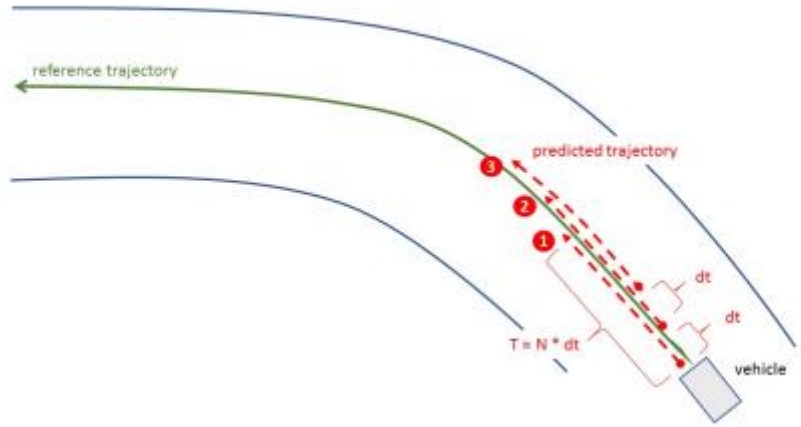
Model Predictive Control P5 – Term 2

Project Writeup

1. Outline

The Model Predictive Control (MPC) predicts an optimal vehicle trajectory based on the vehicle state and a reference trajectory. This prediction is calculated by simulating a kinematic vehicle model and optimizing the result by a cost function over a certain number (`N`) of time-steps (`dt`). The output of the predicted trajectory are the future vehicle states and actuator settings used to control the vehicle over that certain number (`N`) of time-steps (`dt`).
In terms of vehicle control only the actuator results of the 1st predicted time-step are used and the following time-steps will be ignored. Therefore, the Model Predictive Control (MPC) will re-calculate a new vehicle trajectory prediction using an updated vehicle state and reference trajectory every time-step. Rationale for this is that the predicted trajectory is only approximate and needs to be re-calculated in order to be accurate.
The figure below presents the MPC flow.



2. Model

The Model Predictive Control (MPC) uses a kinematic vehicle model which calculates the next state as presented below:

```
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 * dt
v(t+1)  = v(t)    + a(t) * dt
cte(t+1)        = polynom(x(t)) - y(t) + v(t) * sin (psi (t)) * dt
epsi(t+1)       = psi(t) - arctan(polynom(x(t))^) +  v(t) / Lf * delta * dt

x, y            = position of vehicle
psi             = position angle of vehicle
delta           = steering angle (actuator to control)
v               = velocity of vehicle
a               = throttle of vehicle (actuator to control, accel. > 0, break < 0)
cte             = cross track error (goal to minimize)
epsi            = error angle (goal to minimize)
Lf              = constant value dependent on the vehicle
Polymon(x)      = reference trajectory polynomial
```

The MPC predicts the next N states of the vehicle state `(x, y, psi, v, cte, epsi)` and actuator settings `(delta, a)` using the model above, the initial vehicle state, actuator & state constraints and a cost function. The cost function minimizes the errors `cte` and `epsi` and other specified items.

I have used the following constraints and cost function:

a) Actuator constraints (others set to 0): `delta = [-25°, +25°], a = [-1, +1]`

b) Cost Function: the cost function covers 3 x different areas of optimization with different weights:

1. Optimization of Errors `cte` & `epsi` and constant velocity (in order to guarantee that the vehicle is moving). Weights for all are 1.
2. Optimization to minimize of actuators usage steering (`delta`) and throttle (`a`). Weights is 200 for steering and 25 for throttle.
3. Optimization to minimize the value gap between sequential actuations. Weights is 200 for steering and throttle sequential actuations.

3. Time-step Length and Elapsed Duration (N & dt)

The elapsed duration (time-frame) for which the MPC calculates the predicted trajectory can be calculated by the equation `T = N * dt`. The time-frame should be around 1-2s in order to let the system sufficiently reactive (for higher velocities the time-frame should be even smaller). In terms of the time-step length `dt` and `N` the following needs to be considered:

a) the equation `N * dt` = time-frame of 1-2s needs to be considered
b) `N` is proportional to the computing effort therefore it should not be too big
c) the time-step `dt` should not be too big otherwise the actuators may have problems to follow

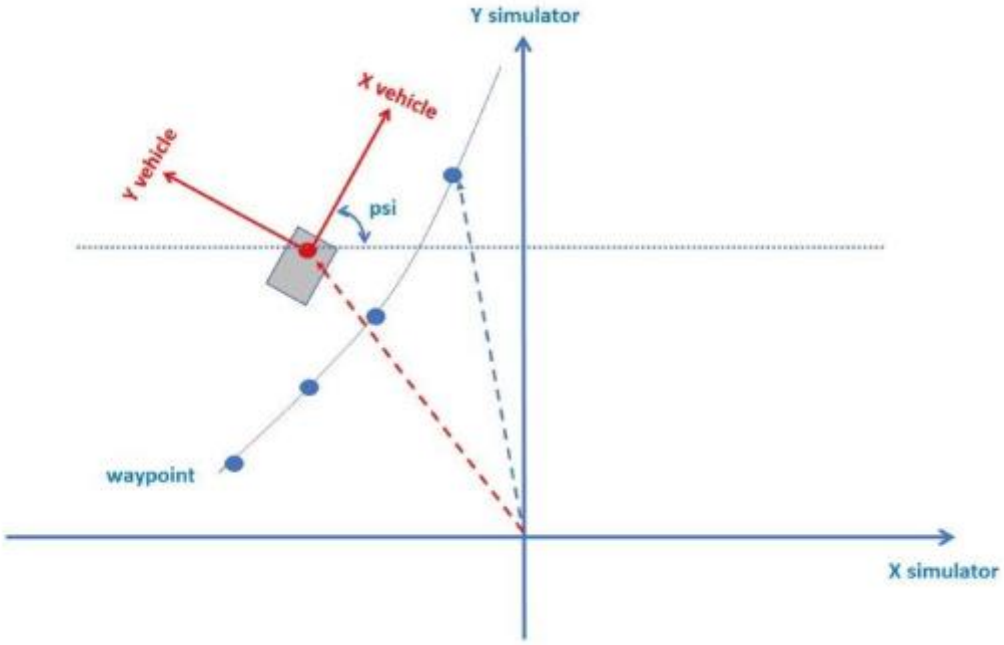
I have chosen the following parameters for that project: `N = 15`, `dt = 0.05s`, time-frame `T = 0.75s`. Additionally, I were trade-off multiple parameters. It seemed that parameters using `N < 10` or `N > 40` did not work using my cost function and adapting the time-step to maintain the time-frame of 0.6 – 1s.

4. Polynomial Fitting and MPC Pre-processing with actuator delay

The first step of the polynomial fitting & pre-processing is to transform from the simulator coordinate system into the vehicle coordinate system by

a) subtracting the centre coordinates of the vehicle coordinate system from the waypoint values of the simulator coordinate system
b) then turning the simulator coordinate system by the angle -psi

see also the figure below:



After this transformation of the waypoints to the vehicle coordinate system the polynomial is calculated.

After that the next step is to create the vehicle state vector and provide that vector with the polynomial to the MCP solver function:

a) For a **Zero Latency Actuator** the state vector values are: `x = 0`, `y = 0`, `psi = 0` `cte = polynomial for x=0`, `epsi = -arctan(derivate of polynomial for x=0)` using the vehicle coordinate system.

b) For the **100ms Actuator Latency** a future state vector is calculated using the kinematic model with a 100ms time-step and then provided with the polynomial to the MPC solver function.

   Note: The **steering** and **throttle** value need to be saved for the next "latency" calculation, initial these values are 0.

   I have implemented the kinematic model with the time-step 100ms delay inside the `main.cpp` file because the model equations could be optimized thanks to the vehicle coordinate system.