BME 7310 Computational Laboratory #4

Due: September 29, 2023

Problem #1. Piece of cake!

Conservation of mass is governed by the PDE

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

where $\vec{v}$ is the velocity of interstitial fluid. Often $\vec{v}$ is expressed with respect to tissue pressure changes, this can be expressed as the gradient of the **pressure** $\boldsymbol{p}$,

$$\vec{v} = -k\nabla p \qquad (2)$$

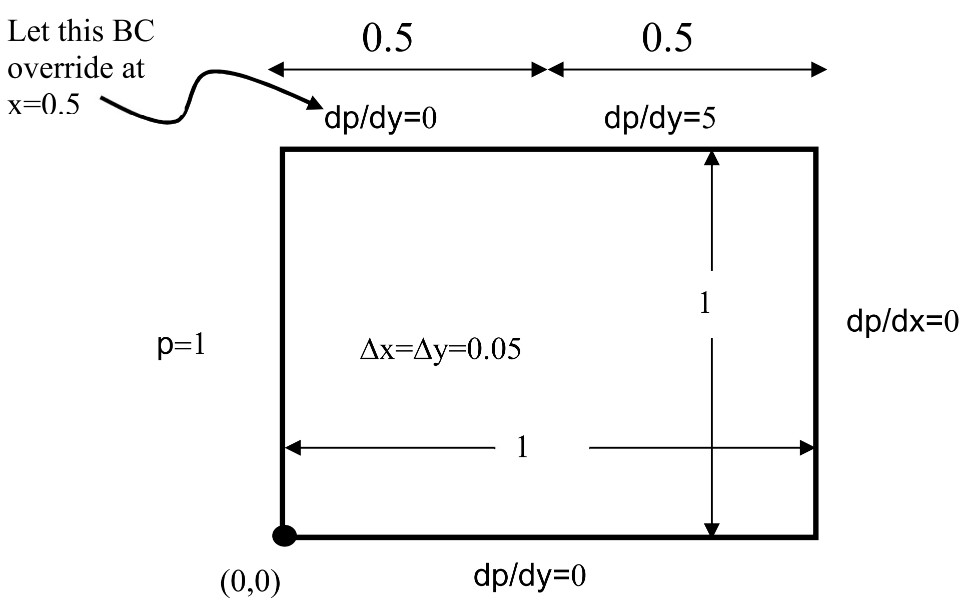
Hence equation (1) can be recast in terms of p as,

$$\nabla \bullet (-k\nabla p) = 0 \qquad (3)$$

under constant hydraulic conductivity, can be written as:

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

(a) Let's look at a more interesting problem. Using **the point iterative method of your choice**. Calculate the solution to the following geometry and: plot using the contour command, report the number of iterations, and report the value for x=0.7, y=0.7. Use a ***relative*** tolerance level of 1e-5 between successive solutions as a stopping criterion.



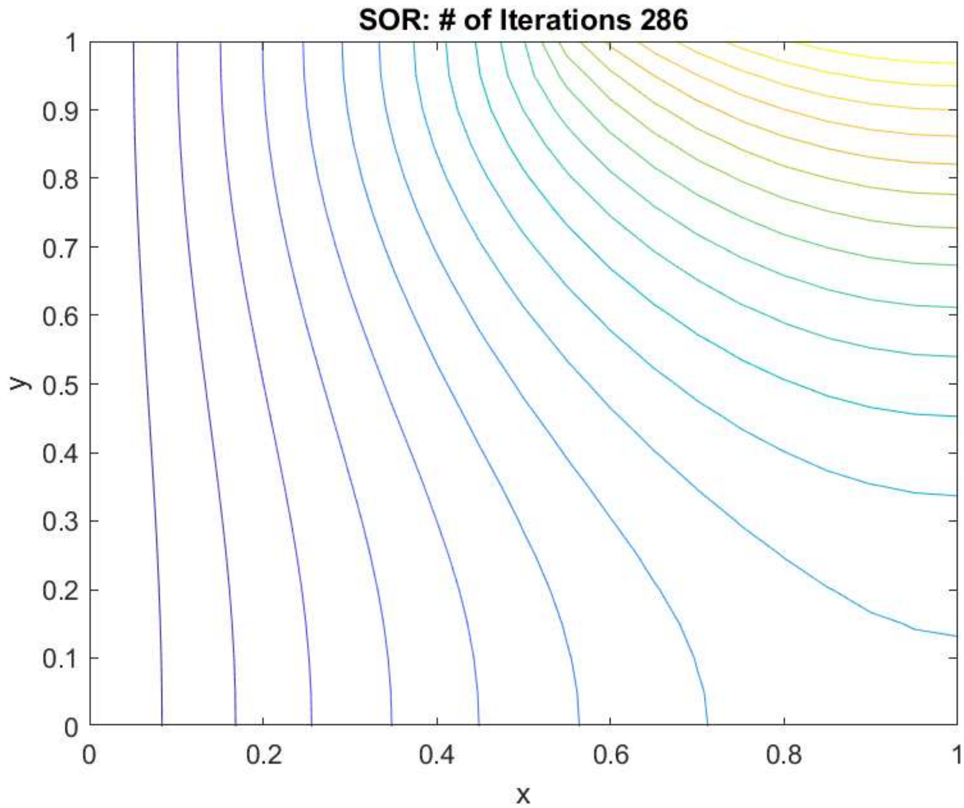


Figure 1. Contour of pressure field

(b) Now calculate the ***fluid velocity vector*** components based on the description in equation (2) and plot using the 'quiver' command (assume k=1 throughout whole domain)? You should produce a vector for every node in the domain. Explain the results you observe and the relationship to the solution obtained in part **(a)**.

Also discuss the relationship between the solution and the boundary conditions, i.e. based on your understanding of equations (1-4) and boundary conditions, describe how these specific boundary conditions are consistent with respect your quiver plot. In addition, look at the contour trend in $\boldsymbol{p}$ and discuss how the contours make sense with respect to the boundary conditions specified.

<u>Answer:</u> The quiver fluid vector plot makes sense since we have a field pumping into the domain at boundary dp/dx = 5 and $v = -k\, dp/dx$ thus the inward direction of the fluid.

Higher velocity (larger vectors) was at regions where more rapid changes in contour of pressure field was shown (top right of quiver plot).

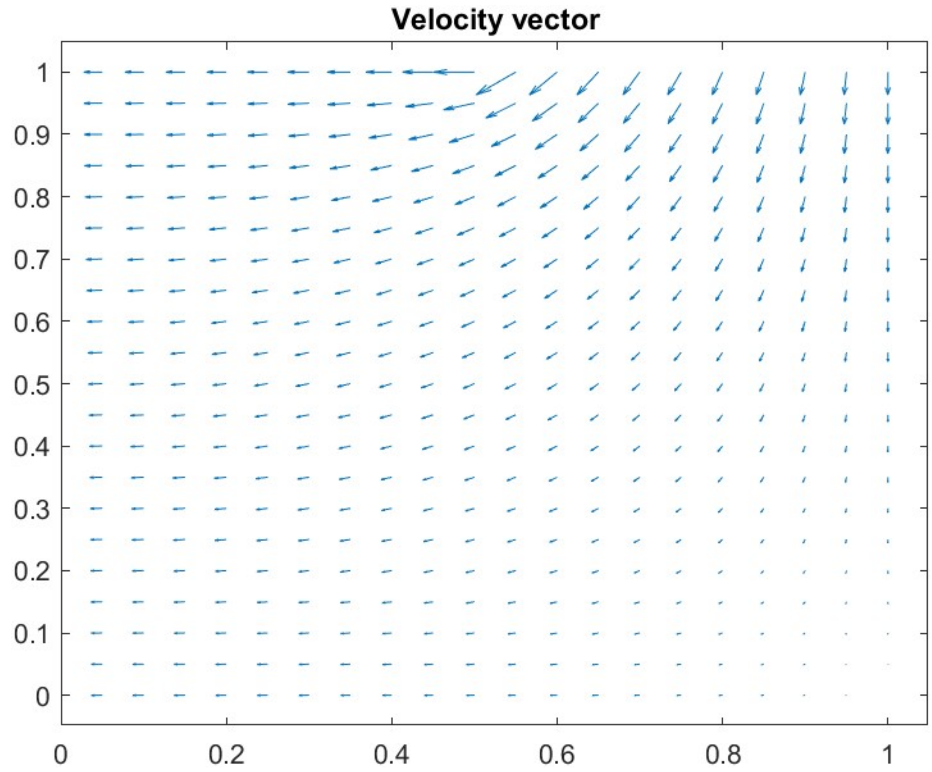


Figure 2. Velocity vector plot. Higher velocity at more rapid changes in contour of pressure field (top right).

```
clear all

figure(1);

% SOR
clear;
h=0.05;
a = 50;
w=1.785;

x=[0:h:1];
y=[0:h:1];
A=zeros(length(y), length(x));

% ---- BC -------
A(:,1) = 1;
error=1;
itr=0;
pitr=0;
while (error > 1e-5 & itr < 10000)
     itr=itr+1;
     Aold=A;
     for i=1:length(y)
        for j=2:length(x)
            if i==1 % bottom condition
                if j<length(x) % bottom condition
```

```
                    A(i,j)=w/4*( 2*Aold(i+1,j) + Aold(i,j+1) +A(i,j-1) )+(1-
w)*Aold(i,j);
                else % bottom right corner
                    A(i,j)=w/4*( 2*Aold(i+1,j) + 2*A(i,j-1) )+(1-
w)*Aold(i,j);
                end

            elseif 1<i & i<length(y) % middle
                if j < length(x) % in middle
                    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);
                else % right condition
                    A(i,j)=w/4*(Aold(i+1,j)+A(i-1,j)+2*A(i,j-1))+(1-
w)*Aold(i,j);
                end

            elseif i==length(y) % top condion segment by j
                if j<=11
                    A(i,j)=w/4*( 2*A(i-1,j) + Aold(i,j+1) + A(i,j-1) )+(1-
w)*Aold(i,j);
                elseif 11<j & j < 21
                    A(i,j)=w/4*( 2*A(i-1,j) + Aold(i,j+1) + A(i,j-1) + 2*a*h
)+(1-w)*Aold(i,j);
                else % top right corner
                    A(i,j)=w/4*( 2*A(i-1,j) + 2*A(i,j-1) + 2*a*h)+(1-
w)*Aold(i,j);
                end
            end
        end
      end
     error=max(max(abs(A-Aold)))/max(max(abs(A)));
    pitr=pitr+1;
    if pitr==5
       figure(1)
       contour(y,x,A,20); % y row, x col
       pitr=0;
    end
end

fprintf('SOR Iterations %d\n',itr);
figure(1), clf
contour(y,x,A,20);
xlabel('x')
ylabel('y')
title(['SOR: # of Iterations ' num2str(itr)]);

fprintf('Value at point  x=0.7, y=0.7 %.12f \n', A(14,14))
%
Vx=zeros(length(y), length(x));
Vy=zeros(length(y), length(x));

% left
Vx(:,1) = 0;
Vy(:,1) = 0;
```

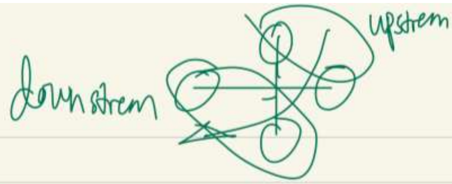
```
%right
Vx(:,end) = 0;
Vy(:,end) = 0; % calc below

% bottom
Vx(1,:) = 0;  % calc below
Vy(1,:) = 0;

% top
Vx(end,:) = 0; % calc below
Vy(end,1:11) = 0;
Vy(end,12:21) = -a;

% mid
Vx(:, 2:end-1)= - ( A(:, 3:end) - A(:, 1:end-2) )/ (2*h);
Vy(2:end-1, 2:end) = - ( A(3:end, 2:end) - A( 1:end-2, 2:end) )/ (2*h);

figure(2);
quiver(x, y, Vx, Vy)
xlim([0,1.05])
ylim([-0.05,1.05])
title('Velocity vector')
```

Problem #2: Test your Mettle! Now let's study the advective diffusion equation of a chemical species released into the potential flow stream derived from part (b) of Problem 1 ($\vec{v} = -k\nabla p$). The advective-diffusive equation can be written as:

$$\nabla^2 C - \vec{v} \bullet \nabla C = 0 \qquad (5)$$

(a) Write out the finite difference molecule using **three** methods of handling the advective term (center difference, upstream weighting, and downstream weighting). Note, you will need to perform **2-D weighting** so your result from Problem 1b will be important in this.

Solution: (next page)



$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial x}\hat{i} \\ \frac{\partial C}{\partial y}\hat{j} \end{bmatrix}$$

$$\nabla C^2 - \vec{v} \cdot \nabla C = 0$$

$$\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) - \begin{vmatrix} v_x \\ v_y \end{vmatrix} \cdot \left(\frac{\partial C}{\partial x}\hat{i} + \frac{\partial C}{\partial y}\hat{j}\right) = 0$$

$$= \begin{bmatrix} \frac{(C_{i+1,j} - C_{i-1,j})}{2h}\hat{i} \\ \frac{(C_{i,j+1} - C_{i,j-1})}{2h}\hat{j} \end{bmatrix}$$

$$\frac{\partial C^2}{\partial x^2} + \frac{\partial C^2}{\partial y^2} - \left(v_x \cdot \frac{\partial C}{\partial x} + v_y \frac{\partial C}{\partial y}\right) = 0$$

Center difference :

$$\frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{h^2} + \frac{C_{i,j+1} - 2C_{ij} + C_{i,j-1}}{h^2} - \left(v_{x_{ij}} \cdot \frac{C_{i+1,j} - C_{i-1,j}}{2h} + v_{y_{ij}} \cdot \frac{C_{i,j+1} - C_{i,j-1}}{2h}\right) = 0$$

$$2C_{i+1,j} - 4C_{ij} + 2C_{i-1,j} + 2C_{i,j+1} - 4C_{ij} + 2C_{i,j-1} - h \cdot v_{x_{ij}}(C_{i+1,j} - C_{i-1,j}) - h \cdot v_{y_{ij}}(C_{i,j+1} - C_{i,j-1})) = 0$$

$$C_{i+1,j}(2 - hv_{x_{ij}}) + C_{i-1,j}(2 + hv_{x_{ij}}) + C_{i,j+1}(2 - hv_{y_{ij}}) + C_{i,j-1}(2 + hv_{y_{ij}})$$

$$- 8C_{ij} = 0$$

Upstream (due to V_{ij} direction)

$$\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2} + \frac{C_{i,j+1} - 2C_{ij} + C_{i,j-1}}{h^2} - \left(v_{x_{ij}} \frac{C_{i+1,j} - C_{ij}}{h} + v_{y_{ij}} \frac{C_{i,j+1} - C_{ij}}{h}\right) = 0$$

$$C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} - 4C_{ij} - v_{x_{ij}}h \cdot C_{i+1,j} + v_{x_{ij}}h \cdot C_{ij} - v_{y_{ij}}h\, C_{i,j+1} + v_{y_{ij}}h\, C_{ij} = 0$$

$$C_{i+1,j}(1 - v_{x_{ij}}h) + C_{i-1,j} + C_{i,j+1}(1 - v_{y_{ij}}h) + C_{i,j-1}$$

$$+ C_{ij}(-4 + v_{x_{ij}}h + v_{y_{ij}}h) = 0$$

Downstream (due to V_{ij} direction)

$$\left(\frac{C_{i+1,j} - 2C_{ij} + C_{i-1,j}}{h^2}\right) + \left(\frac{C_{i,j+1} - 2C_{ij} + C_{i,j-1}}{h^2}\right) - \left(v_{x_{ij}}\frac{C_{ij} - C_{i-1,j}}{h} + v_{y_{ij}}\frac{C_{ij} - C_{i,j-1}}{h}\right) = 0$$

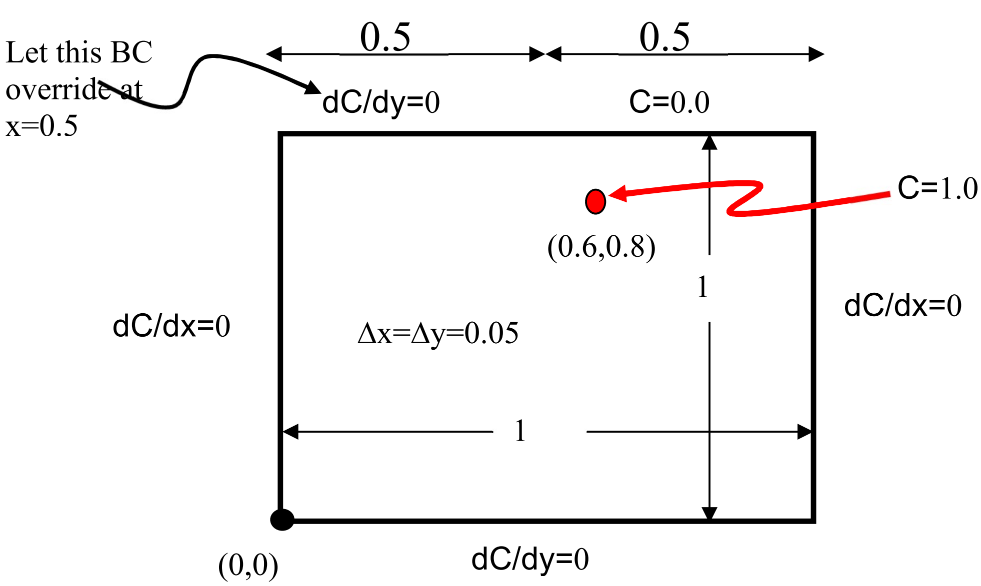
$$C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} - 4C_{ij}$$
$$- v_{x_{ij}} h C_{ij} + v_{x_{ij}} h . C_{i-1,j} - v_{y_{ij}} h C_{ij} + v_{y_{ij}} h C_{i,j-1} = 0$$

$$C_{i+1,j} + C_{i,j+1} + C_{i-1,j}(1 + v_{x_{ij}} h) + C_{i,j-1}(1 + v_{y_{ij}} h)$$
$$+ C_{ij}(-4 - v_{x_{ij}} h - v_{y_{ij}} h) = 0$$

More at bottom Appendix.

(b) Now for this problem, we are going to solve the problem below a total of **9 times**. For the first 3 solutions, use each of three FD expressions found in parte (a) and use the flow field found in Problem 1b above. For the second 3 solutions, resolve the problem from problem 1b but **increase the flux on the upper right boundary to dp/dy=**25, now use this resulting flow field to solve (5) with the below boundary conditions again. For the last 3 solutions, resolve problem 1b again but increase the flux **to dp/dy=**50 and solve (5) again. The boundary conditions for the solution of (5) are below. Use them for all 9 solutions.

Now looking at the results of all 9 graphs, discuss how this behavior is consistent with the discussion we had about the **advection-diffusive equation** in class. HINT: Once you have established your method to solve the below. One strategy to handle the source in the middle of the domain is to set the value to C=1, solve the whole domain as normal for 1 iteration, and then resetting that value to 1 before starting the next iteration.



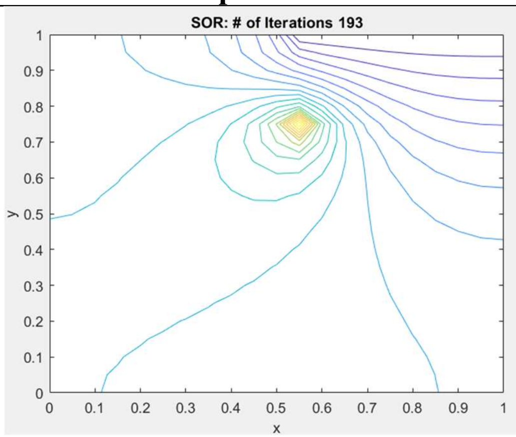
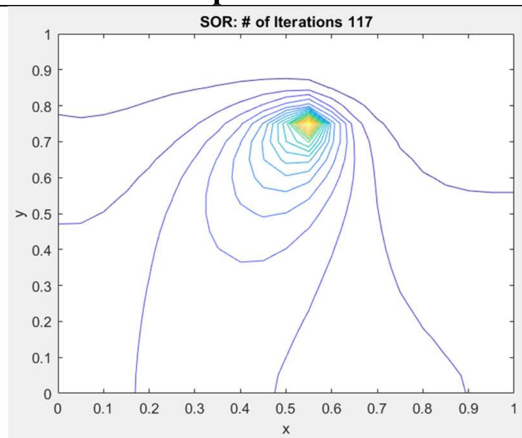
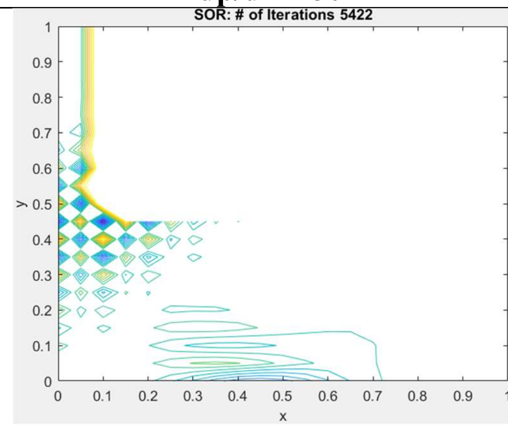
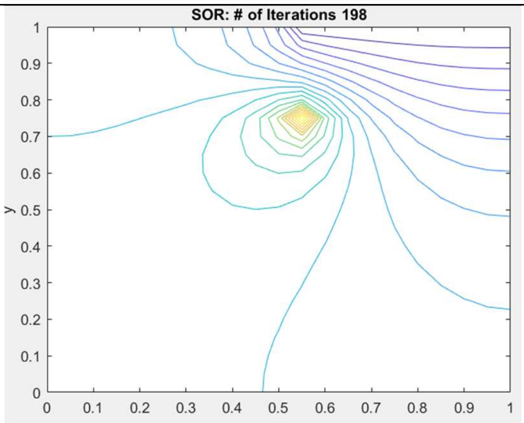
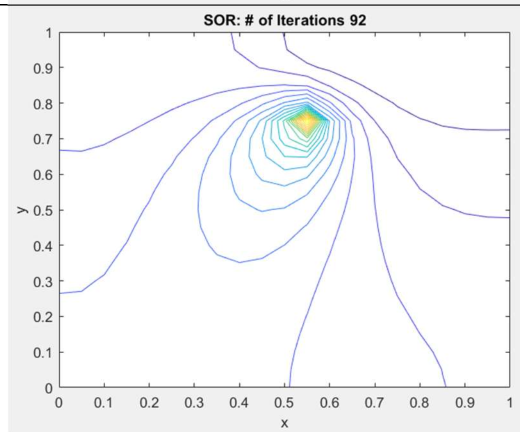
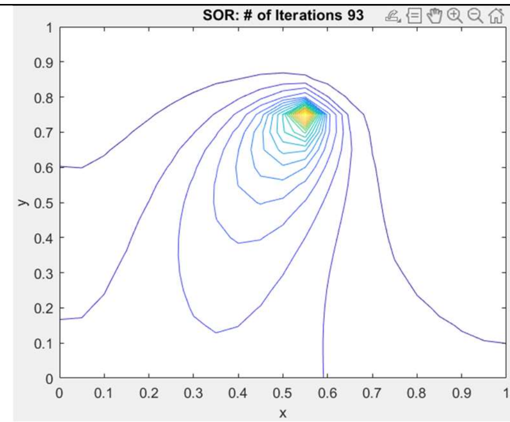
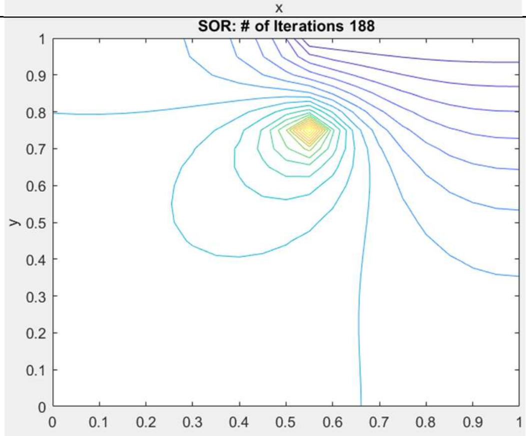
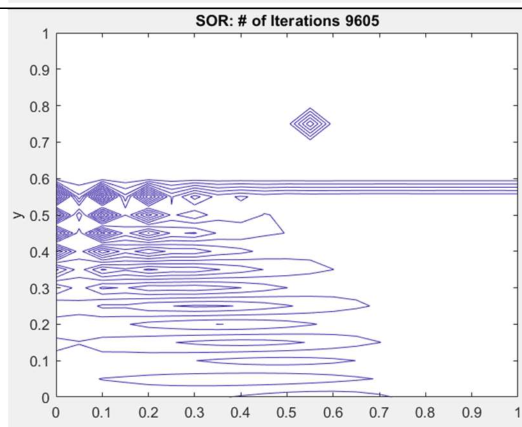
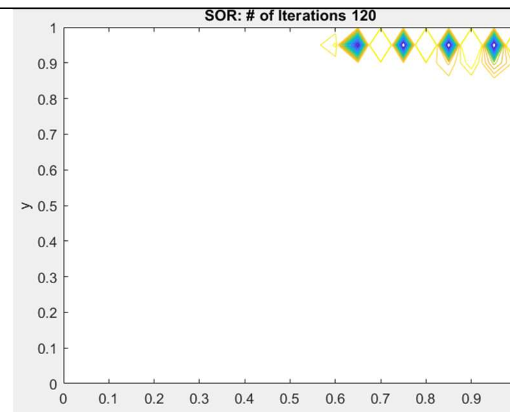
dp/dx = 5	dp/dx = 25	dp/dx = 50
		
		
		

Table 1. Center difference (top), Upstream (mid), Downstream (bottom) at dp/dx = 5, 25, and 50.

From the plots we can see that *dp/dx* = *alpha*, where *alpha* = 5, 25, 50, influences the value of the fluid vector. This fluid vector acts similar to the Peclet number we have seen in class, which determines the oscillatory behavior of our solution in blown-up cases.

```
% ---------------- Problem 2 - Center difference -----------------
clearvars -except Vx Vy x y

%
h=0.05;
w=1.785;

x=[0:h:1];
y=[0:h:1];
A=zeros(length(y), length(x));
A(16,12)=1;
% ---- BC -------
error=1;
itr=0;
pitr=0;
while (error > 1e-5 & itr < 10000)
     itr=itr+1;
     Aold=A;
     A(12,16)=1;
     for i=1:length(y) % row
        for j=1:length(x) % col
            if i==1 % bottom condition
                if j==1 % bottom left
                    A(i,j)=w/8*( 4*Aold(i+1,j) + 4*Aold(i,j+1) ) + (1-
w)*Aold(i,j);
                elseif 1<j & j<length(x) % bottom condition
                    A(i,j)=w/8*( 4*Aold(i+1,j) + Aold(i,j+1)*(2-h*Vy(i,j)) +
A(i,j-1)*(2+h*Vy(i,j)) ) + (1-w)*Aold(i,j);
                elseif j==length(x) % bottom right corner
                    A(i,j)=w/8*( 4*Aold(i+1,j) + 4*A(i,j-1) ) + (1-
w)*Aold(i,j);
                end

            elseif 1<i & i<length(y) % middle
                if j==1 % left
                    A(i,j)=w/8*( Aold(i+1,j)*(2-h*Vx(i,j)) + 4*Aold(i,j+1) +
A(i-1,j)*(2+h*Vx(i,j)) ) + (1-w)*Aold(i,j);
                elseif 1<j & j < length(x) % in middle
                    if i==16 & j==12 % set before loop but just set again
here
                        A(i,j)=1;
                    else
                        A(i,j)=w/8*(    Aold(i+1,j)*(2-h*Vx(i,j)) + ...
                                        Aold(i,j+1)*(2-h*Vy(i,j)) + ...
                                        A(i-1,j) * (2+h*Vx(i,j)) + ...
                                        A(i,j-1)* (2+h*Vy(i,j))  ) + (1-
w)*Aold(i,j);
                    end
                elseif j==length(x)  % right condition
```

```
                    A(i,j)=w/8*( Aold(i+1,j) * (2-h*Vx(i,j)) + A(i-1,j) *
(2+h*Vx(i,j)) + 4*A(i,j-1) ) + (1-w)*Aold(i,j);
                end

            elseif i==length(y) % top condion segment by j
                if j==1 % top left corner
                    A(i,j)=w/8*( 4*A(i-1,j) + 4*Aold(i,j+1) ) + (1-
w)*Aold(i,j);
                elseif 1< j & j<=11
                    A(i,j)=w/8*( 4*A(i-1,j) + Aold(i,j+1)*(2-h*Vy(i,j)) +
A(i,j-1)*(2+h*Vy(i,j)) )+(1-w)*Aold(i,j);
                elseif 11<j & j <= 21
                    A(i,j)=0;
                end
            end
        end
      end
     error=max(max(abs(A-Aold)) / max(max(abs(A)));
     pitr=pitr+1;
    if pitr==5
       figure(3)
       contour(y,x,A,20);
       pitr=0;
    end
end

fprintf('SOR Iterations %d\n',itr);
figure(3)
contour(y,x,A,20);
xlabel('x')
ylabel('y')
title(['SOR: # of Iterations ' num2str(itr)]);

%----------------------- Upstream-----------------------
%
h=0.05;
w=1.785;

x=[0:h:1];
y=[0:h:1];
A=zeros(length(y), length(x));
A(16,12)=1;
% ---- BC -------
error=1;
itr=0;
pitr=0;
while (error > 1e-5 & itr < 10000)
     itr=itr+1;
     Aold=A;
     A(12,16)=1;
     for i=1:length(y) % row
        for j=1:length(x) % col
            if i==1 % bottom condition
                if j==1 % bottom left
```

```
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( (2-
Vx(i,j)*h)*Aold(i+1,j) + (2-Vy(i,j)*h)*Aold(i,j+1) ) + (1-w)*Aold(i,j);
                elseif 1<j & j<length(x) % bottom condition
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( (2-
Vx(i,j)*h)*Aold(i+1,j) + Aold(i,j+1)*(1-h*Vy(i,j)) + A(i,j-1) ) + (1-
w)*Aold(i,j);
                elseif j==length(x) % bottom right corner
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( (2-
Vx(i,j)*h)*Aold(i+1,j) + (2-Vy(i,j)*h)*A(i,j-1) ) + (1-w)*Aold(i,j);
                end

            elseif 1<i & i<length(y) % middle
                if j==1 % left
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( Aold(i+1,j)*(1-
h*Vx(i,j)) + (2-Vy(i,j)*h)*Aold(i,j+1) + A(i-1,j) ) + (1-w)*Aold(i,j);
                elseif 1<j & j < length(x) % in middle
                    if i==16 & j==12 % set before loop but just set again
here
                        A(i,j)=1;
                    else
                        A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) ...
                                *(  Aold(i+1,j)*(1-h*Vx(i,j)) + ...
                                    Aold(i,j+1)*(1-h*Vy(i,j)) + ...
                                    A(i-1,j) + ...
                                    A(i,j-1)  ) + (1-w)*Aold(i,j);
                    end
                elseif j==length(x)  % right condition
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( Aold(i+1,j) *
(1-h*Vx(i,j)) + A(i-1,j) + A(i,j-1)* (2-h*Vy(i,j)) ) + (1-w)*Aold(i,j);
                end

            elseif i==length(y) % top condion segment by j
                if j==1 % top left corner
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( A(i-1,j)*(2-
Vx(i,j)*h) + Aold(i,j+1)*(2-Vy(i,j)*h) ) + (1-w)*Aold(i,j);
                elseif 1< j & j<=11
                    A(i,j)=w/(4 -Vx(i,j)*h - Vy(i,j)*h ) *( A(i-1,j)*(2-
Vx(i,j)*h) + Aold(i,j+1)*(1-h*Vy(i,j)) + A(i,j-1) )+(1-w)*Aold(i,j);
                elseif 11<j & j <= 21
                    A(i,j)=0;
                end
            end
        end
      end
     error=max(max(abs(A-Aold)) / max(max(abs(A)));
     pitr=pitr+1;
    if pitr==5
       figure(4)
       contour(y,x,A,20);
       pitr=0;
    end
end

fprintf('SOR Iterations %d\n',itr);
figure(4)
```

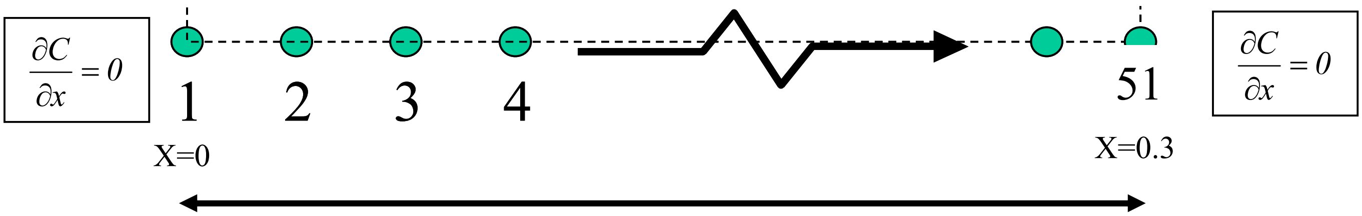
```
contour(y,x,A,20);
xlabel('x')
ylabel('y')
title(['SOR: # of Iterations ' num2str(itr)]);

%
%----------------------- Downstream-----------------------
h=0.05;
w=1.785;

x=[0:h:1];
y=[0:h:1];
A=zeros(length(y), length(x));
A(16,12)=1;
% ---- BC -------
error=1;
itr=0;
pitr=0;
while (error > 1e-5 & itr < 10000)
     itr=itr+1;
     Aold=A;
     A(12,16)=1;
     for i=1:length(y) % row
        for j=1:length(x) % col
            if i==1 % bottom condition
                if j==1 % bottom left
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *( (2+Vx(i,j)*h)
*Aold(i+1,j) + (2+Vy(i,j)*h) *Aold(i,j+1) ) + (1-w)*Aold(i,j);
                elseif 1<j & j<length(x) % bottom condition
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *( (2+Vx(i,j)*h)
*Aold(i+1,j) + Aold(i,j+1) + A(i,j-1)*(1+h*Vy(i,j)) ) + (1-w)*Aold(i,j);
                elseif j==length(x) % bottom right corner
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *( (2+Vx(i,j)*h)
*Aold(i+1,j) + (2+Vy(i,j)*h) *A(i,j-1) ) + (1-w)*Aold(i,j);
                end

            elseif 1<i & i<length(y) % middle
                if j==1 % left
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *( Aold(i+1,j) +
(2+Vy(i,j)*h) *Aold(i,j+1) + A(i-1,j)*(1+h*Vx(i,j)) ) + (1-w)*Aold(i,j);
                elseif 1<j & j < length(x) % in middle
                    if i==16 & j==12 % set before loop but just set again
here
                        A(i,j)=1;
                    else
                        A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) ...
                                *(    Aold(i+1,j) + ...
                                        Aold(i,j+1) + ...
                                        A(i-1,j) * (1+h*Vx(i,j)) + ...
                                        A(i,j-1)* (1+h*Vy(i,j))   ) + (1-
w)*Aold(i,j);
                    end
                elseif j==length(x)  % right condition
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *( Aold(i+1,j) +
A(i-1,j) * (1+h*Vx(i,j)) + (2+h*Vy(i,j)) *A(i,j-1) ) + (1-w)*Aold(i,j);
```

```
                end

            elseif i==length(y) % top condion segment by j
                if j==1 % top left corner
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *(
(2+h*Vx(i,j))*A(i-1,j) + (2+h*Vy(i,j))*Aold(i,j+1) ) + (1-w)*Aold(i,j);
                elseif 1< j & j<=11
                    A(i,j)=w/(4 + Vx(i,j)*h + Vy(i,j)*h) *(
(2+h*Vx(i,j))*A(i-1,j) + Aold(i,j+1) + A(i,j-1)*(1+h*Vy(i,j)) )+(1-
w)*Aold(i,j);
                elseif 11<j & j <= 21
                    A(i,j)=0;
                end
            end
        end
      end
     error=max(max(abs(A-Aold))) / max(max(abs(A)));
     pitr=pitr+1;
    if pitr==5
       figure(5)
       contour(y,x,A,20);
       pitr=0;
    end
end

fprintf('SOR Iterations %d\n',itr);
figure(5)
contour(y,x,A,20);
xlabel('x')
ylabel('y')
title(['SOR: # of Iterations ' num2str(itr)]);
```

Problem 3. Understanding Diffusion. The equation used to describe one-dimensional drug diffusion is:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right)$$

(a) With the above formulation, write out the *finite difference expression* including the leading truncation error terms. Assume a purely ***explicit*** formulation.



Time step error

$u_{i+1} = u_i + h u_i' + \frac{h^2}{2!}u_i'' + \frac{h^3}{3!}u_i''' + \dots$

$u_i' = \frac{u_{i+1} - u_i}{h} - \frac{h}{2!}u_i'' - \dots\dots$ (1)

$u_{i+1} = u_i + h u_i' + \frac{h^2}{2!}u_i'' + \frac{h^3}{3!}u''' + \frac{h^4}{4!}u_i''''$

$+$

$u_{i-1} = u_i - h u_i' + \frac{h^2}{2!}u_i'' - \frac{h^3}{3!}u''' + \frac{h^4}{4!}u_i''''$

$u_{i+1} + u_{i-1} = 2u_i + h^2 u_i'' + \frac{h^4}{12}u_i''''$

Thus $u_i'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - \frac{h^2}{12}u_i''''$ (2)

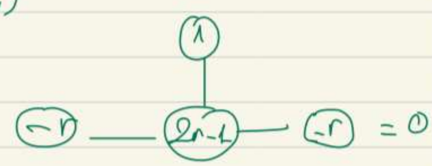
From (1) and (2)

Then for drug diffusion; with $k = \Delta t$

$$\frac{u_i(t+\Delta t) - u_i(t)}{\Delta t} - \frac{k}{2!}u_i'' = D\left(\frac{\delta_x^2 u_i^l}{h^2} - \frac{h^2}{12}u_i''''\right)$$

$$u_i^{l+1} - u_i^l + O(k) = \frac{Dk}{h^2}\left(u_{i+1}^l - 2u_i^l + u_{i-1}^l\right) + O(h^2)$$

$r = \frac{Dk}{h^2}$



$$u_i^{l+1} = u_i^l(1-2r) + u_{i+1}^l r + u_{i-1}^l r + O(k + h^2)$$

(b) Now let’s assume a bolus of some chemoreactant solution has been placed within a column of tissue that represents the initial distribution in the tissue. The bolus is can be respresented as:

$$C(t=0) = \frac{2.5}{\sqrt{2\pi}} e^{-(30x-4.5)^2/2}$$

Using the scheme derived in part (a), **the initial condition found in the above equation,** and the problem description above, generate the time varying solution for this problem. For this problem assume, *dt=0.004, D=0.001*. Allow the process to come to steady state by monitoring a *relative* L_∞ *norm-based change* and stopping the process when this error goes below *1e-4*. Using an overlay function, plot the solutions at *(t=0, 0.2, 0.4, 0.6, 1.0, 1.8, 3, 4 and final time at tolerance)*. Also report the **number of time steps** and **final time at tolerance** satisfaction.

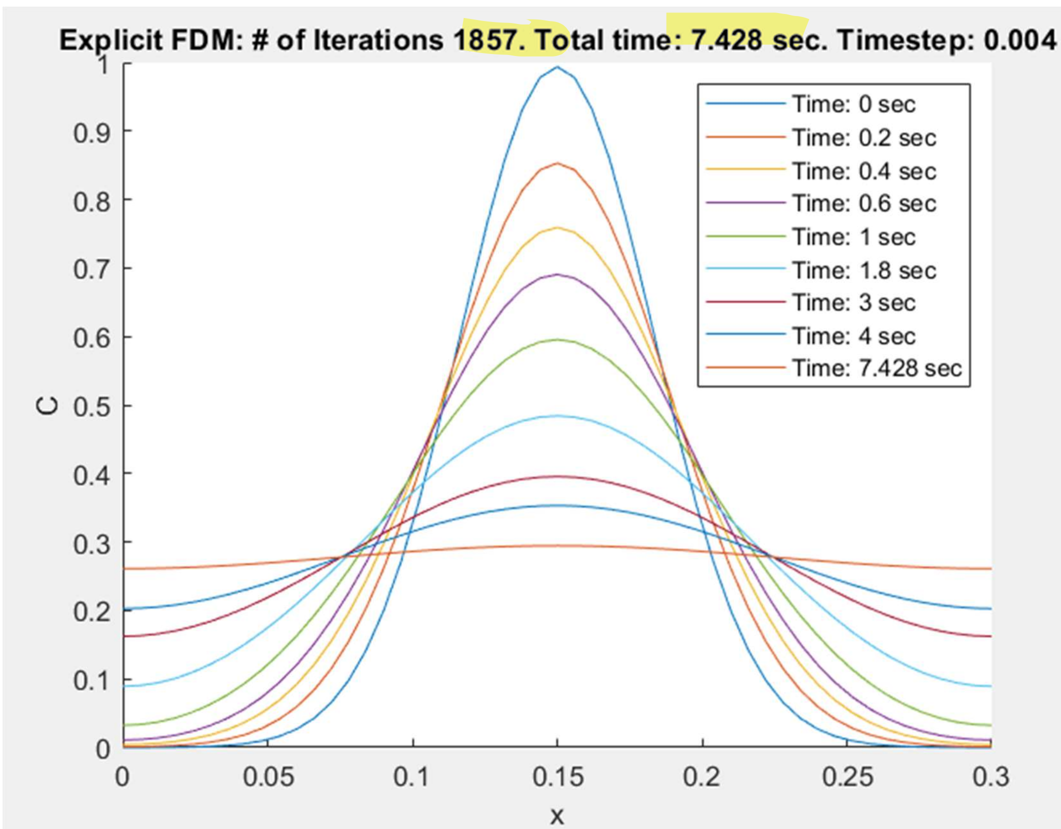
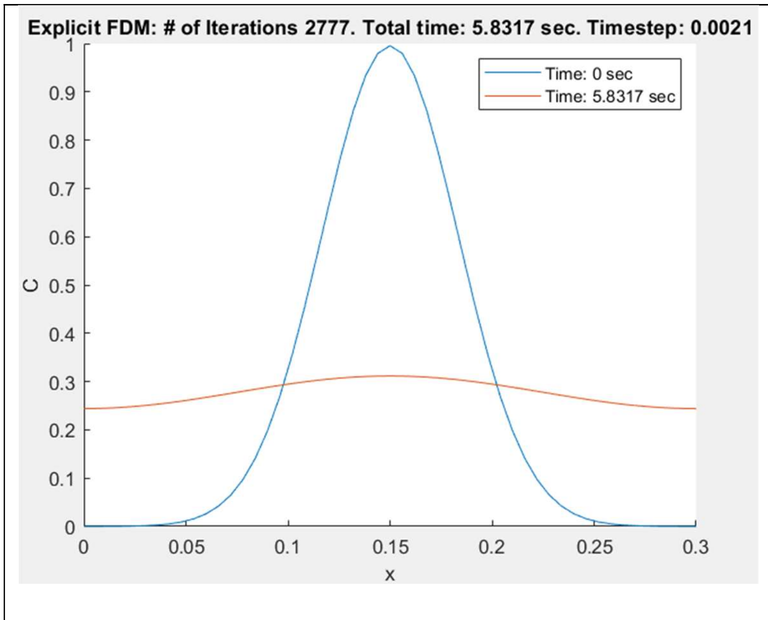
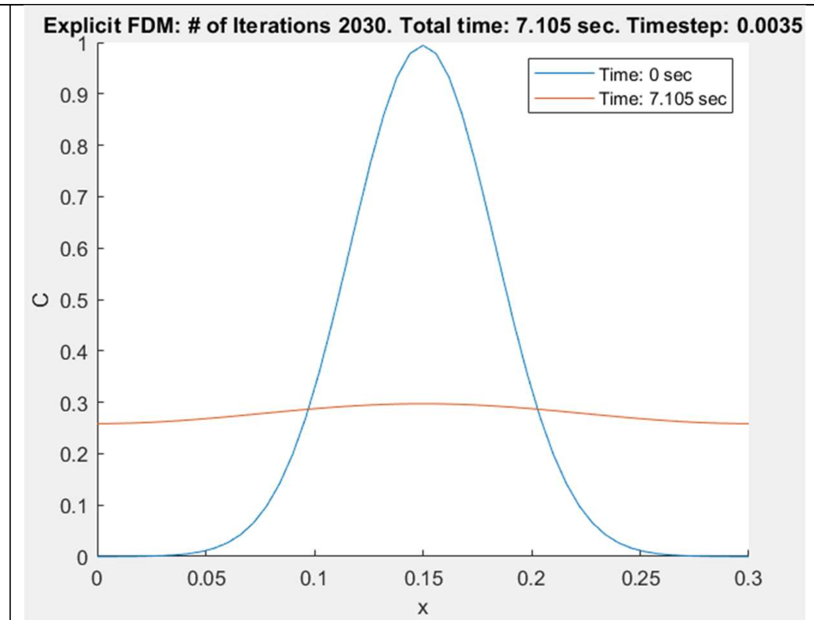


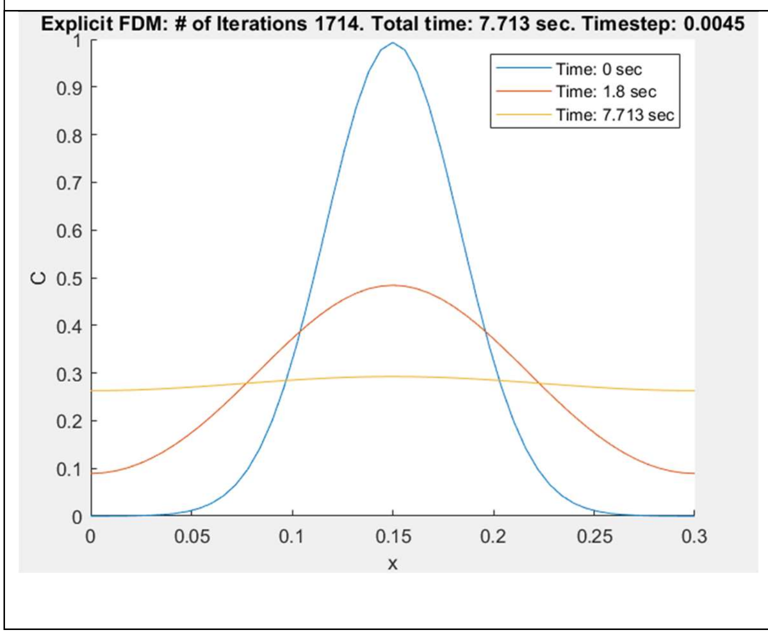
Figure 3. Concentration at different times overlaid.

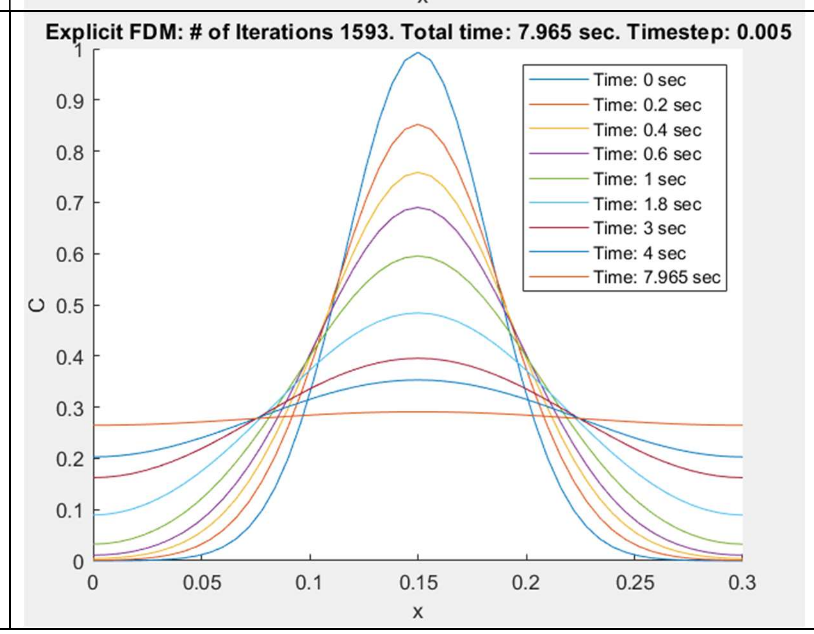
(c) Now, investigate your numerical algorithm from part **(b)**, specifically, does your model go **unstable at different time steps**? if so, can you determine thresholds for when instability occurs empirically?

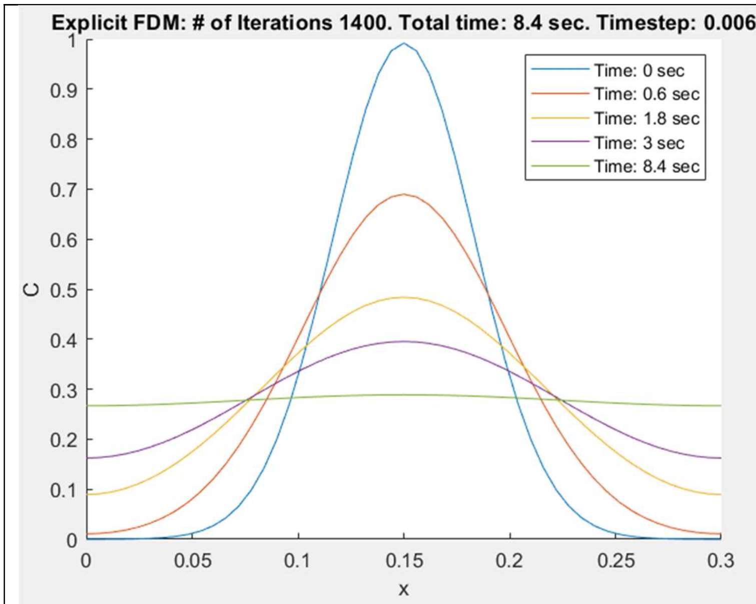
Testing varying time steps from [0.002 – 0.03], **dt = 0.019** showed a blown-up in the simulation.

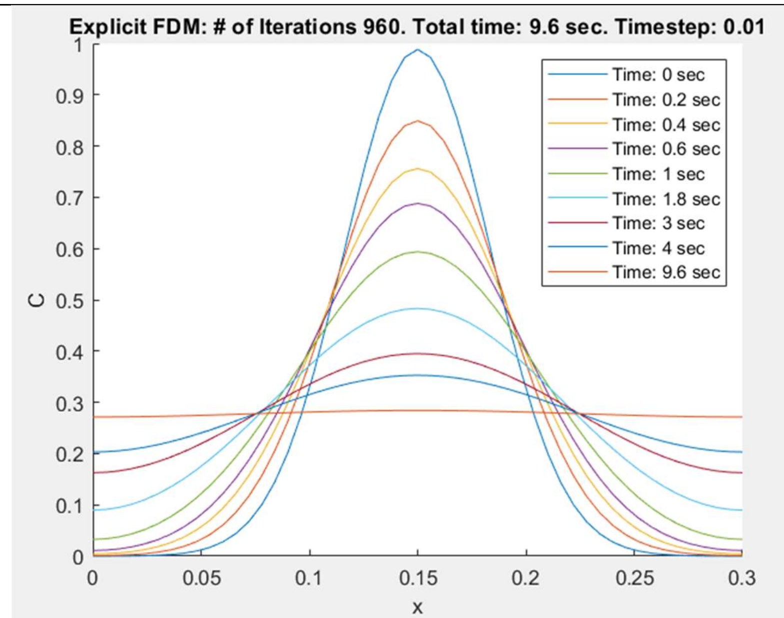

Explicit FDM: # of Iterations 2777. Total time: 5.8317 sec. Timestep: 0.0021
Time: 0 sec
Time: 5.8317 sec
C
x

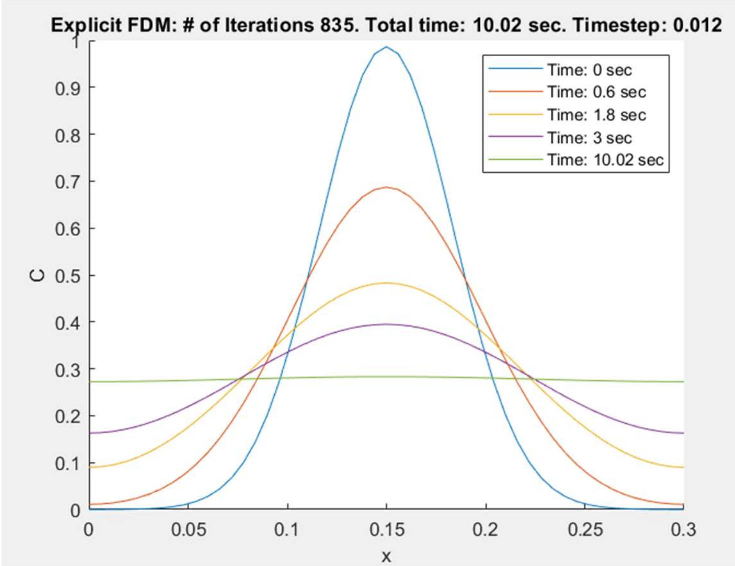

Explicit FDM: # of Iterations 2030. Total time: 7.105 sec. Timestep: 0.0035
Time: 0 sec
Time: 7.105 sec
C
x

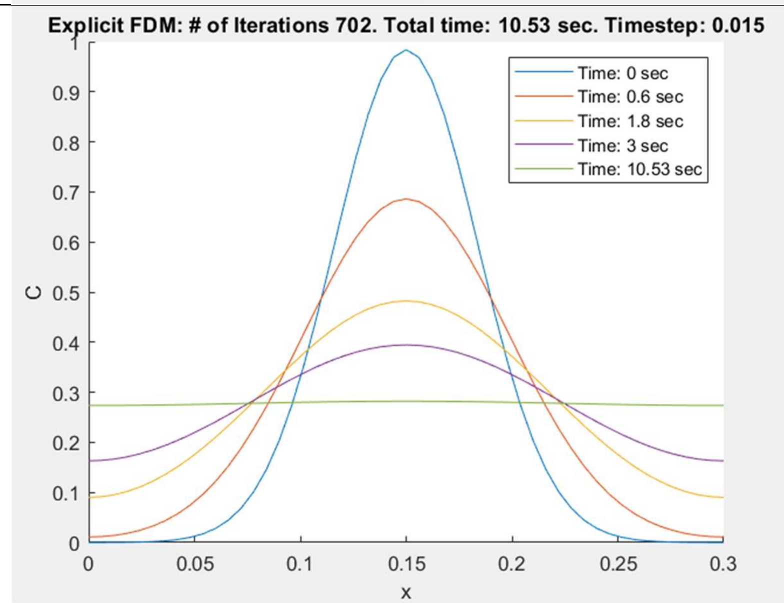

Explicit FDM: # of Iterations 1714. Total time: 7.713 sec. Timestep: 0.0045
Time: 0 sec
Time: 1.8 sec
Time: 7.713 sec
C
x

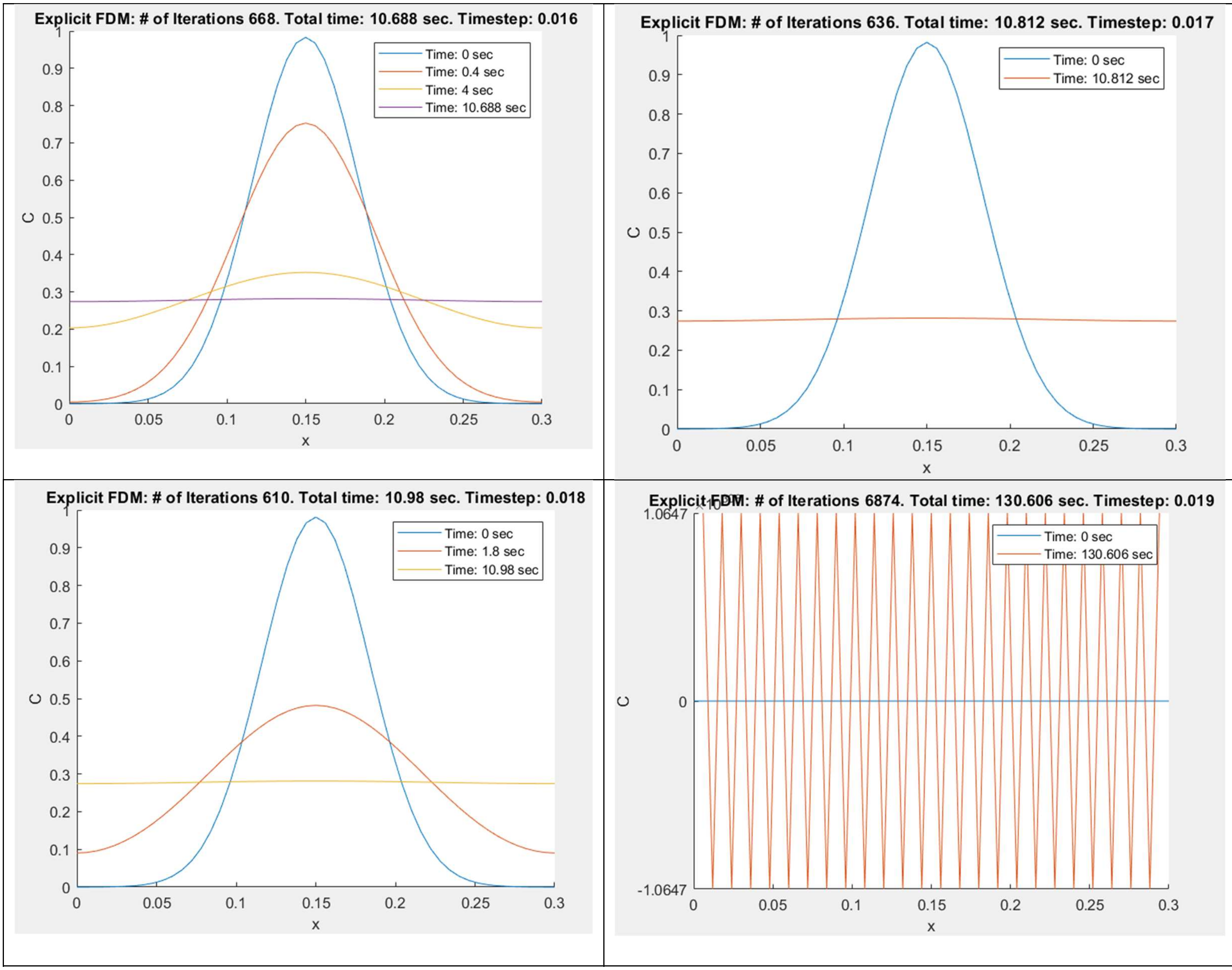

Explicit FDM: # of Iterations 1593. Total time: 7.965 sec. Timestep: 0.005
Time: 0 sec
Time: 0.2 sec
Time: 0.4 sec
Time: 0.6 sec
Time: 1 sec
Time: 1.8 sec
Time: 3 sec
Time: 4 sec
Time: 7.965 sec
C
x


Explicit FDM: # of Iterations 1400. Total time: 8.4 sec. Timestep: 0.006
Time: 0 sec
Time: 0.6 sec
Time: 1.8 sec
Time: 3 sec
Time: 8.4 sec
C
x


Explicit FDM: # of Iterations 960. Total time: 9.6 sec. Timestep: 0.01
Time: 0 sec
Time: 0.2 sec
Time: 0.4 sec
Time: 0.6 sec
Time: 1 sec
Time: 1.8 sec
Time: 3 sec
Time: 4 sec
Time: 9.6 sec
C
x


Explicit FDM: # of Iterations 835. Total time: 10.02 sec. Timestep: 0.012
Time: 0 sec
Time: 0.6 sec
Time: 1.8 sec
Time: 3 sec
Time: 10.02 sec
C
x


Explicit FDM: # of Iterations 702. Total time: 10.53 sec. Timestep: 0.015
Time: 0 sec
Time: 0.6 sec
Time: 1.8 sec
Time: 3 sec
Time: 10.53 sec
C
x



(d) Looking at the time history of results from part **(b)**, discuss the implication of the boundary conditions. Are there any other interesting points about your solution regarding the nature of how this system works? Intepret the solutions generated with respect to your understanding of the boundary conditions and the application.

Answer: BCs causes the Solution to run ahead of itself, especially the inner part of the domain.

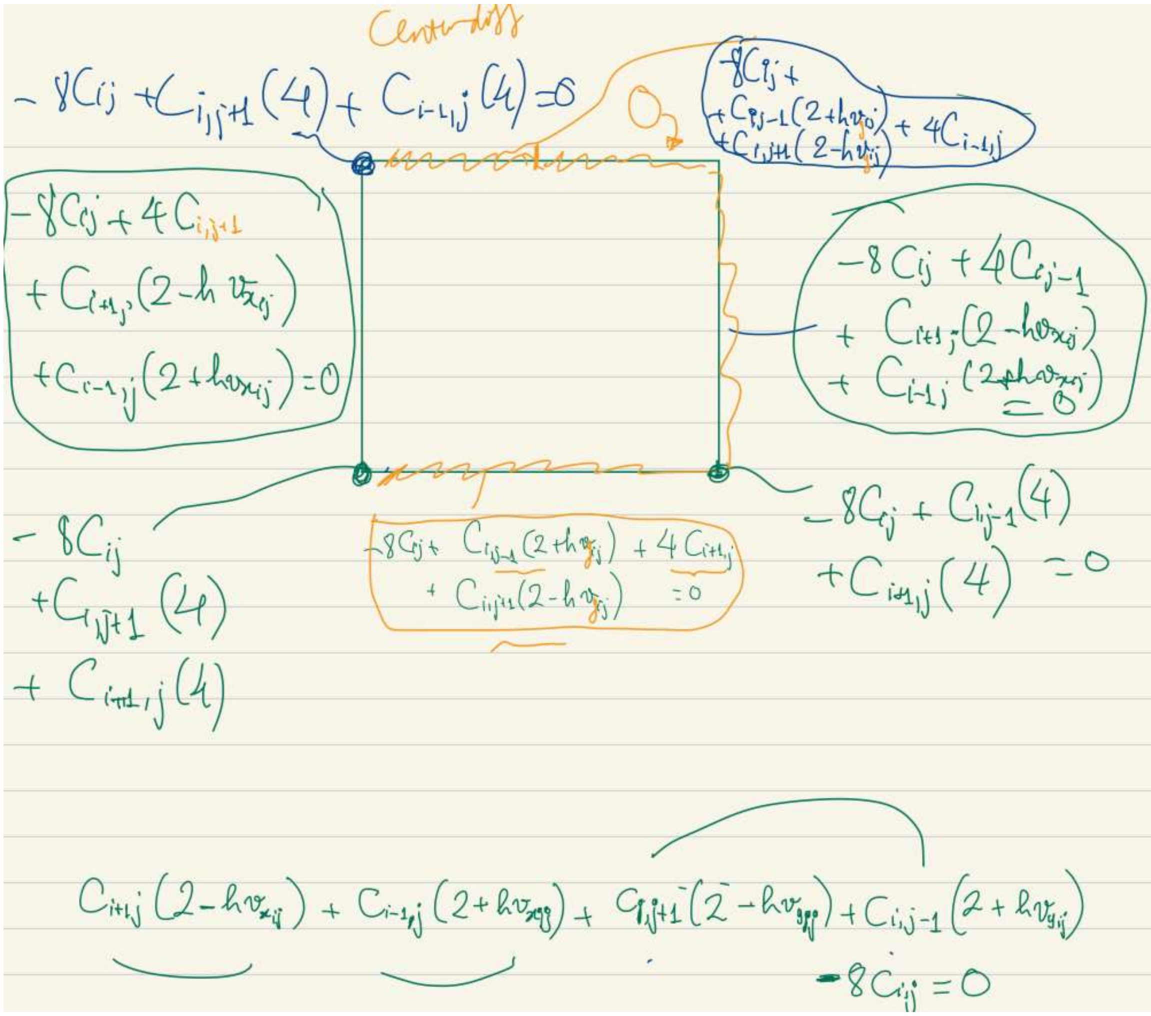
There must be a value of time step that must not be exceed, in our case threshold *dt = 0.019*.

Obviously, we see the gradual diffusion associated with this high spike of concentration. The interesting point regarding the boundary conditions is that no-flux conditions have been

```
%% ------------------ Diffusion - Problem 3 ----------------------
```

```matlab
clear all
h=0.006;
w=1.785;
x=[0:h:0.3];
A= 2.5 ./ (sqrt(2*pi)) .* ( exp(-(30 .*x - 4.5).^2./2) );
figure(20), clf
hold all

% ---- BC -------
error=1;
itr=0;
time_points = [0, 0.2, 0.4, 0.6, 1.0, 1.8, 3, 4];
tolCheck = 10*eps;
dt=0.004;
D = 0.001;
r = D*dt/(h^2);
while (error > 1e-4 & itr < 10000)
     Aold=A;
     for i=1:length(x)
        if i==1
            A(i) = Aold(i)*(1-2*r) + 2*Aold(i+1)*r;
        elseif 1<i & i<length(x)
            A(i) = Aold(i)*(1-2*r) + Aold(i+1)*r + Aold(i-1)*r;
        elseif i==length(x)
            A(i) = Aold(i)*(1-2*r) + 2*Aold(i-1)*r;
        end
     end
     error=max(max(abs(A-Aold)))/max(max(abs(A)));
    if any(abs(time_points - itr*dt) <= tolCheck)
       figure(20)
       plot(x,A, 'DisplayName',['Time: '+  string(itr*dt)+' sec'])
    end
    itr=itr+1;
end

fprintf('Explicit FDM Iterations %d\n',itr);
figure(20)
plot(x,A, 'DisplayName',['Time: '+  string(itr*dt)+' sec'])
legend(gca,'show')
hold off
xlabel('x')
ylabel('C')
title(['Explicit FDM: # of Iterations '+ string(itr)+ '. Total time: ' +
string(itr*dt)+ ' sec. ' + 'Timestep: '+ string(dt)]);
```

Appendix


Center diff
- 8C_ij + C_i,j+1 (4) + C_i-1,j (4) = 0
-8C_ij + C_i,j-1 (2+hv_yij) + C_i,j+1 (2-hv_yij) + 4C_i-1,j
-8C_ij + 4C_i,j+1 + C_i+1,j (2 - h v_xij) + C_i-1,j (2 + hv_xij) = 0
-8 C_ij + 4C_i,j-1 + C_i+1,j (2 - hv_xij) + C_i-1,j (2 + hv_xij) = 0
- 8C_ij + C_i,j+1 (4) + C_i+1,j (4)
-8C_ij + C_i,j-1 (2+hv_yij) + 4 C_i+1,j + C_i,j+1 (2-hv_yij) = 0
-8C_ij + C_i,j-1 (4) + C_i+1,j (4) = 0
C_i+1,j (2 - hv_xij) + C_i-1,j (2 + hv_xij) + C_i,j+1 (2 - hv_yij) + C_i,j-1 (2 + hv_yij) - 8C_ij = 0

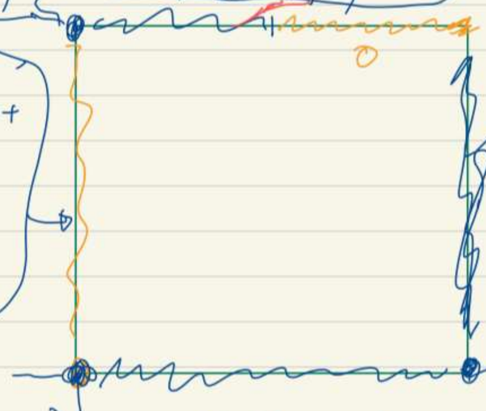
Upstream

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)$
$C_{i-1j}(2-v_{x_{ij}}h)$
$C_{i,j+1}(1-v_{y_{ij}}h)+C_{i,j-1}=0$

$$C_{i+1,j}(1-v_{x_{ij}}h)+C_{i-1j}+C_{i,j+1}(1-v_{y_{ij}}h)+C_{i,j-1}$$
$$+C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)=0$$

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)$
$+C_{i,j+1}(2-v_{y_{ij}}h)+C_{i-1,j}(2-v_{x_{ij}}h)=0$

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)+$
$C_{i,j-1}(2-v_{y_{ij}}h)$,
$+C_{i-1,j}$
$+C_{i+1,j}(1-v_{x_{ij}}h)=0$



$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)+$
$C_{i,j+1}(2-v_{y_{ij}}h)$
$+C_{i-1,j}$
$+C_{i+1,j}(1-v_{x_{ij}}h)=0$

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)$
$C_{i+1,j}(2-v_{x_{ij}}h)$
$C_{i,j-1}(2-v_{y_{ij}}h)$
$=0$

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)$
$C_{i+1,j}(2-v_{x_{ij}}h)$
$C_{i,j+1}(2-v_{y_{ij}}h)=0$

$C_{ij}(-4+v_{x_{ij}}h+v_{y_{ij}}h)$
$+C_{i+1,j}(2-v_{x_{ij}}h)$
$+C_{i,j+1}(1-v_{y_{ij}}h)+C_{i,j-1}=0$

Down stream

$$C_{i+1,j} + C_{i,j+1} + C_{i-1,j}(1+v_{x_{ij}}h) + C_{i,j-1}(1+v_{y_{ij}}h) + C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h) = 0$$

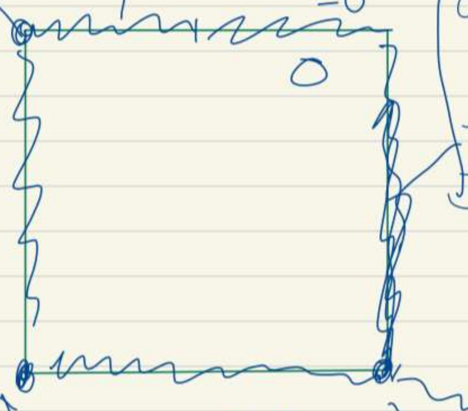
$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h)$

$C_{i+1,j}(2 + v_{x_{ij}}h)$

$C_{i,j+1}(2 + v_{y_{ij}}h)$

$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h) + C_{i,j-1}(1 + v_{y_{ij}}h) + C_{i,j+1} + C_{i-1,j}(2 + v_{x_{ij}}h) = 0$

$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h) + C_{i,j-1}(2 + v_{y_{ij}}h) + C_{i-1,j}(1 + v_{x_{ij}}h) + C_{i+1,j} = 0$



$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h) + C_{i,j+1}(2 + v_{y_{ij}}h) + C_{i-1,j}(1 + v_{x_{ij}}h) + C_{i+1,j} = 0$

$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h)$

$C_{i+1,j}(2 + v_{x_{ij}}h)$

$C_{i,j+1}(2 + v_{y_{ij}}h)$

$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h)$

$C_{i+1,j}(2 + v_{x_{ij}}h)$

$C_{i,j+1}$

$C_{i,j-1}(1 + v_{y_{ij}}h)$

$C_{ij}(-4 - v_{x_{ij}}h - v_{y_{ij}}h$

$C_{i+1,j}(2 + v_{x_{ij}}h)$

$C_{i,j-1}(2 + v_{y_{ij}}h)$