Udacity CarND Term 2 Project 5
Model Predictive Control

Brian Meier
9/2/17

Table of Contents

Project Motivation

Assignment 5 of Term 2 in the Udacity Self-Driving Car Nanodegree Program is to complete the Model Predictive Control Project. The goal of this project is to implement a model predictive control, MPC, algorithm to control a vehicle in the Udacity simulator. Starter code was provided. This code is in the C++ programming language and includes methods for receiving state variables and waypoints and sending actuation values to the simulator. Additional helpful code included code for fitting polynomials, and examples of optimization functions that utilize the COIN-OR Ipopt[1] and COIN-OR CppAD optimization and automatic differentiation packages.

Inputs to the system are the state variables and desired path. The state variables are the car Cartesian coordinates, bearing, speed and current actuations. The current actuations are the steering angle and throttle position. The desired path is conveyed as a set of six sets of Cartesian coordinates. The algorithm outputs a set of actuations that fully define a finite number future states. However, only the first set of actuations is implemented. Then, another set of measurements are taken and the optimization is repeated.

To complete this project, a motion model and cost function were developed. The results of the motion model are fed to a cost function that is minimized in real-time. The outputs of minimization efforts are actuation values that successfully control the car in the Udacity simulator.

Algorithm Description

1. Obtain waypoints and car state.
2. Transform waypoints and car state from map to car reference frame.
3. Fit a third-order polynomial to the waypoints.
4. Minimize a cost function to determine the optimal actuator values at future time steps. The current step is step 0. Future actuator values are given in time steps 1, 2, .. *N*-1. The time between each step is equal to the latency time between a signal being sent the actuation occurring.
5. At the next step, the actuator values are set to the outputs resulting from the previous time step inputs.
6. Return to step 1.

Obtain Waypoints and Car State

Waypoints consist of the next six positions. These are given in the map coordinate system and the unit of measure is meters.

The car state consists of the car x and y coordinates, bearing, speed, and measured steering position and throttle position. Coordinates are in the map reference frame and the unit of measure is meters. The

1. A. Wächter and L. T. Biegler, **On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming**, *Mathematical Programming* 106(1), pp. 25-57, 2006

bearing is the direction the car is heading and is measured in the counter clockwise direction relative to the map x-axis in radians. The car speed is measured in miles per hour.

Variable	Description	UOM	Limits	Ref. Frame
$x_{waypoint1,\,2,\,..\,6}$	Waypoint x-coordinates	meter		Map
$y_{waypoint1,\,2,\,..\,6}$	Waypoint y-coordinates	meter		Map
$x_{car\ map}$	Car x-coordinate	meter		Map
$y_{car\ map}$	Car y-coordinate	meter		Map
$\psi_{car\ map}$	Car bearing, counter clockwise from the x-axis	radians		Map
v_{car}	Car speed	miles/ hour		N/ A
δ_{meas}	Steering angle in the **clockwise** direction	radians	± 25°	Car
$p_{throttle}$	Throttle position	#	± 1	N/ A

Transform Waypoints and Car State

The car x, y and ψ values are each transformed so that they each have a value of 0. Each waypoint is transformed by the below equation.

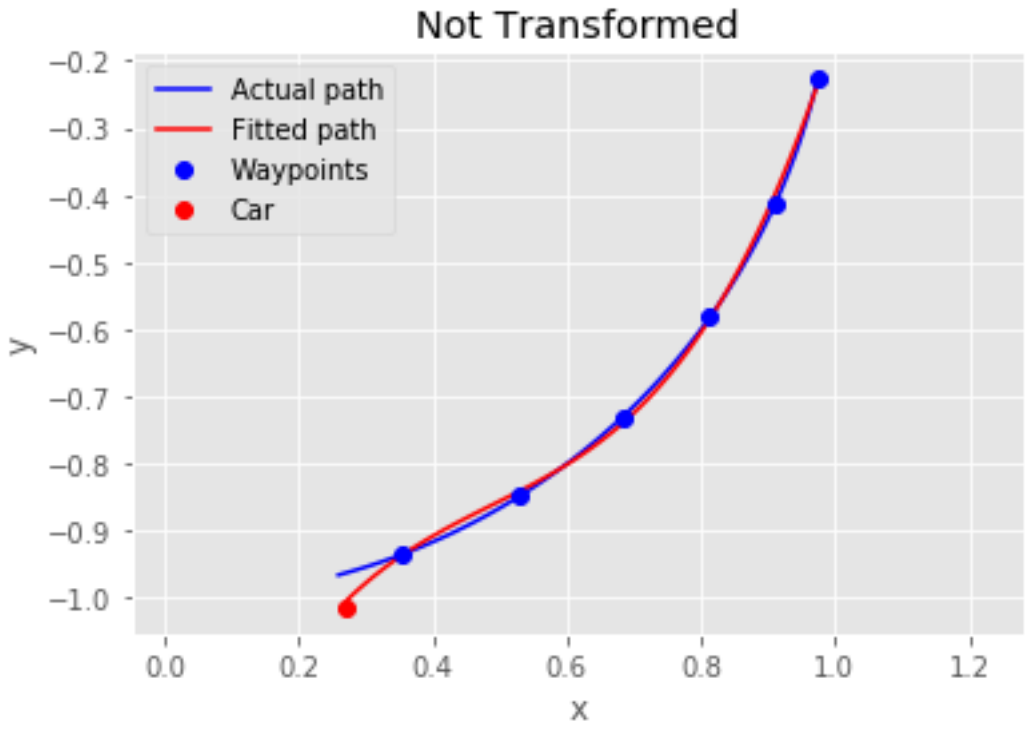
$$\begin{bmatrix} x_{trans} \\ y_{trans} \end{bmatrix} = \begin{bmatrix} \cos(-\psi_{car\ map}) & -\sin(-\psi_{car\ map}) \\ \sin(-\psi_{car\ map}) & \cos(-\psi_{car\ map}) \end{bmatrix} \begin{bmatrix} x_{waypoint} - x_{car\ map} \\ y_{waypoint} - y_{car\ map} \end{bmatrix}$$

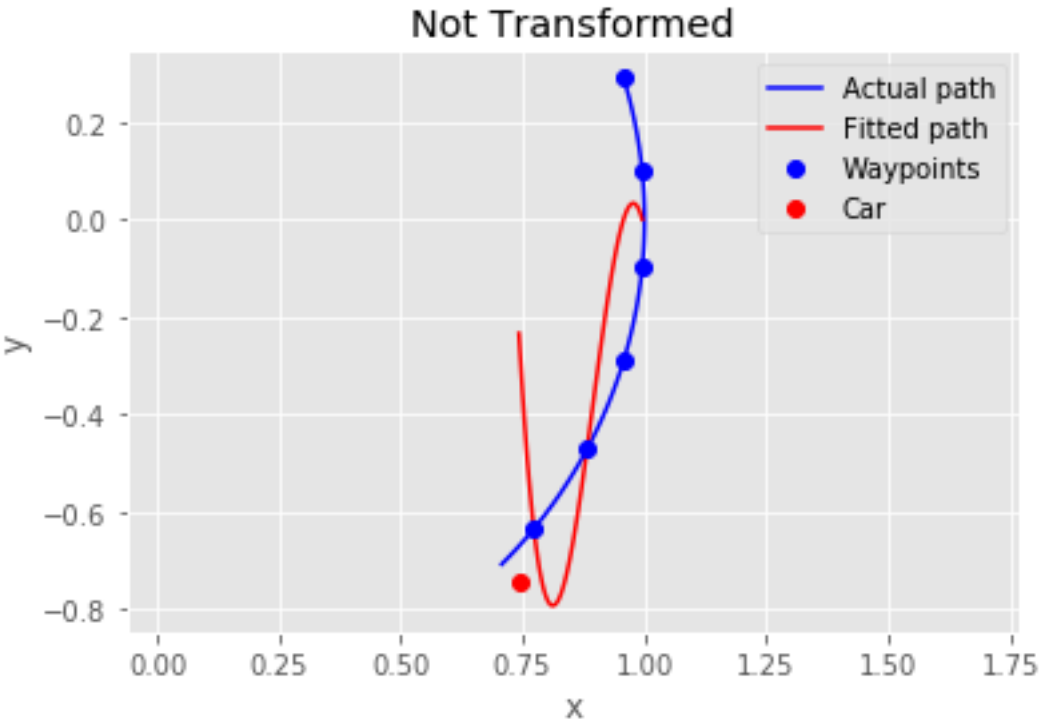
Fit Third-Order Polynomial

Polynomial fits allow for efficient calculation of values and their derivatives. However, if the fit is poor or multi-valued, problems exist. For this project a third-order polynomial fit was used. This fit accommodates changes in curvature and can model S-turns. However, if the data is not transformed, obtaining a function in the below form can lead to poor results. The equation that forms the estimate path, *y*, is given below.

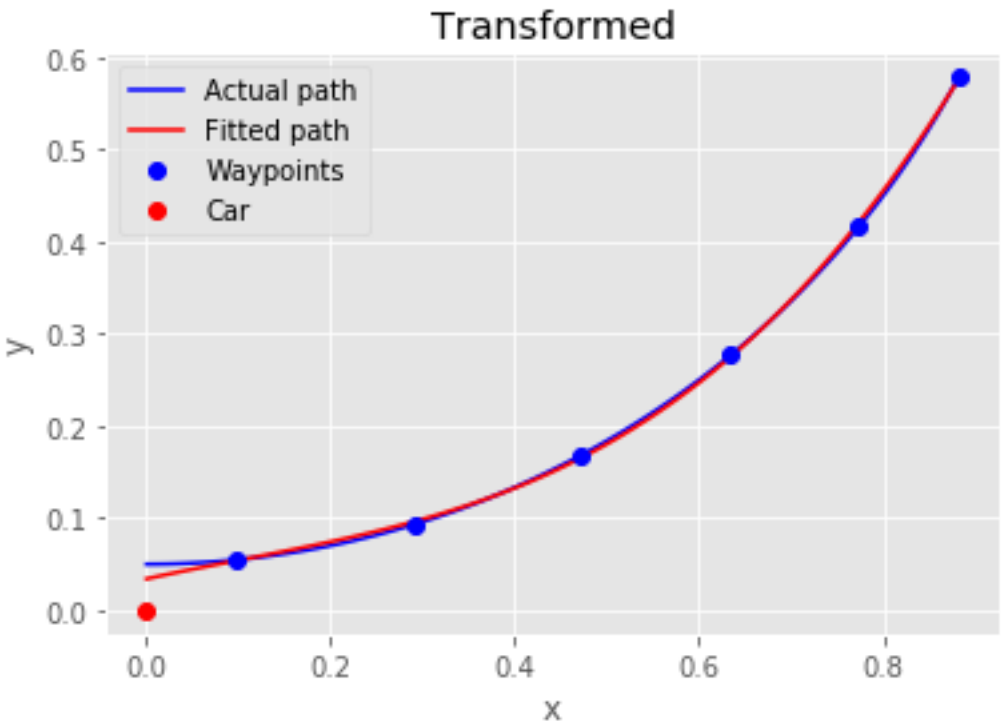
$$y = f(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0$$

To demonstrate potential issues with not performing a transformation, examples of an identical situation are given below. For the below plots, a circle was rotated and fit. A car is slightly offset from the actual circle and the bearing is parallel with the circle at the nearest point on the circle to the car. The only difference is the amount of rotation with respect to the x-axis.


Not Transformed
Actual path
Fitted path
Waypoints
Car
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
y
0.0
0.2
0.4
0.6
0.8
1.0
1.2
x


Not Transformed
Actual path
Fitted path
Waypoints
Car
0.2
0.0
−0.2
−0.4
−0.6
−0.8
y
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
x

Alternatively, transforming the waypoints from map to car coordinates prior to fitting, leads to a good approximation.



An added benefit of performing the translation and rotation transformations is it allows for an efficient estimate of the cross-track error, *CTE*, and the bearing error, ψ_e. The cross-track error and bearing errors are given by the below equations.

$$CTE = c_3 x^3 + c_2 x^2 + c_1 x + c_0 - x_{car} \qquad \psi_e = \frac{df(x)}{dx} - \psi_{car} = 3 c_3 x^2 + 2 c_2 x + c_1 - \psi_{car}$$

Given transformed waypoints, $y_{wt[i]}$ and $x_{wt[i]}$, the below cost function J_{fit}, is minimized to yields the coefficients, c_3, c_2, c_1, and c_0. In the below vectors and matrices, $x_{wt[i]}$ and $y_{wt[i]}$ are the x and y waypoints that have been transformed from the map coordinate system to the car coordinate system.

$$y = \begin{bmatrix} y_{wt1} \\ y_{wt2} \\ y_{wt3} \\ y_{wt4} \\ y_{wt5} \\ y_{w6} \end{bmatrix} \quad X = \begin{bmatrix} x_{wt1}^3 & x_{wt1}^2 & x_{wt1} & 1 \\ x_{wt2}^3 & x_{wt2}^2 & x_{wt2} & 1 \\ x_{wt3}^3 & x_{wt3}^2 & x_{wt3} & 1 \\ x_{wt4}^3 & x_{wt4}^2 & x_{wt4} & 1 \\ x_{wt5}^3 & x_{wt5}^2 & x_{wt5} & 1 \\ x_{wt6}^3 & x_{wt6}^2 & x_{wt6} & 1 \end{bmatrix} \quad c = \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} \quad J_{fit} = (y - Xc)^T (y - Xc) \quad c = (X^T X)^{-1} X^T y$$

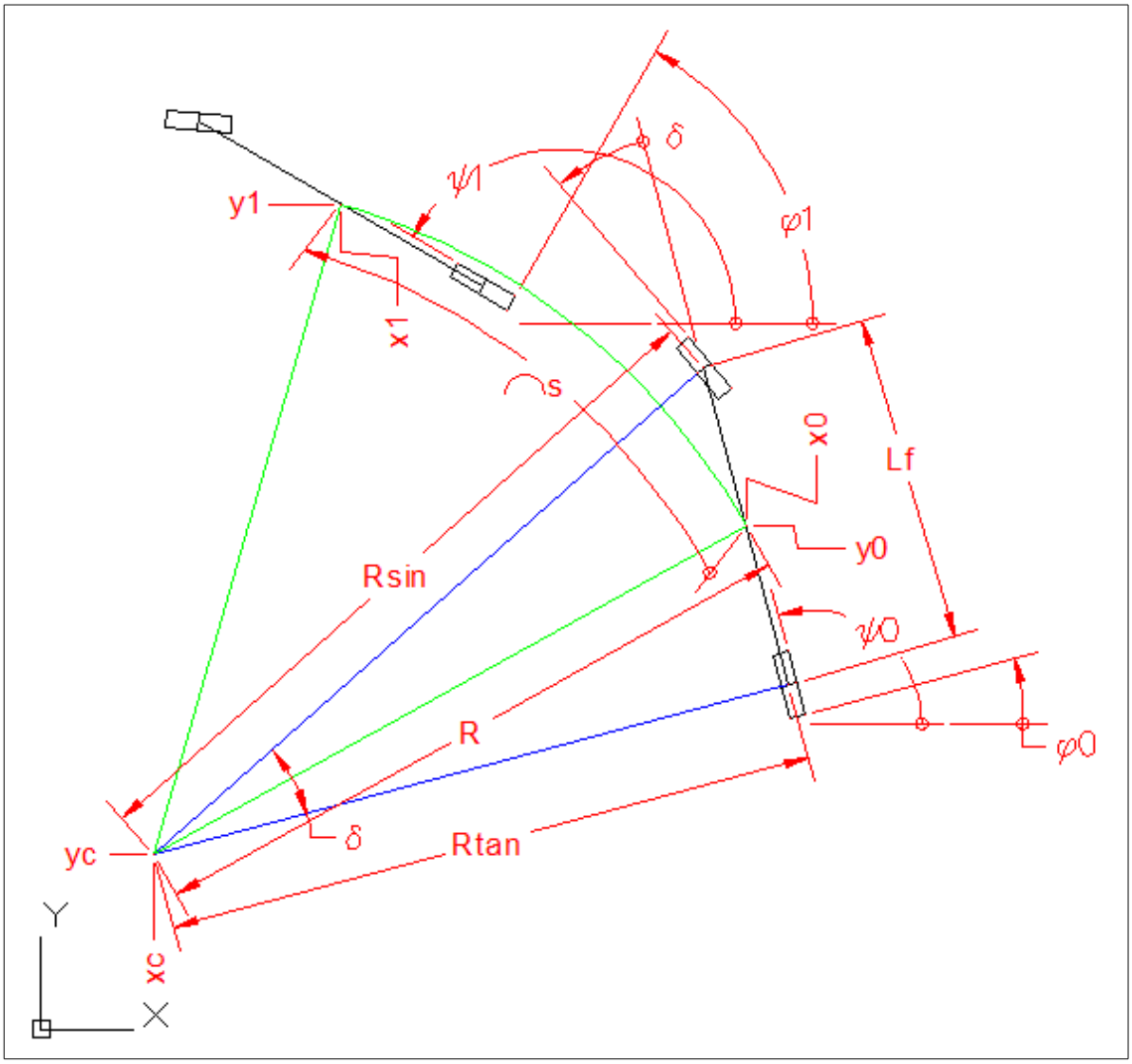
Minimize Cost Function

Three steps are required to determine the cost function.

1. Determine the motion model
2. Define penalty terms
3. Scale and sum penalties

Motion Model

A diagram of the motion model is shown below.



The motion model assumes a constant turning radius and constant acceleration. Below, variables with a subscript of *0* and *1* describe their values at times t and $t + \Delta t$, respectively. Provided system constants are the frame length, L_f, and acceleration to throttle position ratio, a_c. Governing equations are given below.

$$v_{avg} = v_0 + \frac{1}{2} a \Delta t \qquad v_1 = v_0 + a \Delta t \qquad a = a_c \, p_{throttle}$$

If δ is zero, then the below equations hold.

$$x_1 = x_0 + v_{avg} \cos(\psi_0) \Delta t \qquad y_1 = y_0 + v_{avg} \sin(\psi_0) \Delta t \qquad \psi_1 = \psi_0$$

If δ is not zero, then the below equations hold. It is noted that the turning radius, R, is an estimate. The bounds given below show that this estimate is reasonable.

$$R_{\tan} = \frac{L_f}{\tan(\delta)} \; < \; R = \frac{L_f}{\delta} \; < \; R_{\sin} = \frac{L_f}{\sin(\delta)}$$

$$v_{avg} = \frac{s}{\Delta t} = \frac{R \Delta \psi}{\Delta t} \qquad \psi_1 = \psi_0 + \frac{v_{avg} \Delta t}{R} = \psi_0 + \frac{\delta}{L_f} \left(v_0 + \frac{1}{2} a \Delta t \right)$$

$$x_c = x_0 - R \cos(\phi_0) \qquad yc = y_0 - R \cos(\phi_0)$$
$$x_1 = x_r + R \cos(\phi_1) \qquad y_1 = y_r + R \sin(\phi_1)$$
$$x_1 = x_0 + R \cos(\phi_1) - R \cos(\phi_0) \qquad y_1 = y_0 + R \sin(\phi_1) - R \sin(\phi_0)$$
$$x_1 = x_0 + R \left(\cos\left(\psi_1 - \frac{\pi}{2}\right) - \cos\left(\psi_0 - \frac{\pi}{2}\right) \right) \qquad y_1 = y_0 + R \left(\sin\left(\psi_1 - \frac{\pi}{2}\right) - \sin\left(\psi_0 - \frac{\pi}{2}\right) \right)$$
$$x_1 = x_0 + R \left(\cos\left(\frac{\pi}{2} - \psi_1\right) - \cos\left(\frac{\pi}{2} - \psi_0\right) \right) \qquad y_1 = y_0 + R \left(\sin\left(\frac{\pi}{2} - \psi_0\right) - \sin\left(\frac{\pi}{2} - \psi_1\right) \right)$$
$$x_1 = x_0 + R \left(\sin(\psi_1) - \sin(\psi_0) \right) \qquad y_1 = y_0 + R \left(\cos(\psi_0) - \cos(\psi_1) \right)$$
$$x_1 = x_0 + \frac{L_f}{\delta} \left(\sin(\psi_1) - \sin(\psi_0) \right) \qquad y_1 = y_0 + \frac{L_f}{\delta} \left(\cos(\psi_0) - \cos(\psi_1) \right)$$

Definition of Penalty Terms

Errors used in the optimization algorithm are the following:

CTE	Cross-track error, meters
ψ_e	Bearing error, radians
$\Delta\delta$	Change in steering angle in one time step, radians
ΔCTE	Change in cross-track error in one time step, meters
$LA_{setpoint}$	Lateral acceleration set point, meters per second squared
v_{max}	Maximum velocity, meters per second

For a given time step i, errors are given by the below equations.

$$CTE[i]=f(x[i])-y[i] \qquad CTE[i]=c_3x[i]^3+c_2c[i]^2+c_1x[i]+c_0-y[i]$$

$$\psi_e[i]=\frac{df(x)}{dx}[i]-\psi[i]$$
$$\psi_e[i]=3c_3x[i]^2+2c_2+x_1-\psi[i]$$

$$\Delta\delta[i]=\delta[i]-\delta[i-1] \qquad \Delta CTE[i]=CTE[i]-CTE[i-1]$$

$$R_{max}=v_{max}^2/LA_{setpoint}$$
$$\frac{dy}{dx}[i]=3c_3x[i]^2+2c_2x[i]+c_1$$
$$\frac{d^2y}{dx^2}[i]=6c_3x[i]+2c_2$$
$$R_{fit}[i]=\frac{1+\frac{dy}{dx}[i]^{3/2}}{\frac{d^2y}{dx^2}[i]}$$
$$R_{eval}[i]=min(R_{max}[i],R_{fit}[i])$$
$$v_{des}[i]=\sqrt{LA_{setpoint}R_{eval}[i]}$$
$$v_e=v_{des}[i]-v[i]$$

Scaling and Summation of Penalty Terms

The implementation takes as a given the initial state (x, y, ψ, v) and actuations (δ, $p_{throttle}$). After a latency of 100 ms, the next actuation values are implemented. The solution consists of N sets of points, including the initial state at time t = 0, t_0. Actuations are given at times $t_0 + \Delta t$, $t_0 + 2\Delta t$, … $t_0 + (N-2)\Delta t$. Therefore, states are affected at times $t_0 + 2\Delta t$, $t_0 + 3\Delta t$, … $t_0 + (N-1)\Delta t$. The length of each time step, Δt, is set to 100 ms to match the latency. The cost function, J, to be minimized is given below.

$$J=\sum_{i=2}^{N-1}CTE[i]^4+\sum_{i=2}^{N-1}w_{psie}\cdot\psi_e[i]^2+\sum_{i=1}^{N-2}w_{\Delta\delta}\cdot\Delta\delta[i]^2+\sum_{i=2}^{N-1}w_{\Delta CTE}\cdot\Delta CTE[i]^2+\sum_{i=2}^{N-1}w_{ve}\cdot v_e[i]^2$$

Discussion of Optimization

Optimization Inputs and Outputs

With N set to 11, at each time step an 18 input, one output, restricted optimization problem is solved. The inputs are the speed, steering angle, throttle position and a set of coefficients fit to the waypoints in the car coordinate system. By using the car coordinate system, the initial values for each coordinate and bearing are zero. Minimizing J yields the outputs which consist of the steering angle, δ, and throttle position, $p_{throttle}$, at times $t_0 + \Delta t$, $t_0 + 2\Delta t$, .., $t_0 + 9\Delta t$. The steering angle, δ, is restricted to ±25º. The throttle position, $p_{throttle}$ is restricted to ± 1. For the steering angle, the value is inverted prior to

being sent to the simulator so that negative values turn the car left and positive values turn the car to the right. For the throttle position, a value of -1 indicates full brake activation, and a value of +1 indicates full acceleration activation.

Selection of Parameter Values

Parameter values and qualitative descriptions of their impact on performance are given in the below table.

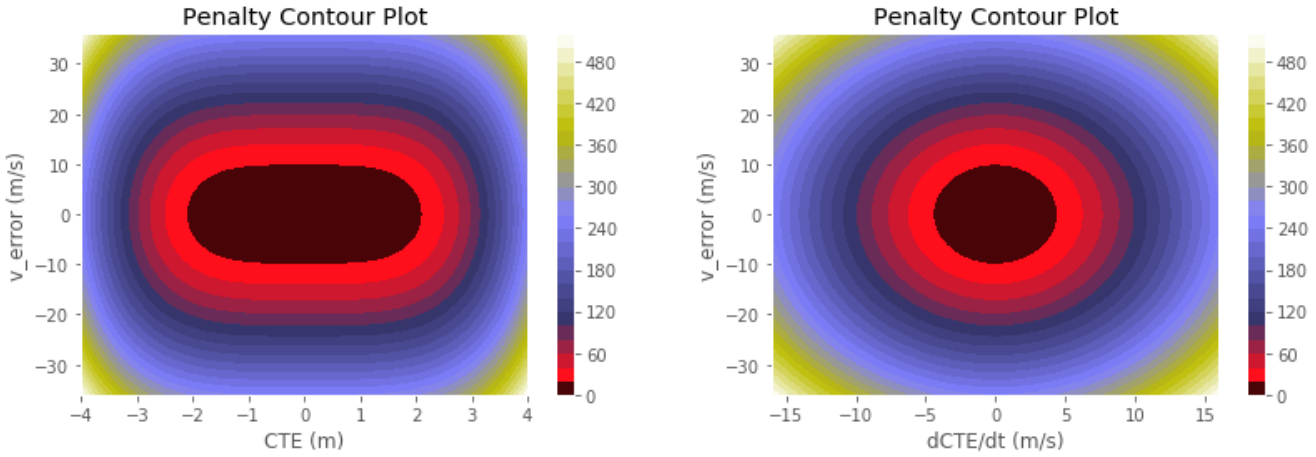
Symbol	Name	Value	Description
L_f	Frame length	2.67 m	Provided system constant. Along with the maximum steering angle, δ, of 25°, this yields a minimum turning radius of 6 m.
a_c	Acceleration constant	1 m/s^2	Provided system constant. Acceleration to throttle position ratio.
N	Number of points	11	In combination with a Δt equal to 0.1 s, sets the time horizon to 1 s.
Δt	Time step	0.1 s	Set to match actuation latency.
CTE exponent	NA	4	Penalize large errors much more than small errors. Allows for small deviations, but not large ones. Helps with both stability and safety.
$w_{\psi e}$	Bearing error weight	10/ radian	Amplifies bearing error. Helps keep car parallel with path. Use of larger values results in a more stable solution, but increases the time for the car to return to the path centerline.
$w_{\Delta\delta}$	Change in steering angle weight	1000/ radian	Penalizes changes in steering angle. Prevents high frequency induced instability, but reduces the time to match changing road curvature.
$w_{\Delta CTE}$	Change in cross-track error weight	100/ m	Penalizes changes in cross-track error. Like the bearing error penalty, this helps keep the car parallel with the path. It also increases the time for the car to return the car to the path centerline. At constant speeds, this term is not needed and this penalty can be replaced with a larger value for $w_{\psi e.}$ If the speed setpoint is variable, as it is in this implementation, this term is required to damp low frequency oscillations.
v_{max}	Maximum velocity setpoint	120 mph	Sets maximum velocity setpoint.
$LA_{setpoint}$	Lateral acceleration setpoint	40 m/s^2	Used to set the maximum lateral acceleration. The lower of the velocity resulting from this calculation and the v_{max} value is used to set the desired speed.

Symbol	Name	Value	Description
w_{ve}	Velocity error weight	0.2 m^2/s^2	Penalize speed errors. If the value is too small, unnecessary braking occurs. If the value is too large, other errors are not able to cause the vehicle to slow.

Comparison of Penalty Contributions

The table below shows a penalty, *J*, in the left most column. The other columns show the required cross-track error, *CTE*, bearing error, ψ_e, rate of change of steering angle, *dδ/dt*, rate of change of cross track error, *dCTE/dt*, and speed error, v_e, that equal the penalty listed in the first column. Because the cross-track error is raised to the fourth power while the other terms are squared, small cross-track errors trigger proportionally smaller responses than large cross-track errors trigger. This is desirable as small path deviations are harmless, but large path deviations can cause the car to crash. These different responses are shown in the Penalty Contour Plots, below.

J (#)	*CTE (m)*	*Ψ_e (degree)*	*dδ/dt (degree/s)*	*dCTE/dt (m/s)*	*v_e (m/s)*
0.00391	0.250	1.13	1.13	0.0625	0.140
0.0625	0.500	4.53	4.53	0.250	0.559
1	1.00	18.1	18.1	1.00	2.24
16	2.00	72.5	72.5	4.00	8.94
256	4.00	260	260	16.0	35.8



Results

As required, the program was successfully compiled without errors using **cmake** and **make**. It is run in an Ubuntu 16.04 LTS environment on a Dell Inspiron-15-7000-Gaming laptop with an Intel® Core™ i5-7300HQ CPU @ 2.50GHz × 4 core specification. The MPC implementation successfully controls

the car. Below are screenshots of, and a link to, a video of the car in the simulator. As shown, the car achieves a top speed of 109 mph while safely navigating the track.

[MPC Project Video]

Conclusion

A Motion Predictive Control algorithm has been described and implemented.