Model Predictive Controller Project

Model

The model used is a **kinematic bicycle model**. The model state includes:

- position of the car w.r.t. x-axis, `px`
- position of the car w.r.t. y-axis, `py`
- orientation of the car, `psi`
- velocity of the car, `v`
- cross-track error, `cte`
- orientation error, `epsi`

The control inputs or actuations are:

- steering angle, `delta`
- acceleration or throttle, `a`

Below is the moving model of the vehicle:

```
// values at timestep [t+1] based on values at timestep [t] after dt seconds
// Lf is the distance between the front of the vehicle and the center of gravity

x[t+1] = x[t] + v[t] * cos(psi[t]) * dt;
y[t+1] = y[t] + v[t] * sin(psi[t]) * dt;
psi[t+1] = psi[t] + v[t]/Lf * delta[t] * dt;
v[t+1] = v[t] + a[t] * dt;
cte[t+1] = f(x[t]) - y[t] + v[t] * sin(epsi[t]) * dt;
epsi[t+1] = psi[t] - desired_psi + v[t]/Lf * delta[t] * dt;
```

Timestep Length and Elapsed Duration (`N` & `dt`)

According to the course material, the prediction horizon, `T` is the product of the timestep length, `N` and elapsed duration, `dt`. Timestep length refers to the number of timesteps in the horizon and elapsed duration is how much time elapses between each actuation.

The prediction horizon that I chose is one second, with `N = 10` and `dt = 0.1`.

I tried different combinations of `N` and `dt`, and among all the pairs (`N = 10, dt = 0.1`) performed the best. `dt` was to kept lower so that actuations could be more dynamic and real time. With N, if it is higher then car would oscillate wildly and drive off the track if it ever overshoots the reference line, and also unnecessarily computation time will increase because of this, but if it is lower then the car may drive straight off the track because it will be near-sighted so it won't be able to anticipate the future turns correctly.

Polynomial Fitting and MPC Preprocessing

The waypoints are first transformed from global to the vehicle's coordinate system. This is done by subtracting each point from the current position of the vehicle.

Next, the orientation is also transformed to 0 so that the heading is straight forward. Each point is rotated by psi degrees.

After that the vector of waypoints is passed to polyfit function where the points are fitted to a 3rd order polynomial. This polynomial function is then later used, using the polyeval function to calculate the cross-track error.

Model Predictive Control with Latency

A delay of 100 ms was needed to be taken care of in the simulator. In order to deal with the latency, I set the initial states to be the states which is going to be after 100 ms. This allows the vehicle to "look ahead" and correct for where it will be in the future instead of where it is currently positioned.