Drones in Wind

Nico Alba, Isabel Anderson, Emilio Gordon, Michael Gray

July, 2017

1 Goal

The "Drones in Wind" simulation simulates the flight of a quadcopter drone. The drone in the simulation will be modeled after the AscTec Pelican, a research-drone by the German company Ascending Technologies. The simulated drone as of the point of writing this paper is equipped with sensors to measure the velocity in the horizontal and vertical axis, the vertical position and the pitch angle.

The goal is to create a controller that linearizes about a trajectory given by the third-party program OptimTraj. The trajectory comes from a cost function that minimizes the time integral of error from a desired position. A requirement for this model is established such that the time of multiple completed simulations will have a coefficient of variation less than one. Since the simulation should be of an optimized trajectory, minimal variations should be expected. With this requirement in mind, a procedure for verification is established. Verification ensures that the system meets the established requirements. Verification will be conducted through several methodologies and are as follows

- Data will be acquired for up to 100 flight simulations. This will be done by implementing the script shown in Figure 1.
- Using Matlab, a histogram can be created in order to visualize the frequency of the drones flight times.
- Using Matlab, the coefficient of variation will be calculated can be computed for the acquired data.

```
% Number of flights
nFlights = 100;
% Loop over each flight
for i=1:nFlights
    % Run simulation without graphics and save data
    DesignProblem0('Controller','datafile','data.mat','display',false);
    % Load data
    load('data.mat');
    % Get t and x
    t = processdata.t
    x = processdata.x
    % DOUBLE CHECK THE CODE.
end
```

Figure 1: Flight Generation Code

2 Model

The state for the system is defined as

$$x = \begin{bmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \end{bmatrix} \tag{1}$$

Where x, z, correspond to horizontal and vertical position, respectively, and θ corresponds to the angle of the quad from upright.

The inputs for the system are defined as

$$u = \begin{bmatrix} \omega \\ f \end{bmatrix} \tag{2}$$

Where ω corresponds to angular pitch rate and f corresponds to thrust. Taking the jacobian lends the following A and B matrices

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{f\cos\theta}{m} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{-f\sin\theta}{m} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & \frac{\sin\theta}{m} \\ 0 & 0 \\ 0 & \frac{\cos\theta}{m} \\ 1 & 0 \end{bmatrix} \tag{4}$$

We now substitute in the equilibrium force, $f = mg$, lending the following matrices

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g\cos\theta \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -g\sin\theta \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & \frac{\sin\theta}{m} \\ 0 & 0 \\ 0 & \frac{\cos\theta}{m} \\ 1 & 0 \end{bmatrix} \tag{6}$$

3 Controller and Optimization

3.1 Controller

The controller is designed to use in the loop control with LQR to follow a given trajectory.

First, the init function creates a symbolic description of the A and B matrices, in which the angle θ is the only variable. These symbolic descriptions are then turned into MATLAB functions and saved in the data struct. The Q and R weight matrices are then initialized. The init function proceeds to load in the trajectory data and also adds it to the data struct. Finally, some parameters used to determine when the quad has reached its destination are initialized.

Logic is used at the start of the control loop to determine if there is still trajectory data available for the current timestep. If there is not, the controller will linearize about the last available trajectory point. This case does not happen in current simulations because the trajectory is planned for the entire duration of the simulation.

After determining that the simulation is still within the regime in which a trajectory is available, the controller will find the respective trajectory point to linearize about for the current timestep. The angle for this trajectory point is passed into the MATLAB functions to determine the relevant A and B matrices, which are then used in conjunction with the already initialized weights to create a gain matrix using LQR.

Finally, the input $u = -Kx$ is applied in which u is a column of two inputs, the first being thrust and the second being pitch rate. Note here that the thrust is augmented by the equilibrium condition for hover, $f = mg$. K is the previously specified gain matrix, and x is the difference between the current state and the trajectory state at the current time.

3.2 OptimTraj

4 Analysis

5 Future Work