AE353: Design Problem 04
(Control of NASA Robonaut Remix — 3D Edition)

T. Bretl

This paper describes the fourth design problem that you will complete in AE353 (Spring 2020). It asks you to design, implement, and test a controller for a fully 3D model of the NASA Robonaut traveling along a road, and to submit a report that describes both your method of approach and your results.

I. Nomenclature

x, y	=	horizontal and vertical position in meters
$\dot{x}, \dot{y}$	=	horizontal and vertical velocity in meters per second
θ	=	pitch angle in radians
$\dot{\theta}$	=	pitch angular rate in radians per second
ϕ	=	elevator angle in radians
$\dot{\phi}$	=	elevator angular rate in radians per second
v	=	airspeed in meters per second
α	=	angle of attack in radians
c_L, c_D	=	coefficients of lift and drag
$\dot{\phi}_{\max}$	=	maximum elevator angular rate in radians per second

II. Goal

The code provided in `DesignProblem04` simulates a two-wheeled robot that is similar to the segway robotic mobility platform*, which has been considered for use with the NASA robonaut (Figure 1). Actuators allow you to specify the torque applied to each wheel. Sensors allow you to measure the angular velocity of each wheel. Sensors also allow you to measure the distance from each wheel to the edge of the road. Finally, sensors tell you the turning radius of a road along which the robot can drive. The goal is to make the robot race along this road—without crashing—as fast as possible.

III. Model

A. Robot dynamics

The motion of the robot (Figure 2) is governed by ordinary differential equations with the form

$$\begin{bmatrix} \ddot{\phi} \\ \dot{v} \\ \dot{w} \end{bmatrix} = f(\phi, \dot{\phi}, v, w, \tau_R, \tau_L) \tag{1}$$

where ϕ is the pitch angle of the chassis, $\dot{\phi}$ is the pitch angular velocity, v is the forward speed, w is the turning rate, and τ_R and τ_L are the torques applied by the chassis to the right and left wheel, respectively [1, 2]. The function f in (1) depends on a number of parameters (e.g., lengths, masses, moments of inertia). You can save both a symbolic description and a numeric description of this function to a file for later analysis if you call the simulator with the optional parameter `'eomfile'`, for example as follows:

*See also `http://www.segwayrobotics.com`.

Fig. 1 The NASA robonaut on top of a segway robotic mobility platform (`http://spaceflight.nasa.gov/gallery/images/station/eva/html/jsc2005e11678.html`).

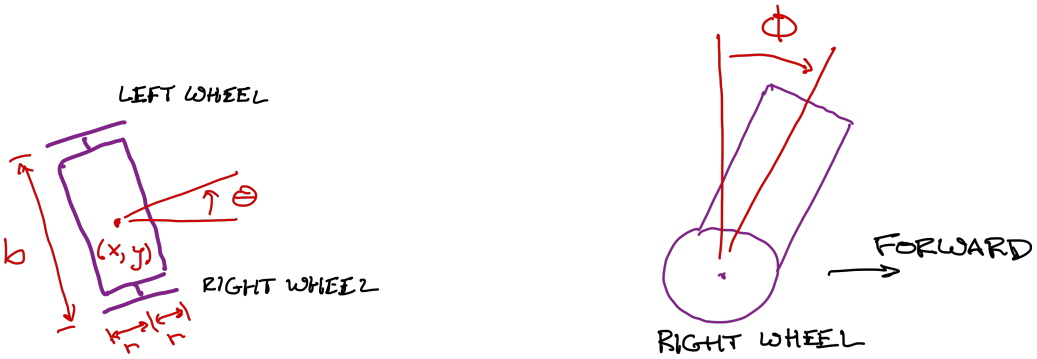

Fig. 2 Top and side view of robot schematic.

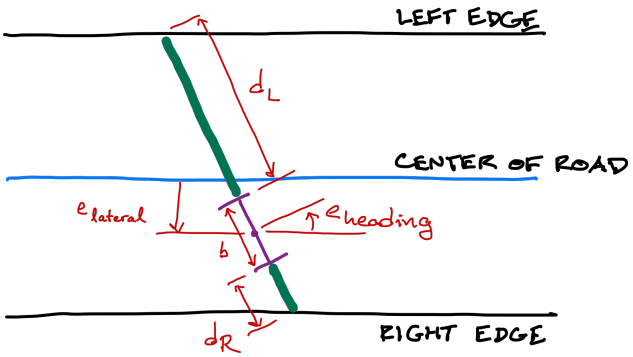

Fig. 3 The relationship between lateral error, heading error, and the distance from each wheel to the corresponding edge of a straight road.

```
% Run the simulator to save the equations of motion
DesignProblem04('Controller', 'eomfile', 'DP04_eom.mat', 'display', 'false');
% Load the equations of motion
load('DP04_eom.mat');
% Parse the equations of motion
f = symEOM.f;
% Define symbolic variables that appear in the equations of motion
syms phi phidot v w tauR tauL real
% Proceed to linearize or do whatever else you like...
%       (see: help sym/jacobian, help sym/diff, help sym/subs, etc.)
```

The simulator integrates three more equations of motion that relate the forward speed v and turning rate w to the position (x, y) and orientation θ of the robot:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= w. \end{aligned}$$

However, for the purpose of control design to follow a road, it is probably more useful to keep track of the position and orientation *relative to the road* and not with respect to an absolute reference frame. In particular, suppose you know the radius of curvature r_{road} of the road along which the robot is currently moving (where $r_{\text{road}} > 0$ means turning left, $r_{\text{road}} < 0$ means turning right, $r_{\text{road}} = 0$ means turning in place, and $r_{\text{road}} = \infty$ means going straight). Then, given a speed v_{road} at which you would like to travel along the road, the turning rate w_{road} necessary to follow its centerline can be computed as

$$w_{\text{road}} = \frac{v_{\text{road}}}{r_{\text{road}}}. \tag{2}$$

Now, define e_{lateral} as the perpendicular distance from the centerline of the road to the position (x, y) of the robot (where $e_{\text{lateral}} > 0$ means being too far to the right and $e_{\text{lateral}} < 0$ means being too far to the left), and define e_{heading} as the difference between the orientation θ of the robot and the direction of the road. It is possible to show that

$$\begin{aligned} \dot{e}_{\text{lateral}} &= -v\sin\left(e_{\text{heading}}\right) \\ \dot{e}_{\text{heading}} &= w - \left(\frac{v\cos\left(e_{\text{heading}}\right)}{v_{\text{road}} + w_{\text{road}} e_{\text{lateral}}}\right) w_{\text{road}}. \end{aligned} \tag{3}$$

A complete set of nonlinear equations of motion for the purpose of control design can be obtained from (1) and (3)—with states $e_{\text{lateral}}, e_{\text{heading}}, \phi, \dot{\phi}, v, w$ and inputs τ_R, τ_L—augmented as usual with a differential equation that describes $d(\phi)/dt$.

B. Robot sensors

Sensors give you three pieces of information:

- The radius of curvature (r_{road}) at the point on the road that is closest to the robot at any given time.
- The angular velocity of both the right wheel (w_R) and the left wheel (w_L), where positive means rolling forward and negative means rolling backward. You may assume that both wheels roll without slipping. Both the radius r of each wheel and the distance b between the wheels are available in `parameters`. Given this information, you should be able to express both w_R and w_L in terms of the forward speed v and turning rate w—you'll need to do so in order to model your sensors, so speak up right away if you have questions.[†]
- The distance d_R and d_L from the right and left wheel to the corresponding edge of the road (Figure 3). If you assume that the road is *straight*, then you should be able to express each distance as a function of the lateral error and the heading error. Again, try to do so right away and ask if you have questions.

[†]The dynamics reference pages from TAM212 on rigid bodies (`http://dynref.engr.illinois.edu/rkg.html`) and on rolling motion (`http://dynref.engr.illinois.edu/rko.html`) may be helpful.

C. Road

The road is defined by its centerline, which has piecewise-constant curvature. You can create a new, random road (saved by default to the file `'road.mat'`) by calling the function `MakeRoad`. This road will be a sequence of circular arcs, placed one after another. The array `road.s` stores the length of each arc. The array `road.w` stores the curvature of each arc. You need not worry about these data structures (unless you want to hand-design your own roads, for the purpose of testing)—your controller will not have access to them. Instead, as described in Section III.A, your controller will know the distance to the right and left side of the road (d_R and d_L, respectively) and the radius of curvature (r_{road}) at the point on the road that is closest to the robot at any given time. This information is available in `sensors`, as usual.

D. Variables

This section provides extra information about how variables are named in `DesignProblem04`, to avoid confusion. The function f in (1) is provided in MATLAB as `parameters.symEOM.f`. The inputs to this function are:

- ϕ, the chassis angle, called `phi` in MATLAB
- $\dot{\phi}$, the time derivative of the chassis angle, called `phidot` in MATLAB
- v, the forward speed, called `v` in MATLAB
- w, the turning rate, called `w` in MATLAB
- τ_R and τ_L, the right and left motor torques, called `tauR` and `tauL` in MATLAB

You can initialize these variables as follows:

```
syms phi phidot v w tauR tauL real
```

The extra two equations of motion in (3) are not provided in MATLAB. You will have to define them. The extra variables in these equations are:

- e_{lateral}, the lateral error in following the road, called `e_lateral` in MATLAB
- e_{heading}, the heading error in following the road, called `e_heading` in MATLAB
- v_{road} and w_{road}, the forward speed and turning rate of a trajectory that would follow the road centerline, called `v_road` and `w_road` in MATLAB—remember that w_{road} can be computed as in (2), given a choice of v_{road} and given the radius of curvature r_{road}, called `r_road` in MATLAB

You can initialize these extra variables as follows:

```
syms e_lateral e_heading v_road w_road real
```

Note that v_{road}, w_{road}, and r_{road} are not states—they are parameters on which your model might depend. Also note that you have choice over the names you use to describe variables in the equations of motion (3), since you'll be defining these equations yourself (they are not given to you in `DesignProblem04`). However, the names listed above are recommended.

IV. Tasks

A. Analysis

The focus of your analysis this time will be on identifying, diagnosing, and eliminating failure modes. "Failure" for the robot is a crash—either running off the road or dropping the chassis. There are many reasons why a particular control design might lead to failure, on certain roads or in certain situations. Your job as a control engineer is to uncover and address these sources of failure. At minimum, you should consider at least three different control designs, and for each design, you should do the following:

- Identify at least one road that causes failure. (Remember—a failed experiment is a blessing! Then you can address the source of the failure. There are always failure modes, and sometimes these are very hard to find.)
- Say why the failure occurred, providing evidence to support your argument. (Was there a large error in the state estimate? Did your control design exceed the maximum torque in each wheel? Is there a limit on how tight a turn your control design can handle, and can you quantify this limit? Thinking of the road as a source of disturbance in

the closed-loop system, do failures have something to do with characteristics of the frequency response? These are examples of questions you might ask when thinking about causes of failure.)
- Suggest a change to the control design that would eliminate the failure, and verify in simulation that it does. (Sometimes, it is impossible to eliminate a failure mode. If you believe that this is the case, and that you have reached a fundamental limit in the performance of the robot, provide evidence for this claim.)

Remember that you have a lot of practice in doing rigorous data collection, analysis, and visualization—this was the focus of the third design problem, and can help a lot in identifying and diagnosing failure modes. Also remember that (in general) your observer has to be working well in order for your controller to work well. So, one good place to look for causes of failure is the error in your state estimate.

B. Presentation

The focus of your presentation this time will be on citations. For example, suppose you want to quote a result that appears in the first edition of the textbook "Feedback Systems: An Introduction for Scientists and Engineers," which was written by Karl Aström and Richard Murray. To do this in LaTeX you would do the following:

- Create a file called `references.bib` and add it, for example, to the list of files in your overleaf project (you'll note that a "bib file" like this one is being used to generate this document—you can use it as a template).
- Add the following text to this file:

```
@book{Astrom2010,
  title={Feedback Systems: An Introduction for Scientists and Engineers},
  author={K. J. Astr{\"o}m and R. M. Murray},
  isbn={9781400828739},
  edition={First},
  url={http://www.cds.caltech.edu/~murray/amwiki/index.php/First_Edition},
  year={2010},
  publisher={Princeton University Press}
}
```

- Add the following text in your main document, where you want to cite the book:

```
\cite{Astrom2010}
```

  For example, you may want to say something like the following:

```
Robustness of the closed-loop system to time delay was verified
by analysis of the Nyquist plot \cite{Astrom2010}.
```

- Add the following text to your main document, just before the call to `\end{document}`:

```
% Display list of references in AIAA format.
\bibliography{references}
```

When you compile your document, all calls to `\cite{...}` will be replaced by numbered references `[...]`, which correspond to items in the list that will appear at the end of your document. For example, here is a citation to Aström and Murray, exactly as described[3]. There is a lot more information online about how to create, manage, and use bibliographies with latex documents[‡]. Do not hesitate to ask for help. At minimum, you should include at least one citation (other than to Aström and Murray) in your report.

C. Report

Your report must be a PDF document that was generated using LaTeX and that conforms to the guidelines for "Preparation of Papers for AIAA Technical Conferences" (`https://go.aerospace.illinois.edu/aiaa-latex-template`). The author must be listed as "Anonymous" (with no affiliations). The report must be exactly **six pages**. It must have the following sections:

- *Abstract*. Summarize your key results in one short paragraph.
- *Nomenclature*. List all symbols used in your report, with units.
- *Goal*. At minimum, this section will describe the system you have been asked to control, define one requirement and one verification (which must depend on rigorous data collection and analysis), and prepare the reader to understand the rest of your report.
- *Model.* At minimum, this section will linearize (1) about some choice of equilibrium point, expressing the result in state-space form.
- *Control Design.* At minimum, this section will present the design of a controller and an observer (remembering to check, first, that the linearized model is both controllable and observable).

[‡]For example: `https://www.overleaf.com/help/97-how-to-include-a-bibliography-using-bibtex`.

- *Results.* At minimum, this section will describe what you did to implement and test your controller and observer in simulation. It will show that you have followed the instructions that you wrote to verify that your requirement has (or has not) been satisfied. It will include at least one figure. It will identify, diagnose, and eliminate at least one source of failure in at least three different designs (see Section IV.A). It will also include an analysis of computation time (see below).

As usual, you are encouraged to go beyond these requirements.

D. Contest

There will be an opportunity to race against your friends in a friendly competition on the last day of class (Wednesday, May 6). The code that will be used to run this race is `DesignProblem04Race.m`. The key difference between this and `DesignProblem04.m` is that, rather than requiring you to specify a single controller, the race code requires you to specify the name of a folder in which you can put any number of controllers. A robot will be created for each controller in the folder—all will race at the same time. To test your controller in race conditions, do the following:

- Put the controllers you want to test in the folder `DesignProblem04/racers`.
- Open MATLAB and make `DesignProblem04` your working directory.
- Seed the random number generator:

```
rng('shuffle');
```

- Make a road:

```
MakeRoad('road.mat');
```

- Run the simulation (there are some optional parameters to play with, if you like):

```
DesignProblem04Race('racers','road.mat');
```

- If desired, show the results:

```
ShowResults('racers/results.mat');
```

There are three small but important differences in your controllers for the race:

- The race code enforces a bound on computation time. If your "runControlSystem" function takes more than 0.02 seconds on my laptop to execute on five separate occasions, your controller will be turned off (disqualified). Note that my computer will likely run your code at least as fast as your computer, but please err on the safe side.
- The race code enforces a strict prohibition on printing anything to the command window (e.g., the result of forgetting to put a semicolon at the end of a line). If either your "initControlSystem" function or "runControlSystem" function prints anything to the command window, your controller will be turned off (disqualified). Please check carefully that it does not!
- The race code allows each controller to—if desired—specify its own value of "iDelay" (the number of time steps that sensor data will be delayed). Look at the example `racers/netid_controller.m` to see how this is done. If you make no change to your controller code, that's fine—the default is `iDelay = 0`. If you choose to change your controller code to specify a non-zero delay, then your total time will be reduced by a number of seconds equal to `10 * iDelay` (so, if `iDelay = 2`, then your time will start at −20 seconds).

To compete in the race, you must submit your code by the deadline (see below). All submitted controllers will race in a qualifying round. The top 9 finishers will advance to the semifinals (in 3 groups of 3). The top finishers in each semifinal round will advance to the finals (in 1 group of 3). The qualifying round will be done before class—the results will be revealed at the start of class (on May 6). The semifinal and final round will be done in class. The winner of the final round will receive everlasting fame.

Note that this race is just for fun. Although you are required to submit your code to the race, the results have no impact at all on the assessment of your fourth design report.

V. Deliverables

A. Submit a first draft by 11:59PM on Wednesday, April 22, 2020

This draft must be a PDF document that follows the guidelines provided in Section IV.C and that includes, at minimum, a **complete draft** of your "Goal" and "Model" sections. It must be submitted here:

`https://forms.illinois.edu/sec/1106288`

You may submit as many times as you want—only the last submission will be recorded.

B. Submit a second draft by 11:59PM on Wednesday, April 29, 2020

This draft must be a PDF document that follows the guidelines provided in Section IV.C and that includes, at minimum, a **complete draft** of your "Control Design" and "Results" sections. It must be submitted here:

`https://forms.illinois.edu/sec/1489230`

You may submit as many times as you want—only the last submission will be recorded.

C. Submit a final report by 11:59PM on Tuesday, May 5, 2020

This report must be a PDF document that follows the guidelines provided in Section IV.C. You must submit three things in addition to the report:

- The LaTeX source files that you used to produce the document (in a ZIP folder with all figures, references, etc., in addition to the `.tex` file). Please do *not* include MATLAB code with your LaTeX source files.
- The MATLAB code that you used to produce all of the figures, tables, and other results that are included in the document (in a ZIP folder). Your code must have a MATLAB script called `GenerateResults.m`. It should be possible for any of your peers to reproduce *everything* in your report by executing this script. Please do *not* include LaTeX source files with your MATLAB code.
- The implementation of your "best" controller as a single file `Controller.m`. If one of your peers runs the simulator with this controller, they should see behavior that is consistent with claims made in the report. **The controller code that you submit will also be used in the race!**

Your report must be submitted here:

`https://forms.illinois.edu/sec/7946260`

You may submit as many times as you want—only the last submission will be recorded.

VI. Evaluation

Your work will be evaluated based on meeting intermediate deadlines (15%) on staff reviews of your code (20%) and of your report (60%), and on submission of controller code that runs in the race without being disqualified (5%)[§].

Reviews of your technical approach will place special emphasis on *analysis of failure*. Reviews of your report will place special emphasis on your *use of citations*.

Staff 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 your two ZIP folders, or in your MATLAB script.

VII. On-Time Submission

Draft reports and final reports must be submitted on time or they will receive zero credit. I am so serious about this that I will even give you extra credit if you submit your final report early:

> **If the last submission of your final report occurs by 11:59PM on Sunday, May 3, 2020—so, 48 hours before the final deadline—then you will receive 5% extra credit.**

Please do not put yourself in the danger zone (Figure 4). An easy way to make sure you do not receive zero credit is to submit each of your first two drafts (Sections V.A-V.B), as soon as you write them, also as your final report.

References

[1] Mak, Y. X., "Realization of mini segway robot using NI MyRIO," Essay (bachelor), University of Twente, 2015. URL `http://purl.utwente.nl/essays/67004`.

[§]Being "disqualified" means that your controller was turned off—e.g., for displaying text to the command window or for taking too much computation time—and not that your controller caused your robot to crash.

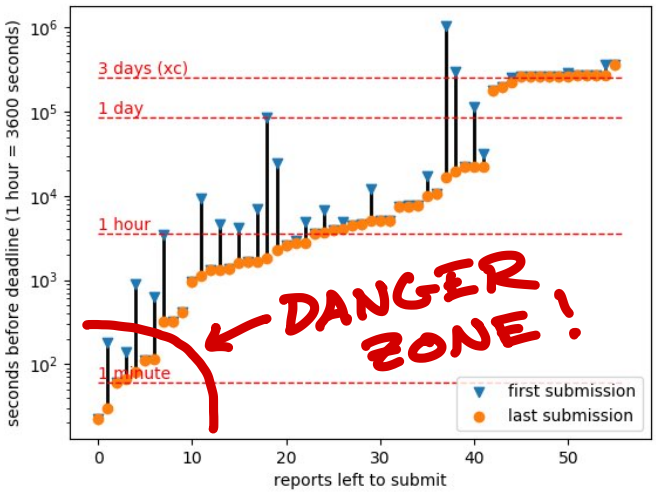

Fig. 4 Submission times for the third design report this semester. If you choose to submit your final report for the first time at the last minute, you are asking for trouble.

[2] Tobias, Z., and Matthias, F., "Tracking Control of A Balancing Robot—A Model-Based Approach," *Archive of Mechanical Engineering*, Vol. 61, No. 2, 2014, pp. 331–346. URL `http://dx.doi.org/10.2478/meceng-2014-0019`.

[3] Aström, K. J., and Murray, R. M., *Feedback Systems: An Introduction for Scientists and Engineers*, 1st ed., Princeton University Press, 2010. URL `http://www.cds.caltech.edu/~murray/amwiki/index.php/First_Edition`.