Model Predictive Control

Kevin La Ra

Timestep and Elapsed Duration

The overall model has several components. The first is to figure out the prediction horizon, or the duration of the predicted trajectory. Then, define the number of points in that trajectory (N) and the time between each (dt).

Ideally, N should be as large as possible and dt should be as small as possible. I set N = 10 to keep the solver from optimizing too many points and set dt = 0.1 so that the car had reasonable time between actuator updates. This computes a trajectory 1 second ahead of the vehicle. It worked well for this track.

In testing values, I tried N=20 with dt=0.05, which resulted in the same 1 second path, but the actuator update was too frequent. The car was too erratic and would not start driving on the track. I also tried N=20, dt =0.1 but again the path kept swinging left and right across the track. This implies that there were too many points to fit. Reducing the number or points to fit to 10 resulted in a smoother diving experience.

Vehicle State

The vehicle state is not complicated. We assume a two dimensional world with position (x,y). The vehicle is pointed in a direction which is signified by the Greek letter ψ, psi. The vehicle also has a velocity, v. This captures the current state.

We also model the control inputs. Delta δ is used to represent the angle of the vehicle front wheels. Acceleration (change in velocity) is denoted by a.

Vehicle State Update Equations

We want to know how the vehicle moves through time. In order to compute a new set of state values, the following vehicle state update equations were created to model the vehicle motion.

Next X Location: $x_{t+1}=x_t+v_t*\cos(\psi_t)*dt$

Next Y Location: $y_{t+1}=y_t+v_t*\sin(\psi_t)*dt$

Next Orientation Angle: $\psi_{t+1}=\psi_t+v_t/L_f*\delta_t*dt$

Next Velocity: $v_{t+1}=v_t+a_t*dt$

The steering angle has the addition factor v_t/L_f to account for two physical properties of turning. Lf measures the distance between the front of the vehicle and its center of gravity. The

larger the vehicle, the slower the rate of turn. Also, at higher speeds, a given turn rate will result in more rapid turns. Velocity it added to this factor to account for this action. The updates location, orientation and speed of the vehicle are determined by geometry and physics. For location, we assume a small dt such that the steering angle is constant.

Vehicle Error Equations

These equations are a way to compute the difference between our desired position and heading to the desired position and heading. Ideally, these should be 0.

Next Cross Track Error: $cte_{t+1}=f(x_t)-y_t+(v_t*\sin(e\psi_t)*dt)$

The distance between the center of the road and the vehicle position on the road.

Next Orientation Error: $e\psi_{t+1}=\psi_t-\psi des_t+(v_t/L_f *\delta_t*dt)$

The difference between the desired orientation and the current orientation.

Vehicle Constraints

The actuators have limitations. The steering angle is between -25/+25 degrees. However, the simulator returns a value between -1/+1, so this must be converted to radians for use in the algorithms. Multiplying the steering input by -0.46332 converted the input value to a radian angle and corrected for the difference in left/right direction between the simulator and the algorithm. The throttle has an input range from -1/+1.

$\delta \in [-25°, 25°]$ $a \in [-1, +1]$

Cost functions

The following cost equations are used in the model by the fit function to find the set of values that minimize the cost. In the first three equations, we subtract a reference value from the measured value. This forces the fit algorithm to try to match the reference value. For example, the speed cost function penalizes the vehicle for not maintaining reference velocity. When it deviates from the reference, the cost increases. For this implementation, the cross-track error and the reference orientation were set to 0.

These are the values that were used to tune the vehicle in the simulator. They are included here as a reference to the cost function. They are used to change the importance and weight of each cost in the final cost value. The absolute value is not important. The relative value between each parameter is what is important.

```
double tuning[7] = {5, 100, 1, 5, 15, 10, 10};

These cost functions minimize the error is the vehicle state
  distance error cost = tuning[0] * (cte_start - ref_cte) ^ 2
  angle error cost    = tuning[1] * (epsi_start - ref_epsi) ^ 2
  speed error cost    = tuning[2] * (v_start - v_ref) ^ 2

These equations minimize the use of the actuators to smooth out motion
  wheel angle cost  = tuning[3] * (delta_start) ^ 2
  acceleration cost = tuning[4] * (a_start) ^ 2

These equations minimize the changes between sequential actuations
  wheel angle change cost = tuning[5] * ([delta_start + 1] - delta_start) ^ 2
  accelerator change cost = tuning[6] * ([a_start + 1] - a_start]) ^ 2
```

Algorithm Processing

Now we start the state feedback loop. First we pass the current state to the model predictive controller. The optimization solver is called which uses the inputs, the model, the constraints and the cost function to return a set of control outputs that minimizes the cost function. This model uses the Ipopt solver.

Actuator Latency

This model also simulates actuator latency by delaying command outputs by 100 ms. The MPC compensates for this by taking the current state measurements and computing the x,y location, orientation and velocity 100ms in the future. These values are then sent to the MPC controller to determine the next set of control output values.