ROB 535 Control Project Team26 Documentation

Chengtian Zhang
zctchn@umich.edu

1 Task Definition

The control project of controlling a bicycle model mainly contains two parts as follows.

- **Task 1** We are required to design a controller for the system to get from the beginning to the end of a pre-defined track as rapidly as possible;
- **Task 2** We need to develop a control design algorithm (which may or may not modify the controller constructed in the first task) to avoid obstacles that are known only at run-time.

2 My Contribution

MPC adopts a compromise strategy, not only considering the current state, but considering the future finite time domain, sacrificing optimality to a certain extent. I mainly focused on MPC method, which was not used because my system was unable to tackle with task 2 perfectly.

2.1 Task 1

In task 1, the MPC system will periodically accept the state quantity state, inputs the MPC model, and calculates the optimal control solution based on model, cost and constraints.
General working steps of MPC can be summarized as follows:

- Build prediction model based on state information, design loss function and constraint function based on model constraints and output requirements;
- Combined with previous information, current state and prediction model, calculate the optimal control solution and predict the n-step system output;
- Output execution;
- Enter next iteration, obtain the detection information, and repeat steps 1, 2 and 3.

Figure 1, 2, 3 shows the calculation and implement for MPC input and state (A_i and B_i) parts.
Figure 4 shows MPC worked well on task 1.

2.2 Task 2

In task 2, I proposed to first run on the left lane and detect obstacle, if the current running lane exists an obstacle, I will switch to another lane. For the left and right reference, I used Zimin Lu's results. Before I could build up a correct system, my teammates investigating PID methods finished their prototype so we just devote together to test and verify the PI method.

3 Teammates' Work

- **Jiachen Jiang** Control project part2
- **Zimin Lu** Control project part1
- **Shuoqi Wang** MPC method
- **Lingfei Huo** Preception task

```
function [ip] = inputpart_hand(Y_vec,U_in)
%Initialization.
m = 1400; N_w = 2.00; f = 0.01; I_z = 2667; a = 1.35; b = 1.45; B_y = 0.27;
C_y = 1.2; D_y = 0.70; E_y = -1.6; g = 9.806;

X = Y_vec(1); u = Y_vec(2); Y = Y_vec(3); v = Y_vec(4); psi = Y_vec(5); r = Y_vec(6);
Fx = U_in(1); delta_f = U_in(2);

%Partial Derivatives of phi_yf.
alpha_f = delta_f - atan((v+a*r)/u); alpha_r = - atan((v-b*r)/u);
dphi_yf = [0, 1 - E_y + E_y/(1+(B_y*alpha_f)^2)];

%Partial Derivatives of F_yf.
phi_yf = (1 - E_y)*alpha_f + E_y*atan(B_y*alpha_f)/B_y;
F_zf = b*m*g/(a+b); F_yf = F_zf*D_y*sin(C_y*atan(B_y*phi_yf));
dF_yf = [0, F_zf*D_y*cos(C_y*atan(B_y*phi_yf))*C_y*(1/(1+(B_y*phi_yf)^2))*B_y*dphi_yf(2)];

%Compute Partial Derivatives for Jacobian Matrix.
ip = zeros(6,2);
ip(2,:) = [N_w/m, -sin(delta_f)*dF_yf(2)/m - cos(delta_f)*F_yf/m];
ip(4,:) = [0, cos(delta_f)*dF_yf(2)/m - F_yf*sin(delta_f)/m];
ip(6,:) = [0, a*cos(delta_f)*dF_yf(2)/I_z - a*F_yf*sin(delta_f)/I_z];
end
```

Figure 1: input part by hand

```
function [pd] = statepart_hand(Y_vec,U_in)
%Initialization
m = 1400; N_w = 2.00; f = 0.01; I_z = 2667; a = 1.35; b = 1.45; B_y = 0.27;
C_y = 1.2; D_y = 0.70; E_y = -1.6; g = 9.806;
X = Y_vec(1); u = Y_vec(2); Y = Y_vec(3); v = Y_vec(4); psi = Y_vec(5); r = Y_vec(6);
Fx = U_in(1); delta_f = U_in(2);

%Partial Derivatives of alpha_f and alpha_r.
temp1 = -1/(1 + ((v+a*r)/u)^2); temp2 = -1/(1 + ((v-b*r)/u)^2);
dalpha_f = [0, temp1*-(v+a*r)/u^2, 0, temp1/u, 0, temp1*a/u];
dalpha_r = [0, temp2*-(v-b*r)/u^2, 0, temp2/u, 0, temp2*-b/u];

%Partial Derivatives of phi_yf and phi_yr.
alpha_f = delta_f - atan((v+a*r)/u); alpha_r = - atan((v-b*r)/u);
temp3 = E_y/(1 + (B_y*alpha_f)^2); temp4 = E_y/(1 + (B_y*alpha_r)^2);
dphi_yf = [0, (1 - E_y + temp3)*dalpha_f(2), 0, (1 - E_y + temp3)*dalpha_f(4) ...
    0, (1 - E_y + temp3)*dalpha_f(6)];
dphi_yr = [0, (1 - E_y + temp4)*dalpha_r(2), 0, (1 - E_y + temp4)*dalpha_r(4) ...
    0, (1 - E_y + temp4)*dalpha_r(6)];
```

Figure 2: state part by hand 1

```
%Partial Derivatives of F_yf and F_yr.
phi_yf = (1 - E_y)*alpha_f + E_y*atan(B_y*alpha_f)/B_y;
phi_yr = (1 - E_y)*alpha_r + E_y*atan(B_y*alpha_r)/B_y;
F_zf = b*m*g/(a+b); F_yf = F_zf*D_y*sin(C_y*atan(B_y*phi_yf));
F_zr = a*m*g/(a+b); F_yr = F_zr*D_y*sin(C_y*atan(B_y*phi_yr));
temp5 = F_zf*D_y*cos(C_y*atan(B_y*phi_yf))*C_y*(1/(1+(B_y*phi_yf)^2))*B_y;
temp6 = F_zr*D_y*cos(C_y*atan(B_y*phi_yr))*C_y*(1/(1+(B_y*phi_yr)^2))*B_y;
dF_yf = [0, temp5*dphi_yf(2), 0, temp5*dphi_yf(4), 0, temp5*dphi_yf(6)];
dF_yr = [0, temp6*dphi_yr(2), 0, temp6*dphi_yr(4), 0, temp6*dphi_yr(6)];

%Partial Derivatives for Jacobian Matrix.
pd = zeros(6);
temp7 = a*cos(delta_f)/I_z;
pd(1,:) = [0, cos(psi), 0, -sin(psi), -u*sin(psi)-v*cos(psi), 0];
pd(2,:) = [0, -sin(delta_f)/m*dF_yf(2), 0, -sin(delta_f)/m*dF_yf(4)+r, ...
    0, -sin(delta_f)/m*dF_yf(6)+v];
pd(3,:) = [0, sin(psi), 0, cos(psi), u*cos(psi)-v*sin(psi), 0];
pd(4,:) = [0, (cos(delta_f)*dF_yf(2)+dF_yr(2))/m - r, 0, ...
    (cos(delta_f)*dF_yf(4)+dF_yr(4))/m, 0, (cos(delta_f)*dF_yf(6)+dF_yr(6))/m - u];
pd(5,:) = [0, 0, 0, 0, 0, 1];
pd(6,:) = [0, temp7*dF_yf(2) - b*dF_yr(2)/I_z, 0, temp7*dF_yf(4) - b*dF_yr(4)/I_z, ...
    0, temp7*dF_yf(6) - b*dF_yr(6)/I_z];
end
```

Figure 3: state part by hand 2

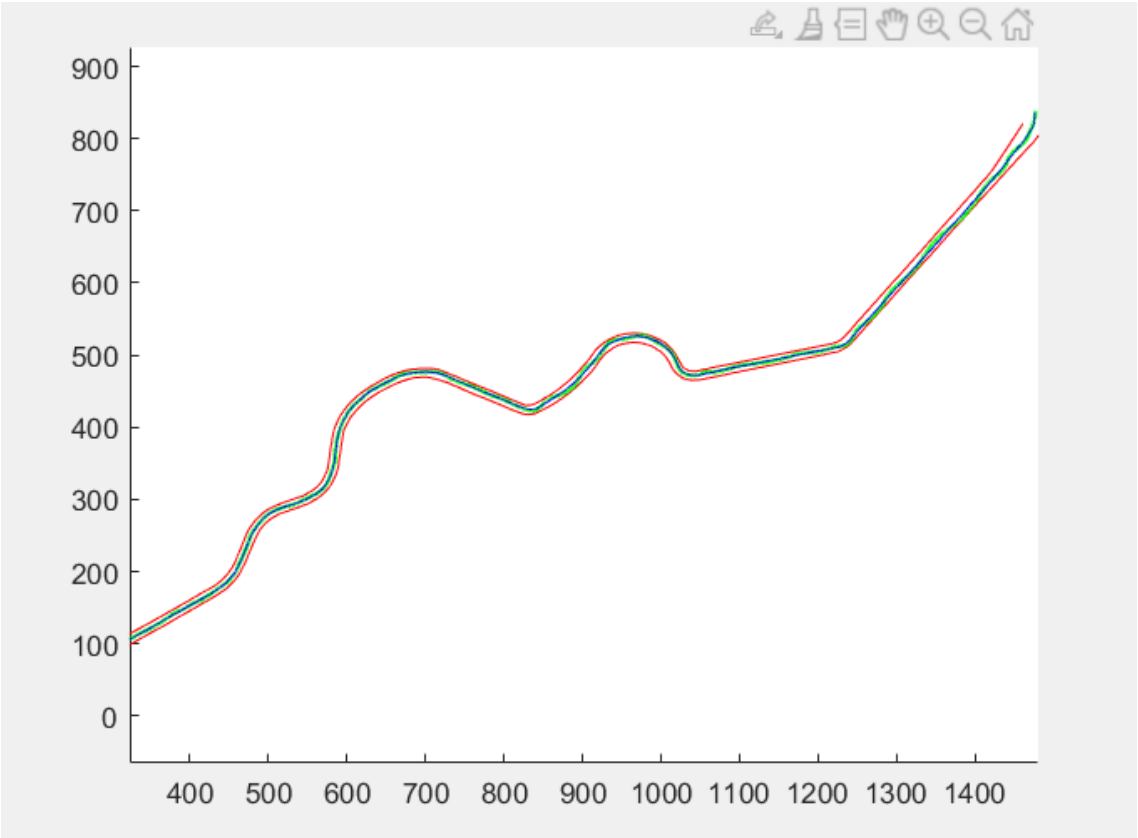

Figure 4: MPC implement on Task 1