MPC Project Reflection

Jakrin Juangbhanich

Overview

The MPC Controller actuates a vehicle based on cost optimization. The vehicle's desired waypoint is used as an input, and is measured against the vehicle's projected trajectory over a certain time. A cost is calculated from this information, and the ‘path’ that has the least cost will be chosen for the next actuation input.

I have broken up the State, Motion Model, and Cost Modules into separate entities in order to think about them more clearly about each piece, and also be modular. In a real world situation I think this will allow me more freedom to experiment with different motion models, states, and cost functions.

The Model

This project assumes a 2D kinematic motion model, on the x and y axis. The state includes the position of the vehicle, its yaw angle, velocity, and actuators (steering and acceleration). The waypoints are given as input from the simulator, and my program will first transform these map waypoints into local space relative to the car (translating and rotating).

When calculating the cost, the kinematic motion model is applied to the state, using the velocity and yaw values to determine the x and y trajectory. The actuators are also taken into account, and will affect the yaw and velocity values in this time period too. The actuators have a limit, between 1.0 and -1.0 for the acceleration, and 25 degrees in either direction for the steering.

Timestep Length and Elapsed Duration (N & dt)

I have experimented with various time-steps for N and dt. The final values I have chosen are:

N = 8
dt = 0.15s

I arrived at these values through experimentation, but there is also some intuition behind these values. Firstly, my T is very small, at 1.2s. If it is much smaller, the vehicle will not handle turns well, because it does not ‘see’ the horizon it is approaching. But if it is too large, I am wasting computation since we only care about the first set of actuations anyway - and the horizon is bound to change every time step.

Common sense tells me to use a low value as possible for dt, because cars must be able to react quickly, especially in the case of an emergency situation (this is also why I did not put a cost on sudden braking actuation).

But lower dt also means needing to have higher N for the same horizon, which means more computations per second. I found a longer dt, say of 0.15s, was still good enough to react to sharp turns.

Additionally, I also found that when the dt is too small, or the N value is too large, the vehicle tends to over-actuate in the first step, in order to minimize the CTE overall as fast as possible. This can of course

be tuned with the cost modules, but in the end I found a smaller N and a larger dt are more effective for handling sharp turns, especially at high speeds.

> *One of the downside of 0.15s dt is that when my vehicle attempts to maintain a lower speed, it constantly overshoots its target speed and then brakes to compensate, instead of smoothly actuating. This could probably be tuned with the costs too, but I did tune my MPC to perform best at high speeds.*

Polynomial Fitting and MPC Preprocessing

I have fit the waypoints to a 3rd order polynomial, and I preprocess the waypoints by converting them from world space to local space. I then create vectors for the independent variables and constraints to be used in the nonlinear solver, and I initialize the first state based on my motion model - where the vehicle is predicted to be after the actuation latency.

Model Predictive Control with Latency

The reference latency I used was 100ms. The model takes this into account, by assuming an initial state in the future based on this latency and current actuation inputs. The biggest problem I had with latency was with the steering. The vehicle tends to oversteer, and then ends up oscillating all over the road trying to correct itself. The most reliable solution for me was to set a high cost for steering actuation, and design the model not to try too harshly to correct its close range CTE.

Conclusion

Despite these measures I still must concede how fragile my model is. Whilst I found some combination of parameters that work very well and is stable on the road, I noticed that very subtle tweaks can result in my vehicle failing catastrophically.

For example, I found that a low velocity reference cost weight lets my vehicle brake to handle sharp turns at high speeds beautifully, but it will cause it to stop if the reference speed is too low. On the other hand, I do not want to add additional logic for detecting turns and modulating the speed. I feel that the concept of this model is good enough to handle these situations, but I just need to have more understanding of the parameters.

If I wanted to improve this further, I could write the cost output to a graph, so that each frame I can see which cost module is the dominant one. I think the key to a good MPC Controller is understanding how the costs interact with each other.