MPC Controller

Model predictive control (MPC) is an advanced method of process control that has been in use in the process industries in chemical plants and oil refineries since the 1980s. In recent years it has also been used in power system balancing models.

We choose 2D motion kinematic equations for our project and we are ignoring other forces such as tire forces, gravity, mass, longitudinal, lateral, air resistance, drag etc. this simplification reduce the accuracy of the model but it makes it more traceable.

Our state is [x, y, ψ, v, cte, eψ] x position, y position, orientation (steering), cross track error of position (difference b/w vehicle center and the actual waypoint position), & cross track error of orientation.

## Model

*dt=0.5*

*N = 6*

## Constraints

*T (seconds) = 3*

= distance from the front wheel to the center of the gravity of car (CoG).

It is calculated by moving a car in a circular path.

*dt* = time lapse b/w previous and current sensor reading.

MPC works great even with latency (delay in getting input).

is the steering angle which is constrained with -25 to 25 degree.

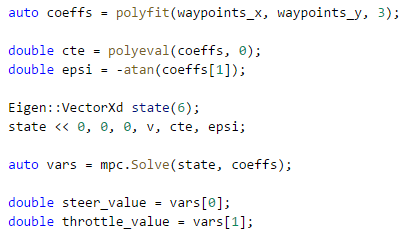
a is the acceleration, 1 means full and -1 is full reverse (or can be used to apply brake).

## Cost

I calculate the cost of the model as following equation:

## Fitting curve

We find the coefficient of the waypoints using 3rd degree polynomial. Then we used those coefficients to evaluate cross track error and orientation error using arc tangent.



The cross-track error is calculated by evaluating a polynomial and subtracting the original value.

To calculate orientation error. eψ. We use eψ = ψ−ψdes, where **ψdes** is can be calculated as arctan(f​′​​(x)).

## **Prediction Horizon**

The prediction horizon is the duration over which future predictions are made. We’ll refer to this as *T*.

*T* is the product of two other variables, *N* and *dt*.

*N* is the number of timesteps in the horizon. *dt* is how much time elapses between actuations. For example, if *N* were 20 and *dt* were 0.5, then *T* would be 10 seconds.

## **Number of Timesteps**

The goal of Model Predictive Control is to optimize the control inputs: [δ,a]. An optimizer will tune these inputs until a low-cost vector of control inputs is found. The length of this vector is determined by *N*:

[δ​1​​,a​1​​,δ​2​​,a2,...,δN−1,a​N−1​​]

Thus, *N* determines the number of variables the optimized by MPC. This is also the major driver of computational cost.

Choosing large dt (timestep) cause discretization error, which result in late actuations. It also makes it harder to predict continuous trajectory.

## Choosing N

Large value of N destabilizes the vehicle as we predict too far in future and environment is dynamic. Large values increase the wiggle behavior of the car. I tried value of 30 and 12. Following are the screenshot of N=12

|  |  |  |
| --- | --- | --- |
|  |  |  |

Then I tried to lower the value of N to 4. After 2 sec car goes off-road.



Then I tried value of 8.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Still car have little wobbliness. After 3 laps, wobbliness increased. Then I changed value to 6.

|  |  |  |
| --- | --- | --- |
|  |  |  |

After 5 laps, the cross-track error remains low, So I the value of N to 6.

## Simulating Latency:

To simulate latency of 100ms I paused the execution of main loop to 100ms. I tried the latency of 1s (car get out of control as it is getting fixed steep values for the actuators) and 300ms (which start wobbliness after few seconds) so I keep the value of 100ms for latency.

## Reference

<https://en.wikipedia.org/wiki/Model_predictive_control>

<https://classroom.udacity.com/nanodegrees/nd013>