The Model

I used model from the class.

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

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

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

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

Not sure what to comment on. Equations are implemented in file main.cpp from line 149.

Initially I tried to add velocity for X and ψ calculations, but that always lead to behaviour where car simply drove in circle. Circles are general issue with current implementation. If car increase speed from current 30mph it drives well, but when leaving the bridge controls go completely crazy. Bot sure how to debug, so input would be appreciated.

Non zero values that are passed to MPC

- **Velocity** = velocity + throttle value * latency
- **Cross tr err** = cte + velocity * sin(epsi) * latency
- **Epsi** = epsi + velocity * steering value / Lf * latency

Latency is set to target 0.1 seconds. Velocity is factoring in throttle, since that what it is about. Errors are just incorporating velocity into standard equations.

Timestep Length and Elapsed Duration (N & dt)

- **N = 12** empirically found sweet spot to match planned trajectory length. If I further increase this number predicted path will exceed planned path and that lead to 180 degrees turn.
- **dt = 0.1;** 0.2 does not work, as car moves too far and predictions got useless, bellow is pointless. Considering that we also have a latency of 0.1 overall fair number, however only for lower speeds.

Review Update

- Why smaller dt is better? (finer resolution)

Smaller time period will allow to put more dots to our planned path, so final result should be smoother. But there are more things co compute, so it is better to stop on some sweet spot.

- Why larger N isn't always better? (computational time)

Yes, time. Also this weird lag when you exceed the length of the planned path. In addition, predictions that are too far ahead are simply useless. We are not building the model that tracks historical values.

- How does time horizon (N*dt) affect the predicted path? This relates to the car speed too.

More measurements (N) with more time gap (dt) means further horizon, as we have more points with large space between.

Polynomial Fitting and MPC Preprocessing

I did not do any distinct preprocessing for MPC. Only transformed into the proper coordinate system with

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

It is code from exercises, but I made some optimizations to cut down number of operations in the loop.

Model Predictive Control with Latency

Latency was implemented into the functions that were discussed above. In short, latency is used as a discount factor to all values that passed to MPC.

The vehicle must successfully drive a lap around the track.

2 laps.

https://youtu.be/kEoPQ3B9dLo

Review comments

What is this cost function supposed to do? It looks like you are deducting the same variable from itself which will always result in 0.

Is this what you're trying to do?

fg[0] += 1000 * CppAD::pow(vars[a_start + i + 1] - vars[a_start + i], 2);

That is exactly my code, so not sure what to comment.

It is calculating “[t + 1] – [t]”. I add squared error and enhance it by 1000, since numbers are small. I had a bug bellow; I guess that what was actually referred to.

You should invert the sign of steering_angle. In our model, we assume positive angle is counter-clockwise which is turning left but simulator takes negative value for left.

Why is that a problem that I invert it later?

You should also update x, y and psi

If I do, model stops working. You can uncomment code in the end of comment block marked “review”. You can see that everything breaks completely. I do not mind fixing, but it was the main issue that I faced and I have no idea what else is wrong that does not allow it to function properly.

cte = desired_y - actual_y

= polyeval(coeffs,px)-py

= polyeval(coeffs,0) because px=py=0

epsi = actual psi-desired psi

= psi - atan(coeffs[1]+coeffs[2]*2*px+...)

= -atan(coeffs[1]) because px=psi=0

That part confuses me. In my code I have

```
// estimate cte

double cte = polyeval(coeffs, 0.0);

// estimate epsi

double epsi = -atan(coeffs[1]);
```

I cannot figure out what is wrong here