CarND MPC Project

Udacity Self Driving Car Nano Degree, Term 2, Project 5

Project

The goal of this project is to build a MPC (Model Predictive Control) Controller in C++ and drive a vehicle in a simulator environment. The MPC controller and the simulator communicate with each other using WebSocket. The simulator provides the car telemetry data. The MPC controller calculates the cross track error (cte) and sends the steering angle and throttle to the simulator. The simulator applies the steering angle and the throttle to the car to drive it around the track.

Project Steps

- Implement the MPC controller including the setting constraints, cost function and variables.
- Fit a reference (yellow line) based on the road way points provided by the simulator.
- Evaluate the current state based on the reference line.
- Calculate the actuator values, steering angle and throttle based on the MPC calculations.
- Account for latency of the actuators (100 ms).
- Test the solution on the simulator.
- Tune the coefficients of the cost function to make the car drive smoothly around the track.

Reflection

The Model

There are multiple vehicle models for our consideration. The MPC project uses the Kinematic model which ignores the tire forces, gravity and mass. The Kinematic model is represented by the following state equations:

- `x[t+1] = x[t] + v[t] * cos(psi[t]) * dt`
- `y[t+1] = y[t] + v[t] * sin(psi[t]) * dt`
- `psi[t+1] = psi[t] + v[t] / Lf * delta[t] * dt`
- `v[t+1] = v[t] + a[t] * dt`
- `cte[t+1] = f(x[t] - y[t] + v[t] * sin(epsi[t] * dt`
- `epsi[t+1] = psi[t] = psides[t] + v[t] * delta[t] / Lf * dt`

where:

x, y → Car’s position coordinates

psi → Car’s heading direction

v → Car’s velocity

cte → Cross track error

epsi → Heading error

Lf → Distance between the center of gravity of the car and the front wheels.
The value of Lf is assumed to be 2.67

a → Car’s throttle

delta → Steering angle

Timestep Length (N) and Elapsed Duration (dt)

The prediction horizon is determined by the number of points (N) and the time interval (dt). Since the values of car’s throttle (a) and steering angle (delta) are calculated during every time interval, there is no need to keep the value of N very high. A high value of N also affects the controller’s performance in terms the time taken to compute the values.
Code reference: MPC.cpp, lines: 9 - 10

The various values of N and dt I tried and the results are shown in the following table.

N	**dt (milliseconds)**	**Result**
20	500	The car went off the track.
20	300	The car went off the track.
20	100	The car stayed on the track longer but was weaving from one side of the road to the other. Eventually, car went off the track.
10	100	The car stayed on the track. Submitted values.

Since there is no need to keep the value of N high and based on the experimental value shown in the table, I settled on the value of N = 10 and dt = 0.1 (100 milliseconds).

Polynomial Fitting and MPC Preprocessing

- **Reference Transformation**
  The waypoints provided by the simulator are transformed to the car's perspective.
  That is, the values of px, py and psi are assumed to be 0, to keep the calculations simple.
  Code reference: main.cpp, lines: 107 – 117.

- **Polynomial Fit**
  A 3rd-degree polynomial is fitted to the transformed waypoints. These polynomial coefficients are to compute the cte and epsi. The coefficients are also used by the MPC solver to create a reference trajectory.
  Code reference: main.cpp, lines: 125 – 135.

Model Predictive Control with Latency

The MPC project requires the MPC controller to handle 100 millisecond latency. Since the elapsed duration, dt is chosen as 100 milliseconds, a very simple approach is implemented to handle latency. The actuations for steering angle and throttle are applied an additional timestep later.
Code reference: MPC.cpp, lines: 128 - 131

Tuning the Cost Function Coefficients

I started with the reference velocity, ref_v (MPC.cpp, line 26) set to 40 mph. By tuning the cost function coefficients (MPC.cpp, lines: 64 – 85), I was able to make the care go around the track without any problem. Then, I increased the reference velocity to 70 mph. With this velocity, I was able to get a maximum speed of around 55 mph. For the final submission, I increased the reference velocity to 100 mph and tuned the cost function coefficients. With the submitted cost function coefficient values, the car is able to reach a maximum speed of 78 mph.