Udacity Self-Driving Car Nanodegree

Term 2, Project 5: Model Predictive Control

Mark Helms, May 21, 2018

Goal/Summary

The goal in this project is to autonomously control a vehicle in a simulator using a model predictive controller (MPC) and knowledge about the waypoints, steering angle and speed of the vehicle. The vehicle must make a single lap around the track without leaving the road.

Approach

Step 1: Model Selection

The model chosen was the simple Kinematic Model described by the following state equations:

$x_{t+1} = x_t + v*\cos(\psi)*dt$	(x position changes as velocity times the cosine of the orientation angle)
$y_{t+1} = y_t + v*\sin(\psi)*dt$	(y position changes as velocity times the sine of the orientation angle)
$\psi_{t+1} = \psi_t + v/L_f*\delta*dt$	(orientation angle changes as velocity * the steering angle over a factor)
$v_{t+1} = v_t + a_t*dt$	(velocity changes at the rate of acceleration)

Additionally, the following error functions were used to keep track of performance:

$cte_{t+1} = y_t - f(x_t) + v_t*\sin(e\psi_t)*dt$	(cross-track error is the error in the y-position and changes with the error in the orientation angle)
$e\psi_{t+1} = e\psi_t + v_t/L_f*\delta_t*dt$	(error in heading changes with velocity times the steering angle divided by a factor)

The factor being mentioned, L_f, is the distance between the center-of-gravity of the vehicle and the front axle, where steering occurs. In effect, this factor dictates a turning radius at a given speed and can be discovered experimentally. This factor was given in this project at 2.67 m.

The actuators in the equations above are acceleration, a_t, and steering angle, δ_t. In the simulator, both of these values can vary between [-1,1]. For acceleration, this is the true range. For the steering angle, [-1,1] represents a range of [-25 degrees, 25 degrees].

The equation within cte referred to as $f(x_t)$ is the waypoint equation. This equation is represented by a 3rd-order polynomial which describes the course of the road that is to be followed. These points are provided by the simulator in map coordinates and are converted to vehicle coordinates in order to be plotted and to be more easily used in calculations. The conversion from map coordinates to vehicle coordinates involves calculating the radius and angle from the vehicle to the points, subtracting the map-coordinate-orientation angle of the vehicle and then multiplying the radius by the cosine and sine of the resulting angle.

$$\rho_{vc} = \rho_{mc} = \sqrt{((x_{veh\_mc} - x_{wp\_mc})^2 + (y_{veh\_mc} - y_{wp\_mc})^2)}$$
$$\theta_{mc} = \text{atan2}(y_{veh\_mc} - y_{wp\_mc}, x_{veh\_mc} - x_{wp\_mc})$$
$$\theta_{vc} = \theta_{mc} - \theta_{veh}$$
$$x_{wp\_vc} = \rho_{vc}*\cos(\theta_{vc})$$
$$y_{wp\_vc} = \rho_{vc}*\sin(\theta_{vc})$$

Once the waypoints have been converted into vehicle coordinates, they are sent to a polyfit command, which returns coefficients for a 3rd-order polynomial used to define f(x).

Step 2: Discovering a Best Course of Action

In order to discover the best actuation values for the vehicle to perform, they need to be discovered by finding the values with the best performance. They can be found by applying a cost to errors or undesirable actions and using the provided IpOpt and CppAD libraries to minimize a representation of the system.

The bounds on the system are as follows:

1) the initial state must be constant
2) acceleration is limited to [-1,1] and steering is limited to [-25 deg., +25 deg]

The model has a certain number of timesteps, N, at a certain duration, dt, that it considers when minimizing costs. The costs used for this implementation are:

1) the square of the cross-track error
2) the square of the heading error
3) the square of the velocity error against a reference velocity (defined as 30 mph)
4) a total steering error, squared – used to minimize the amount of steering performed
5) a total acceleration error, squared – used to minimize the amount of acceleration used

Costs 1, 2 and 3 are fundamental to the system. These costs must be minimized in order for the vehicle to follow the waypoints without stopping. Costs 4 and 5 are important in that they allow the system to operate smoothly. Additional factors were used to portray to the system the importance of one cost vs the others. In this implementation, cost 1 was deemphasized by multiplying by 0.5 in order to avoid resonance in steering. Cost 3 was also multiplied by 0.5 to avoid resonance in throttle/braking. Cost 4 was multiplied by a large factor 800, to penalize excessive steering.

The result is a very smoothly driving car, however, because of this approach, the total time that may be considered must be fairly short relative to the car's speed. This is because the minimal steering requirement will attempt to take very tight turns if it can see far enough ahead around a very sharp curve. This resulted in a choice for N * dt of about 0.6 seconds at 30 mph – represented by 12 points separated by 0.05 seconds.

While this approach is successful at the moderately low speed of 30 mph, a more sophisticated system might be a better choice for a vehicle that may travel at variable speeds.

Step 3: Dealing with Latency

Introducing latency into the system without adjusting the model accordingly may mean the actuations that are attempted will be too late to have the desired effect and may result in negative side-effects, such as resonance. Because latency is in effect dead time between our current time and when the changes become possible, we can shift our model over in time accordingly to match the lag. This is accomplished by using the state update equations above to advance the initial state given to the IpOpt/CppAD cost minimizer.