North Carolina A&T State University

Optimal Control Based Fixed-Wing Aircraft Landing Controller Design

ECEN 865: Optimal Control System

Final Project

Full name	**Abenezer Taye**
Banner ID	**950413723**
Level	**Ph.D.**
Email	**agtaye@aggies.ncat.edu**

Instructor: Ali Karimoddini, Ph.D.

May 9, 2020

Abstract

One of the most difficult tasks in aircraft control is achieving a safe and comfortable landing. A review of accident statistics indicates that over 45 percent of all general aviation accidents occur during the approach and landing phases of a flight. A closer look shows that the cause of over 90 percent of those cases was pilot related. Loss of control was also a major contributing factor in 33 percent of these cases. In many cases, the accidents have happened due to not landing within a targeted landing time. Assuming that the lateral and longitudinal dynamics of the aircraft are decoupled, and the lateral controller takes care of the lateral deviations from the center line of the runway, in this project, we have designed a longitudinal controller for a fixed wing aircraft to track the desired landing trajectory to safely land the aircraft at Charlotte Douglas International Airport on runway 18L. The approach plate for the runway 18L is attached with this document.

Table of Contents

List of figures

List of tables

Introduction

The aircraft landing process is composed of five different phases: the base leg, the glide slope, the flare out, the touch down, and the after-landing roll. Among these five phases, the glide slope phase and the flare out phase are the most important phases[1]. In this project, we design a tracking controller that guides an aircraft to have a smooth and comfortable landing. The flare-out phase of the landing is considered in this project.

The model of the system that we use in this project is taken from[1] . In that paper, the kinematic model of the airplane is considered. In order to come up with the model of the system, the following assumptions have been taken: wind disturbances are assumed to be zero, deviation from the equilibrium flight condition is zero, glide path angle is assumed to be small so that small angle approximation can be made, velocity of aircraft is constant, longitudinal motion is governed by elevator deflection, lateral motion is ignored.

$$\dot{X} = Ax(t) + B\delta_e(t)$$

$$y = Cx(t)$$

Where, $x(t) = [h(t)\ \ \dot{h}(t)\ \ \theta(t)\ \ \ \dot{\theta}(t)]^T$, $B = [0\ 0\ 0\ b_4]^T$,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Table 1: Model Parameters

a_{22}	$\frac{-1}{T_s}$
a_{23}	$\frac{V}{T_s}$
a_{42}	$\frac{1}{vT_s^2} - 2\frac{\zeta\omega_s}{vT_s} + \frac{\omega_s^2}{v}$
a_{43}	$2\frac{\zeta\omega_s}{T_s} - \omega_s^2 - \frac{1}{T_s^2}$
a_{44}	$\frac{1}{T_s} - 2\zeta\omega_s$
b_4	$\omega_s^2 k_s T_s$

No	*Symbol*	*Parameter value*
1	K_s	$-0.95sec^{-1}$
2	T_s	$40\ sec$
3	ω_s	$1\ rad/sec$
4	ζ	0.5

Design of reference landing trajectory

The reference landing trajectory of an aircraft is a path that governs the movement of the aircraft during the landing phase. This trajectory is important because the it ensures safe and comfortable landing [quote Solomon's paper]. Glide slope phase and flare out phase are the two important phases of the landing trajectory. The glide slope phase is desired to be a straight line and the flare out phase is desired to have an exponential profile in order to ensure the smooth and comfortable landing requirement. The following figure illustrates the landing trajectory of an aircraft.

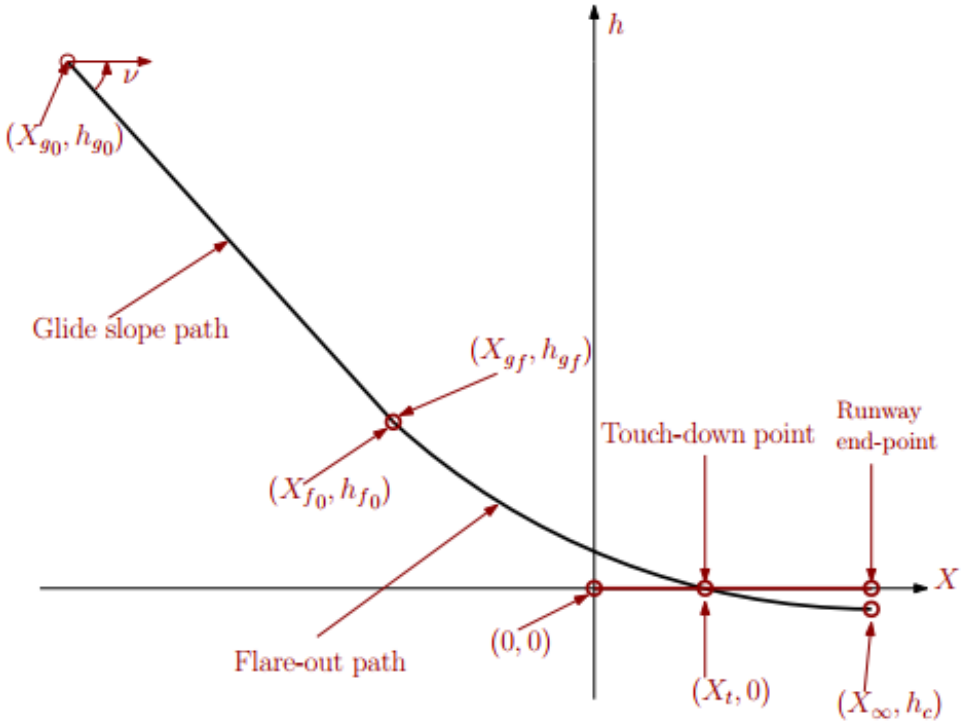

Figure 1: Aircraft landing profile

The desired altitude trajectory for the glide slope phase and the flare out phase is given as

$$h^d = -\tan(\nu^d)\,(X - X_{go}) + h_{g0}$$

$$h^d = -h_c + (h_{f0} + h_c)e^{-K_x(X - X_{f0})}$$

In the above equations X_{f0}, h_c, K_x are unknowns. To find these parameters, we used the following constraints. $\dot{h}^d = -\tan(\nu^d)$ this called slope continuity constraint and it guarantees the slope at the beginning of the flare-out path is equal to the slope of the glide slope path. Also, from the slope at the flare-out phase, we obtain $K_x = \frac{\tan(\nu^d)}{h_{f0}+h_c}$. Path constraint, which is given by the equation $X_{f0} = \frac{h_{g0}-h_{f0}}{\tan(\nu^d)} + X_{g0}$. Lastly, we have also a touchdown constraint $-h_c + (h_{f0} + h_c)e^{-K_\infty(X_t - X_{f0})} = 0$, which guarantees that the landing path intersects the ground at the touchdown point X_t. These three equations are solved simultaneous to obtain the values of K_x *and* h_c. The parameters X_{g0}, h_{g0}, X_t, and X_{f0} are extracted from the Charlotte Douglas International Airport 18L runway.

No	*Name*	Parameter value
1	X_{g0}	-30380 ft
2	h_{g0}	1620 ft
3	X_t	2892 ft
4	K_x	$6.161x10^{-4}$
5	V	256

6	K	0.1579
7	h_c	6.017
8	X_{f0}	$-1376.7\ ft$

Table 2: Parameter values used to design the reference trajectory

No	$Name$	Parameter value
1	t_0	0 sec
2	t_f	17 sec
3	h_0	100ft/sec
4	$\dot{h}_0$	-14 ft/sec
5	θ_0	-0.05 rad
6	$\dot{\theta}_0$	0 rad/sec
7	h_f	0 ft
8	$\dot{h}_f$	0 ft/sec
9	θ_f	0 rad
10	$\dot{\theta}_f$	0 rad/sec

Table 3: Initial and final conditions

Design of performance index

The following are the requirements that our controller is expected to satisfy.

Constraint	Description	Mathematical formulation
C1	The landing path should be an exponential path to insure a safe and comfortable landing.	Exponential (smooth) trajectory
C2	The rate of decent should be non-zero to avoid overshooting.	$60 \leq \left\|\dot{h}^{d}(t_f)\right\| \leq 180\ ft/min$
C3	The nose wheel of an aircraft and the tail gear can't touch the ground first.	$0^o \leq \theta(t_f) \leq +10^0$
C4	The rate of change of angle attack is restricted to a value less than 20 percent of the stall value.	$\alpha(t) < 18^0$ $\Delta\alpha(t) < 3.6^0$
C5	To avoid saturation, the elevator is not permitted to operate against the mechanical stops during the landing process.	$-35^0 \leq \delta_e(t) \leq +15^0$

Table 4: List of constraints

From the above description, we can understand that our problem is a tracking problem, which is a constrained optimization problem. The general form of the problem can be given as

$$\min J(x,u) = g(x(tf),tf) + \int_{to}^{tf} h(x(t),u(t),t)dt$$

Such that: $\dot{X} = f(x(t),u(t),t)$

From the above general arrangement, we can formulate a Hamiltonian function $H = h + \lambda^T f$.

$$H = \frac{1}{2}[(x(t) - r(t))]^T Q\big[\big(x(t)\big) - r(t)\big] + \frac{1}{2}\delta_e^T(t) R \delta_e(t) + \lambda^T\big(Ax(t) + B\delta_e(t)\big)$$

The co-state vector can be defined as $\lambda(t) = S(t)x(t) - v(t)$, where $S(t)$ and $v(t)$.

$$S(t) = C^T P(t) C(t)$$

$$v(t) = C^T P r(t)$$

$$\underset{\vee \epsilon[t_0\ tf]}{min} \bar{J} = [Cx(tf) - r(tf)]^T P[Cx(tf) - r(t)] + \int_{t0}^{tf} \big(H - \lambda^T \dot{x}(t)\big)d\tau$$

Design of optimal tracking controller

Using the above formulated performance index, we can design an LQT controller to track the reference trajectory. The following algorithm has been implemented using MATLAB to design the LQT controller.

	LQT Algorithm to design an optimal tracking controller
1	**Initialize**
	Plant = $\begin{cases}\dot{X} = Ax + Bu \\ Y = Cx + Du\end{cases}$ Reference trajectory Initial and Final Conditions P, Q, R
2	$-\dot{S} = A^T S + AS\ - SBR^{-1}B^T S + C^T QC, S(t_f) = C^T PC$
3	$K(t) = R^{-1}B^T S(t)$
4	$-\dot{v} = (A - BK)^T v + C^T Qr, v\big(t_f\big) = C^T \mathrm{Pr}(t_f)$
5	$\delta_e(t) = -K(t)X(t) + R^{-1}B^T v(t)$
6	$\dot{X} = Ax + B(-K(t)X(t) + R^{-1}B^T v(t))$
7	Tune P, Q, R
8	Go to 2.

9	End Procedure

Table 5: LQT algorithm to solve an optimal tracking controller

Results and discussion

After tuning the P, Q, and R parameters, the best result we could achieve was

$$\boldsymbol{P} = \begin{bmatrix} \mathbf{0.9} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.042} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} \mathbf{0.0006} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.015} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{182} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{45} \end{bmatrix}$$

$$\mathbf{R = 1000}$$

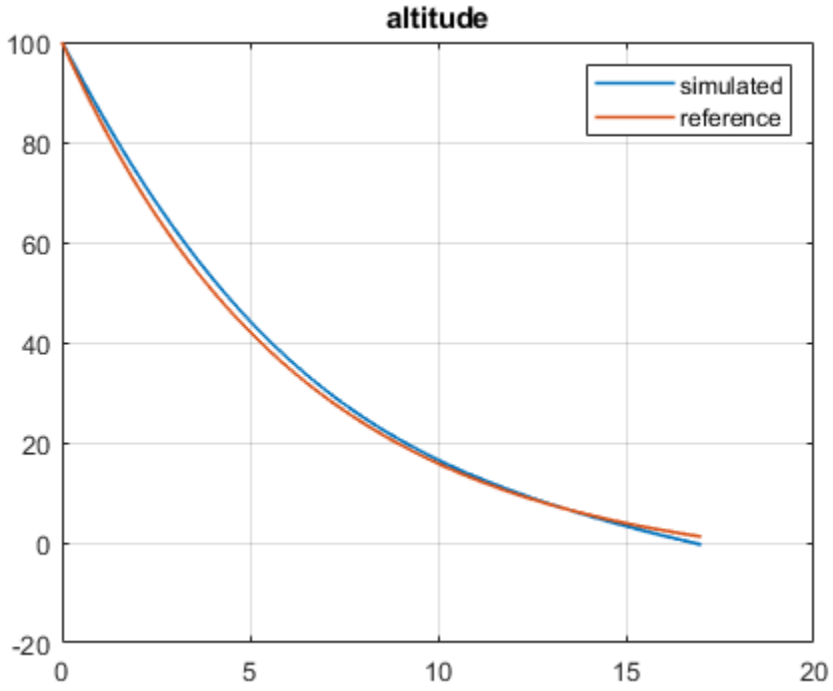

Figure 2: Reference and Simulated Altitude

As it can be seen from the above figure, the designed controller was able to guide the aircraft to move along with its desired reference trajectory. Also, the controller was able to make the airplane arrive at the touch-down point while avoiding a tangential motion of the airplane at the touchdown point.

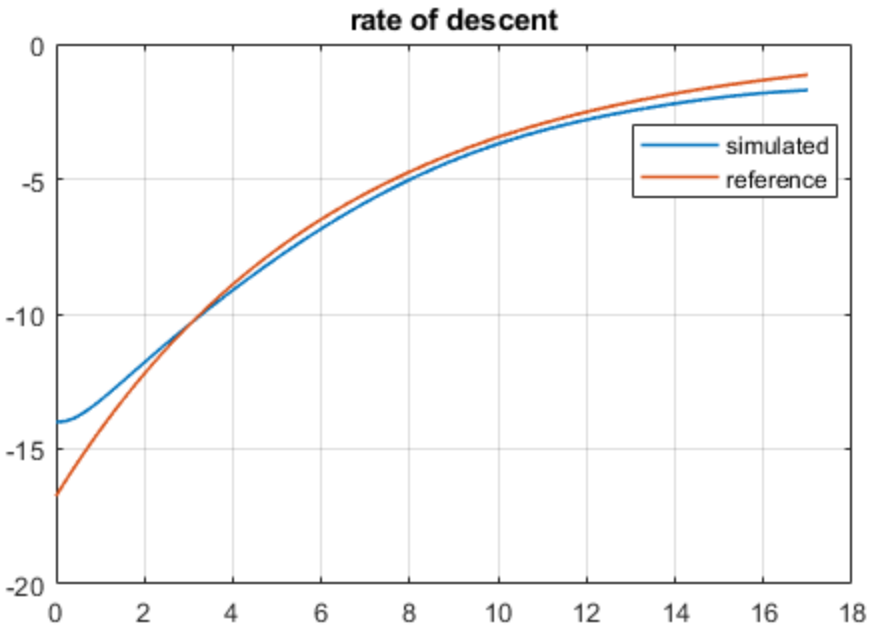

Figure 3: Reference and Simulated Rate of descent

From the above figure, we can observe that the designed controller was able to track the reference for the rate of descent with acceptable precision. In addition, the controller can satisfy the second constraint that dictates the system to have a final rate of descent to be between 60 and 180 ft/min.

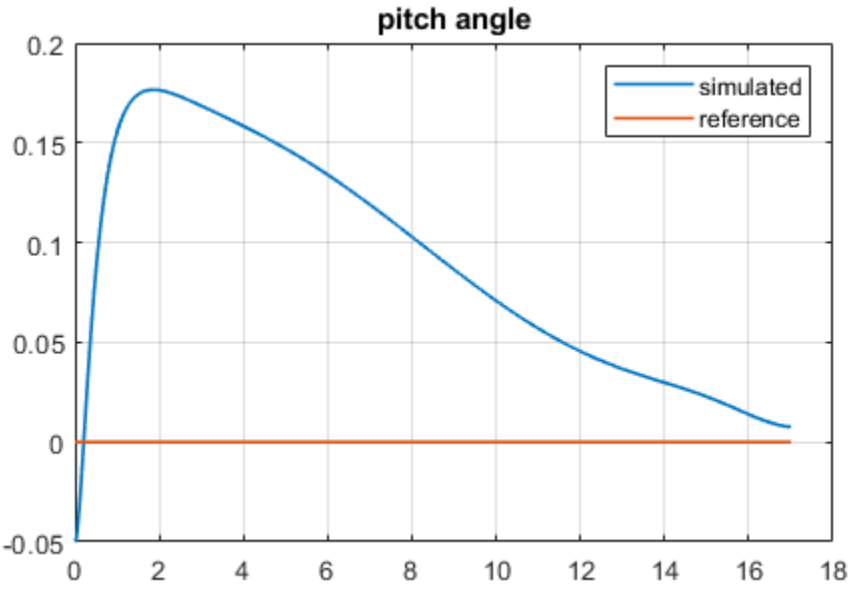

Figure 4: Reference and Simulated Pitch angle

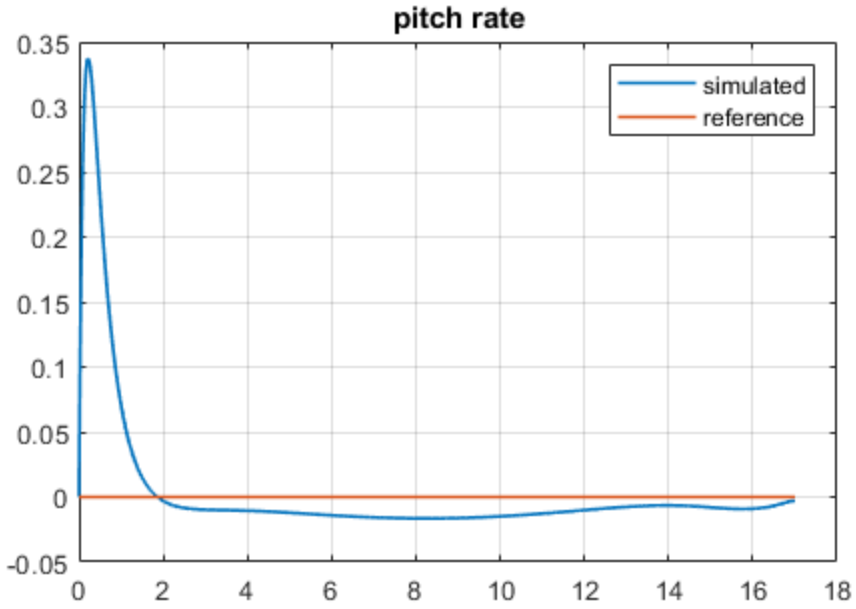

Figure 5: Reference and Simulated Pitch rate

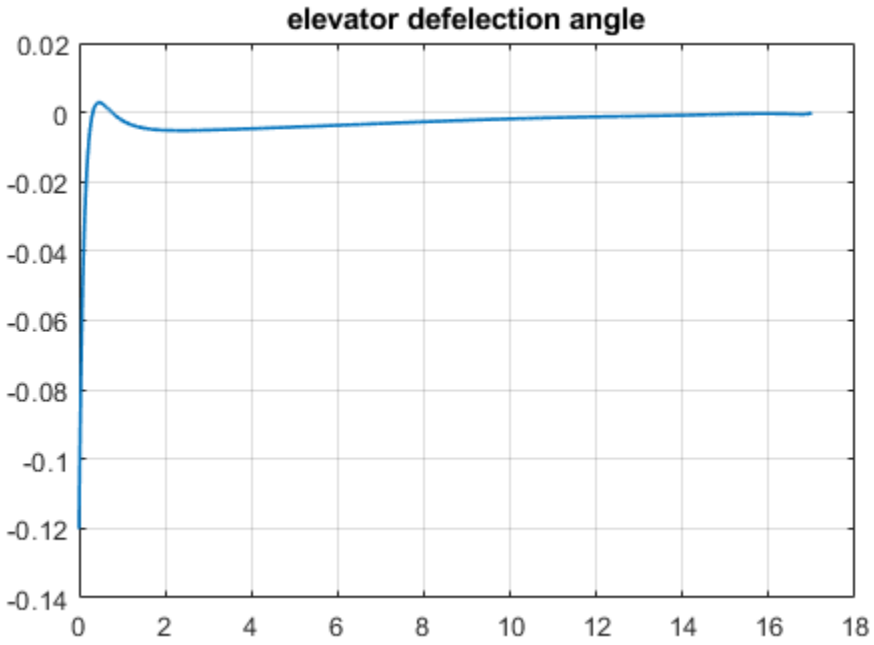

Figure 6: Control input profile

The above figure demonstrates what the control input profile looks like. From the figure, we can see that the saturation region of the elevator deflection angle, which is one of the constraints, has been avoided.

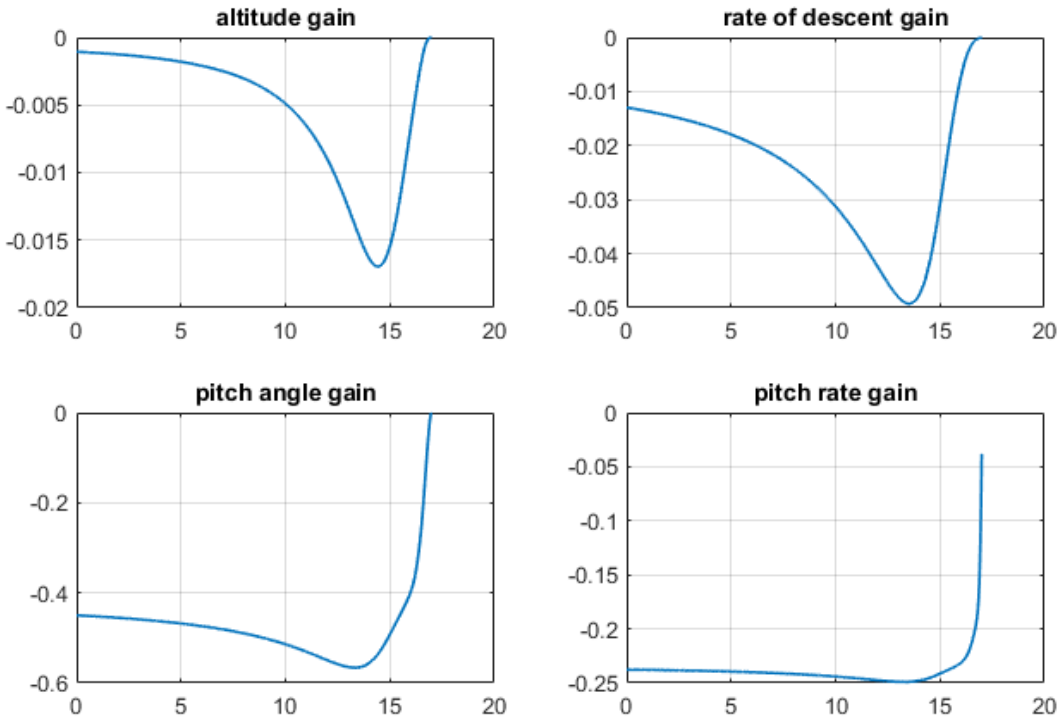

Figure 7: Controller gains for the simulation results

No	Constraint	Constraint validation based on simulated flight data
C1	Exponential (smooth) trajectory	Satisfied by trajectory design
C2	$60 \leq \left\|\dot{h}^{d}(t_f)\right\| \leq 180\ ft/min$	$\left\|\dot{h}^{d}(t_f)\right\| = 102.6\ ft/min$
C3	$0^{o} \leq \theta(t_f) \leq +10^{0}$	$\theta(t_f) \approx 0$
C4	$\alpha(t) < 18^{0}$ $\Delta\alpha(t) < 3.6^{0}$	Satisfied through $\theta \approx 0$ and $\dot{\theta} \approx 0$
C4	$-35^{0} \leq \delta_e(t) \leq +15^{0}$	$-6.124^{0} \leq \delta_e(t) \leq +0.15^{0}$

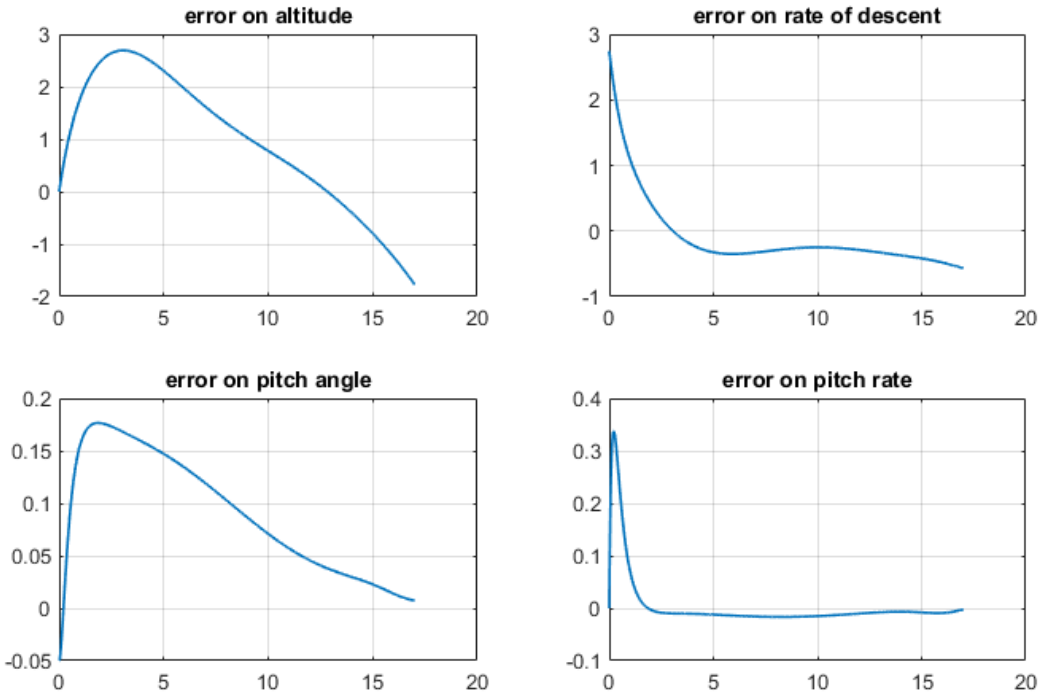

Figure 8: Error between desired and simulated states

Conclusion

In this project, we have designed an optimal linear quadratic tracking controller that will guide an aircraft to have a smooth and comfortable landing. The kinematic model of the aircraft has been used to design the controller. The approach plate that we used in this project to extract the landing parameters is Charlotte Douglas international airport, runway 18L. Different flight constraints has been considered to design a reference trajectory and a landing controller for the flare out phase. The designed LQT controller has been demonstrated to satisfy all the performance indices and constraints while tracking the desired reference trajectory.

References

[1] S. Gudeta and A. Karimoddini, "Design of a Smooth Landing Trajectory Tracking System for a Fixed-wing Aircraft," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 5674–5679, doi: 10.23919/ACC.2019.8814912.

Appendix

```
%%%%%%%%%%%% Morth Carolina A&T State University %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Optimal Control Final Project %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Abenezer Taye %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% May 9, 2020 %%%%%%%%%%%%%%


clear all; close all; clc
 Parameters % Load the given parameters
 %sinit  = zeros(10,1); % S(tf) = C'PC, where C is eye(4). And P=[0.9 0 0 0;0
0.01 0 0;0 0 1 0;0 0 0 1]
 s0 = [0.9 0 0 0 0.042 0 0 1 0 1];
 v0 = [0,0.000142,0,0]';

 %First solve for S using the DRE
 [t1,s] = ode45(@DRE_S,[-Tf:0.01: 0],s0);
 sf = flipud(s);

 %calculate K using K = inv(R)*B'*S(t)
 for i =1:length(s)
     Kf(i,:)= (1/r)*B'*[sf(i,1) sf(i,2) sf(i,3) sf(i,4);
                        sf(i,2) sf(i,5) sf(i,6) sf(i,7);
                        sf(i,3) sf(i,6) sf(i,8) sf(i,9);
                        sf(i,4) sf(i,7) sf(i,9) sf(i,10)];
 end

 %solve for V using the DRE
 [t2,v] = ode45(@(t,v) DRE_V(t,v,Kf,t1),[-Tf:0.01: 0],v0);


%closed loop simulation
 tt = -flipud(t2);
 vf = flipud(v);
 %calucalte xdot
 [tx, x] = ode45(@(t,x) Xstates(t,x,tt,vf,tt,Kf),[0:0.01:Tf],x0);

 %calculate control input
  for i=1:length(x)
     u(i) = -Kf(i,:)*x(i,:)' + (1/r)*B'*vf(i,:)';
  end

plot(tx,u,'LineWidth',1.2 )
grid on
title('elevator defelection angle')

reference = desired_trajectory(Tf);

subplot(2,2,1)
plot(tx,x(:,1),'LineWidth',1.2 )
hold on
plot(tx,reference(1,:),'LineWidth',1.2 )
grid on
```

```
legend('simulated','reference')
title('altitude')

subplot(2,2,2)
plot(tx,x(:,2),'LineWidth',1.2 )
hold on
plot(tx,reference(2,:),'LineWidth',1.2 )
grid on
legend('simulated','reference')
title('rate of descent')

subplot(2,2,3)
plot(tx,x(:,3),'LineWidth',1.2 )
hold on
plot(tx,reference(3,:),'LineWidth',1.2 )
grid on
legend('simulated','reference')
title('pitch angle')

subplot(2,2,4)
plot(tx,x(:,4),'LineWidth',1.2 )
hold on
plot(tx,reference(4,:),'LineWidth',1.2 )
grid on
legend('simulated','reference')
title('pitch rate')


%
plot(tx,u,'LineWidth',1.2 )
grid on
title('elevator defelection angle')


%%
subplot(2,2,1)
plot(tx,Kf(:,1),'LineWidth',1.2 )
grid on
title('altitude gain')

subplot(2,2,2)
plot(tx,Kf(:,2),'LineWidth',1.2 )
grid on
title('rate of descent gain')

subplot(2,2,3)
plot(tx,Kf(:,3),'LineWidth',1.2 )
grid on
title('pitch angle gain')

subplot(2,2,4)
plot(tx,Kf(:,4),'LineWidth',1.2 )
grid on
title('pitch rate gain')
```

```
%%
subplot(2,2,1)
e1 = x(:,1)-reference(1,:)';
plot(tx,e1,'LineWidth',1.2 )
grid on
title('error on altitude')


subplot(2,2,2)
e2 = x(:,2)-reference(2,:)';
plot(tx,e2,'LineWidth',1.2 )
grid on
title('error on rate of descent')


subplot(2,2,3)
e3 = x(:,3)-reference(3,:)';
plot(tx,e3,'LineWidth',1.2 )
grid on
title('error on pitch angle')

subplot(2,2,4)
e4 = x(:,4)-reference(4,:)';
plot(tx,e4,'LineWidth',1.2 )
grid on
title('error on pitch rate')
```

Parameters

```
Q = [0.00058 0 0 0;0 0.015 0 0;0 0 182 0;0 0 0 45];

r = 1000;

x0 = [100, -14,-0.05, 0]';
xf = [0,2,0,0]';
Tf = 17;
Ts = 40;
V = 256; %Ask solomon about this value
ws = 1;
%nu = 0.05; %Ask solomon about this value
zeta = 0.5;
ks = -0.95;

a22 = -1/Ts;
a23 = V/Ts;
a42 = (1/(V*(Ts)^2)) - 2*(zeta*ws)/(V*Ts) + (ws^2)/V;
a43 = (2*zeta*ws)/(Ts) - (ws)^2 - 1/((Ts^2));
a44 = (1/Ts) - (2*zeta*ws);
b4 = (ws^2)*ks*Ts;

A = [0 1 0 0;0 a22 a23 0;0 0 0 1;0 a42 a43 a44];
```

```
B = [0 0 0 b4]';
C = eye(4);

function Sd = DRE_S(t,s)
 Parameters % defined parameter values
 Sd = [Q(1,1) - (1444*s(4)^2)/r;
    s(1) - s(2)/40 + (1098456096613335*s(4))/288230376151711744 -
(1444*s(4)*s(7))/r;
    (32*s(2))/5 - (1561*s(4))/1600 - (1444*s(4)*s(9))/r;
    s(3) - (39*s(4))/40 - (1444*s(4)*s(10))/r;

    Q(2,2) + 2*s(2) - s(5)/20 + (1098456096613335*s(7))/144115188075855872 -
(1444*s(7)^2)/r;
    s(3) + (32*s(5))/5 - s(6)/40 - (1561*s(7))/1600 +
(1098456096613335*s(9))/288230376151711744 - (1444*s(7)*s(9))/r;
    s(4) + s(6) - s(7) + (1098456096613335*s(10))/288230376151711744 -
(1444*s(7)*s(10))/r;

    Q(3,3) + (64*s(6))/5 - (1561*s(9))/800 - (1444*s(9)^2)/r;
    (32*s(7))/5 + s(8) - (39*s(9))/40 - (1561*s(10))/1600 -
(1444*s(9)*s(10))/r;
    Q(4,4) + 2*s(9) - (39*s(10))/20 - (1444*s(10)^2)/r];
end

function Vd = DRE_V(t,v,k,t1)
Parameters % defined parameter values
tf=17;
% reference=(ones(4,1)*(f*t));
[reference] = desired_trajectory(tf);
k1 = interp1(t1,k(:,1),t,'linear','extrap'); % Interpolate the data set
(ft,f) at time t
k2 = interp1(t1,k(:,2),t,'linear','extrap');
k3 = interp1(t1,k(:,3),t,'linear','extrap');
k4 = interp1(t1,k(:,4),t,'linear','extrap');
% Evaluate ODE at time t




%     Vd =                                          [Q(1,1)*reference(1) +
38*v(4)*conj(k1);
%  v(1) - v(2)/40 + Q(2,2)*reference(2) + v(4)*(38*conj(k2) +
1098456096613335/288230376151711744);
%                               (32*v(2))/5 + Q(3,3)*reference(3) +
v(4)*(38*conj(k3) - 1561/1600);
%                                        v(3) + Q(4,4)*reference(4) +
v(4)*(38*conj(k4) - 39/40)];
%
     Vd =                                          [Q(1,1)*reference(1) +
38*v(4)*(k1);
 v(1) - v(2)/40 + Q(2,2)*reference(2) + v(4)*(38*(k2) +
1098456096613335/288230376151711744);
                             (32*v(2))/5 + Q(3,3)*reference(3) +
v(4)*(38*(k3) - 1561/1600);
```

```
                                        v(3) + Q(4,4)*reference(4) +
v(4)*(38*(k4) - 39/40)];


end

function dx = Xstates(t,x,tv,v,tk,K)
Parameters % defined parameter values
% interpolate data points
k1 = interp1(tk,K(:,1),t,'linear','extrap');
k2 = interp1(tk,K(:,2),t,'linear','extrap');
k3 = interp1(tk,K(:,3),t,'linear','extrap');
k4 = interp1(tk,K(:,4),t,'linear','extrap');
v1 = interp1(tv,v(:,1),t,'linear','extrap');
v2 = interp1(tv,v(:,2),t,'linear','extrap');
v3 = interp1(tv,v(:,3),t,'linear','extrap');
v4 = interp1(tv,v(:,4),t,'linear','extrap');
% calculate dx


       dx =           [x(2);
                     (32*x(3))/5 - x(2)/40;
                     x(4);
 (1098456096613335*x(2))/288230376151711744 - (1561*x(3))/1600 - (39*x(4))/40
+ 38*k1*x(1) + 38*k2*x(2) + 38*k3*x(3) + 38*k4*x(4) + (1444*v4)/r];



end
```