function [F, M] = controller(t, state, des\_state, params)

%CONTROLLER Controller for the quadrotor

%

% state: The current state of the robot with the following fields:

% state.pos = [x; y; z], state.vel = [x\_dot; y\_dot; z\_dot],

% state.rot = [phi; theta; psi], state.omega = [p; q; r]

%

% des\_state: The desired states are:

% des\_state.pos = [x; y; z], des\_state.vel = [x\_dot; y\_dot; z\_dot],

% des\_state.acc = [x\_ddot; y\_ddot; z\_ddot], des\_state.yaw,

% des\_state.yaw\_dot

%

% params: robot parameters

% Using these current and desired states, you have to compute the desired

% controls

% =================== Your code goes here ===================

% mass: 0.1800

% I: [3x3 double]

% invI: [3x3 double]

% gravity: 9.8100

% arm\_length: 0.0860

% minF: 0

% maxF: 3.5316

g = params.gravity;

m = params.mass;

% Thrust

%Derivative constant

Kdz = 400;

%Proportion constant

Kpz = 800;

%Current values

z = state.pos(3);

z\_dot = state.vel(3);

%Desired values

z\_t = des\_state.pos(3);

z\_dot\_t = des\_state.vel(3);

z\_ddot\_t = des\_state.acc(3);

%Compute thrust

F = m\*(g + z\_t + Kdz\*(z\_dot\_t-z\_dot) + Kpz\*(z\_t-z));

% Moment

%Constants

Kdx = 100;

Kpx = 30;

Kdy = 100;

Kpy = 30;

%Current state

rx = state.pos(1);

rx\_dot = state.vel(1);

ry = state.pos(2);

ry\_dot = state.vel(2);

%Desired state

rx\_des = des\_state.pos(1);

rx\_dot\_des = des\_state.vel(1);

rx\_ddot\_des = des\_state.acc(1);

ry\_des = des\_state.pos(2);

ry\_dot\_des = des\_state.vel(2);

ry\_ddot\_des = des\_state.acc(2);

%-------------

%Current roll, pitch and yaw

phi = state.rot(1);

theta = state.rot(2);

sigh = state.rot(3);

%Desired roll, pitch and yaw

phi\_des = 0; %Unused

theta\_des = 0; %Unused

sigh\_des = des\_state.yaw;

%-------------

%Commanded State

rx\_ddot\_c =rx\_ddot\_des + Kdx\*(rx\_dot\_des-rx\_dot)+ Kpx\*(rx\_des-rx);

ry\_ddot\_c =ry\_ddot\_des + Kdy\*(ry\_dot\_des-ry\_dot)+ Kpy\*(ry\_des-ry);

%--- ---

%Commanded roll, pitch and yaw

%phi\_c = %Use eqn 14-a from handout

%theta\_c = %Use eqn 14-b from handout

phi\_c = (1/g)\*(rx\_ddot\_c\*sin(sigh\_des)-ry\_ddot\_c\*cos(sigh\_des));

theta\_c = (1/g)\*(rx\_ddot\_c\*cos(sigh\_des)+ry\_ddot\_c\*sin(sigh\_des));

sigh\_c = sigh\_des;

%--- ---

%Constants

Kdphi = 0.2;

Kpphi = 60;

Kdtheta = 0.2;

Kptheta = 60;

Kdsigh = 0.2;

Kpsigh = 60;

%Current Angular velocities

p = state.omega(1);

q = state.omega(2);

r = state.omega(3);

%Desired Angular velocities

p\_des = 0;

q\_des = 0;

r\_des = des\_state.yawdot;

%Comanded Angular velocities

p\_c = 0;

q\_c = 0;

r\_c = r\_des;

%Compute moment elements

u2\_11 = (Kpphi\*(phi\_c-phi)+Kdphi\*(p\_c-p));

u2\_21 = (Kptheta\*(theta\_c-theta)+Kdtheta\*(q\_c-q));

u2\_31 = (Kpsigh\*(sigh\_c-sigh)+Kdsigh\*(r\_c-r));

M = [u2\_11;u2\_21;u2\_31];

% =================== Your code ends here ===================

end