# Obstacle Avoidance via Model Predictive Control with Spatial Constraints

**Jay Guthrie**
guthriejd1@gmail.com

## 1 Introduction

Obstacle avoidance is a crucial component of motion planning and control algorithms for self-driving vehicles. Often the motion planning algorithms consider the vehicle to be a point mass. To account for the dimensions of the vehicle, obstacles are inflated to ensure there is sufficient margin/clearance despite planning with a point mass. For aggressive maneuvers which strongly couple the vehicle chassis dynamics and tire nonlinearities, this conservative approach is not adequate. Instead it is necessary to develop real-time planning algorithms that can work with higher fidelity vehicle models while still addressing obstacle avoidance. A very recent paper from the Stanford Dynamic Design Lab proposes one approach to achieving this goal. This project attempts to replicate the results of [1].

Section 2 introduces the model of [1]. Section 3 discusses the formulation of the obstacle avoidance problem. Section 4 provides the results. Section 5 discusses a related topic of estimating the region of attraction of a vehicle's lateral dynamics which was explored while completing this project. Section 6 concludes the paper.

## 2 System Model

Figure 1 shows the single-track bicycle model used in [1]. The model is represented in path coordinates with $s$ being the distance along the path and $e$ being the lateral displacement. The vehicle states are: $r$ (yaw rate), $u_y$ (lateral velocity at center-of-mass), $u_x$ (longitudinal velocity at center-of-mass), $\Delta\psi$ (relative heading), and path coordinates $s$ and $e$. The tire model is a slightly modified Pacejka model with the maximum lateral force derated by the current longitudinal force. The normal forces $F_{zf}, F_{zr}$ are calculated assuming a static equilibrium. For convenience, the equations and parameters are reprinted below from [1].

**Reformulation:** In its original form, the control signals would be the steering input $\delta$ and the front and rear longitudinal forces $F_{xf}, F_{xr}$. It is assumed the forces are tracked by a lower level controller. The authors note that it is difficult for the underlying optimization methods to ensure the front and rear longitudinal forces have the same sign (to prevent the motor and brakes from fighting each other). To address this, they rewrite the problem with $F_x$ and $\lambda$ as control signals, with the former being the total longitudinal force and the latter being how it is distributed between the front and rear tires when braking. Driving (vice braking) is assumed to occur with a fixed distribution between the front and rear tires. With this formulation, it is possible to ensure the front and rear forces are of the same sign by appropriately distributing $F_x$ according to $\lambda$. For further details please see [1].

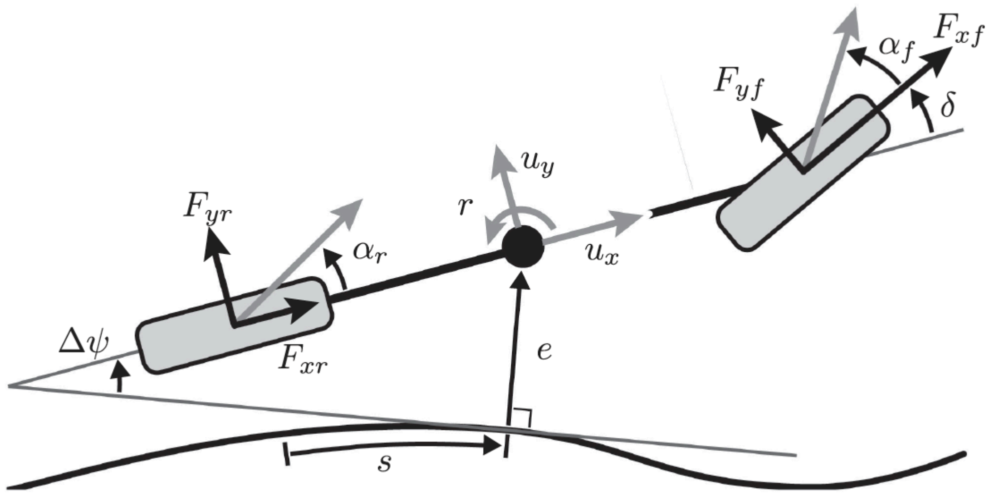

Figure 1: Single-Track Model in Curvilinear Coordinate System

$$\dot{r} = \frac{1}{I_{zz}}\left(aF_{yf}\cos\delta + aF_{xf}\sin\delta - bF_{yr}\right) \qquad F_{y,\text{max}} = \sqrt{(\mu F_z)^2 - {F_x}^2}$$

$$\dot{u}_y = \frac{1}{m}\left(F_{yf}\cos\delta + F_{xf}\sin\delta + F_{yr}\right) - ru_x \qquad F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3F_{y,\text{max}}}|\tan\alpha|\tan\alpha \ldots \\ -\frac{C_\alpha^3}{27(F_{y,\text{max}})^2}\tan^3\alpha, & |\alpha| < \tan^{-1}\left(\frac{3F_{y,\text{max}}}{C_\alpha}\right) \\ -F_{y,\text{max}}\,\text{sgn}\,\alpha, & \text{otherwise} \end{cases}$$

$$\dot{u}_x = \frac{1}{m}\left(-F_{yf}\sin\delta + F_{xf}\cos\delta + F_{xr} - F_{x\text{drag}}\right) + ru_y$$

$$\Delta\dot{\psi} = r - \kappa\dot{s}$$

$$\dot{s} = \frac{u_x\cos\Delta\psi - u_y\sin\Delta\psi}{1-\kappa e} \qquad \alpha_f = \tan^{-1}\left(\frac{u_y + ar}{u_x}\right) - \delta \qquad F_{zf} = \frac{1}{L}(mbg - h_{cm}(F_{xf} + F_{xr}))$$

$$\dot{e} = u_x\sin\Delta\psi + u_y\cos\Delta\psi \qquad \alpha_r = \tan^{-1}\left(\frac{u_y - br}{u_x}\right) \qquad F_{zr} = \frac{1}{L}(mag + h_{cm}(F_{xf} + F_{xr}))$$

| Parameter | Description | Unit | Value |
|---|---|---|---|
| $a$ | distance between CoM and front axle | m | 1.53 |
| $b$ | distance between CoM and rear axle | m | 1.23 |
| $m$ | mass | kg | 2000 |
| $I_{zz}$ | moment of inertia in vertical direction | kg·m$^2$ | 3764 |
| $h_{cm}$ | height of CoM above ground | m | 0.3 |
| $C_{\alpha f}$ | front cornering stiffness | kN/rad | 150 |
| $C_{\alpha r}$ | rear cornering stiffness | kN/rad | 280 |
| $\mu$ | coefficient of friction | - | 0.9 |
| $C_{d0}$ | constant drag term | N | 241 |
| $C_{d1}$ | linear drag coefficient | N·s/m | 25.1 |
| $\delta_{\text{UB/LB}}$ | bounds on steering angle | deg | $\pm18$ |
| $\dot{\delta}_{\text{UB/LB}}$ | bounds on steering angle rate | deg/s | $\pm90$ |
| $F_{x,\text{max}}$ | maximum force due to motor torque limit | kN | 7.2 |

## 3 Problem Formulation

The authors of [1] propose a model predictive control (MPC) algorithm that tracks a desired path with given curvature $\kappa$. Unique to their approach is that they do not treat the vehicle as a point mass when addressing obstacle avoidance. Instead, they approximate the vehicle and obstacles using a combination of circles as shown in Figure 2. By using circles one can encode minimum clearance constraints by ensuring the centers of two circles (one a vehicle, one an obstacle) have a minimum distance between them that exceeds the sum of their radii.

**Sidenote:** This project originally aimed to replace the circle-based models with more realistic quartics (or higher-order) polynomials that more closely approximate rectangular shapes. Upon beginning the project it was realized this was a problematic idea as distance between generic sets (even convex ones) do not generally admit closed-form solutions. Distance between two convex sets can be calculated using convex optimization, but this would require a bilevel optimization problem in which the optimization problems computing distance are embedded within a larger optimization problem computing the trajectory. This is orders-of-magnitude more complicated and not realistic for a class project.

The authors pose the problem as a standard nonlinear MPC algorithm with a look-ahead time of 2.5s (time step 50ms, so N = 50 stages). It is assumed that a higher-level planner has provided a crude reference trajectory of desired lateral displacement $e^*$ and forward velocity $u_x^*$ relative to position $s$. This is tracked with standard quadratic penalties (weighted by parameters $Q$). To prevent overly aggressive control, penalties are also applied on the derivatives of steering angle and longitudinal force. Lastly, a penalty on $\lambda$ encourages a 70/30 split when braking (i.e. braking with the front tires is preferred).

$$J = \int_{T(0)}^{T(f)} Q_{u_x}(u_x - u_x^*)^2 + Q_e(e - e^*)^2 + Q_\delta\dot{\delta}^2 + Q_{\dot{F}_x}\dot{F}_x^2 + Q_\lambda(\lambda - \lambda_{nat})^2 \tag{1}$$

**Constraints:** In addition to penalizing control action, hard constraints are imposed on steering angle (and its derivative) and longitudinal force.

**Obstacles As Soft Constraints:** The authors formulate the obstacle avoidance constraints as additional penalized functions that are added to the objective. They use a quadratic penalty on signed distance functions with a logical/switch condition that causes it be inactive (zero) when the vehicle is sufficiently far from the obstacle. They note that the function is continuous in terms of its first derivative. However, they utilize a nonlinear interior point solver which typically requires functions to be twice continuously differentiable. In attempting to replicate this paper, convergence issues were witnessed when using this formulation with the open-source solver IPOPT. For that reason it was decided to formulate obstacle avoidance as hard constraints.

**Obstacles As Hard Constraints:** Reformulating a problem with hard constraints leads to its own issues when applied in a receding horizon fashion such as MPC. Crudely speaking, if the solver is returning aggressive trajectories which are operating at the limits of the constraints, it is possible that at the next time step the problem will be infeasible (no solution exists). Thus slack variables are typically introduced to allow temporary violations of constraints. This is similar to soft constraints used above but offers the advantage of being twice continuously differentiable. We introduced a constraint that the vehicle remain a distance 0.7m from the obstacles, but we also added a slack variable which allows this to be violated if necessary.

**Transcription Method:** Nonlinear MPC problems must be transformed into a nonlinear optimization problem. A key component of this is the transcription method used for approximating the continuous dynamics with a discrete-time model with some underlying integration scheme. The authors used a 2nd-order Runge-Kutta (RK2) method to reduce solve times while sacrificing accuracy (and possibly numerical stability). For this project, the more standard 4th-order Runge-Kutta method (RK4) was utilized to minimize the likelihood of numerical instability arising from crude integration. The same time step (50ms) and time horizon (2.5s) was utilized as in the paper. Decision variables were shifted and scaled to all roughly be in the same range (-1 to 1). This is standard practice when using nonlinear optimization solvers such as IPOPT (used here). Gradients and Hessian functions were supplied to IPOPT using the automatic differentiation tool CasADi.

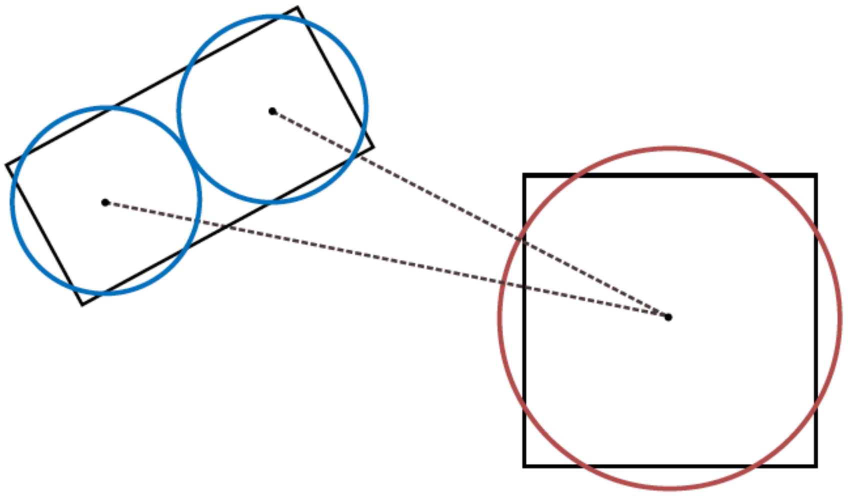

Figure 2: Modeling Vehicle and Obstacle Shapes using Circles

## 4 Results

We attempted to replicate the scenario described in [1]. Unfortunately, not all of the problem data is available so some guesses were made based on the available figures. In the demo scenario, the vehicle is traveling on a two-lane road with no curvature. The center of the road is $e = 0$ and the lane centerlines are at $e = +/-2.25m$. The vehicle is initially traveling in the center of its lane when two obstacles appear. The first is centered at $s = 200, e = -2.25$ and forces the vehicle to go into the other lane. It then must avoid the second obstacle located at $s = 215, e = +2.25$. Figure 3 shows the resulting trajectory. The red trace shows the center-of-mass. For obstacle avoidance, the vehicle is modeled with two circles. The blue trace shows the convex hull of these circles. The gray lines shows the planned trajectory at each time iteration.

This result is a conservative model. The vehicle dimensions are relatively small and the reference velocity is 5 m/s (slow). Figure 4 shows the resulting states and controls. The red dots show the full trajectory while the gray traces show the planned trajectory at each time instant. The vehicle goes around the first obstacle around time 3-5s. During this time, the minimum distance of 0.7m is slightly relaxed to 0.5m. In solving this, no convergence issues were observed.

Figure 5 shows the same general trajectory but with a higher initial velocity (17.5 m/s) and larger vehicle dimensions. Based on our interpretation of the figures in [1], this is trajectory demonstrated therein. Unfortunately the paper does not provide details on the reference speed profile ($u_x^*$) which is to be tracked, only the reference lateral displacement ($e^*$). Figure 6 shows the resulting states and controls. Due to the high initial velocity, the vehicle must brake in order to make it around the first obstacle. While we are able to solve this problem initially, eventually the solver failed (at time 1.3s). Multiple variations in the controller tuning were explored and yet none were able to reliably solve for the whole trajectory with both obstacles present. It is still unclear as to whether a constraint is causing the problem to be infeasible or if the (improvised) reference speed profile is causing issues.

Figure 7 shows the runtime performance of the algorithm for the narrow, slow case when solving with the general solver IPOPT. The solve times are generally about 200ms. Currently the gradient and Hessian functions are MATLAB intepreted code generated by CasADi. From previous experience, compiling these functions to C++ code generally yields a 4-7x speed improvement. In [1] the authors achieve a mean runtime of 15 ms (70ms worst-case) using a custom, compiled solver generated with Forces Pro (commercial software sold by Embotech) which also uses CasADi for gradient and Hessian generation. With additional effort (and perhaps the commercial Forces Pro solver) it is likely our implementation would achieve similar run times. Thus although we have experienced convergence issues, the overall performance suggests we are close to replicating the paper.

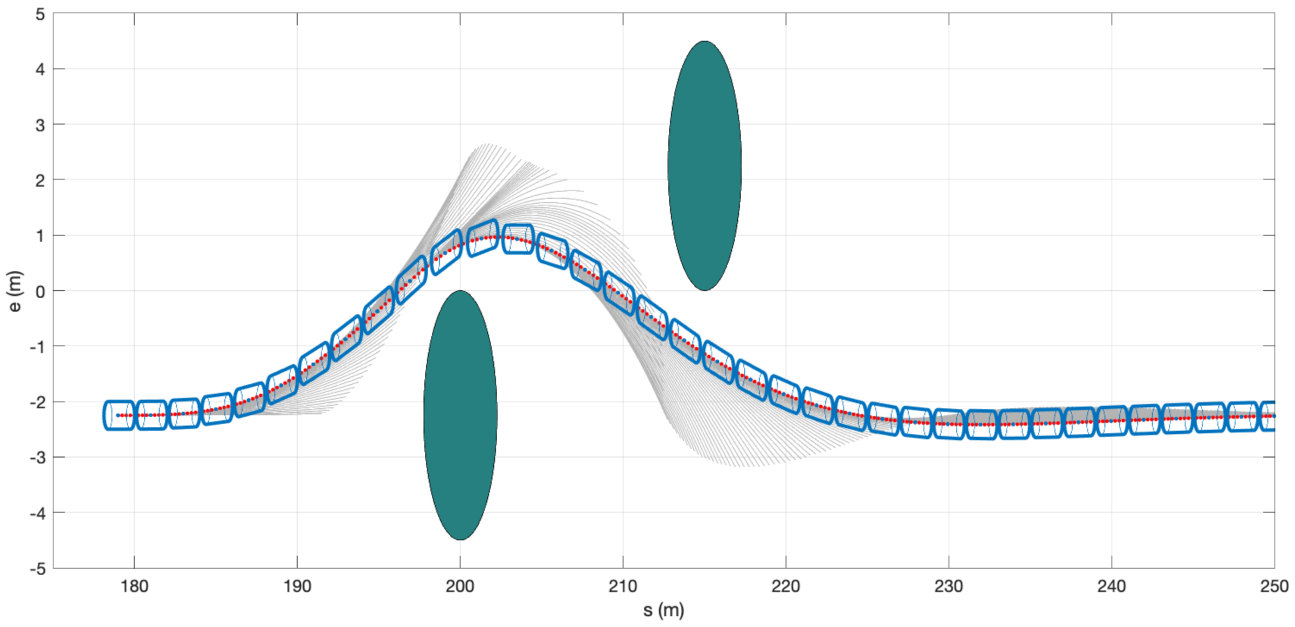

Figure 3: Obstacle Avoidance with Slow, Narrow Vehicle

## 5 Additional Results: Region of Attraction Estimate

The original project plan was to utilize higher-order (non-ellipsoidal) approximations of the vehicle dimensions. As this was infeasible, it was decided to pursue other techniques of interest. In [2], the authors demonstrate the use of sum-of-squares optimization to estimate the region-of-attraction (ROA) of a vehicle's lateral dynamics. The model utilized is the standard bicycle model with the following simplifying assumptions: 1) body-x velocity is constant, 2) the steering angle is constant, 3) small-angle approximations hold for trigonometric functions.

To use sum-of-squares optimization requires that the system model be polynomial (or at least rational). The authors demonstrate that tire models can be well-approximated with a rational function consisting of a 3rd-order numerator and 4th-order denominator. The main results of [2] were replicated and then applied to the model parameters of [1]. The associated scripts contain the polynomial fit optimization procedure. The

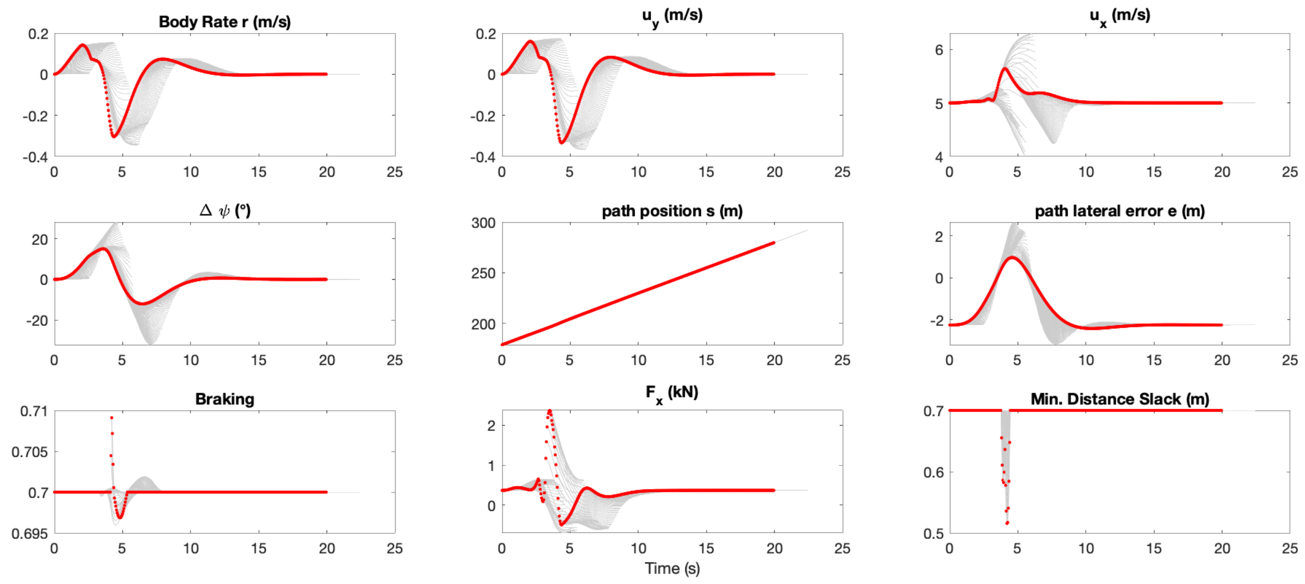

Figure 4: Obstacle Avoidance with Slow, Narrow Vehicle

objective was a standard least-squares objective of minimizing the mismatch for slip angles ranging from -0.3 to +0.3. The function and its rational approximation are shown in Figure 8. To ensure the denominator was strictly positive (to avoid divide-by-zero errors), we used a rational function of the following form:

$$f(\alpha) = \frac{c_1\alpha + c_3\alpha^3}{1 + c_2\alpha^2 + c_4\alpha^4} \tag{2}$$

In the interest of brevity we show only two results below. Figure 9 shows the phase-portrait for the lateral dynamics (body-y velocity vs. rotational velocity) when the vehicle is moving forward with body-x velocity 10 m/s and a steering angle of 0. The red lines show the initial conditions that converge to the equilibrium while the black are unstable (outside the ROA). Our objective is to obtain a Lyapunov function (and associated level set) that provides a conservative (inner approximation) of the ROA. We first linearize the system and solve a standard linear matrix inequality to obtain a candidate Lyapunov function $V(x) = x^T Px$. We then use sum-of-squares optimization to identify the region over which $\dot{V}(x) < 0$, that is the maximum level set of $V(x)$ for which $\dot{V}(x) < 0$. We then repeat the process, searching for a new Lyapunov function (and associated level set) that strictly contains the previous solution. This method is commonly referred to as V-S iteration as it consists of alternating between solving for the Lyapunov function $V(x)$ and the associated multipliers arising from using the S-procedure for set containment. For further details please see [2] and the references therein.

Figure 10 shows the ROA with the body-x velocity doubled (u = 20 m/s). The ROA is clearly smaller indicating the vehicle is more susceptible to spinning out of control upon being disturbed.

If we pursue higher-order Lyapunov functions, we typically can get better approximations of the ROA. Figure 11 and Figure 12 show the iterations of Lyapunov functions achieved when searching for a quartic function (via quadratic). It is clear that the maximum ROA estimates are larger than the quadratic case.

## 6 Conclusion

Obstacle avoidance is a crucial component of self-driving vehicle technology. For rapid, evasive maneuvering it is necessary to utilize models that account for nonlinear tire dynamics. This project attempted to replicate a recent algorithm that addresses this challenge. For tightly-constrained situations, the current code shows promise but experiences convergence issues. Lacking further documentation of the implementation details of [1] it is currently unclear where the problem lies. Future work may examine this issue in detail.

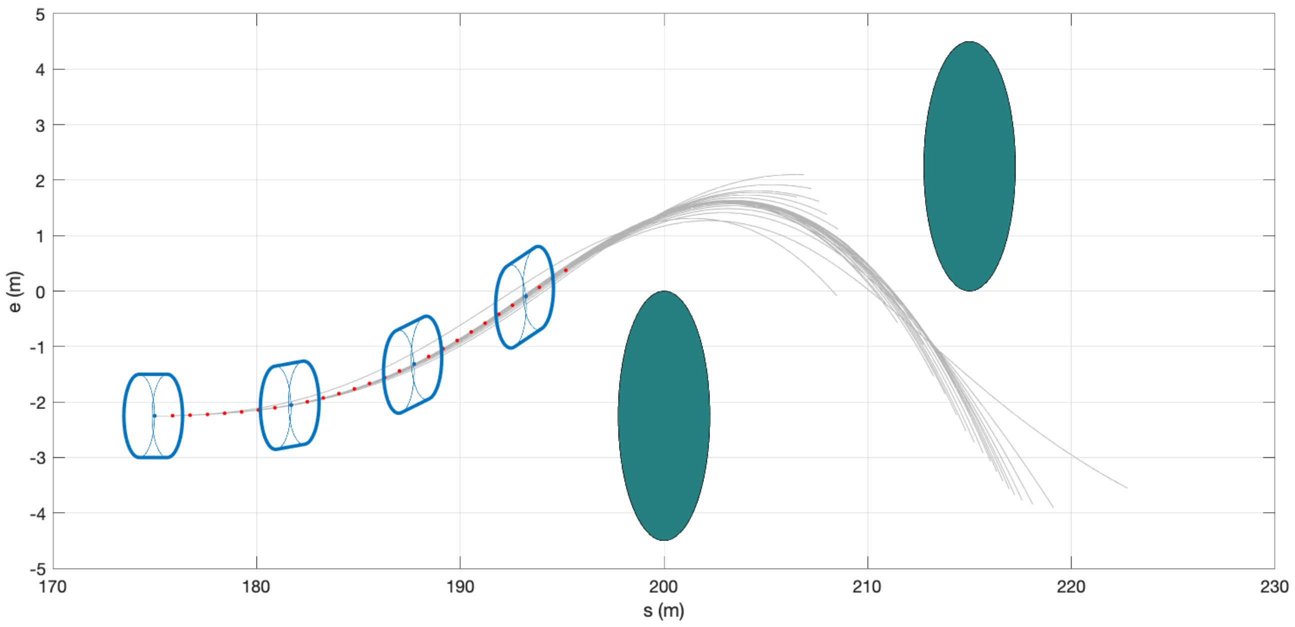

Figure 5: Obstacle Avoidance with Fast, Wide Vehicle

## References

[1] M. Brown and J. C. Gerdes. Coordinating tire forces to avoid obstacles using nonlinear model predictive control. *IEEE Transactions on Intelligent Vehicles*, 5(1):21–31, 2020.

[2] Mehdi Imani Masouleh and David J. N. Limebeer. Region of attraction analysis for nonlinear vehicle lateral dynamics using sum-of-squares programming. *Vehicle System Dynamics*, 56(7):1118–1138, 2018.

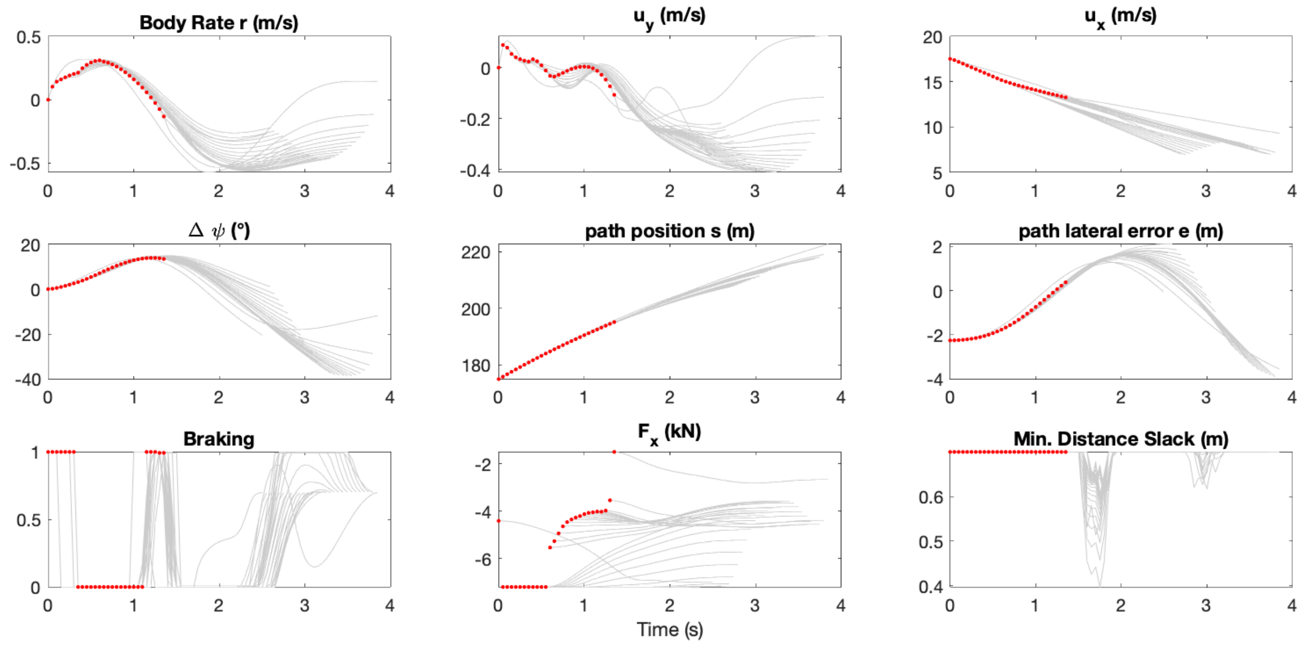

Figure 6: Obstacle Avoidance with Fast, Wide Vehicle

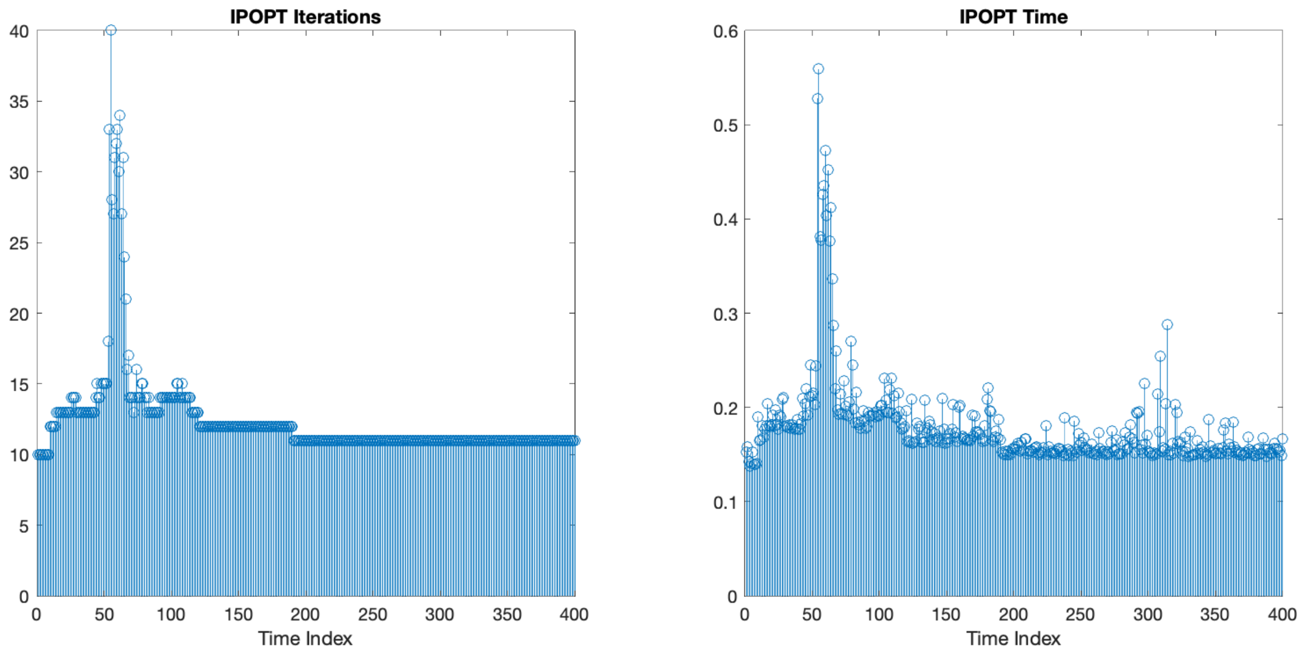

Figure 7: Solver (IPOPT) iterations and run time performance

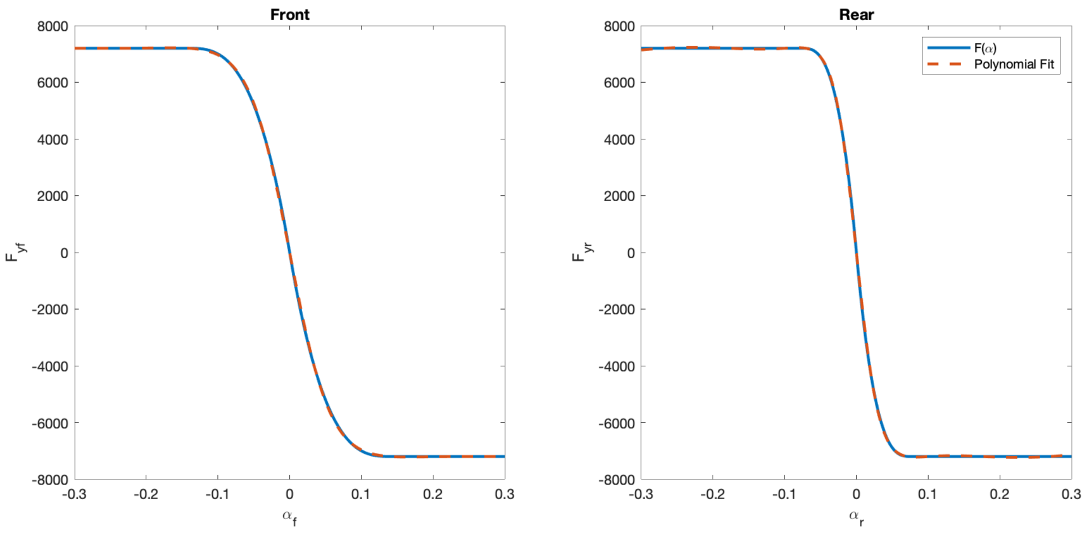

Figure 8: Rational Approximation of Lateral Forces Tire Model

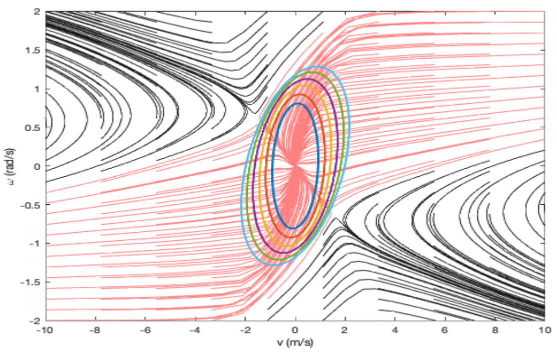

Figure 9: Quadratic ROA Estimate (u = 10 m/s)

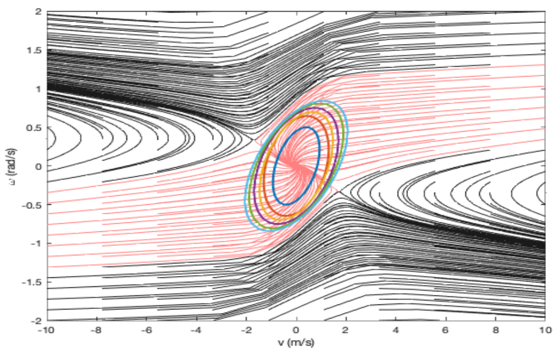

Figure 10: Quadratic ROA Estimate (u = 20 m/s)

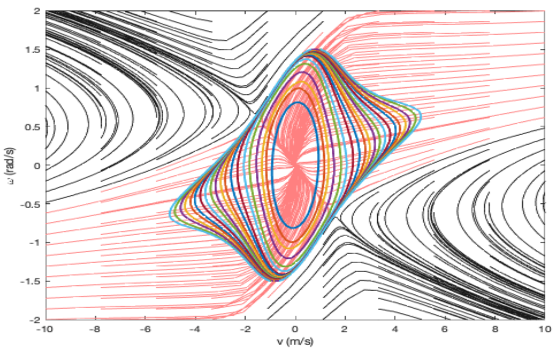

Figure 11: Quartic ROA Estimate (u = 10 m/s)

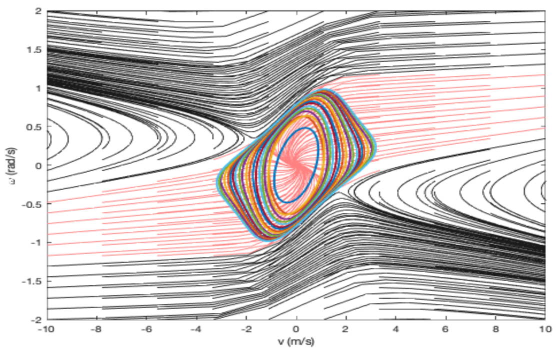

Figure 12: Quartic ROA Estimate (u = 20 m/s)