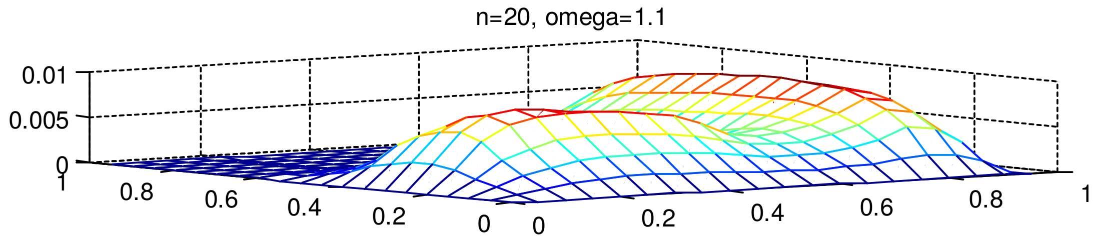
Nicole Deyerl
Numerical Methods
12/9/15

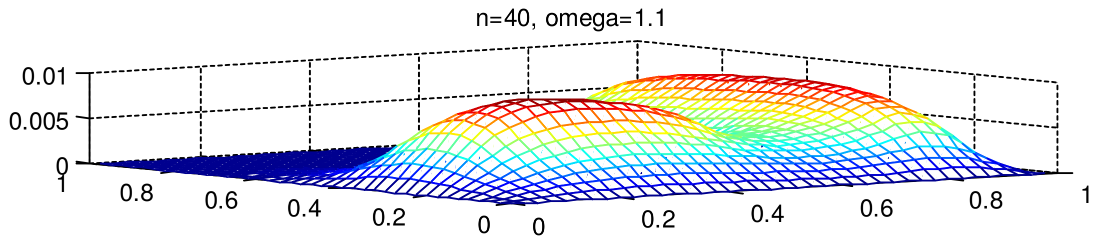
Project 4

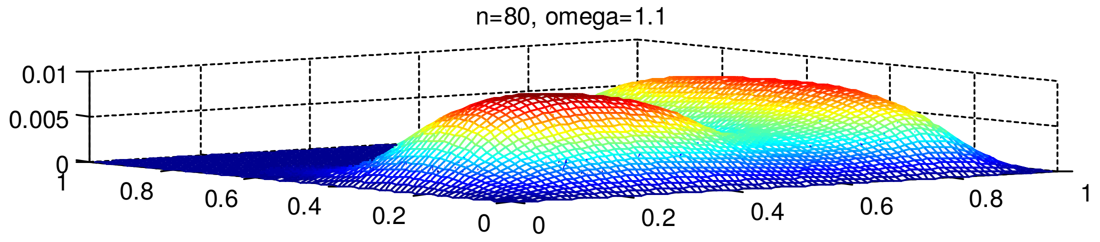
```
function [u,x,y,eps]=poisson(n,nIter,omega)
h=1/n;
x=linspace(0,1,n+1);
b=[];
for i=1:n
    for j=1:n
    b(i,j)=(h.^2)*fcn(x(i),x(j));
    end
end

u=zeros(n,n);
eps=[];
for k=1:nIter
    uold=u;
    for i=2:n-1
        for j=2:n-1
            u(i,j)=(1-omega)*u(i,j)+omega*((1/4)*(b(i,j)+u(i-1,j)+u(i+1,j)
+u(i,j-1)+u(i,j+1)));
            u(i,j)=u(i,j)*inGeom(i,j,n);
        end
    end
    eps(k)=max(max(abs(u-uold)));
end
[x,y]=meshgrid(0:h:1-h,0:h:1-h);

function f=fcn(x,y)
if (x>=.2) && (x<=.4) && (y>=.1) && (y<=.5)
  f=1;
elseif (x>=.1) && (x<=.9) && (y>=.8) && (y<=.9)
  f=1;
else
  f=0;
end
end


function m=inGeom(x,y,n)
    x=x/n;
    y=y/n;
if (x>=.4) && (x<=.8) && (y>=.3) && (y<=.6)
  m=0;
else
  m=1;
end
end


end
```

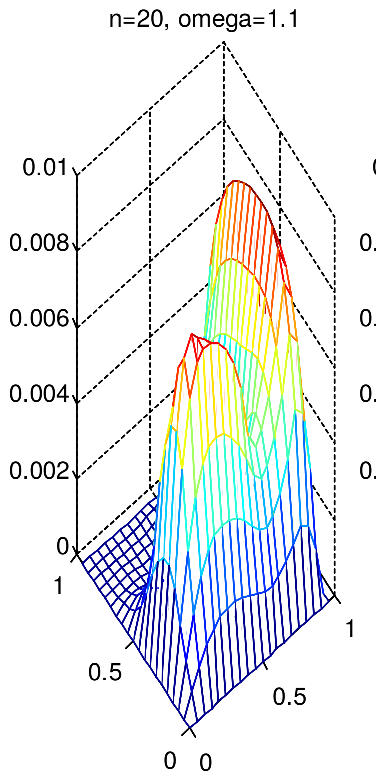
```
% Part 1.
% [u,x,y,eps]=poisson(20,200,1.1);
% subplot(3,1,1); mesh(x,y,u)
% hold on
% [u,x,y,eps]=poisson(40,400,1.1);
% subplot(3,1,2); mesh(x,y,u)
% hold on
% [u,x,y,eps]=poisson(80,800,1.1);
% subplot(3,1,3); mesh(x,y,u)
% axis([0 1 0 1 0 .01])
% Part 2.
% counter=0;
% eVec=[];
% for omega=1:.05:2
% counter=counter+1;
% [u,x,y,eps]=poisson(40,400,omega);
% eps=eps';
% eVec(:,counter)=eps;
% end
% semilogy(eVec)
```

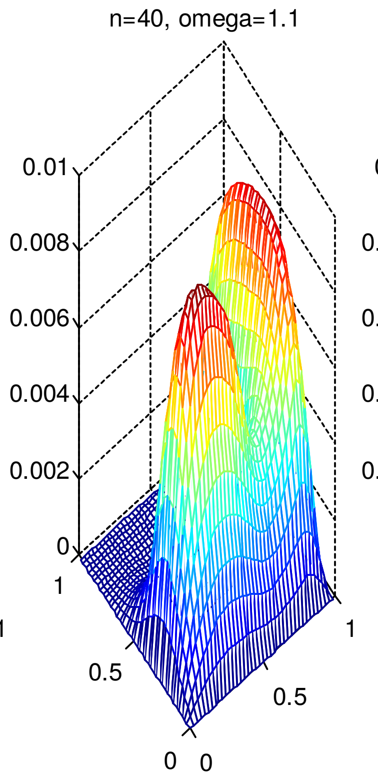
Part I.

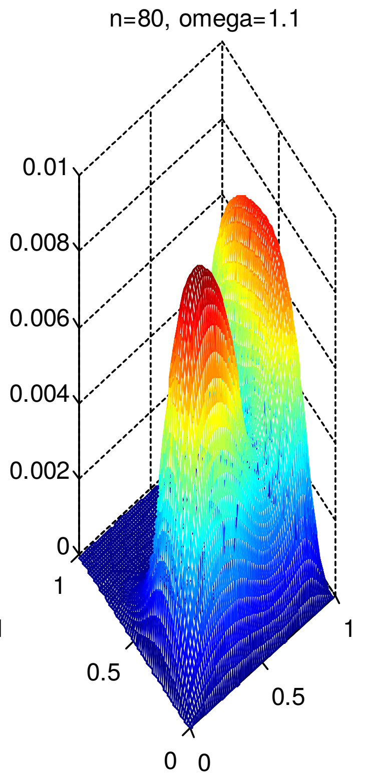

n=20, omega=1.1
0.01
0.005
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


n=40, omega=1.1
0.01
0.005
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

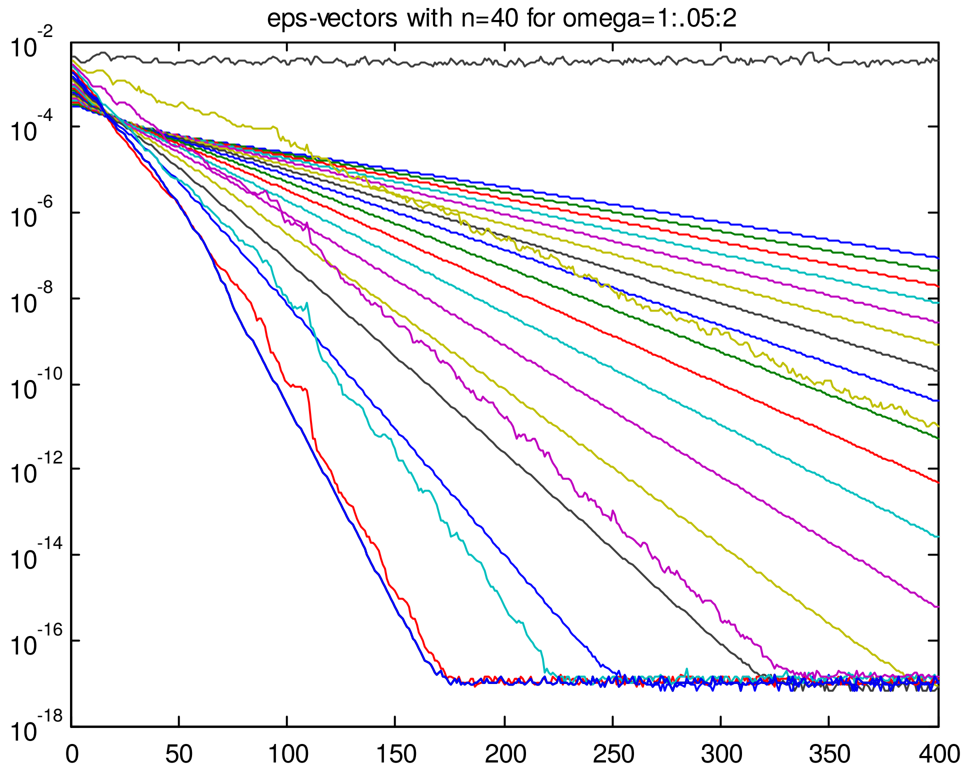

n=80, omega=1.1
0.01
0.005
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

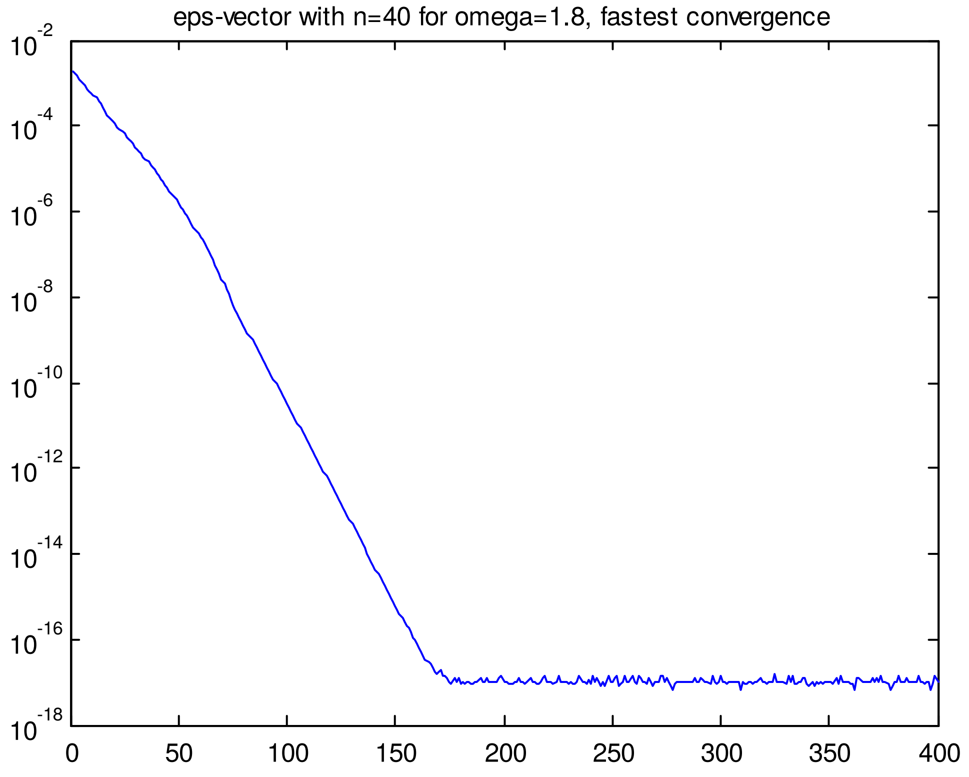

n=20, omega=1.1
0.01
0.008
0.006
0.004
0.002
0
1
0.5
0
0
0.5
1


n=40, omega=1.1
0.01
0.008
0.006
0.004
0.002
0
1
0.5
0
0
0.5
1


n=80, omega=1.1
0.01
0.008
0.006
0.004
0.002
0
1
0.5
0
0
0.5
1

Part II.


eps-vectors with n=40 for omega=1:.05:2
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
10^-16
10^-18
0
50
100
150
200
250
300
350
400


eps-vector with n=40 for omega=1.8, fastest convergence
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
10^-16
10^-18
0
50
100
150
200
250
300
350
400

Part I.

The solution mesh converged independently of n and for n=20, 40 and 80 the mesh graphs appear to be very similar and have the same max peaks, etc. But the graphs got much smoother as n increased due to having more nodes/data to graph/depict. For greater iterations and matrix size (when nIter=800 for n=80) the function took longer to compute u, x and y, which was due to the number of flops (and thereby computation time) being dependent on the size of the matrix.

Part II.

Convergence, or the minimization of eps, the difference between the kth and k-1th step of u, was fastest with an omega value of 1.8. Omega values greater than 1 represent over-relaxation, the goal of which is to make the convergence happen as fast as possible, and omega equal to 1 is the same as doing Gauss Seidel instead of SOR. By speeding up convergence we will achieve a minimal difference between the kth and k-1th step of u in the fewest iterations possible, since once u(k) and u(k-1) are very close, our final answer cannot get much better. SOR converged with increasing speed from omega equal to 1 to omega equal to 1.8, when convergence began to decrease in speed. If we wanted to really increase the speed of convergence, maybe we could use a variable omega method which alters omega depending on the difference between u(k) and u(k-1). We could also save computational time by exiting the SOR program once eps is below a certain error tolerance.