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,

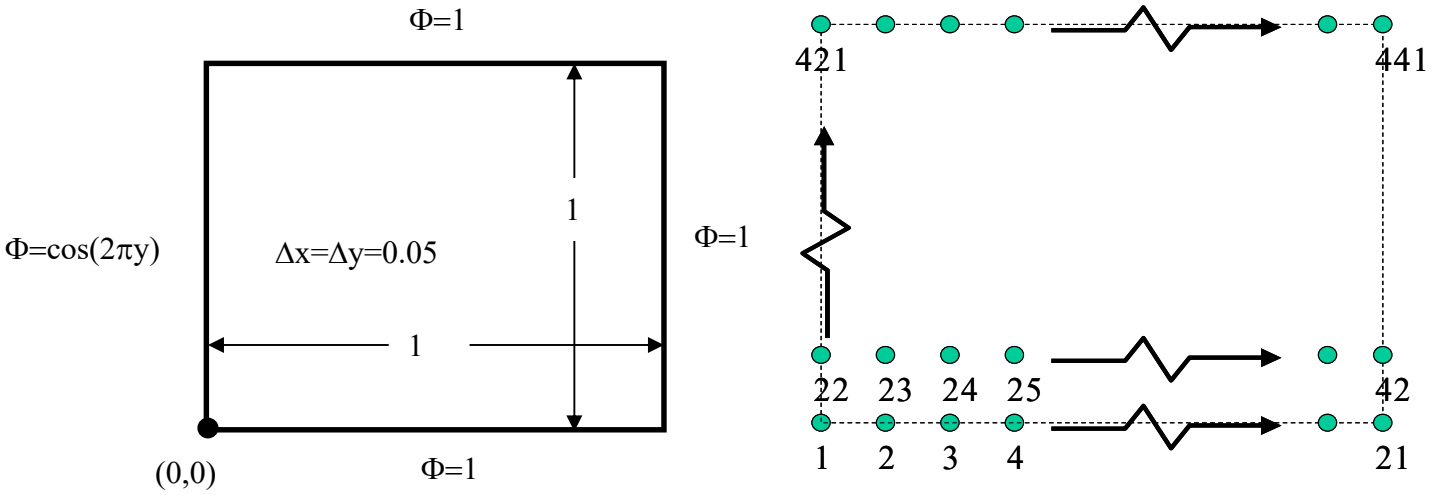
$$\nabla \bullet (-\sigma \nabla \Phi) = 0 \qquad (3)$$

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?

Solution:

- Simulation with the point iterative method SOR and determine the theoretical optimal ω

  First,
  i) we computed convergent state with Gauss Seidel, OR
  ii) use the analytical formula for Gauss Seidel spectral radius
  to get the Gauss Seidel spectral radius. Then, we determine *omega* from the spectral radius of Gauss Seidel.

  Gauss Spectral Radius: 0.975528
  ```
  omega = 2 / (1 + sqrt(1 - gauss_spectral)) = 1.72945381728060
  ```

  Then we run SOR:
  ```
  SOR Iterations 48
  Spectral Radius 0.726608
  ```

- 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*

  Jacobi Iterations 582
  Jacobi V(x==0.7, x==0.7): 0.869710123935

  Gauss Seidel Iterations 322
  Gauss Seidel V(x==0.7, x==0.7): 0.869996373963

  SOR Iterations 48
  SOR V(x==0.7, x==0.7): 0.870227218371

- 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.

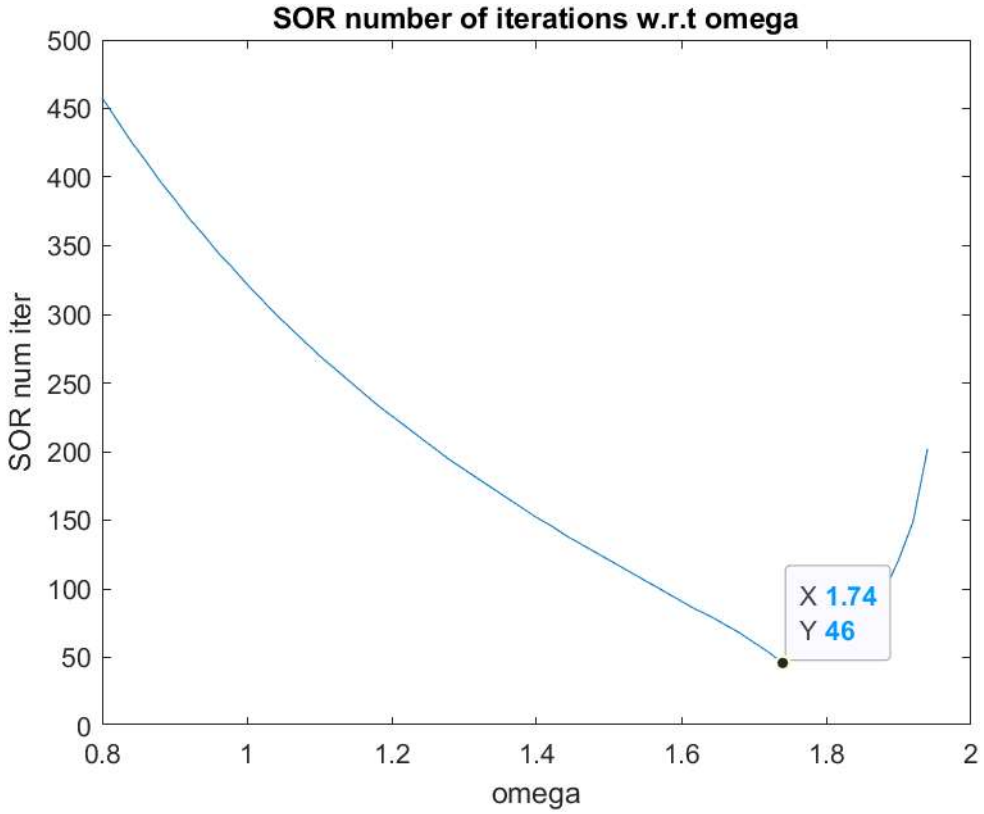


Figure 1. SOR number of iterations vs, omega. Kahan theorem in Burden & Faires states that SOR can converge only if 0 <ω< 2. Range of omega was limited to between 0.8 – 1.95 since going beyond either limits caused the number of iterations to shoot up extremely large, thus scale up the visualization.

```
Clear
% ------------------------------------
%perform a series of simulations that enable you to plot iteration count as
a function of omega to confirm whether
% the theoretical optimum is the same as that found in practice for this
problem
interations_v = zeros(1);
itr=0;

for omega=0.8:0.02:1.95
    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

    error=1;
    itr=0;
    pitr=0;
    while (error > 1e-5 & itr < 10000)
```

```
        itr=itr+1;
        Aold=A;
        for i=2:20
            for j=2:20
                A(i,j)=omega * 1/4 * ( A(i-1,j) + Aold(i+1,j) + A(i,j-1) +
Aold(i,j+1)) ...
                    + (1-omega)*Aold(i,j);
            end
        end

        errorold=error;
        error=max(max(abs(A-Aold)));
        errornew=error;
        spectral(itr)=errornew/errorold;
        pitr=pitr+1;
        if pitr==5
            pitr=0;
        end
    end
    if omega==0.8
        interations_v(end)=itr;
    else
        interations_v(end+1)=itr;
    end
end

figure(5)
plot([0.8:0.02:1.95], interations_v)
xlabel('omega')
ylabel('SOR num iter')
title('SOR number of iterations w.r.t omega')
```

Is the speed up in terms of convergence rate relative to Gauss-Seidel in agreement with theory?

Yes!
SOR has a *theoretical* spectral radius 0.729453817281.
Gauss has a *theoretical* spectral radius 0.975325988997.

Then, the theoretical predicted number of iterations:
N_iter_Gauss ~ log10(1e-5)/log10(0. 975325988997) = 460.8209 iterations
N_iter_SOR ~ log10(1e-5)/log10(0.729453817281) = 36.4958 iterations.

Thus the speed up agrees with *theoretical* predictions.

(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 total computational cost. Which method do you comes out on top?

Answer:

Method	Spectral R	Num iter	Computational effort for iter (for computing domain matrix only)	Total computation cost
Jacobi	0.987764	582	19 * 19 (matrix size)	19 * 19 * 582
Gauss	0.975528	322	19 * 19 (2 loops)	19 * 19 * 322
SOR	0.726608	48	19 * 19 (2 loops)	19 * 19 * 48

SOR wins! in both 1) number of iterations & 2) total computation cost.

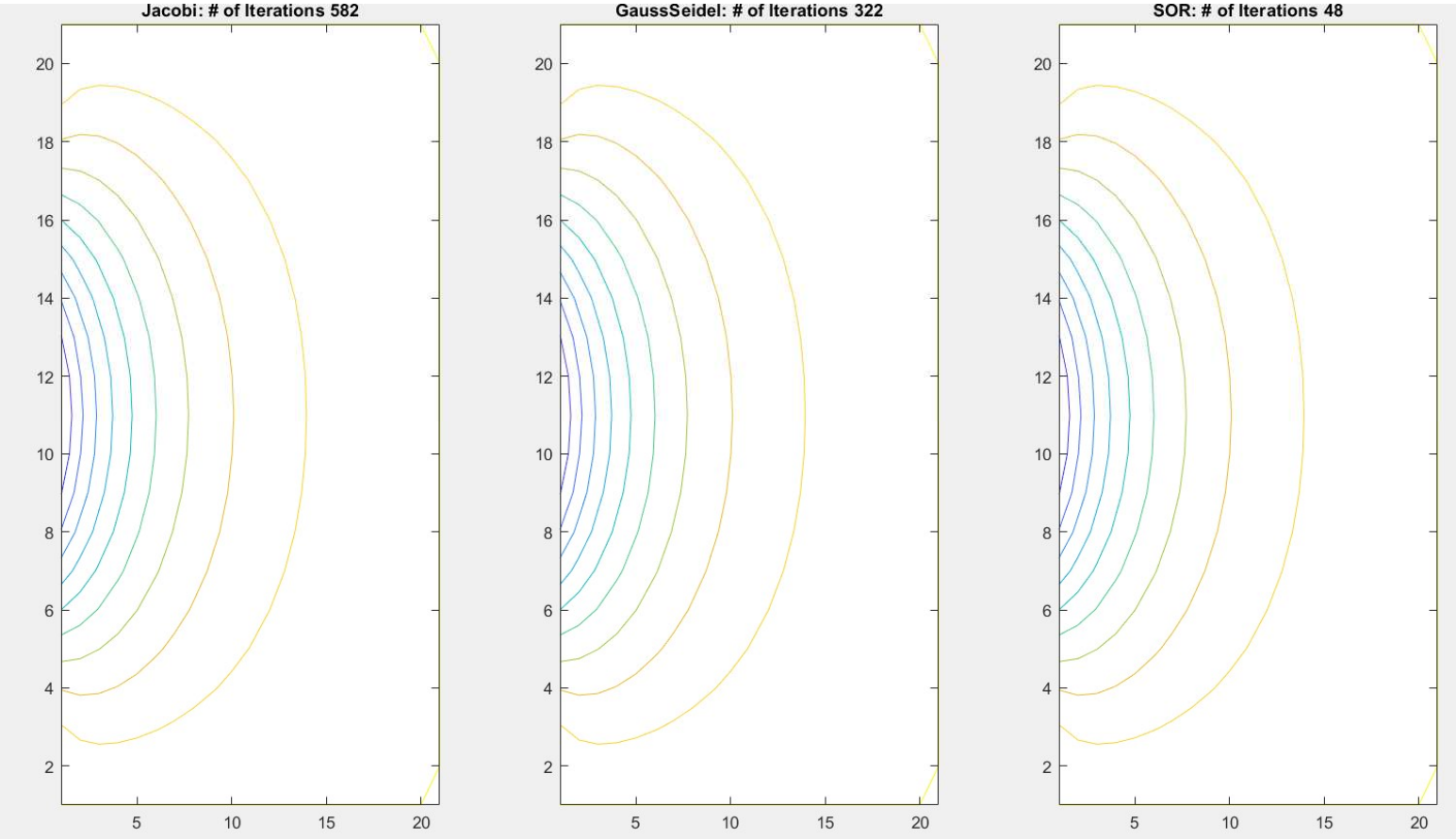


Figure 2. Contours of the domain for each method.

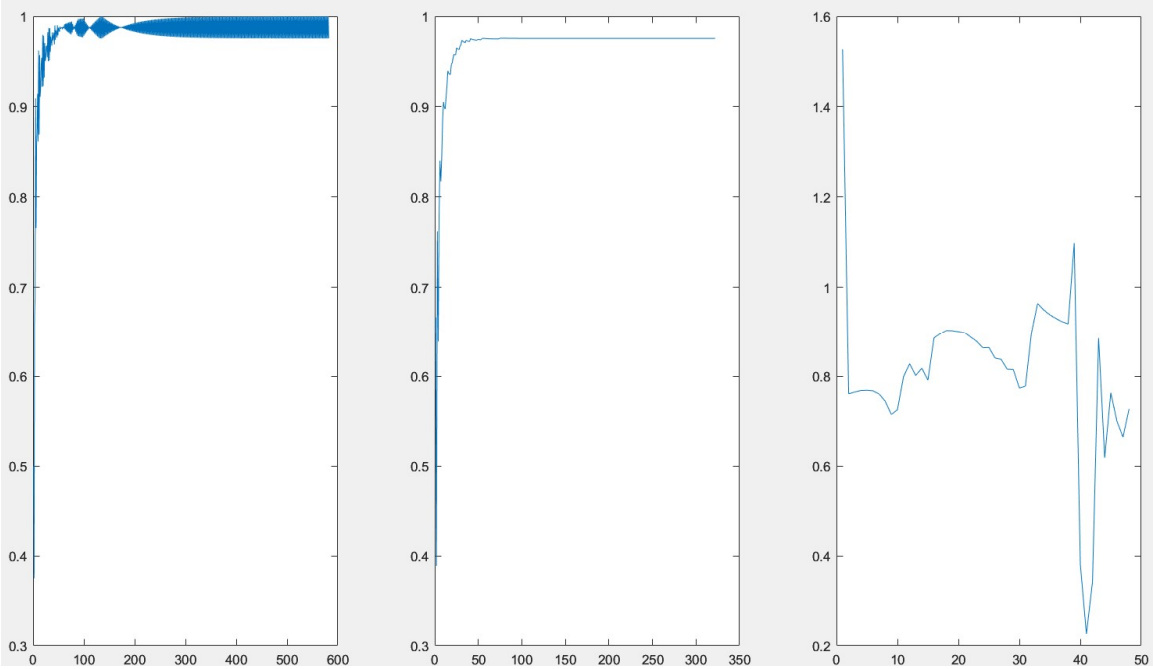


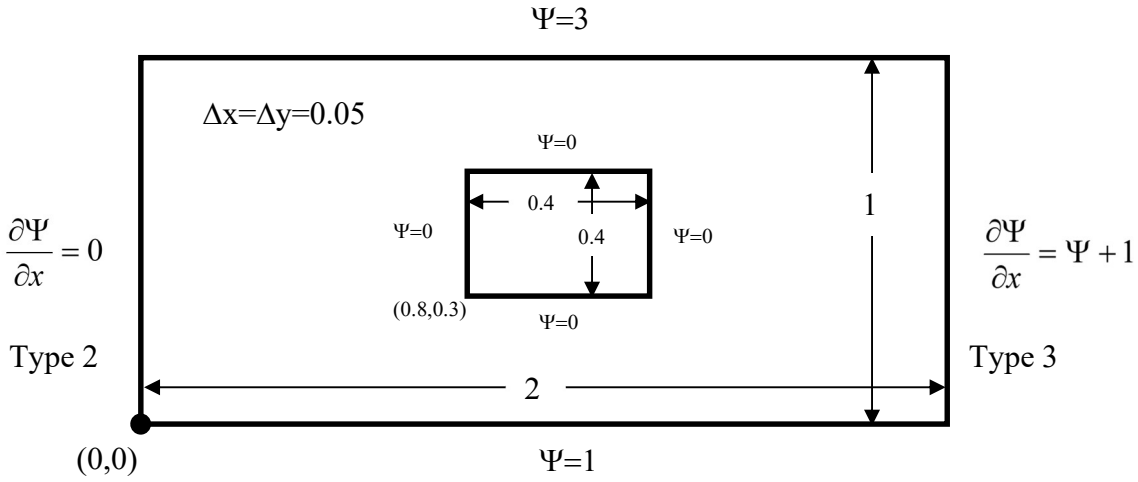
Figure 3. Spectral radius: Jacobi (left), Gauss Seidel (mid), SOR (right).

Problem #2. Stream Function:

(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.

$$\nabla^2\Psi = 0 \qquad (5)$$

Use whatever *point iterative* method you prefer and solve the problem below**. 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.*



Solution: Here Gauss Seidel was implemented.

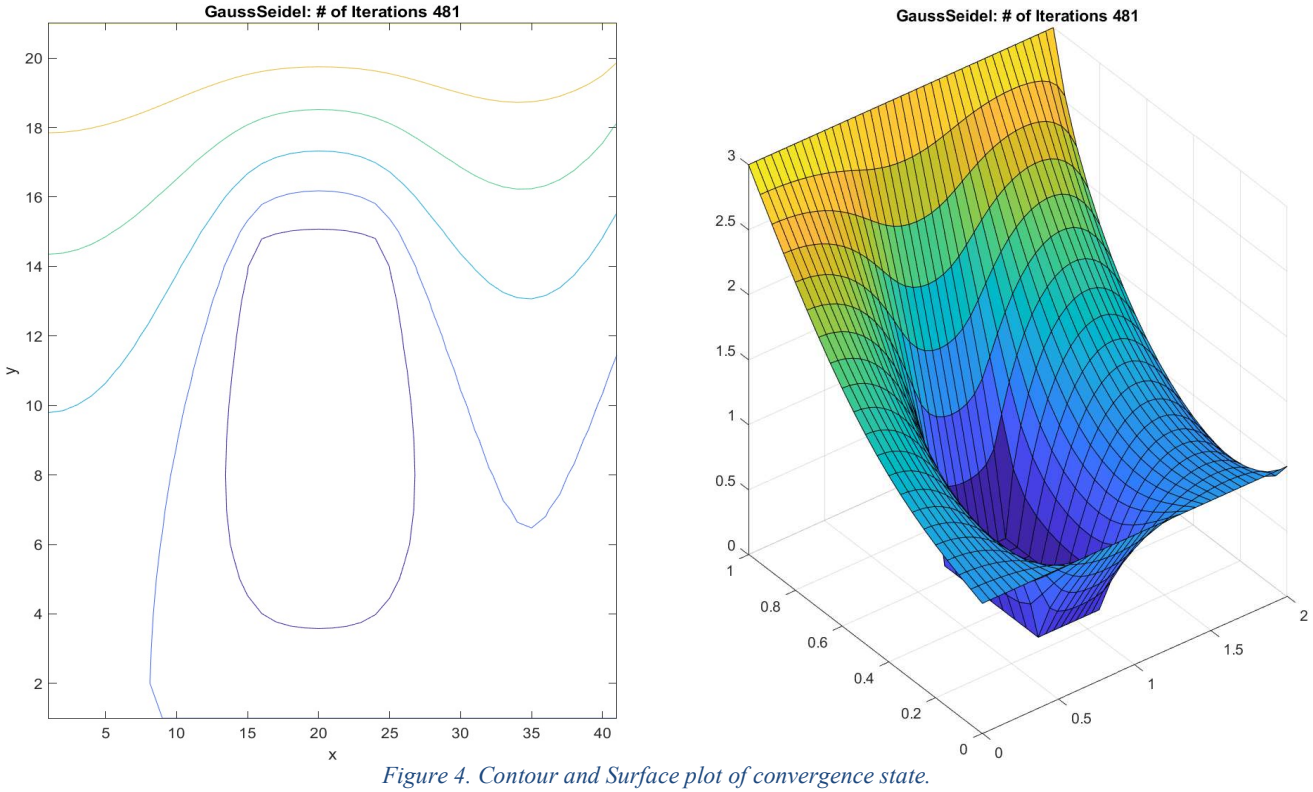


Figure 4. Contour and Surface plot of convergence state.

Value at point x=0.5, y=0.5: 1.072052267354

```
figure(1); subplot(1,2,1);subplot(1,2,2);
figure(2);

% Gauss-Seidel
clear;
h=0.05;
x=[0:h:2];
y=[0:h:1];
A=zeros(length(y), length(x));
%% ---- BC -------
A(2:end-1,1) = (1/4) * ( A(1:end-2, 1) + A(3:end, 1) + 2*A(2:end-1, 2));
A(2:end-1,end) = 1/(4+2*h) * ( A(1:end-2, end) + A(3:end, end) + 2*A(2:end-
1, end-1) - 2*h); % type3;
A(1,:)=1;
A(end,:)=3;
% ---- obstruction -----
A(6:14,16:24) = 0;

error=1;
itr=0;
pitr=0;
while (error > 1e-5 && itr < 10000)
    itr=itr+1;
    Aold=A;
    for i=2:(length(y)-1)
        for j=1:length(x)
            if j==1
                A(i,j) = 1/4 * (A(i-1, j) + Aold(i+1, j) + 2*Aold(i, j+1));
            elseif j==length(x)
                A(i,j) = 1/(4+2*h) * ( A(i-1, j) + Aold(i+1, j) + 2*A(i, j-
1) - 2*h);
            elseif 6<=i && i<=14 && 16<=j && j<=24
                A(i,j) = 0;
            elseif j~=length(x) && j~=1
                A(i,j)=1/4*(A(i-1,j)+Aold(i+1,j)+A(i,j-1)+Aold(i,j+1));
            end
        end
    end
    errorold=error;
    error=max(max(abs(A-Aold)));
    errornew=error;
    spectral(itr)=errornew/errorold;
    pitr=pitr+1;
    if pitr==5
       figure(1);subplot(1,2,1);
       contour(A);
       pitr=0;
    end
end

fprintf('Gauss Seidel Iterations %d\n',itr);
```

```
figure(1); subplot(1,2,1);
contour(A);
xlabel('x')
ylabel('y')
title(['GaussSeidel: # of Iterations ' num2str(itr)]);
figure(1); subplot(1,2,2);
[X,Y]=meshgrid(x, y);
surf(X,Y,A)
title(['GaussSeidel: # of Iterations ' num2str(itr)]);

figure(2), clf
plot(spectral);
gauss_spectral = mean(spectral(itr-5:itr));
fprintf('Spectral Radius %f\n',gauss_spectral);

fprintf('Value at point  x=0.5, y=0.5 %.12f \n', A(10,10))
```

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

$$V_x = \frac{\partial \psi}{\partial y}, V_y = -\frac{\partial \psi}{\partial x} \qquad (6)$$

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 (5), and (6) as compared to that of equations (2) and (3) and the respective solutions each produced, each describes 'flow', what observations can you make regarding how they differ with respect to the representation of flow?

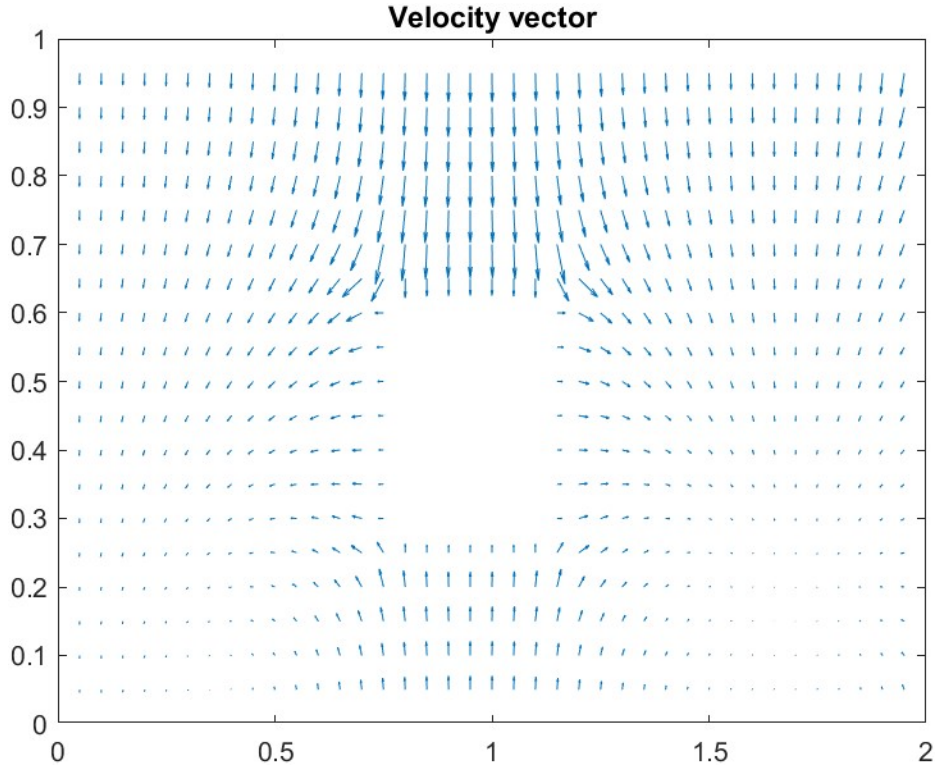


Figure 5. Velocity vector of field.

Answer:

- Explain the results you see with respect to plot in part **(a)**

  Velocity goes perpendicular to domain contour, and the velocity plot represents changes in space at each position.

  If we compare the domain contour plot and 'velocity plot', upper part of contour plot presents more rapid change (closer contours with more changes in color), thus bigger vector lengths in velocity plot (faster flow).

- In problem 1, we have 3 Type 1 boundaries and an incoming flow from the left that is perpendicular to the contour, thus create the contour plot as seen.

- In problem 2, we have more flow coming from the right, a gradient/difference of potential from top to bottom of domain, and an obstruction in the middle, thus flow directions like in the velocity plot is seen. The surface plot might illustrate better.

```
%%
Vx=zeros(length(y), length(x));
Vy=zeros(length(y), length(x));

Vx(2:end-1, 2:end-1) = ( A(3:end, 2:end-1) - A(1:end-2, 2:end-1) ) ./ (2*h);
Vy(:, 2:end-1) = - ( A(:,3:end) - A(:,1:end-2) ) ./ (2*h);
figure(3), clf
quiver(x, y, Vx, Vy)
title('Velocity vector')
```