AE353: Design Problem 03

Parthiv Kukadia

November 8, 2020

This paper describes the design, implementation, and testing of a controller for an unpowered glider. The controller was run through a simulator 1000 times, in order to collect flight data.

1 Goal

The code provided in DesignProblem03 simulates an unpowered glider. The glider in question has one control surface; an elevator. There is an actuator, that allows me to specify my angular rate for this elevator. There are also sensors that measure both pitch angle of the glider and relative angle of the elevator. The goal of this design problem is to make the glider fly as long a distance as possible, if it is released from a height of about two meters, with a forward speed of 6 meters per second, and a pitch angle and elevator length of my choice.

2 Requirements and Verification

Given the above specifications, we know that I am unable to control the elevator angle, however, I am able to control the elevator angular rate (ϕ). I will attempt to design an observer that will allow my glider to fly an average of 10 meters. After a 1000 test simulations, we will verify if this requirement was met through cross checking the mean and median of my 1,000 simulations, to make sure that they are above 10 m. Therefore, my requirements can be broken down as shown below;

1. Glide the unpowered glider an average of 10 m., which will be verified through summing the final distances of 1,000 trials, and finding it's average.
2. Obtain a standard deviation of ≤ 7. (random value)

3 Model

The motion of the glider is governed by ordinary differential equations with the form

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = f(\theta, \phi, \dot{x}, \dot{y}, \dot{\theta}, \dot{\phi}) \tag{1}$$

where θ is the pitch angle, ϕ is the elevator angle, $\dot{x}$ is the forward speed, $\dot{y}$ is the vertical speed, $\dot{\theta}$ is the pitch angular rate, and $\dot{\phi}$ is the elevator angular rate (which an actuator allows you to specify). You might be interested to know that these equations were derived by applying a flat-plate model of lift c_L and drag c_D as a function of angle of attack α, for both the wing and elevator:

$$c_L = 2\sin\alpha\cos\alpha \qquad\qquad c_D = 2\sin^2\alpha$$

In our case, we want to maximize our lift to drag ratio to achieve maximum glide, given as:

$$\frac{c_L}{c_D} = \frac{2\sin\alpha\cos\alpha}{2\sin^2\alpha} = \frac{1}{\tan\alpha} \tag{2}$$

This can be maximized when the ratio reaches infinity, which is when the equilibrium elevator angle is 0.

To linearize the non-linear model give to us, we must linearize it about an equilibrium point to provide it in a linear state space model, which is represented by two equations:

$$\dot{x} = Ax + Bu \qquad\qquad y = Cx + Du \tag{3}$$

Therefore to begin linearizing the non-linear equations of motion, we chose an equilibrium point. The function f in Eq. (1) was run through DesignProblem03 and stored in a data.mat file symbolically as $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$. The state variables of interest, input vector, and output vector are defined from the variables of the parsed EOMs, and an initial guess was made about the values of these variables to determine the equilibrium points.

$$x = \begin{bmatrix} \theta - \theta_e \\ \phi - \phi_e \\ \dot{x} - \dot{x}_e \\ \dot{y} - \dot{y}_e \\ \dot{\theta} - \dot{\theta}_e \end{bmatrix} \qquad u = \begin{bmatrix} \dot{\phi} - \dot{\phi}_e \end{bmatrix} \qquad y = \begin{bmatrix} \theta - \theta_e \\ \phi - \phi_e \end{bmatrix} \qquad x_{\text{guess}} = \begin{bmatrix} \theta_{\text{guess}} \\ \phi_{\text{guess}} \\ \dot{x}_{\text{guess}} \\ \dot{y}_{\text{guess}} \\ \dot{\theta}_{\text{guess}} \\ \dot{\phi}_{\text{guess}} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0 \\ 7 \\ -0.1 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

The guesses for equilibrium values in Eq.((4)) were such that there would be no pitch or elevator change ($\dot{\theta}_{guess}$, $\dot{\phi}_{guess} = 0$). I made θ_{guess} small and set ϕ_{guess} to 0. In order to maximize horizontal flight, I set $\dot{x}_{guess}$ and $\dot{y}_{guess}$ as the values shown above. From these guessed equilibrium values, I obtained the following equilibrium values through fsolve:

$$\begin{bmatrix} \theta_e \\ \phi_e \\ \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} 0.0039 \\ -0.0624 \\ 7.0149 \\ -0.4788 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

The values that I obtained in Eq.((5)) were used in the Jacobian function to produce the state space matrices of A,B,C, and D. The matrix A was computed by taking the partial derivative of f with respect to (w.r.t) x. B was computed the partial of f w.r.t u, C from the partial of y w.r.t x, and D from the partial of y w.r.t. u.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -6.5711 & 3.1236 & -0.0314 & -0.4596 & 0.2411 \\ 191.1751 & 63.7969 & 0.9324 & -27.3165 & 3.5291 \\ -207.2970 & -239.2626 & 2.0077 & 29.4141 & -15.4288 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \\ 0.0532 \\ 0.9070 \\ -3.4018 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \end{bmatrix}$$

This provides us with the following state space equations: (**??**)

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -6.5711 & 3.1236 & -0.0314 & -0.4596 & 0.2411 \\ 191.1751 & 63.7969 & 0.9324 & -27.3165 & 3.5291 \\ -207.2970 & -239.2626 & 2.0077 & 29.4141 & -15.4288 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0.0532 \\ 0.9070 \\ -3.4018 \end{bmatrix} \begin{bmatrix} \dot{\phi} \end{bmatrix}$$

and

$$y = \begin{bmatrix} \theta \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \end{bmatrix} \tag{6}$$

Now that we have the state space matrices, we can check for controllability, which is given from the matrix W. A system is controllable when the rank of matrix W = the length of matrix A, which in this case, it does, and it = 5. Length(A) = rank(W) = 5.

$$W = ctrb(A, B) = \begin{bmatrix} 0 & -3.4018 & -159.99 & 3973.598 & -85632.48 \\ 1 & 0 & 0 & 0 & 0 \\ 0.0532 & 1.885 & -28.71 & 2907.50 & -63733.7 \\ 0.907 & 27.066 & -1952.57 & 36747.26 & -543652.35 \\ -3.402 & -159.99 & 3973.60 & . - 85632.48 & 1584217.71 \end{bmatrix} \tag{7}$$

Despite the system being controllable, it's asymptotically unstable. This means that the real eigenvalues of A are not all negative. Since that is the case, it is highly likely that the glider will either stall or take a nose-dive. Therefore, we must design a controller that will allow us to improve our gliding distance.

$$eig(A) = \begin{bmatrix} -21.3947 + 8.2165i \\ -21.3947 - 8.2165i \\ 0.00634 + 1.2396i \\ 0.00634 - 1.2396i \\ 0 \end{bmatrix}$$

We can go on to check if our system is observable or not now. This is done using the MATLAB function obsv(A,C), and we compare the rank of this O matrix with length of the C matrix, which are both 5. Therefore we can say that our system is observable. Length(C) = rank(O) = 5.

$$O = obsv(A, C) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -207.297 & -239.26 & 2.008 & 29.41 & -15.43 \\ 0 & 0 & 0 & 0 & 0 \\ 8808.40 & 5574.34 & -3.61 & -1258.24 & 135.04 \\ 0 & 0 & 0 & 0 & 0 \\ -268513.77 & . - 112593.45 & -901.93 & 38344.46 & 2283.48 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

4 Control Design

To design the controller, the system needs to be controllable. We need to check for controllability as it allows system to move via the input from an intial state to the final state. We found controllability above. Therefore, we know go on to find the controller gain matrix, K, which can be found using the lqr function in MATLAB, using state cost weighted matrix and input cost weighted matrix.

$$K = lqr(A, B, Q_c, R_c) \tag{9}$$

The cost weighted matrices used were;

$$Q_c = 1.8 \begin{bmatrix} 200 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 150 \end{bmatrix}, R_c = 0.1 \begin{bmatrix} 1 \end{bmatrix}$$

This outputted a K matrix of;

$$K = \begin{bmatrix} -115.1978 & 57.1724 & 12.7627 & -6.6945 & -48.8563 \end{bmatrix} \tag{10}$$

In our case, as we don't have complete knowledge of the state, a new ODE is formed, where we require L as shown below:

$$\hat{\dot{x}} = A\hat{x} + Bu - L(C\hat{x} - y) \tag{11}$$

Where L, the observer gain matrix is formed using:

$$L = lqr(A', C', inv(R_o), inv(Q_o)) \tag{12}$$

With the observer matrices being:

$$R_o = 0.2 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, Q_o = 0.8 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Outputting a L matrix of:

$$L = \begin{bmatrix} 2.4605 & -0.4200 \\ -0.4200 & 0.2713 \\ -4.5172 & 7.6708 \\ 15.6679 & -1.9323 \\ 2.9902 & -1.1474 \end{bmatrix} \tag{13}$$

We then go on to make 2 final checks regarding the asymptotic stability of the system. If they have all negative real parts, then the system is stable.

$$eig(A - BK) = \begin{bmatrix} -160.55 + 0.0000i \\ -77.65 + 0.0000i \\ -19.45 + 0.0000i \\ -1.55 + 0.0036i \\ -1.55 - 0.0036i \end{bmatrix} \qquad eig(A - CL) = \begin{bmatrix} -21.3979 + 8.2201i \\ -21.3979 - 8.2201i \\ -1.2729 + 1.7319i \\ -1.2729 - 1.7319i \\ -0.1668 + 0.0000i \end{bmatrix} \tag{14}$$

For reference tracking, we choose a Kref using:

$$Kref = \frac{1}{-C(A - BK)^{-1}B} \tag{15}$$

Providing me a Kref value of:

$$\begin{bmatrix} 0 & -761.1921 \end{bmatrix} \tag{16}$$

I then chose a reference of r:

$$r = \begin{bmatrix} \frac{sensors.theta}{20} \\ 0 \end{bmatrix} \tag{17}$$

So that we have a positive angle of attack, and we are able to control $\dot{\phi}$ to lower it back down to an equilibrium value of 0.

5 Implementation and Testing of Controller

To test that my controller verified my requirements, I ran it 1,000 times, and took the average distance it travelled over the 1,000 trials to obtain my mean. I also found the median, the max distance it travelled, and the standard deviation. I created a for loop to record the data of each flight and plot it during each simulation.

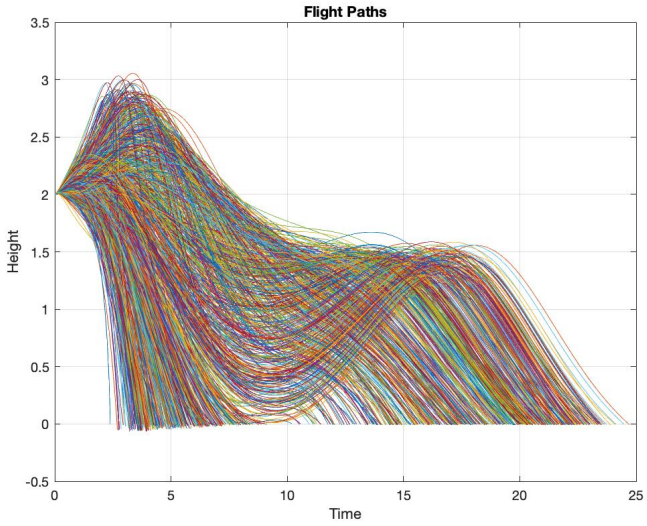

Figure 1: Flight Paths

The mean, median, maximum, and standard deviation are shown below:

$$\begin{aligned} x_{max} &= max(x) = 24.7039m \\ \bar{x} &= mean(x) = 13.4204m \\ x_{median} &= median(x) = 15.4316m \\ \sigma &= std(x) = 6.95 \end{aligned}$$

Plotting a histogram of my results, shows:

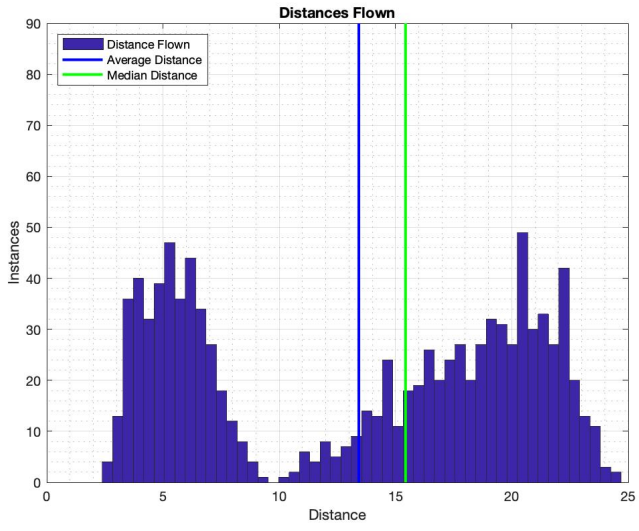

Figure 2: Flight Distances' Histogram

We can see that our controller was successful and verified the requirements set for it as it averaged a flight distance of 13.42 meters. Since my standard deviation is 6.95 we can say that around 68% of the time, the glider would land in between 10 to 20 meters.
If I eliminate all the nose dives done by my glider, the mean, median, and standard deviation would change. These are the main instances where my initial state guesses were off, not allowing m controller time to stabilize in time. The histogram then looks like:

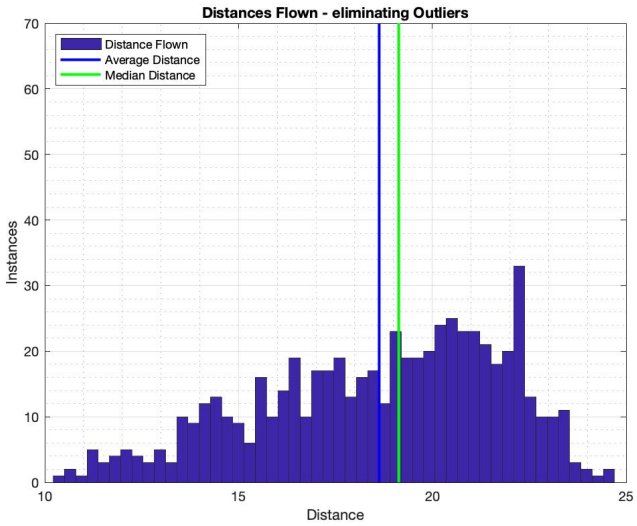

Figure 3: Flight Distances without Nose-Dives' Histogram

The new mean, median, maximum, and standard deviation are shown below:

$$\begin{aligned} x_{new,max} &= max(x) = 24.7039m \\ \bar{x}_{new} &= mean(x) = 18.6288m \\ x_{new,median} &= median(x) = 19.1350m \\ \sigma_{new} &= std(x) = 3.1320 \end{aligned}$$

6 Discussion of Results

After playing around with $Q_c, R_c, Q_o, and R_o$, I was able to get a relatively consistent system. I was mostly able to remove most of the nose dives that produced a binomial distribution. However, my average was about 13.42 meters. The launch angle that worked best with my controller was 10°, and the elevator length that worked best was 0.2 meters. After a lot of trial and error, I received an output that I was satisfied with. However, this showed that my controller was unable to correct itself enough, showing how my initial guesses were off. This is why we see the instances of a normal distribution regarding the low values, and this is why my standard deviation was high, which I expected. Tough I was able to improve the nature of the consistency of the glide distance, I was unable to produce a flight distance

that was further than the record in this class. The outputs showed the limitations to my controller, and if more time was spent on playing around with observer matrices and cost matrices, I could have produced even better results. However, that meant trying millions of combinations, which would have been manually impossible. Therefore, if I were to improve this next time, I would try to build an optimizing function, that would allow me to compute for the best observer and cost matrices, to efficiently glide my glider, and produce the best output.

7 Acknowledgements

[1] Melkior Ornik: Provided the DesignProblem03 files, from which the code was extracted to find the equations of motions, equilibrium values, and run the controller.

[2] Pranay Thangeda: Provided feedback on my thought process around finding my K and L matrices, and guided me around how to play around with Q_c, R_c, Q_o, R_o.

[3] Ben Schultz: Helped me out with general questions, and was a great work buddy. He motivated me to stay on task, and made sure I was grinding through. Would not have been able to sit through all those trial runs whilst playing around with my observer and cost matrices without him.