MPC Model Predictive Control Project David Simon

Description (see all runs at end of paper)

Due to success with the randomization of the particles in the particle filter and the PID project using the Hangout videos, the full coding for the complicated MPC project was originally derived from the MPC Hangout video. The car was barely moving, getting up to 3mph at most, despite testing all variations of the cost multipliers or weights, focusing on cte, epsi, and delta. Time steps and time period, steer positive and negative (recommended) were also tested. Sometimes the green line for the predicted car path was present, sometimes not. A major problem, a waypoints yellow path bending leftward out of the road that the green predicted line tried to follow, was determined to be a sign error of mine in the transformation code for the waypoints to the car orientation.

With the yellow line down the middle of the road the car movement was even less, typically little if any forward movement, despite the car revving. Again numerous weight variations were tested to no effect. Another sign change was made on the transform to disastrous effect and was changed back. Lf contained in the video code for the steer angle and the limits was removed as in the quiz solution to no impact. Full code check line by line revealed no difference between mine and the Hangout video.

Decided to try vars[6] and vars[7] which seemed to me to involve delta for steering and a for the throttle instead of the vars[0] and vars[1] from the video. However, while the car moved forward, the best runs still crashed in the curve after getting up the hill. This was achieved with a very high cte weight of 80000 and delta change weight of 3000. Often the biggest problem was the car tires rapidly turning back and forth trying to decide where to go, and sometimes the car would even go backward. As it was becoming clear this was not working, my original post on the forum about the car not moving was answered by a forum mentor. He indicated vars[0] and vars[1] were definitely correct and to return to that. However, I noted I had tried that for a couple of days.

But as vars[6] and vars[7] were only somewhat working and not well, I returned to vars[0] and vars[1]. When I turned off the weights for acceleration, the car now moved. Testing gave some surprising results. The car moved if the lower limit on acceleration was 0 instead of -1.0 which had been changed to prevent the car from going backward in the vars[6], vars[7] testing. Further testing revealed that the acceleration weights acted as brakes on the car. With those weights off the car would move. Again, the weights were increased mainly on the cte of 80000 and the delta change of 80000, but usually the car would crash to the right or ride on the right edge of the road. Epsi weight was sensitive and even 1000 caused a leftward crash. Adjusting the throttle with a multiplier from 0.75 to 0.10 did not help other than slowing down the car.

Interestingly, changing the acceleration lower limit to 0.1 allowed the car to move, but slowly, with brakes on, even with acceleration weights. Best run at a slow 2mph using the 0.10 throttle multiplier and the 80000 cte and 3000 delta change weights still crashed on the right side of the bridge. Increasing the throttle multiplier to 0.5 pushed the speed to 9mph, but then the crash was on the sharp curve before the bridge. As this was not looking good, I turned to the quiz model which most had used successfully. However, I had been testing one I was building already and had some pesky compiling errors.

So I followed a recommendation (believe Spartacus) from a forum member who had difficulty with the video model and had merged successfully the mpc of the quiz model with the main of the video model while using the video return in the mpc. His overall point was that the iteration sequences of the video model for the costs and the constraints, while appearing to be equivalent, were not actually the same as the quiz model, and caused problems. After some errors, I successfully merged the two files.

Initial runs were disturbing as appeared jittery, quickly crashing either to the left side or the right side. However, one major and odd improvement was the consistent appearance of the yellow waypoint and green predicted lines. With the video model, there would be variation, with the green line disappearing or flashing with the use of the epsi and delta costs. Sometimes even the green line would only be dots. What turned out to be surprising was the much higher sensitivity of epsi especially in the merged video/quiz model compared to the video model. As the model kept crashing with weight adjustments, even after using the combination from the Hangout video and one from a forum mentor, I reduced the speed with throttle multiplier set down to 0.1 and surprisingly on the 7th run, got around the track at 5mph without crashing.

Cte	epsi	v	delta	a	deltaΔ	aΔ	
200	200	1	1	1	1	1	0.5throttle, quick crash, but a works and good lines
2000	2000	1	1	1	1	1	same result, quiz model has N25, .05dt
4000	1	1	80000	1	1000	1	per A Cui forum mentor similar crash, good lines
80000	1000	1	80000	1	80000	1	immediate crash, curving as with previous
2000	2000	1	5	5	200	10	video numbers, still crashing
80000	500	100	1000	20	5000	20	.25throttle still crashing
*8000	50	100	1000	20	500	20	.10throttle SUCCESS, 5mph, but no crash

Taken from runs below, had repeated success, increasing the throttle and achieving smooth 40mph runs about the track. Turned out cte and epsi weights could be low at 200 and 50, but oscillation of the car became a problem with increasing speed. Large delta and delta change weight values of 70000 solved that issue.

Cte	epsi	v	delta	a	deltaΔ	aΔ	
800	50	100	100	20	500	20	SUCCESS, 10mph, no crash, N10, .1dt
200	50	100	100	20	100	20	SUCCESS, 10mph, no crash, oscillating
200	50	100	100	20	1000	20	0.2throttle, 15mph, no crash, more oscillating
200	50	100	1000	20	10000	20	0.3throttle, 30mph, no crash, much oscillating
200	50	100	10000	20	10000	20	no change

200	50	100	30000	20	30000	20	SUCCESS, 35mph, no crash, very smooth
200	50	100	30000	20	30000	20	0.4Throttle, 40mph, no crash, some oscillating
200	50	100	50000	20	50000	20	better, oscillation on last S curve
200	50	100	70000	20	70000	20	0.5Throttle, 40mph, smooth
**200	50	100	70000	20	70000	20	0.75Throttle, still 40mph due to braking, smooth

Increasing the reference speed to 60 mph turned out to be difficult. It was the maximum speed achieved, but only by increasing the delta and delta change to a very large 200000. As the car just barely made it through, I did not try to optimize the delta weight values between 90000 and 200000.

Cte	epsi	v	delta	a	deltaΔ	aΔ	
200	50	100	70000	20	70000	20	refv40 to 60, 0.75throttle, oscillated, crash bridge
200	50	100	90000	20	90000	20	crashed after bridge, oscillation, 10N .1dt
200	50	100	90000	20	90000	20	20N, .05dt, much worse, crashed early
200	50	100	90000	20	90000	20	5N, 0.2dt, much worse, crashed early
200	50	100	125000	20	125000	20	10N, 0.1dt, crashed at bridge
1000	100	100	90000	20	90000	20	crashed early
100	100	100	90000	20	90000	20	still crashed at last S curve
100	1000	100	90000	20	90000	20	crashed after bridge
100	100	100	90000	100	90000	100	still crashed after bridge
***200**	**50**	**100**	**200000**	**20**	**200000**	**20**	**60mph, barely making it, large oscillations**

Further runs below 60 mph

**200	50	100	200000	20	200000	20	refv50, full throttle, some oscillation
**200	100	100	200000	20	200000	20	refv50, full throttle, similar
****200**	**75**	**100**	**200000**	**20**	**200000**	**20**	**refv50, full throttle SMOOTH**
*200	75	100	200000	20	200000	20	refv55, full throttle, barely making it

Return to refv50 as best maximum speed, full throttle, and incorporate latency time of 0.001 sec

200	75	100	200000	20	200000	20

runs without sleep, but barely hits some road edge markings in S curve

same with chrono sleep command, max speed attained

SUBMITTED RUN

Try refv45, full throttle, latency time of 0.001 sec

Cte	epsi	v	delta	a	deltaΔ	aΔ
200	**75**	**100**	**200000**	**20**	**200000**	**20**

Smoothest run, fully within road between road edge markings

Note that latency implementation seemed to worsen the run with some oscillation. This is why without latency, reference speed at 50mph was smooth, but with latency, reference speed of 45 mph was better. Reference speed at 50mph with latency is still quite good, but there is more oscillation with some hitting of the road edge markings.

THE PROGRAM

MAIN

The simulator sends out telemetry for waypoint x and y location (ptsx, ptsy); car location (px, py); psi angle, velocity v, steer angle (steer_value) and throttle (throttle_value).

If latency time is present, the car px, py, v, psi values are adjusted into the future.

```
v = v + throttle_value*Lt;

psi = psi + v/Lf*steer_value*Lt;

px = px + v*cos(psi)*Lt + 0.5*throttle_value*cos(psi)*Lt*Lt;

py = py + v*sin(psi)*Lt + 0.5*throttle_value*sin(psi)*Lt*Lt;
```

from $v = v + at$, psi equation from lecture, x and y from x or y = x or y + vt + $(1/2)at^2$

adjusted using trigonometry, cos for x, sin for y

As car reference is 0,0 for x and y, with x going in the path of the car and y to the right in the simulator, the waypoints are transformed from the global map of (x,y) of (0,0) at the left corner. As a result, the Hangout video transformed the waypoints, rotating 90 degrees.

```
//change waypts to car reference by 90 deg

    size_t i;

    for(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)); //was +
          ptsy[i] = (shift_x * sin(0 - psi)) + (shift_y * cos(0 - psi));
        }
```

The transformed waypoints are sent to polyfit function to determine the 3rd order polynomial coefficients that will be used to predict the path of the car.

The transformation simplifies the calculation of cross track error cte and psi angle error epsi.

```
double cte = polyeval(coeffs, 0);
//double epsi = psi - atan(coeffs[1]+2*px*coeffs[2]+3*coeffs[3]*pow(px,2));
  double epsi = -atan(coeffs[1]); //simplified due to psi = 0
```

Another advantage of this transformation to car coordinates is that the state vector initialization can be

```
State << 0, 0, 0, v, cte, epsi;
```

The state and the coefficients are put into the Solver.

The optimum trajectory is created

```
       double poly_inc = 2.5;
        int num_points = 25;

        for(int i = 0; i < num_points; i++){  //i was 1
          next_x_vals.push_back(poly_inc*i);
          next_y_vals.push_back(polyeval(coeffs, poly_inc*i));
```

while the predicted trajectory uses the vars vector from the Solver

```
for(int i = 2; i < vars.size(); i++){//try i=1
          if(i%2 == 0){
            mpc_x_vals.push_back(vars[i]);
 }
          else{
            mpc_y_vals.push_back(vars[i]);
```

}

The Solver steer_value is sent back to the simulator. Sign is changed as normally angle is positive rotating counterclockwise while simulator angle is positive going clockwise.

steer_value = -vars[0]/(deg2rad(25)); //was positive sign and was (deg*Lf)

msgJson["steering_angle"] = steer_value;

The Solver throttle_value is sent back to the simulator. Multipliers were used to test the car runs at lower speeds.

throttle_value = vars[1]; //use multiplier to adjust

msgJson["throttle"] = throttle_value;

The simulator is sent the optimum trajectory to plot (yellow line)

msgJson["next_x"] = next_x_vals;

msgJson["next_y"] = next_y_vals;

And the simulator is sent the predicted trajectory to plot (green line)

msgJson["mpc_x"] = mpc_x_vals;

msgJson["mpc_y"] = mpc_y_vals;

Lastly if latency time is greater than zero, the sleep_for command is triggered, in this program for 100 milliseconds.

if(Lt > 0){

std::this_thread::sleep_for(chrono::milliseconds(100));

MPC

Number of time steps N and time period dt were tested in the final runs with the working program. The Hangout video had N = 10, dt = 0.1 sec while the quiz had N = 25, dt = 0.05 sec. As I attempted to get the 60 mph maximum speed to run without crashing, the car got to the bridge using N = 10, dt = 0.1 sec, while N = 5, dt = 0.2 sec and N = 20, dt = 0.05 sec both quickly crashed. Perhaps the dt of 0.2 sec is just too slow to accurately predict and respond to the optimum trajectory and maybe the N of 20 is too many time steps, again ending up being too slow in the calculations to respond accurately.

Reference v sets the maximum velocity and this can be seen as the brakes will flash at different times, keeping the speed at this maximum. I removed the reference cte and reference epsi which was in the

video file, but not in the quiz file. Perhaps that could be useful to enable an adjustable limit, but the Solver should be minimizing cte and epsi anyway.

Size_t is used to set up the large vars 1-dimensional vector.

```
size_t x_start = 0;

size_t y_start = x_start + N;

size_t psi_start = y_start + N;

size_t v_start = psi_start + N;

size_t cte_start = v_start + N;

size_t epsi_start = cte_start + N;

size_t delta_start = epsi_start + N;

size_t a_start = delta_start + N - 1;
```

The fg vector contains both the cost and constraints.

Costs are contained in the first position fg[0]

```
fg[0] = 0;
  // The part of the cost based on the reference state.
  for (int t = 0; t < N; t++) {
    fg[0] += 200*CppAD::pow(vars[cte_start + t], 2);
    fg[0] += 75*CppAD::pow(vars[epsi_start + t], 2);
    fg[0] += 100*CppAD::pow(vars[v_start + t] - ref_v, 2);
  }
  // Minimize the use of actuators.
  for (int t = 0; t < N - 1; t++) {
    fg[0] +=200000*CppAD::pow(vars[delta_start + t], 2);
    fg[0] += 20*CppAD::pow(vars[a_start + t], 2);
  }
  // Minimize the value gap between sequential actuations.
  for (int t = 0; t < N - 2; t++) {
    fg[0] +=200000*CppAD::pow(vars[delta_start + t + 1] - vars[delta_start + t], 2);
```

```
    fg[0] += 20*CppAD::pow(vars[a_start + t + 1] - vars[a_start + t], 2);
  }
```

As stated before, the delta weights were by far the largest to reduce the large oscillations that caused crashing. Usually this started in a curve but could continue into the straightaways. Cte weight was far lower at 200 and epsi at 75 as those were more sensitive than expected to the weights.

The remaining fg[] sets the state and actuator constraints for the simulation using the Solver.

```
// Setup Constraints
  //
  // NOTE: In this section you'll setup the model constraints.
  // Initial constraints
  //
  // We add 1 to each of the starting indices due to cost being located at
  // index 0 of `fg`.
  // This bumps up the position of all the other values.
  fg[1 + x_start] = vars[x_start];
  fg[1 + y_start] = vars[y_start];
  fg[1 + psi_start] = vars[psi_start];
  fg[1 + v_start] = vars[v_start];
  fg[1 + cte_start] = vars[cte_start];
  fg[1 + epsi_start] = vars[epsi_start];

  // The rest of the constraints
  for (int t = 1; t < N; t++) {
    // The state at time t+1 .
    AD<double> x1 = vars[x_start + t];
    AD<double> y1 = vars[y_start + t];
    AD<double> psi1 = vars[psi_start + t];
    AD<double> v1 = vars[v_start + t];
```

```
    AD<double> cte1 = vars[cte_start + t];
    AD<double> epsi1 = vars[epsi_start + t];
    // The state at time t.
    AD<double> x0 = vars[x_start + t - 1];
    AD<double> y0 = vars[y_start + t - 1];
    AD<double> psi0 = vars[psi_start + t - 1];
    AD<double> v0 = vars[v_start + t - 1];
    AD<double> cte0 = vars[cte_start + t - 1];
    AD<double> epsi0 = vars[epsi_start + t - 1];

    // Only consider the actuation at time t.
    AD<double> delta0 = vars[delta_start + t - 1];
    AD<double> a0 = vars[a_start + t - 1];

    AD<double> f0   = coeffs[0] + coeffs[1] * x0 + coeffs[2] * CppAD::pow(x0,2) + coeffs[3] *
CppAD::pow(x0,3);
    AD<double>psides0 = CppAD::atan(3 * coeffs[3] * x0 * x0 + 2 * coeffs[2] * x0 + coeffs[1]);

    //AD<double> f0 = coeffs[0] + coeffs[1] * x0; //first order for quiz
    //AD<double> psides0 = CppAD::atan(coeffs[1]); //first order for quiz

    fg[1 + x_start + t] = x1 - (x0 + v0 * CppAD::cos(psi0) * dt);
    fg[1 + y_start + t] = y1 - (y0 + v0 * CppAD::sin(psi0) * dt);
    fg[1 + psi_start + t] = psi1 - (psi0 + v0 * delta0 / Lf * dt);
    fg[1 + v_start + t] = v1 - (v0 + a0 * dt);
    fg[1 + cte_start + t] =
      cte1 - ((f0 - y0) + (v0 * CppAD::sin(epsi0) * dt));
    fg[1 + epsi_start + t] =
      epsi1 - ((psi0 - psides0) + v0 * delta0 / Lf * dt);
```

```
  }
```

This last group are subtractions that should equal zero as a constraint.

As the program continues, we see initialization and then limits for variables.

```
double x = state[0];
 double y = state[1];
 double psi = state[2];
 double v = state[3];
 double cte = state[4];
 double epsi = state[5];
// Initial value of the independent variables.
 // SHOULD BE 0 besides initial state.
 Dvector vars(n_vars);
 for (int i = 0; i < n_vars; i++) {
  vars[i] = 0;
 }
// Set the initial variable values
 vars[x_start] = x;
 vars[y_start] = y;
 vars[psi_start] = psi;
 vars[v_start] = v;
 vars[cte_start] = cte;
 vars[epsi_start] = epsi;

Dvector vars_lowerbound(n_vars);
 Dvector vars_upperbound(n_vars);
 // TODO: Set lower and upper limits for variables.
 for (int i = 0; i < delta_start; i++) {
  vars_lowerbound[i] = -1.0e19;
```

```
  vars_upperbound[i] = 1.0e19;
}

// The upper and lower limits of delta are set to -25 and 25
// degrees (values in radians).
// NOTE: Feel free to change this to something else.
for (int i = delta_start; i < a_start; i++) {
  vars_lowerbound[i] = -0.436332;
  vars_upperbound[i] = 0.436332;
}

// Acceleration/decceleration upper and lower limits.
// NOTE: Feel free to change this to something else.
for (int i = a_start; i < n_vars; i++) {
  vars_lowerbound[i] = -1.0;
  vars_upperbound[i] = 1.0;
}

// Lower and upper limits for the constraints
// Should be 0 besides initial state.
Dvector constraints_lowerbound(n_constraints);
Dvector constraints_upperbound(n_constraints);
for (int i = 0; i < n_constraints; i++) {
  constraints_lowerbound[i] = 0;
  constraints_upperbound[i] = 0;
}
constraints_lowerbound[x_start] = x;
constraints_lowerbound[y_start] = y;
constraints_lowerbound[psi_start] = psi;
```

```
constraints_lowerbound[v_start] = v;
constraints_lowerbound[cte_start] = cte;
constraints_lowerbound[epsi_start] = epsi;
constraints_upperbound[x_start] = x;
constraints_upperbound[y_start] = y;
constraints_upperbound[psi_start] = psi;
constraints_upperbound[v_start] = v;
constraints_upperbound[cte_start] = cte;
constraints_upperbound[epsi_start] = epsi;
```

Rest of the program continues with the Solver and returns the information at the end.

Return code was from the Hangout video; commented return code was from the quiz.

Objective of the code is to minimize the cost within the constraints in determining the predicted trajectory.

```
// object that computes objective and constraints
FG_eval fg_eval(coeffs);
//
// NOTE: You don't have to worry about these options
//
// options for IPOPT solver
std::string options;
// Uncomment this if you'd like more print information
options += "Integer print_level  0\n";
// NOTE: Setting sparse to true allows the solver to take advantage
// of sparse routines, this makes the computation MUCH FASTER. If you
// can uncomment 1 of these and see if it makes a difference or not but
// if you uncomment both the computation time should go up in orders of
// magnitude.
```

```
options += "Sparse  true       forward\n";
options += "Sparse  true       reverse\n";
// NOTE: Currently the solver has a maximum time limit of 0.5 seconds.
// Change this as you see fit.
options += "Numeric max_cpu_time         0.5\n";
// place to return solution
CppAD::ipopt::solve_result<Dvector> solution;

// solve the problem
CppAD::ipopt::solve<Dvector, FG_eval>(
   options, vars, vars_lowerbound, vars_upperbound, constraints_lowerbound,
   constraints_upperbound, fg_eval, solution);

// Check some of the solution values
ok &= solution.status == CppAD::ipopt::solve_result<Dvector>::success;

// Cost
auto cost = solution.obj_value;
std::cout << "Cost " << cost << std::endl;

// TODO: Return the first actuator values. The variables can be accessed with
// `solution.x[i]`.
//
// {...} is shorthand for creating a vector, so auto x1 = {1.0,2.0}
// creates a 2 element double vector.
//return {solution.x[x_start + 1],   solution.x[y_start + 1],
      //solution.x[psi_start + 1], solution.x[v_start + 1],
      //solution.x[cte_start + 1], solution.x[epsi_start + 1],
      //solution.x[delta_start],   solution.x[a_start]};
```

```
 vector<double>result;
 result.push_back(solution.x[delta_start]);
 result.push_back(solution.x[a_start]);
 for (int i = 0; i < N - 1; i++){
    result.push_back(solution.x[x_start + i + 1]);
    result.push_back(solution.x[y_start + i + 1]);
 }
 return result;
```

THE RUNS

Hangout MPC Video Code

N = 10 dt = 0.1 Car moves slowly ahead but waypoints and path bending ahead off road

N = 20 dt = 0.05 Car maybe barely moving, though path straighter, waypoints still bending

N = 5 dt = 0.1 Car maybe barely moving, path straighter

Go back to N = 10, dt = 0.1

Modify costs with CppAD multiplier, ref cte = 0, ref epsi = 0, ref v = 40

N	N	N	N-1	N-1	N-2	N-2	
Cte	epsi	v	delta	a	delta	a	
2000	2000	1	5	5	200	10	original from lecture video
1000	2000	1	5	5	200	10	only slowly moving, increase speed to 70
1000	2000	1	5	5	200	10	slowly moving, waypts off, straighter path
1000	1000	1	5	5	200	10	slowly moving, both waypts and path varying
1000	1	1	5	5	200	10	same as previous
1000	1	1	1	1	1	1	barely moving forward, waypts off, path varying
1	0	0	0	0	0	0	similar, change dt = 0.2, no green path, no move
1	0	0	0	0	0	0	N 5, dt .05, no green path, no movement
1	0	0	0	0	0	0	N 5 dt .1, 70mph, no green, no move
10	0	0	0	0	0	0	N 10 dt .1, no green, no move

1000	1000	1	1	1	1	1	same, back to 40mph, green back, little move
1000	1000	1	1	0	1	0	-steer value, waypts off, green ok, no move
1000	1000	10	10	10	10	10	same, barely moving, waypts and green close
1000	1000	10	10	1	10	1	epsi calc, best, 3mph but stopped, straighter
1000	1000	10	10	1	10	1	back to +steer value, best 3mph for awhile
1500	1000	10	10	1	10	1	worse
1000	1500	10	10	1	10	1	best awhile, but lines poor and reversed
1000	500	10	10	1	10	1	waypt problem, but green straight

*1000	1000	10	10	1	10	1	***switch transform on x, yellow straight**, green?
1500	1000	10	10	1	10	1	yellow straight, some green, some move
1500	1000	10	10	1	10	1	-steer value, straighter car, some green
2000	1000	10	10	1	10	1	similar, slow move
2000	1000	10	10	1	10	1	25 N .05 dt, not moving
200	100	10	10	1	10	1	20 N .1dt, not moving
1	0	0	0	0	0	0	10 N .1dt, no moving
10	0	0	0	0	0	0	no moving
10	0	0	0	1	0	0	no moving
10	0	1	0	1	0	0	yellow, green straight, no moving
10	0	1	0	1	0	1	lines present, revving, but not moving
10	0	1	0	1	1	1	lines present, still revving, no move
10	0	1	1	1	1	1	revving, no move, no green line
10	1	1	0	1	1	1	flashing green, not straight, no move
10	1	1	0	1	0	1	revving, no move, no green
10	1	1	1	1	0	1	revving, no move, no green
10	1	1	1	1	1	1	revving, no move, not straight green
10	1	1	1	1	1	1	go back to old epsi, no change
10	0	1	0	1	0	1	straight lines, rev, no move

100	0	1	0	1	0	1	same
1000	0	1	0	1	0	1	same, but some throttle
2000	0	1	0	1	0	1	same, tiny throttle, no move
2000	0	1	0	1	0	10	some movement forward
2000	0	1	0	10	0	10	worse, less movement
2000	0	1	0	1	0	100	same
2000	0	1	0	5	0	10	original, no move
2000	0	1	0	0	0	10	no move
2000	0	1	0	1	0	20	no move
2000	0	1	0	1	0	5	some movement forward
2000	0	1	0	1	0	5	change x, y to 0,0 original, no move
2000	0	1	0	1	0	5	flip y transform, very wrong so put back
2000	0	10	0	1	0	10	
2000	2000	1	5	5	200	10	lecture video no green, backward
3000	0	1	0	0	1	10	no movement
3000	0	1	0	0	1	10	remove Lf, no move
1000	0	1	0	1	1	10	
1000	1	5	1	5	1	10 cost8500	
1000	1	1	1	5	1	10 cost 2200 little move, slow, green off	
5000	1	1	1	5	1	10 cost 5000	
2500	1	1	1	5	1	10 cost 3050 some moving	
2000	1	1	1	5	1	10 cost 2700	
1500	1	1	1	5	1	10 cost 2400	
500	1	1	1	5	1	10 cost 1875	
500	1	1	1	5	1	5 cost still 1875, no move	
500	5	1	1	5	1	5 cost still 1875, no move	
100	1	1	1	5	1	5 cost 1650, still no move	
100	1	1	1	5	1	5 put Lf back json, no change	
100	1	1	1	5	1	5 put Lf back bounds, some move, ugly green	

100	1	1	1	5	1	5steer back to +, green lines worse
100	1	1	1	5	1	5steer back to -, x10, forward briefly, ugly green
100	1	1	0	5	1	5forward more, ugly green
100	1	1	0	5	0	5 worse
100	0	1	0	5	1	5 no move, green straight
100	0	1	0	5	1	5throttle multiplier10000 no move
100	0	1	0	1	1	1 green straight no move, increase throttle
1000	0	1	0	1	1	1 throttle multiplier causing array issues
1000	0	1	0	1	1	1 timestep.05, no throttlemultiplier, no move
1000	1000	1	0	1	1	1

Solved car movement by changing json to vars[6] steer angle and vars[7] throttle but crashing by turning

Changed initialization with small v, still crashed by turning

Changed initialization with x, y from start -40, 108 still crashed by turning

Remove Lf from delta limits and steer angle, no impact, crashed on turning

Change cost multipliers, back to 0x, 0y, 1v for start

2000	0	1	0	1	0	1 improved but still crashed, turning right

Use original epsi calculation

2000	0	1	0	1	1	1 seems improved, still crashed right
3000	0	0	0	10	10	10 improved, still crashing right
3000	10	0	0	10	10	10 much worse, but went slower
3000	0	0	10	10	10	10 worse, hard turn right
3000	0	0	0	10	100	10 worse, hard turn right
3000	0	0	0	50	10	50 worse, hard turn right
1000	0	0	0	1	10	1 worse, hard turn right
5000	0	1	0	10	10	10 better, but still hard right
5000	0	1	0	10	20	20 better

15000	0	1	0	10	20	20
15000	5	1	0	10	20	20 speed5 back and forth deciding refcte1
15000	5	1	0	10	20	20 speed2 refcte0.2, refepsi0.1 similar
15000	5	1	0	10	20	20put Lf back to reduce steer, worse
15000	20	1	0	10	20	20 Lf out, crashed early
15000	20000	1	0	10	20	20 back and forth
15000	10000	1	0	10	20	20 20N .2dt back and forth
10000	5000	1	10	10	20	20 back and forth
10000	2500	1	10	10	20	20 back and forth
50000	2500	1	50000	10	20	20 per forum cte, delta back and forth
5000	2500	1	5000	10	5000	20 per forum cte, delta back and forth
1	1	1	1	1	5000	1 hard left off
4000	1	1	80000	1	1000	1 per A Cui forum mentor, N5, .2dt
40000	1	1	40000	1	40000	1f0 code and first constraint limits per forum
Went a bit generally straight, then stopped						
20000	0	1	0	1	20000	1 turned right quickly
40000	0	5	0	1	2000	1 was trying to stay on, but going backwards
60000	0	20	0	5	2500	5 Lf back for steer, similar,10refv
80000	0	100	0	100	3000	100 got up hill before crashing off right
80000	0	1	0	1	5000	1 crashed to left
80000	0	1	0	1	5000	1 no Lf steer, better
80000	0	1	1000	10	5000	10, almost up hill
80000	0	1	1000	10	5000	10, crashed ref cte 0.5
80000	0	1	100	100	5000	100 almost up hill, crashed on right
80000	0	100	100	100	5000	100 crashed
80000	0	1	100	100	10000	100 did not get far, turning hard
80000	0	1	0	1000	5000	1000 not up hill
80000	0	1	0	500	5000	500 not up hill
80000	0	1	0	300	5000	300 not up hill

80000	0	1	0	300	2000	300 tried to go straight and crashed on left

80000	**0**	**1**	**0**	**1**	**3000**	**1 back to vars[0], vars[1] per forum mentor**

Also put lower limit on a = 0.0, trying at 0.2 speed

80000	0	1	0	0	3000	0 went up hill and straight, crashed to right
80000	0	1	0	1	3000	0 very slow braking the car
80000	0	1	0	0	3000	1 very slow braking the car
80000	0	0	0	0	3000	0 no green line, went up hill, straight, but crash
80000	0	1	0	0	5000	0 green line, went up hill on edge of road
80000	0	1	0	0	5000	0 Lf back in steer, bit worse, remove it
80000	0	1	0	0	8000	0 seemed worse
80000	0	1	0	0	15000	0 held longer then crashed to right
80000	0	1	0	0	25000	0 still riding rail on right
80000	0	1	0	0	50000	0 still riding rail on right
80000	0	2	0	0	50000	0 still same
80000	0	2	0	0	50000	0 go back to +ve steer, even worse

80000	0	10	0	0	50000	0 -ve steer seemed better but still right rail
80000	0	100	0	0	50000	0 improved as stayed in middle but too fast for curve
80000	0	200	0	0	50000	0 speed too high for curve 40+, lower refv to 10
80000	0	200	0	0	50000	0 seemed worse, return to 20 refv
80000	0	500	0	0	50000	0 still riding rail, go back to 150
80000	0	150	0	0.1	50000	0.1 low a little moving
80000	0	150	0	0	50000	0 refv=0, worse? Just green dot
80000	0	300	0	0	50000	0 refv=1, worse? Just short green dots
80000	0	300	0	0	80000	0 refv20 green line, but still to right edge
80000	0	1000	0	0	80000	0 similar, try refv40 from quiz

80000	0	1000	0	0	80000	0 better, solid green, still to right edge
80000	0	1000	0	0	80000	0 remove ref cte per quiz, still to right, but dif look
80000	1000	1000	0	0	80000	0 0refepsi,epsi huge impact to left, but no green line
80000	100	1000	0	0	80000	0 green flash, but to right side
*80000	500	1000	0	0	80000	0 N10, 0.1dt, got up hill, crashed on sharp curve
80000	500	1000	0	0	80000	0 Try N20, 0.05dt, worse, right edge of road
80000	500	1000	0	0	80000	0 N10, 0.1dt try cubic epsi cost, worse
80000	750	1000	0	0	80000	0 N10, 0.1dt remove cubic, to right side
80000	750	1000	0	0	80000	0 0.75throttle, best run, crash sharp curve
80000	1000	1000	0	0	80000	0 0.5throttle, 1000 kicked left crash
80000	750	1000	0	0	80000	0 0.25throttle, farthest, crashed at sharp curve
80000	750	1000	0	0	80000	0 0.10throttle, slow, still crashed at sharp curve
80000	750	1000	100	0	80000	0 0.20throttle, a= -1.0 again, no move
*80000	750	1000	100	1	80000	1 try a = 0.1 a costs now working,
slow 2mph but stopped right side on bridge						
*80000	750	1000	100	1	80000	1 0.5throttle, crashed curve before bridge,6mph
80000	750	1000	100	1	80000	1 0.75throttle, same crash, 9mphf
80000	750	1000	100	1	80000	1 0.75throttle, f2 epsi sign back to -ve, same crash
80000	1000	1000	1000	1	80000	1 0.5throttle
100000	2000	1000	1000	1	80000	1 0.5throttle same crash

Merged Hangout Video and Quiz Models

Switch models-merging main of video with mpc of quiz (return from video) per Spartacus rx in forum

200	200	1	1	1	1	1 0.5throttle, quick crash, but a works and good lines
2000	2000	1	1	1	1	1 same result, quiz model has N25, .05dt
4000	1	1	80000	1	1000	1 per A Cui forum mentor similar crash, good lines
80000	1000	1	80000	1	80000	1 immediate crash, curving as with previous
2000	2000	1	5	5	200	10 video numbers, still crashing
80000	500	100	1000	20	5000	20 .25throttle still crashing

*8000	50	100	1000	20	500	20	.10throttle SUCCESS, 5mph, but no crash
800	50	100	100	20	500	20	SUCCESS, 10mph, no crash, N10, .1dt
200	50	100	100	20	100	20	SUCCESS, 10mph, no crash, oscillating
200	50	100	100	20	1000	20	0.2throttle, 15mph, no crash, more oscillating
200	50	100	1000	20	10000	20	0.3throttle, 30mph, no crash, much oscillating
200	50	100	10000	20	10000	20	no change
200	50	100	30000	20	30000	20	SUCCESS, 35mph, no crash, very smooth
200	50	100	30000	20	30000	20	0.4Throttle, 40mph, no crash, some oscillating
200	50	100	50000	20	50000	20	better, oscillation on last S curve
200	50	100	70000	20	70000	20	0.5Throttle, 40mph, smooth
**200	50	100	70000	20	70000	20	0.75Throttle, still 40mph due to braking, smooth
200	50	100	70000	20	70000	20	refv40 to 60, 0.75throttle, oscillated, crash bridge
200	50	100	90000	20	90000	20	crashed after bridge, oscillation, 10N .1dt
200	50	100	90000	20	90000	20	20N, .05dt, much worse, crashed early
200	50	100	90000	20	90000	20	5N, 0.2dt, much worse, crashed early
200	50	100	125000	20	125000	20	10N, 0.1dt, crashed at bridge
1000	100	100	90000	20	90000	20	crashed early
100	100	100	90000	20	90000	20	still crashed at last S curve
100	1000	100	90000	20	90000	20	crashed after bridge
100	100	100	90000	100	90000	100	still crashed after bridge
*200	50	100	200000	20	200000	20	60mph, barely making it, large oscillations
**200	50	100	200000	20	200000	20	refv50, full throttle, some oscillation
**200	100	100	200000	20	200000	20	refv50, full throttle, similar
**200	75	100	200000	20	200000	20	refv50, full throttle SMOOTH
*200	75	100	200000	20	200000	20	refv55, full throttle, barely making it

Return to refv50, full throttle, and incorporate latency time of 0.001 sec

200	75	100	200000	20	200000	20

runs without sleep, but barely hits some road edge markings in S curve

same with chrono sleep command, max speed attained

SUBMITTED RUN

Try refv45, full throttle, latency time of 0.001 sec

200 75 100 200000 20 200000 20

Smoothest run, fully within road between road edge markings