

AE353: Design Problem 03

T. Bretl (**this is my name, not your name!**)

March 13, 2019

1 Goal

The code provided in **DesignProblem03** simulates an unpowered glider that is similar to one used for experiments on fixed-wing perching [1, 2]. This glider has one control surface, an elevator. An actuator allows you to specify the angular rate of this elevator. Sensors allow you to measure both the pitch angle of the glider and the relative angle of the elevator. The goal is to make the glider fly as long a distance as possible, if released at a height of about two meters with a forward speed of about six meters per second and with a pitch angle of your choice.

2 Model

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

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

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:

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

Experimental results show that this flat-plate model is a good approximation for the glider and that it remains accurate at high angles of attack and even post-stall [2]. The function f in (1) depends on a number of parameters (e.g., mass, moment of inertia, surface area of the wing). You can access symbolic descriptions of this function either within your controller code (**Controller.m**) as `parameters.symEOM.f`, or in a separate piece of code that you write for the purpose of control design. In particular, the first time you run **DesignProblem03**, it will create a file that you can load to access the equations of motion. Here is an example:

```

1 % Load the equations of motion.
2 load('DesignProblem03_EOMs.mat');
3 % Parse the equations of motion.
4 f = symEOM.f;
5 % Define symbolic variables that appear in the equations of motion.
6 syms theta phi xdot ydot thetadot phidot
7 % Proceed to linearize or do whatever else you like...
8 %     (see: help sym/jacobian, help sym/diff, help sym/subs, etc.)

```

3 Tasks

3.1 Analysis

The focus of your analysis this time will be on data collection. The initial conditions in the simulation are random. So, for a given control design, the flight distance will vary (perhaps significantly). It is not enough to look at the results of one flight—you will have to look at the results of many flights. At minimum, for each design and for each launch angle that you consider, you should do the following:

- Collect data from at least 1000 flights.
- Compute the minimum, maximum, median, mean, and standard deviation of the flight distance (e.g., `min`, `max`, `median`, `mean`, and `std` in MATLAB).
- Plot a histogram of the flight distance (e.g., `hist` in MATLAB).

Remember that `DesignProblem03` provides options to save data and to turn the graphics off, both of which are very useful for data collection. Here is an example of how to use these options to collect and analyze data from a large number of flights.

```

1 % Number of flights
2 nFlights = 1e3;
3 % Loop over each flight
4 for i=1:nFlights
5     % Run simulation without graphics and save data
6     DesignProblem03('Controller','datafile','data.mat','display',false);
7     % Load data
8     load('data.mat');
9     % Get t and x
10    t = processdata.t
11    x = processdata.x
12    % Do analysis...
13    %     (your code here)
14 end

```

3.2 Presentation

The focus of your presentation this time will be on equations. For example, you should be sure to do the following:

- Number any equation to which you will refer, and do the reference with `\eqref`.
- Use standard notation for operations like multiplication (e.g., Ax for “ A times x ” and not $A \times x$ or $A * x$).
- Place square brackets around matrices. For example,

```
1 \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix}
```

produces the standard notation

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix},$$

while

```
1 \begin{matrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{matrix}
```

produces the non-standard and confusing notation

$$\begin{matrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{matrix}$$

- Align equations properly. For example,

```
1 \begin{align*}
2 c_{\text{L}} &= 2 \sin \alpha \cos \alpha \\
3 c_{\text{D}} &= 2 \sin^2 \alpha \\
4 \end{align*}
```

produces the nice-looking result

$$\begin{aligned} c_L &= 2 \sin \alpha \cos \alpha \\ c_D &= 2 \sin^2 \alpha \end{aligned}$$

while

```

1 \begin{equation*}
2 c_{\text{L}} = 2\sin\alpha \cos\alpha
3 \end{equation*}
4 \begin{equation*}
5 c_{\text{D}} = 2\sin^2\alpha
6 \end{equation*}

```

produces the messy-looking result

$$c_L = 2 \sin \alpha \cos \alpha$$

$$c_D = 2 \sin^2 \alpha.$$

- Use the correct font for functions like cosine (`\cos\alpha` produces a nice-looking $\cos \alpha$, while `\cos\alpha` produces a messy-looking $\cos \alpha$).

Please be responsive to feedback on your draft reports—this feedback will focus on helping you improve your presentation of equations.

3.3 Process

Your report will include the following four parts:

- Define one requirement and one verification.
- Derive a model (i.e., linearize (1) and express your result in state-space form).
- Design a controller and an observer (remembering to check, first, that your linearized model is both controllable and observable).
- Implement and test your control design in simulation. Follow the instructions you wrote to verify that your requirement has been satisfied. Characterize the performance of your control design by looking at aggregate results, as described in Section 3.1. Include at least one figure (e.g., of a histogram).

As usual, you are encouraged to go beyond these requirements. For example, it is likely that you will want to iterate on your design and on your choice of initial launch angle, in order to maximize flight distance. In this case, it would be helpful to describe the process that you used to improve your design.

4 Deliverables

4.1 First draft

You must submit one thing by 11:59PM on Wednesday, March 27, 2019:

- A PDF document that was generated using L^AT_EX and that includes, at minimum, a draft of your requirement and verification and of your state-space model.

Please submit this draft report at the following URL:

<https://forms.illinois.edu/sec/3565187>

You may submit as many times as you want—only the latest submission will be recorded. So please, submit early and often.

4.2 Second draft

You must submit one thing by 11:59PM on Wednesday, April 3, 2019:

- A PDF document that was generated using L^AT_EX and that includes, at minimum, a draft of your requirement and verification, of your state-space model, of your control design and analysis, and of your results in simulation.

Please submit this draft report at the following URL:

<https://forms.illinois.edu/sec/7322921>

You may submit as many times as you want—only the latest submission will be recorded. So please, submit early and often.

4.3 Final report

You must submit four things by 11:59PM on Friday, April 5, 2019:

- Code — Controller and Test. The controller code will be written in MATLAB, using the template `Controller.m`. The test code will be written in MATLAB, using the template `Test.m`. If `Test.m` is executed by one of your peers in a folder with your controller code `Controller.m`, the simulation code `DesignProblem03.m`, and a file with the equations of motion `DesignProblem03_EOMs.mat`, it will produce results that verify your requirement has been satisfied.
- Report — PDF and Source Files. This report will be written in L^AT_EX. You will submit both a PDF document and the L^AT_EX source files that you used to produce this document. The PDF document must be exactly four pages. The source files must be in a ZIP folder and must include all figures, references, etc., in addition to the `.tex` file. *Your report must conclude with a section titled “Acknowledgements” that lists the colleagues with whom you worked and describes the nature of your collaboration (discussion, sharing code, etc.). If you did not work with or benefit from interaction with anyone, please say so in your Acknowledgements section.*

Please submit this final report at the following URL:

<https://forms.illinois.edu/sec/7142286>

You may submit as many times as you want—only the latest submission will be recorded. So please, submit early and often.

5 Evaluation

Your work will be evaluated based on meeting intermediate deadlines (10%), on peer reviews both of your code (20%) and of your report (60%), and on your peer reviews of other reports (10%). Reviews of your technical approach will place special emphasis on data collection and analysis. Reviews of your report will place special emphasis on your presentation of equations—reviews will not critique style, grammar, or any other aspect of the text other than equations, as long as there is no barrier to understanding your work.

Peer review will be “double-blind.” You won’t know who reviewed your report, and your reviewers won’t know whose report they reviewed. To enable a double-blind review process, it is **very important** that your final report be completely anonymous. To repeat, **DO NOT INCLUDE YOUR NAME** or anything else that would identify you in any of the materials you submit—not in your PDF, in any of the files in your ZIP folder, or in either of your two MATLAB scripts. If you do include your name, your report will be returned without review, and you will receive a zero score. No exceptions will be made. Other details of peer review will be posted to piazza.

References

- [1] J. W. Roberts, R. Cory, and R. Tedrake, “On the controllability of fixed-wing perching,” in *American Control Conference*, Jun. 2009, pp. 2018–2023. [Online]. Available: <http://dx.doi.org/10.1109/ACC.2009.5160526>
- [2] J. Moore, R. Cory, and R. Tedrake, “Robust post-stall perching with a simple fixed-wing glider using lqr-trees,” *Bioinspiration & Biomimetics*, vol. 9, no. 2, p. 025013, 2014. [Online]. Available: <http://stacks.iop.org/1748-3190/9/i=2/a=025013>