Sahib Azgar
ME4565

Due: 01-31-24
Prof Allshouse

ME 4565: Introduction to Computational Fluid Dynamics

Spring 2024

Problem Set #3

Due January 31st, 2:50 PM (Submit to Canvas)

Problem 1 - First Order ODE: For the equation

$$\frac{dx}{dt} = -tx$$

a. Derive the analytic solution using separation of variables. Use the initial condition that $x(0) = 1$.

(Hand derivation)

$\frac{dx}{dt} = -tx \rightsquigarrow \int \frac{1}{x}\,dx = \int -t\,dt \qquad \ln|x| = -\frac{t^2}{2} + C$

$e \qquad e$

$x(t) = e^{\frac{-t^2}{2}} \cdot e^{C} \qquad x(0) = 1$

$\boxed{x(t) = e^{-0.5t^2}}$

$1 = e^{0/2} \cdot e^{C}$

$\ln 1 = \ln e^{C}$

$0 = C$

b. Derive the explicit Euler method equation.

(Hand derivation)

$x_{n+1} = x_n + \Delta t \frac{dx}{dt}$

given: $\frac{dx}{dt} = -tx$

$\Delta t = 0.1$

$x_{n+1} = x_n + \Delta t(-t x_n)$

$\boxed{x_{n+1} = x_n - x_n t_n(\Delta t)}$

Problem 1c) Write a code that performs 10 iterations of the explicit Euler method to approximate the value of x(t=1), where the time step used is Δt =0.1. (Code and approximated value).

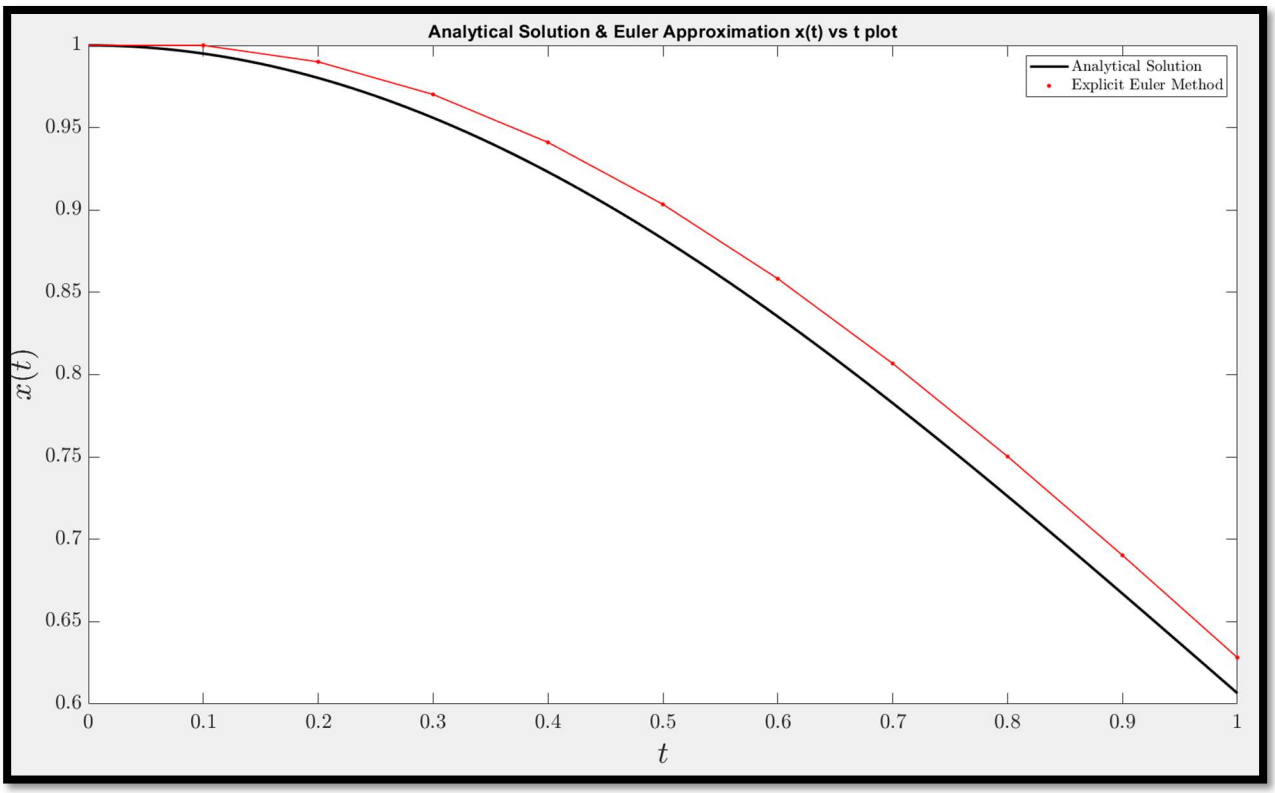


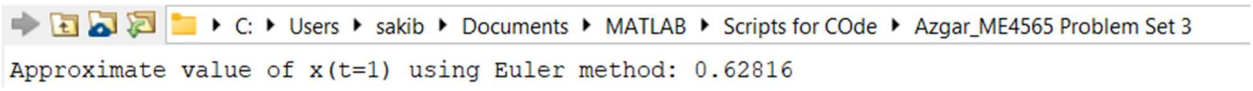
Figure 1: X(t) vs t plot comparing Analytical and Explicit Euler Solution

Script for Problem 1c Below: (HW3P1C)

```
clear; close all; clc;

%% Plot the analytical solution
t_analytic = 0:.01:1; % Define time points for the analytical solution
x_analytic = exp(-0.5 * t_analytic.^2); % Calculate analytical solution
plot(t_analytic, x_analytic, 'k', 'LineWidth', 2); % Plot the analytical solution
hold on; % Hold the plot for further additions
set(gca, 'FontSize', 16, 'TickLabelInterpreter', 'latex') % Set plot properties
xlabel('$$t$$', 'FontSize', 24, 'Interpreter', 'latex') % Set x-axis label
ylabel('$$x(t)$$', 'FontSize', 24, 'Interpreter', 'latex') % Set y-axis label

%% Perform Explicit Euler Solver
dt = 0.1; % Set the time step
x_n = 1; % Set the initial condition
t = 0:dt:1; % Create a list of times for approximating x

for n = 1:length(t)-1 % Loop over each time step
    t_n = t(n); % Current time

    % Explicit Euler step to get the next state
    x_np1 = x_n - dt * t_n * x_n;
```

```
    % Plot the step
    plot(t(n+1), x_np1, '.r', 'MarkerSize', 10);

    % Connect the dots with a red line
    plot([t_n, t(n+1)], [x_n, x_np1], '-r', 'LineWidth', 1);

    % Update the x_n (current state) values
    x_n = x_np1;
end

% Add legend with smaller font size
legend('Analytical Solution', 'Explicit Euler Method', 'Interpreter', 'latex',
'FontSize', 12);
title({'Analytical Solution & Euler Approximation x(t) vs t plot'}, 'FontSize', 14,
'Interpreter', 'latex');

% Display the approximate value of x(t=1)
disp(['Approximate value of x(t=1) using Euler method: ', num2str(x_n)]);
```

Command Window Display



Problem 2 - Particle Advection in an Analytic Flow: For the stream function

$$\psi = \frac{1}{1+x^2} + \frac{1}{1+y^2}$$

a. Derive the velocity field $\mathbf{u} = (u, v)$ using the stream function.

(Hand derivation)

$\psi \to u$

$u = \frac{\partial \psi}{\partial y}$ $\quad v = \frac{-\partial \psi}{\partial x}$

$\psi = \frac{1}{1+x^2} + \frac{1}{1+y^2}$

$u = \frac{\partial}{\partial y}\left[\frac{1}{1+x^2} + \frac{1}{1+y^2}\right]$ $\quad v = \frac{-\partial}{\partial x}\left[\frac{1}{1+x^2} + \frac{1}{1+y^2}\right]$

$u = \frac{\partial}{\partial y}\left[\frac{1}{1+y^2}\right]$ $\quad v = -\frac{\partial}{\partial x}\left[\frac{1}{1+x^2}\right]$

bdeki - hideki / b^2

$u = \frac{(1+y^2)(0) - 1(2y)}{(1+y^2)^2}$ $\quad v = -\left(\frac{-2x}{(1+x^2)^2}\right) \to v = \frac{2x}{(1+x^2)^2}$

$u = \frac{-2y}{(1+y^2)^2}$

$$\vec{u} = \left(\frac{-2y}{(1+y^2)^2}, \frac{2x}{(1+x^2)^2}\right)$$

b. Derive the system of explicit Euler equations you will need to use to advect particles moving within this flow. As we saw in class, the trajectories $(x(t), y(t))$ of particles in the system satisfy the equations

$$\frac{dx}{dt} = u(x, y), \qquad \frac{dy}{dt} = v(x, y).$$

(Hand derivation)

$u = -\frac{2y}{(1+y^2)^2}$ $\quad v = \frac{2x}{(1+x^2)^2}$

recall Euler eqs:

$x_{n+1} = x_n + \Delta t \cdot \frac{dx}{dt}$ $\quad y_{n+1} = y_n + \Delta t \frac{dy}{dt}$

$x_{n+1} = x_n + \Delta t \cdot u(x_n, y_n)$ $\quad y_{n+1} = y_n + \Delta t \cdot v(x_n, y_n)$

$$x_{n+1} = x_n - \Delta t \left(\frac{2y_n}{(1+y_n^2)^2}\right)$$

$$y_{n+1} = y_n + \Delta t \cdot \left(\frac{2x_n}{(1+x_n^2)^2}\right)$$

Problem 2c) Write a function, euler_advect that takes in as inputs (t,x0,y0) where t is the time vector for the solution and (x0,y0) are a pair of scalars corresponding to the initial position of the trajectory. The function should then implement the explicit Euler's solver and output (xTraj, yTraj), which are discretization's of the solutions x(t) and y(t). A template file is available.

Script for Problem 2c Below: (euler_advect)

```
function [xTraj, yTraj] = euler_advect(t, x0, y0)
%% Create our time step and number of time instances
dt = t(2) - t(1); % Calculation per time step
nT = length(t);   % Get # of time instances

%% Create the trajectory variables
xTraj = zeros(size(t)); % Initialize array for x trajectory
yTraj = zeros(size(t)); % Initialize array for y trajectory

%% Set the initial condition
xTraj(1) = x0; % Initial x position set
yTraj(1) = y0; % Initial y position set

% Begin time marching
% Loop over each time, starting from the first time instance
% and ending at the second-to-last time instance (nT-1)
% The for loop will help perform explicit Euler time integration
for n = 1:nT-1
   % Perform the explicit euler steps
   dx_divided_dt = -2 * yTraj(n) / (1 + yTraj(n)^2)^2;  % Calculate x time derivative
   dy_divided_dt = 2 * xTraj(n) / (1 + xTraj(n)^2)^2;    % Calculate y derivative

   xTraj(n+1) = xTraj(n) + dt * dx_divided_dt;  % Explicit Euler Eq for x
   yTraj(n+1) = yTraj(n) + dt * dy_divided_dt;  % Explicit Euler Eq for y
end
```

Problem 2d) Write a script that produces the streamlines for the system using the function streamslice, then calls your function euler advect to solve the system with the initial position (0, 2) over the time interval t = 0 : 0.1 : 25. Plot the particle trajectory on top of your streamlines.

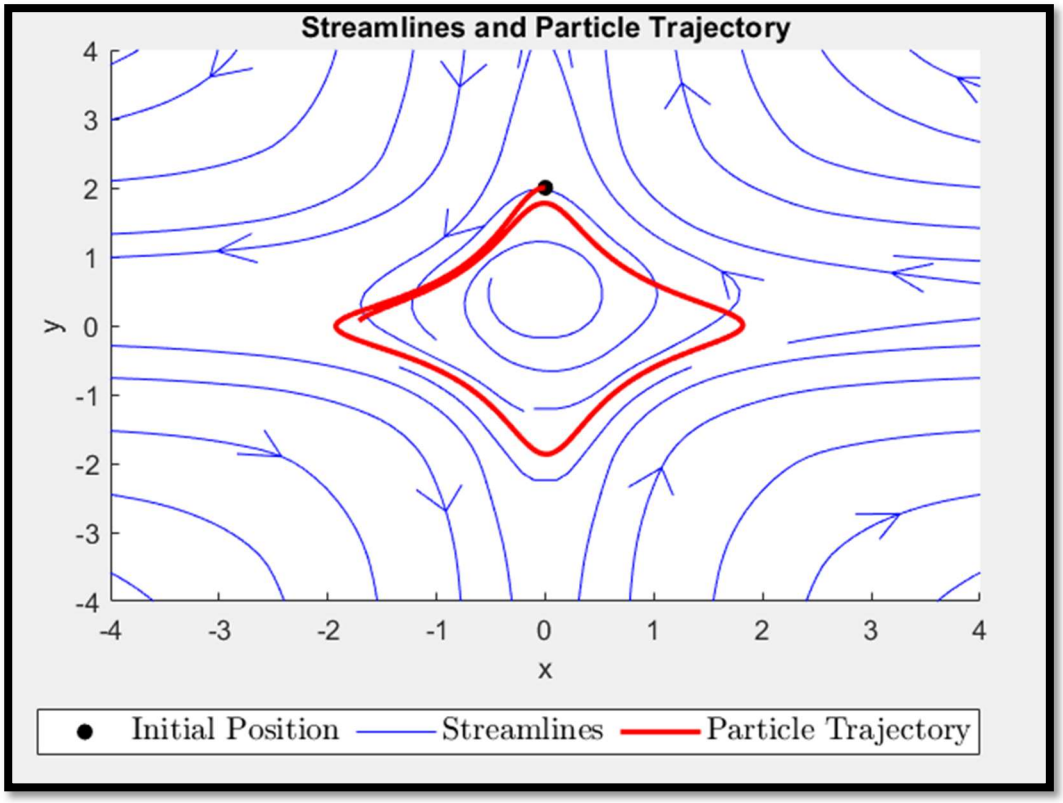


Figure 2: Streamlines and Particle Trajectory Plot using Euler Method

Script for Problem 2d Below: (HW3P2D)

```
% Homework 3 Problem 2 Part D
% Initial conditions
x0 = 0;  % initial x position
y0 = 2;  % initial y position

% Time vector
t = 0:0.1:25;

% Call the euler_advect function
[xTraj, yTraj] = euler_advect(t, x0, y0);

% Generate meshgrid for streamlines
[x, y] = meshgrid(linspace(-10, 10, 20), linspace(-10, 20, 20));

% Define the velocity field based on your system's equations
u = -2 * y ./ (1 + y.^2).^2;
v = 2 * x ./ (1 + x.^2).^2;

% Plot streamlines with a specified color
figure;
hStreamlines = streamslice(x, y, u, v, 2); %use hStreamlines because streamslice
would cause legend to have lines be same color
```

```
set(hStreamlines, 'Color', 'b');  % Set the color to blue

hold on;

% Scatter initial position
scatter(x0, y0, 'k', 'filled');

% Plot particle trajectory on top of streamlines
plot(xTraj, yTraj, 'r', 'LineWidth', 2);

% Set axis limits
axis([-4 4 -4 4]);%axis bounds first two are x and last two are y
title('Streamlines and Particle Trajectory'); %title of graph
xlabel('x'); %x-axis label
ylabel('y'); %y-axis label

% Specify legend entries and colors, place the legend outside at the
% bottom, set orientation to horizontal so it looks nice below x-axis ->
% had to look up how to set this legend up.
legend([scatter(x0, y0, 'k', 'filled'), hStreamlines(1), plot(xTraj, yTraj, 'r',
'LineWidth', 2)], 'Initial Position', 'Streamlines', 'Particle Trajectory',
'Interpreter', 'latex', 'FontSize', 12, 'Location', 'southoutside', 'Orientation',
'horizontal');

hold off;
```

Problem 2e) Modify your script to call euler_advect inside a for-loop where you vary the particle initial position to be (0, 1), (0, 1.5), and (0, 2), for t = 0 : 0.1 : 25. Plot each of your trajectories on the same figure, on top of the streamlines again.

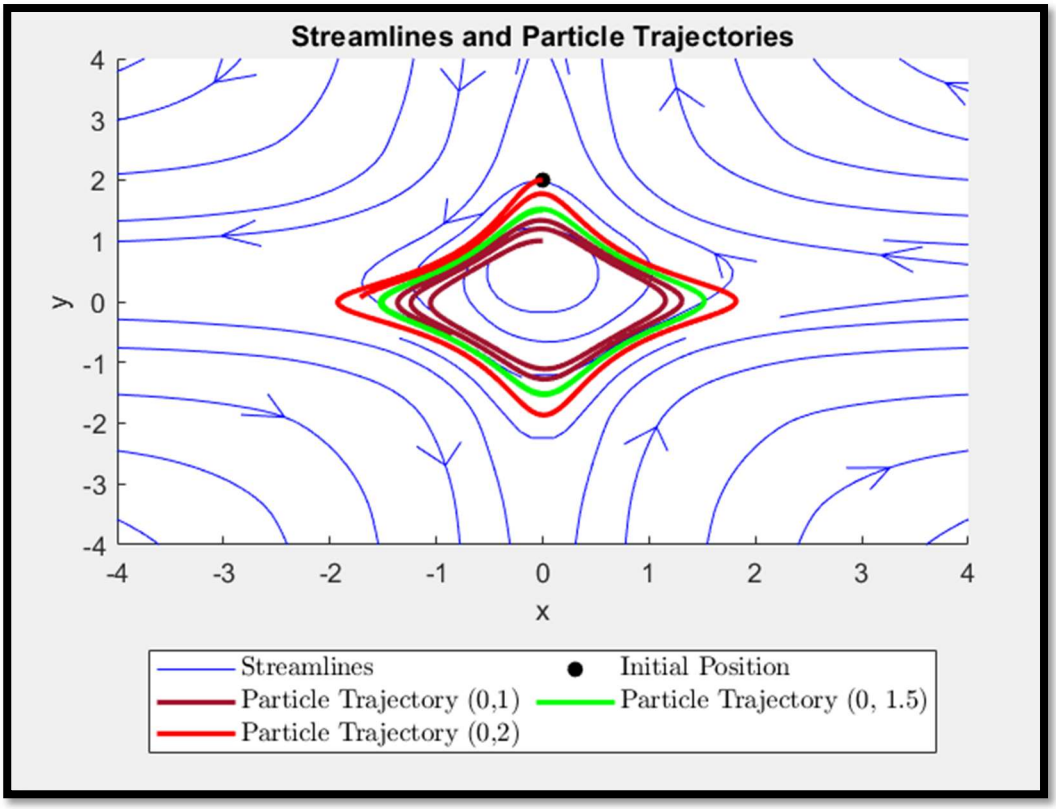


Figure 3: Streamlines and Particle Trajectory of y=1, y=1.5, and y=2

Script for Problem 2e Below: (HW3P2E)

```
% Homework 3 Problem 2 Part E
% Time vector
t = 0:0.1:25;

% Generate meshgrid for streamlines
[x, y] = meshgrid(linspace(-10, 10, 20), linspace(-10, 20, 20));

% Define the velocity field based on your system's equations
u = -2 * y ./ (1 + y.^2).^2;
v = 2 * x ./ (1 + x.^2).^2;

% Plot streamlines with a specified color
figure;
hStreamlines = streamslice(x, y, u, v, 2); %cannot make legends separate colors
unless I use hstreamlines
set(hStreamlines, 'Color', 'b');  % Set the color to blue

hold on;

% Initialize a cell array to store trajectory data for each initial position
trajectoryData = cell(1, 3);

% Loop through different initial y positions
for i = 1:3 % loop will run 3 times to run each iteration aka the three points given

    % Call the euler_advect function for the current initial position
    [xTraj, yTraj] = euler_advect(t, x0, 1 + 0.5 * (i-1)); %inital y value changes
each loop 1, 1.5, 2 respec

    % Plot particle trajectory on top of streamlines
    if i == 3
        % If initial position is (0, 2), plot in red
        plot(xTraj, yTraj, 'r', 'LineWidth', 2);
    elseif i == 2
        % If initial position is (0, 1.5), plot in dark green
        plot(xTraj, yTraj, 'g', 'LineWidth', 2);
    else
        % Otherwise, plot in blue aka (0,1)
        plot(xTraj, yTraj, 'b', 'LineWidth', 2);
    end

    % Save trajectory data in the cell array
    trajectoryData{i} = [xTraj; yTraj];
end

% Scatter initial position
scatter(x0, y0, 'k', 'filled');

% Set axis limits
axis([-4 4 -4 4]); % did not want a really large graph
title('Streamlines and Particle Trajectories'); %title of graph
xlabel('x'); % title of x-axis
ylabel('y'); % title of y-axis
```

```
% Specify legend entries and colors, place the legend outside at the
% bottom, set orientation to horizontal, and set two rows (one row is too
% big. Had to look up how to set this legend up since was not able to
% myself and it kept showing legend before with lines all of same color)

legend([hStreamlines(1), scatter(x0, y0, 'k', 'filled'), ...
        plot(trajectoryData{1}(1, :), trajectoryData{1}(2, :), 'LineWidth', 2), ...
        plot(trajectoryData{2}(1, :), trajectoryData{2}(2, :), 'g', 'LineWidth', 2),
...
        plot(trajectoryData{3}(1, :), trajectoryData{3}(2, :), 'r', 'LineWidth', 2)],
...
    'Streamlines', 'Initial Position', 'Particle Trajectory (0,1)', ...
    'Particle Trajectory (0, 1.5)','Particle Trajectory (0,2)','Interpreter', ...
    'latex', 'FontSize', 10,'Location', 'southoutside', 'Orientation', ...
    'horizontal', 'NumColumns', 2);

hold off;
```

Problem 3 - Transient Ansys: This weeks problem investigates the vortex shedding that occurs before a cylinder. This problem is inherently transient, so we must perform a transient solution. This requires us to tune the following parameters that impact the solver: residuals, time step, and max iterations. Ideally, the simulation will converge at each time step meaning that the residuals satisfy the tolerance set by the user. For the following exercise, record the console transcript so you can evaluate the residuals and convergence. Submit these file(s). Remember to reinitialize between each simulation.

***My transcript would not download on vlab no matter how many times I tried, so I attached a pdf copy on the zip file called CFD HW 3 Problem 3 ANSYS Transcript in PDF.pdf where I just copied and pasted the console window.

Problem 3a) For the case where $\mu = 0.001$ and the inlet velocity is $u = 1$, perform a simulation from $t = 0$ to $t = 20$ sec using a time step of $\Delta t = 0.5$, 0.25, and 0.1. Set the maximum number of iterations to 50. How many iterations were necessary to complete the simulation? Does this indicate that the simulation was able to achieve convergence for the given time step?

Case 1a

$\mu = 0.001$, u=1, number of time steps = 20, $\Delta t = 0.5$, max # of iterations = 50

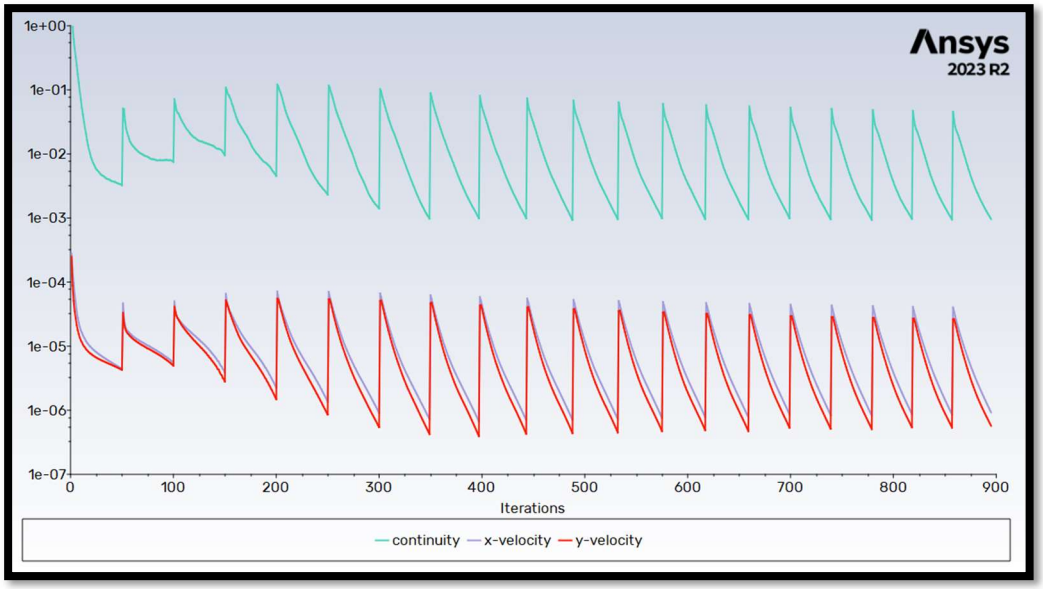


Figure 4: Residual Graph of Set Parameters for Case 1a

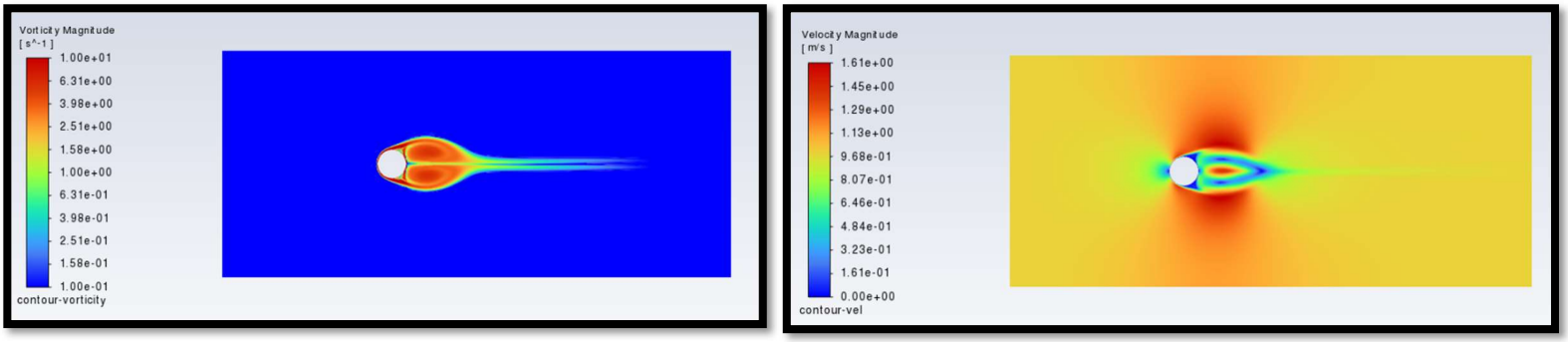


Figure 5: Contours of Vorticity and Velocity of Set Parameters for Case 1a

Case 1b

$\mu = 0.001$, u=1, number of time steps = 40 (I did 20-0/Δt), $\Delta t = 0.5$, max # of iterations = 50

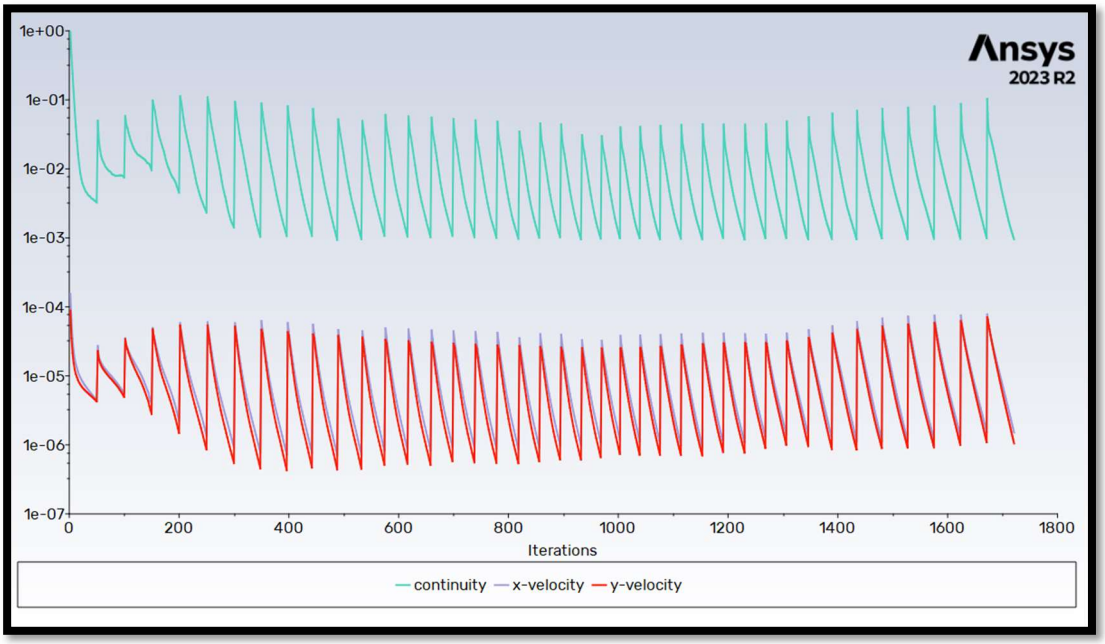


Figure 6: Residual Graph of Set Parameters for Case 1b

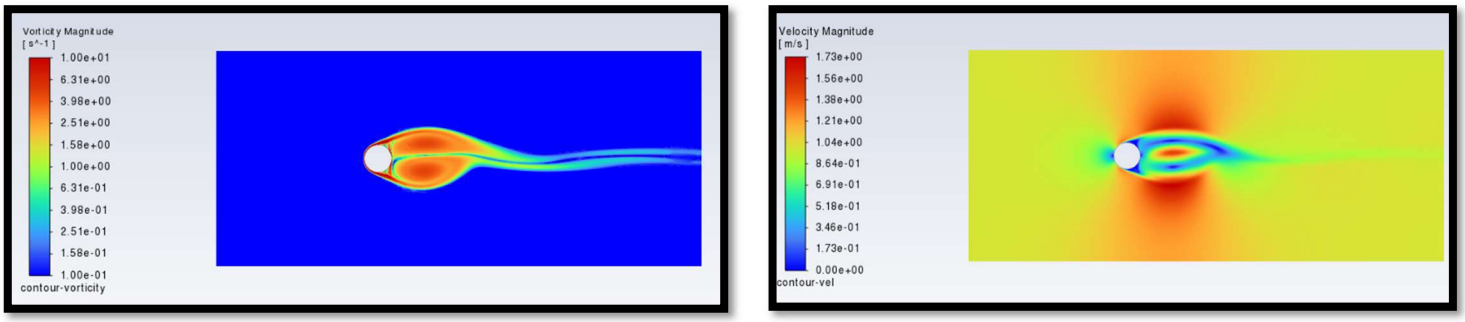


Figure 7: Contours of Vorticity and Velocity of Set Parameters for Case 1b

Case 2a: $\mu = 0.001$, u=1, number of time steps = 20, $\Delta t = 0.25$, max # of iterations = 50

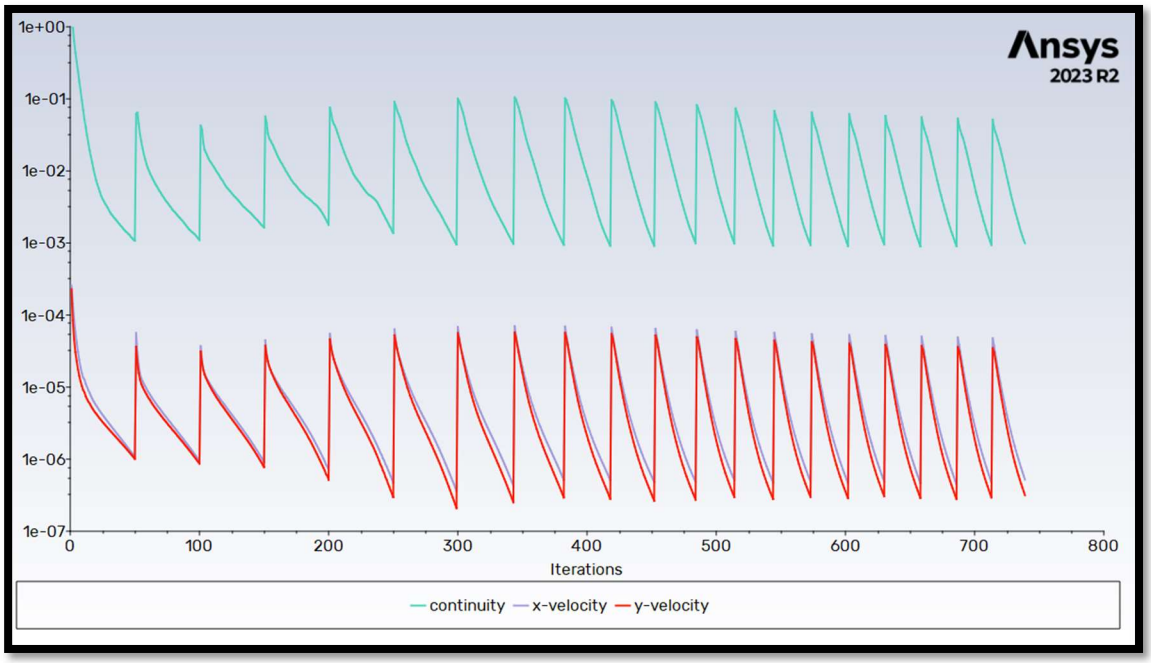


Figure 8: Residual Graph of Set Parameters for Case 2a

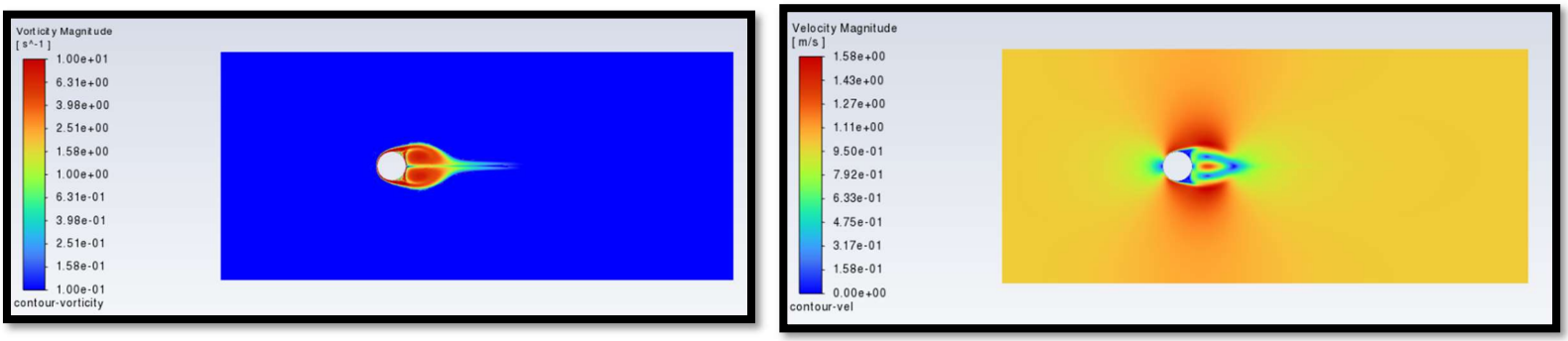


Figure 9: Contours of Vorticity and Velocity of Set Parameters for Case 2a

Case 2b

$\mu = 0.001$, u=1, number of time steps = 80 (I did 20-0/Δt), $\Delta t = 0.25$, max # of iterations = 50

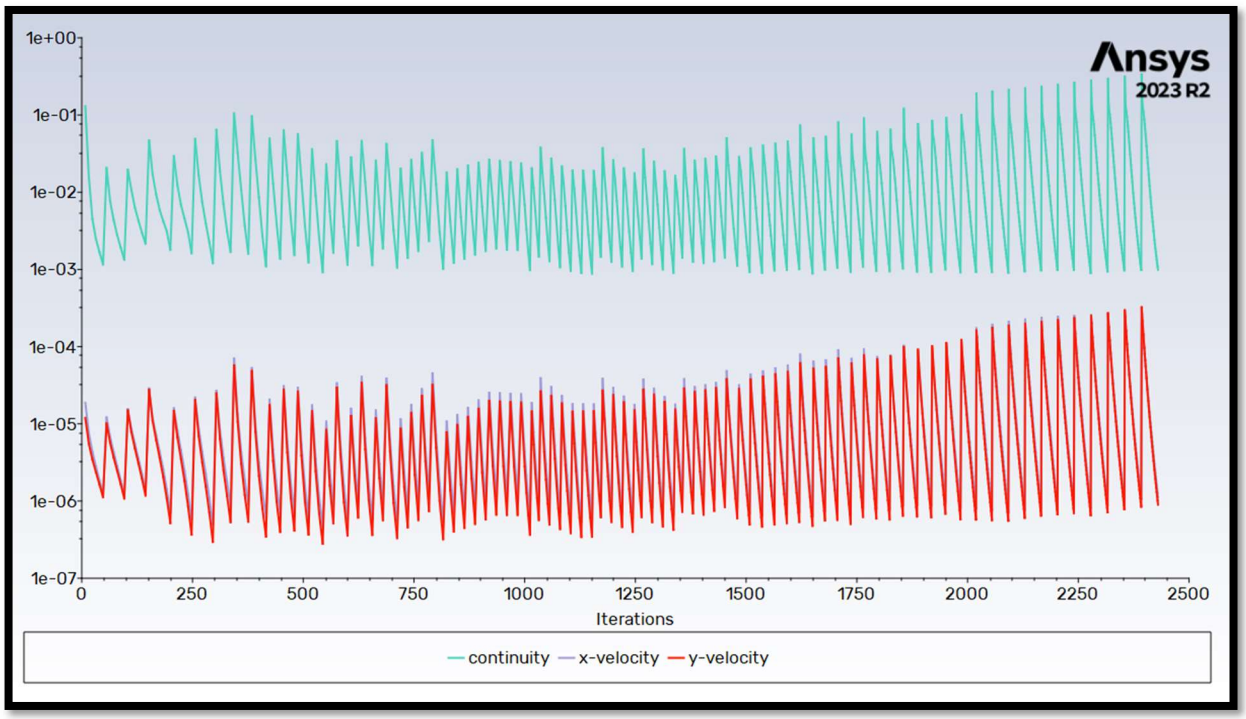


Figure 10: Residual Graph of Set Parameters for Case 2b

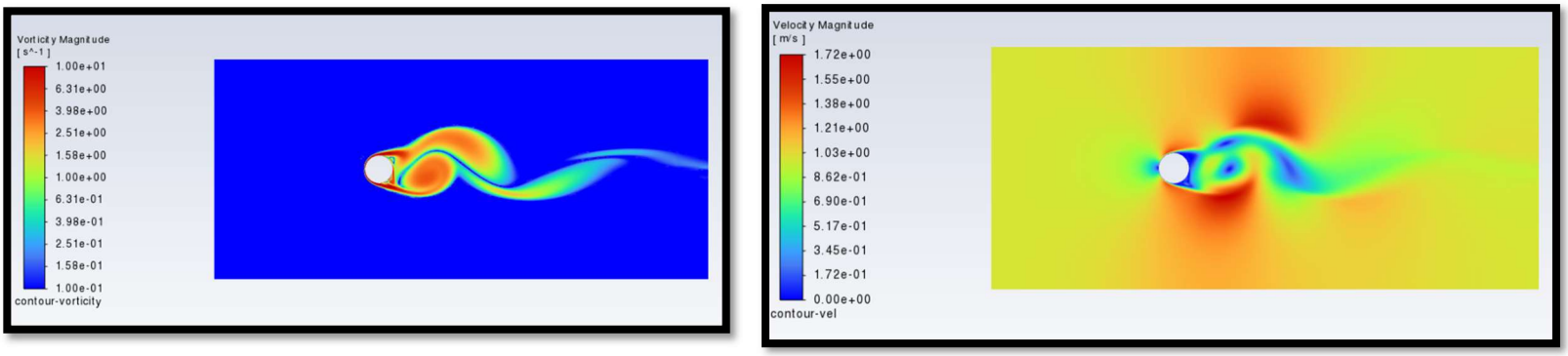


Figure 11: Contours of Vorticity and Velocity of Set Parameters for Case 2b

Case 3a

$\mu = 0.001$, u=1, number of time steps = 20, $\Delta t = 0.1$, max # of iterations = 50

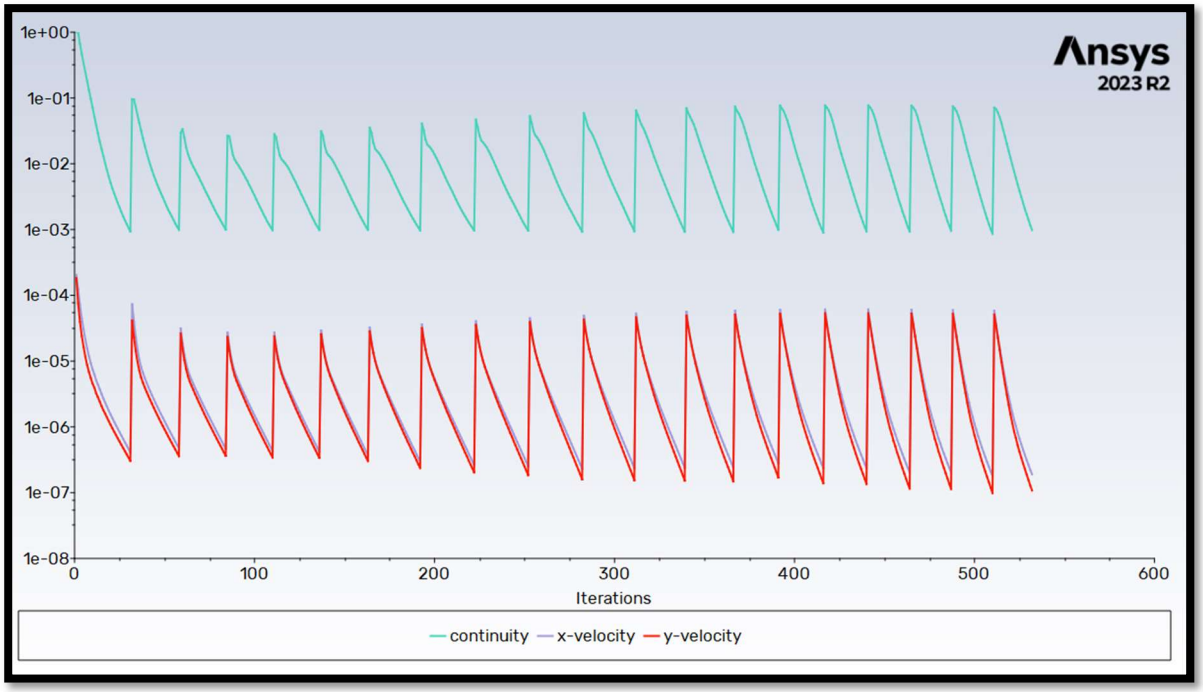


Figure 12: Residual Graph of Set Parameters for Case 3a

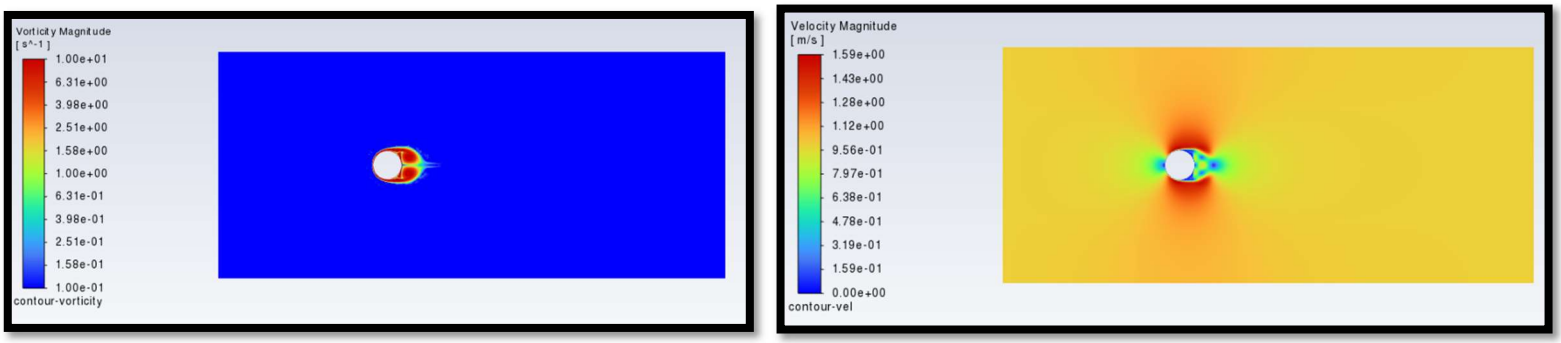


Figure 13: Contours of Vorticity and Velocity of Set Parameters for Case 3a

Case 3b

$\mu = 0.001$, u=1, number of time steps = 80 (I did 20-0/Δt), $\Delta t = 0.25$, max # of iterations = 50

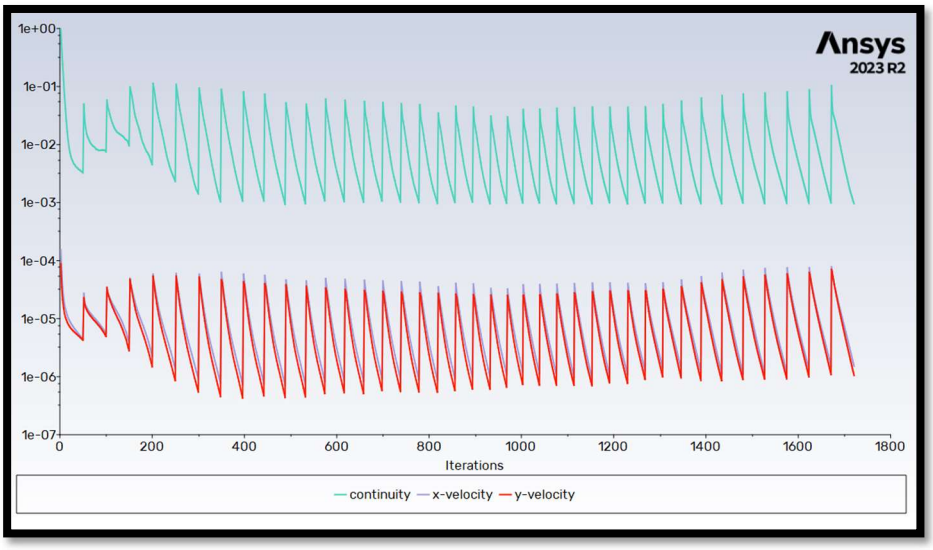


Figure 14: Residual Graph of Set Parameters for Case 3b

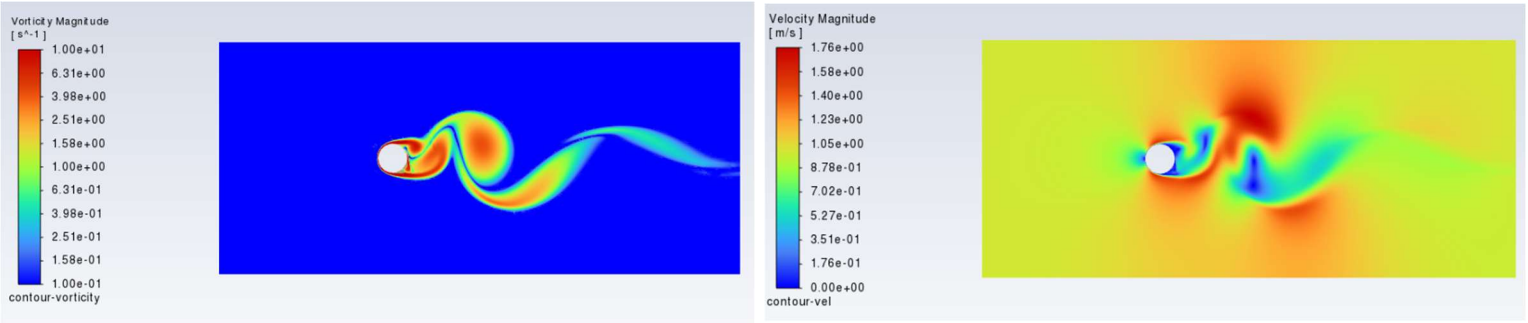


Figure 15: Contours of Vorticity and Velocity of Set Parameters for Case 3b

Comments:

If observing case 1a (number of time step = 20, $\Delta t = 0.5$), case 2a (number of time step = 20, $\Delta t = 0.25$) and case 3a (number of time step = 20, $\Delta t = 0.1$), the residual graphs look different. Based on figure 4 for case 1a it took about 900 iterations, figure 8 for case 2a it took about 750 iterations, and figure 12 for case 3a it took about 525 iterations. The smaller time steps, the less iterations needed to reach a solution and more frequent convergence. The suggested simulation does not seem to converge toward a specific value as it proceeds to move up and down.

If observing case 1b (number of time step = 40, $\Delta t = 0.5$), case 2b (number of time step = 80, $\Delta t = 0.25$) and case 3b (number of time step = 200, $\Delta t = 0.1$), the residual graphs look different. Based on figure 6 for case 1b it took about 1,700 iterations, figure 10 for case 2b it took about 2,400 iterations, and figure 14 for case 3b it took about 1,700 iterations.

3b) Increase the maximum number of iterations to 100 for the time step 0.5. Do you regularly see convergence now? Comment about what steps you would take to efficiently solve the simulation while getting a converged result.

Case 1a for Problem 3b)

$\mu = 0.001$, u=1, number of time steps = 20, $\Delta t = 0.5$, max # of iterations = 50

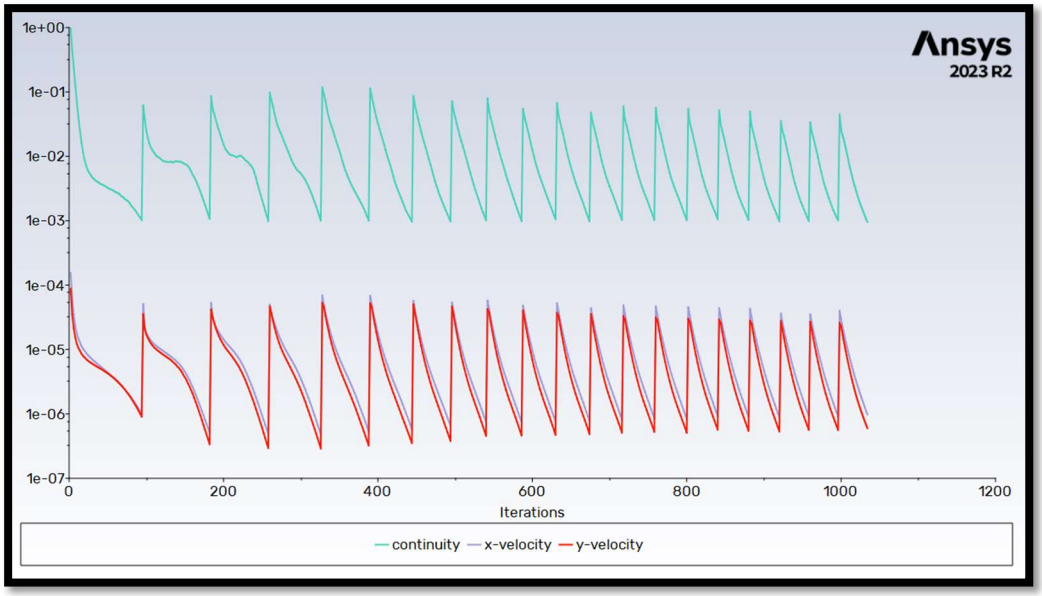


Figure 16: Residual Graph of Set Parameters for Problem 3b Case 1a

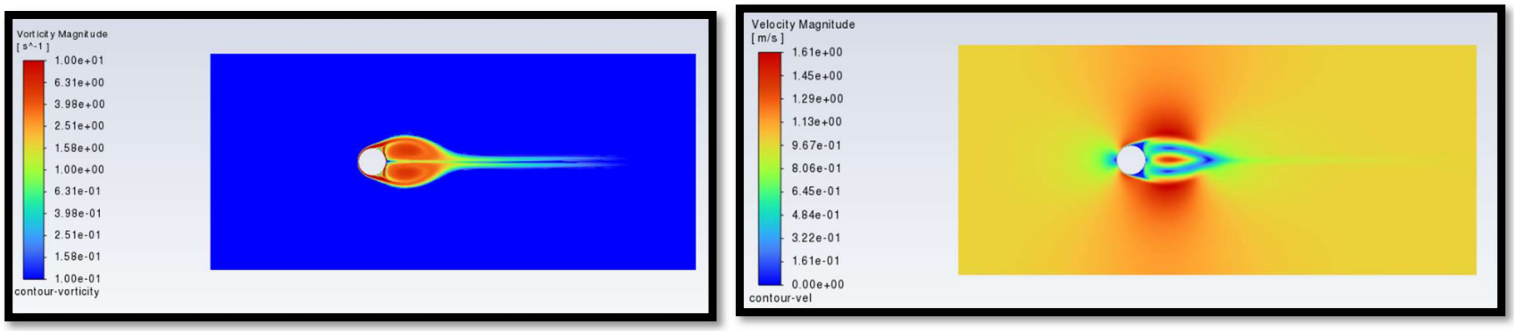


Figure 17: Contours of Vorticity and Velocity of Set Parameters for Problem 3b Case 1a

$\mu = 0.001$, u=1, number of time steps = 40, $\Delta t = 0.5$, max # of iterations = 100

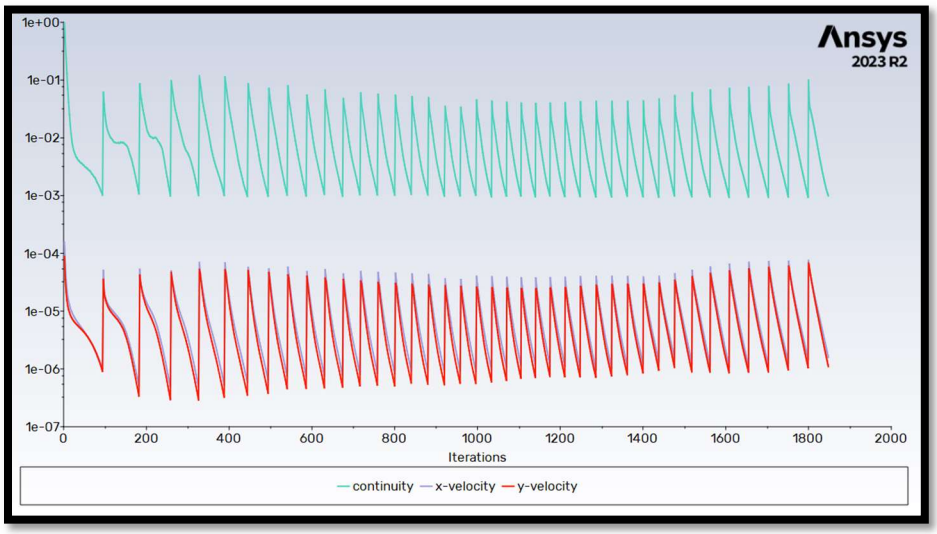


Figure 18: Residual Graph of Set Parameters for Problem 3b Case 1b

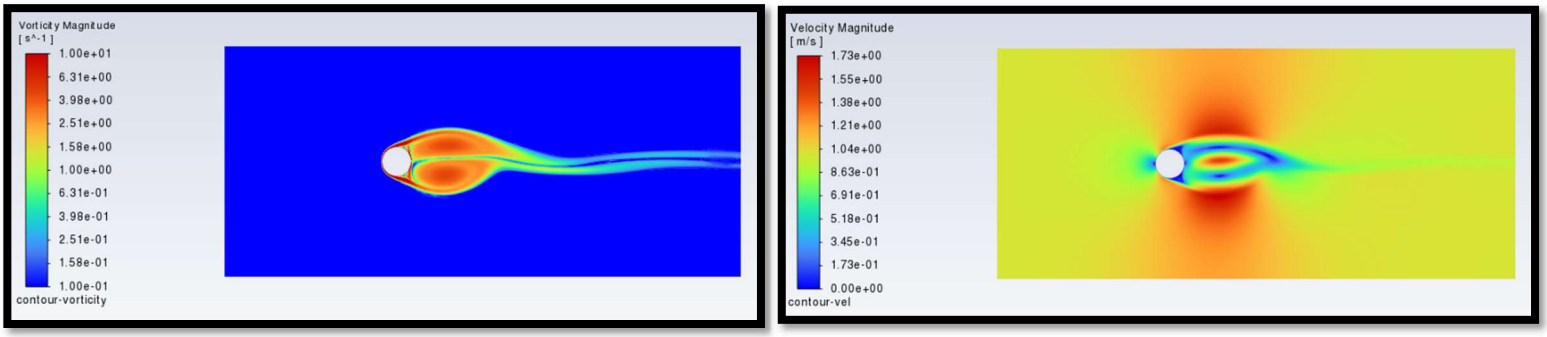


Figure 19: Contours of Vorticity and Velocity of Set Parameters for Problem 3b Case 1b

Comments

Convergence is still not obvious for the plots in problem 3b. It does seem to take less iterations to solve the problem. Although, time step is still quite large than 0.25 and 0.1, we were given more iterations to work with. If the simulation converges with increased limit, it suggests that allowing more iterations can help achieve convergence. There are many things we could do to improve the convergence of this simulation. We could try adjusting the time step size. Smaller time steps improve convergence but takes a long time to compute. I could also provide a better initial condition and initial guess for a solution; this can help simulation converge quickly. Meshes can also be better as poorly structure or high skewed meshes can lead to convergence issues. This may be due to the problem of being transient and not turbulent.

Problem 3c) For the previous simulation, we used a fixed time step. Now we will look at the impact of picking a variable time step. For this calculation, perform only 50 time steps. Using the "Error-based" approach, set the "Error Tolerance" to 0.01, 0.005, and 0.001. What is the final time reached after 50 time steps? You can find this information in the read out in the console.

Case 1)

Number of Time Steps = 50, Error Tolerance = 0.01, Initial Time Step Size = 0.5, Max Iterations = 100

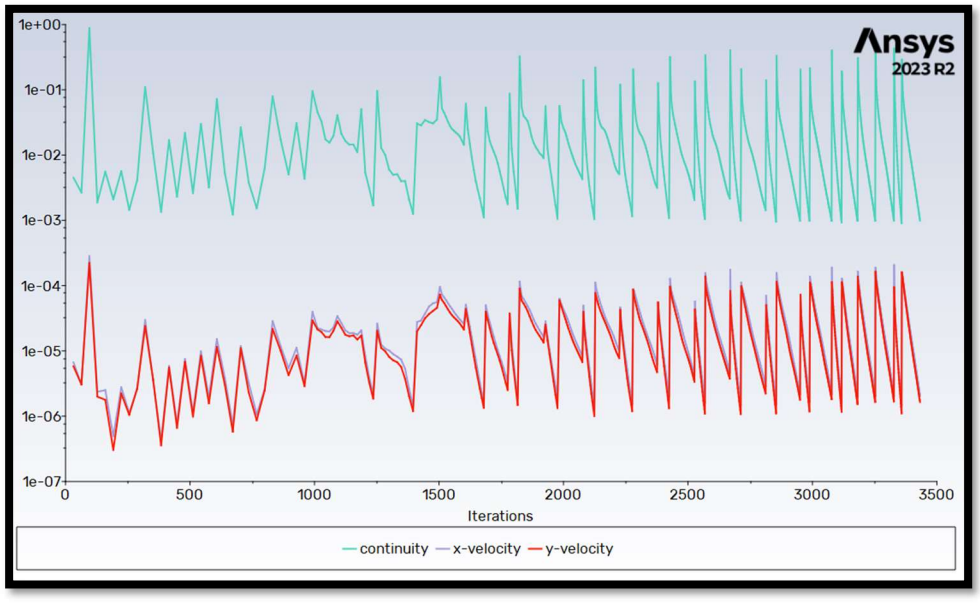
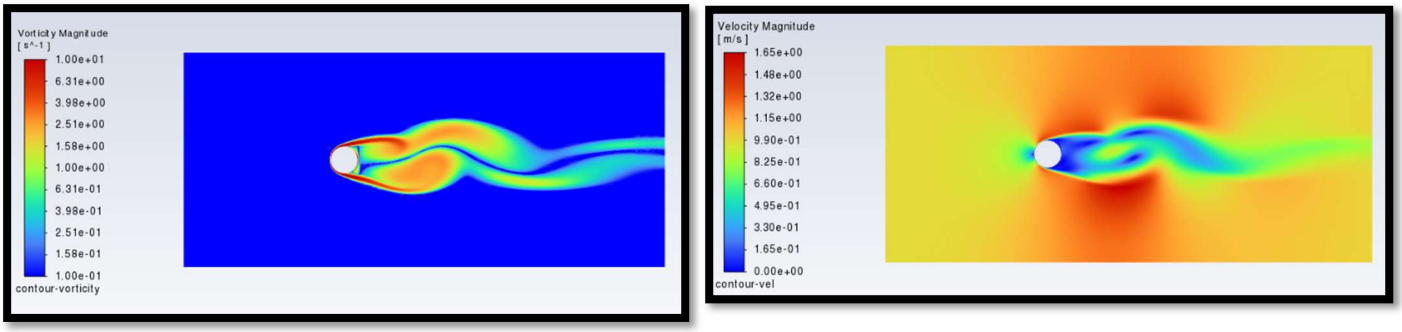


Figure 20: Residual Graph of Set Parameters for Problem 3 Part C Case 1




```
  3431  1.0527e-03  2.1738e-06  1.7480e-06  0:00:03   27
  3432  9.9003e-04  2.0412e-06  1.6446e-06  0:00:02   26
! 3432 solution is converged
(update-animation-object "animation-vorticity")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vorticity.cxa
()
(update-animation-object "animation-vel")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vel.cxa
()
Flow time = 33.64479827880859s, time step = 40
Writing "| gzip -2cf > SolutionMonitor.gz"...
Writing temporary file C:\Users\azgars\AppData\Local\Temp\fltgz-22725 ...
Done.

'\\winfiles.wincoe.coe.neu.edu\cifs.homedir\Win10Files\Desktop\Flow Around a Cylinder\cylinder_flow_hw_files\dp0\FFF\Fluent'
CMD.EXE was started with the above path as the current directory.
```

Figure 21: Console Window & Contours for Problem 3c Case 1

Case 2)

Number of Time Steps = 50, Error Tolerance = 0.005, Initial Time Step Size = 0.5, Max Iterations = 100

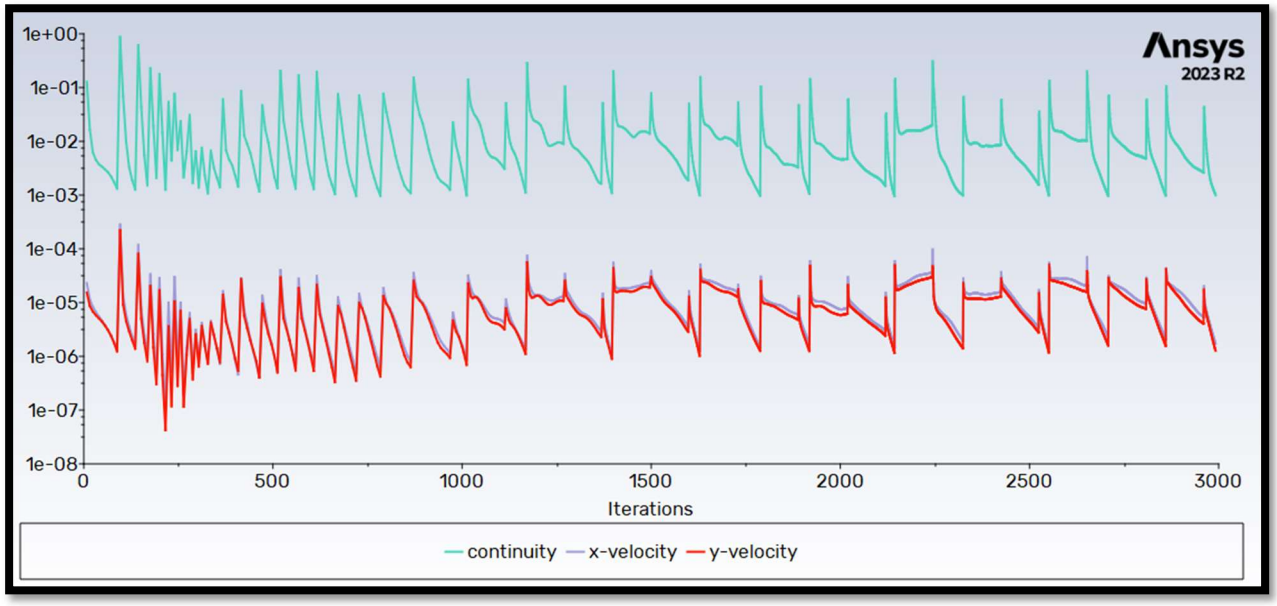
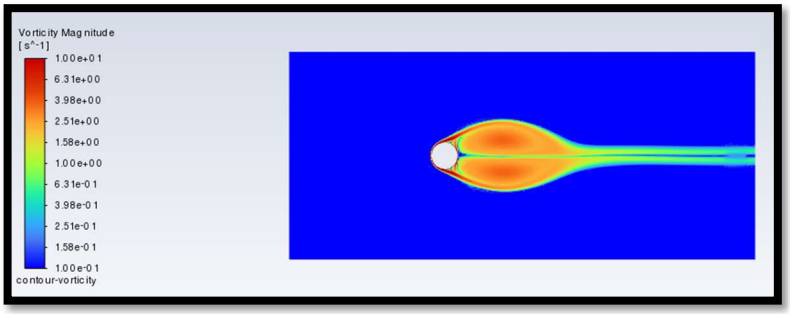
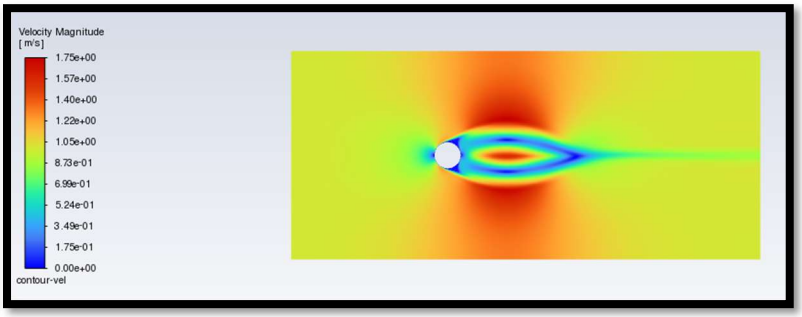


Figure 22: Residual Graph of Set Parameters for Problem 3 Part C Case 2






```
  2992  9.8416e-04  1.7104e-06  1.2692e-06  0:00:02   68
! 2992 solution is converged
(update-animation-object "animation-vorticity")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vorticity.cxa
()
(update-animation-object "animation-vel")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vel.cxa
()
Flow time = 39.58744812011719s, time step = 43
Writing "| gzip -2cf > SolutionMonitor.gz"...
Writing temporary file C:\Users\azgars\AppData\Local\Temp\flntgz-22726 ...
Done.
```

Figure 23: Console Window & Contours for Problem 3c Case 2

Case 3)

Number of Time Steps = 50, Error Tolerance = 0.001, Initial Time Step Size = 0.5, Max Iterations = 100

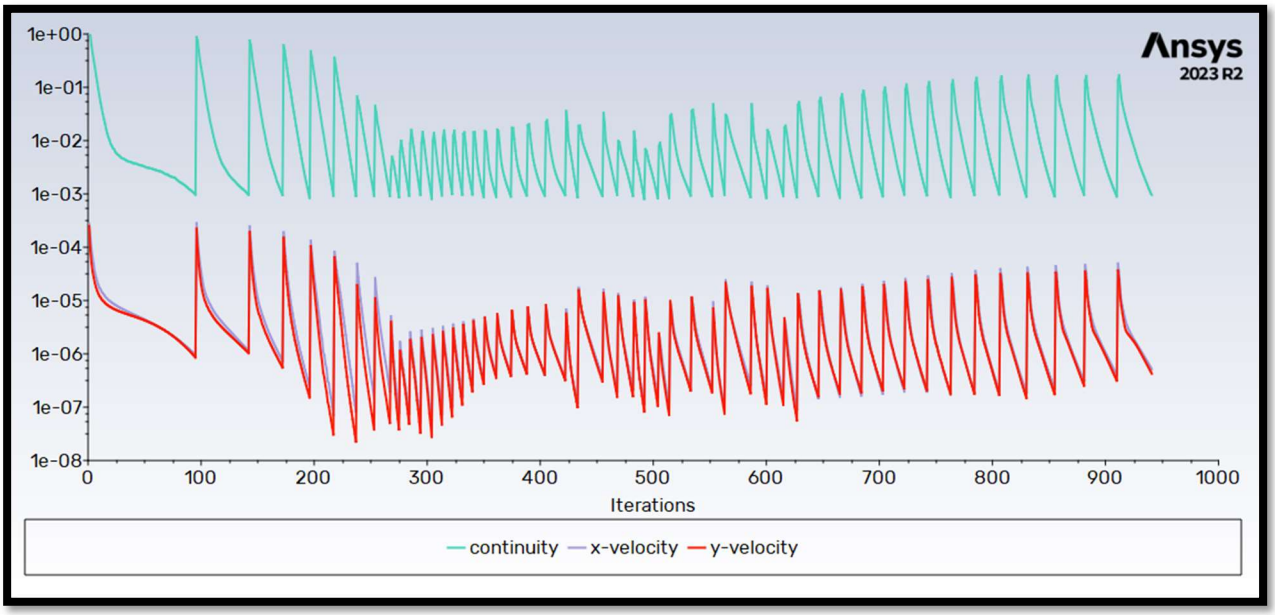
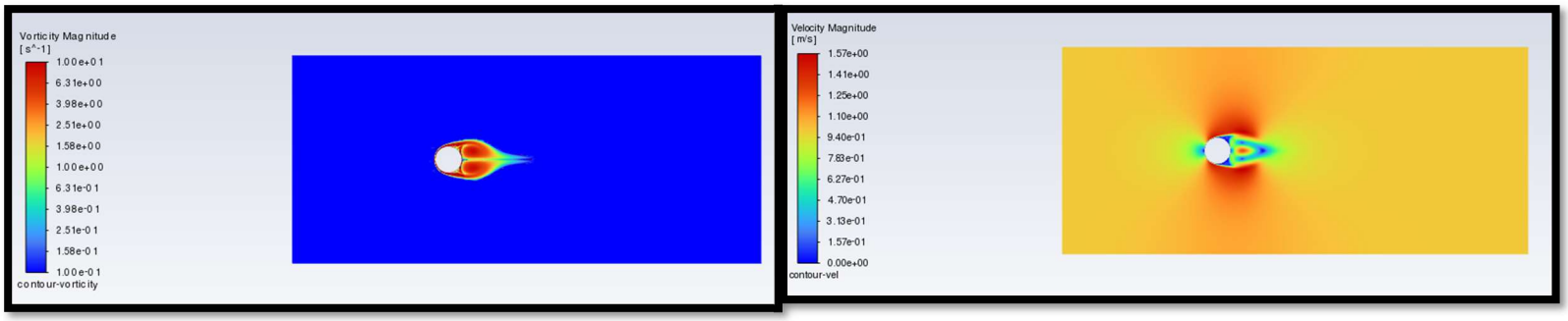


Figure 24: Residual Graph of Set Parameters for Problem 3c Case 3




```
   941  9.5394e-04  5.1468e-07  4.2401e-07  0:00:01   69
!  941 solution is converged
(update-animation-object "animation-vorticity")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vorticity.cxa
()
(update-animation-object "animation-vel")
Creating animation sequence file: //winfiles.wincoe.coe.neu.edu/cifs.homedir/Win10Files/Desktop/Flow Around a Cylinder/cylinder_flow_hw_files/dp0/FFF/
Fluent/.//animation-vel.cxa
()
Flow time = 4.047153472900391s, time step = 39
Writing "| gzip -2cf > SolutionMonitor.gz"...
Writing temporary file C:\Users\azgars\AppData\Local\Temp\fltgz-22727 ...
Done.
```

Figure 25: Console Window & Contours for Problem 3c Case 3

When there is an error tolerance of 0.01, the flow time was 33.644 s with time step being 40.
When there is an error tolerance of 0.005, the flow time was 39.587 with time step being 43.
When there is an error tolerance of 0.001, the flow time was 4.047 s with time step being 39.