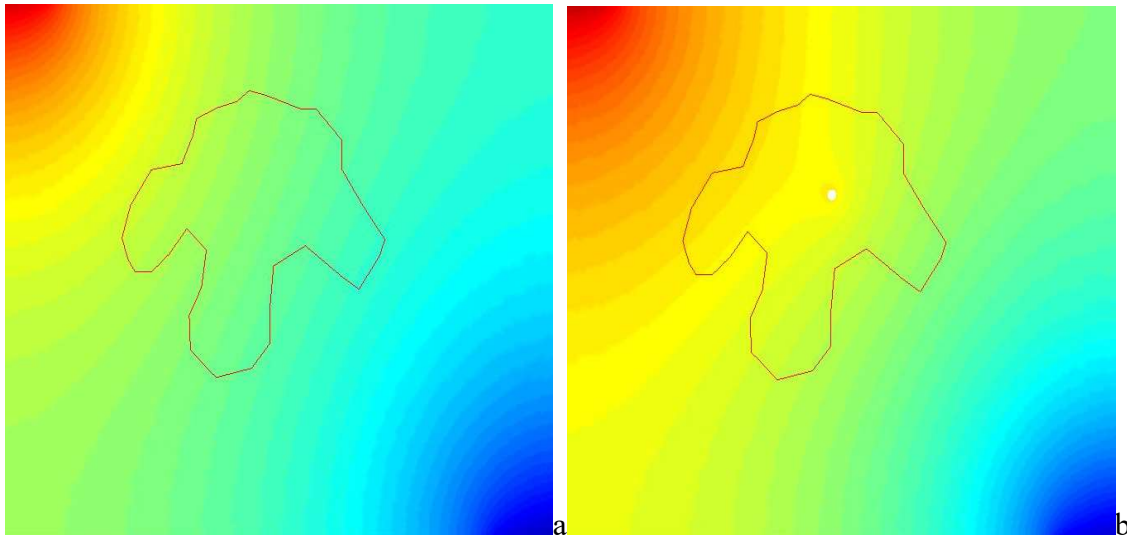


BME 7310 Computational Laboratory #2

Due: 9/15/2023

Problem 1: Concepts in Divergence.



Electro-potential distributions (a) Scenario #1, (b) Scenario #2.

Often electrostatic potential theory is used to represent the distribution **of current within the brain**. In the two scenarios above, a very basic distribution of potential (i.e. voltage) is illustrated. Laplace's equation was used for each problem. It highlights the flow of current from the top left to the bottom right of the images. Within the domain, an arbitrary contour is shown. The image to the right shows **an additional structure which influences the domain**.

You have been provided 2 files for each scenario above. The two files for each scenario are: POINT_LIST_#.DAT, and LINE_LIST_#.DAT where the # is associated with each scenario.

POINT_LIST_#.DAT has the format: [point index] [x position] [y position] [**Jx**] [**Jy**]

LINE_LIST_#.DAT has the format: [**contr. line index**] [**contr. node 1**] [**contr. node 2**]

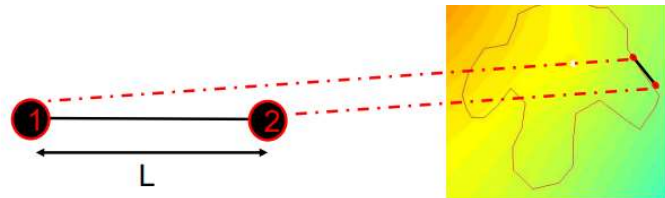
For reference, the contour nodes are arranged **such that for each contour line segment, it is arranged as traversing the boundary in a counter-clockwise manner**.

a. Your task is to **calculate the following line integral**:

$$\int_{\ell} (\vec{J} \cdot \hat{n}) d\ell = ???$$

and **report the values for each of the scenarios above**. Be sure to submit any MATLAB code used to achieve your results.

Solution:


$$\iint_S \mathbf{J} \cdot \hat{\mathbf{n}} dS \xrightarrow{\text{3D-2D rep}} \int_{\ell} J_n d\ell = J_{n1} \int_1^2 \varphi_1(\ell) d\ell + J_{n2} \int_1^2 \varphi_2(\ell) d\ell$$
$$\iint_S \mathbf{J} \cdot \hat{\mathbf{n}} dS \xrightarrow{\text{3D-2D rep}} \int_{\ell} J_n d\ell = J_{n1} \frac{L}{2} + J_{n2} \frac{L}{2}$$

This represents the integration of \mathbf{J}_n over this length.
Just need to **go length by length** and **sum up!**

Figure 1. Summary of the approach for the integration

Line integration for **pointlist1**: -0.040104
Line integration for **pointlist2**: -10.327510

The same code can be used, just assign line_list and points_list to the respective line and point list that match the scenario:

```
line_list_1 = importdata('LINE_LIST_1.DAT');  
line_list_2 = importdata('LINE_LIST_2.DAT');  
point_list_1 = importdata('POINT_LIST_1.DAT');  
point_list_2 = importdata('POINT_LIST_2.DAT');
```

```
% create edges
```

```
line_list = line_list_2;  
point_list = point_list_2;  
% -----  
line = [line_list(1,2)];  
for n = 2: length(line_list(:,1))  
    index = find(line_list(:,2) == line(end));  
    line(n) = line_list(index,3);  
end
```

```
line = line';
```

```
wrapped_line = [line(end);line;line(1)];  
XY = point_list(line, 2:3);  
wrappedXY = point_list(wrapped_line, 2:3);  
figure(1), clf  
plot(wrappedXY(:,1), wrappedXY(:,2))
```

```
deltaXY = wrappedXY(3:end,:) - wrappedXY(1:end-2,:);  
n_v = deltaXY ./ sqrt(deltaXY(:,1).^2 + deltaXY(:,2).^2) ;
```

```

n_v = [n_v(:,2), -n_v(:,1)];
quiver(XY(:,1), XY(:,2), n_v(:,1), n_v(:,2))
title('normal vector')
% hold on
for n = 1:length(XY(:,1))
    if mod(n, 10)==0
        text(XY(n,1),XY(n,2), string(n))
    end
end

quiver(XY(:,1), XY(:,2), J_field(:,1), J_field(:,2))
title('J vector field')
% The direction of the vector field were plotted at each points and it
% points from the bottom right to the top left
J_field = point_list(line, 4:5);
J_field_projection_for_each_point = dot(J_field, n_v, 2);
J_field_projection = [J_field_projection_for_each_point;
    J_field_projection_for_each_point(1)];
% calculate length between each 2 points / 2
new_wrapped_line = wrapped_line(2:end);
new_wrappedXY = point_list(new_wrapped_line, 2:3);
differenceXY = new_wrappedXY(2:end,:) - new_wrappedXY(1:end-1,:);
length_segments = sqrt(differenceXY(:,1).^2 + differenceXY(:,2).^2);

line_integration = sum(J_field_projection(1:end-1).*(length_segments/2) ...
    + J_field_projection(2:end).*(length_segments/2));

fprintf('Line integration for pointlist: %f', line_integration)

```

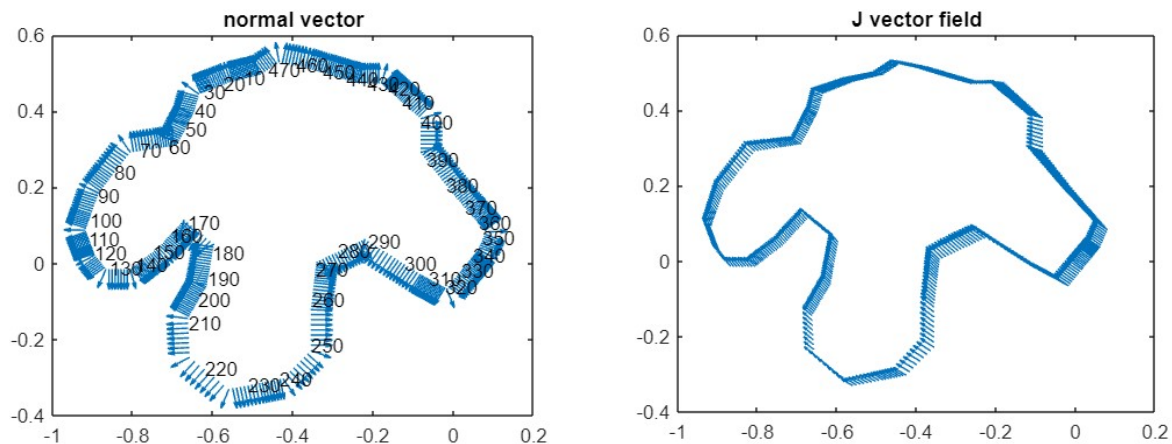
b. After getting your results, what can you say about your solution with respect to your understanding of the **divergence operator**? Also, if the results were not exactly what you expected, can you suggest a reason why? What is the influence of the **additional structure in the additional domain**?

What can you say about your solution with respect to your understanding of the divergence operator

- Here we have J as a vector field. Divergence, or the integration we calculated, represents the **flux of a vector field** across the curve.

Also, if the results were not exactly what you expected, can you suggest a reason why?

- The direction of the normal vector and J_field at each point were plotted. The J_field points from the bottom right to the top left, and a big part of the line has the normal vector and J opposite each other. Thus, we get more negative values from the dot products, and thus the negative value for integration.



What is the influence of the additional structure in the additional domain?

- The additional structure in the domain causes more rapid change from high to low potential through the curve. As expected, magnitude-wise, b) has a higher magnitude compared to a).

c. In class, we performed an integration **over a single contour line** and produced the following:

$$\int_1^2 (\vec{J} \cdot \vec{n}) d\ell = J_{n1} \frac{L}{2} + J_{n2} \frac{L}{2}$$

If I were to change the integral to $\int_1^2 (\vec{J} \cdot \vec{n}) \phi_1 d\ell$, what would the result be **for all points**. If I

were to change the integral to $\int_1^2 (\vec{J} \cdot \vec{n}) \phi_2 d\ell$ **for all points**, what would the result be? For this problem assume the same 2-node domain that was in the lecture with the “rooftop” function.

$$\phi_1 = \frac{x_2 - x}{h}, \phi_2 = \frac{x - x_1}{h}$$

Solution: We can write out the integration formula by hand as below.

$$\begin{aligned}
\int_1^2 (\vec{T} \cdot \vec{n}) \phi_4 dl &= T_{n1} \int_1^2 \phi_1 \phi_1 dl + T_{n2} \int_1^2 \phi_1 \phi_2 dl \\
&= -T_{n1} \int_1^2 \frac{(x_2 - x)^2}{h^2} dx + T_{n2} \int_1^2 \frac{(x_2 - x)}{h} \cdot \frac{(x - x_1)}{h} dx \\
&= -\frac{T_{n1}}{h^2} \int_1^2 (x_2 - x)^2 dx + \frac{T_{n2}}{h^2} \int_1^2 (x_2 x - x_2 x_1 - x^2 + x x_1) dx \\
&= \underbrace{-\frac{T_{n1}}{h^2} \frac{(x_2 - x)^3}{3} \Big|_{x_1}^{x_2}}_{(1)} + \underbrace{\frac{T_{n2}}{h^2} \left(-x_1 x_2 x + (x_1 + x_2) \frac{x^2}{2} - \frac{x^3}{3} \right) \Big|_{x_1}^{x_2}}_{(2)} \\
&= (1) + (2)
\end{aligned}$$

$$(1) = -\frac{T_{n1}}{h^2} \left(0 - \left(\frac{x_2 - x_1}{3} \right)^3 \right)$$

$$= T_{n1} \cdot \frac{h}{3}$$

$$(2) = \frac{T_{n2}}{h^2} \left(-x_1 x_2 (x_2 - x_1) + \frac{(x_1 + x_2)}{2} (x_2^2 - x_1^2) - \frac{1}{3} (x_2^3 - x_1^3) \right)$$

$$= \frac{T_{n2}}{6h^2} \left(-6x_1 x_2 h + 3(x_1 + x_2)^2 h - 2h(x_2^2 + x_1 x_2 + x_1^2) \right)$$

$$= \frac{T_{n2}}{6h^2} \left(-\cancel{6x_1 x_2 h} + 3(\cancel{x_1^2} - \cancel{2x_1 x_2} + \cancel{x_2^2}) - 2h(\cancel{x_2^2} + \cancel{x_1 x_2} + \cancel{x_1^2}) \right)$$

$$= \frac{T_{n2}}{6h^2} h \left(x_1^2 - 2x_1 x_2 + x_2^2 \right)$$

$$= \frac{T_{n2} \cdot h \cdot (x_1 - x_2)^2}{6h^2}$$

$$= T_{n2} \cdot \frac{h}{6}$$

$$\text{Thus } \int_1^2 (\vec{T} \cdot \vec{n}) \phi_1 dl = T_{n1} \frac{h}{3} + T_{n2} \cdot \frac{h}{6}$$

$$\int_1^2 (\vec{T} \cdot \vec{n}) \phi_2 dl = \underbrace{T_{n1} \int_1^2 \phi_1 \phi_2 dl}_{(3)} + \underbrace{T_{n2} \int_1^2 \phi_2 \phi_2 dl}_{(4)}$$

The integration in (3) is same as (2), so we only need to find (4)

$$\begin{aligned} (4) &= T_{n2} \int_1^2 \phi_2 \phi_2 dl \\ &= T_{n2} \int_{x_1}^{x_2} \frac{(x - x_1)^2}{h^2} dx \end{aligned}$$

$$= \frac{T_{n2}}{h^2} \left(\frac{(x - x_1)^3}{3} \right) \bigg|_{x_1}^{x_2}$$

$$= \frac{T_{n2}}{h^2} \frac{(x_2 - x_1)^3}{3}$$

$$= T_{n2} \cdot \frac{h}{3} \quad (\text{reasonable, symmetric with } T_{n1} \int_1^2 \phi_1 \phi_1 dl)$$

$$\begin{aligned}\text{Thus } \int_1^2 (\vec{J} \cdot \vec{n}) \phi_2 dl &= J_{n1} \int_1^2 \phi_1 \phi_2 dl + J_{n2} \int_1^2 \phi_2 \phi_2 dl \\ &= J_{n1} \cdot \frac{h}{6} + J_{n2} \frac{h}{3}\end{aligned}$$

For point list1:

Line integration with **phi1** for pointlist: -0.016334
Line integration with **phi2** for pointlist: -0.038600

For point list2:

Line integration with **phi1** for pointlist: -5.164692
Line integration with **phi2** for pointlist: -13.773760

All code attached below:

IF we have **phi1** in the integration

```
line_list_1 = importdata('LINE_LIST_1.DAT');
line_list_2 = importdata('LINE_LIST_2.DAT');
point_list_1 = importdata('POINT_LIST_1.DAT');
point_list_2 = importdata('POINT_LIST_2.DAT');

% create edges
line_list = line_list_1;
point_list = point_list_1;
% -----
line = [line_list(1,2)];
for n = 2: length(line_list(:,1))
    index = find(line_list(:,2) == line(end));
    line(n) = line_list(index,3);
end
line = line';
wrapped_line = [line(end);line;line(1)];
XY = point_list(line, 2:3);
wrappedXY = point_list(wrapped_line, 2:3);
deltaXY = wrappedXY(3:end,:) - wrappedXY(1:end-2,:);
n_v = deltaXY ./ sqrt(deltaXY(:,1).^2 + deltaXY(:,2).^2) ;
n_v = [n_v(:,2), -n_v(:,1)];

J_field = point_list(line, 4:5);
```



```

J_field_projection_for_each_point = dot(J_field, n_v, 2);
J_field_projection = [J_field_projection_for_each_point;
    J_field_projection_for_each_point(1)];
% calculate length between each 2 points / 2
new_wrapped_line = wrapped_line(2:end);
new_wrappedXY = point_list(new_wrapped_line, 2:3);
differenceXY = new_wrappedXY(2:end,:) - new_wrappedXY(1:end-1,:);
length_segments = sqrt(differenceXY(:,1).^2 + differenceXY(:,2).^2);

line_integration = sum(J_field_projection(1:end-1).*(length_segments/3) ...
    + J_field_projection(2:end).*(length_segments/6));

fprintf('Line integration with phi1 for pointlist: %f', line_integration)

```

For line integration with **phi2**

```

line_list_1 = importdata('LINE_LIST_1.DAT');
line_list_2 = importdata('LINE_LIST_2.DAT');
point_list_1 = importdata('POINT_LIST_1.DAT');
point_list_2 = importdata('POINT_LIST_2.DAT');

% create edges
line_list = line_list_2;
point_list = point_list_2;
% -----
line = [line_list(1,2)];
for n = 2: length(line_list(:,1))
    index = find(line_list(:,2) == line(end));
    line(n) = line_list(index,3);
end
line = line';
wrapped_line = [line(end);line;line(1)];
XY = point_list(line, 2:3);
wrappedXY = point_list(wrapped_line, 2:3);
deltaXY = wrappedXY(3:end,:) - wrappedXY(1:end-2,:);
n_v = deltaXY ./ sqrt(deltaXY(:,1).^2 + deltaXY(:,2).^2) ;
n_v = [n_v(:,2), -n_v(:,1)];

J_field = point_list(line, 4:5);
J_field_projection_for_each_point = dot(J_field, n_v, 2);
J_field_projection = [J_field_projection_for_each_point;
    J_field_projection_for_each_point(1)];
% calculate length between each 2 points / 2
new_wrapped_line = wrapped_line(2:end);

```



```
new_wrappedXY = point_list(new_wrapped_line, 2:3);
differenceXY = new_wrappedXY(2:end,:) - new_wrappedXY(1:end-1,:);
length_segments = sqrt(differenceXY(:,1).^2 + differenceXY(:,2).^2);

line_integration = sum(J_field_projection(1:end-1).*(length_segments/1) ...
    + J_field_projection(2:end).*(length_segments/3));

fprintf('Line integration with phi2 for pointlist: %f', line_integration)
```

Problem #2. Understanding Current: Conservation of cortical current is governed by the PDE

$$\nabla \cdot \vec{J} = 0 \quad (1)$$

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 .

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

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

$$\nabla \cdot (-\sigma \nabla \Phi) = 0 \quad (3)$$

under **homogeneous electrical conductivity**, it can be written as (Laplace equation):

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

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

(a) Even though you will be solving equation (4) below with respect to the computational model, in this part write out the **full FD description approach for equation (3)**. Be sure to address the material property.

$$\nabla \cdot (-\sigma \nabla \Phi) = 0$$

$$\frac{\partial(-\sigma_x \nabla \Phi_x)}{\partial x} + \frac{\partial(-\sigma_y \nabla \Phi_y)}{\partial y} = 0$$

$$\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0,$$

$$\text{where } F = (-\sigma_x \nabla \Phi_x), G = (-\sigma_y \nabla \Phi_y)$$

$$\frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{2(\frac{h}{2})} + \frac{G_{i+\frac{1}{2}} - G_{i-\frac{1}{2}}}{2(\frac{k}{2})} = 0$$

$$\frac{-\sigma_{i+\frac{1}{2}} \frac{\partial \Phi}{\partial x}_{i+\frac{1}{2}} - \left(-\sigma_{i-\frac{1}{2}} \frac{\partial \Phi}{\partial x}_{i-\frac{1}{2}} \right)}{h} + \frac{-\sigma_{i+\frac{1}{2}} \frac{\partial \Phi}{\partial y}_{i+\frac{1}{2}} - \left(-\sigma_{i-\frac{1}{2}} \frac{\partial \Phi}{\partial y}_{i-\frac{1}{2}} \right)}{k} = 0$$

$$\frac{-\sigma_{i+\frac{1}{2},j} \frac{\Phi_{i+1,j} - \Phi_{i,j}}{2(\frac{h}{2})} - \left(-\sigma_{i-\frac{1}{2},j} \frac{\Phi_{i,j} - \Phi_{i-1,j}}{2(\frac{h}{2})} \right)}{h}$$

$$+ \frac{-\sigma_{i,j+\frac{1}{2}} \frac{\Phi_{i,j+1} - \Phi_{i,j}}{2(\frac{k}{2})} - \left(-\sigma_{i,j-\frac{1}{2}} \frac{\Phi_{i,j} - \Phi_{i,j-1}}{2(\frac{k}{2})} \right)}{k} = 0$$

$$\frac{-\sigma_{i+\frac{1}{2},j}(\Phi_{i+1,j}-\Phi_{i,j})+\left(\sigma_{i-\frac{1}{2},j}(\Phi_{i,j}-\Phi_{i-1,j})\right)}{h^2}$$

$$+\frac{-\sigma_{i,j+\frac{1}{2}}(\Phi_{i,j+1}-\Phi_{i,j})+\left(\sigma_{i,j-\frac{1}{2}}(\Phi_{i,j}-\Phi_{i,j-1})\right)}{k^2}=0$$

$$-\sigma_{i+\frac{1}{2},j}(\Phi_{i+1,j}-\Phi_{i,j})+\left(\sigma_{i-\frac{1}{2},j}(\Phi_{i,j}-\Phi_{i-1,j})\right)$$

$$+\frac{h^2}{k^2}\left(-\sigma_{i,j+\frac{1}{2}}(\Phi_{i,j+1}-\Phi_{i,j})+\left(\sigma_{i,j-\frac{1}{2}}(\Phi_{i,j}-\Phi_{i,j-1})\right)\right)=0$$

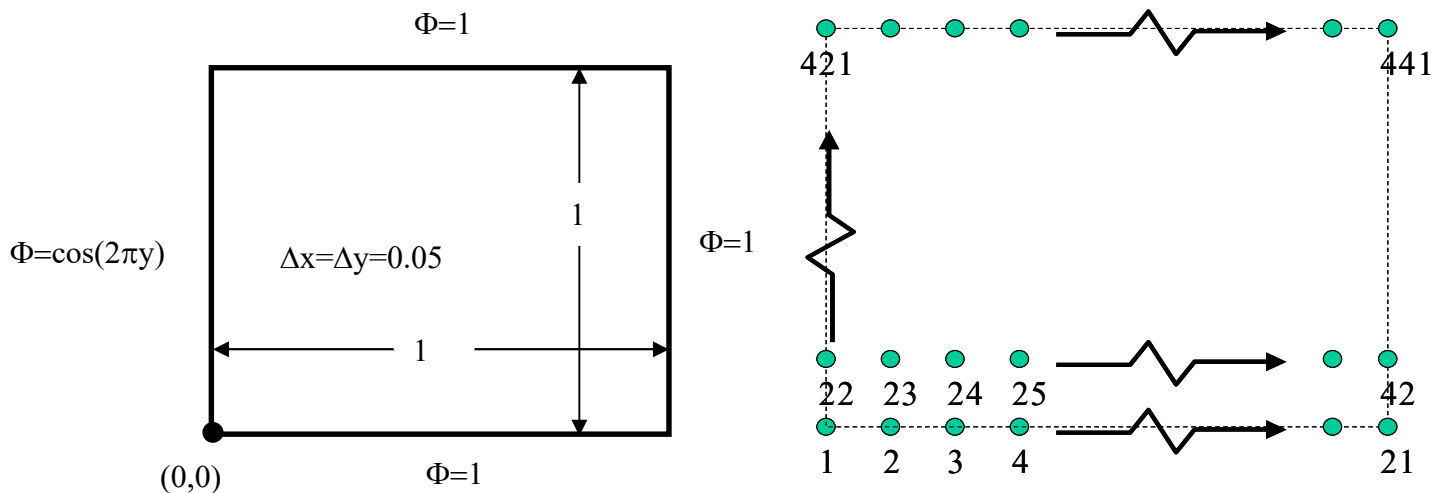
(b) Given the problem **domain** and **boundary conditions** below, **compute the solution of Φ** using **Jacobi iteration** with an initial solution vector of $\Phi=0$ everywhere.

- Iterate until reaching an absolute L_∞ norm-based error of successive iterates of less than 1×10^{-5} and report the number of iterations needed to reach this convergence criterion.

(Answer below)

- 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 at $y=0.7$ Estimate the spectral radius of the Jacobi iteration matrix during the course of the iterations and compare with the theoretically expected value.

(Answer below)



(c) Repeat part (a) using **Gauss-Seidel**. Be sure to report the solution value requested in part (a) in order to verify that your solution is essentially unchanged. Is the speed-up in terms of convergence rate relative to Jacobi in agreement with theory? If the convergence criterion were extended to 1×10^{-6} how many more iterations would you **predict** would be needed with Gauss-Seidel? Is your prediction in reasonable agreement with practice?

Answer: for both b) and c):

>> For Tol=1e-5 we need:

787 iterations for Jacobi,

Jacobi $V(x=0.7, y=0.7)$: 0.870189353342

theoretical_spectral_r_Jacobi 0.987662994499

numerical_spectral_Jacobi 0.993177463840

425 iterations for GaussSeidel,

Gauss Seidel $V(x=0.7, y=0.7)$: 0.870212161185

theoretical_spectral_r_Gauss 0.975325988997

numerical_spectral_r_Gauss 0.979454652024

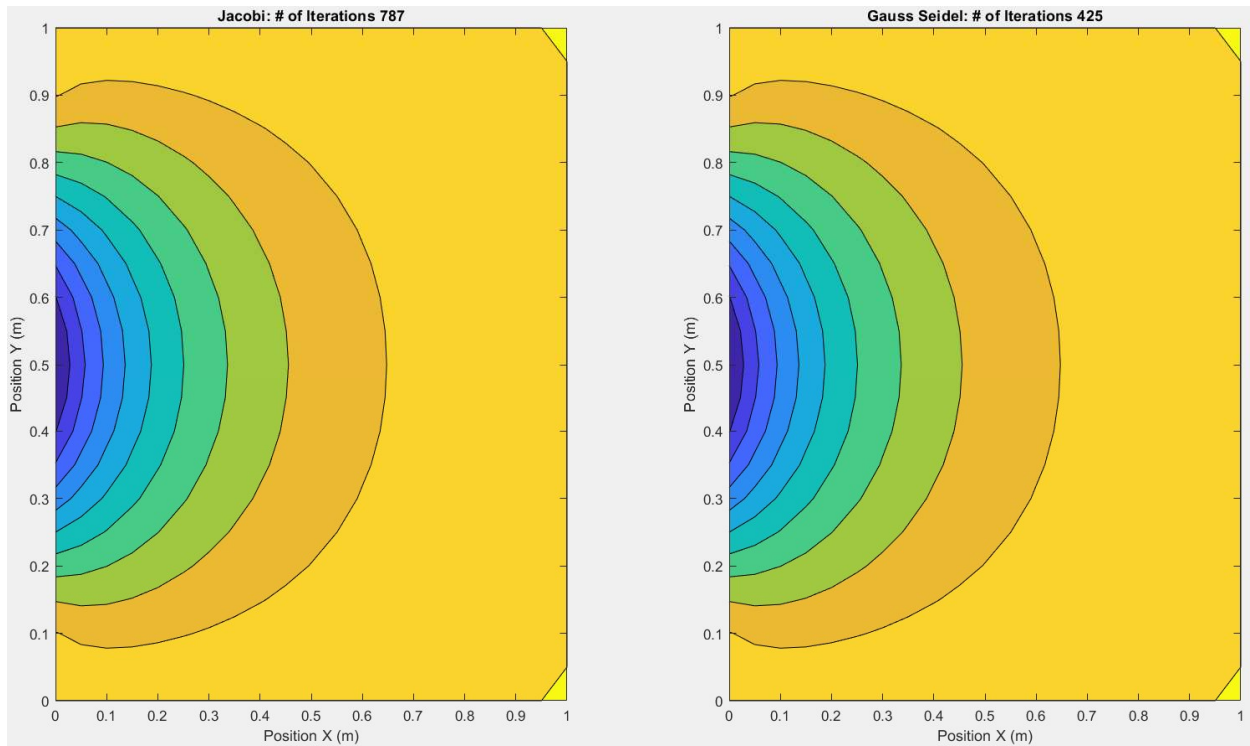


Figure 2. Contour plots with number of iterations for Jacobi (left) and Gauss Seidel (right) for Tol=1e-5

For Tol=1e-6 we need

973 iterations for Jacobi

Jacobi $V(x=0.7, x=0.7)$: 0.870226300688

theroretical_spectral_r_Jacobi 0.987662994499

numerical_spectral_Jacobi 0.993177463840

518 iterations for GaussSeidel

Gauss Seidel $V(x=0.7, x=0.7)$: 0.870228577832

theroretical_spectral_r_Gauss 0.975325988997

numerical_spectral_r_Gauss 0.979454652024

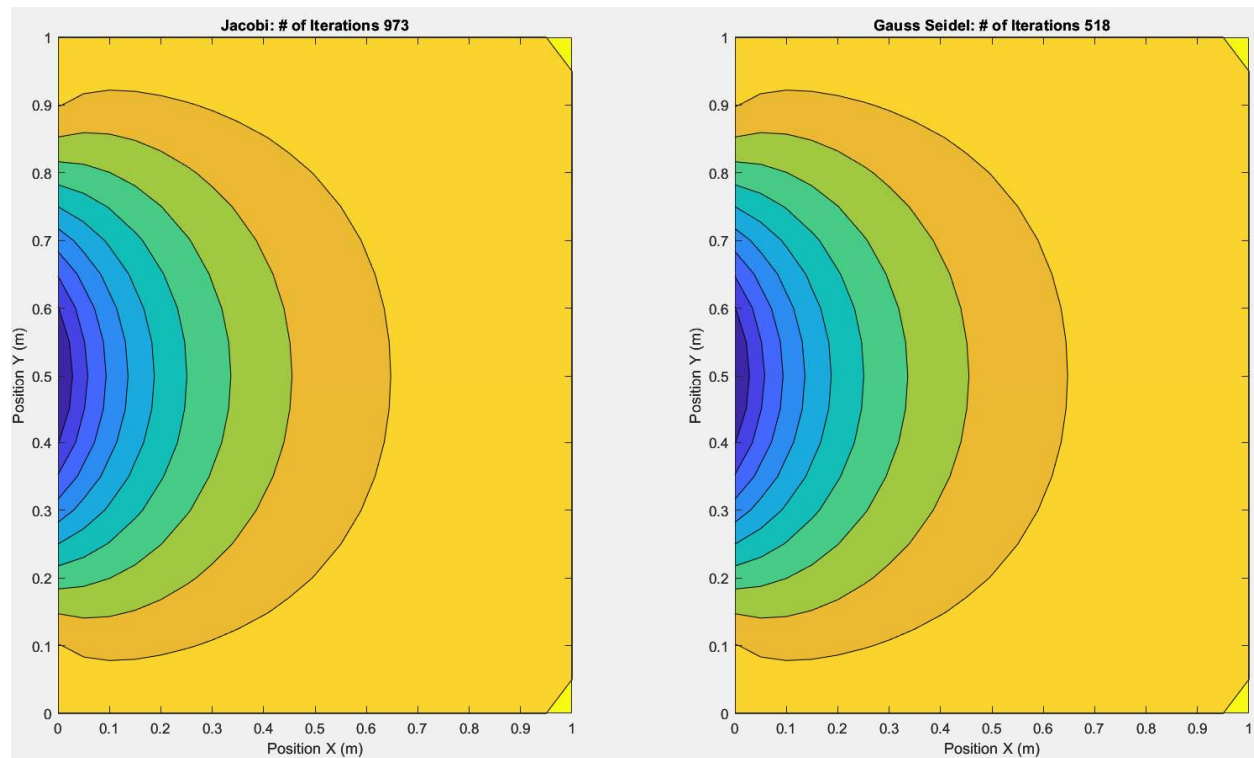


Figure 3. Contour plots with number of iterations for Jacobi (left) and Gauss Seidel (right) for Tol=1e-6

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

- In both cases of Tol=1e-5 and Tol=1e-6 we saw a close to half reduction in the number of iterations for GaussSeidel vs. Jacobi.

From theory, Jacobi has a convergence rate of $O(h^4)$ whereas GaussSeidel has $O(h^2)$, thus the fact GaussSeidel saw a close to x2 times speed-up for both cases agrees with theory.

If the convergence criterion were extended to 1×10^{-6} how many more iterations would you **predict** would be needed with Gauss-Seidel? Is your prediction in reasonable agreement with practice?

We have the theoretical spectral radius as:

$$\rho \approx 1 - \frac{\pi^2 h^2}{2a^2}$$

where h is our step size and a is the size of 1 edge of the domain. Thus, from the formula Tolerance does not necessarily influence the spectral radius. In other words, we cannot really estimate a good number of iterations from the theoretical formula above.

But once we numerically calculated spectral radius based on the largest error ratio through all iterations, we can estimate the number of iterations by using

$$num_iter = 1/abs(log_{10}(\rho))$$

, which differs quite a bit from the actual number of iterations used for Jacobi and Gauss in both Tol limits.

For Tol=1e-5

- Theoretical number of iterations for Jacobi at Tol=0.000010: **185.486843133081**
- Estimated number of iterations for Jacobi at Tol=0.000010: **336.344336190154**

- Theoretical number of iterations for GaussSeidel at Tol=0.000010: **92.164172437790**
- Estimated number of iterations for GaussSeidel at Tol=0.000010: **110.918031224449**

For Tol=1e-6

- Theoretical number of iterations for Jacobi at Tol=0.000001: **185.486843133081**
- Estimated number of iterations for Jacobi at Tol=0.000001: **336.344336190154**

- Theoretical number of iterations for GaussSeidel at Tol=0.000001: **92.164172437790**
- Estimated number of iterations for GaussSeidel at Tol=0.000001: **110.918031224449**

We can see the theoretical and estimated number of iterations do not change with respect to Tol value. We can see the estimation differs from the actual number of iterations reported earlier.