Udacity SDC Nanodegree

Model Predictive Controller Project

Introduction

This project implements the a Model Predictive Controller based on a simplified version of kinematic model, taking into account only the lateral and longitudinal forces experienced by a car. The project makes use of the Udacity's term 2 simulator to validate the program.

The Model Outline

As mentioned above, the implementation is based on the Kinematic model discussed in the lessons. The state of the vehicle consists of the coordinates x & y, the velocity v, Orientation of vehicle psi, the cross track error cte (which is the distance between the vehicles positions and optimal reference trajectory), and the orientation error. These parameters are updated using the following equations:

$$x_{t+1} = x_t + v_t \cos(\psi_t) dt$$

$$y_{t+1} = y_t + v_t \sin(\psi_t) dt$$

$$v_{t+1} = v_t + a_t dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t \delta_t}{L_f} dt$$

$$cte_t = f(x_t) - y_t$$

$$e\psi_{t+1} = e\psi_t + \frac{v_t \delta_t}{L_f} dt$$

The overall flow of the controller is depicted in the figure below:

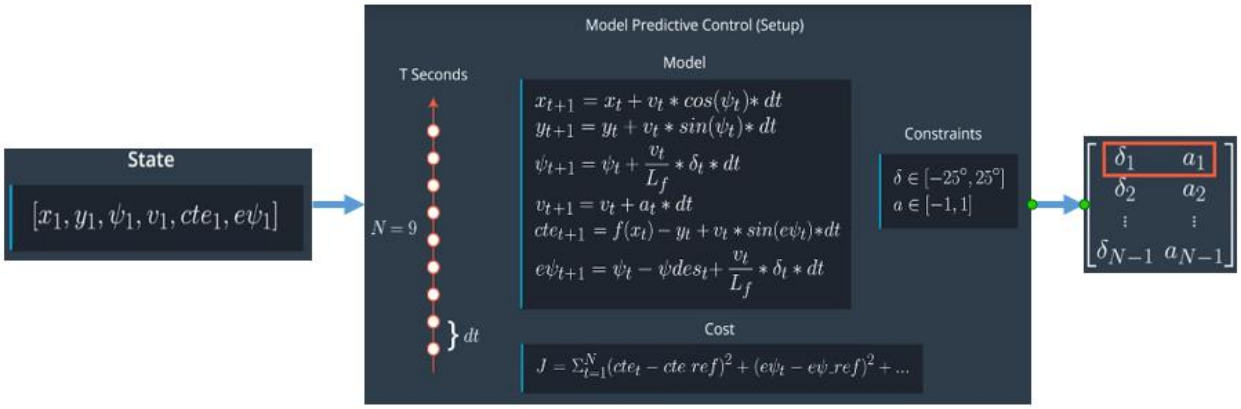

Figure 1 - Credit - Udacity Selfdriving Nano Degree, MPC Lesson 19

The above model takes the vehicle state and calculates a set of desired actuator inputs delta and acceleration (a single actuator that includes vehicle braking for simplicity) based on a cost function that takes into account vehicle constraints such as acceleration, rate of steering, velocity, and a change rate to the actuator inputs for smoothening, to select the optimal inputs to reduce the cost to match the desired trajectory.

Steps involved in the Algorithm:

1. Transform the waypoints generated by the simulator from map coordinate system to the vehicle coordinate system.
2. Calculate coefficients of 3rd degree polynomial fitted to waypoints.
3. Load initial state vector and use the MPC solver to generate actuator inputs with optimal cost.
4.

Timestep Length, Elapsed Duration, and Actuator Latency

I began testing the program without taking into consideration the latency involved in actuators in a vehicle. As for tuning the number timesteps (N) and the time interval between actuations (dt) of the prediction horizon, I started off with an N of 30 with a dt of .04. This resulted in the vehicle driving fairly well on a straight track, but quickly went off course at the first corner of the track.

After taking a look at the project walk through, it made sense having large parameters would result in larger computational needs by predicting too many steps in the future. As an increase in the number of time steps increases the computational needs by calculating actuations for a longer time period, this also makes calculating the cost more computationally intensive. I tried a couple of combinations around N =10 & dt =0.1 as suggested in the walk through. This resulted in the car successfully completing multiple laps with minimal oscillations. I would however like to explore using better constraints on the acceleration actuator to brake a bit smoother. The program was able to attain a top speed of about 95MPH.

I've also implemented the project to model actuator latency to simulate typical real world delays involved with vehicle actuators. My mentor was able to point me in identifying a simple solution that involves predicting the location of the car considering the latency of 100ms. The implementation is present in the update equations for the cars position (x & Y) and orientation. As shown below:

```
x_car = 0.0 + v * 0.1;
psi_car = 0.0 + v * -delta / Lf * 0.1;
```