Model

The model is defined by Udacity. The system uses a kinematic model to define the motion of the car, actuator values and bases the output on the last set of values. Below are the equations used, this is a screenshot from the Udacity course material.

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t - \frac{v_t}{L_f} * \delta_t * dt$$

$$v_{t+1} = v_t + a_t * dt$$

$$cte_{t+1} = f(x_t) - y_t + (v_t * sin(e\psi_t) * dt)$$

$$e\psi_{t+1} = \psi_t - \psi des_t + (\frac{v_t}{L_f} * \delta_t * dt)$$

Timestep and dt

This is something I tested and got some info form the forum. I ended up settling on N of 10 a dt of 0.1. It seems that most people had success with similar or these values on the forum, so I started here and increased/decreased a bit but ended up finding these were the best. It’s a fine line between accuracy and computation cost and the optimum seemed to be 10 and 0.1.

Dealing with latency

Udacity specifies in the code to use 100 milliseconds for the latency value, so this is what I did. However since the calculation is “rolling” or the current state is based on the last state I had to alter the state equations a bit to get the quality I was looking for. I also had to add another cost item in the main section. This helped when the car is making a turn since a latency of 100 milliseconds is also the dt value so the system is always one step behind.