 Lab 1: Simulation and Control using MATLAB

By: Shawon Dey

Instructor: Dr. Randy Hoover

Date: 26 October 2019

1. **Introduction**

The purpose of this lab was to simulate and design a control law for the altitude of a quad-rotor helicopter. A side view of a traditional quad copter was given where h(t) are is from the inertial reference frame. For one, designing of PID controller was performed to track result for r(t) =50 meters in altitude., second, limiting the thrust to ±10N and checking the if the control still works. Finally, setting the thrust limit to zero and see if the control still works.

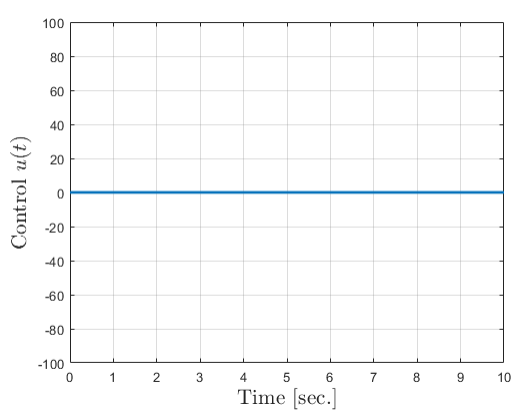
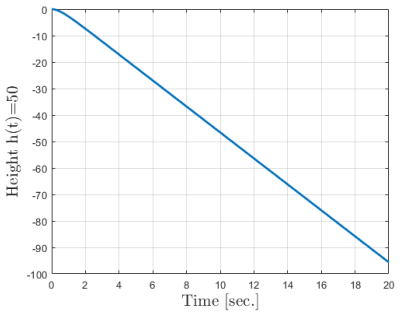
1. **Modeling**

Designing the dynamics,

Let, m = 1, g = , r = 2

1. **Deliverables**

Below the plots showing with no control input and initial conditions:

(a) (b)

Figurer 1 (a) (b): Input and Response with No Control

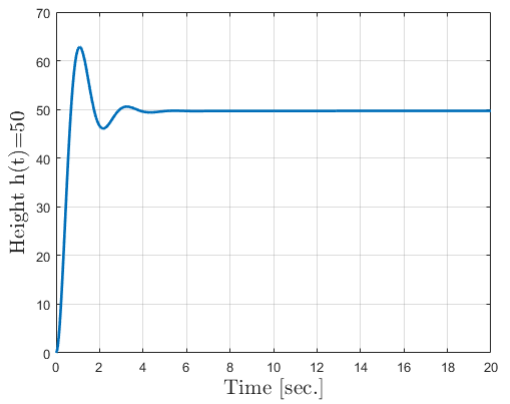
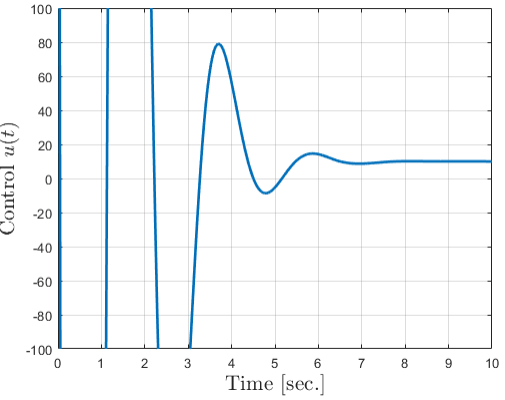
1. Is the system stable? (Marginal or asymptotic)?

Answer: The system is not stable. The is falling due to gravity pull.

2. Does it track a desired reference(t)?

Answer: No, it’s falling continuously.

1. **Design**

(a) (b)

Figures 2 (a) (b): Controlled Input and Response of System

1. Design a controller for the system using P, PI, PD, or PID to get a desired tracking result for r(t) = 50.

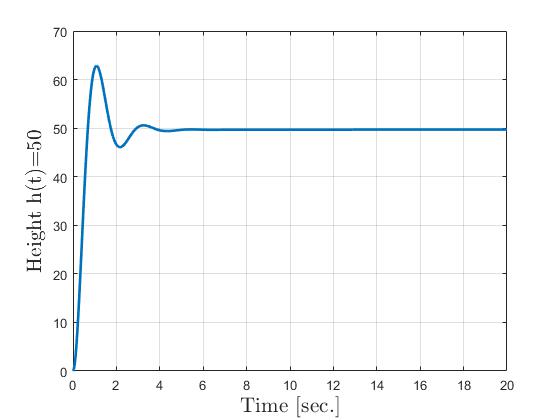
2. In a normal" system, actuators have finite ability. We can replicate this by limiting the elevator response. Add a section in your simulation to limit the elevator (i.e., your control input) to never exceed ±10N. Does your control still work?

Answer: The controller tracks the desired height of 50 meters, however, initially there’s an anomaly, the quadcopter oscillates up and down, after 4 seconds stables at 50 meters.

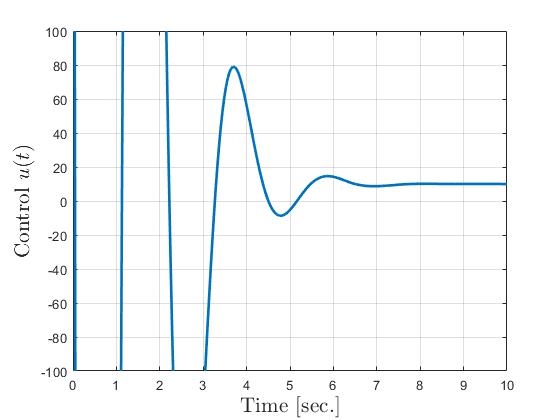
1. **Submission**

Quadrotor control system

%Shawon dey  
%%Main%%  
clc  
clear  
close all  
% Simulate the autopilot %  
options = odeset('RelTol',1e-4,'AbsTol',[1e-5 1e-5]);  
tspan = linspace(0,20,500);  
% Calling sequence: ode45(System function, time length, ICs, options)  
[T,X] = ode45(@QuadrotorAltitude,tspan,[0 0],options);  
figure;  
plot(T,X(:,1),'linewidth',2)  
xlabel('Time [sec.]','fontsize',16,'interpreter','latex')  
ylabel('Height h(t)=50',...  
'fontsize',16,'interpreter','latex')  
grid on



for i = 1:length(T)  
[dx tu] = QuadrotorAltitude(T(i),X(i,:));  
uk(i) = tu;  
end  
figure;  
plot(T,uk,'linewidth',2)  
xlabel('Time [sec.]','fontsize',16,'interpreter','latex')  
ylabel('Control $u(t)$','fontsize',16,'interpreter','latex')  
grid on  
axis([0 10 -100 100])



%Shawon dey  
function [dx ut] = QuadrotorAltitude(t,x)  
% Model of the aircraft system and Control gains %  
kp = 10; % Proportional control gain  
kd = 1000; % Dirivative control gain (recall this has\_t wrapped  
in) %  
ki = 0.0001; % Integral control gain (recall this has\_t wrapped  
in) %  
r = 50; % Desired geight %  
e = r - x(1); % Pitch error %  
m = 1; %mass of the quadrotor %  
gamma = 2; %darg coefficient %  
g = 9.81; % garvity %  
% Account for actuator saturation %  
u\_limit = 10;  
% PID regulation - need to keep a running total of the error  
% as well as the old error for the derivative term %  
% ek is the total error accumulated, e\_old is the previous error  
persistent ek e\_old  
edot = (e-e\_old); %the differentiacial of error  
% First time through e\_old will be empty because there's no  
% intitialization (need to account for that)  
if(isempty(e\_old))  
u = kp\*e; % Propotional controlle  
%u =110;  
else  
% Using sum(e)to indicate the accumulated number  
u = kp\*e + kd\*edot + ki\*sum(ek); % The PID controller  
expression  
%u = 110;  
end  
% Check and limit actuators here %  
% Keep old values for control %  
e\_old = e;  
ek = [ek;e];  
% Enter system dynamics here %

dx = zeros(2,1);

% The dynamic models, using the state-space form  
dx(1)= m \* x(2) ;  
dx(2)= -gamma\*x(2) + u - g;

% Need this to get the control signal out later %  
if nargout>1  
ut = u;  
end  
end

*Published with MATLAB® R2019b*