1. The Model
   The model is defined by the state, error, and actuators. The states are position (x,y), orientation (psi) and velocity (v). The errors are relative to cross track error (cte) and orientation (epsi). The actuators, or inputs to the system, are steering wheel (delta) and acceleration (a/throttle).

   The car is updated as time passes using the following equation:

   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

   Lf - this is the length from front of vehicle to its Center-of-Gravity

   The cte & epsi can be predicted using:

   cte_[t+1] = cte[t] - v[t] * sin(epsi[t]) * dt

   epsi_[t+1] = epsi[t] + v[t] / Lf * (-delta[t]) * dt

2. Timestep Length & Elapsed Duration (N & dt)
   The prediction horizon is the duration that we're making predictions for. This is broken down further into the number of steps in this horizon (N) and the elapsed time(dt) between the steps. We tune N and dt to give us the best results. Too large an N will slow down computation because of the large number of variables needed to be optimized. Larger dt values result in less frequent actuations, making it less accurate to predict.

3. Polynomial Fitting and MPC Preprocessing
   We used a 3^{rd} order polynomial to fit our curve as this should suffice for most trajectories we will encounter.

4. Model Predictive Control with Latency
   We deal with latency by setting the initial state 100 mSeconds into the future. We do this by calculating where that position and orientation is going to be in the future, then input that into our solver.

```
//Find state at 100ms in the future
   double lat = 0.1;
   psi = (-v* steer_angle * lat)/2.67;
   px = v*cos(psi) * lat;
   py = v*sin(psi) * lat;
   v = v + throttle_value * lat;
```