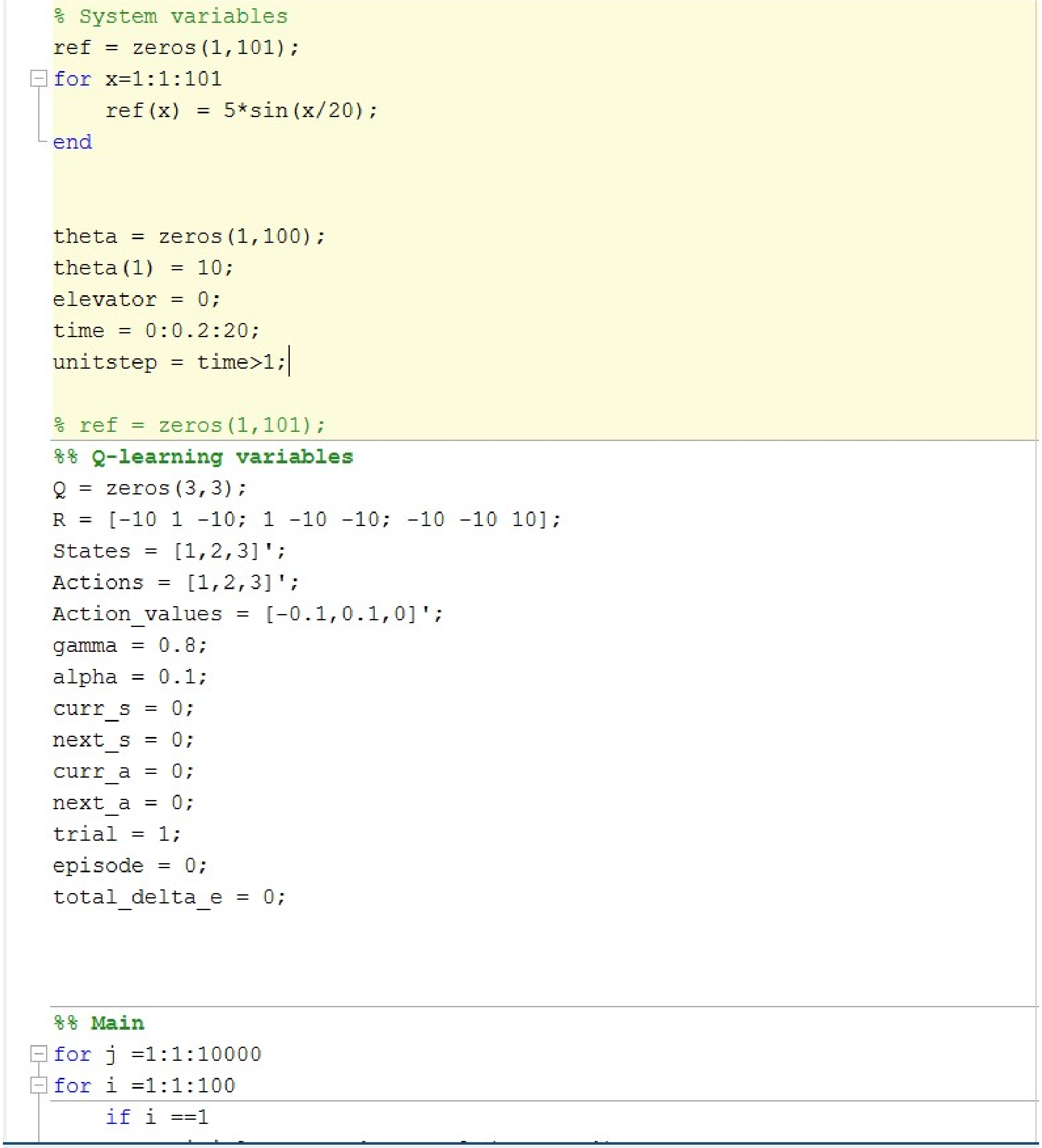
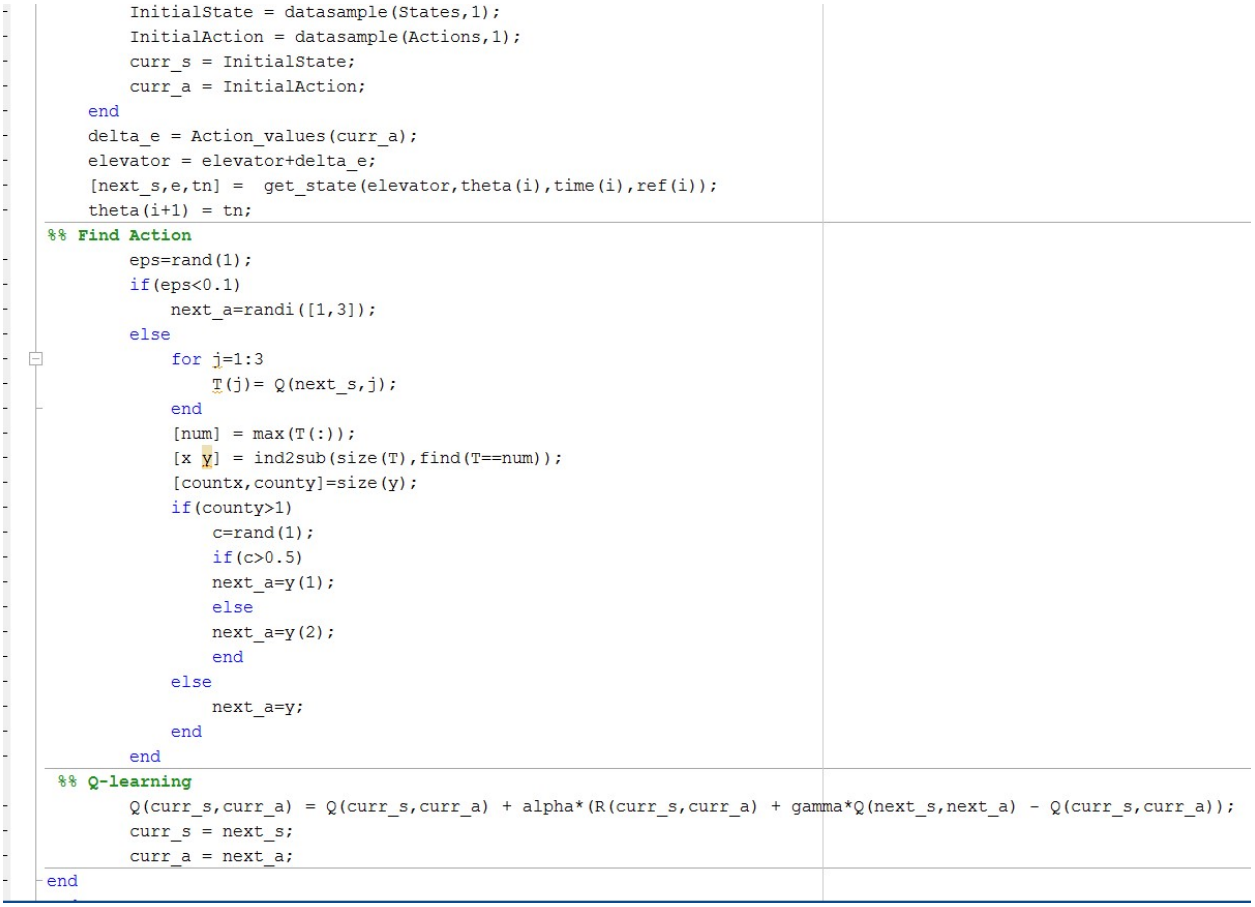
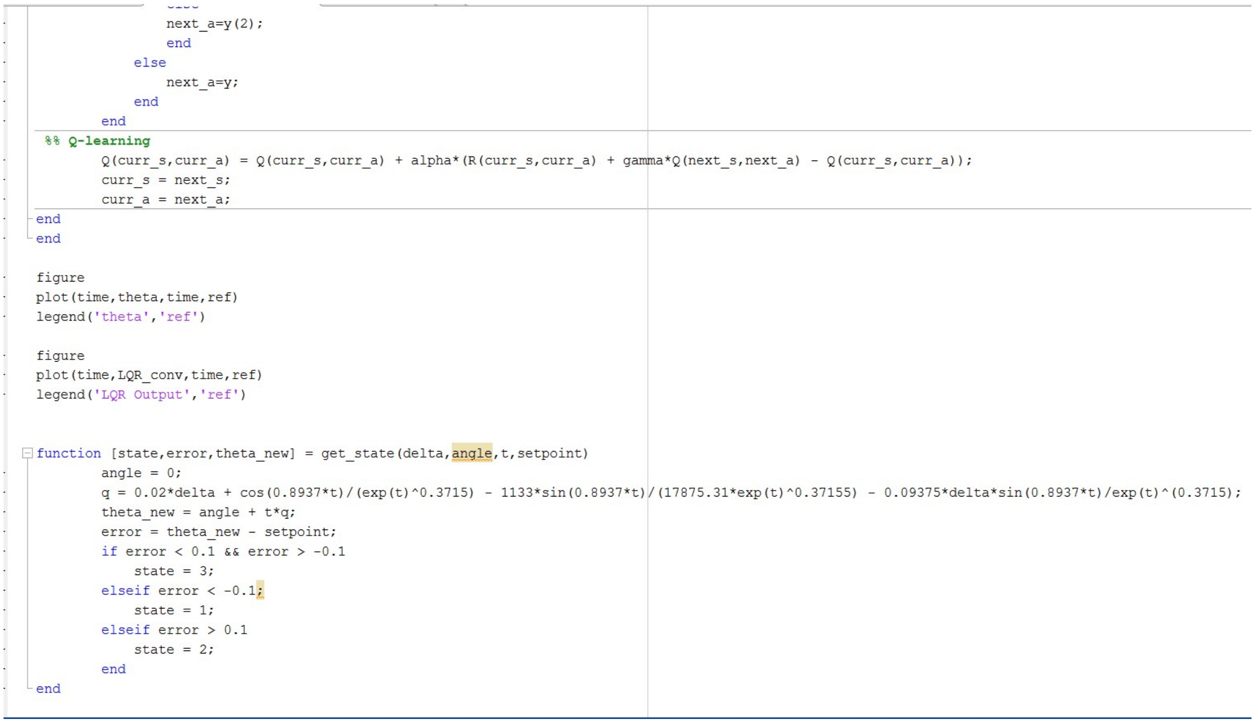
APPENDIX:



```
% System variables
ref = zeros(1,101);
for x=1:1:101
    ref(x) = 5*sin(x/20);
end


theta = zeros(1,100);
theta(1) = 10;
elevator = 0;
time = 0:0.2:20;
unitstep = time>1;

% ref = zeros(1,101);
%% Q-learning variables
Q = zeros(3,3);
R = [-10 1 -10; 1 -10 -10; -10 -10 10];
States = [1,2,3]';
Actions = [1,2,3]';
Action_values = [-0.1,0.1,0]';
gamma = 0.8;
alpha = 0.1;
curr_s = 0;
next_s = 0;
curr_a = 0;
next_a = 0;
trial = 1;
episode = 0;
total_delta_e = 0;



%% Main
for j =1:1:10000
for i =1:1:100
    if i ==1
```



```
        InitialState = datasample(States,1);
        InitialAction = datasample(Actions,1);
        curr_s = InitialState;
        curr_a = InitialAction;
    end
    delta_e = Action_values(curr_a);
    elevator = elevator+delta_e;
    [next_s,e,tn] =  get_state(elevator,theta(i),time(i),ref(i));
    theta(i+1) = tn;
%% Find Action
        eps=rand(1);
        if(eps<0.1)
            next_a=randi([1,3]);
        else
            for j=1:3
                T(j)= Q(next_s,j);
            end
            [num] = max(T(:));
            [x y] = ind2sub(size(T),find(T==num));
            [countx,county]=size(y);
            if(county>1)
                c=rand(1);
                if(c>0.5)
                next_a=y(1);
                else
                next_a=y(2);
                end
            else
                next_a=y;
            end
        end
 %% Q-learning
        Q(curr_s,curr_a) = Q(curr_s,curr_a) + alpha*(R(curr_s,curr_a) + gamma*Q(next_s,next_a) - Q(curr_s,curr_a));
        curr_s = next_s;
        curr_a = next_a;
end
```



```
                next_a=y(2);
                end
            else
                next_a=y;
            end
        end
 %% Q-learning
        Q(curr_s,curr_a) = Q(curr_s,curr_a) + alpha*(R(curr_s,curr_a) + gamma*Q(next_s,next_a) - Q(curr_s,curr_a));
        curr_s = next_s;
        curr_a = next_a;
end
end

figure
plot(time,theta,time,ref)
legend('theta','ref')

figure
plot(time,LQR_conv,time,ref)
legend('LQR Output','ref')


function [state,error,theta_new] = get_state(delta,angle,t,setpoint)
        angle = 0;
        q = 0.02*delta + cos(0.8937*t)/(exp(t)^0.3715) - 1133*sin(0.8937*t)/(17875.31*exp(t)^0.37155) - 0.09375*delta*sin(0.8937*t)/exp(t)^(0.3715);
        theta_new = angle + t*q;
        error = theta_new - setpoint;
        if error < 0.1 && error > -0.1
            state = 3;
        elseif error < -0.1;
            state = 1;
        elseif error > 0.1
            state = 2;
        end
end
```