The model used in this project is model predictive control model, it has the following code:

```
px =  (x0 + v * cos(psi0) * dt);
py =  (y0 + v * sin(psi0) * dt);
psi = (psi0 - v/Lf * steer_value * dt);
v = (v + throttle_value* dt);
cte = (f0-y0 + v * sin(psi0-atan(coeffs[1])) * dt);
epsi = (psi0 - psides0 - v/Lf * steer_value * dt);
```

The state is {x,y,psi,v,cte, epsi}, cte is the cross check error, epsi is the orientation error. The actuators are the acceleration a, and the steering angle delta.

In the project, I choose the N = 10, dt=0.1. Because first the total time equal to 1 second, which is a reasonable amount of time for drivers to apply a new steering angle and acceleration. I pick dt=0.1 second, because it is approximately equal to the latency time of the system.

The way points from the simulator are transformed from vehicle coordinate system to the global coordinate system by using the following code:

```
for (int i=0; i< ptsx.size();i++){
              double shift_x = ptsx[i]-px;
              double shift_y = ptsy[i]-py;
              ptsx[i] = (shift_x*cos(0-psi) - shift_y*sin(0-psi));
              ptsy[i] = (shift_x*sin(0-psi) + shift_y*cos(0-psi));
            }
```

To consider the latency of the system, I used the following code in the main.cpp.

```
          //Include the latency by assume the throttle approximately equal accel.
          double x0 = 0;
          double y0 = 0;
          double psi0 = 0;
          double dt = 0.1;
          double Lf = 2.67;
          double f0 = coeffs[0] + coeffs[1] * x0+ coeffs[2] *x0*x0+ coeffs[3] * x0*x0*x0;
          double psides0 = atan(3*coeffs[3] * x0*x0 + 2*coeffs[2] *x0+ coeffs[1]);
          px =  (x0 + v * cos(psi0) * dt);
          py =  (y0 + v * sin(psi0) * dt);
          psi = (psi0 - v/Lf * steer_value * dt);
          v = (v + throttle_value* dt);
          cte = (f0-y0 + v * sin(psi0-atan(coeffs[1])) * dt);
          epsi = (psi0 - psides0 - v/Lf * steer_value * dt);
```

Basically, I use the current data got from the simulator and apply the update equal with dt=100 milliseconds. Then use the predicted point as the starting point in the optimization. Because ideally, the car will be in that position when it start to execute the actuators.