 Project 1: Simulation and Control using MATLAB

By: Shawon Dey

Instructor: Dr. Randy Hoover

Date: 22 October 2019

1. **Introduction**

The purpose of this lab was to simulate and design for the longitudinal axis of an aircraft in MATLAB. The first part was to design the PID controller to get a desired tracking result for pitch angle 11*◦*. Second, we analyzed control by limiting the elevator response and see if the control still works.

1. **Modeling**

The following dynamics were given:

*α*˙ = *−*0*.*313*α* + 56*.*7*q* + 0*.*232*δ q*˙ = *−*0*.*0139*α −* 0*.*426*q* + 0*.*0203*δ*

*θ*˙ = 56*.*7*q*

where q is the pitch rate and elevator deflection is our control input. Our goal is to design a pitch controller for our aircraft system (assuming the reference trajectory r(t) = *θd*(*t*) is a constant measure in radians. Note that if we define x1(*t*) = *α*(*t*) *, x*2(*t*) = *q*(*t*) *, x*3(*t*) = *θ*(*t*) *,* and u(t) = *δ*(*t*)

% system dynamics here %  
dx = zeros(3,1);  
dx(1) = (-0.313)\*x(1) + 56.7\*x(2) + 0.232\*u;  
dx(2) = (-0.0139)\*x(1) + (-0.426)\*x(2) + 0.0203\*u;  
dx(3) = 56.7\*x(2);

1. **Deliverables**
   1. **Simulation**

With no control input and initial conditions:

*x*(0) = [0 0*.*1 0]

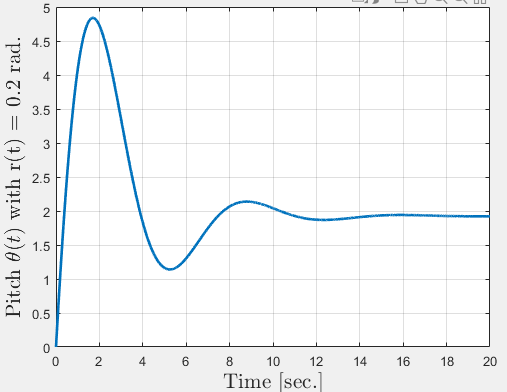
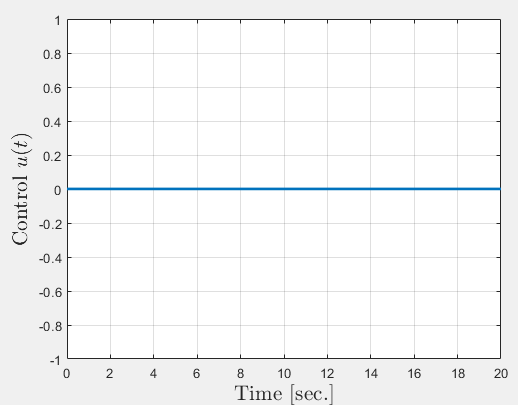
(a) (b)

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

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

Ans: The system is steady state but with some overshot, stable.

1. Does it track a desired reference r(t)?

Ans: No, the system does not track the desired pitch of 0.2 rad, the current pitch is apx. 2 rad.

* 1. **Design**

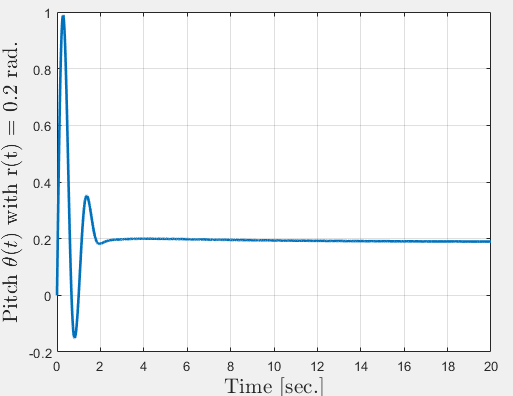
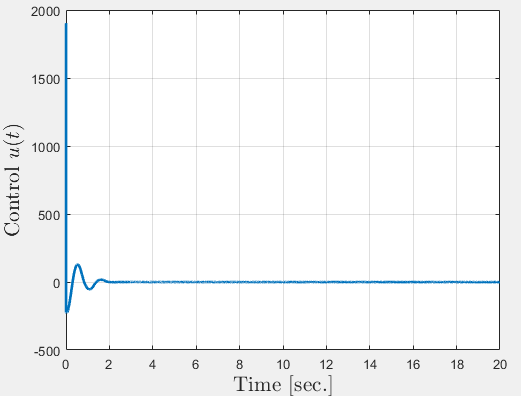
(a)****(b) ****

Figure 2 (a) (b): Controlled input and response of the system

1. Design a controller for the system using P, PI, PD, or PID to get a desired tracking result for r(t) =0.2 rad.
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 *π/*4*.*

Does your control still work?

Ans: The designed controller still tracks the 0.2 rad.

* 1. **Submission**
     1. **With No Control Input**

% Shawon dey %

% Robotic Control System %

clc

clear

close all

% Simulate the autopilot %

options = odeset('RelTol',1e-4,'AbsTol',[1e-5 1e-5 1e-5]);

tspan = linspace(0,20,5000);

% Calling sequence: ode45(System function, time length, ICs, options)

[T,X] = ode45(@AircraftPitchSystem,tspan,[0 0.1 0],options);

figure;

plot(T,X(:,3),'linewidth',2)

xlabel('Time [sec.]','fontsize',16,'interpreter','latex')

ylabel('Pitch $\theta(t)$ with r(t) = 0.2 rad.',...

'fontsize',16,'interpreter','latex')

grid on

for i = 1:length(T)

[dx tu] = AircraftPitchSystem(T(i),X(i,:));

uk(i) = tu;

end

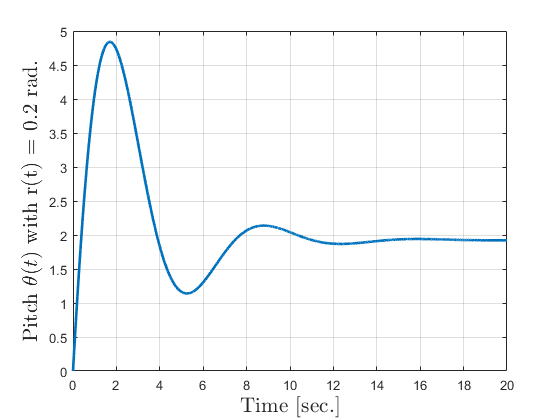
figure;

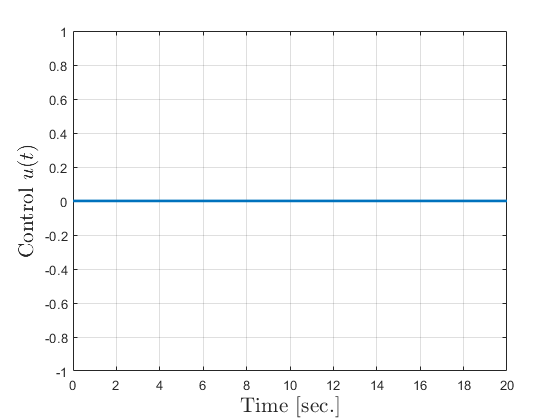
plot(T,uk,'linewidth',2)

xlabel('Time [sec.]','fontsize',16,'interpreter','latex')

ylabel('Control $u(t)$','fontsize',16,'interpreter','latex')

grid on





Function-

% Shawon dey %

% Robotic Control System %

function [dx ut] = AircraftPitchSystem(t,x)

% Aircraft system and Control gains %

kp = 0; % Proportional control gain

kd = 0; % Dirivative control gain (recall this has\_t wrapped

ki = 0.0; % integral control gain (recall this has\_t wrapped

r = 0.2; % Desired pitch angle %

e = r - x(3); % Pitch error %

integral = 0; % defining integral value %

% Account for actuator saturation %

u\_limit = pi/4;

% 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); % current error %

% First time through e\_old will be empty because there's no

% integralitialization (need to account for that)

if(isempty(e\_old))

u = kp\*e; % if e\_old empty %

else

%integral = integral + e; % if e\_old is not empty, increment %

u = kp\*e + kd\*edot + ki\*sum(ek); % PID expression

end

% Keep old values for control %

e\_old = e;

ek = [ek;e];

% system dynamics here %

dx = zeros(3,1);

dx(1) = (-0.313)\*x(1) + 56.7\*x(2) + 0.232\*u;

dx(2) = (-0.0139)\*x(1) + (-0.426)\*x(2) + 0.0203\*u;

dx(3) = 56.7\*x(2);

% Need this to get the control signal out later %

if nargout>1

ut = u;

end

# *Published with MATLAB® R2019a*

# Control with input

# % Shawon dey % % Robotic Control System % clc clear close all % Simulate the autopilot % options = odeset('RelTol',1e-4,'AbsTol',[1e-5 1e-5 1e-5]); tspan = linspace(0,20,5000); % Calling sequence: ode45(System function, time length, ICs, options) [T,X] = ode45(@AircraftPitchSystem,tspan,[0 0.1 0],options); figure; plot(T,X(:,3),'linewidth',2) xlabel('Time [sec.]','fontsize',16,'interpreter','latex') ylabel('Pitch $\theta(t)$ with r(t) = 0.2 rad.',... 'fontsize',16,'interpreter','latex') grid on

# for i = 1:length(T)

# [dx tu] = AircraftPitchSystem(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

# 

# Function-

# % Shawon dey % % Robotic Control System % function [dx ut] = AircraftPitchSystem(t,x) % Aircraft system and Control gains % kp = 30; % Proportional control gain kd = 10000; % Dirivative control gain (recall this has\_t wrappedin) % ki = 0.0001; % integral control gain (recall this has\_t wrappedin) % r = 0.2; % Desired pitch angle % e = r - x(3); % Pitch error % integral = 0; % defining integral value % % Account for actuator saturation % u\_limit = pi/4; % 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); % current error % % First time through e\_old will be empty because there's no % integralitialization (need to account for that) if(isempty(e\_old)) u = kp\*e; % if e\_old empty % else %integral = integral + e; % if e\_old is not empty, increment % u = kp\*e + kd\*edot + ki\*sum(ek); % PID expression end % Keep old values for control % e\_old = e; ek = [ek;e]; % system dynamics here % dx = zeros(3,1); dx(1) = (-0.313)\*x(1) + 56.7\*x(2) + 0.232\*u; dx(2) = (-0.0139)\*x(1) + (-0.426)\*x(2) + 0.0203\*u; dx(3) = 56.7\*x(2); % Need this to get the control signal out later % if nargout>1 ut = u; end

# *Published with MATLAB® R2019a*