Assignment 5: Quadrotor Control via LQR

Tufts ME 149: Optimal Control For Robotics

Assigned: Feb 13 — Due: Feb 27

Introduction

In this assignment you will design two controllers for a planar model of a quadrotor helicopter. The first controller will be designed using infinite-horizon LQR and will be used to regular the state of the quadrotor in a stationary hover. The second controller will be designed using finite-horizon (trajectory) LQR and will stabilize an arbitrary reference trajectory that starts and ends in a static hover.

Unlike previous assignments, you will not need to submit any plots. Instead, for each part of the assignment you will submit a Matlab function that returns a function handle for a feedback controller. Your controller will be evaluated by a set of automated unit tests. These tests will run simulations of the closed loop system with a variety of different perturbations, disturbances, and other error sources.

You should write your own test scripts to ensure that your controllers work. There are no direct requirements on these tests, although you should submit them along with your other code. A few things that you could test,

Planar Quadrotor Model

This assignment is centered around controlling a planar quadrotor model. All of the dynamics functions that you will need are provided in the ME-149 code library:
`ME149/codeLibrary/modelSystems/planarQuadrotor/`.

There are several parameters for the quadrotor. Your controller design function should work for any valid set of parameters, but you may choose the following set of nominal values for your tests: `param.m = 0.4`, `param.w = 0.4`, `param.g = 10`

In addition to the dynamics functions, the code library also provides a variety of useful diagnostics tools for the quadrotor. For example, you can visualize the output of a simulation using the `planarQuadrotorPlot()` to generate a plot of the state and control versus time, or `planarQuadrotorAnimate()` to generate an interactive animation.

For the purposes of testing, I have included a special version of the dynamics function:
`planarQuadrotorRealDyn()` that implements a more realistic version of the dynamics. In particular, it adds a random disturbance force that is a rough proxy for wind. By default it will always use the same disturbance, but you can pass a seed to the random number generator to test additional disturbances. You can also adjust the magnitude of the disturbances (see the help file for details).

Part One: Hover Controller

For the first part of this assigment you will need to implement the file `getHoverController.m`, included below. The output of this function is a feedback controller (as a function handle) that can be passed into a simulation. Also write a test for the controller (`TEST_getHoverController.m`). This function (or script) should generate one or more simulations that test your controller and demonstrate that it is reliable.

```
function hoverController = getHoverController(xRef, yRef, param)
% hoverController = getHoverController(xRef, yRef, param)
%
% This function designs a controller that will allow the quadrotor to
% hover at a fixed position, despite the presence of various disturbances.
% The linear controller gains are designed using infinite-horizon LQR
%
% INPUTS:
%   xRef = scalar = horizontal position for the hover
%   yRef = scalar = vertical position for the hover
%   param = struct with constant scalar parameters:
%       .m = mass of the quadrotor
%       .w = distance between the rotors (width)
%       .g = gravity acceleration
% OUTPUTS:
%   hoverController = function handle:  u = hoverController(z)
%       IN: z = [6, n] = [x; y; q; dx; dy; dq] = state
%             x = horizontal position
%             y = vertical position
%             q = absolute angle (zero for hover)
%             dx = time-derivative of horizontal position
%             dy = time-derivative of vertical position
%             dq = time-derivative of absolute angle (zero for hover)
%       OUT: u = [2, n] = [u1; u2] = control
%             u1 = left rotor force
%             u2 = right rotor force
%
% NOTES:
%   The nominal system dynamics for the quadrotor are defined in:
%
%        ME149/codeLibrary/modelSystems/planarQuadrotor/
%

%%%% TODO: %%%%
%
% Design a linear controller that stabilizes a hover. Useful functions:
%
% lqr()
% planarQuadrotorHoverThrust()
% planarQuadrotorLinDyn()

% Return a handle to the controller.
TODO = 'replace this placeholder with a useful set of parameters';
hoverController = @(z)( planarQuadrotorHoverController(z, TODO ) );

end

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%

function u = planarQuadrotorHoverController(z, TODO )
%
% TODO: documentation for this function
% TODO: additional arguments
% TODO: implement this function
%
end
```

Part Two: Trajectory Tracking

The second part of this assignment is similar to the first, except that now you will design a trajectory tracking controller instead of a hovering controller. As with the first part, you do not need to submit plots - instead you will implement two matlab functions:
`getTrackingController.m`
`trajectoryLqr.m`

The first function will design a trajectory tracking controller, and the second function computes the time-varying LQR gains along a reference trajectory. The details for implementation are included in the files. Your controller will be evaluated by running it through a series of automated test simulations.

You should thoroughly test you controller before submitting to make sure that it works. Submit the test script (`TEST_getTrackingController.m`) to Trunk along with the rest of your code. The only requirement for this test script is that it should demonstrate that you controller works by running one or more simulations and then generating some meaningful output.

The assignment includes an single reference trajectory that guides the quadrotor through a back-flip while moving horizontally between two static hover poses. The function `importReferenceTrajectory.m` can be used to import this data and convert it into a useful format.

Write-Up

In addition to the Matlab files listed above, please also upload a short write-up for this assignment:
`hw_05_studentName_writeup.txt`

- Header: full name, date, assignment name and number
- List any other students that you worked with.
- How long did this assignment take you?
- Briefly describe your approach to testing your controllers.