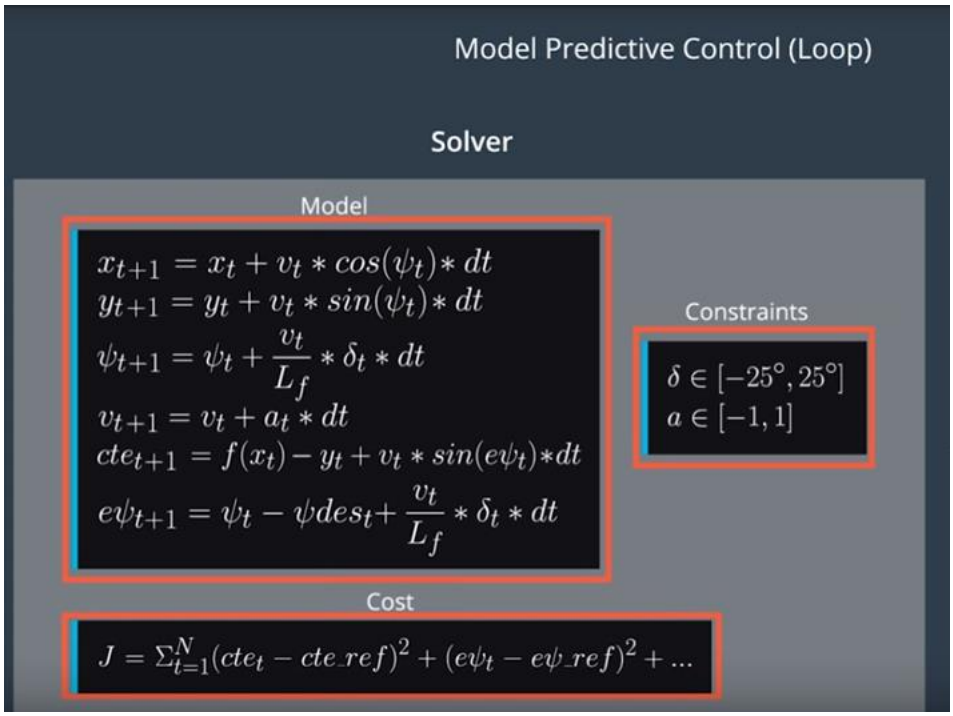
Rubic points:

1. **Student describes their model in detail. This includes the state, actuators and update equations.**



Above is the MPC solver equations from the lecture.

The MPC model receives data per N timestamps. Duration of time is dt. For each timestamp, all the vehicle states will need to be updated according to the equations

The vehicle state contains:

- X: x position of the host
- Y: y position of the host
- Psi: yaw
- V: velocity:
- Cte: cross traffic error
- epsi : yaw error
- delta: steering angle
- a : acceleration

The cost function is calculating the cte sum for speed and trajectory. The cost function will need the state, actuation for getting a value. The weight for each state matters for actual model behavior. I this project, I manually tuned the params to make it stable in simulator. There could be better automatic ways for tuning that, which is my todo item after finishing the project.

2. **Student discusses the reasoning behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values. Additionally the student details the previous values tried.**

The value are set to N= 10 and dt= 0.1. I don't actually get the intuition of tuning this param. I tried different values including:

N= 5 and dt= 0.1.

N= 10 and dt= 0.1.

N= 20 and dt= 0.1.

N= 30 and dt= 0.1.

N= 10 and dt= 0.2.

N= 10 and dt= 0.3.

Seems N= 10 and dt= 0.1 is the best according to observations.

I also visit other people's repo for idea about this, but seems the default is the best option for most people. I think in real sensor the timestamp might be a fixed value provided by manufacture, so I need to understand more about what is the impact of tuning this param.

3. **A polynomial is fitted to waypoints. If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.**

In the simulator, the yellow line is the received trajectory poly fit while the green one is the MPC predicted trajectory. The positions x, y are transformed to AUTOSAR coordinate for fitting the curve correctly. The handing code is located in main.cpp

4. **The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.**

This is handled in main.cpp. The dt is accounting for state change, therefore the all the states are updated for dt per the updating equations.

The code is as follows:

```
// Account for latency 100ms for all the state
double dt = uilatency / 1000;
double x_lat = v * dt;
double y_lat = 0;
double psi_lat = v * steer_value / lf * dt;
double v_lat = v + throttle_value * dt;
double cte_lat = cte + v * sin(epsi) * dt;
double epsi_lat = epsi + v * steer_value / lf * dt;
```