TECHNICAL REPORT
Design, Modeling, and Simulation of a Branded Replica of the DEERC D2OS Quadcopter Using SolidWorks and MATLAB

By Chinedum Junior (Daniel) Echedom
Summer 2025

TABLE OF CONTENTS

1. Abstract

This report details the design and simulation of a quadcopter fully modeled after the **DEERC D20S**, customized with the **Ascende Lux** logo to reflect a personal engineering brand. The drone was completely modeled and assembled in SolidWorks, with accurate dimensions, materials, and realistic visual details applied. MATLAB simulations analyzed vertical lift-off, hovering, and landing under both ideal and realistic flight conditions, accounting for thrust, drag, and gravity using numerical methods.
The simulation results indicated a landing time of approximately 11.3 seconds. This project highlights skills in CAD modeling, engineering analysis, and numerical simulation while integrating a unique branded design element. It establishes a strong foundation for future aerospace engineering projects and technical design work.

2. Project Objectives

- Create hardcopy rough sketch of DEERC D2OS Quadcopter based on physical observation.
- Design and assemble the exterior of the DEERC D2OS Quadcopter on SolidWorks including the central drone body, propeller arms, motor mounts, etc.
- Assign realistic materials to drone parts (e.g., plastic for the body, arms, and propellers, and metal for the motor.)
- Extract physics-related data (gravity values, total mass, etc.) from the SolidWorks file and document it in Microsoft Excel.
- Generate Technical drawing from SolidWorks Assembly File.
- Render drone model with detailed appearances using SolidWorks visualize.
- Plot a 3-dimensional graph to show the position of drone's center of gravity in space.
- Simulate vertical lift-off, ascent, hover, descent and landing of the drone incorporating the forces of thrust, lift, and gravity.
- Plot a 2-dimensional graph of the drone's altitude against time spent throughout its vertical flight cycle.

3. Background and Design Motivation

Quadcopters have widespread applications in today's world, ranging from recreational use to research, surveillance, and even military operations. This project draws inspiration from the **DEERC D20S**, a lightweight consumer drone selected for its blocky geometry, compact structure, and easily detachable parts—features that made it ideal for CAD modeling and simulation.

To add a personal engineering touch, the finished CAD model features the **Ascende Lux** logo, reflecting both my branding and commitment to creating designs that merge functionality with identity.

The motivation for selecting this drone and undertaking the project includes:

- To learn end-to-end product modeling using SolidWorks, starting from no prior experience with the software.
- To simulate aerodynamic and gravitational effects on a simplified UAV system using MATLAB, also from a beginner's standpoint.

Timeline

Date Range	Task / Milestone
May 20 – June 6	Learning Phase (2.5 weeks): Gained beginner proficiency in SolidWorks (part modeling, assemblies, mates) and MATLAB (basic numerical methods and plotting).
June 7 – June 29	CAD Modeling (3.5 weeks): Modeled central body, propeller arms, motor mounts, and landing skids in SolidWorks.
June 30 – July 6	Assembly and Constraints: Integrated all components, applied mates, verified mass properties and center of gravity.
July 7 – July 13	Material Assignments and Logo Integration: Applied ABS and aluminum materials, extracted mass properties.
July 14 – July 20	Rendering and Technical Drawings: Created photorealistic renders in SolidWorks Visualize with Ascende Lux branding and exported detailed technical drawings.
July 21 – July 27	MATLAB Flight Simulation and Analysis: Implemented vertical lift-off, hover, and landing simulation; generated altitude-time and 3D trajectory plots.
July 28 – Aug 1	Report Compilation: Integrated results, visuals, and reflections into the final technical report.

4 SolidWorks Design and Modeling

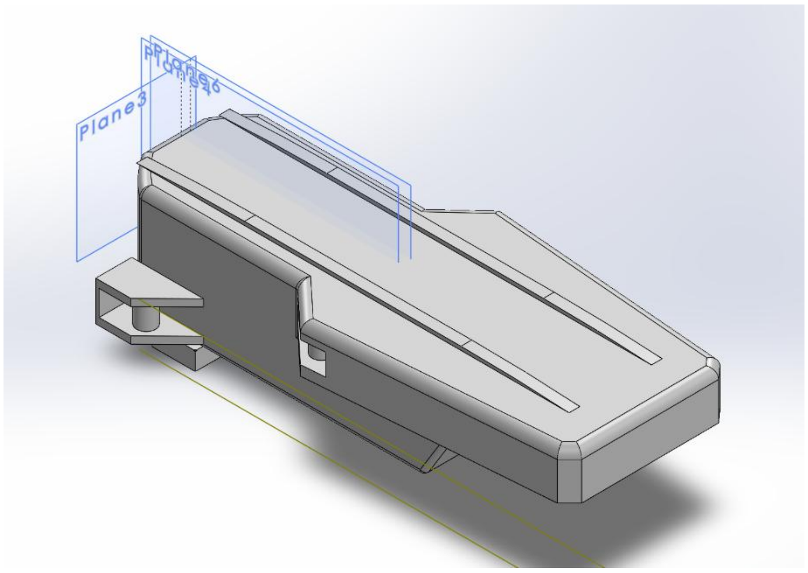
4.1 Initial Approach and Planning

Once comfortable with the software interface and core functions, I conducted a detailed visual inspection of the physical quadcopter.
To ensure efficient workflow, the project was divided into modular parts: central body, four propeller arms with motor mounts and landing skids, four motors, and four propellers.
To lay the groundwork for the CAD modeling I created hand-drawn sketches of each modular part in 5 views: plan (top), plan(bottom), front (side profile), left, and right end. This approach allowed for parallel modelling of each component and a more streamlined assembly. Additionally, realistic material properties were researched and assigned to parts, supporting accurate mass and center of gravity calculations necessary for later MATLAB simulations.

4.2 Modeling the Drone Components

Each quadcopter component was modelled as an individual part before being assembled into a complete system.

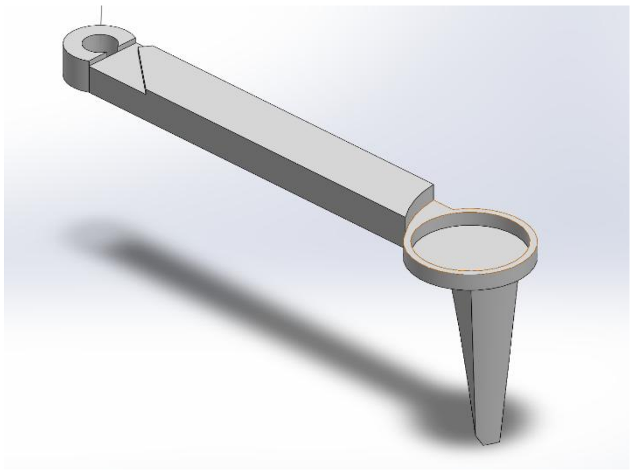
Central Body:
A sketch of the plan (top) view of the quadcopter was created, followed by extrude boss feature applied to it. Rounded edges and cavities were given to it using extruded cuts and fillets for structural detail. Other features such as the battery slot and internal components were also created using extruded bosses and cuts. All features were approximated by visual observation. Reference planes were added to ensure symmetry and alignment of the drone’s center of gravity

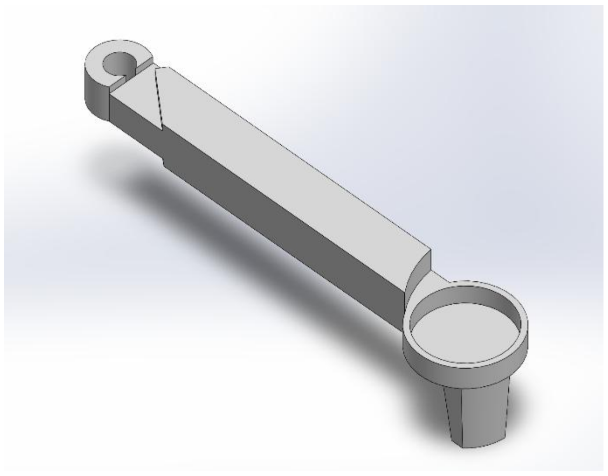


Central Body (*Isometric* view)

Propeller Arms and Motor Mounts:
One front arm and one back arm were created and then mirrored for the opposite arm. Each arm was designed with a rectangular sketch with curved end segments to accommodate the motors followed by an extrude boss feature applied to it. Concentric cuts were used to match the motor base dimensions in the physical quadcopter.

Landing Skids
The cross-sections of the landing skids were sketched directly onto the underside of the motor mounts and then an extruded boss applied. Each landing skid was dimensioned to match the height clearance observed by the physical drone to provide balance during take-off and landing.

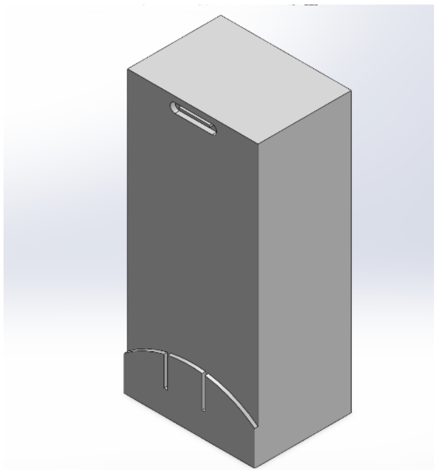

Front Propeller (*Isometric view)*

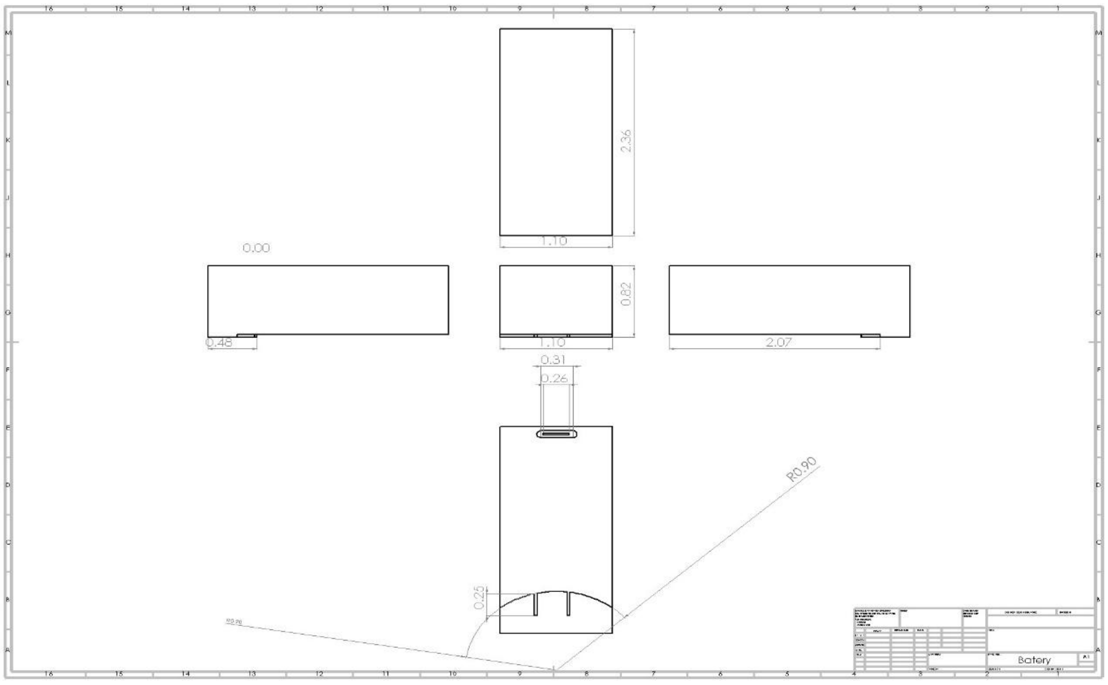

Back Propeller (*Isometric* view)

Motors and Propellers:
Both the motors and Propellers were carefully chosen and sourced from the 3DEXPERIENCE to match quadcopter specifications platform and then imported into the SolidWorks assembly. The motors and Propellers were then scaled to fit the motor mounts and drone weight values. This approach ensured higher accuracy and detail. The imported components were oriented and positioned to reflect correct configuration.

Battery:
The quadcopter battery was modeled as a simplified rectangular solid to represent its geometry and mass contribution to the drone. The plan (top) view of the battery was first sketched and extruded to the measured thickness of the physical component. Cavities and structural details were added using extruded cuts, and a

curved latching tab was modeled at the base to replicate the snap-fit feature that locks the battery securely into the drone's body. All geometric features were approximated by visual inspection of the actual DEERC D2OS battery.
In SolidWorks, the battery was assigned a custom material, with density adjusted so the mass matched the realistic value of approximately 30 g, which is about 20% of the quadcopter's total weight. This ensured that the center of gravity and total mass calculated in the assembly accurately reflected the physical drone, which was critical for the MATLAB simulation of lift-off, hover, and landing.
All dimensions were based on a combination of physical measurements, scaled estimation and proportional reasoning based on the DEERC D2OS design.


Drone Battery (*Isometric* view)



Drone Battery (Technical Drawing)

4.3 Assembly and Constraints

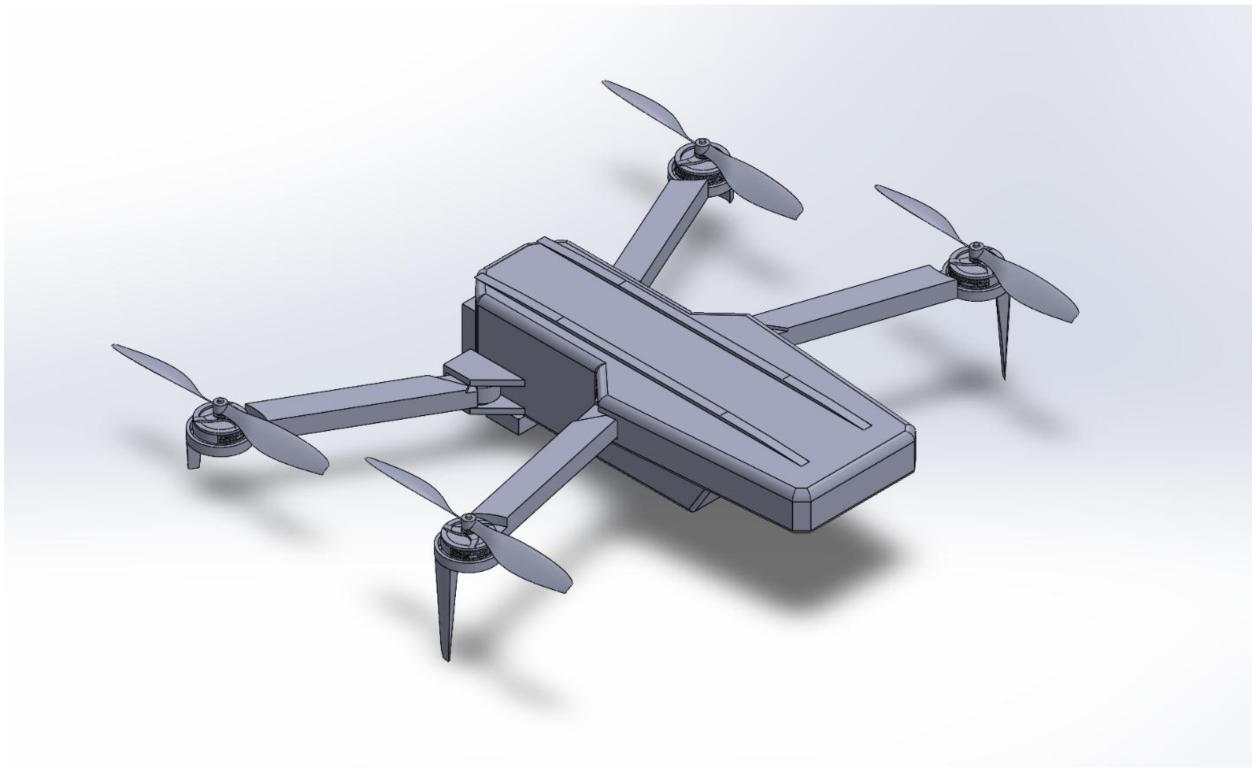
After modelling the individual components, they were imported into a SolidWorks Assembly file for integration. The central body was used as a fixed reference part. All other components were mated relative to it using available and appropriate constraints to replicate the geometry of the real quadcopter.

Mating Strategy:
Standard SolidWorks mates including coincident, concentric, parallel, and distance were applied to accurately align and position components to the central body. Each propeller arm was attached using coincident and parallel mates. Motors were mated to their corresponding position using concentric and coincident mates for proper alignment.

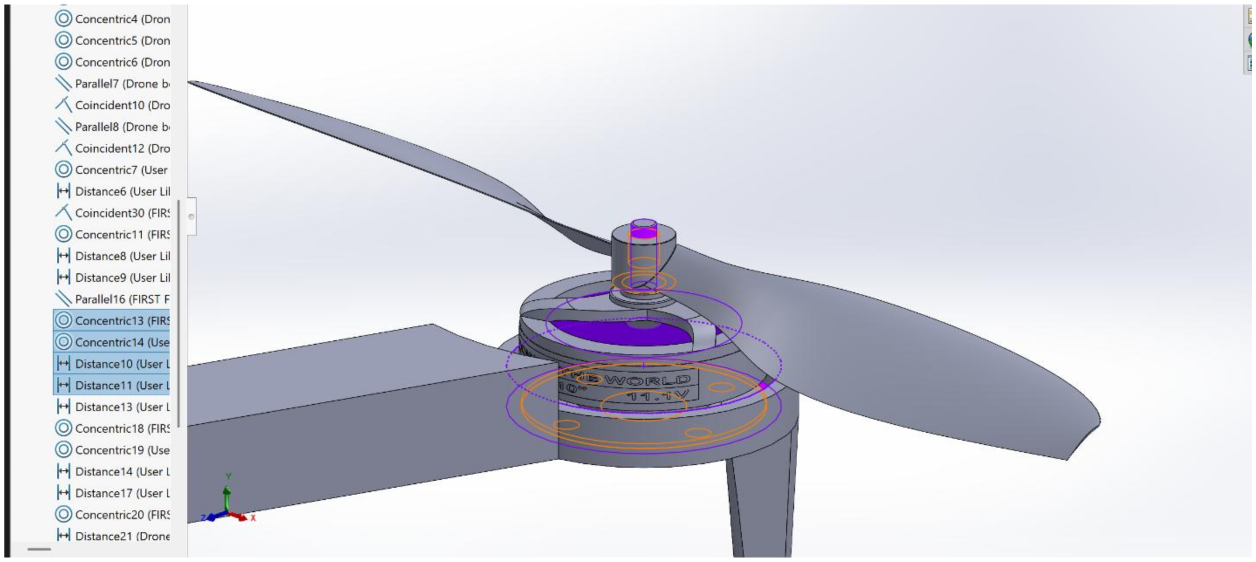
Propeller Placement:
The imported propellers were mounted onto the motors using concentric mates around the motor shafts. Distance mates were also used to elevate the propellers above the motors replicating the physical offset seen in real quadcopters. Due to limitations in the SolidWorks student version used, angular orientation and clockwise/counterclockwise rotation alignment could not be implemented precisely.

Final Assembly Checks:
The assembled quadcopter was reviewed for interferences, alignment errors, over defined constraints, and overall scale consistency. The Mass Properties evaluation tool in SolidWorks was used to verify total mass, individual part mass contributions, and the exact location of the center of gravity. These figures were exported to Microsoft Excel for documentation and later used in the MATLAB simulation phase.

This assembly process created a close digital replica of the DEERC D2OS with minor personal modifications. The final model enabled analysis of mass distribution critical to flight dynamics.



Ful assembled quadcopter model (*isometric* view)



Close-up of arm-motor-propeller mating

4.4 Material Assignment:

Realistic materials were selected from the SolidWorks material library and applied to each part. The central body, propeller arms and landing skids, and propellers were assigned ABS plastic, reflecting the light-weight plastic generally used in consumer-grade drones. The motors were assigned aluminum to represent the typical metal used in brushless motors. These material choices enable SolidWorks to calculate realistic mass properties, which were critical for the MATLAB simulation phase.

4.5 Appearances and Rendering

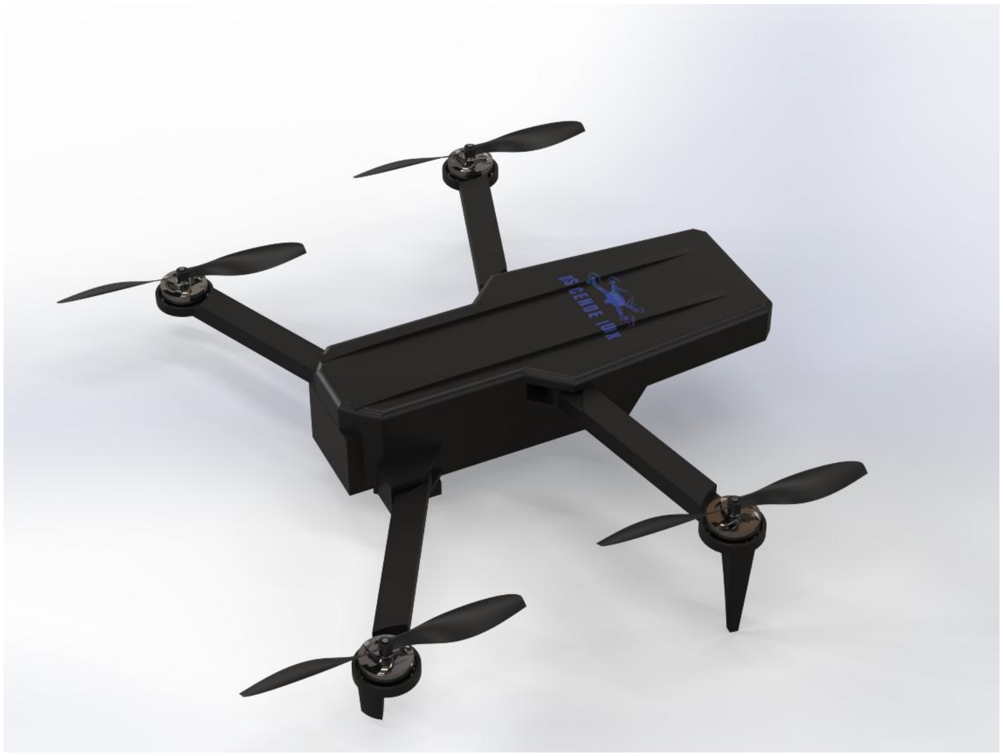
After completing the full assembly, visual realism and material properties were applied to enhance presentation quality. This process involved assigning materials to each component in SolidWorks, adjusting appearances for clarity and realism, and generating high-resolution images using SolidWorks Visualize.

The final assembled model was Imported into SolidWorks Visualize to create realistic renders. Lighting environments were chosen to highlight the model features as well as possible. The Background was kept simple and neutral to maintain focus on the model itself. Rendered images were exported at high resolution for use in presentations and technical documentation.

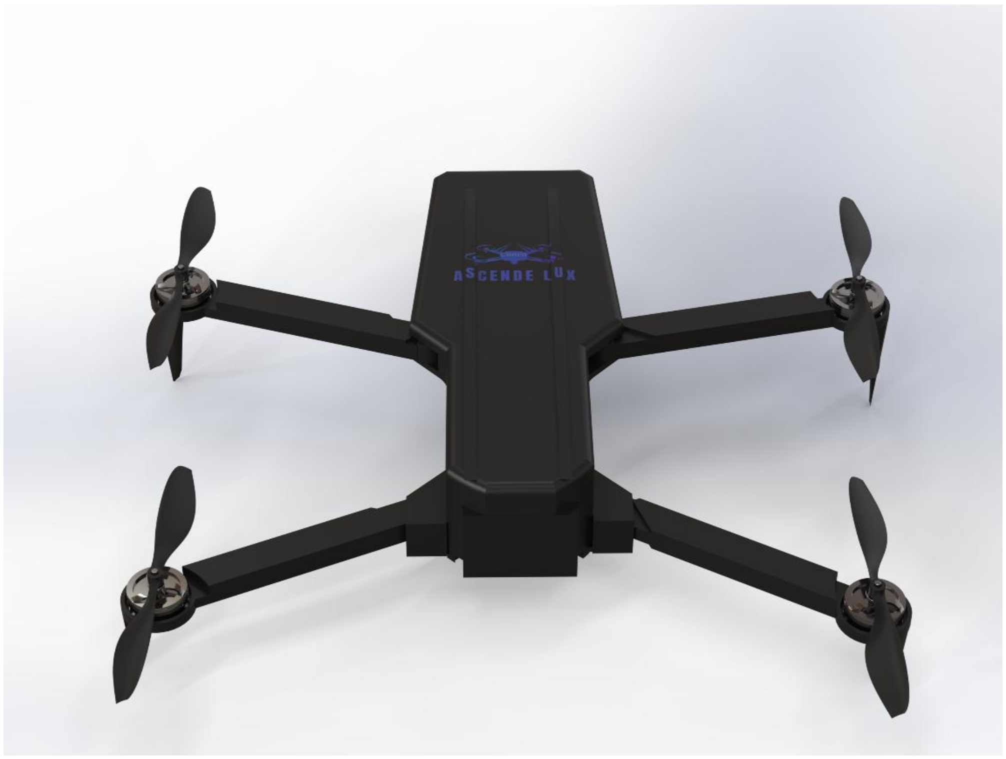
This final stage re-enforced material selection, weight distribution and presentation design.

photorealistic render (*top* view)

ASCENDE LUX



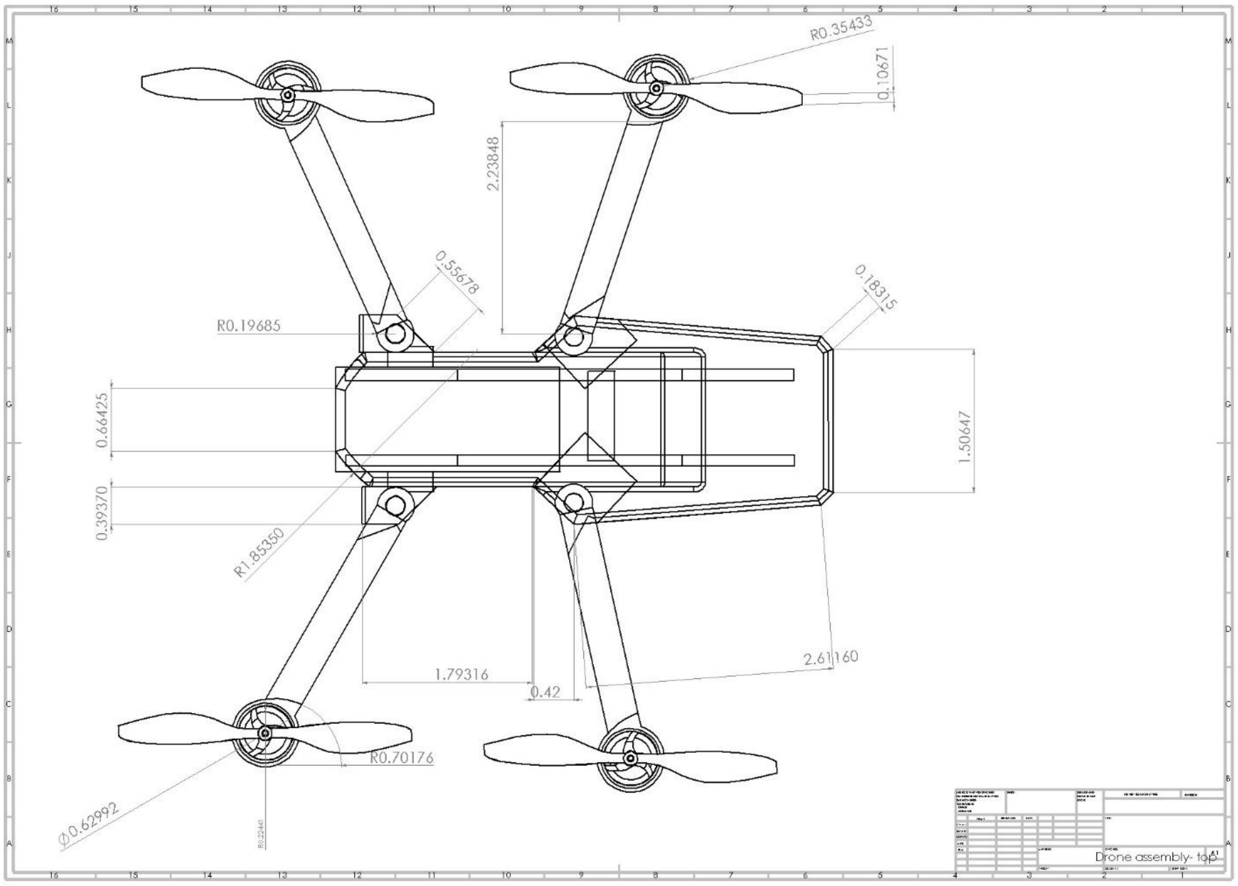
Photorealistic render (*Isometric* view)



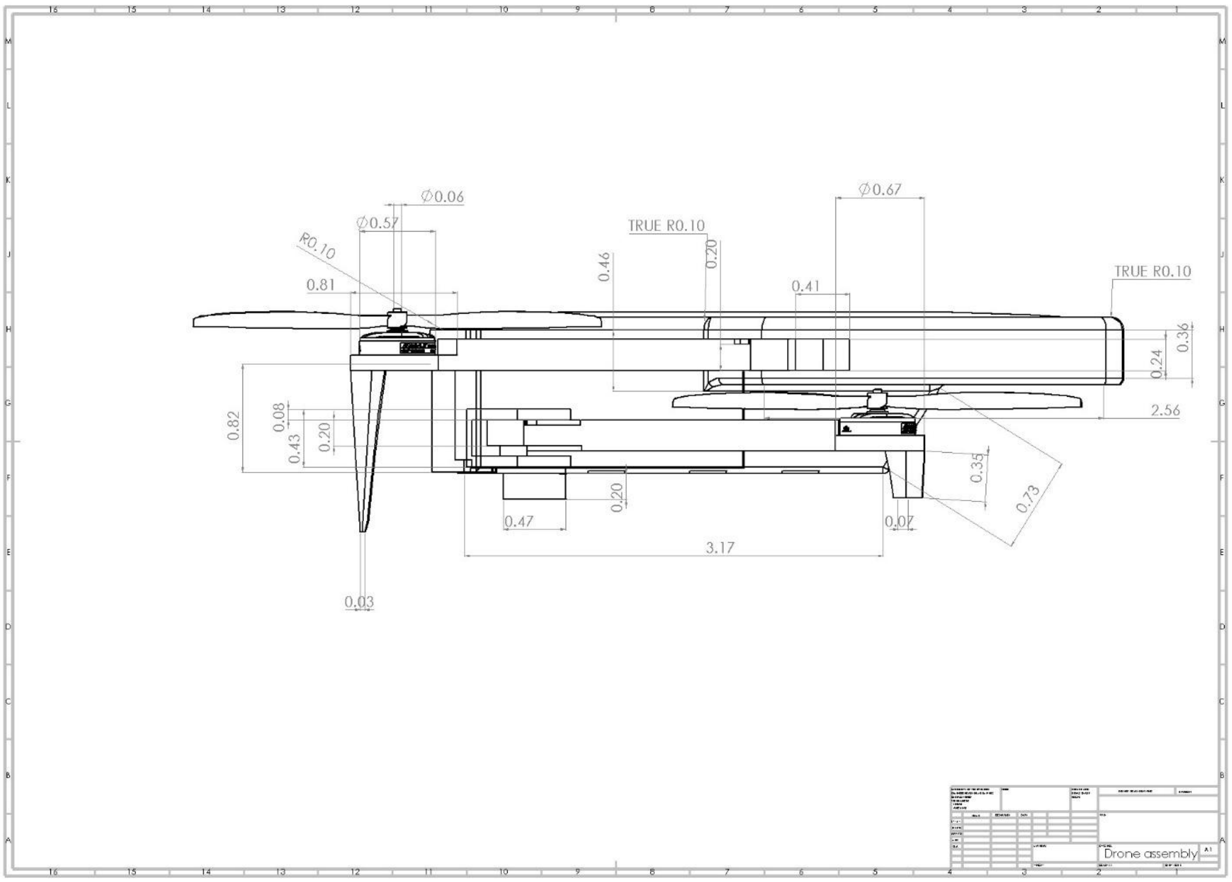
Photorealistic render (*angled* view)

4.6 Technical Drawings

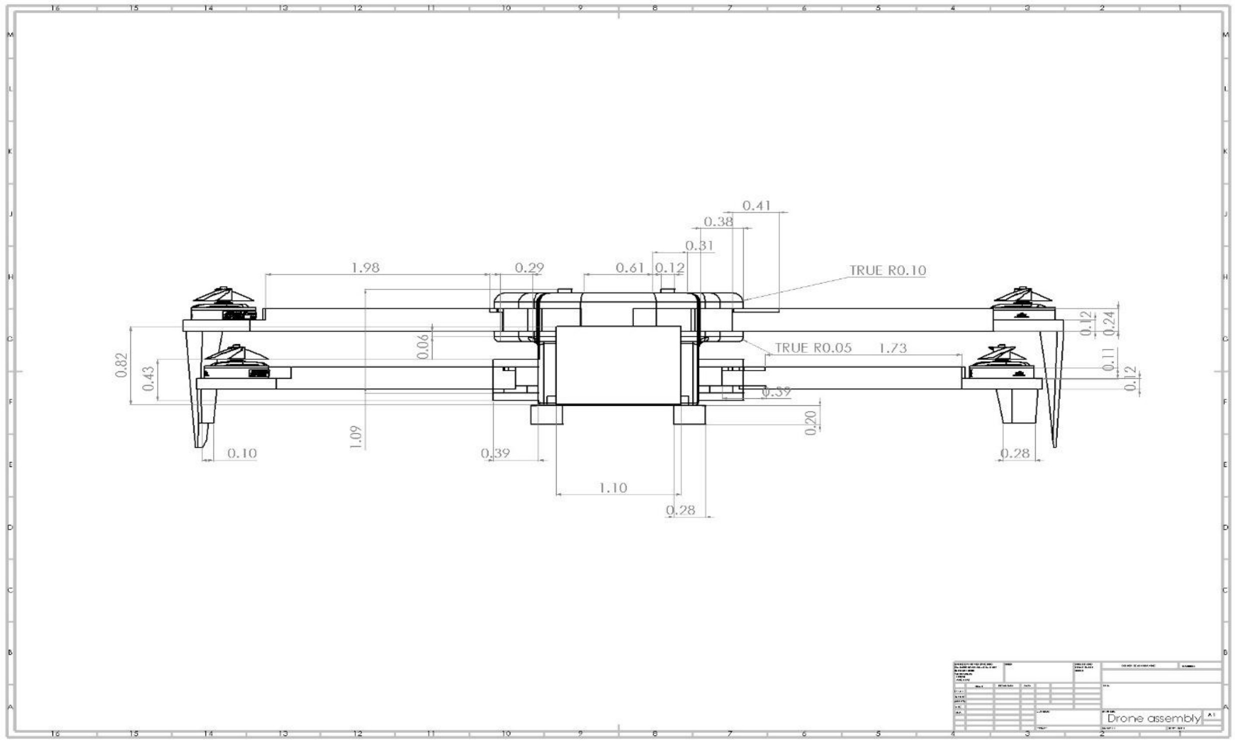
Technical Drawings of the quadcopter model were created from the assembly after integration.



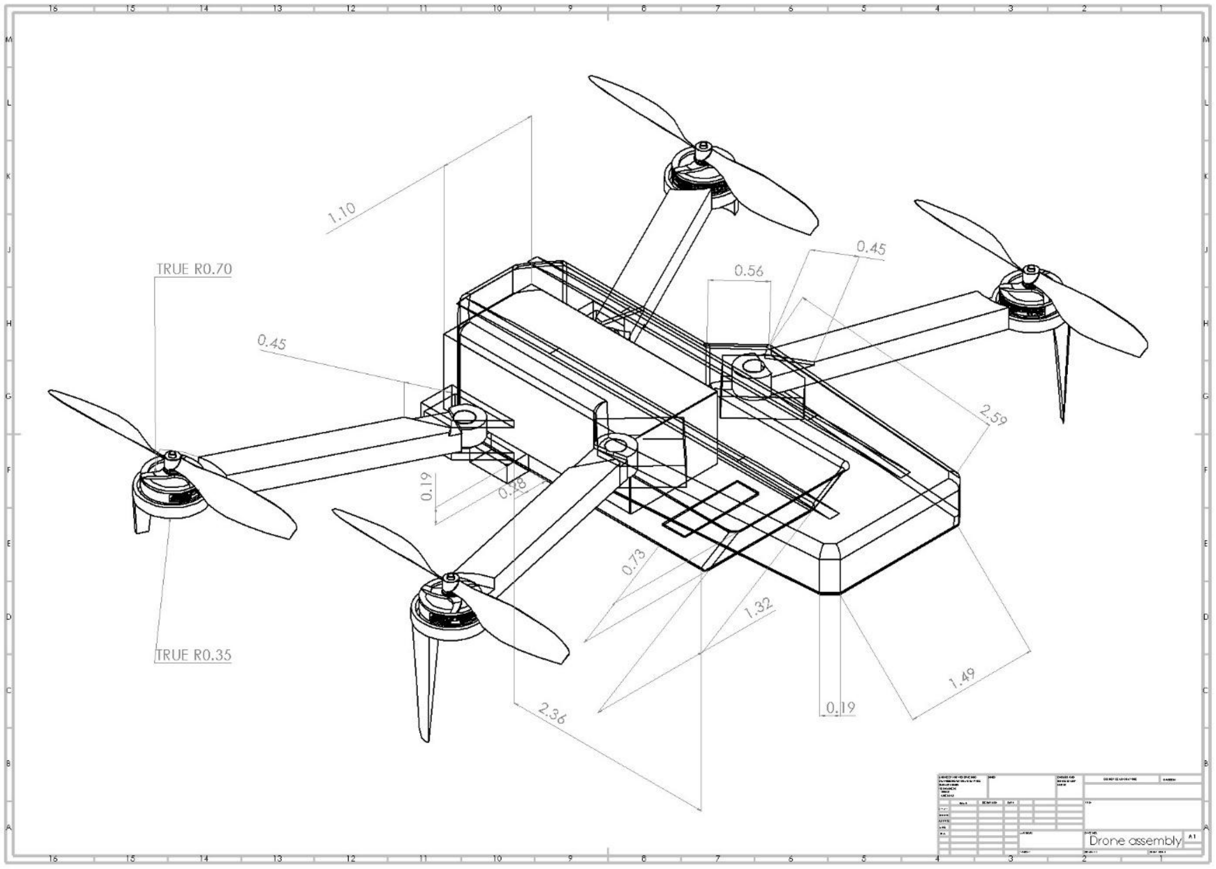
Drone Assembly (*top* view)



Drone Assembly (*front* view)



Drone Assembly (*rear* view)



Drone Assembly (*Isometric* view)

5 MATLAB Simulation and Analysis

5.1 Overview/Objective

This phase involved a 3-D simulation of the vertical motion of the quadcopter in a complete flight cycle (lift-off, hover, and landing) using MATLAB. The simulation was built from scratch using numerical methods (including Euler's method) to solve equations of motion based on thrust, gravity, and drag. With this data, a graph of altitude against time was plotted and the coordinates of the center of gravity of the drone plotted to get a view of its orientation in space.

5.2 Data Collecting and Preprocessing

Physical parameters including total mass and center of gravity coordinates were extracted from the SolidWorks assembly using the Mass Properties evaluate tool. These values were exported to Excel and then imported to MATLAB for the simulation. Other inputs such as air density and acceleration due to gravity were assumed based on standard data.

5.3 Equations and Modeling Approach

The simulation was generally governed by Newton's second law of motion.

$F_{net}=ma$

$F_{net} = T - mg - D$

where:

T = Total Thrust

m = mass of drone

g = acceleration due to gravity

D = Drag force (modeled proportional to velocity

a=acceleration

The Drag force was modelled using the equation for Drag:

D = 0.5 * p * A * Cd * v(i)^2

where:

p = density of air

A = surface area of top face of drone

Cd = coefficient of Drag

v(i) = initial velocity

These equations were implemented using time-stepped numerical integration (Euler's method), allowing simulation of the drone's height and velocity over time.

Parameter	Value	Notes
Mass (m)	0.150 kg	Estimated from drone design in SolidWorks
Gravity (g)	9.81 m/s²	Standard gravitational acceleration
Time step (dt)	0.01 s	For numerical accuracy
Air density (ρ)	1.225 kg/m³	At sea level
Drag coefficient (Cd)	0.9	Estimated for quadcopters

Table of Fixed Parameters for Calculations

5.4 Flight Simulation Phases:

a. Lift-off and Ascent: To simulate smooth ideal lift-off, thrust was set greater than (1.5X) gravitational force (weight) causing upward acceleration for the first 2 seconds.

b. Hover: To simulate ideal Hover, thrust was set equal (1X) to weight, resulting in stable altitude for the next 5 seconds.

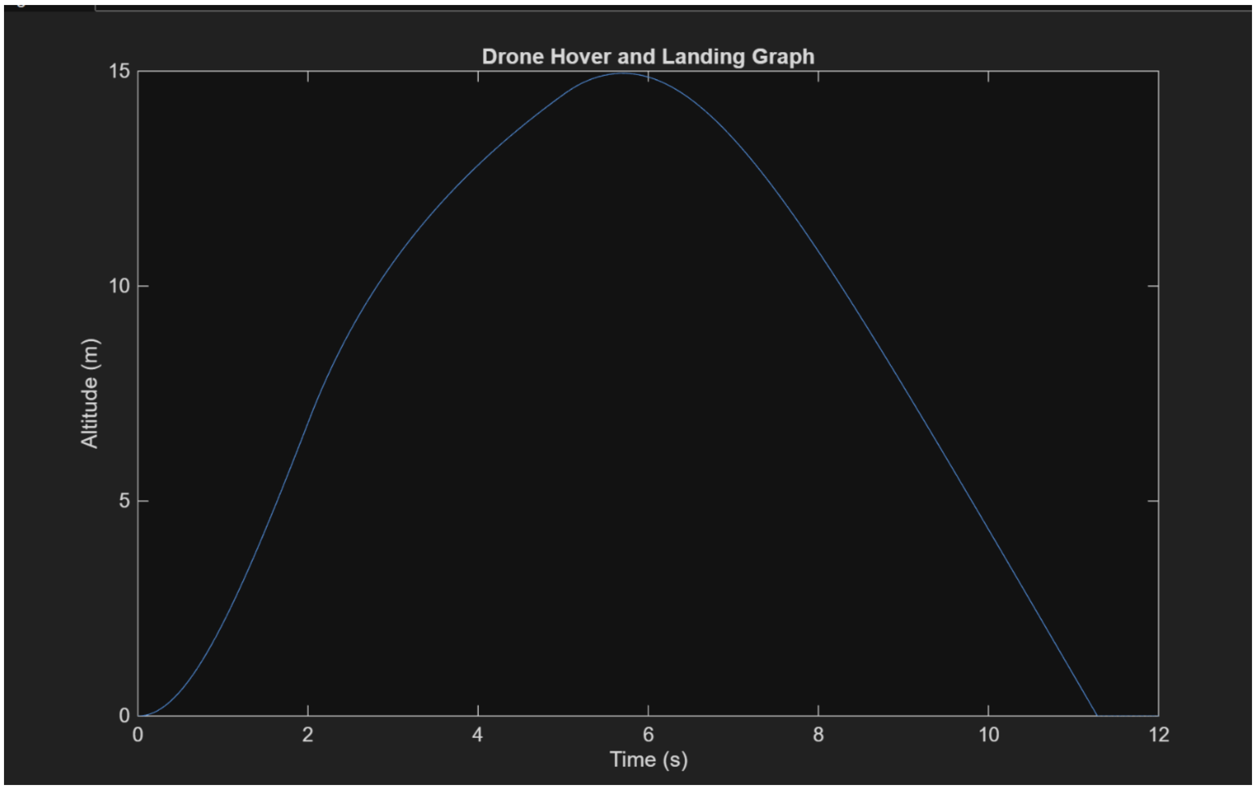
c. Descent and Landing: To simulate ideal Descent, thrust was set less than (0.8X) gravitational force (weight) causing downward acceleration for the remainder of the flight.

The code ensures that the drone does not go "below" ground level, or have a negative altitude by setting altitude, velocity, and acceleration to zero and cutting of the loop-statement once the altitude gets less than zero.

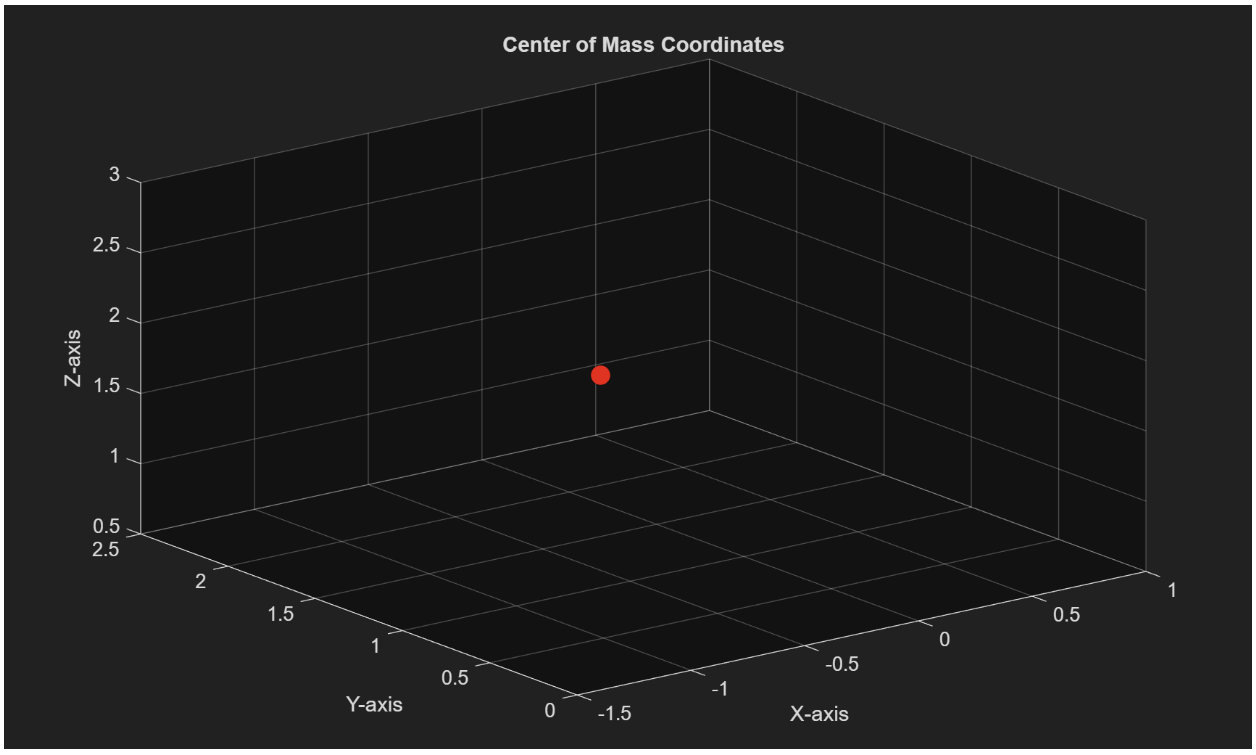
Transitions between stages of the flight cycle were triggered based on time thresholds for change in thrust.

The position of the drone was plotted with each acceleration, velocity, and position value calculated (every 0.01 seconds) and it's trail left behind.

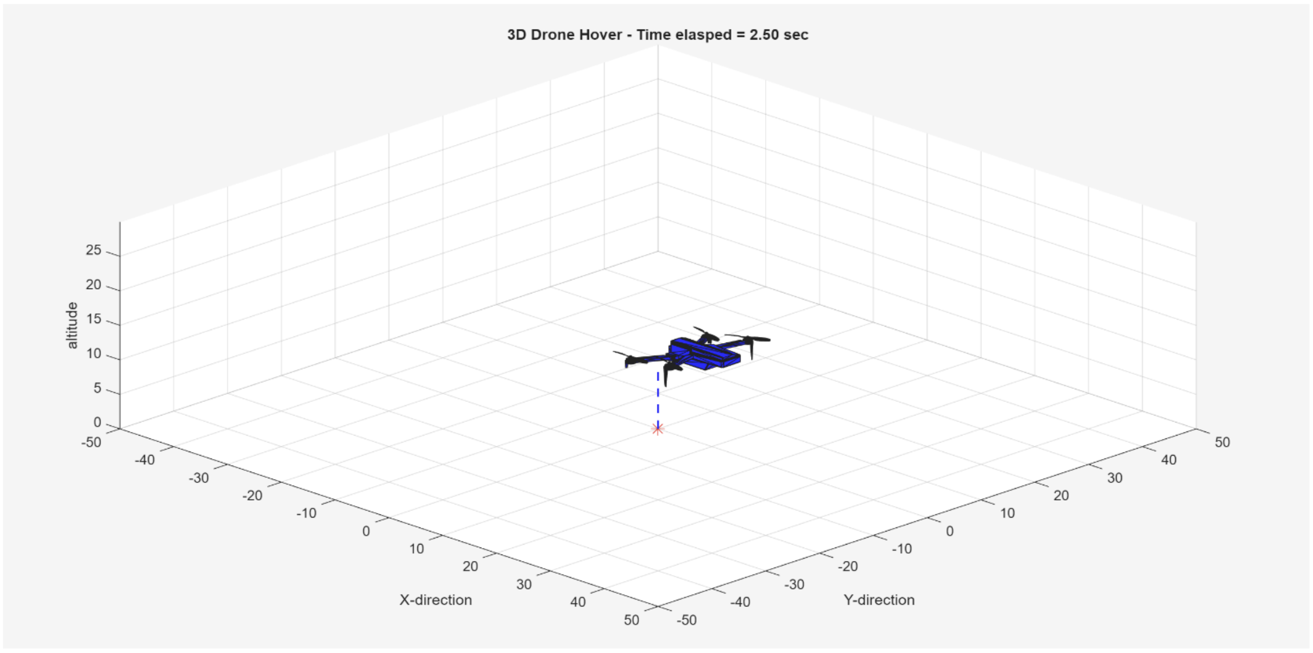
Results and Visualization:



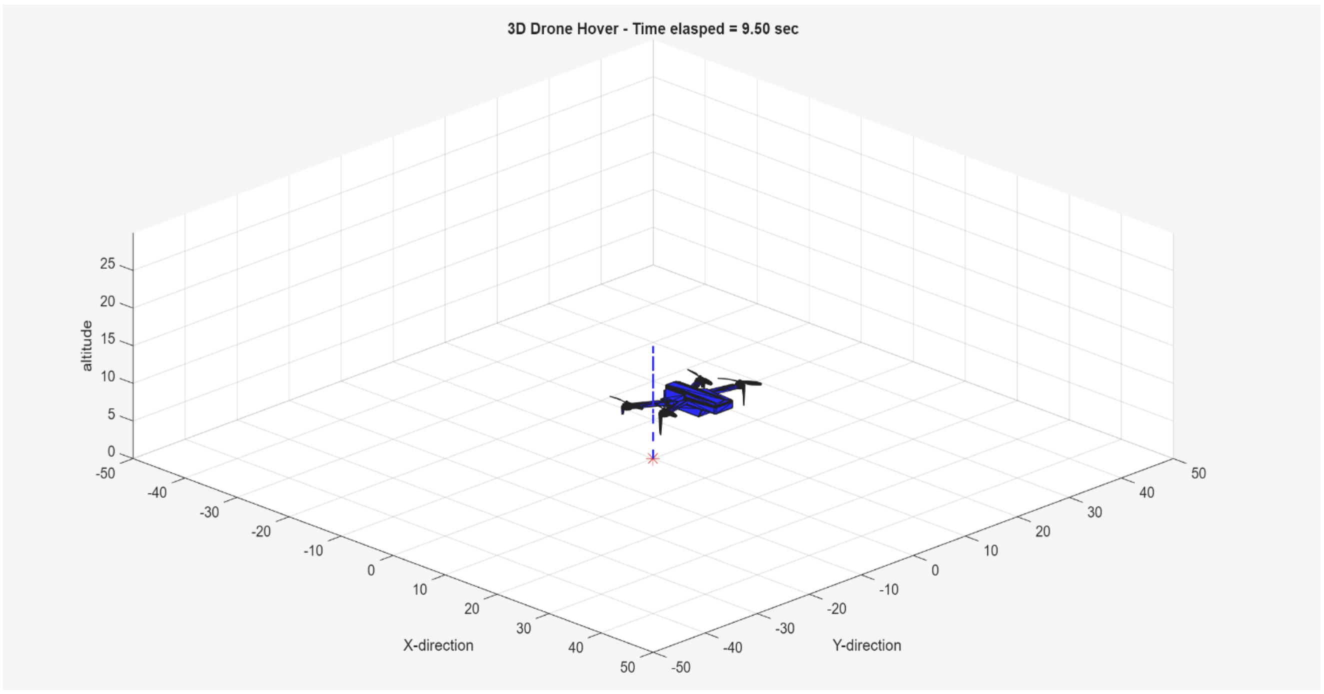
Shows smooth ascent, steady hover, and controlled descent.



Spatial representation of the drone's center of gravity derived from the final SolidWorks assembly. The red point indicates the center of gravity.



3D trajectory plot showing the drone’s liftoff, hovering, and landing phases over a 12-second interval. The drone follows a vertical path with no lateral displacement in the X and Y directions, resulting in a straight-line trajectory along the altitude axis. The drone is currently ascending 2.5 seconds into the simulation.



3D trajectory plot showing the drone’s liftoff, hovering, and landing phases over a 12-second interval. The drone follows a vertical path with no lateral displacement in the X and Y directions, resulting in a straight-line trajectory along the altitude axis The drone is currently descending 9.5 seconds into the simulation.

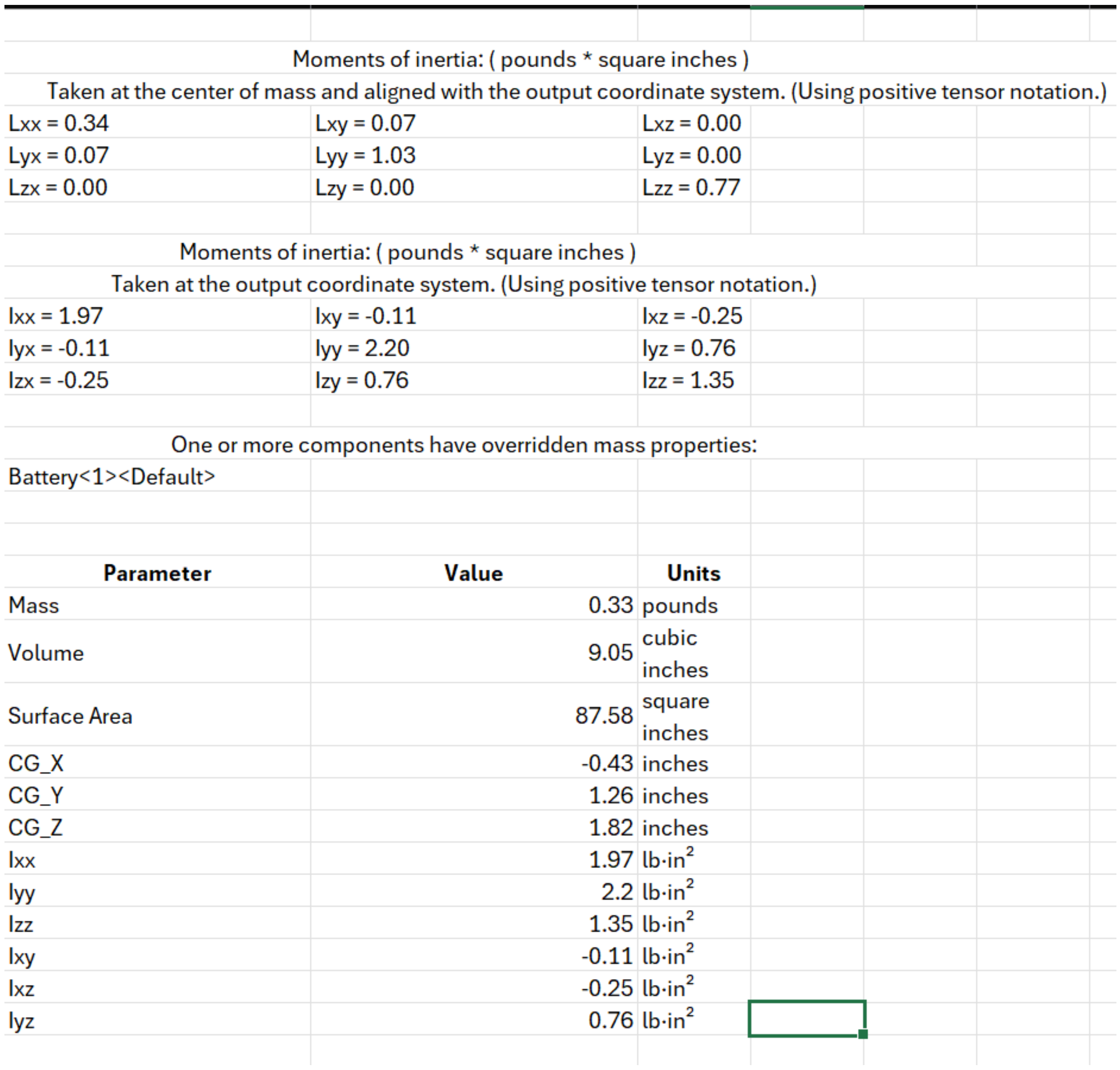
5.5 MATLAB Simulation Setup

Objective

The goal of the simulation was to model the basic flight path of the quadcopter under the typical physical conditions of thrust, drag, and acceleration due to gravity.

MATLAB was chosen due to its compatibility with SolidWorks outputs, ease if numerical modeling, and highly intuitive syntax, making it well-suited for syntax.

Parameters and Inputs
The following mass properties were extracted from SolidWorks to Microsoft Excel and used as input parameters for the MATLAB simulation. Some values such as mass and center of gravity were especially critical for ensuring a realistic hover and thrust profile and preparing an accurate graph of the spatial representation of the drone's center of gravity.



Moments of inertia: (pounds * square inches)
Taken at the center of mass and aligned with the output coordinate system. (Using positive tensor notation.)

Lxx = 0.34	Lxy = 0.07	Lxz = 0.00
Lyx = 0.07	Lyy = 1.03	Lyz = 0.00
Lzx = 0.00	Lzy = 0.00	Lzz = 0.77

Moments of inertia: (pounds * square inches)
Taken at the output coordinate system. (Using positive tensor notation.)

Ixx = 1.97	Ixy = -0.11	Ixz = -0.25
lyx = -0.11	lyy = 2.20	lyz = 0.76
Izx = -0.25	Izy = 0.76	Izz = 1.35

One or more components have overridden mass properties:
Battery<1><Default>

Parameter	Value	Units
Mass	0.33	pounds
Volume	9.05	cubic inches
Surface Area	87.58	square inches
CG_X	-0.43	inches
CG_Y	1.26	inches
CG_Z	1.82	inches
Ixx	1.97	lb·in^2
lyy	2.2	lb·in^2
Izz	1.35	lb·in^2
Ixy	-0.11	lb·in^2
Ixz	-0.25	lb·in^2
lyz	0.76	lb·in^2

Simulation Properties

5.6 Simulation Results and Interpretation

Performance Metrics:
The Flight path simulation indicated the maximum altitude achieved by the quadcopter to

be approximately 14.8 meters, with a total flight duration of approximately 11.3 seconds from takeoff to landing. As expected, the trajectory maintained a purely vertical flight path.

Physical Insight / Engineering Interpretation:
The simulated flight behavior of the quadcopter closely matched theoretical expectations based on its estimated mass and applied thrust (for each stage).
Takeoff Phase (1.5X Weight Thrust)
The drone ascended rapidly and smoothly, demonstrating that the thrust-weight ratio was sufficient enough to overcome gravitational force and the minimal drag considered in the model.
Hover Phase (1.0X Weight Thrust)
The drone remained stable, with no observable lateral drift, as the simulation was constrained to only vertical motion.
Landing Phase (0.8X Weight Thust)
The drone descend smoothly, demonstrating that the thrust-weight ratio was sufficient enough to ensure a gentle descent rather than free fall.

Altitude-Time Graph Analysis:
The altitude-time graph reflects key implications of the thrust settings of 1.5X weight for ascent and 0.8X for descent such as:
a. Steeper Ascent than descent
The initial climb is rapid due to the larger net upward force.
The ascent portion of the altitude-time graph appears *wavy* because the higher velocity amplifies the drag effect, creating small oscillations in altitude as the drone accelerates upwards. This is a key observation that is very evident in real-world drone flying.
b. Smooth and Gentle Descent
With thrust reduced to 0.8X weight, the net downward force is small, resulting in a slow, more controlled landing.
The descent path is smooth and nearly linear, as the lower speed minimizes drag fluctuations.

This pattern demonstrates that aggressive thrust during takeoff can introduce dynamic vertical oscillations, while gentle descent results in stable and predictable motion.

5.7 Limitations and Future Improvements

While the simulation provided valuable insights, some limitations were present in the current approach:

1. Simplified Dynamics

The model only simulated vertical motion with no lateral aerodynamic forces. Rotational Dynamics and altitude control (roll, pitch, and yaw) were not considered

2. Idealized Thrust and Environment

The motor thrust was assumed to be supplied instantly and uniformly, and the environment was treated as wind-free, with constant air density.

Future Improvements

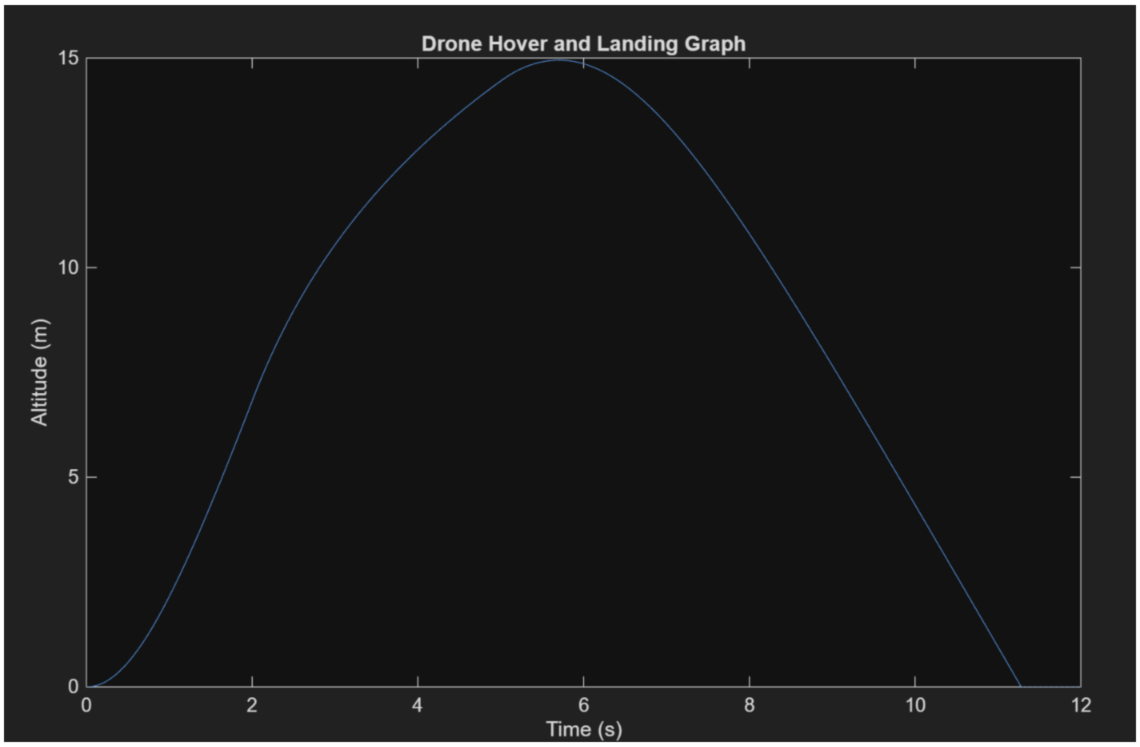
In order to enhance the realism and accuracy of the simulation, the following improvements are suggested:

1. Full 3-Dimensional Simulation

Incorporate lateral motion and rotational dynamics.

2. Realistic Motor response and Environmental Effects

Include motor response delay, wind gusts, turbulence and variable air density to study stability physical performance in real-world environments.



6 Conclusion and Reflection:

Conclusion
The project successfully modeled and simulated the DEERC D2OS quadcopter design, achieving the expected flight performance standards within the defined parameters. The integration of SolidWorks and MATLAB provided valuable insights into the dynamics of quadcopters and performance of the chosen design.

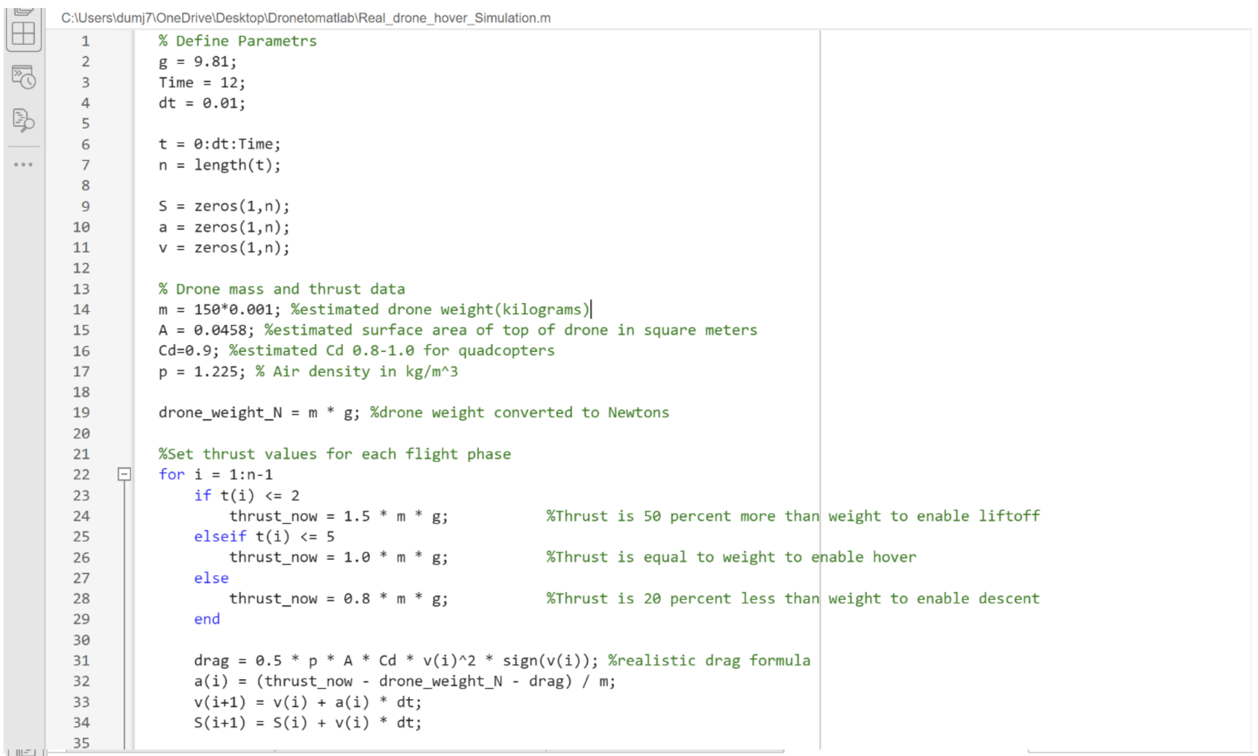
Reflection
Throughout this project, I enhanced my skills in CAD modelling as well as numerical and simulation analysis.
One major obstacle was mastering the SolidWorks and MATLAB environment which required considerable learning time. In future, I hope to incorporate more advanced aerodynamic modeling techniques and design and conduct physical testing to validate simulations.
Overall, this project strengthened my understanding of aerospace design and computational principles and tools.

7 References:

1. MathWorks. (2024). *MATLAB (R2024a) [Computer software]*. MathWorks, Inc.
https://www.mathworks.com
2. Dassault Systèmes. (2024). *SolidWorks (2024) [Computer software]*. Dassault Systèmes.
https://www.solidworks.com
3. DEERC. (2024). *DEERC D20S Mini Drone*.
https://www.deerc.com/products/deerc-d20s
4. Dassault Systèmes. (2024). *3DEXPERIENCE Platform*.
https://www.3ds.com/3dexperience

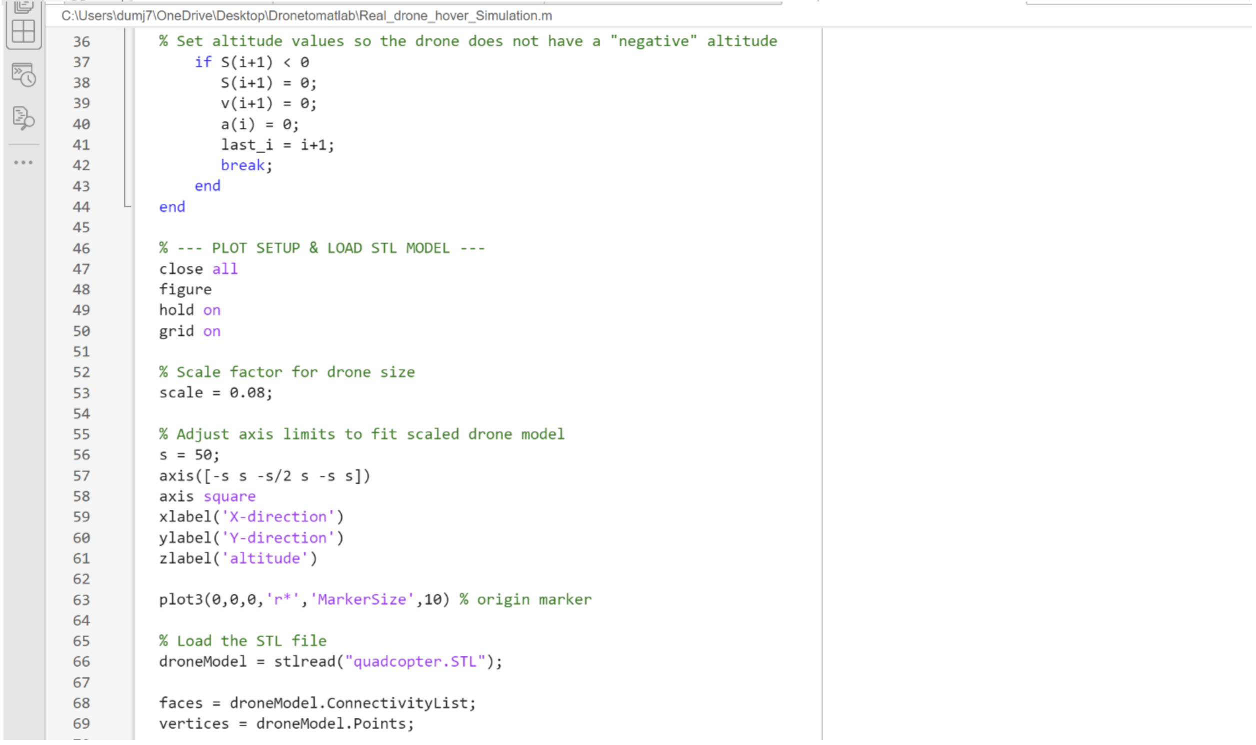
Appendix A: MATLAB Code for 3D Drone Flight Simulation with STL Model

This script simulates the drone’s full vertical flight cycle while visualizing the motion of the STL drone model in 3D. The animation shows the drone’s vertical trajectory and stops upon landing.



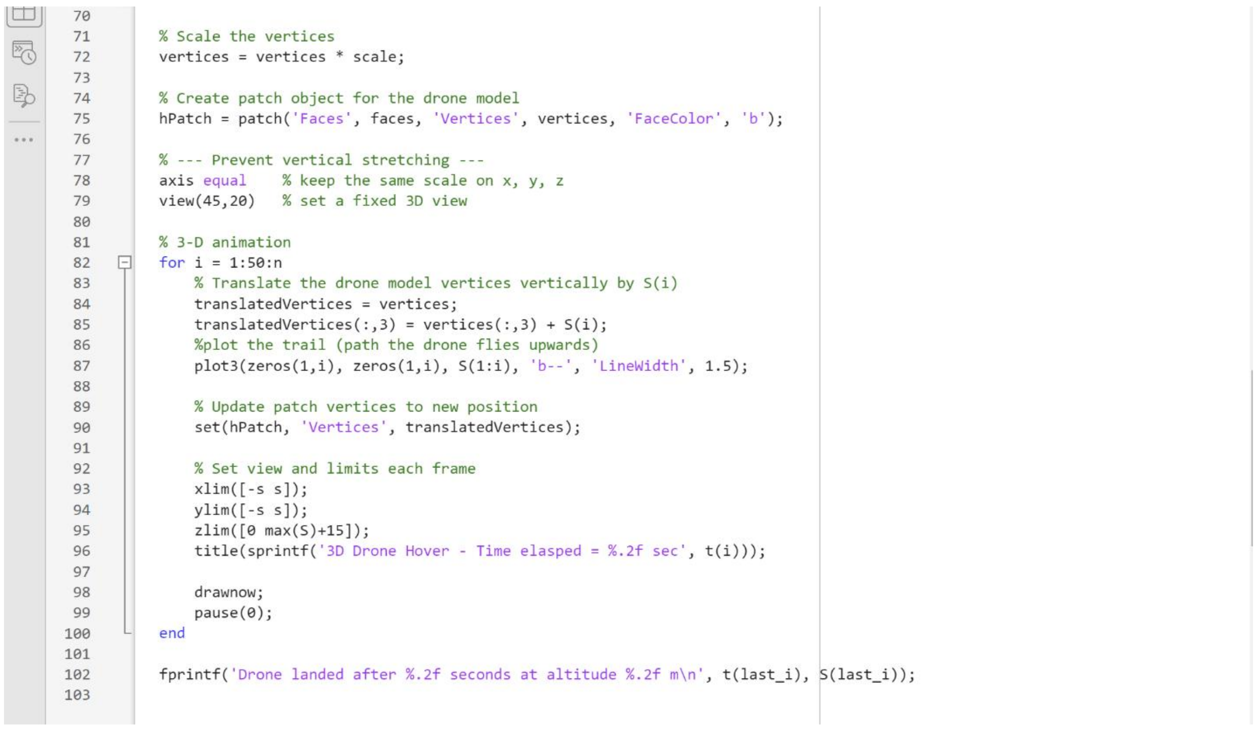
C:\Users\dumj7\OneDrive\Desktop\Dronetomatlab\Real_drone_hover_Simulation.m

```matlab
% Define Parametrs
g = 9.81;
Time = 12;
dt = 0.01;

t = 0:dt:Time;
n = length(t);

S = zeros(1,n);
a = zeros(1,n);
v = zeros(1,n);

% Drone mass and thrust data
m = 150*0.001; %estimated drone weight(kilograms)
A = 0.0458; %estimated surface area of top of drone in square meters
Cd=0.9; %estimated Cd 0.8-1.0 for quadcopters
p = 1.225; % Air density in kg/m^3

drone_weight_N = m * g; %drone weight converted to Newtons

%Set thrust values for each flight phase
for i = 1:n-1
    if t(i) <= 2
        thrust_now = 1.5 * m * g;           %Thrust is 50 percent more than weight to enable liftoff
    elseif t(i) <= 5
        thrust_now = 1.0 * m * g;           %Thrust is equal to weight to enable hover
    else
        thrust_now = 0.8 * m * g;           %Thrust is 20 percent less than weight to enable descent
    end

    drag = 0.5 * p * A * Cd * v(i)^2 * sign(v(i)); %realistic drag formula
    a(i) = (thrust_now - drone_weight_N - drag) / m;
    v(i+1) = v(i) + a(i) * dt;
    S(i+1) = S(i) + v(i) * dt;

```



C:\Users\dumj7\OneDrive\Desktop\Dronetomatlab\Real_drone_hover_Simulation.m

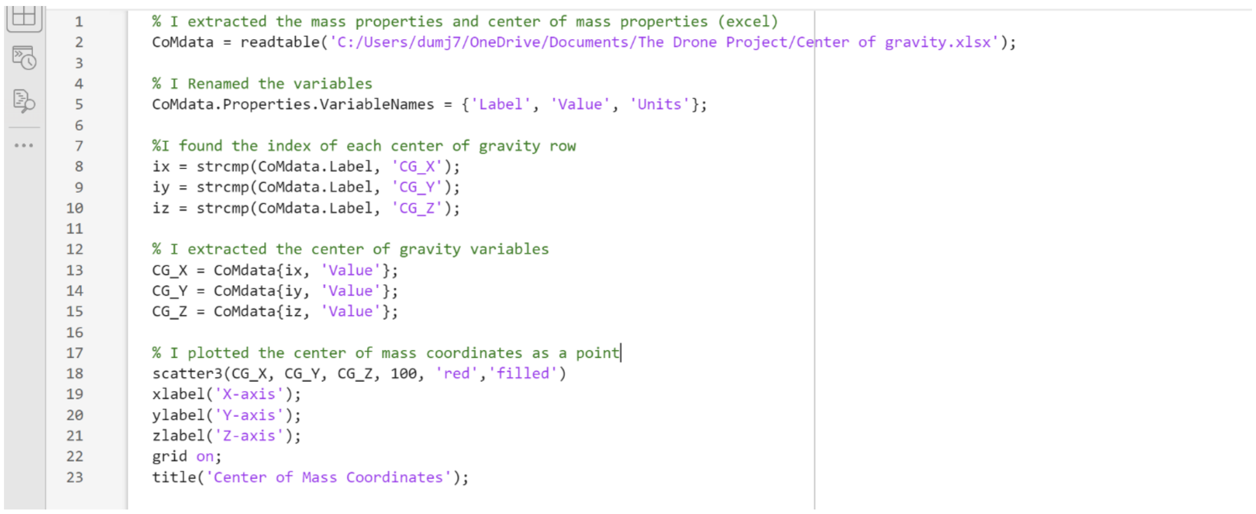
```matlab
% Set altitude values so the drone does not have a "negative" altitude
    if S(i+1) < 0
       S(i+1) = 0;
       v(i+1) = 0;
       a(i) = 0;
       last_i = i+1;
       break;
    end
end

% --- PLOT SETUP & LOAD STL MODEL ---
close all
figure
hold on
grid on

% Scale factor for drone size
scale = 0.08;

% Adjust axis limits to fit scaled drone model
s = 50;
axis([-s s -s/2 s -s s])
axis square
xlabel('X-direction')
ylabel('Y-direction')
zlabel('altitude')

plot3(0,0,0,'r*','MarkerSize',10) % origin marker

% Load the STL file
droneModel = stlread("quadcopter.STL");

faces = droneModel.ConnectivityList;
vertices = droneModel.Points;
```



```

% Scale the vertices
vertices = vertices * scale;

% Create patch object for the drone model
hPatch = patch('Faces', faces, 'Vertices', vertices, 'FaceColor', 'b');

% --- Prevent vertical stretching ---
axis equal    % keep the same scale on x, y, z
view(45,20)   % set a fixed 3D view

% 3-D animation
for i = 1:50:n
    % Translate the drone model vertices vertically by S(i)
    translatedVertices = vertices;
    translatedVertices(:,3) = vertices(:,3) + S(i);
    %plot the trail (path the drone flies upwards)
    plot3(zeros(1,i), zeros(1,i), S(1:i), 'b--', 'LineWidth', 1.5);

    % Update patch vertices to new position
    set(hPatch, 'Vertices', translatedVertices);

    % Set view and limits each frame
    xlim([-s s]);
    ylim([-s s]);
    zlim([0 max(S)+15]);
    title(sprintf('3D Drone Hover - Time elasped = %.2f sec', t(i)));

    drawnow;
    pause(0);
end

fprintf('Drone landed after %.2f seconds at altitude %.2f m\n', t(last_i), S(last_i));

```

Appendix B: MATLAB Code for Plotting Center of Mass Coordinates

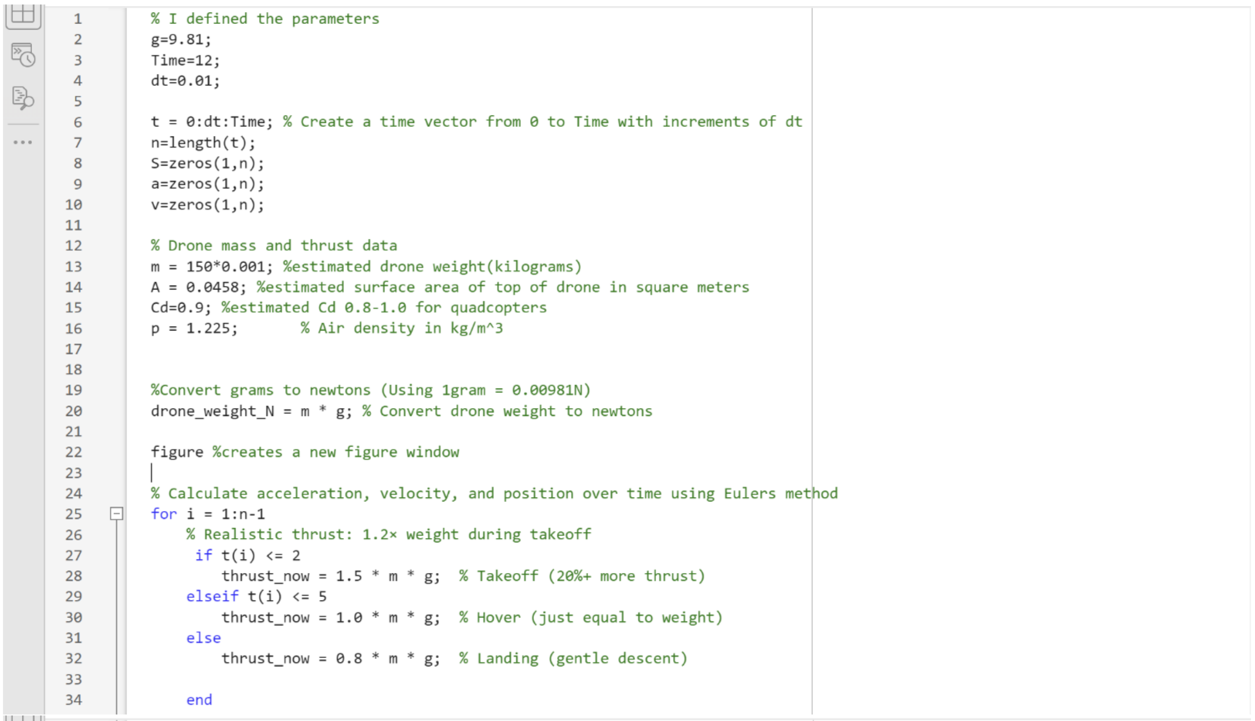
This script reads center of gravity data from an Excel file and plots the 3D coordinates of the drone's center of mass as a red point in space.



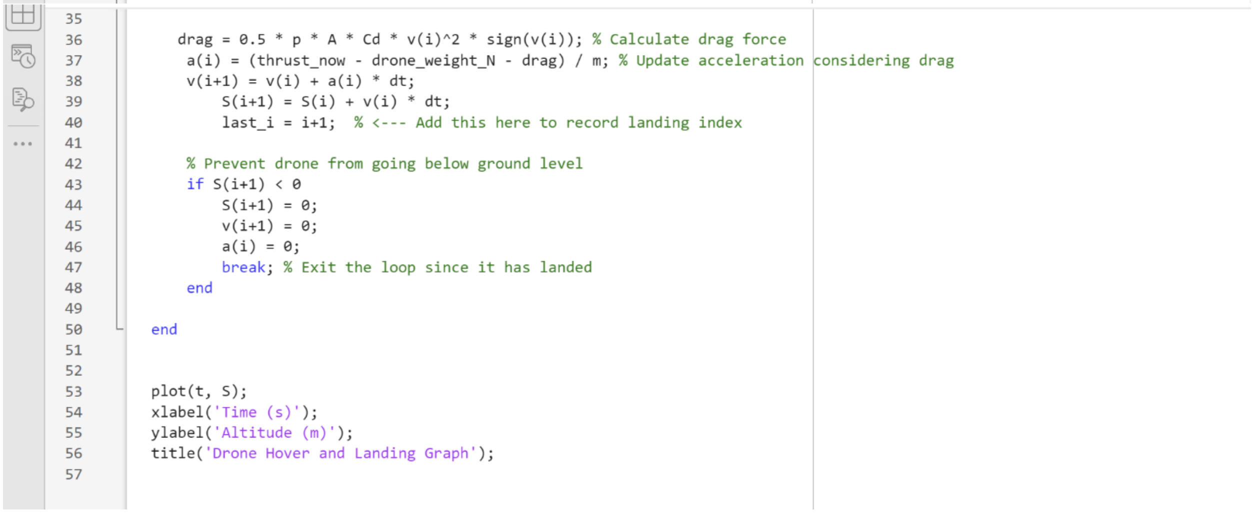
```matlab
% I extracted the mass properties and center of mass properties (excel)
CoMdata = readtable('C:/Users/dumj7/OneDrive/Documents/The Drone Project/Center of gravity.xlsx');

% I Renamed the variables
CoMdata.Properties.VariableNames = {'Label', 'Value', 'Units'};

%I found the index of each center of gravity row
ix = strcmp(CoMdata.Label, 'CG_X');
iy = strcmp(CoMdata.Label, 'CG_Y');
iz = strcmp(CoMdata.Label, 'CG_Z');

% I extracted the center of gravity variables
CG_X = CoMdata{ix, 'Value'};
CG_Y = CoMdata{iy, 'Value'};
CG_Z = CoMdata{iz, 'Value'};

% I plotted the center of mass coordinates as a point
scatter3(CG_X, CG_Y, CG_Z, 100, 'red','filled')
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
grid on;
title('Center of Mass Coordinates');
```

Appendix C: MATLAB Code for Altitude vs. Time Plot

This script generates the **2D altitude-time graph** of the drone’s vertical flight using Euler’s method, providing a simple visual of liftoff, hover, and landing.



```matlab
% I defined the parameters
g=9.81;
Time=12;
dt=0.01;

t = 0:dt:Time; % Create a time vector from 0 to Time with increments of dt
n=length(t);
S=zeros(1,n);
a=zeros(1,n);
v=zeros(1,n);

% Drone mass and thrust data
m = 150*0.001; %estimated drone weight(kilograms)
A = 0.0458; %estimated surface area of top of drone in square meters
Cd=0.9; %estimated Cd 0.8-1.0 for quadcopters
p = 1.225;       % Air density in kg/m^3


%Convert grams to newtons (Using 1gram = 0.00981N)
drone_weight_N = m * g; % Convert drone weight to newtons

figure %creates a new figure window

% Calculate acceleration, velocity, and position over time using Eulers method
for i = 1:n-1
    % Realistic thrust: 1.2× weight during takeoff
     if t(i) <= 2
        thrust_now = 1.5 * m * g;  % Takeoff (20%+ more thrust)
    elseif t(i) <= 5
        thrust_now = 1.0 * m * g;  % Hover (just equal to weight)
    else
        thrust_now = 0.8 * m * g;  % Landing (gentle descent)

    end
```



```matlab

   drag = 0.5 * p * A * Cd * v(i)^2 * sign(v(i)); % Calculate drag force
    a(i) = (thrust_now - drone_weight_N - drag) / m; % Update acceleration considering drag
    v(i+1) = v(i) + a(i) * dt;
        S(i+1) = S(i) + v(i) * dt;
        last_i = i+1;  % <--- Add this here to record landing index

    % Prevent drone from going below ground level
    if S(i+1) < 0
        S(i+1) = 0;
        v(i+1) = 0;
        a(i) = 0;
        break; % Exit the loop since it has landed
    end

end


plot(t, S);
xlabel('Time (s)');
ylabel('Altitude (m)');
title('Drone Hover and Landing Graph');

```