# Model Predictive Control

Srikant Rao

## 1 Kinematic Model

A modified version of the simple kinematic model was used to update the state vector.The modification accounts for the inherent delay in the steering and acceleration actuation. The time step $dt$ was chosen to be 100 ms which is equivalent to the actuation delay. Hence, the equations of motion can be represented as shown below -

$$x_{t+1} = x_t + v * cos(\psi_t) * dt \tag{1.1}$$
$$y_{t+1} = y_t + v * sin(\psi_t) * dt \tag{1.2}$$
$$\psi_{t+1} = \psi_t + v/L_f * \delta_{\mathbf{t\text{-}1}} * dt \tag{1.3}$$
$$v_{t+1} = v_t + a_{\mathbf{t\text{-}1}} * dt \tag{1.4}$$

Now that every actuator value is delayed by 1 timestep, **the first actuator value comes from the final value of the previous step**

This was achieved by creating two new variables in the MPC class as shown below

Listing 1: Actuator Delay Variables in class MPC

```
public:
MPC();

double latent_steering;
double latent_acc ;

virtual ~MPC();
```

Listing 2: Accounting for Actuator Delay in MPC.cpp

```
// Keep track of the final previous actuation to use for the next step
latent_acc = solution.x[a_start];
latent_steering = solution.x[delta_start];
```

## 2 Hyperparameter Optimization

**Total Time** - The total time to track the car is set to 1 s. Values between 1 and 2 s were tried but values higher than 1s did not lead to any significant improvement although there was noticeable increase in computational cost while running the simulator.

**Time Step dt** - The time step was chosen to be 100ms so that the actuator delay could be accounted for easily with one variable to store the previous steering and acceleration actuation values. This is the minimum time step needed to account for it accurately in the model. Smaller time steps could be implemented with a slight modification but were not tried.

**Number of Steps N** - The number of steps N is dependent on the total time and time step. the total time and time step were varied first and once those values were determined the number of steps was determined ($N = T/dt$)

**Weights for Cost Functoin** - The weight chosen for each error determines how important that error turns out to be. The following recipe was followed for choosing weights depending on how "adventurous" the car ride was going to be.
If the car is going to be driven in *Safe Mode*, then the weight of the error from the constant velocity expected should be between (0.5,1) and the constant velocity should be set between 50-70 mph.
If the car is going to be driven in *Fast Mode*, then the weight of the error from the constant velocity expected should be between (2,2) and the constant velocity should be set between 70-90 mph.

## 3 Polynomial Fitting and MPC Preprocessing

A third order polynomial is used to to fit the road trajectory points.
The $e\psi$ is based on calculation of $tan^{-1}(f^{'}(x))$ at x =0 in the vehicle co-ordinate system.
The x and y points were also transformed using 2D rotation matrix equations.
Option 2. in Tips and Tricks was used for the correct $\psi$ update. It was multiplied by -1 before being sent back to the server.

## 4 Model Predictive Control with Latency

The latency implementation has been absorbed in the model and the code modified to account for the previous actuator values that affect the performance of the **First time step**
The video shows the car driving in *Safe Mode*

`https://www.youtube.com/watch?v=uFVRoRYfZ9w`

The below videos hows the car driving in *Adventurous Mode*.
`https://youtu.be/7c2HUHzwieE`