ROB 535 Control Project Team2 Documentation

Siyuan Yin
yinsy@umich.edu

1. Task Definition

The control Project has 2 tasks, which require us to use a bicycle model to track a given trajectory and avoid some random obstacles.

- **Task 1** We are required to design a controller for the bicycle model to get from the beginning to the end of a given track.
- **Task 2** We are required to design a controller which could help the bicycle avoid random-generated obstacles on the basis of task1.

2. My Contribution

My main contribution is task1, which is to generate the input for the model to get to the end of the track, and the input could also be used as the reference for task2's MPC controller.

2.1 Input generalization using fmincon

First I tried to generate the input along with the corresponding state using fmincon function, which is similar to that used in Assignment2. Track boundaries are used as the boundary constraints and distance to the end point are used as the cost functions, and Euler integration was implemented to construct the nonlinear constraint function. However, the program got stuck when it computed fmincon, and we think this is due to the large dimension of the constraint which may cost too much time to run the program

2.2 Input generalization using PID controller

We think PID controller is a more reliable choice and turn to it later. At each step the controller will detect whether the reference point on the trajectory is on the left or on the right of current position, and the distance will be the control error for a PID controller to generate the control input for the next step. We successfully generate the input for task1 and it completes the tracking task. The input and reference state could also be used for task 2.

3. Teammates' Work

- **Ziqi Han** Control part for task2, the implementation of MPC
- **Yifan Wang** Attempt to use reinforcement learning for task1 and task2(not being chosen eventually). PID controller for task1.
- **Qilin He** Attempt to use fmincon for task1. The implementation of MPC
- **Yuzhou Chen** Attempt to use PID for task2 (not being chosen eventually due to collision risk)

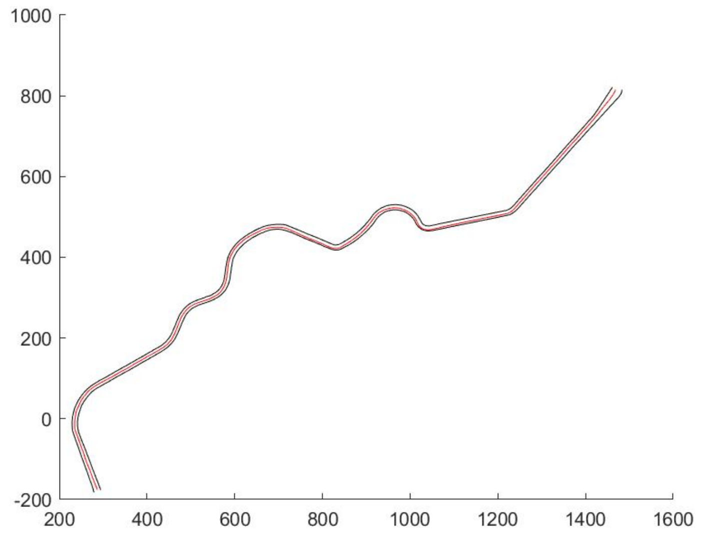
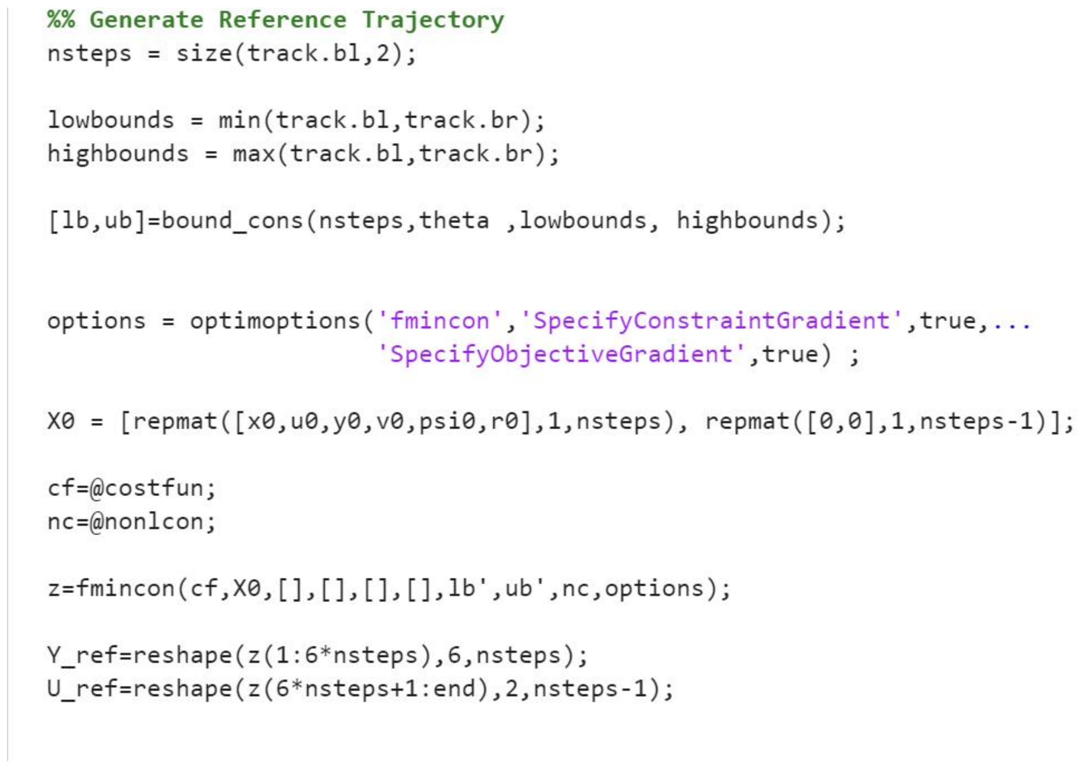
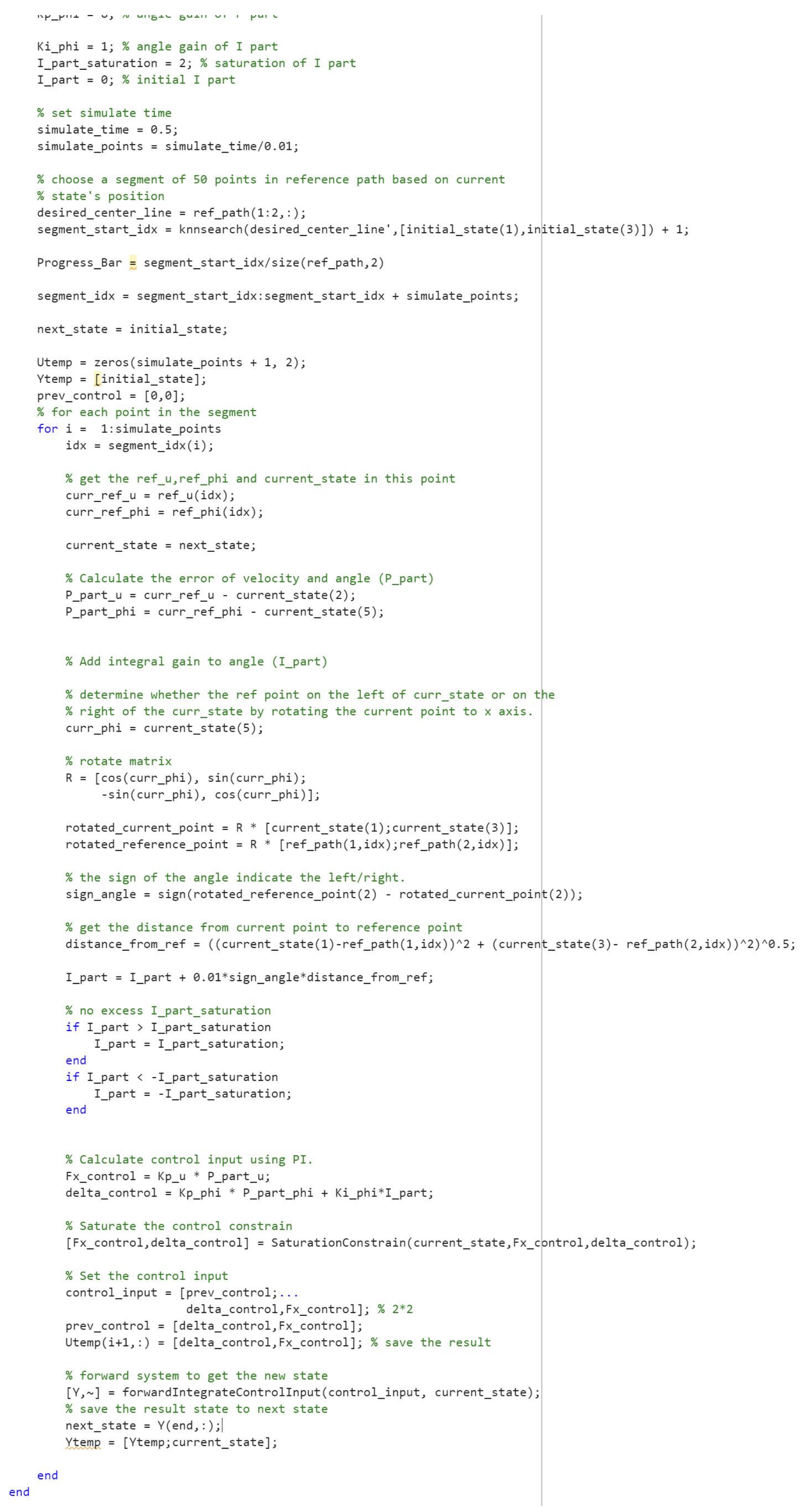


figure1. Trajectory for task1 using generated input



```
%% Generate Reference Trajectory
nsteps = size(track.bl,2);

lowbounds = min(track.bl,track.br);
highbounds = max(track.bl,track.br);

[lb,ub]=bound_cons(nsteps,theta ,lowbounds, highbounds);


options = optimoptions('fmincon','SpecifyConstraintGradient',true,...
                       'SpecifyObjectiveGradient',true) ;

X0 = [repmat([x0,u0,y0,v0,psi0,r0],1,nsteps), repmat([0,0],1,nsteps-1)];

cf=@costfun;
nc=@nonlcon;

z=fmincon(cf,X0,[],[],[],[],lb',ub',nc,options);

Y_ref=reshape(z(1:6*nsteps),6,nsteps);
U_ref=reshape(z(6*nsteps+1:end),2,nsteps-1);
```

figure2. Generate reference input and state using fmincon



```
    Kp_phi = 8; % angle gain of P part

    Ki_phi = 1; % angle gain of I part
    I_part_saturation = 2; % saturation of I part
    I_part = 0; % initial I part

    % set simulate time
    simulate_time = 0.5;
    simulate_points = simulate_time/0.01;

    % choose a segment of 50 points in reference path based on current
    % state's position
    desired_center_line = ref_path(1:2,:);
    segment_start_idx = knnsearch(desired_center_line',[initial_state(1),initial_state(3)]) + 1;

    Progress_Bar = segment_start_idx/size(ref_path,2)

    segment_idx = segment_start_idx:segment_start_idx + simulate_points;

    next_state = initial_state;

    Utemp = zeros(simulate_points + 1, 2);
    Ytemp = [initial_state];
    prev_control = [0,0];
    % for each point in the segment
    for i =  1:simulate_points
        idx = segment_idx(i);

        % get the ref_u,ref_phi and current_state in this point
        curr_ref_u = ref_u(idx);
        curr_ref_phi = ref_phi(idx);

        current_state = next_state;

        % Calculate the error of velocity and angle (P_part)
        P_part_u = curr_ref_u - current_state(2);
        P_part_phi = curr_ref_phi - current_state(5);


        % Add integral gain to angle (I_part)

        % determine whether the ref point on the left of curr_state or on the
        % right of the curr_state by rotating the current point to x axis.
        curr_phi = current_state(5);

        % rotate matrix
        R = [cos(curr_phi), sin(curr_phi);
             -sin(curr_phi), cos(curr_phi)];

        rotated_current_point = R * [current_state(1);current_state(3)];
        rotated_reference_point = R * [ref_path(1,idx);ref_path(2,idx)];

        % the sign of the angle indicate the left/right.
        sign_angle = sign(rotated_reference_point(2) - rotated_current_point(2));

        % get the distance from current point to reference point
        distance_from_ref = ((current_state(1)-ref_path(1,idx))^2 + (current_state(3)- ref_path(2,idx))^2)^0.5;

        I_part = I_part + 0.01*sign_angle*distance_from_ref;

        % no excess I_part_saturation
        if I_part > I_part_saturation
            I_part = I_part_saturation;
        end
        if I_part < -I_part_saturation
            I_part = -I_part_saturation;
        end


        % Calculate control input using PI.
        Fx_control = Kp_u * P_part_u;
        delta_control = Kp_phi * P_part_phi + Ki_phi*I_part;

        % Saturate the control constrain
        [Fx_control,delta_control] = SaturationConstrain(current_state,Fx_control,delta_control);

        % Set the control input
        control_input = [prev_control;...
                         delta_control,Fx_control]; % 2*2
        prev_control = [delta_control,Fx_control];
        Utemp(i+1,:) = [delta_control,Fx_control]; % save the result

        % forward system to get the new state
        [Y,~] = forwardIntegrateControlInput(control_input, current_state);
        % save the result state to next state
        next_state = Y(end,:);
        Ytemp = [Ytemp;current_state];

    end
end
```

Figure3. Part of PID controller for task1