Numerical Simulation of a Freely Falling Rectangular Plastic Sheet

Mechanical Operations [CB2201]

Contributors:

Aryan Sambare (2301CB08)

Krunal Baraskar (2301CB21)

Tushar Shrivastav (2301CB44)

Parth Ganjewar (2301CB48)

Submission Date: [16 April 2025]

Contents

1 Introduction

The primary goal of this project is to simulate the falling dynamics of a thin plastic sheet using MATLAB. The simulation accounts for both translational and rotational motions under the influence of various aerodynamic and gravitational forces. This modeling is important in fields such as fluid dynamics, aerospace engineering, and the study of chaotic systems.

2 Theoretical Background

2.1 Forces Acting on the Plastic Sheet

The following forces are considered in the simulation:

Gravity: Gravity exerts a downward force given by

$$\mathbf{F}_g = -m\,g\,\hat{k},$$

where m is the mass, g is gravitational acceleration, and $\hat{k}$ is the unit vector in the vertical direction.

Buoyancy: The buoyancy force, due to the displaced air, is given by

$$\mathbf{F}_b = \rho_{\text{air}}\,V\,g\,\hat{k},$$

with ρ_{air} as air density and V the plastic sheet's volume.

Aerodynamic Drag: Drag force opposes the relative motion of the plastic sheet with respect to the airflow:

$$\mathbf{F}_d = -\frac{1}{2}\,\rho_{\text{air}}\,C_d\,A_{\text{proj}}\,|\mathbf{v}_{\text{rel}}|\,\mathbf{v}_{\text{rel}},$$

where C_d is the drag coefficient, A_{proj} is the effective projected area, and $\mathbf{v}_{\text{rel}}$ is the relative velocity between the plastic sheet and the wind.

Lift: The lift force acts perpendicular to the relative wind direction:

$$\mathbf{F}_l = \frac{1}{2}\,\rho_{\text{air}}\,C_l\,A_{\text{proj}}\,|\mathbf{v}_{\text{rel}}|^2\,\hat{l},$$

where C_l is the lift coefficient and $\hat{l}$ is the unit vector in the lift direction. The vector $\hat{l}$ is typically computed based on the cross products involving the sheet's normal vector.

2.2 Rotational Dynamics

Rotational motion is captured via Euler angles (yaw, pitch, and roll). For a rectangular sheet, the moment of inertia is approximated by:

$$I_{xx} = \frac{1}{12}\,m\,(h^2 + t^2), \quad I_{yy} = \frac{1}{12}\,m\,(w^2 + t^2), \quad I_{zz} = \frac{1}{12}\,m\,(w^2 + h^2),$$

where w, h, and t denote the width, height, and thickness, respectively.

The angular acceleration is derived from:

$$\boldsymbol{\alpha} = \mathbf{I}^{-1}\Big(\boldsymbol{\tau} - \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})\Big),$$

with the torque generated by the lift force:

$$\boldsymbol{\tau} = \mathbf{r}_{\text{lift}} \times \mathbf{F}_l.$$

2.3 Net Force Equation

The net force acting on the falling sheet is the vector sum of all individual forces:

$$\mathbf{F}_{\text{net}} = \mathbf{F}_g - \mathbf{F}_d + \mathbf{F}_l + \mathbf{F}_b$$

Expanding each term, we obtain:

$$\boxed{F = m\,g - \tfrac{1}{2}\,\rho\,C_d\,A_{\text{proj}}\,|v|\,v + \tfrac{1}{2}\,\rho\,C_l\,A_{\text{lift}}\,|v|^2\,\hat{n} - \rho\,V\,g\,\hat{z}}$$

Where:

- $\mathbf{F}$: Net Force acting on the sheet (N)
- m : Mass of the sheet = 0.02 kg
- $\mathbf{g}$: Gravitational acceleration vector, magnitude $g = 9.81\,\text{m/s}^2$
- ρ : Air density = 1.225 kg/m^3
- C_d : Drag coefficient = 1.28
- A_{proj} : Projected area (depends on orientation relative to airflow)
- $\mathbf{v}$: Velocity vector of the sheet
- C_l : Lift coefficient = 0.8
- A_{lift} : Effective area contributing to lift (can also be approximated by A_{proj})
- $\hat{n}$: Unit vector in the direction of lift (perpendicular to relative wind)
- V : Volume of the sheet
- $\hat{z}$: Unit vector in the upward vertical direction

3 Code Explanation and Methodology

3.1 Numerical Integration

The simulation uses Euler's method for time integration. For a small time step Δt, the state updates are:

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t,$$

$$\mathbf{p}(t+\Delta t) = \mathbf{p}(t) + \mathbf{v}(t+\Delta t)\Delta t.$$

Similarly, the angular quantities are updated. Although `ode45` is available in MATLAB, Euler's method is used here for simplicity.

3.2 Rotation Matrix

To transform vectors from the body-fixed (local) coordinate frame to the inertial (global) frame, we use a rotation matrix constructed from Euler angles — yaw (ψ), pitch (θ), and roll (ϕ).

The rotation matrix $\mathbf{R}$ is orthogonal ($\mathbf{R}^T = \mathbf{R}^{-1}$) with determinant 1, ensuring preservation of angles and vector magnitudes.

Let the Euler angles be:

- ϕ: Roll (rotation about X-axis)
- θ: Pitch (rotation about Y-axis)
- ψ: Yaw (rotation about Z-axis)

The corresponding rotation matrices for each axis are:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The composite rotation matrix is:

$$\mathbf{R} = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

This matrix transforms local direction vectors into the global frame. It is essential for computing orientation-dependent aerodynamic quantities such as projected area, drag, and lift.

The rotation matrix R converts Euler angles to a transformation matrix using the ZYX convention:

$$R = \begin{bmatrix} \cos\theta\cos\psi - \sin\phi\sin\theta\sin\psi & -\cos\phi\sin\psi & \sin\theta\cos\psi + \sin\phi\cos\theta\sin\psi \\ \cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi & \sin\theta\sin\psi - \sin\phi\cos\theta\cos\psi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix}.$$

This matrix is used to update the sheet's orientation and transform its local corner coordinates to world coordinates for visualization.

3.3 Code Structure and Pseudocode

```
Initialize simulation parameters:
    - Define sheet properties: width, height, thickness, mass, etc.
    - Set aerodynamic coefficients (C_d, C_l), air density, and gravity.
    - Initialize position, velocity, Euler angles, and angular velocity.
    - Compute moment of inertia I for a thin rectangular sheet.

Setup visualization:
    - Create 3D subplots for the overall trajectory and a close-up view.
    - Initialize patch objects to represent the sheet.
    - Initialize a quiver object to display the velocity vector.

For each time step:
    - Compute rotation matrix R from current Euler angles.
    - Calculate relative velocity: v_rel = velocity - wind_velocity.
    - Compute projected area A_proj using the sheet's normal and v_rel.
    - Compute forces:
        Drag:   F_d = -0.5 * rho_air * C_d * A_proj * |v_rel| * v_rel
        Lift:   F_l = 0.5 * rho_air * C_l * A_proj * |v_rel|^2 * unit_vector
            (lift)
        Buoyancy and Gravity forces.
    - Sum forces to obtain net acceleration: a = F_net / mass.
    - Compute torque due to lift: tau = r_lift x F_l.
    - Compute angular acceleration: alpha = I^{-1}*(tau - omega x (I*omega)).
    - Update translational and rotational states using Euler integration.
    - Log simulation data.
    - Update visualizations (patch objects, trajectory line, velocity arrow).

End loop.

Post-process data:
    - Generate time-series graphs (position, velocity, Euler angles, etc.).
```

Listing 1: Pseudocode for Simulation

4 MATLAB Implementation Overview

The MATLAB code integrates the above methodology to simulate the falling sheet dynamics with both translational and rotational components. Key tools and techniques implemented in the code are described below.

Patch and Quiver Objects

The code uses `patch` objects to render the sheet as a polygon in 3D space. This visualization is updated continuously to show the sheet's orientation as it falls. In addition, the `quiver3` function is used to display a velocity vector (drawn as a black arrow) that indicates the magnitude and direction of the sheet's instantaneous velocity. Together, these visual tools help the user understand both the position and the orientation of the sheet during the simulation.

Euler Integration and ode45

A simple, first-order Euler integration scheme is applied to update the state of the sheet (position, velocity, Euler angles, and angular velocities) in discrete time steps:

$$\texttt{state}(t+\Delta t) = \texttt{state}(t) + \frac{d\,\texttt{state}}{dt}\Delta t.$$

This method, while straightforward, provides insights into the step-by-step evolution of the system.

Although Euler integration is used here for clarity, the code comments note that MATLAB's `ode45` solver can serve as a more accurate alternative. With `ode45`, adaptive time-stepping and enhanced numerical stability can be achieved. For example, one could define an ODE function for the entire state vector and integrate it as follows:

```
[t, y] = ode45(@(t, y) dynamics(t, y), [0 T], y0);
```

where `dynamics` computes the derivatives of the state variables, and `y0` contains the initial conditions.

Custom Rotation Matrix Function

To handle the orientation changes, a custom function (`eul2rotm`) converts Euler angles (yaw, pitch, roll) into a 3D rotation matrix using the ZYX convention. This matrix is then applied to transform vectors from the body-fixed (local) frame to the global (inertial) frame. Such transformation is essential for:

- Updating the sheet's orientation in the visualization.
- Correctly computing aerodynamic forces, which depend on the sheet's projected area and direction relative to the wind.

Additional Implementation Details

The overall structure of the MATLAB code is organized as follows:

1. **Initialization:** The simulation begins by defining the sheet parameters (dimensions, mass, and aerodynamic coefficients), the environmental settings (air density, gravitational constant, and wind velocity), and the inertia tensor for a thin rectangular sheet. Initial conditions for the sheet's position, velocity, Euler angles, and angular velocity are also set.
2. **Visualization Setup:** Two 3D subplots are created:
   - An **Overall Trajectory View** where the sheet's path is traced.
   - A **3D Close-Up View** that displays the sheet's instantaneous orientation along with a velocity arrow.
3. **Simulation Loop:** For each time step:
   - The `eul2rotm` function converts the current Euler angles into a rotation matrix.
   - The relative velocity is computed by subtracting the wind velocity from the sheet's velocity.
   - Aerodynamic forces are calculated:
     - **Drag Force** ($F_D = -0.5\,\rho\,C_d\,A_{\text{proj}}\,|\vec{v}|\,\vec{v}$)
     - **Lift Force** ($F_L = 0.5\,\rho\,C_l\,A_{\text{proj}}\,|\vec{v}|^2\,\hat{n}$)
     - **Buoyancy and Gravity Forces** are also computed.
   - The net force is calculated as:

     $$F_{\text{net}} = F_G - F_D + F_L + F_B,$$

     which is then used to update the linear motion using Euler integration.
   - The code computes the torque generated by the lift force about a small offset from the center and uses it (with the inertia tensor) to update the angular velocity and Euler angles.
   - Finally, the visualization objects (`patch` and `quiver3`) are updated to reflect the new state.
4. **Post-Simulation Analysis:** The simulation logs key data (e.g., time, position, velocity, Euler angles, angular velocity, projected area, kinetic energy) for further analysis, and subsequently, several graphs are generated to illustrate the system dynamics.

This combination of visualization, numerical integration, and a modular approach to computing forces and rotations ensures that the simulation is both robust and informative, providing clear insights into the dynamic behavior of the falling sheet.

5 Results and Discussion

Example: Initial Parameters

```
% Sheet Parameters
width = 0.2;        % x-dimension (m)
height = 0.15;      % y-dimension (m)
thickness = 0.001; % z-dimension (m) (thin sheet)
mass = 0.01;        % (kg)
rho_air = 1.225;    % air density (kg/m^3)
g = 9.81;           % gravitational acceleration (m/s^2)
Cd = 1.2;           % drag coefficient
Cl = 0.8;           % lift coefficient
wind_velocity = [1.5; 0; 0];  % Wind in x-direction (m/s)

% Inertia tensor (for a thin rectangular sheet)
I = diag([(1/12)*mass*(height^2 + thickness^2), ...
          (1/12)*mass*(width^2 + thickness^2), ...
          (1/12)*mass*(width^2 + height^2)]);

% Initial Conditions
pos = [0; 0; 1.5];
vel = [0; 0; 0];
eul = deg2rad([20; 10; 0]);  % Euler angles [yaw; pitch; roll]
omega = [2; 2; 2];

% Simulation Settings
dt = 0.01;
T = 5;
N = floor(T/dt);
```

Listing 2: MATLAB Code for Initial Simulation Parameters

Source Links

Watch the simulation video here.
See the source code here.

6 Simulation Graphs

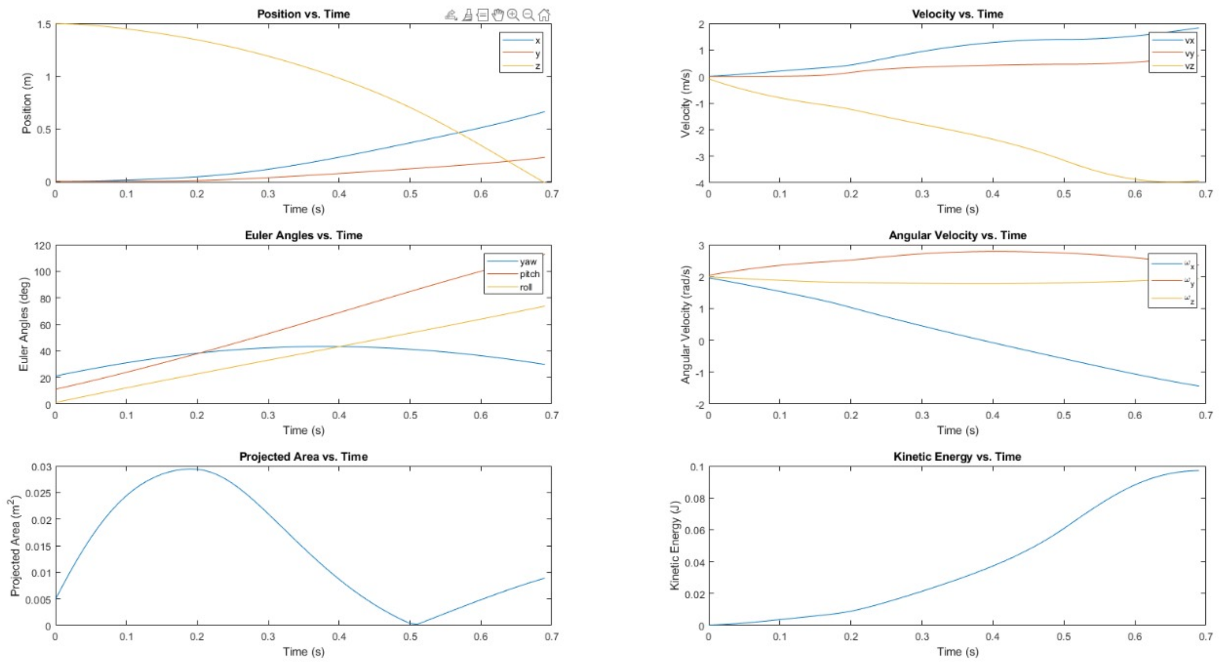


Figure 1: Physical Quantities Vs Time.

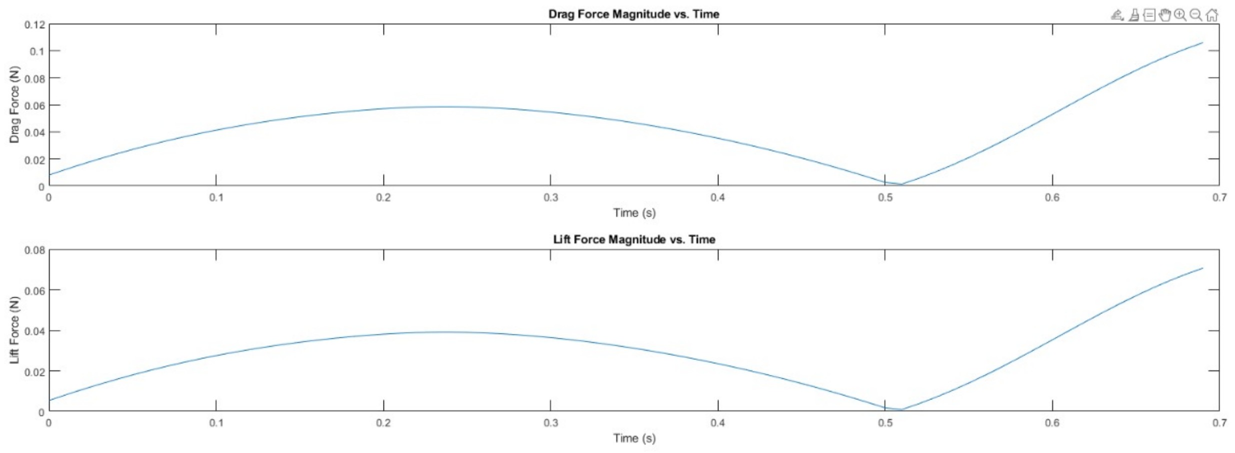


Figure 2: Drag and Lift Force Vs Time.

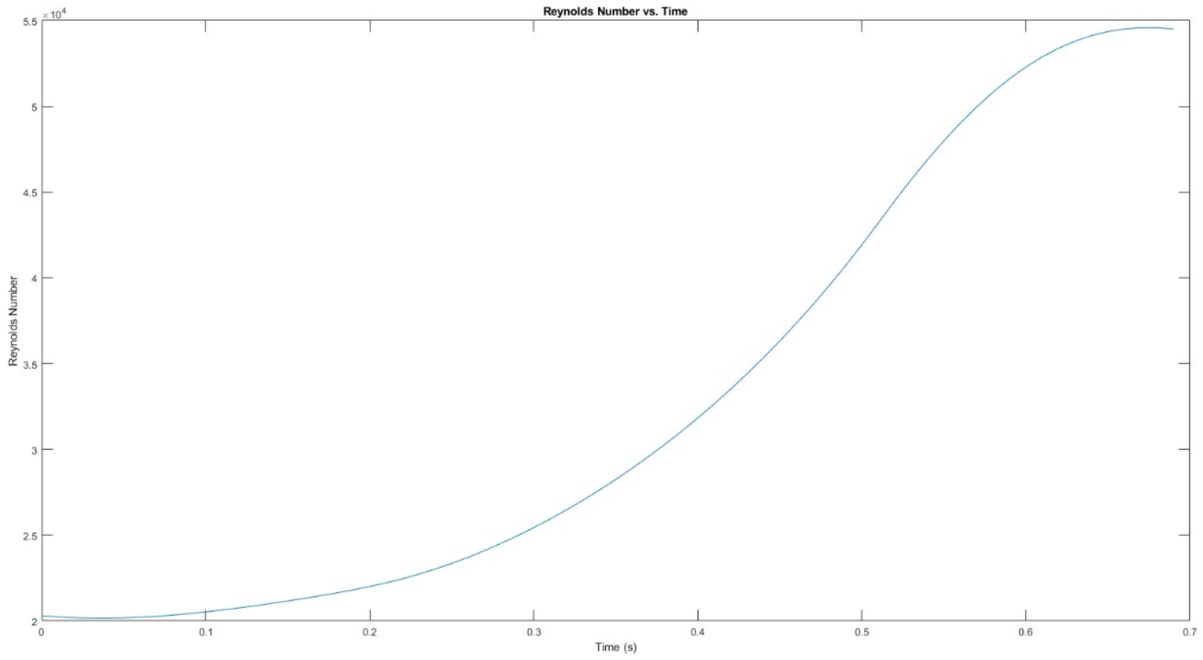
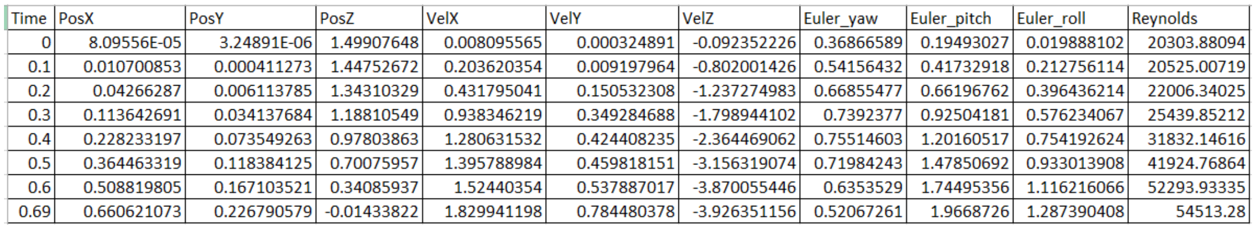
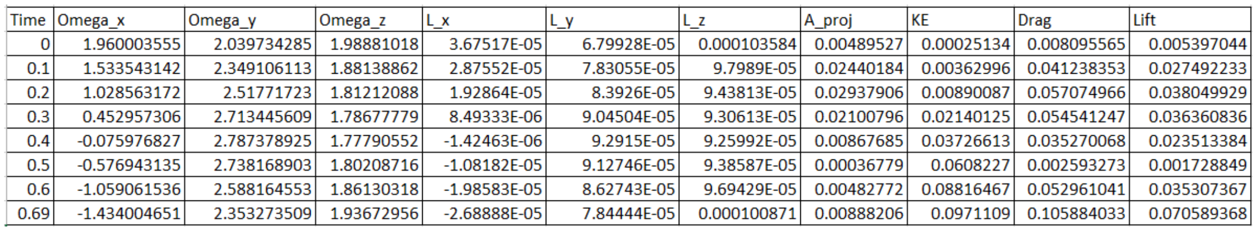


Figure 3: Reynolds number vs time.



Time	PosX	PosY	PosZ	VelX	VelY	VelZ	Euler_yaw	Euler_pitch	Euler_roll	Reynolds
0	8.09556E-05	3.24891E-06	1.49907648	0.008095565	0.000324891	-0.092352226	0.36866589	0.19493027	0.019888102	20303.88094
0.1	0.010700853	0.000411273	1.44752672	0.203620354	0.009197964	-0.802001426	0.54156432	0.41732918	0.212756114	20525.00719
0.2	0.04266287	0.006113785	1.34310329	0.431795041	0.150532308	-1.237274983	0.66855477	0.66196762	0.396436214	22006.34025
0.3	0.113642691	0.034137684	1.18810549	0.938346219	0.349284688	-1.798944102	0.7392377	0.92504181	0.576234067	25439.85212
0.4	0.228233197	0.073549263	0.97803863	1.280631532	0.424408235	-2.364469062	0.75514603	1.20160517	0.754192624	31832.14616
0.5	0.364463319	0.118384125	0.70075957	1.395788984	0.459818151	-3.156319074	0.71984243	1.47850692	0.933013908	41924.76864
0.6	0.508819805	0.167103521	0.34085937	1.52440354	0.537887017	-3.870055446	0.6353529	1.74495356	1.116216066	52293.93335
0.69	0.660621073	0.226790579	-0.01433822	1.829941198	0.784480378	-3.926351156	0.52067261	1.9668726	1.287390408	54513.28



Time	Omega_x	Omega_y	Omega_z	L_x	L_y	L_z	A_proj	KE	Drag	Lift
0	1.960003555	2.039734285	1.98881018	3.67517E-05	6.79928E-05	0.000103584	0.00489527	0.00025134	0.008095565	0.005397044
0.1	1.533543142	2.349106113	1.88138862	2.87552E-05	7.83055E-05	9.7989E-05	0.02440184	0.00362996	0.041238353	0.027492233
0.2	1.028563172	2.51771723	1.81212088	1.92864E-05	8.3926E-05	9.43813E-05	0.02937906	0.00890087	0.057074966	0.038049929
0.3	0.452957306	2.713445609	1.78677779	8.49333E-06	9.04504E-05	9.30613E-05	0.02100796	0.02140125	0.054541247	0.036360836
0.4	-0.075976827	2.787378925	1.77790552	-1.42463E-06	9.2915E-05	9.25992E-05	0.00867685	0.03726613	0.035270068	0.023513384
0.5	-0.576943135	2.738168903	1.80208716	-1.08182E-05	9.12746E-05	9.38587E-05	0.00036779	0.0608227	0.002593273	0.001728849
0.6	-1.059061536	2.588164553	1.86130318	-1.98583E-05	8.62743E-05	9.69429E-05	0.00482772	0.08816467	0.052961041	0.035307367
0.69	-1.434004651	2.353273509	1.93672956	-2.68888E-05	7.84444E-05	0.000100871	0.00888206	0.0971109	0.105884033	0.070589368

Figure 4: Data plot with respect to time.
(All the quantities are in their SI units.)

7 Conclusion

- **Simulation Scheme:**
  - The simulation models a thin plastic sheet falling under gravity, with air drag and lift forces, using both translational and rotational dynamics.
  - Euler integration is used to update position, velocity, orientation (Euler angles), and angular velocity at each time step.
- **General Physics:**
  - The sheet does not fall straight down; it follows a complex, non-linear path due to aerodynamic effects.
  - Drag slows the vertical fall, while lift and initial orientation cause significant horizontal displacement and tumbling.
  - The Reynolds number increases as the sheet accelerates, indicating a transition to turbulent flow.
- **Result Analysis:**
  - Key variables such as position, velocity, orientation, angular velocity, projected area, kinetic energy, drag, and lift are tracked and plotted.
  - The data shows how the sheet's position, speed, and orientation evolve over time, confirming the strong influence of aerodynamic forces.
  - The simulation results match physical expectations: the sheet accelerates downward, develops horizontal motion, and exhibits rotational/tumbling behavior.
- **Summary:**
  - This approach provides a realistic and visual understanding of how thin objects behave when falling through air, highlighting the importance of both aerodynamic forces and rotational dynamics.

8 References

1. Anderson, J. D. (2010). *Fundamentals of Aerodynamics.* McGraw-Hill.
2. Hibbeler, R. C. (2017). *Engineering Mechanics: Dynamics.* Pearson.
3. MATLAB Documentation. (2023). *MATLAB and Simulink Documentation.*
4. Code generation support provided by *ChatGPT* (OpenAI), used for algorithm development and MATLAB scripting.