UNIVERSITY OF TECHNOLOGY SYDNEY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

A DIGITAL TWIN FOR DRONE CONTROL IN AN AUTONOMOUS DELIVERY SYSTEM

MATLAB SIMULINK PACKAGE USAGE INSTRUCTION

By

Hoang Trung Le

Student ID number: 13993807

Project number: AUT-22-07547 & SPR-22-09257

Major: Mechatronic Engineering

Supervisor: Associate Professor Quang Ha

A 12 Credit Point Capstone Project submitted in partial fulfilment of the requirement for the Degree of Bachelor of Engineering (Honours)

Sydney, November 2022

Introduction

This Matlab/Simulink package is the coding part for *A Digital Twin for Drone Control in an Autonomous Delivery System* thesis. The Matlab/Simulink results reported in the capstone are from the files in this package "Capstone_DeliveryQuadcopter_13993807". The code is still expected to be dynamic until the final submission of the capstone report according to the schedule of the University of Technology Sydney. For a **latest version** of the delivery quadcopter code, please refer to this GitHub repository at this link https://github.com/Hoang-Trung-Le/UTSCapstone_Quadcopter.

Further updates beyond the deadline can be considered but out of scope of this instruction document. However, the package will probably be transformed into an open source project in the future.

Matlab App

Matlab app is a powerful tool to create a Graphical User Interface, enhance the visualisation of the program and abstract the complicated coding logic for certain audience. Thus, it is employed as part of the coding section of the thesis.

Step 1: Initiation

To initiate the app, please navigate and select the file Quadcopter_App (Quadcopter_App.mlapp). The App Designer window will appear as Figure 1 and 2.

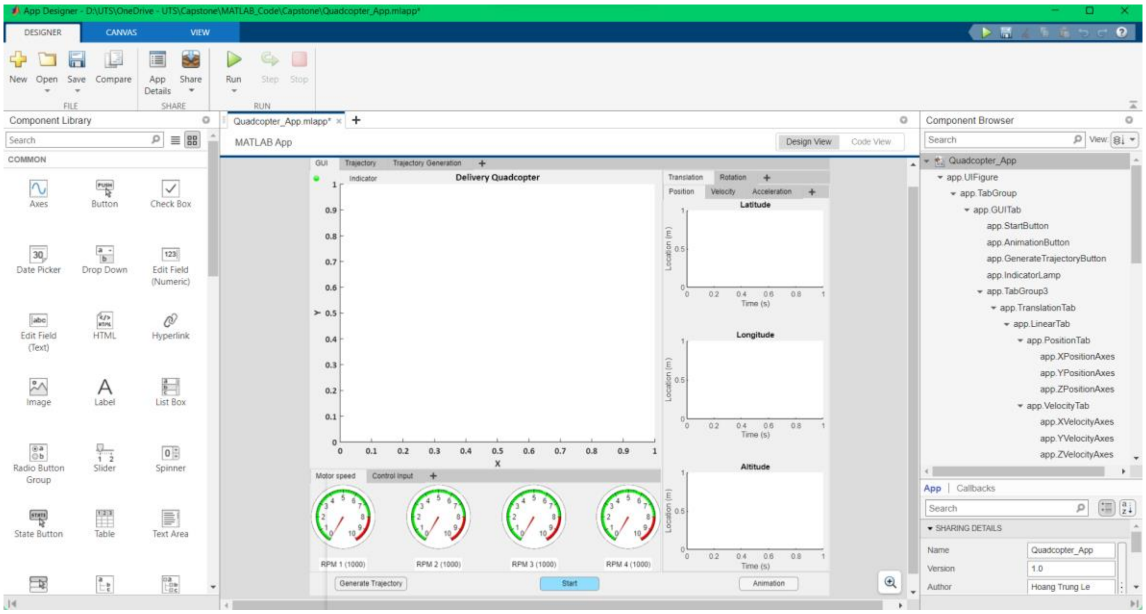

Figure 1. Design canvas of Quadcopter_App

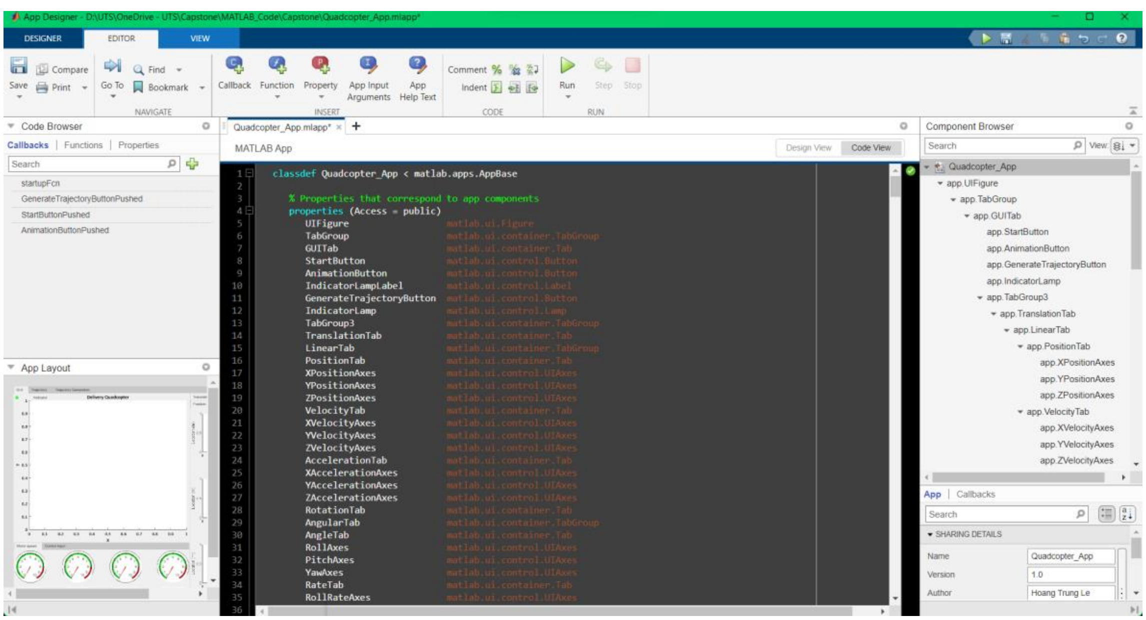

Figure 2. Code editor of Quadcopter_App

The App Designer tool will open the design canvas and the code editor. After pushing the run button, the main interface of the Delivery Quadcopter Application will appear similar to Figure 3. Internally, it includes the Support folder in the package into the app's execution path. Moreover, a quadcopter object is created from Quadcopter class (Matlab file Quadcopter.m) and its 3D model (CloverAssemblyP.ply) is imported into the plot from the CAD subfolder in the Support folder. The scenario of Wentworth Park area is generated with the geographic terrain from Matlab built-in database and an open source OpenStreetMap file containing 3D building data (mapWPark.osm file). The indicator light is now red to represent standing-by state of the app.

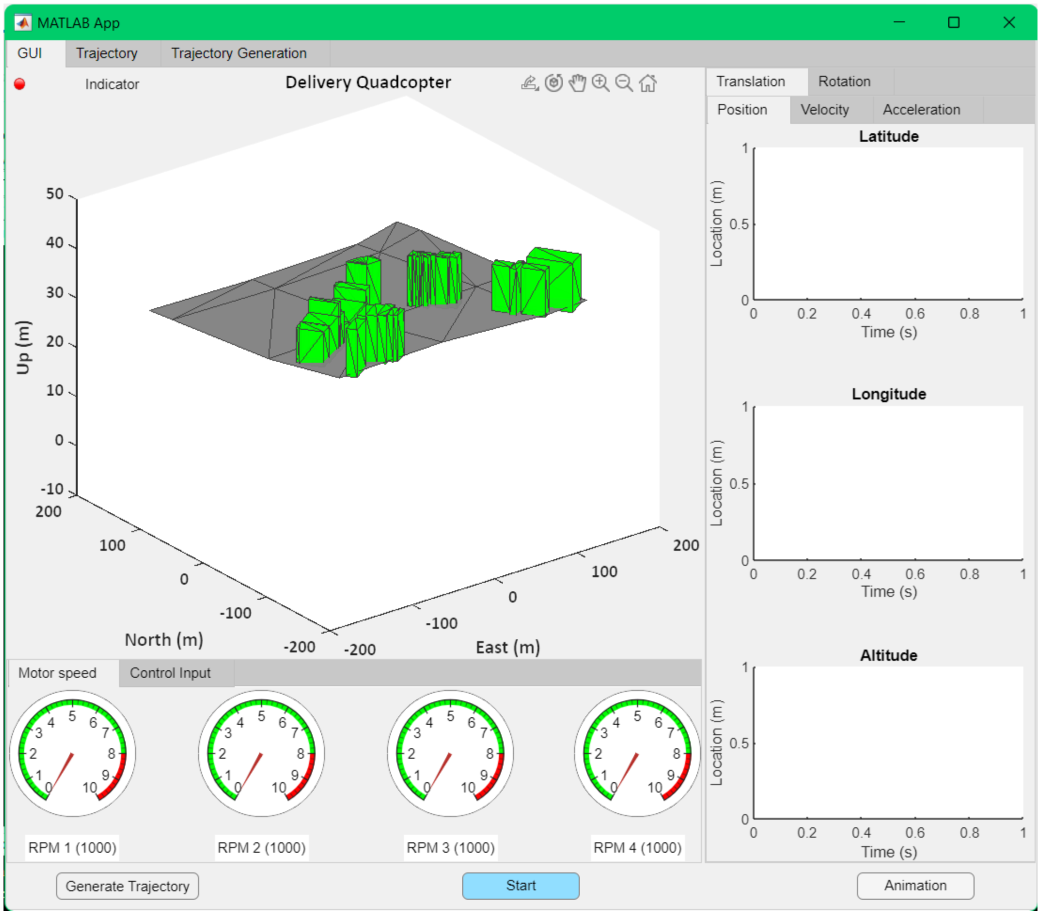

Figure 3. Main interface of Delivery Quadcopter App

Step 2: Trajectory generation

The next step is to generate the trajectory of the delivery quadcopter by pushing the button with that name. The font of the button title changes to bold, its background color turns light blue and the indicator lamp switches to bright green mean the quadcopter receives its delivery path. Please navigate to the trajectory tab for a graphical visualisation of the desired motion profile of the quadcopter.

Figure 4. Trajectory generated state

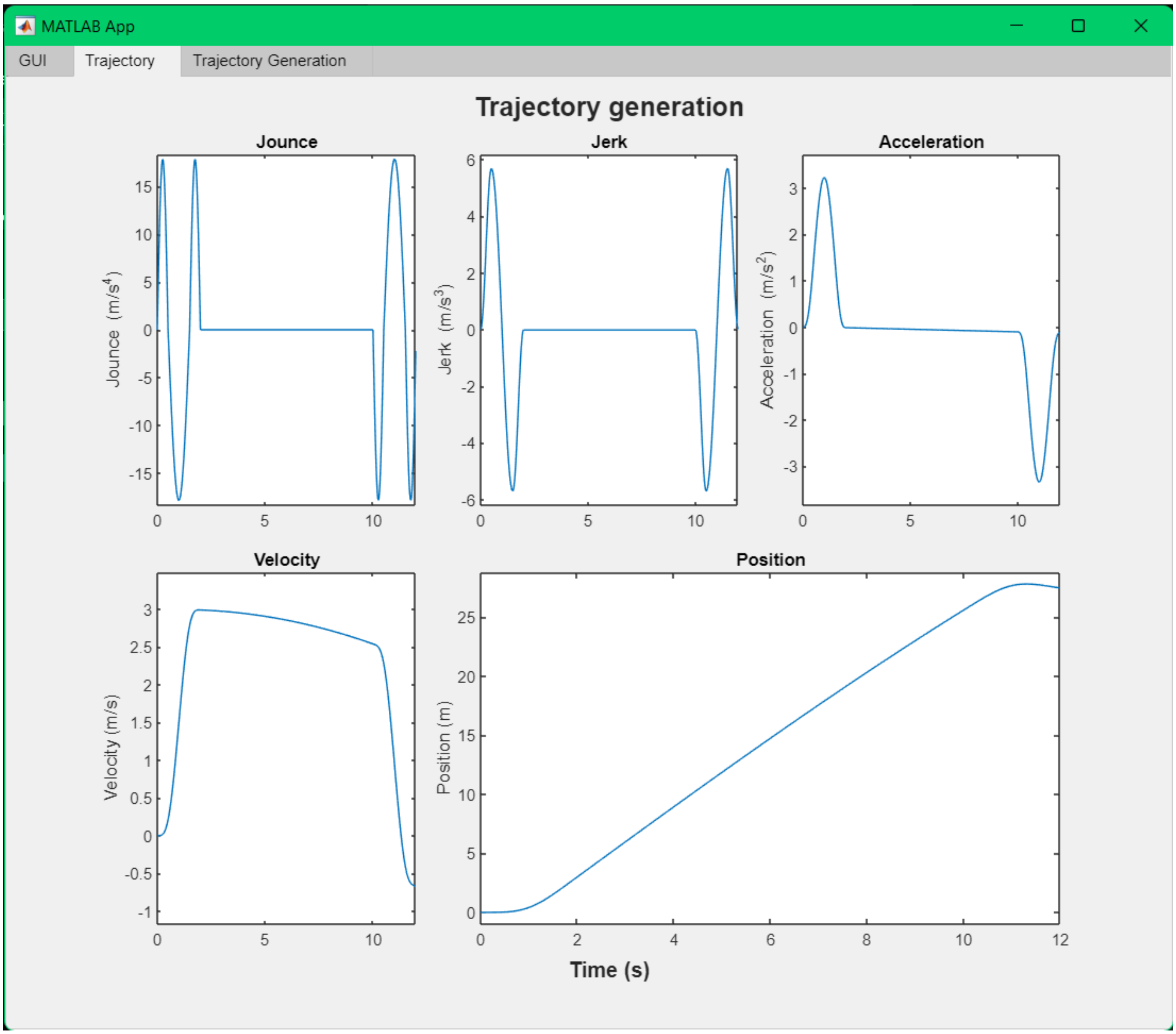

Figure 5. Trajectory generated tab

Step 3: Calculation

By pushing the Start button in the GUI tab, the app enters a new stage where the control inputs of the quadcopter are computed based on the desired commands and the vehicle's actual dynamic responses. The indicator lamp will flash red with the message "Started computation" while it is running necessary calculations. This step requires most of the computational resources so it would take time depending on the performance of host computer. The lamp turning green with the message "Virtually simulated" indicates calculations are finished.

Figure 6. Virtually simulated state

Step 4: Animation

By pressing the Animation button, the quadcopter will manoeuvre as calculated from the previous step. The indicator lamp turns into a shade of dark blue. The quadcopter leaves behind a red-x trail to mark its past position. The gauges below illustrate the four motors' speed in rpm in acordance with the current position of the quadcopter. The control input tab displays the roll-pitch-yaw torques and resultant thrust. The translation and rotation tab demonstrate the position/angle, velocity and acceleration values of each component. Please navigate between them to better visualise the dynamic response of the quadcopter.

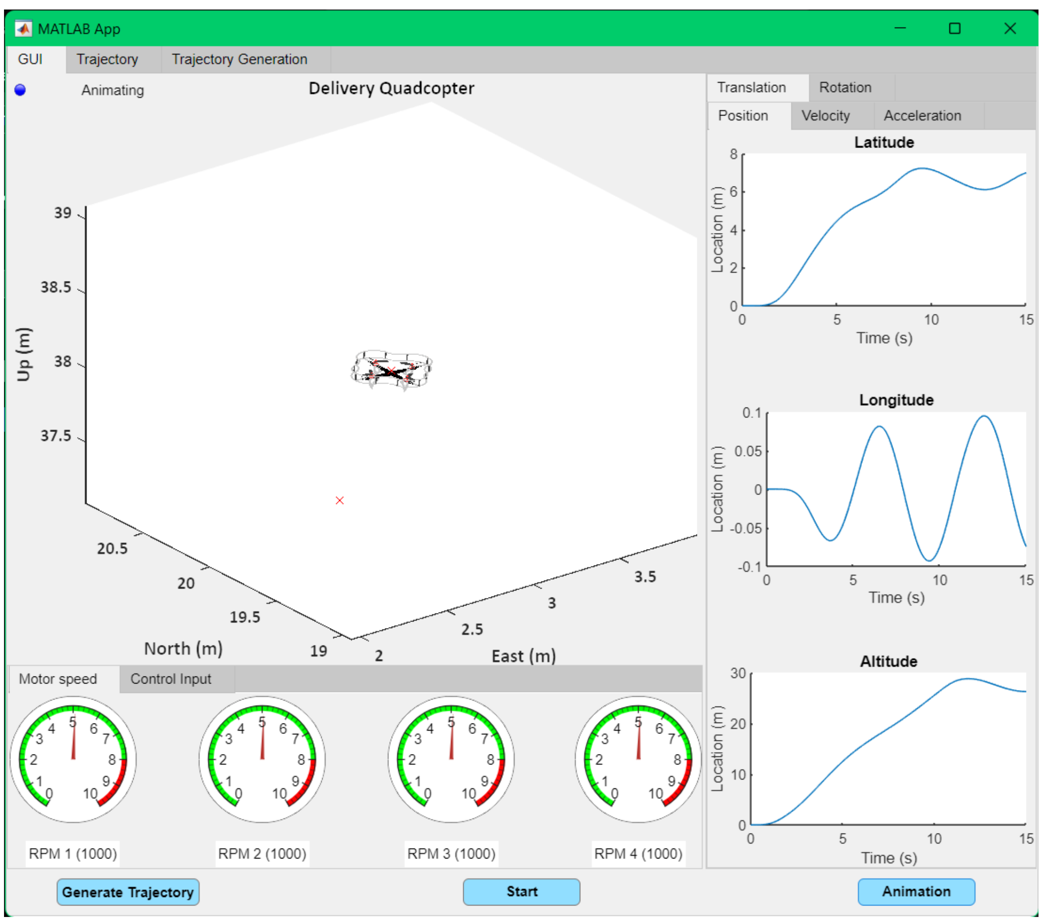

Figure 7. The animation state of Delivery Quadcopter app

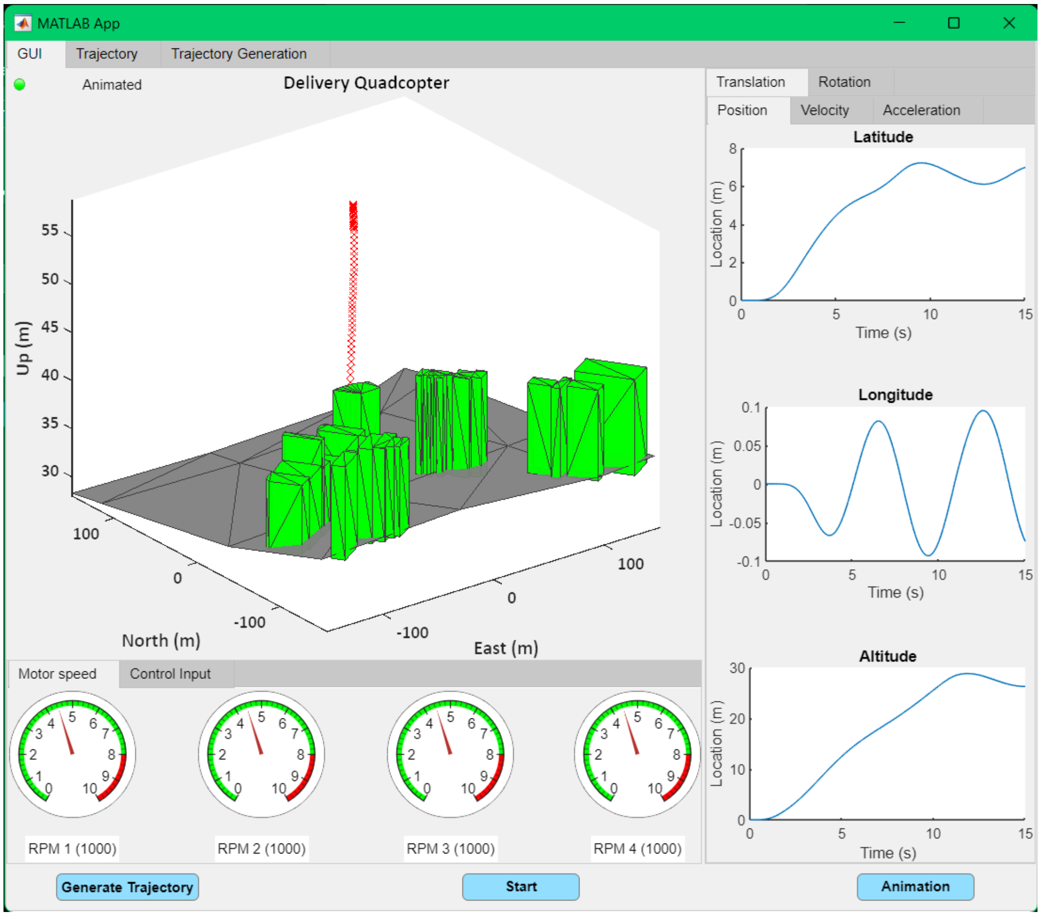

Figure 8. Final outcome of the animation

After the animation is finished, the indicator lamp turns green again and shows the final message "Animated". The main plot of the app zooms out for the user to visualise the entirety of the quadcopter trajectory in the virtual Wentworth Park environment.

Simulink

A realistic simulation of the Clover quadcopter performing a complete delivery mission is conducted in the powerful Simscape Multibody of Simulink. The surrounding environment attempts to replicate that of Wentworth Park and the Viaduct (a supporting infrastructure for the Sydney Inner West Light rail).

To open the Simulink simulation, please navigate to the to the Delivery_Quadcopter folder and select Delivery_Quadcopter.slx. The initial Simulink canvas should resemble Figure 9.

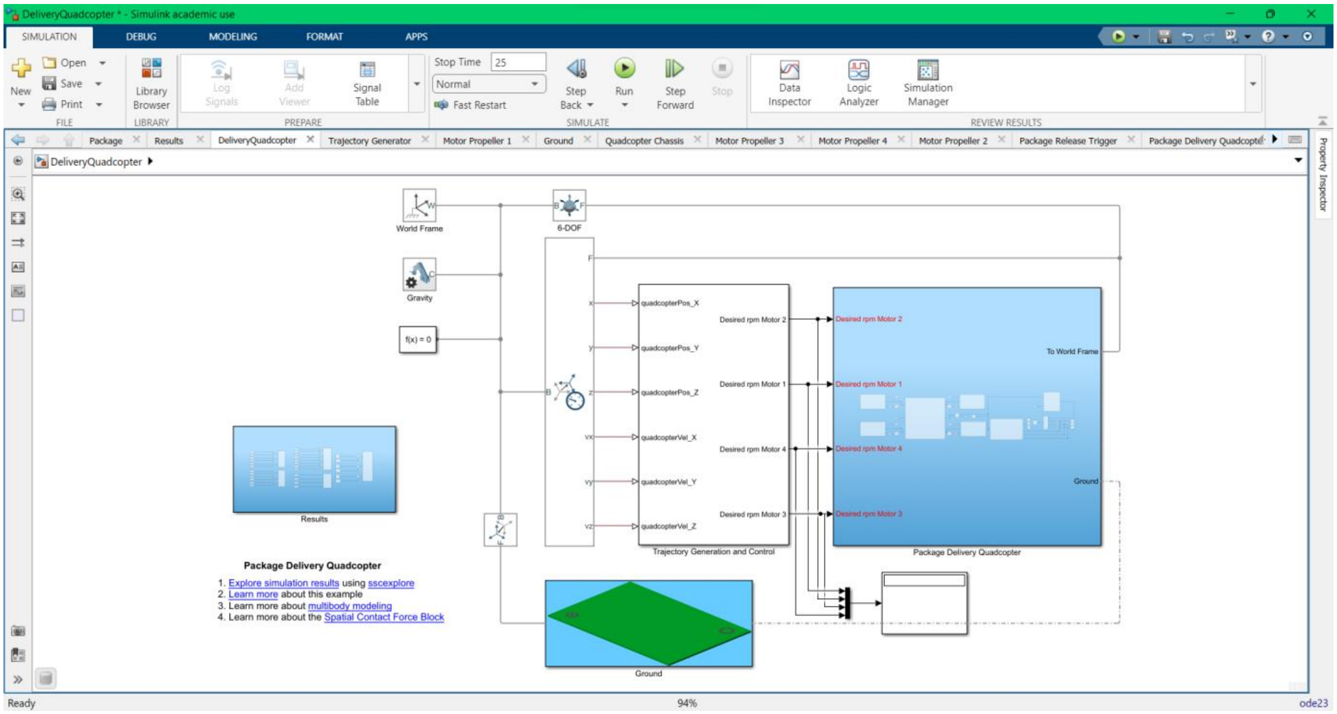

Figure 9. Simulink canvas of Delivery Quadcopter

When the Run button is pressed, the Mechanics Explorers will appear in Matlab, next to the usual Matlab editor. The simulation is similar to the attached GIF in Figure 10.

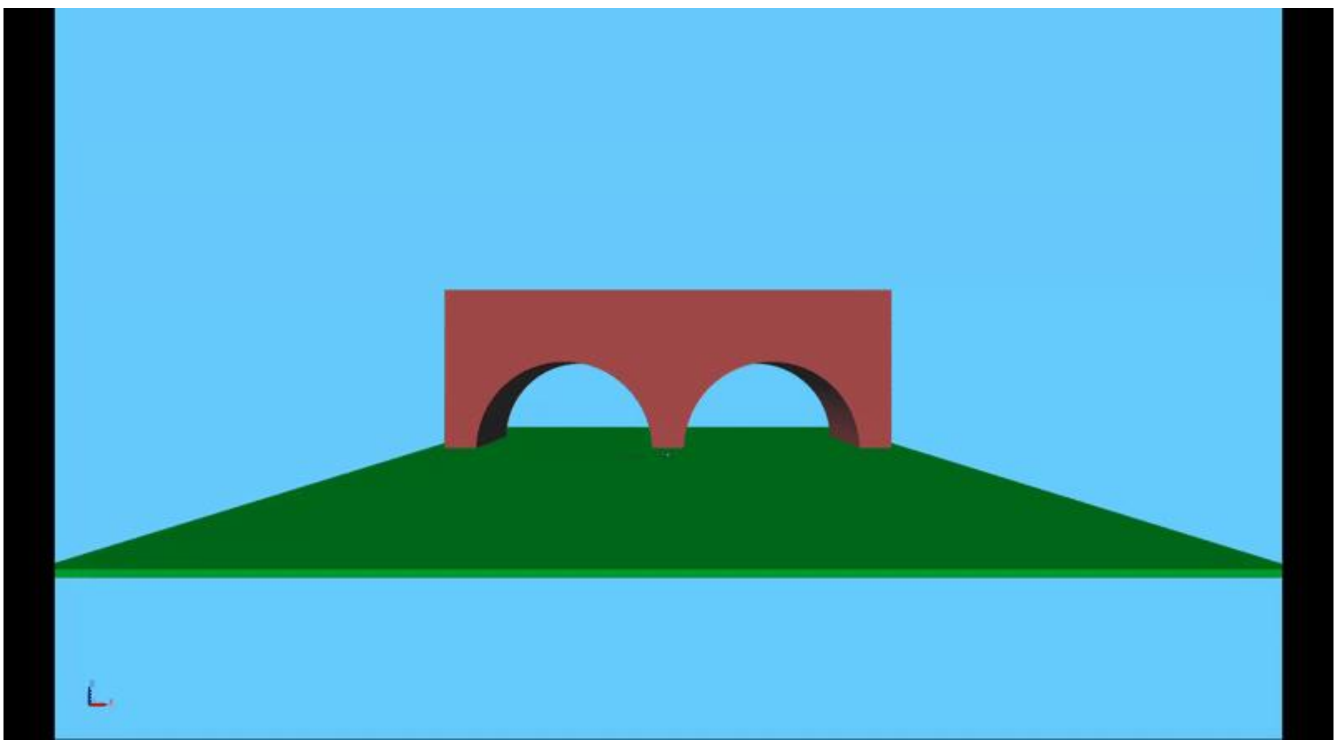
Figure 10. Simulink simulation

To visualise the simulation results, please navigate to the Results block and select the desired Scope(s). Names of the scopes clearly represent their functionalities.

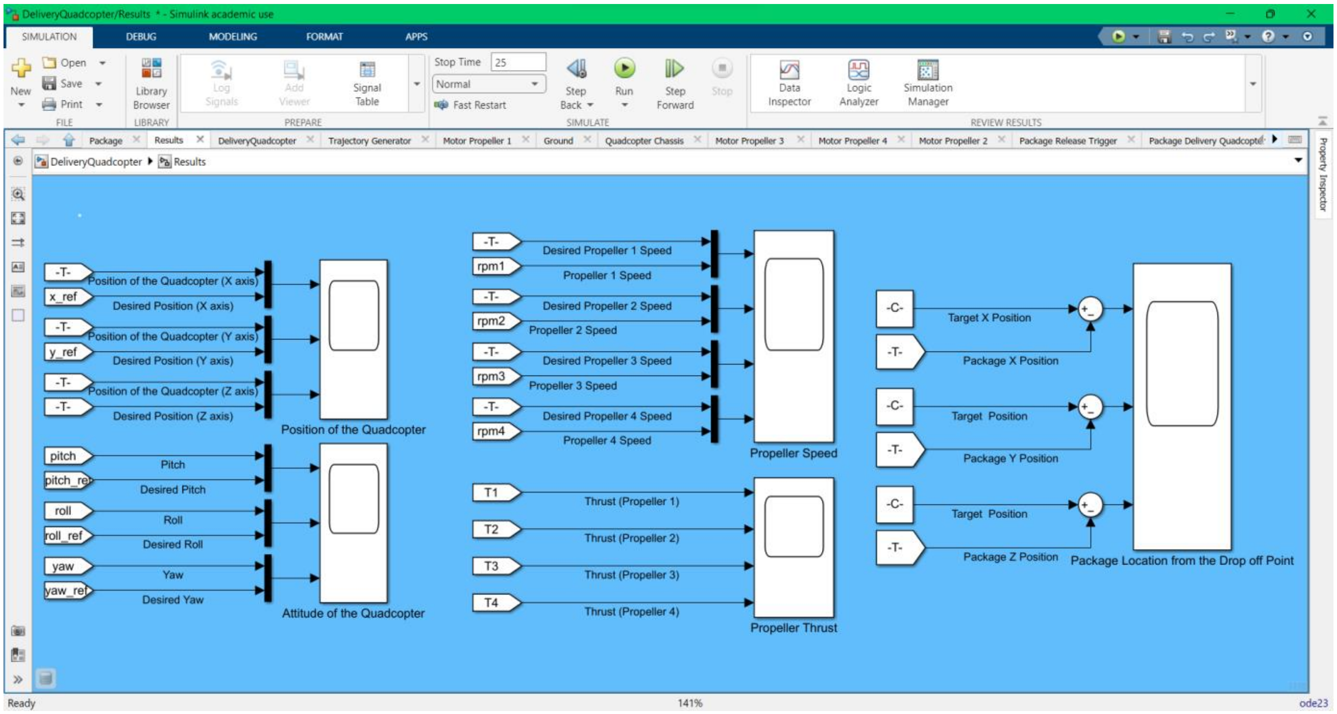

Figure 11. Result canvas to explore dynamic characteristics of Delivery Quadcopter

The file QuadcopterParameters.m initiates the required variables and the desired package drop-off location of the Simulink simulation. The Simulink model has an internal workspace, which can be located in the database icon in the bottom left corner of the canvas. It is currently loaded with parameters from the Matlab script. Please adjust the file path in accordance with your computer's directory.

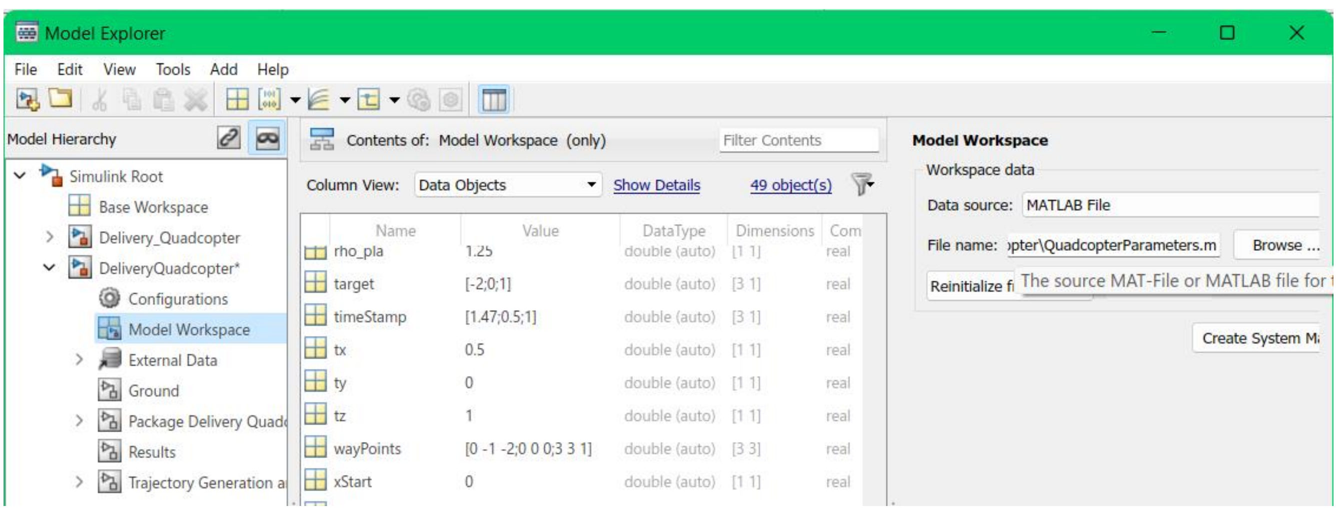

Figure 12. Model workspace of Simulink

At the moment, either the parameters are constant or finely tuned to this specific simulation scenario. It is not advised to change the parameters and variables in this file. Further updates of Simulink to enhance the versability and adaptability of the model to other virtual test scenarios will be considered.