BME 7310 Computational Laboratory #3

Due: 9/22/2023, midnight

Problem #1. Understanding Current: Conservation of cortical current is governed by the PDE

$$\nabla \bullet \vec{J} = 0 \qquad (1)$$

where $\vec{J}$ is the electrical current density. Often $\vec{J}$ is expressed with respect to tissue potential changes, this can be expressed as the gradient of a scalar potential Φ, and electrical conductivity σ.

$$\vec{J} = -\sigma \nabla \Phi \qquad (2)$$

Hence equation (1) can be recast in terms of Φ as,

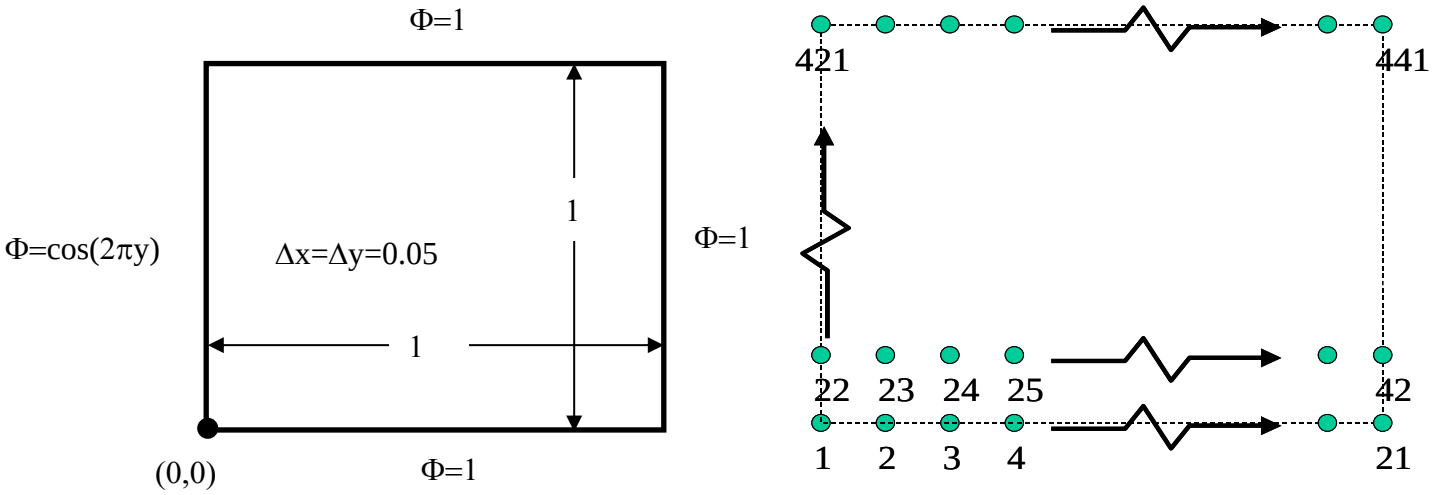
$$\nabla \bullet (-\sigma \nabla \Phi) = 0 \qquad (3)$$

under homogeneous electrical conductivity, it can be written as:

$$\nabla^2 \Phi = 0 \qquad (4)$$

Your job is to compute *point iterative solutions* of (4) when discretized by center finite differences under the following scenarios:

Given the problem domain and boundary conditions below, compute the solution of Φ using the iterative method below with an initial solution vector of Φ=0 everywhere. Iterate until reaching an *absolute L_∞ norm-based error* of successive iterates of less than $1x10^{-5}$ and report the number of iterations needed to reach this convergence criterion. Estimate the spectral radius of the iteration method during the course of the iterations and compare with the theoretically expected value. Plot contours of your solution over the computational domain and report the actual numerical value of Φ for at the point x=0.7 and y=0.7 at convergence.



(a) Perform the above simulation with the point iterative method SOR and determine the theoretical optimal ω. Using this value in the iteration. Be sure to report the solution values requested above in order to verify that your solution is essentially unchanged from

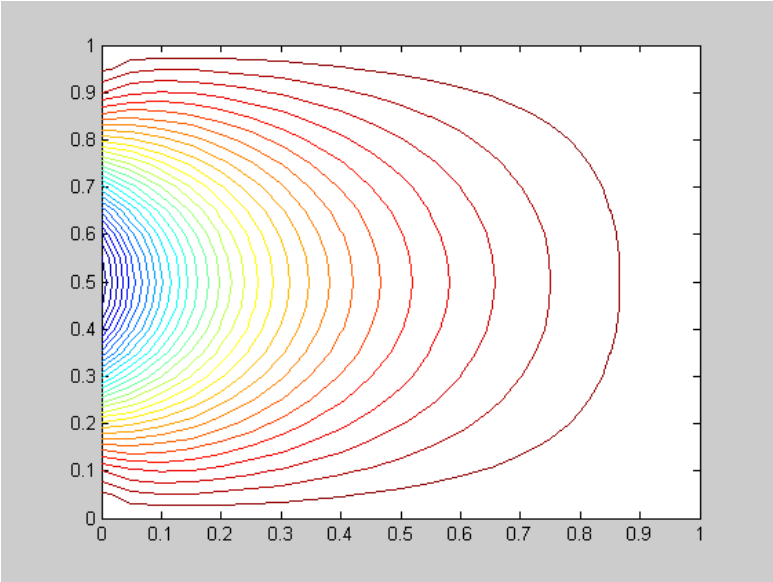
previous work that used *Jacobi* and *Gauss Seidel*. Also, perform a series of simulations that enable you to plot iteration count as a function of ω to confirm whether the theoretical optimum is the same as that found in practice for this problem. Is the speed up in terms of convergence rate relative to Gauss-Seidel in agreement with theory?

(b) In previous work, you solved the above using the point interative methods *Jacobi*, *Gauss Seidel*, and now SOR. Specifically, for each method list: spectral radius, the number of iterations, the computational effort (in terms of number of multiplies and divides) per iteration, and the total computational cost to reach a solution. Rank the methods according to iteration count and then according to toal computational cost. Which method do you comes out on top?

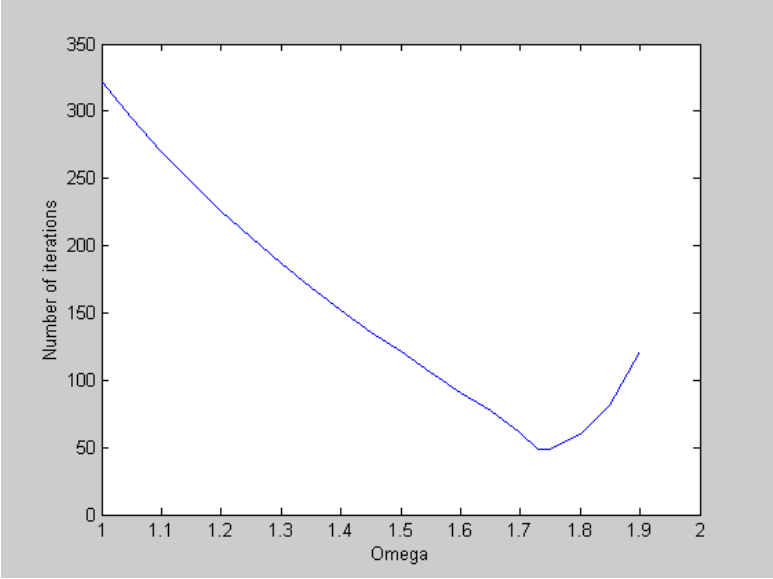
Problem #1 SOLUTION:

(a) Repeat part (a) with SOR by determining the theoretical optimal ω and using this value in the iteration. Be sure to report the tabular values of the solution requested in part (a) in order to verify that your solution is essentially unchanged. Also, plot iteration count as a function of ω to confirm whether the theoretical optimum is the same as that found in practice for this problem. Is the speed up in terms of convergence rate relative to Gauss-Seidel in agreement with theory?

$\omega_{opt}=2/(1-\sqrt{(1-\rho_{Gauss\text{-}Seidel}}))=1.7293$
$\rho_{SOR}=1.7293-1=0.7293$
$\rho_{estimate}=0.8339$
SOR # of iterations = 49
$\Phi(0.7,0.7)=0.870227$



Plotting total number of iteration to convergence as a function of the relaxation parameter ω. Numerical result confirms the result found analytically.



With respect to convergence rate, we expect the ratio of
Analytic: $R(G_{SOR})/R(G_{Gauss\text{-}Seidel})=-\log(\rho_{sor})/-\log(\rho_{GS})=12.7$. You should use the analytic representation of spectral radius for GS and then use the optimum SOR expressions to determine that.
Estimated based on spectral radius: -log(.8339)/(-log(0.97553))=7.33
Our experimental ratio: is 322/49 = 6.6. Not quite as good an approximation to analytic but is consistent with numerical ratio.

Also, you can do a little better than this with w=1.74.

(b) Now tabulate your results from parts (a) through (c) and analyze the performance of these methods for this problem. Specifically, for each method list: spectral radius, the number of iterations, the computational effort (in terms of number of multiplies and divides) per iteration, and the total computational cost to reach a solution. Rank the methods according to iteration count and then according to toal computational cost. Which method do you prefer?

Method	Spectral Radius	# of iterations	Effort per iteration	Total Effort
Jacobi	0.9877	582	$5(N-2)^2=1805$	1050510
Gauss-Seidel	0.97553	322	$5(N-2)^2=1805$	581210
SOR	0.8339	49	$6(N-2)^2=2166$	106134

Where N=21, i.e. NxN grid.
Although SOR requires more operations per iteration, it's speed in convergence more than makes up for the deficit. Interestingly, for no added cost per iteration, Guass-Seidel's total effort is half that of Jacobi. To rank these methods from best to worst in terms of total effort, SOR, Gauss-Seidel and Jacobi.

Also, the above is a little loose… depending on how you arrange your system, it could be different, the order of change and the order itself is what I am looking for.

```
% SOR METHOD
clear;
h=0.05;
y=[0:.05:1]';

A=zeros(21,21);
for i=1:21
  A(i,21)=1;
  A(1,i)=1;
  A(21,i)=1;
  A(i,1)=cos(2*pi*(i*h-0.05));
end
Aold=A;
w=1.74;
error=1;
itr=0;
while (error > 1e-5 & itr < 10000)
   itr=itr+1;
   for i=2:20
        for j=2:20
         A(i,j)=w/4*(A(i-1,j)+Aold(i+1,j)+A(i,j-1)+Aold(i,j+1)) + (1-
w)*Aold(i,j);
     end
    end
   errorold=error;
   error=max(max(abs(A-Aold)));
   errornew=error;
   spectral(itr)=errornew/errorold;
   Aold=A;
end

fprintf('SOR Iterations %d\n',itr);
figure(5);
contour(A);
figure(6);
plot(spectral);

% SEARCH FOR OPTIMAL OMEGA FOR SOR METHOD
clear;
dw=0.02;
w=1;
for mmm=1:50

h=0.05;
y=[0:.05:1]';

A=zeros(21,21);
Aold=zeros(21,21);
for i=1:21
  A(i,21)=1;
  A(1,i)=1;
  A(21,i)=1;
  A(i,1)=cos(2*pi*(i*h-0.05));
end

error=1;
```

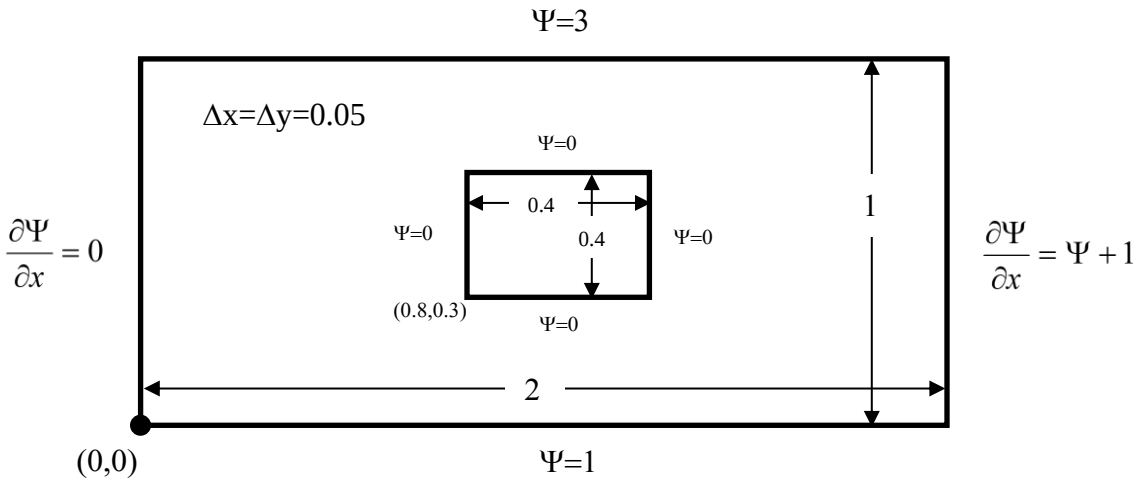
```
itr=0;
while (error > 1e-5 & itr < 10000)
   itr=itr+1;
   Aold=A;
   for i=2:20
        for j=2:20
         A(i,j)=w/4*(A(i-1,j)+Aold(i+1,j)+A(i,j-1)+Aold(i,j+1)) + (1-
w)*Aold(i,j);
     end
    end
   errorold=error;
   error=max(max(abs(A-Aold)));
   errornew=error;
   spectral(itr)=errornew/errorold;
end

WWW(mmm,1)=w;
NNN(mmm,1)=itr;
w=w+dw;
end
figure(7);
plot(WWW,NNN);
ylabel('Number of iterations');
xlabel('Omega');
```

Problem #3. Stream Function (10 pts):

(a) A more interesting problem is shown below for flow around an obstruction. The model has been written in an alternative idealized flow formulation in terms of the stream function Ψ which also satisfies Laplace's equation.

$$\nabla^2 \Psi = 0 \qquad (4)$$

Use whatever method you prefer and solve the problem. Plot contours of your solution and report values at the point x=0.5, y=0.5. Hint: A simple way to create the effect of the inner square is to reset Ψ=0 at these node positions for each iteration.

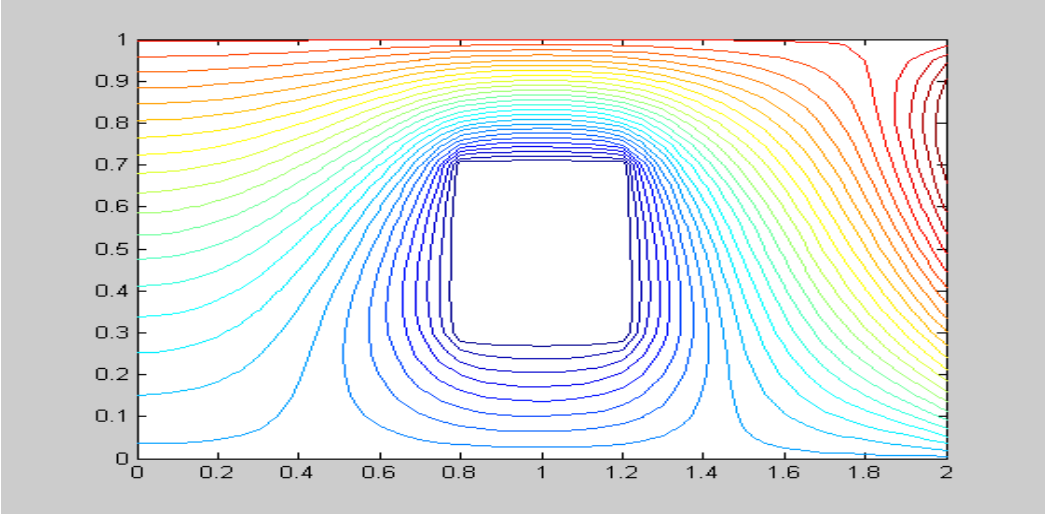


(b) For the stream function formulation of ideal flow, the following is true:

$$\boldsymbol{V}_x = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{y}}, \boldsymbol{V}_y = -\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}} \qquad (5)$$

Using this description, plot the velocity vectors with the 'quiver' command. Explain the results you see with respect to plot in part **(a)**. In light of equations (4), and (5) as compared to that of equations (2) and (3), discuss the differences in these formulations regarding the model being used.

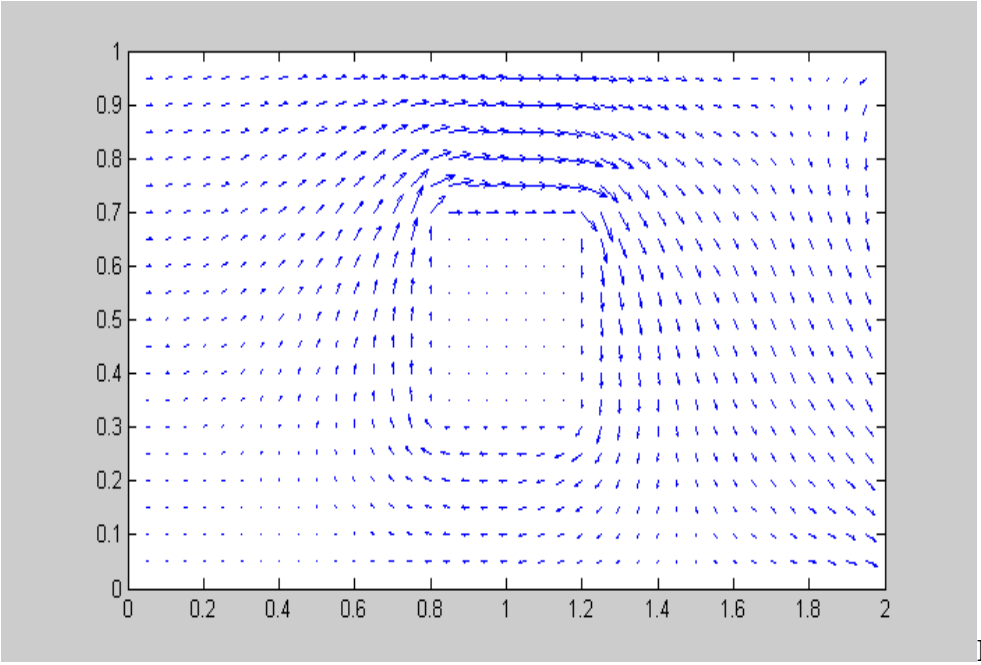
(a)



C

Φ(0.5,0.5)=1.1410, ω=1.785, # of iterations=72

(b)



D

The velocity 'potential' isocontours (problem #1 physics) are orthogonal to the streamlines, i.e. if we were to be solving the potential problem of #1 with the boundary conditions of problem #2, the quiver plot shown above would be crossing potential lines orthogonally in the potential solution. As we can see in problem #2 streamlines, the stream function follows the vectors nicely. So streamlines actually indicate flow direction and where streamlines are concentrated such as at the top of the obstruction, this indicates large flow.

More specifically, as opposed to the potential where concentrated contours indicate rapid flow across the contours, concentrated contours in the stream function indicate rapid flow along the contours.

CODE FOR PROBLEM 3:

```
h=0.05;
y=[0:.05:1]';

Aold=zeros(21,41);
for i=1:41
  Aold(21,i)=3;
  Aold(1,i)=1;
end
A=Aold;
w=1.785;
error=1;
itr=0;
while (error > 1e-5 & itr < 10000)
   itr=itr+1;
   for i=2:20
      j=1;
      A(i,j)=w/4*(2*Aold(i,2)+Aold(i+1,j)+A(i-1,j))+(1-w)*Aold(i,j);
      for j=2:40
         if i>=7 & i<=15 & j>=17 & j <=25
            A(i,j)=0;
         else
            A(i,j)=w/4*(Aold(i+1,j)+Aold(i,j+1)+A(i-1,j)+A(i,j-1))+(1-w)*Aold(i,j);
         end
      end
      j=41;
      A(i,j)=w/(4-2*h)*(2*A(i,j-1)+A(i-1,j)+Aold(i+1,j)+2*h)+ (1-w)*Aold(i,j);
   end

   errorold=error;
   error=max(max(abs(A-Aold)))
   errornew=error;
   spectral(itr)=errornew/errorold;
   Aold=A;
end

xi=[0:.05:2];
yi=[0:0.05:1];
figure(1);
contour(xi,yi,A,30);
```

```
 Vy=-(A(2:20,3:41)-A(2:20,1:39))/2/h;
 Vx=(A(3:21,2:40)-A(1:19,2:40))/2/h;

 c1=0;
 for i=1:19
   for j=1:39
    c1=c1+1;
     xpos(c1)=j*.05;
     ypos(c1)=i*.05;
     vx(c1)=Vx(i,j);
     vy(c1)=Vy(i,j);
   end
  end

figure(2);
quiver(xpos,ypos,vx,vy);
```