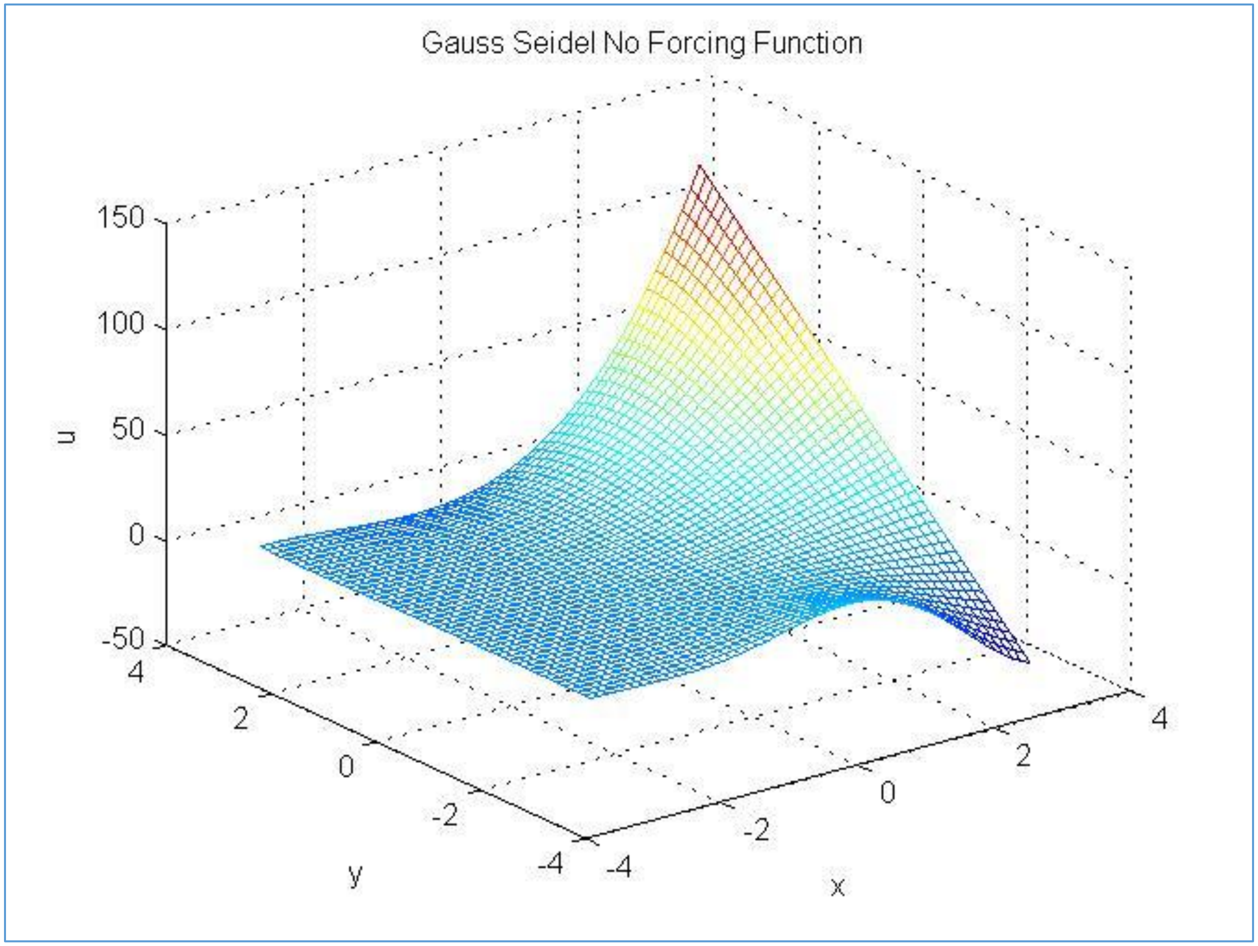


Figure 17 Gauss Seidel 50 x 50 No Forcing

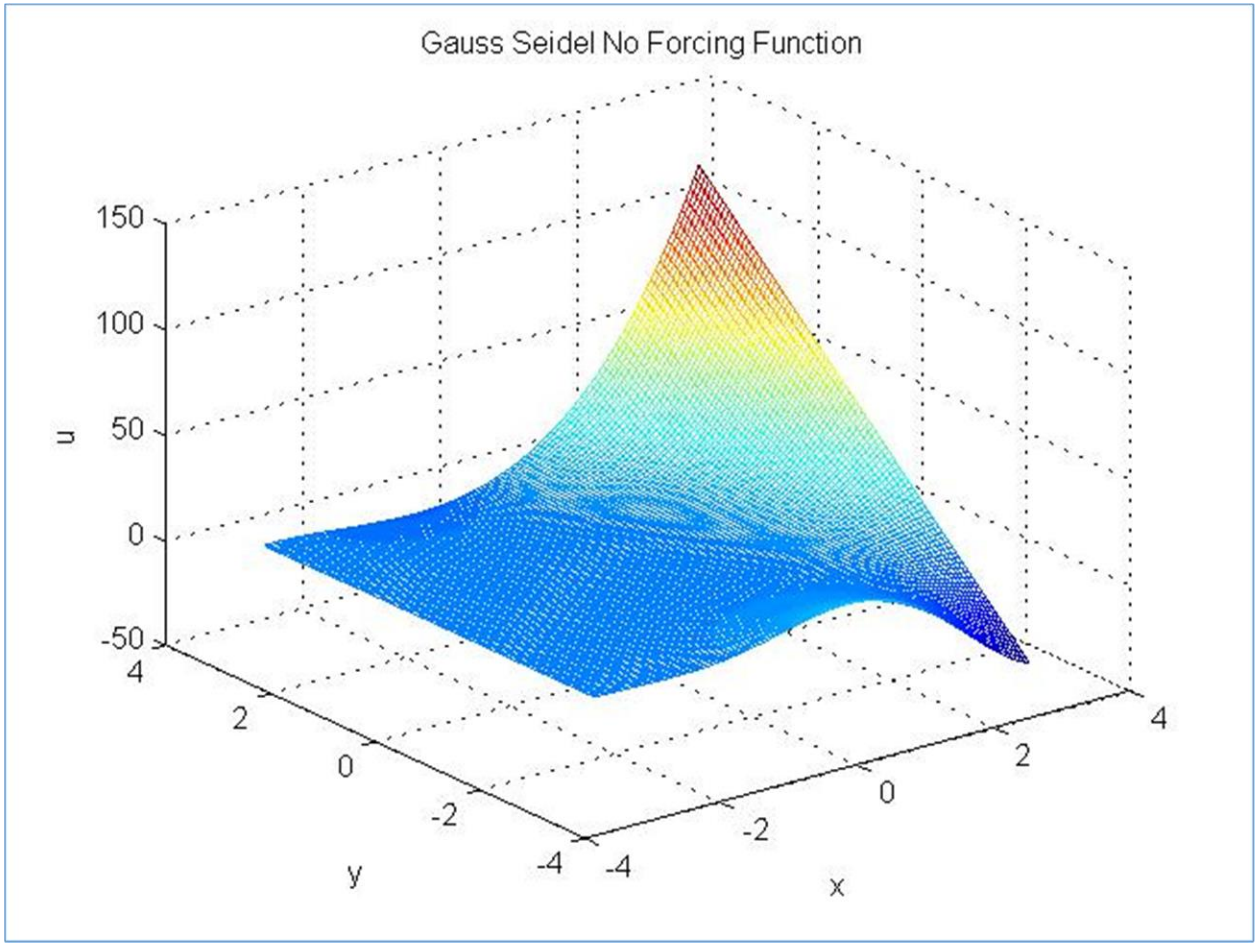


Figure 18 Gauss Seidel 100 x 100 No Forcing

Difference Between Results

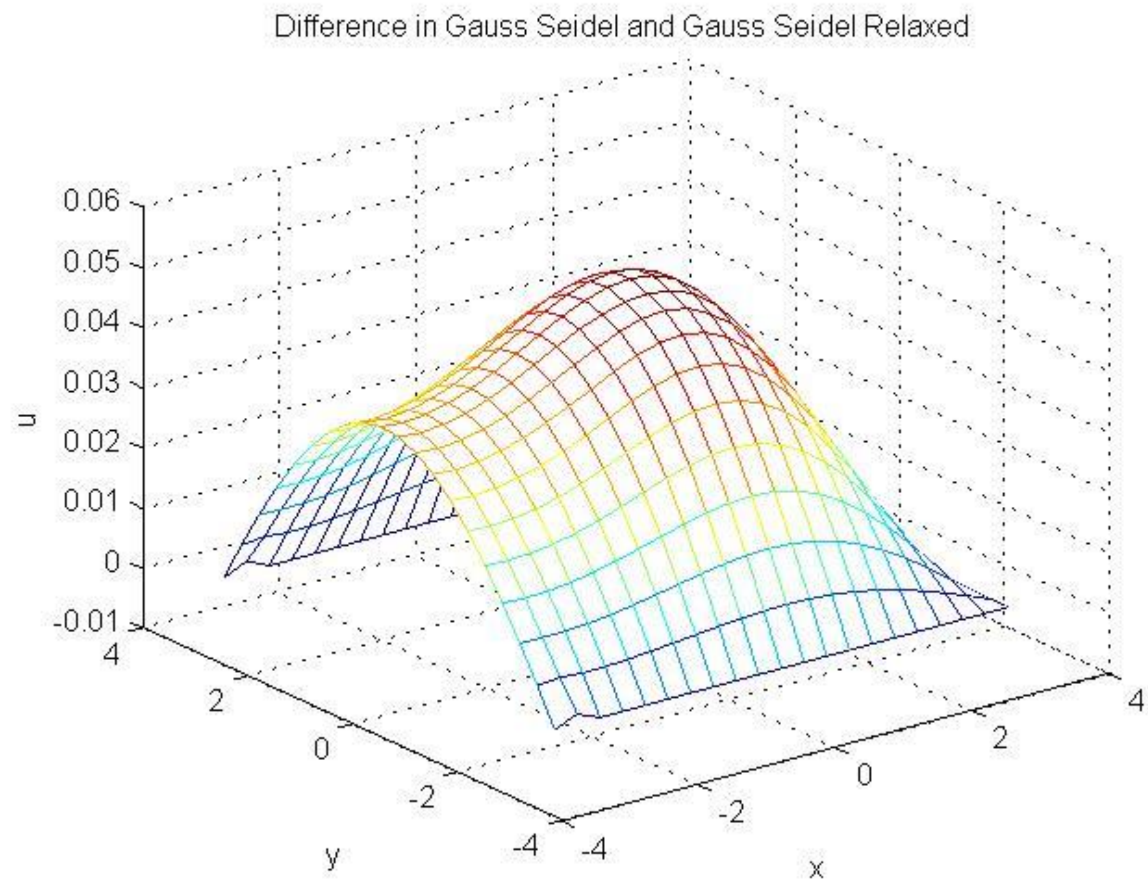


Figure 19 20 x 20 Difference

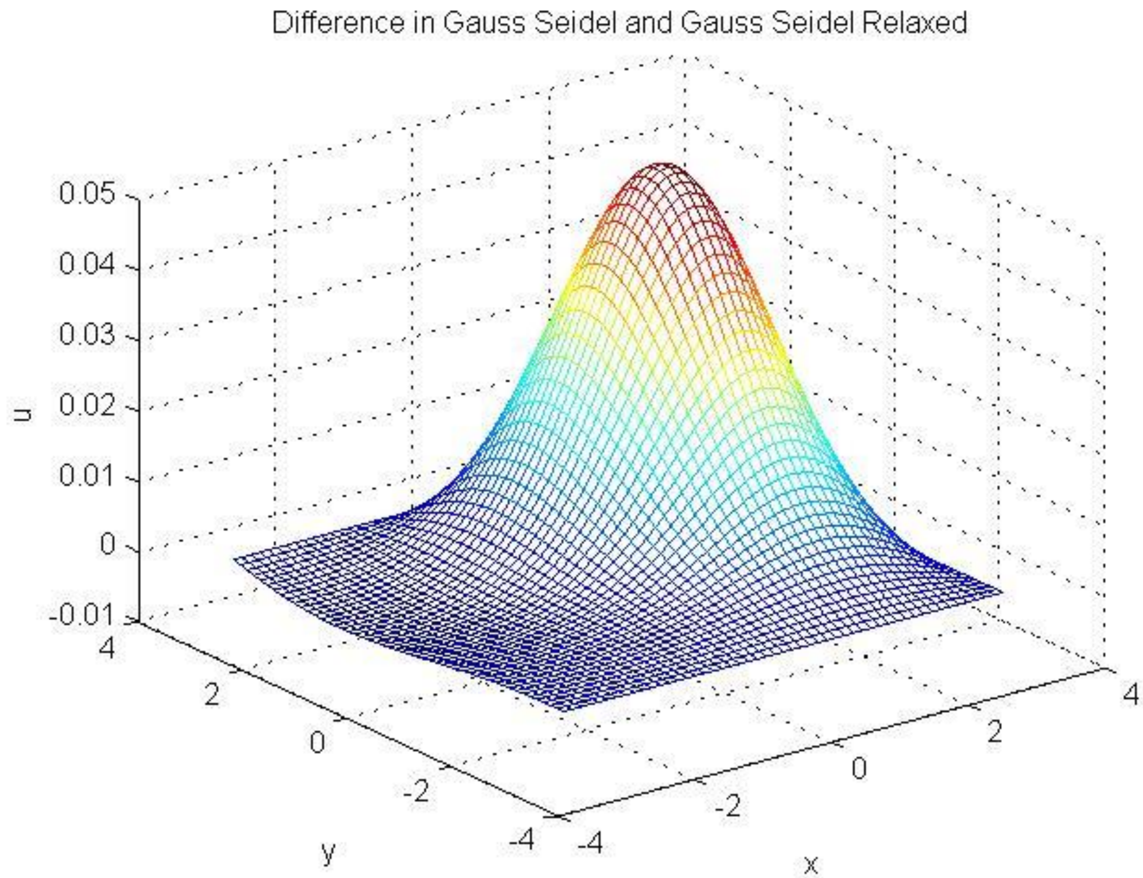


Figure 20 50 x 50 Difference

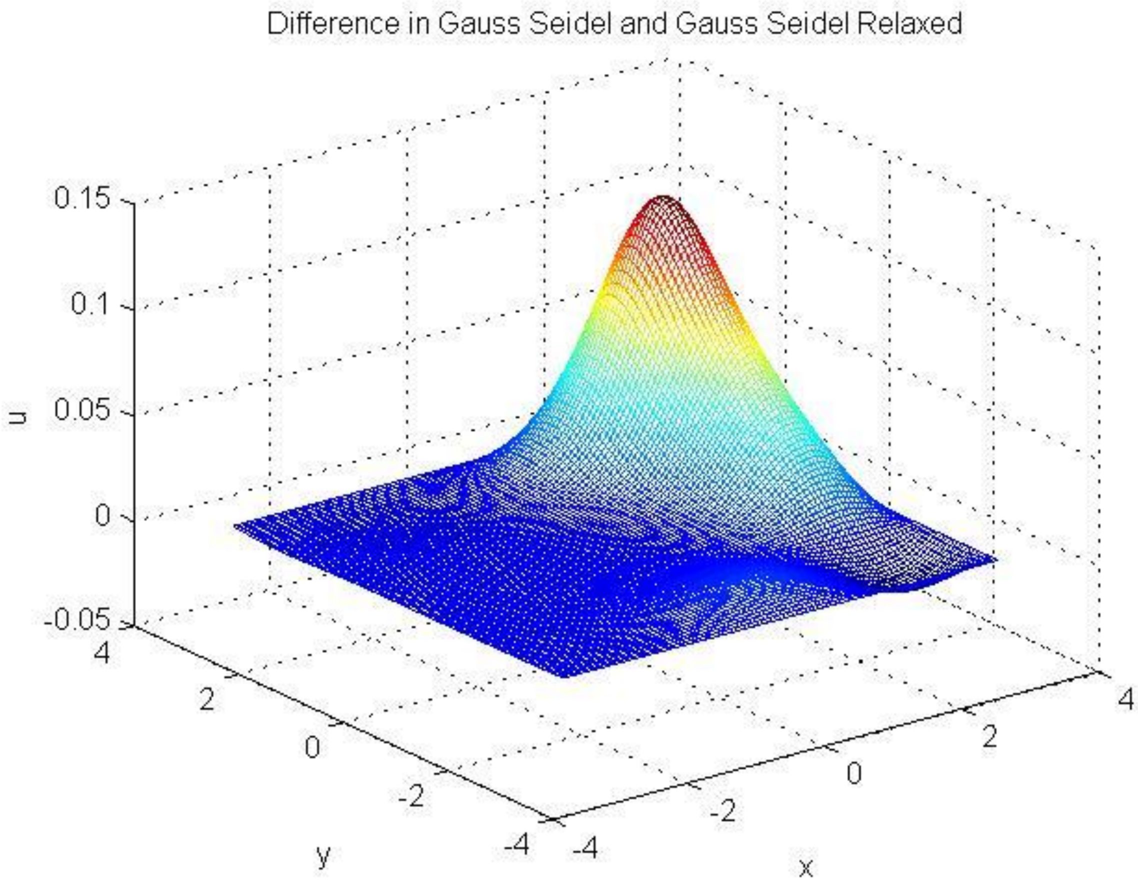


Figure 21 100 x 100 Difference