Lab 6 - Round Jet PIV Measurement

Parham Khodadi*
A E 303, Section 3, with Dr. Xiaofeng Liu

This experiment aimed to investigate the flow characteristics of a turbulent round jet using Particle Image Velocimetry (PIV). The EduPIV system was used to capture high-resolution velocity fields by tracking tracer particles in a water tank, illuminated by an LED light sheet and recorded by a camera connected to DynamicStudio. The collected data were processed in MATLAB to compute time-averaged velocities, Reynolds stresses, and vorticity. Results revealed a well-defined shear layer and the onset of turbulence downstream of the nozzle. Challenges included particle seeding uniformity and lighting consistency. Overall, the experiment successfully demonstrated key PIV principles and provided insight into turbulent jet behavior.

I. Nomenclature

$\bar{u}, \bar{v}$	=	Time-averaged streamwise and cross-stream velocities (m/s)
u', v'	=	Velocity fluctuations in x and y directions (m/s)
u'^2, v'^2	=	Reynolds normal stresses (m^2/s^2)
$u'v'$	=	Reynolds shear stress (m^2/s^2)
ω_z	=	Spanwise vorticity (1/s)
X, Y	=	Spatial coordinates in the measurement plane (mm)
D	=	Nozzle diameter (mm)
$x/D, y/D$	=	Non-dimensional streamwise and cross-stream coordinates (–)

II. Introduction

This lab aimed to investigate the turbulent structure of a round jet using Particle Image Velocimetry (PIV). PIV enables non-intrusive measurement of fluid velocity fields by tracking the motion of seeded particles illuminated by a light sheet. The experiment focused on visualizing and quantifying jet development, shear layer growth, and turbulence characteristics. Using the EduPIV system, flow images were captured, analyzed in DynamicStudio, and further processed in MATLAB to obtain time-averaged velocities, Reynolds stresses, and vorticity.

III. Theory

The goal of this experiment was to study the near-field behavior of a free, round turbulent jet using Particle Image Velocimetry (PIV), a modern optical measurement technique that has transformed experimental fluid dynamics [1]. Compared to older pointwise methods like Pitot tubes or Laser Doppler Velocimetry (LDV), PIV enables non-intrusive, planar velocity measurements by tracking seeding particles illuminated by a laser sheet across consecutive images [2, 3]. With a sufficiently high acquisition rate and particle density, PIV captures detailed instantaneous velocity fields in space and time.

In this lab, the EduPIV system was used to investigate the velocity structure and turbulence characteristics of a round jet in water. Velocity vectors were extracted using Dantec DynamicStudio and exported as a series of 300 CSV files—each capturing a time step. These files were post-processed in MATLAB to produce time-averaged velocity fields, Reynolds stresses, and vorticity maps [4].

To analyze turbulence, Reynolds decomposition was applied to the velocity fields. The instantaneous streamwise and cross-stream velocities $u(t)$ and $v(t)$ were decomposed into time-averaged components and fluctuations as follows:

*Aerospace Engineering student, San Diego State University

$$u(t) = \bar{u} + u'(t) \tag{1}$$
$$v(t) = \bar{v} + v'(t) \tag{2}$$

The fluctuations u' and v' represent the unsteady portion of the flow. From these, Reynolds stresses were computed to quantify the turbulence intensity and shear:

$$\overline{u'^2} \text{ : Reynolds normal stress in the } x \text{ direction} \tag{3}$$
$$\overline{v'^2} \text{ : Reynolds normal stress in the } y \text{ direction} \tag{4}$$
$$\overline{u'v'} \text{ : Reynolds shear stress} \tag{5}$$

These stresses represent the turbulent transport of momentum and appear as additional terms in the Reynolds-Averaged Navier-Stokes (RANS) equations [5].

To characterize rotational structures and vortex formation, 2D spanwise vorticity ω_z was calculated using the velocity gradients:

$$\omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{6}$$

This helped identify regions of shear and rotational motion, particularly in the shear layer surrounding the jet core.

According to previous studies [6, 7], round jets exhibit a potential core near the nozzle and develop shear-layer instabilities that grow into turbulence. These instabilities can be axisymmetric or helical, and their evolution is governed by the Reynolds number and boundary conditions.

In this lab, MATLAB was used to automate the processing of raw data, including averaging, interpolating onto a uniform grid, and calculating stresses and vorticity using gradient-based methods. The uniform grid enabled consistent contour and profile plotting, making it easier to identify jet structure and compare locations nondimensionalized by nozzle diameter D.

Sample Calculation Reference

A step-by-step calculation for time-averaged velocity is included in Appendix B. This uses one vector point from a raw CSV file and applies the same logic used in the MATLAB processing script to compute $\bar{u}$ and $\bar{v}$. The values used are directly from `EduPIV_lab.62tbxosb.000000.csv`, and the method aligns with the code shown in `ImportData_EduPIV.m` [4].

IV. Experimental Setup

The experiment was conducted using the EduPIV system by Dantec Dynamics to analyze a free turbulent round jet in a controlled environment. A schematic of the setup is shown in Figure 1, while real images of the physical layout are provided in Figures 2 and 3.

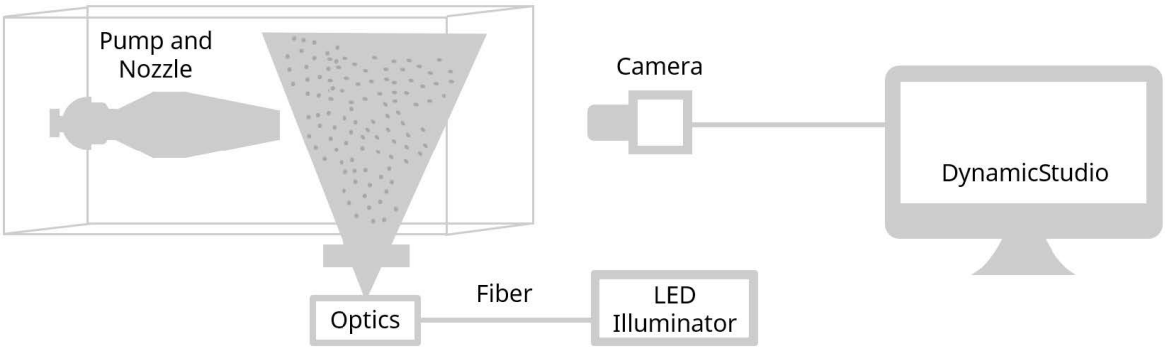


Fig. 1 Schematic of the EduPIV round jet experiment setup.

The system consists of a transparent water tank, a 3D printed round nozzle (exit diameter 5 cm), and a controlled pump for steady flow generation. Seeding particles were added to the water to enable velocity field measurements via PIV. Illumination was provided by a Dantec LED light source, which projected a 4 mm thick light sheet through fiber optics and optics modules. This light sheet illuminated the seeding particles within the measurement plane.

A Dantec FlowSense USB camera (2M-165) captured sequential images of the particle movement. The camera featured a 35 mm low-distortion lens and was positioned perpendicularly to the flow direction. All image acquisition and PIV analysis were performed using *DynamicStudio* software.

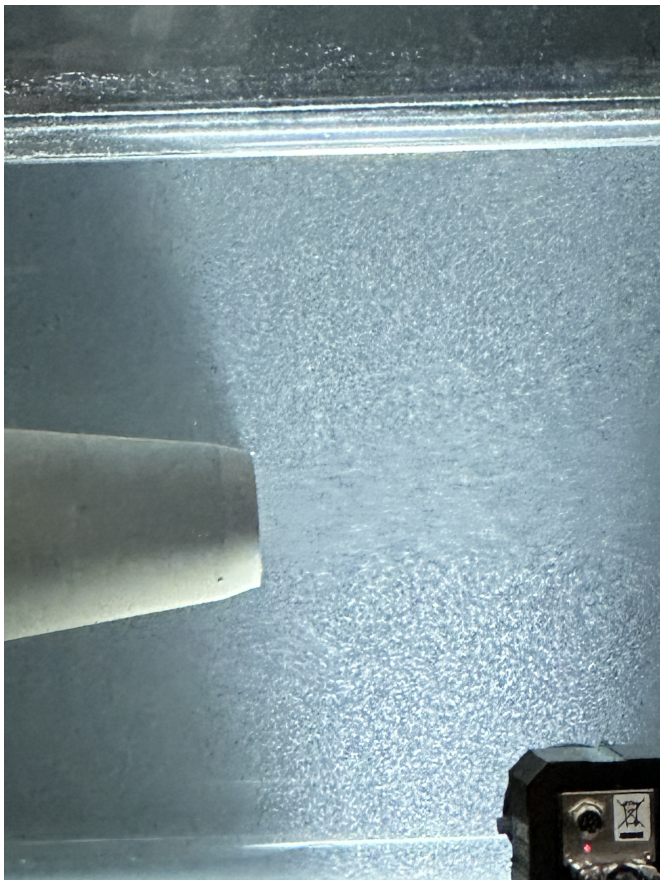


Fig. 2 Close-up of the nozzle and flow field during operation.

Fig. 3 Full experimental setup showing the water tank, camera, optics, and pump system.

Materials and Equipment

The components used in the experiment are summarized in Table 1. All parts are manufactured or supplied by Dantec Dynamics unless otherwise noted.

Table 1 Equipment and Materials Used

Component	Description
DynamicStudio Software	Used for image acquisition and PIV vector field computation
FlowSense USB 2M-165 Camera	1920x1200 resolution, 160 fps, USB 3.0
35mm Low-Distortion Lens	Adjustable aperture, C-mount
EduPIV LED Light Source	150W LED with fiber optic light sheet generation
Optics Module	7.6 cm wide, 35° divergence angle, 4 mm sheet thickness
Seeding Particles	Polyamide particles, 50 µm diameter, 1.03 g/cm^3
3D Printed Round Nozzle	5 cm diameter, submerged in tank
Flow Loop Tank	80 × 35 × 40 cm, 112 L capacity
EcoDrift 8.1 Pump	1600–8000 L/hr, 8–20 W, speed-controlled
Black Backdrop	Used to improve contrast and reduce reflection

V. Experimental Procedure

This experiment followed a structured procedure to ensure repeatability and data quality across all phases: preparation, calibration, and execution. The procedure was carried out using the EduPIV system in conjunction with DynamicStudio software.

Preparation Phase

1) The tank was filled with tap water and the nozzle was mounted horizontally inside the tank using a magnetic pump attachment.
2) The fiber optic cable was connected to the LED light source to form a planar light sheet.
3) Seeding particles were suspended in a test tube with water and rubbing alcohol to break the surface tension and were mixed into the tank.
4) The camera was mounted approximately 19 cm from the tank and connected to the computer via USB.
5) DynamicStudio was launched. The light source was set to full power and room lights were turned off.
6) The optics were adjusted to produce a 4 mm thick light sheet, and a black backdrop was placed at the tank's rear wall to minimize reflections.
7) The camera was oriented perpendicular to the flow and aligned parallel to window orientation to minimize glare.

Calibration Phase

1) A calibration target was placed in the plane of the light sheet and viewed through the camera.
2) Camera settings were adjusted (750 µs exposure, reduced frame rate) to clearly capture the target.
3) 20 calibration images were acquired and saved in DynamicStudio.
4) These calibration images were averaged to determine the pixel-to-mm scale factor.

Execution Phase

1) The room lights were turned off again and the pump was set to a low speed for 5–10 minutes to allow flow stabilization.
2) The camera was reset to a 150 Hz trigger rate and 750 µs exposure time.
3) At least 300 images were recorded to ensure temporal averaging, leveraging the assumption of ergodicity for turbulent flow.
4) The captured image pairs were processed using Adaptive PIV analysis in DynamicStudio with:
    - Minimum window size: 32×32 pixels
    - Maximum window size: 64×64 pixels
    - 50% overlap (16-pixel step)
5) The resulting velocity vector fields were exported as .csv files and imported into MATLAB for time-averaging, Reynolds stress computation, vorticity analysis, and visualization.

VI. Results and Data Reduction

This section presents time-averaged and statistical flow quantities derived from planar PIV measurements of a round turbulent jet. Results include velocity contours, mean velocity profiles, Reynolds stress distributions, and vorticity characteristics.

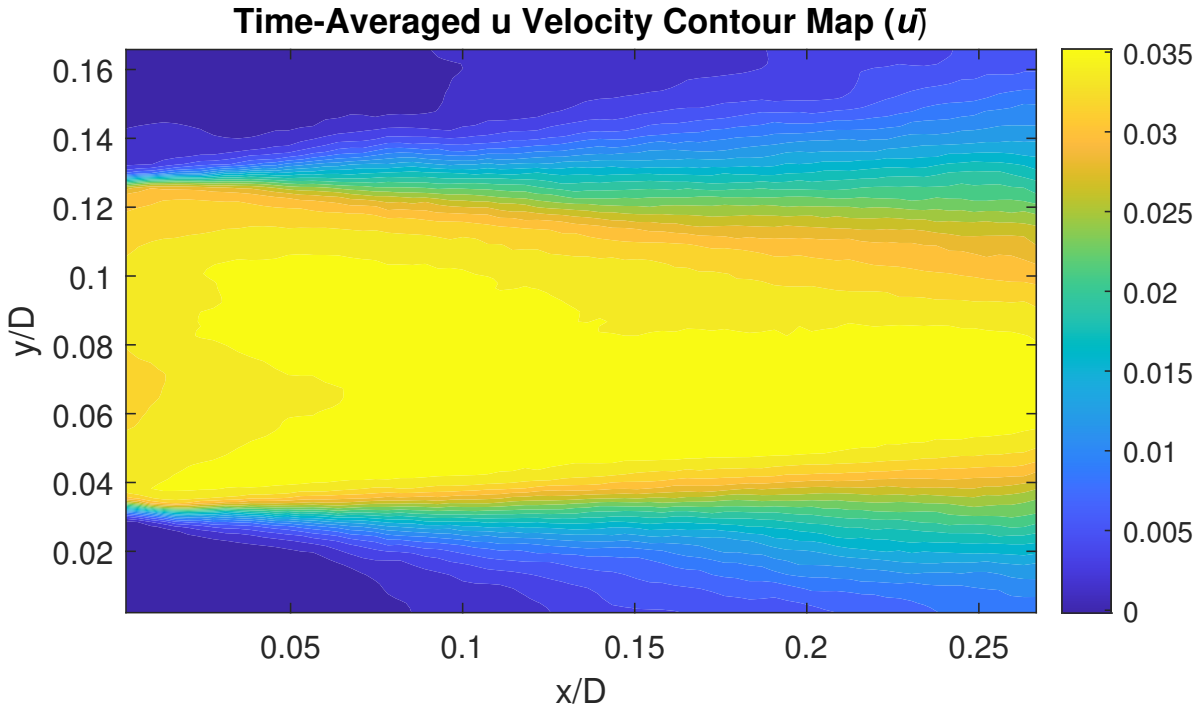


Fig. 4 Time-averaged streamwise velocity contour map, $\bar{u}$, showing the development of the jet.

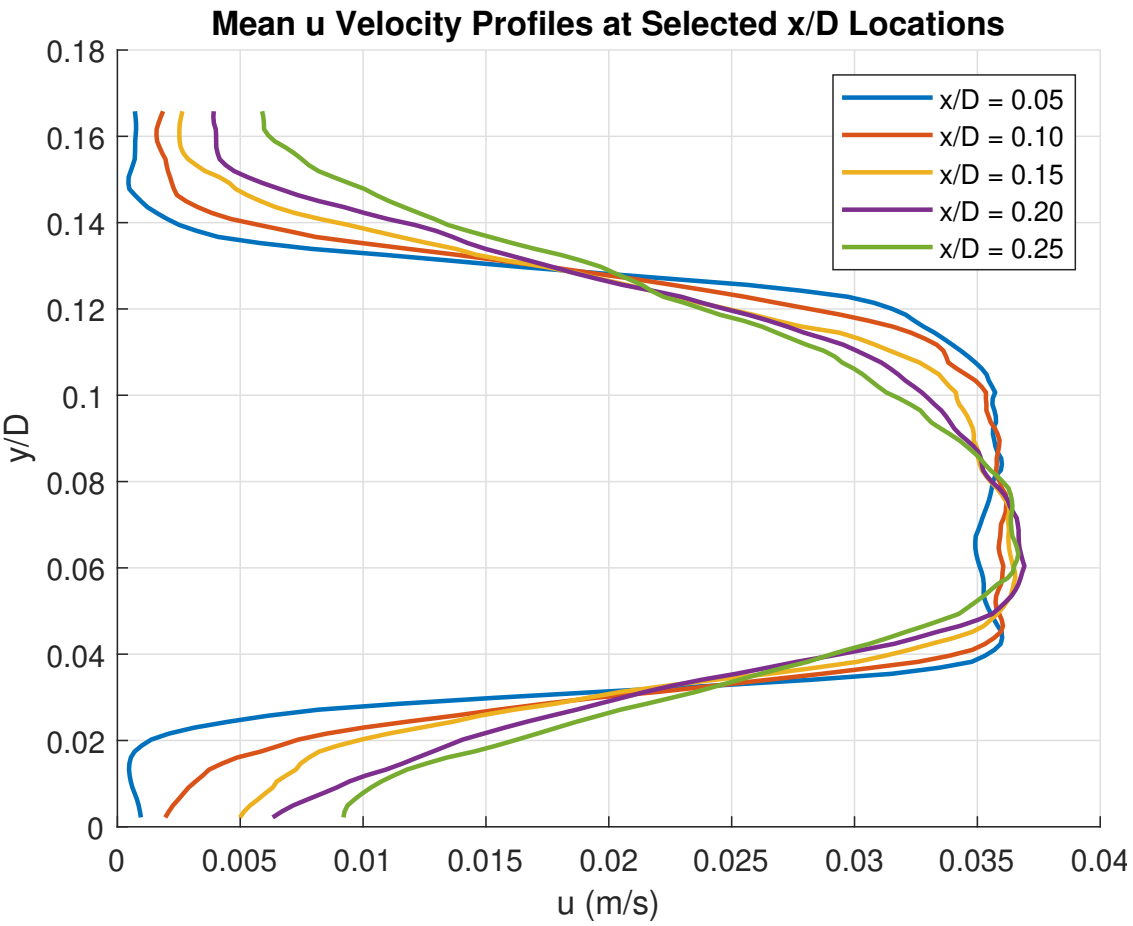


Fig. 5 Streamwise velocity profiles ($\bar{u}$) at different x/D locations.

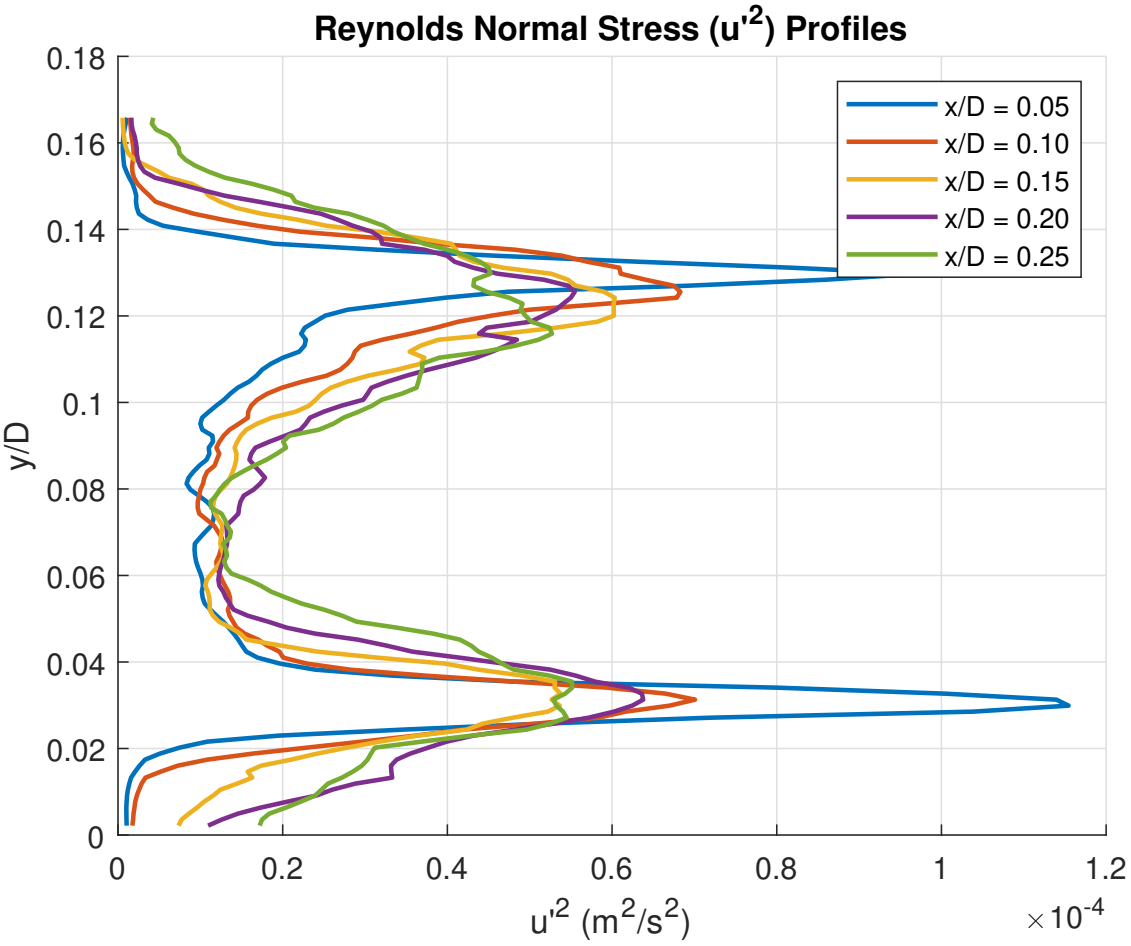


Fig. 6 Reynolds normal stress profiles in the x direction, $\overline{u'^2}$, at multiple x/D stations.

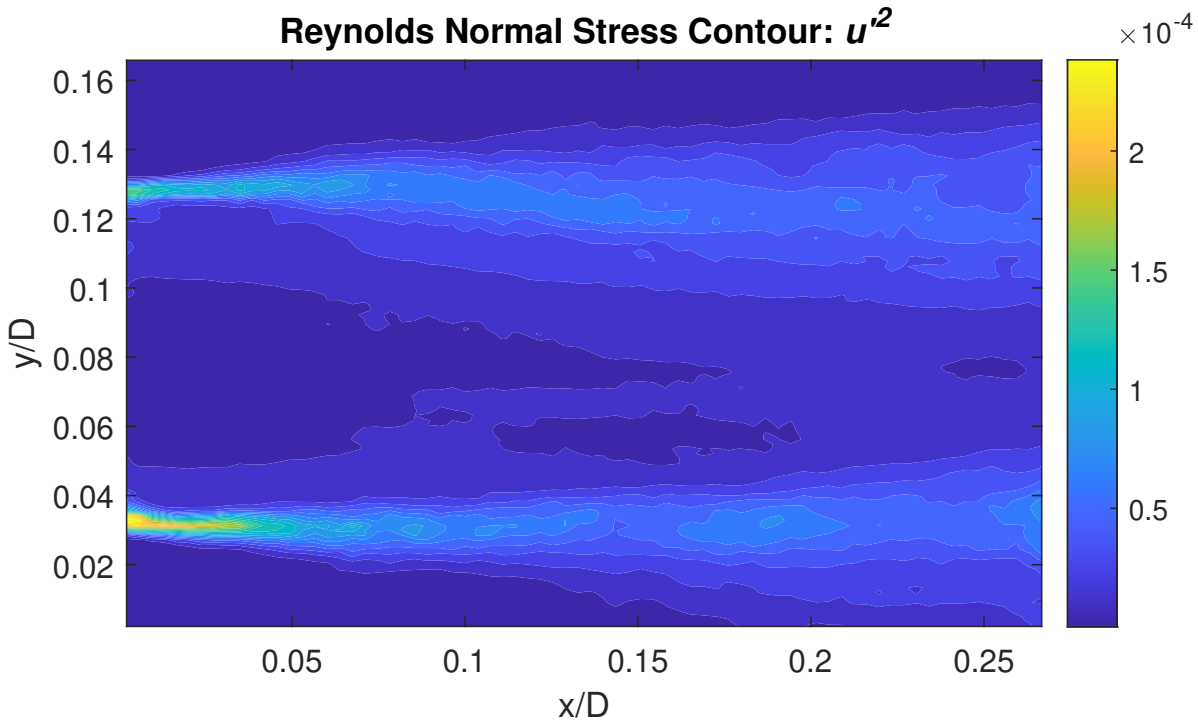


Fig. 7 Contour map of $\overline{u'^2}$ revealing the extent and peak intensity of streamwise turbulence.

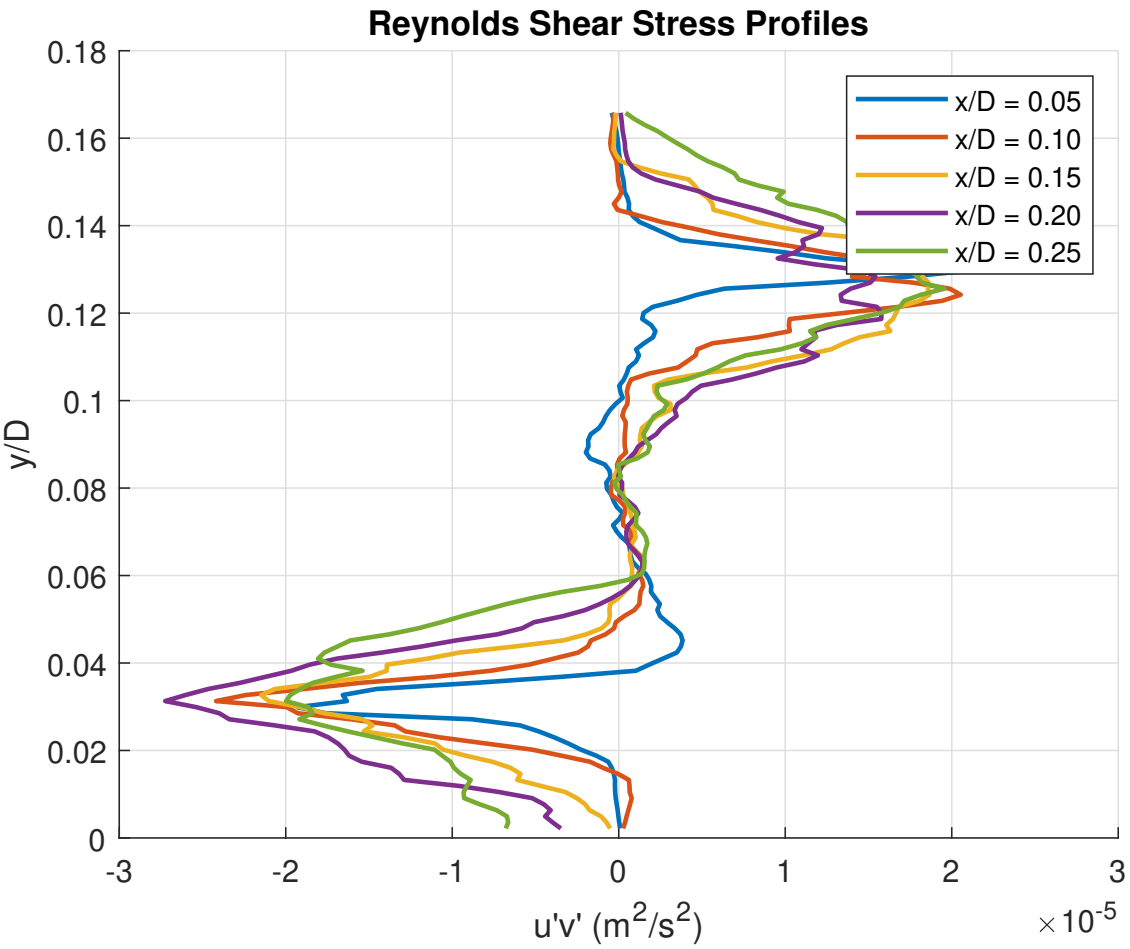


Fig. 8 Reynolds shear stress profiles ($\overline{u'v'}$) at selected streamwise locations.

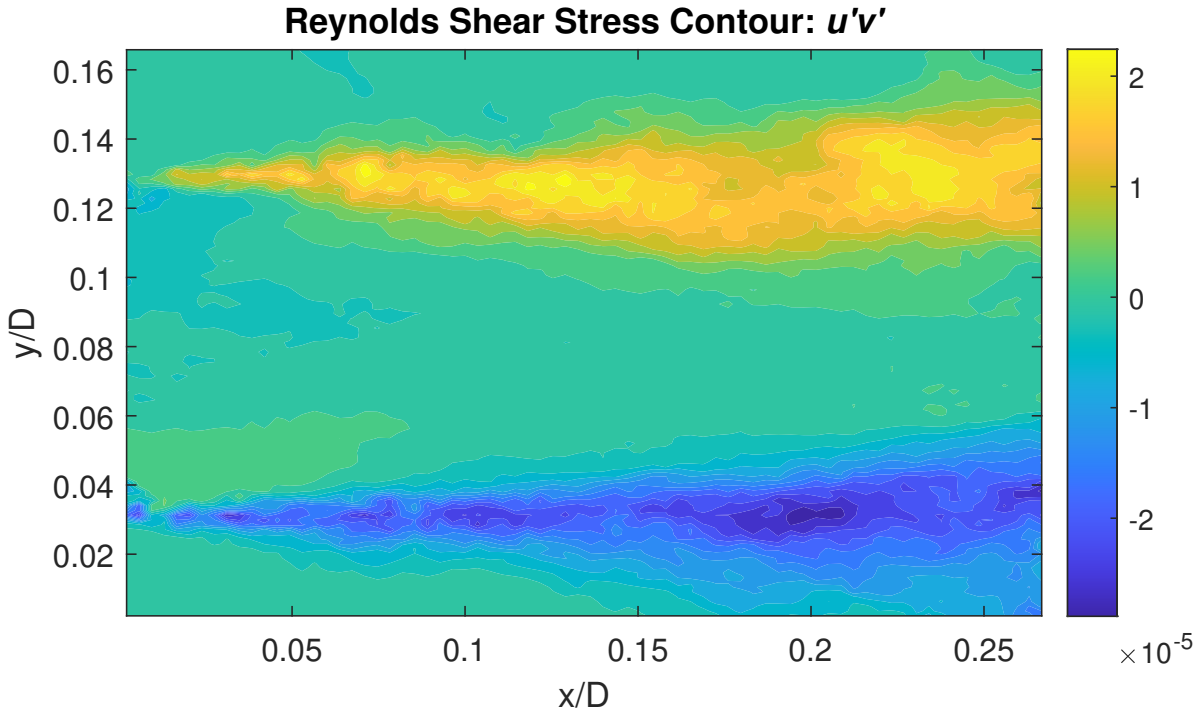


Fig. 9 Reynolds shear stress contour map, $\overline{u'v'}$, indicating shear layer development.

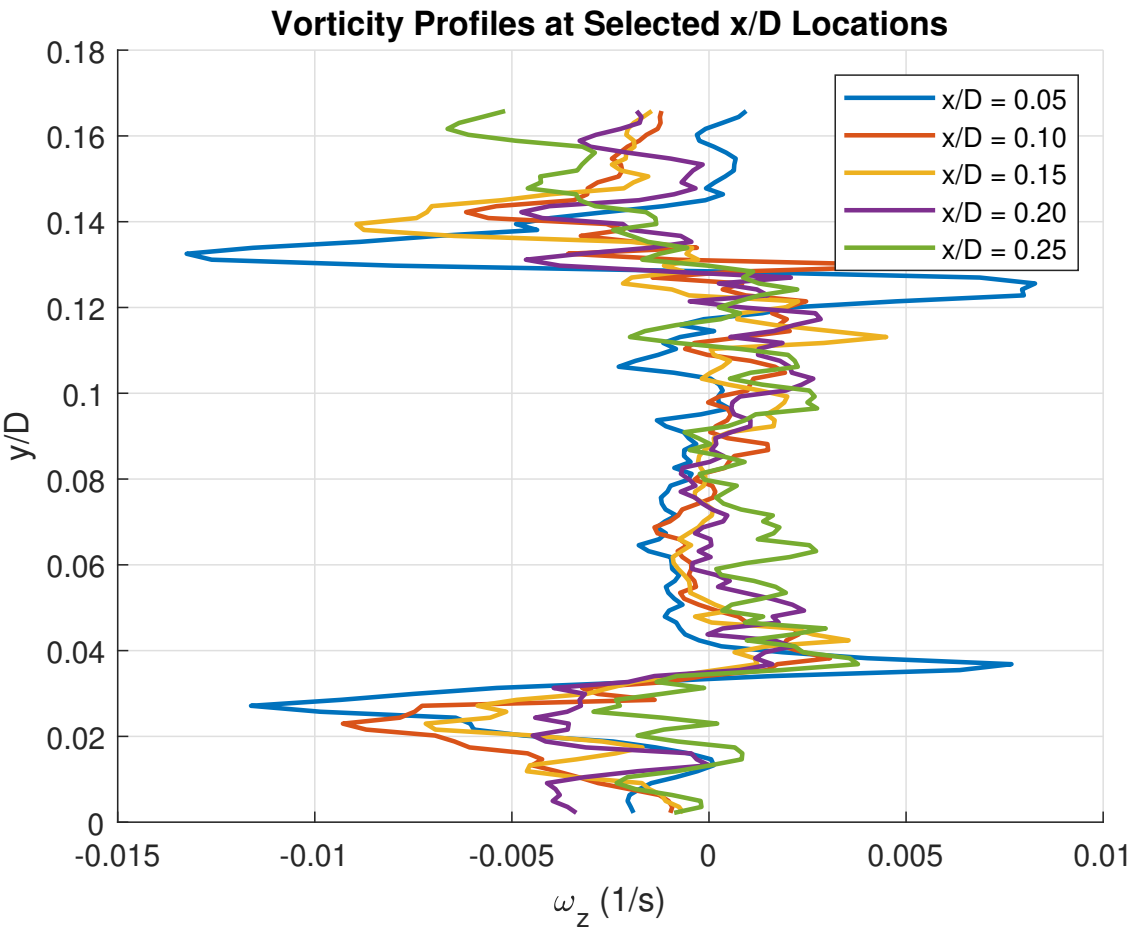


Fig. 10 Vorticity profiles (ω_z) at various x/D locations.

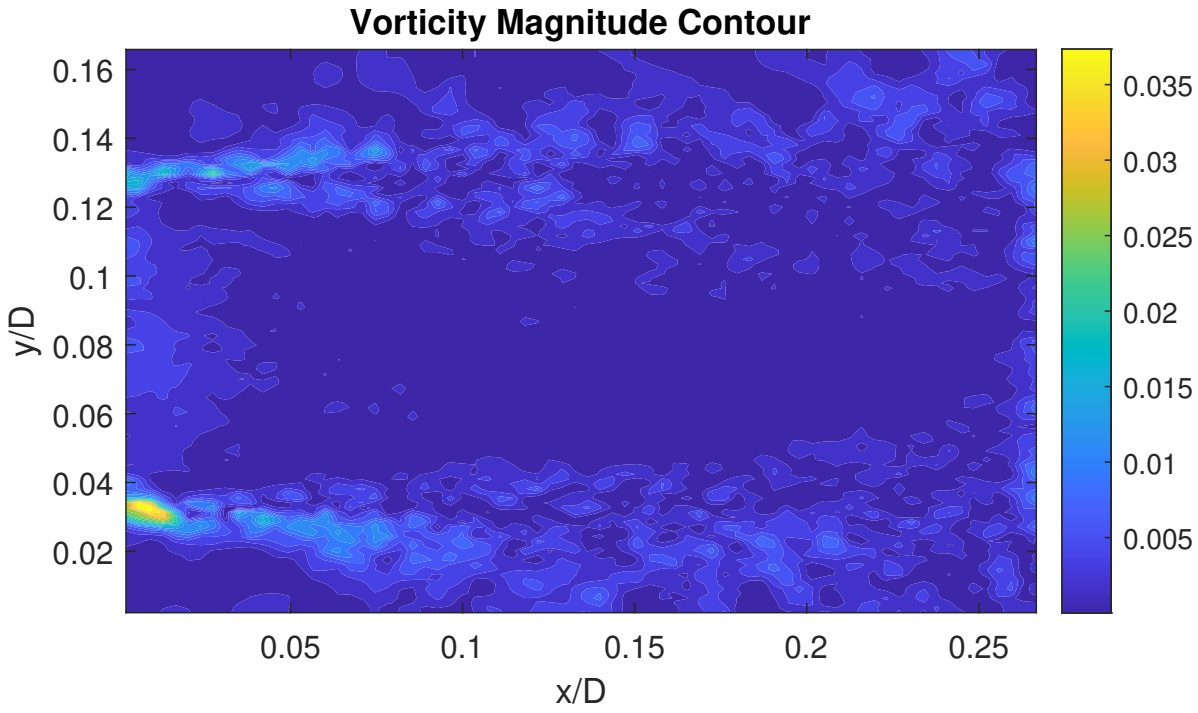


Fig. 11 Contour plot of vorticity magnitude, $|\omega_z|$, highlighting regions of rotational flow.

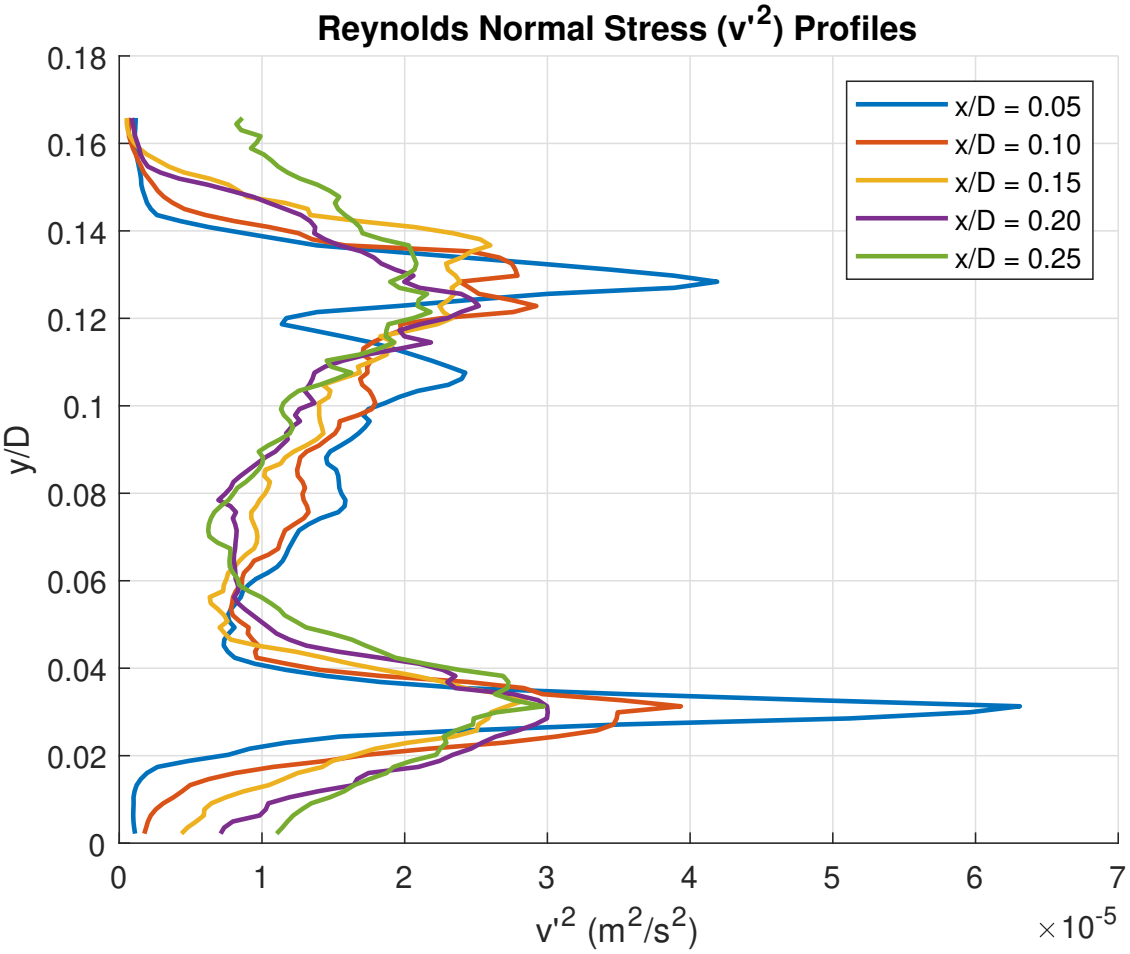


Fig. 12 Reynolds normal stress in the y direction, $\overline{v'^2}$, across multiple x/D locations.

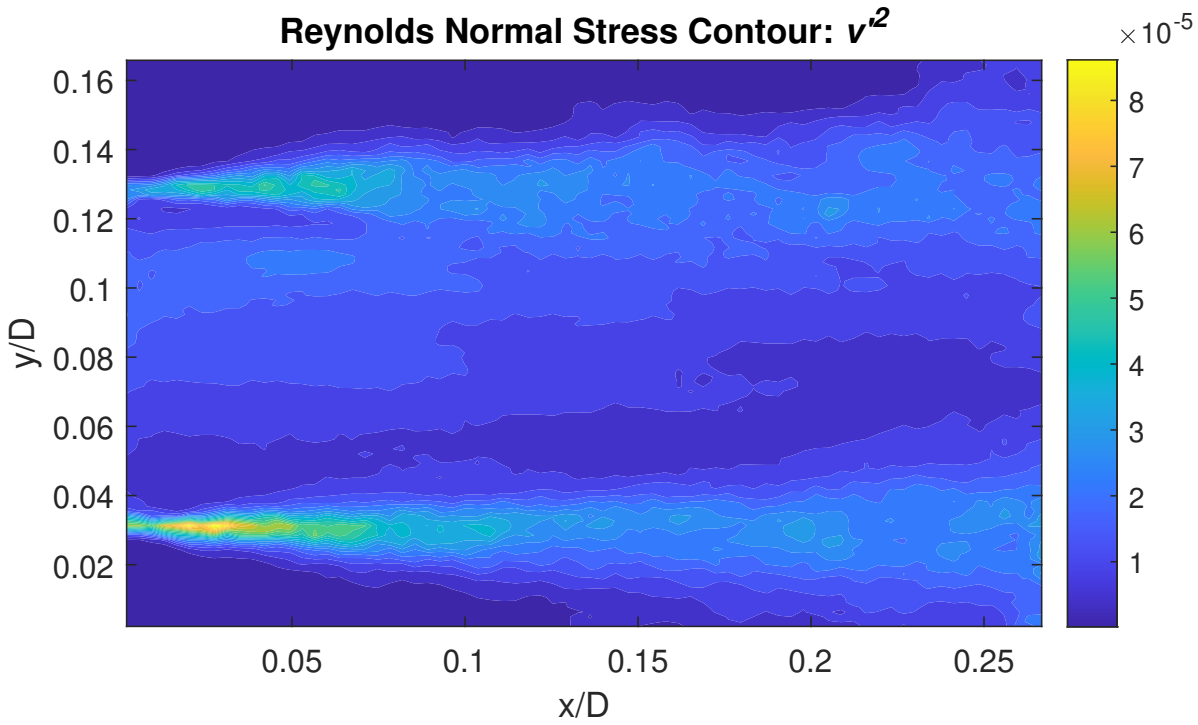


Fig. 13 Contour of $\overline{v'^2}$, showing lateral turbulent intensity in the jet flow.

Each plot above is normalized spatially by nozzle diameter D. Reynolds stresses and vorticity were computed using time-resolved velocity fields as described in Section III.

VII. Discussion

The contour plots and velocity profiles clearly illustrate the evolution of the round turbulent jet, including the presence of a potential core near the nozzle exit and shear layer development downstream. Reynolds normal and shear stress maps confirm that turbulence intensifies as the jet progresses, with peak values occurring around $x/D \approx 0.15$–0.20, consistent with literature predictions [6, 7].

One notable observation is the asymmetry in some of the Reynolds stress profiles, particularly in $u'v'$ and v'^2. This may be due to slight misalignments in the experimental setup or non-uniform lighting across the field. The vorticity magnitude plots also suggest the emergence of coherent vortical structures, highlighting the shear layer instabilities described in theoretical studies [1, 2].

Design Recommendations

- **Camera Mount Stability**: To reduce image jitter and improve vector quality, ensure the camera is mounted on a vibration-isolated platform. Any camera movement directly impacts vector consistency across time steps.
- **Seeding Uniformity**: While flow structures were visible, enhancing the uniformity of seeding particles—particularly near the nozzle—could improve correlation accuracy, especially for early jet development.
- **Illumination Balance**: Ensure the laser sheet maintains consistent intensity across the field. Areas closer to the fiber-optic source appeared slightly brighter, which could affect image pair matching.
- **Edge Padding in Data Processing**: To avoid artifacts introduced by missing data at the field edges, consider expanding the field of view or capturing higher-resolution vector grids for better post-processing interpolation.

Overall, the experiment effectively demonstrated key features of turbulent jet flow, and the proposed improvements would further enhance data quality and repeatability for future implementations.

VIII. Conclusion

This experiment successfully visualized and quantified the development of a turbulent round jet using Particle Image Velocimetry (PIV). Time-averaged velocity fields and Reynolds stress maps revealed a well-defined potential core and the growth of turbulence in the shear layer. Minor asymmetries in stress profiles and vorticity could be attributed to non-uniform seeding or illumination. These discrepancies highlighted the importance of setup precision and lighting uniformity in PIV-based measurements. Overall, the results validated key theoretical principles of jet flow and demonstrated a clear understanding of turbulence characteristics.

Acknowledgments

The author would like to thank Dr. Xiaofeng Liu for his guidance and Teacher's Assistant Andrew Balolong for assistance during the experiment.

References

[1] Liu, X., *A Brief Introduction about Particle Image Velocimetry (PIV)*, 2025. Available from class materials.

[2] Zhao, L., and An, R., "PIV Study of the Near-Field Region of a Turbulent Round Jet," *Proceedings of FEDSM/ICNMM2010*, 2010. ASME 2010 Fluids Engineering Division Summer Meeting.

[3] Anderson, J., *Fundamentals of Aerodynamics*, 6^{th} ed., McGraw-Hill, New York, 2017.

[4] The MathWorks, Inc., *MATLAB Documentation*, The MathWorks, 2024. `https://www.mathworks.com/help/matlab/`.

[5] Liu, X., "Reynolds Stress Calculation based on Planar PIV Measurements," Tech. rep., San Diego State University, 2024. Lecture notes for AE 303 Experimental Aerodynamics.

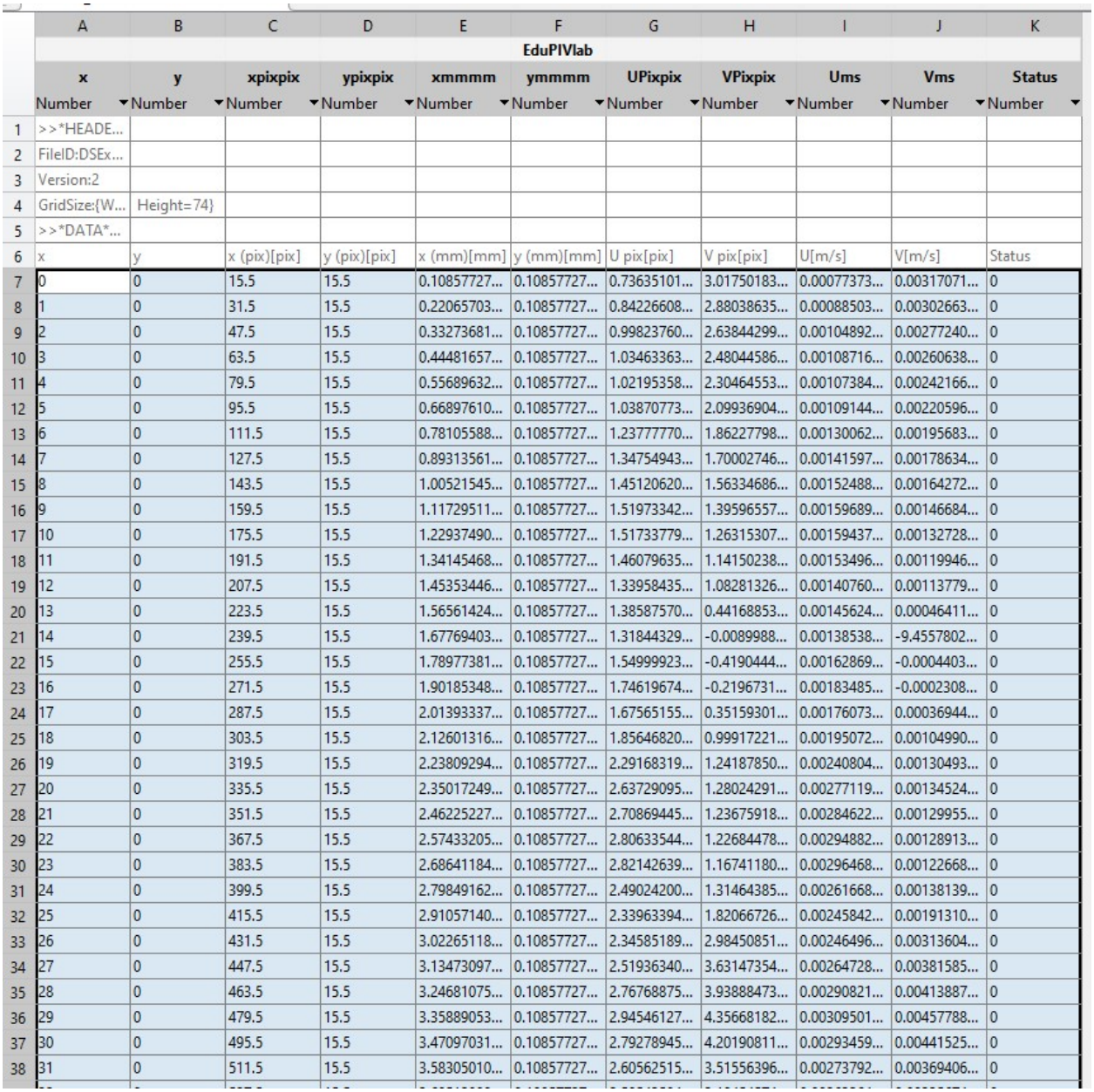
[6] Mattingly, G., and Yeh, T., "Experiments on the Instability of a Round Jet in the Transition Region," *Journal of Fluid Mechanics*, Vol. 14, No. 4, 1962, pp. 399–417.

[7] Cooper, A., and Jackson, J., "The Stability of Low Reynolds Number Round Jets," *International Journal of Heat and Fluid Flow*, Vol. 25, No. 3, 2004, pp. 363–379.

IX. Appendix

A. Data Example

Figure 14 shows a screenshot of one of the 300 raw data files produced by the PIV system used in this experiment. Each file contains instantaneous velocity field data for a single image pair, with vector components recorded in both pixel units and meters per second. These files were processed using EduPIV and MATLAB to compute mean velocities, vorticity, and Reynolds stress components.



	A	B	C	D	E	F	G	H	I	J	K
	EduPIVlab										
	x	y	xpixpix	ypixpix	xmmmm	ymmmm	UPixpix	VPixpix	Ums	Vms	Status
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	>>*HEADE...										
2	FileID:DSEx...										
3	Version:2										
4	GridSize:{W...	Height=74}									
5	>>*DATA*...										
6	x	y	x (pix)[pix]	y (pix)[pix]	x (mm)[mm]	y (mm)[mm]	U pix[pix]	V pix[pix]	U[m/s]	V[m/s]	Status
7	0	0	15.5	15.5	0.10857727...	0.10857727...	0.73635101...	3.01750183...	0.00077373...	0.00317071...	0
8	1	0	31.5	15.5	0.22065703...	0.10857727...	0.84226608...	2.88038635...	0.00088503...	0.00302663...	0
9	2	0	47.5	15.5	0.33273681...	0.10857727...	0.99823760...	2.63844299...	0.00104892...	0.00277240...	0
10	3	0	63.5	15.5	0.44481657...	0.10857727...	1.03463363...	2.48044586...	0.00108716...	0.00260638...	0
11	4	0	79.5	15.5	0.55689632...	0.10857727...	1.02195358...	2.30464553...	0.00107384...	0.00242166...	0
12	5	0	95.5	15.5	0.66897610...	0.10857727...	1.03870773...	2.09936904...	0.00109144...	0.00220596...	0
13	6	0	111.5	15.5	0.78105588...	0.10857727...	1.23777770...	1.86227798...	0.00130062...	0.00195683...	0
14	7	0	127.5	15.5	0.89313561...	0.10857727...	1.34754943...	1.70002746...	0.00141597...	0.00178634...	0
15	8	0	143.5	15.5	1.00521545...	0.10857727...	1.45120620...	1.56334686...	0.00152488...	0.00164272...	0
16	9	0	159.5	15.5	1.11729511...	0.10857727...	1.51973342...	1.39596557...	0.00159689...	0.00146684...	0
17	10	0	175.5	15.5	1.22937490...	0.10857727...	1.51733779...	1.26315307...	0.00159437...	0.00132728...	0
18	11	0	191.5	15.5	1.34145468...	0.10857727...	1.46079635...	1.14150238...	0.00153496...	0.00119946...	0
19	12	0	207.5	15.5	1.45353446...	0.10857727...	1.33958435...	1.08281326...	0.00140760...	0.00113779...	0
20	13	0	223.5	15.5	1.56561424...	0.10857727...	1.38587570...	0.44168853...	0.00145624...	0.00046411...	0
21	14	0	239.5	15.5	1.67769403...	0.10857727...	1.31844329...	-0.0089988...	0.00138538...	-9.4557802...	0
22	15	0	255.5	15.5	1.78977381...	0.10857727...	1.54999923...	-0.4190444...	0.00162869...	-0.0004403...	0
23	16	0	271.5	15.5	1.90185348...	0.10857727...	1.74619674...	-0.2196731...	0.00183485...	-0.0002308...	0
24	17	0	287.5	15.5	2.01393337...	0.10857727...	1.67565155...	0.35159301...	0.00176073...	0.00036944...	0
25	18	0	303.5	15.5	2.12601316...	0.10857727...	1.85646820...	0.99917221...	0.00195072...	0.00104990...	0
26	19	0	319.5	15.5	2.23809294...	0.10857727...	2.29168319...	1.24187850...	0.00240804...	0.00130493...	0
27	20	0	335.5	15.5	2.35017249...	0.10857727...	2.63729095...	1.28024291...	0.00277119...	0.00134524...	0
28	21	0	351.5	15.5	2.46225227...	0.10857727...	2.70869445...	1.23675918...	0.00284622...	0.00129955...	0
29	22	0	367.5	15.5	2.57433205...	0.10857727...	2.80633544...	1.22684478...	0.00294882...	0.00128913...	0
30	23	0	383.5	15.5	2.68641184...	0.10857727...	2.82142639...	1.16741180...	0.00296468...	0.00122668...	0
31	24	0	399.5	15.5	2.79849162...	0.10857727...	2.49024200...	1.31464385...	0.00261668...	0.00138139...	0
32	25	0	415.5	15.5	2.91057140...	0.10857727...	2.33963394...	1.82066726...	0.00245842...	0.00191310...	0
33	26	0	431.5	15.5	3.02265118...	0.10857727...	2.34585189...	2.98450851...	0.00246496...	0.00313604...	0
34	27	0	447.5	15.5	3.13473097...	0.10857727...	2.51936340...	3.63147354...	0.00264728...	0.00381585...	0
35	28	0	463.5	15.5	3.24681075...	0.10857727...	2.76768875...	3.93888473...	0.00290821...	0.00413887...	0
36	29	0	479.5	15.5	3.35889053...	0.10857727...	2.94546127...	4.35668182...	0.00309501...	0.00457788...	0
37	30	0	495.5	15.5	3.47097031...	0.10857727...	2.79278945...	4.20190811...	0.00293459...	0.00441525...	0
38	31	0	511.5	15.5	3.58305010...	0.10857727...	2.60562515...	3.51556396...	0.00273792...	0.00369406...	0

Fig. 14 Sample raw data file from EduPIV output: `EduPIV_lab.62tbxosb.000000.csv`

B. Sample Calculation

To demonstrate how time-averaged velocity components are calculated, consider the following single data point from `EduPIV_lab.62tbxosb.000000.csv` (Figure 14):

x [mm]	y [mm]	U [m/s]	V [m/s]
0.1086	0.1086	0.0007737	0.0031071

Table 2 Sample Data Point at Grid Location from CSV file

Assume the same data point is repeated for all 300 time steps (for illustration). Then, the time-averaged horizontal

velocity at that location is:

$$\bar{u}(x_i, y_j) = \frac{1}{N}\sum_{t=1}^{N} u(x_i, y_j, t) = \frac{1}{300} \cdot \sum_{t=1}^{300} 0.0007737$$
$$= 0.0007737 \text{ m/s}$$

In practice, the MATLAB script `ImportData_EduPIV.m` reads each CSV file and stores the u and v data in a 3D matrix. The time-averaged fields are computed using:

```
u_mean = mean(u_raw, 3, 'omitnan');
v_mean = mean(v_raw, 3, 'omitnan');
```

This yields $\bar{u}(x_i, y_j)$ and $\bar{v}(x_i, y_j)$ across the entire field. These fields are then interpolated and plotted to visualize flow structures.

C. MATLAB: Instructor Provided Code

The following MATLAB script [4] was provided by the instructor to analyze the raw lab data like the ones shown in Figure 14:

Listing 1 Instructor provided MATLAB Code

```
%% Description -- Please Read!
%
% This script imports the data from the .csv files for the PIV lab of
% AE303. A status is printed to the command window as each file is imported
% and then once again as each file's data is formatted. Each .csv file
% represents one instant in time for the flow. The data was recorded at 150
% Hz, so 300 files corresponds to 2 seconds of data.
%
% This script checks if the data has been read before so that one does not
% spend extraneous time re-importing data. However, if the import portion
% of the script is terminated early, the parsing portion of the script will
% not function properly and the workspace should be cleared before running
% this script again.
%
% To further avoid repeating portions of the script, divide the script
% into sections with the '%% Title' as is done already and take advantage
% of the 'Run Section' option to the right of 'Run' in Matlab.
%
% The purpose of this script is to allow the student to work with the data
% presented rather than put unnecessary time into writing their own import
% and formatting script since there is such a large amount of data.
%
%% Notes on the script output
%
% Upon this script's completion, there will exist 10 new variables in the
% workspace. Each is important for processing the PIV data. They are as
% follows:
%
%   data        This is the raw 3D array containing all of the data from
%               all 300 of the .csv files; dimensions 8806x11x300
%
%   N           The number of time steps in the dataset; 300
%
```

```
%    winsize      The 1x2 array with the number of rows and columns,
%                 respectively, in the vector grid; should be [119,74]
%
%    X            The 2D matrix of the x-coordinates of the vector grid
%
%    Y            The 2D matris of the y-coordinates of the vector grid
%
%    u            The 2D matrix of the u-component of the velocity on the
%                 vector grid with likely erroneous vectors replaced by NaN
%                 with units of m/s
%
%    v            The 2D matrix of the v-component of the velocity on the
%                 vector grid with likely erroneous vectors replaced by NaN
%                 with units of m/s
%
%    u_raw        The 2D matrix of the u-component of the velocity on the
%                 vector grid with no replaced vectors and units of m/s
%
%    v_raw        The 2D matrix of the v-component of the velocity on the
%                 vector grid with no replaced vectors and units of m/s
%
%    u_pix        The 2D matrix of the u-component of the velocity on the
%                 vector grid with likely erroneous vectors replaced by NaN
%                 with units of pixel displacement
%
%    v_pix        The 2D matrix of the v-component of the velocity on the
%                 vector grid with likely erroneous vectors replaced by NaN
%                 with units of pixel displacement
%
% The velocity output arrays are all 3D arrays, where the first two
% dimensions represent the vector grid and the third dimension is how each
% time step is stored. Moving, for example, from u(:,:,1) to u(:,:,2) is
% moving one time step forward in time, while referencing the full vector
% grid of u velocity components.
%
% The bottom of the parse portion of the script includes a commented out
% code which will plot and animation of the flow as the data is parsed.
% Running this portion can cause the code to take longer than desired to
% run, which is why it is commented out.
%
%% Read Data
winsize = [119 74]; % Size of vector grid
L = winsize(1)*winsize(2); % Get size of vector field

if ~exist('data','var') % Check if data read already

    N = 300; % Number of files (time steps)

    data = zeros(L,11,N); % Pre-allocate

    for i = 1:N
        fprintf('Now Retrieving File %d of 300\n',i);
        file = sprintf('EduPIV_lab.62tbxosb.%06d.csv',i-1);
        data(:,:,i) = readmatrix(file); % Save each file to data
```

```
    end
end

%% Parse Data
s = 1;
t = 1;
u = zeros(119,74,300); v = u;
u_raw = u; u_pix = u;
v_raw = v; v_pix = v; % Pre-Allocate
x = u; y = v;
for i = 1:N
    fprintf('Now Parsing Set %d of 300\n',i); % Status display
    for q = 1:8806
        if data(q,11,i) ~= 0
            u(s,t,i) = nan; % Replace flagged vectors with nan
            v(s,t,i) = nan;
            u_pix(s,t,i) = nan;
            v_pix(s,t,i) = nan;
        else
            u(s,t,i) = data(q,9,i); % Parse valid vectors
            v(s,t,i) = data(q,10,i);
            u_pix(s,t,i) = data(q,7,i);
            v_pix(s,t,i) = data(q,8,i);
        end

        u_raw(s,t,i) = data(q,9,i);
        v_raw(s,t,i) = data(q,10,i);

        x(s,t,i) = data(q,5,i); % Record grid
        y(s,t,i) = data(q,6,i);

        s = s + 1; % Move to next row

        if mod(q,winsize(1)) == 0 % Detect is done with column
            t = t + 1; % Move to next vector column
            s = 1; % Reset to first row
        end
        if q == L
            t = 1; % Finished with frame, reset t
            % figure(1) % Update only figure 1
            % contourf(x(:,:,i),y(:,:,i),sqrt(u(:,:,i).^2 + v(:,:,i).^2),20,'LineStyle','no
            % axis equal % Update cool animation of the flow
            % xlabel('x [cm]')
            % ylabel('y [cm]')
            % drawnow
        end
    end
end

X = x(:,:,1); % Grid doesn't change, so we just need a 2D matrix
Y = y(:,:,1);

clear file i q s t x y L
```

```
%% Data Processing
```

D. MATLAB: My Code

The following MATLAB script [4] was used for all Data Reduction and Graph Plotting:

Listing 2 My MATLAB Code for Data Analysis

```
%% AE 303 Lab 6
% Author: Parham Khodadi
% Instructor: Xiaofeng Liu

clear; clc; close all;

%% Load Data
% Set working directory to data folder
cd('EduPIV_Lab_Data');

% Run the import script
run('ImportData_EduPIV.m');

% Return to parent folder for analysis output
cd('..');

%% Data Processing: Time-Averaged Fields, Reynolds Stresses, and Vorticity

% Nozzle diameter in mm (given as 5 cm)
D = 50;

% Time-averaged velocities (raw is more complete, avoids NaNs)
u_mean = mean(u_raw, 3, 'omitnan');
v_mean = mean(v_raw, 3, 'omitnan');

% Fluctuations (Reynolds decomposition from filtered velocities)
u_fluct = bsxfun(@minus, u, mean(u, 3, 'omitnan'));
v_fluct = bsxfun(@minus, v, mean(v, 3, 'omitnan'));

% Reynolds stresses from filtered vectors
uu = mean(u_fluct.^2, 3, 'omitnan');
vv = mean(v_fluct.^2, 3, 'omitnan');
uv = mean(u_fluct .* v_fluct, 3, 'omitnan');

% --- Uniform Grid Generation ---
x_vec = linspace(min(X(:)), max(X(:)), size(X,2));
y_vec = linspace(min(Y(:)), max(Y(:)), size(Y,1));
[X_uniform, Y_uniform] = meshgrid(x_vec, y_vec);

% Interpolated Reynolds stresses
uu_uniform = griddata(X, Y, uu, X_uniform, Y_uniform, 'linear');
vv_uniform = griddata(X, Y, vv, X_uniform, Y_uniform, 'linear');
uv_uniform = griddata(X, Y, uv, X_uniform, Y_uniform, 'linear');

% Interpolate u_mean and v_mean onto the uniform grid
u_uniform = griddata(X, Y, u_mean, X_uniform, Y_uniform, 'linear');
v_uniform = griddata(X, Y, v_mean, X_uniform, Y_uniform, 'linear');
```

```
% Fill any remaining NaNs at the edges
u_uniform = fillmissing(u_uniform, 'nearest');
v_uniform = fillmissing(v_uniform, 'nearest');

% Compute gradients and vorticity
dx = mean(diff(x_vec));
dy = mean(diff(y_vec));
[du_dy, ~] = gradient(u_uniform, dy, dx);
[~, dv_dx] = gradient(v_uniform, dy, dx);
vorticity = dv_dx - du_dy;

% Non-dimensional coordinates
X_nd = X_uniform / D;
Y_nd = Y_uniform / D;

%% Objective 2: Time-Averaged u-Velocity Contour Map

figure;
contourf(X_nd, Y_nd, u_uniform, 20, 'LineStyle', 'none');
axis equal;
colorbar;
xlabel('x/D');
ylabel('y/D');
title('Time-Averaged u Velocity Contour Map (\itu \rm)');
saveas(gcf, 'figs/u_mean_contour.eps', 'epsc'); % Save .eps file

%% Objective 3: Reynolds Normal Stress Contour Maps (\itu'^2\rm and \itv'^2\rm)

% u'2 contour
figure;
contourf(X_nd, Y_nd, uu_uniform, 20, 'LineStyle', 'none');
axis equal;
colorbar;
xlabel('x/D'); ylabel('y/D');
title('Reynolds Normal Stress Contour: \itu''^2\rm');
saveas(gcf, 'figs/uu_reynolds_normal_stress.eps', 'epsc'); % Save .eps file

% v'2 contour
figure;
contourf(X_nd, Y_nd, vv_uniform, 20, 'LineStyle', 'none');
axis equal;
colorbar;
xlabel('x/D'); ylabel('y/D');
title('Reynolds Normal Stress Contour: \itv''^2\rm');
saveas(gcf, 'figs/vv_reynolds_normal_stress.eps', 'epsc');

%% Objective 4: Reynolds Shear Stress Contour Map (\itu'v'\rm)

figure;
contourf(X_nd, Y_nd, uv_uniform, 20, 'LineStyle', 'none');
axis equal;
colorbar;
xlabel('x/D'); ylabel('y/D');
```

```
title('Reynolds Shear Stress Contour: \itu''v''\rm');

% Save .eps file
saveas(gcf, 'figs/uv_reynolds_shear_stress.eps', 'epsc');

%% Objective 5: Vorticity Magnitude Contour Map

% Compute magnitude of vorticity
vort_mag = abs(vorticity);

figure;
contourf(X_nd, Y_nd, vort_mag, 20, 'LineStyle', 'none');
axis equal;
colorbar;
xlabel('x/D'); ylabel('y/D');
title('Vorticity Magnitude Contour');

% Save .eps file
saveas(gcf, 'figs/vorticity_magnitude.eps', 'epsc');

%% Objective 6: Profiles at Selected Streamwise Locations

% Define five x/D positions to extract vertical profiles
xD_locs = [0.05, 0.10, 0.15, 0.20, 0.25];

% Initialize color options for visual distinction
colors = lines(length(xD_locs));

% Function to find the closest column index for each x/D location
get_x_index = @(x_val) find(abs(X_nd(1,:) - x_val) == min(abs(X_nd(1,:) - x_val)), 1);

% Plot: Mean u-velocity profiles
figure; hold on;
for i = 1:length(xD_locs)
    col = get_x_index(xD_locs(i));
    plot(u_uniform(:,col), Y_nd(:,col), 'LineWidth', 1.5, 'Color', colors(i,:), ...
        'DisplayName', sprintf('x/D = %.2f', xD_locs(i)));
end
xlabel('u (m/s)');
ylabel('y/D');
title('Mean u Velocity Profiles at Selected x/D Locations');
legend;
grid on;
saveas(gcf, 'figs/u_profiles_vs_y.eps', 'epsc');

% Plot: Reynolds normal stress u''
figure; hold on;
for i = 1:length(xD_locs)
    col = get_x_index(xD_locs(i));
    plot(uu_uniform(:,col), Y_nd(:,col), 'LineWidth', 1.5, 'Color', colors(i,:), ...
        'DisplayName', sprintf('x/D = %.2f', xD_locs(i)));
end
xlabel('u''^2 (m^2/s^2)');
ylabel('y/D');
```

```
title('Reynolds Normal Stress (u''^2) Profiles');
legend;
grid on;
saveas(gcf, 'figs/uu_profiles_vs_y.eps', 'epsc');

% Plot: Reynolds normal stress v''
figure; hold on;
for i = 1:length(xD_locs)
    col = get_x_index(xD_locs(i));
    plot(vv_uniform(:,col), Y_nd(:,col), 'LineWidth', 1.5, 'Color', colors(i,:), ...
        'DisplayName', sprintf('x/D = %.2f', xD_locs(i)));
end
xlabel('v''^2 (m^2/s^2)');
ylabel('y/D');
title('Reynolds Normal Stress (v''^2) Profiles');
legend;
grid on;
saveas(gcf, 'figs/vv_profiles_vs_y.eps', 'epsc');

% Plot: Reynolds shear stress u''v''
figure; hold on;
for i = 1:length(xD_locs)
    col = get_x_index(xD_locs(i));
    plot(uv_uniform(:,col), Y_nd(:,col), 'LineWidth', 1.5, 'Color', colors(i,:), ...
        'DisplayName', sprintf('x/D = %.2f', xD_locs(i)));
end
xlabel('u''v'' (m^2/s^2)');
ylabel('y/D');
title('Reynolds Shear Stress Profiles');
legend;
grid on;
saveas(gcf, 'figs/uv_profiles_vs_y.eps', 'epsc');

% Plot: Vorticity profiles
figure; hold on;
for i = 1:length(xD_locs)
    col = get_x_index(xD_locs(i));
    plot(vorticity(:,col), Y_nd(:,col), 'LineWidth', 1.5, 'Color', colors(i,:), ...
        'DisplayName', sprintf('x/D = %.2f', xD_locs(i)));
end
xlabel('\omega_z (1/s)');
ylabel('y/D');
title('Vorticity Profiles at Selected x/D Locations');
legend;
grid on;
saveas(gcf, 'figs/vorticity_profiles_vs_y.eps', 'epsc');
```