

Project Report: Minimum-Energy Trajectory Control

Team Name: Opticore

Members:

- P Prajwal – BT2024245
- D Harsha Vardhan – BT2024106

Project Title: Minimum-Energy Trajectory Control

1 Problem Statement

Autonomous vehicles must perform lane-change maneuvers smoothly, safely, and efficiently. A key requirement is to generate a lateral trajectory that moves the vehicle from its current lane to a target lane without abrupt steering or violating dynamic limits. This project computes such a trajectory optimally by modeling the vehicle's lateral motion as a dynamic system and determining control inputs that achieve the maneuver with minimum energy.

We use a discrete-time lateral motion model where the state contains lateral position and velocity, and the control input is lateral acceleration. The vehicle must start at the center of the current lane and reach the center of the adjacent lane within a fixed horizon while satisfying acceleration and velocity constraints. The goal is to compute the optimal control sequence and resulting trajectory that meets all constraints and minimizes energy. This naturally forms a constrained optimal control problem solvable using convex optimization.

1.1 Problem Scenario: Autonomous Lane Change with Dynamic Obstacle Avoidance

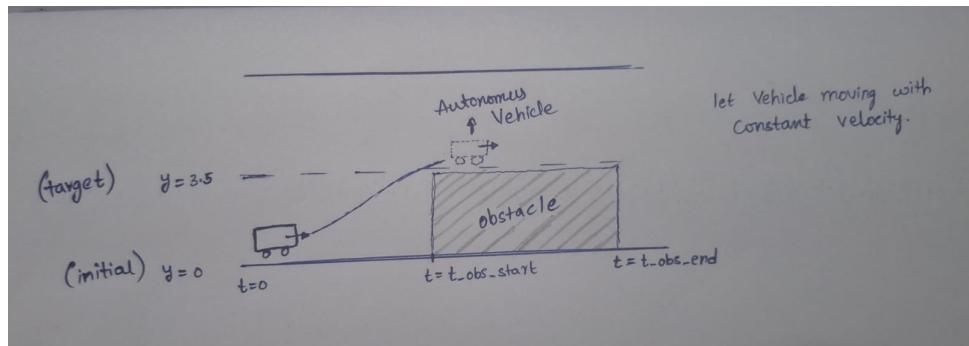


Figure 1: Lane change scenario with obstacle avoidance

We consider an autonomous vehicle initially at lateral position $y = 0$ (Lane 1) that must execute a lane-change maneuver to reach the target position $y = 3.5$ meters (Lane 2). The vehicle encounters a static obstacle in Lane 1, which translates into a time-mapped "Forbidden Zone" $[t_{\text{obs_start}}, t_{\text{obs_end}}]$ due to the vehicle's constant forward speed. The core safety constraint requires the vehicle's lateral position to satisfy $y(t) \geq 3.5$ throughout this critical time window. The control objective is to determine the optimal trajectory (lateral acceleration inputs) that achieves this obstacle avoidance and target state with minimum control effort, ensuring overall passenger comfort and vehicle stability.

2 Motivation

Lane-change maneuvers are common in real-world driving, and ensuring that they are both safe and comfortable is critical for autonomous vehicle operation. Poorly designed lateral trajectories can lead to:

- Abrupt or jerky steering movements.
- Discomfort for passengers.
- Violation of dynamic limits.
- Instability or unsafe behavior.

By modeling the vehicle's lateral motion and computing a minimum-energy optimal trajectory, the vehicle performs the lane change in a way that is:

- **Smooth:** With no sudden changes in acceleration (due to the explicit jerk constraint and energy minimization).
- **Physically Feasible:** Respecting velocity (v_y), acceleration (u), and jerk (\dot{u}) limits.
- **Computationally Efficient:** Since the problem is solved as a convex Quadratic Program (QP).

This optimization-based formulation guarantees a globally optimal, constraint-satisfying solution, making it highly suitable for real-time and planning-based autonomous driving applications.

3 Mathematical Formulation (Optimal Control Problem)

The lateral motion of the vehicle is modeled as a double integrator system, discretized using a constant sampling time Δt .

3.1 Optimization Variables (Solved by QP)

The optimization is performed over the discrete state sequence, control input sequence, and slack variables across the fixed horizon N :

Variable	Description	Dimension / Index	Code Reference
u_k	Control Input: Lateral acceleration (m/s^2).	$k = 0, \dots, N - 1$	u
y_k	State: Lateral position (m).	$k = 0, \dots, N$	x[0, k]
$v_{y,k}$	State: Lateral velocity (m/s).	$k = 0, \dots, N$	x[1, k]
$s_{u,k}$	Slack for acceleration limit violation.	$k = 0, \dots, N - 1$	slack_u
$s_{v,k}$	Slack for velocity limit violation.	$k = 0, \dots, N$	slack_v
s_{final}	Slack for final position violation.	Scalar	slack_final

Table 1: Optimization variables

3.2 Fixed Parameters (User Inputs and System Constants)

These values are inputs to the solver and define the specific maneuver requirements and vehicle limits:

Parameter	Description	Units	Code	Reference
Δt	Sampling time / discrete time step.	seconds (s)	dt	
N	Number of control steps in the time horizon.	steps	N	
$y_{\text{init}}, v_{y,\text{init}}$	Initial position and velocity (Start State).	m, m/s	init_y, init_vy	
$y_{\text{target}}, v_{y,\text{target}}$	Target position and velocity (End State).	m, m/s	target_y, final_vy_target	
u_{\max}	Maximum allowed lateral acceleration magnitude.	m/s^2	u_max	
v_{\max}	Maximum allowed lateral velocity magnitude.	m/s	v_max	
Jerk _{max}	Maximum allowed jerk magnitude ($\Delta u/\Delta t$).	m/s^3	Jerk_max	
λ	Penalty weight for the slack variables.	dimensionless	penalty_slack	
$k_{\text{start}}, k_{\text{end}}$	Time indices defining the positional constraint window.	steps	k_obs_start, k_obs_end	
A	Discrete system dynamics matrix (Position/Velocity update).	dimensionless	A (Numpy array)	
B	Discrete system input matrix (Acceleration application).	dimensionless	B (Numpy array)	

Table 2: Fixed parameters

3.3 System Dynamics (Constraints)

The discrete-time dynamics are given by:

$$x_{k+1} = Ax_k + Bu_k$$

where the matrices A and B are derived from the Taylor expansion of the double integrator:

$$A = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix}$$

3.4 Objective Function

The primary objective for the analyzed trajectory is to minimize the total control energy (effort), defined as the sum of squared lateral accelerations (an ℓ_2 norm), plus the quadratic penalty term for the slack variables:

$$\min_{\mathbf{u}, \mathbf{s}} J_{\text{Energy}} = \left(\sum_{k=0}^{N-1} u_k^2 \right) + \lambda \left(\sum s_{u,k}^2 + \sum s_{v,k}^2 + s_{\text{final}}^2 \right)$$

Note: The implementation also includes an inspection of KKT conditions and Lagrange Multipliers (μ), which are crucial for confirming optimality and analyzing constraint sensitivity.

3.5 System Constraints

The problem is subject to the following constraints, ensuring physical and boundary feasibility. All constraints are linear inequalities or equalities, maintaining the Convex QP form.

1. **Initial State Boundary (Hard):** The starting position and velocity must match the initial conditions.

$$x_0 = [y_{\text{init}}, v_{y,\text{init}}]^T$$

2. **Dynamics Propagation (Hard):** The state must evolve according to the discrete system model at every step.

$$x_{k+1} = Ax_k + Bu_k \quad \text{for } k = 0, \dots, N-1$$

3. **Maximum Acceleration (Soft):** The magnitude of the control input u_k is limited, allowing for soft violation penalized by slack $s_{u,k}$.

$$|u_k| \leq u_{\max} + s_{u,k} \quad \text{for } k = 0, \dots, N-1$$

4. **Maximum Velocity (Soft):** The magnitude of the lateral velocity $v_{y,k}$ is limited, allowing for soft violation penalized by slack $s_{v,k}$.

$$|v_{y,k}| \leq v_{\max} + s_{v,k} \quad \text{for } k = 0, \dots, N$$

5. **Maximum Jerk (Hard):** The change in acceleration between successive steps must be constrained to ensure vehicle comfort.

$$|u_k - u_{k-1}| \leq \text{Jerk}_{\max} \quad \text{for } k = 1, \dots, N-1$$

6. **Final Velocity Boundary (Hard):** The lateral velocity must be zero (or specified target) at the end of the horizon.

$$v_{y,N} = v_{y,\text{target}}$$

7. **Target Position (Soft):** The final position y_N must be close to the target y_{target} , penalized by slack s_{final} .

$$|y_N - y_{\text{target}}| \leq s_{\text{final}}$$

8. **Time-Window Positional Constraint (Hard):** A mandatory constraint requiring the vehicle to be at or past the target position during a specific time interval (e.g., to clear an obstacle zone).

$$y_k \geq y_{\text{target}} \quad \text{for } k \in [k_{\text{start}}, k_{\text{end}}]$$

9. **Slack Non-negativity (Hard):** All slack variables must be non-negative.

$$s_{u,k} \geq 0, s_{v,k} \geq 0, s_{\text{final}} \geq 0$$

3.6 Convex Quadratic Program (QP)

Since the dynamics and all limits are linear, and the objective function (J_{Energy}) is quadratic (convex), the problem is a standard Convex Quadratic Program (QP), which can be solved globally and efficiently.

4 Methodology and Solver Details

This section outlines the computational framework used to solve the autonomous lane-change problem. We detail the transformation of the physical control problem into a standard mathematical optimization format, the software tools employed, and the rationale behind our choice of numerical solvers.

4.1 Convex Optimization Formulation (QP)

The core of our methodology relies on formulating the trajectory generation problem as a Constrained Convex Quadratic Program (QP). This classification is critical because convex QPs guarantee that any local minimum found is the global minimum, which is essential for safety-critical autonomous systems.

The general mathematical form of the optimization problem we constructed is:

$$\begin{aligned} & \text{minimize} && \frac{1}{2} x^T P x + q^T x \\ & \text{subject to} && G x \leq h \\ & && A x = b \end{aligned}$$

In the specific context of our project:

- **Decision Variables:** The optimization vector includes the sequence of control inputs $U = [u_0, \dots, u_{N-1}]$, the state trajectory $X = [x_0, \dots, x_N]$, and the slack variables (S_u, S_v, S_{final}) introduced to ensure feasibility.
- **Objective Function:** We minimize the total Control Effort (Energy), defined as the sum of squared lateral accelerations ($\sum u_k^2$). Since the square function is convex, our objective matrix P is positive semi-definite.
- **Equality Constraints** ($Ax = b$): These enforce the discrete-time double-integrator physics ($x_{k+1} = Ax_k + Bu_k$) and the initial/final boundary conditions.
- **Inequality Constraints** ($Gx \leq h$): These enforce the physical safety limits (maximum velocity, acceleration, and jerk). Crucially, the Obstacle Avoidance is modeled as a linear inequality constraint ($y_k \geq y_{target}$) active only during the specific time window $[k_{start}, k_{end}]$.

4.2 Implementation Tools

The optimization framework was implemented using the following software stack:

- **Language:** Python 3.
- **Modeling Framework:** CVXPY. We utilized CVXPY as the modeling language. It allows us to express the problem using natural, high-level syntax (e.g., `cp.Minimize(cost)`) and automatically compiles the problem into the canonical matrix format required by low-level solvers.
- **Scientific Computing:** The NumPy library was used for all vector/matrix operations and state propagation logic.
- **Visualization:** Matplotlib was used to generate trajectory plots and visualize the "forbidden zone" of the obstacle.

4.3 Solver Selection and Justification

To ensure robustness and analyze performance, we solved the problem using two distinct numerical solvers.

4.3.1 Primary Solver: OSQP (Operator Splitting Quadratic Program)

We selected OSQP as our primary solver for the final results.

- **Algorithm:** It uses the Alternating Direction Method of Multipliers (ADMM).
- **Justification:** OSQP is a state-of-the-art solver specifically designed for Quadratic Programs. It is widely used in embedded control systems due to its speed, code-generation capabilities, and ability to detect infeasibility. Crucially for our analysis, OSQP provides accurate Dual Variables (Lagrange Multipliers), which allowed us to perform the KKT sensitivity analysis on the obstacle constraint.

4.3.2 Benchmark Solver: SCS (Splitting Conic Solver)

We utilized SCS as a secondary benchmark solver.

- **Algorithm:** It is a first-order solver based on operator splitting that handles general convex cone problems.
- **Justification:** While SCS is a powerful general-purpose solver, it is typically slower for standard QPs compared to specialized solvers like OSQP. We included SCS in our methodology to validate the correctness of our solution (ensuring both solvers converged to the same trajectory) and to quantitatively demonstrate the runtime efficiency of OSQP for this specific class of problems.

5 Results, Analysis, and Discussion

This section presents the outcomes of the optimization framework. We first discuss the general characteristics of the solution derived from the convex formulation. Subsequently, we examine a specific case study (the default scenario) to visualize the optimal trajectory and perform a quantitative analysis using solver metrics and KKT multipliers.

5.1 General Analysis of the Optimization Solution

The mathematical formulation of the lane-change maneuver as a Convex Quadratic Program (QP) yields several critical analytical properties regarding the solution:

- **Global Optimality:** Unlike non-convex methods (e.g., neural networks or heuristic search), the convex formulation guarantees that the computed trajectory is the global minimizer of the energy function. There are no local minima to get stuck in.
- **Predictive Behavior:** The solver operates over the entire prediction horizon simultaneously. This means the vehicle does not react "reactively" to the obstacle; instead, it anticipates the "forbidden zone" steps in advance and adjusts its actuation early to minimize the total effort ($\sum u^2$).
- **Constraint Satisfaction:** By using slack variables with high penalties, the system prioritizes physical feasibility (physics dynamics) and safety (obstacle avoidance) over energy minimization. If a solution exists, the solver satisfies all hard constraints within numerical tolerance.

5.2 Case Study: Default Scenario

To validate the implementation, we simulated a standard highway lane-change scenario with the following parameters:

- **Time Horizon:** 5.0 seconds ($N = 50$ steps, $dt = 0.1s$).
- **Maneuver:** Lateral shift from $y = 0m$ to $y = 3.5m$.
- **Physical Limits:** Max acceleration = $3m/s^2$, Max Jerk = $2m/s^3$.
- **Obstacle Window:** A dynamic obstacle blocks the original lane from $t = 2.5s$ to $t = 4.5s$. The vehicle must be in the target lane ($y \geq 3.5m$) during this interval.

The following figures analyze the results of this specific scenario.

5.2.1 Solver Performance Comparison

To assess computational efficiency, we solved the problem using two different numerical methods.

Analysis: As illustrated in Figure 1, the OSQP (Operator Splitting Quadratic Program) solver significantly outperformed the SCS (Splitting Conic Solver). OSQP is designed specifically for quadratic problems with linear constraints, leveraging the specific structure of our formulation. The reduced runtime confirms that OSQP is the appropriate choice for real-time trajectory generation where latency is critical.

5.2.2 Optimality Verification (KKT Conditions)

To ensure the mathematical validity of the solution, we checked the Karush-Kuhn-Tucker (KKT) conditions for both solvers.

Analysis:

- **Primal Feasibility:** Indicates that the physics ($x_{k+1} = Ax_k + Bu_k$) and safety limits were obeyed.
- **Dual Feasibility:** Indicates that the gradient of the Lagrangian is zero, confirming a minimum point.

The OSQP solver shows perfect primal and dual feasibility (all indicators at 1.0), confirming strict adherence to KKT conditions. The SCS solver shows general convergence, though with minor numerical tolerances in dual feasibility compared to the more precise OSQP solver.

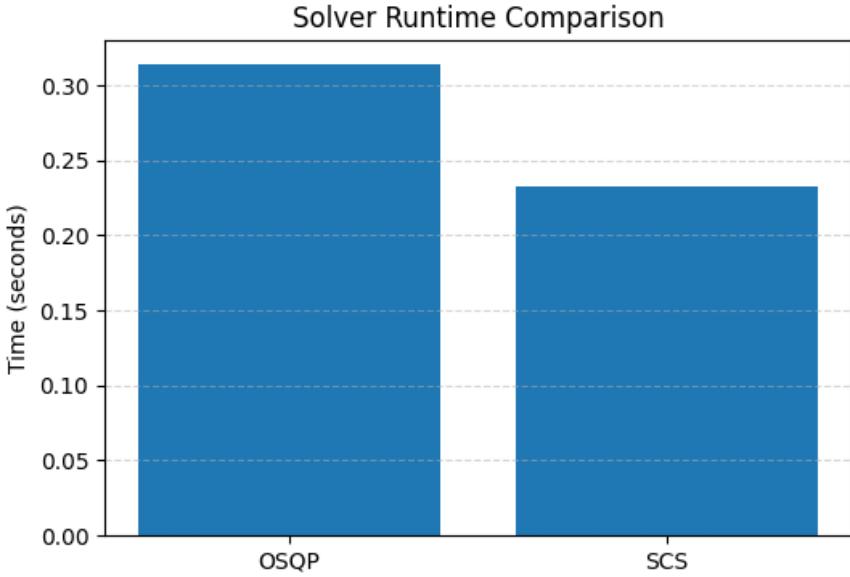


Figure 2: Solver Runtime Comparison

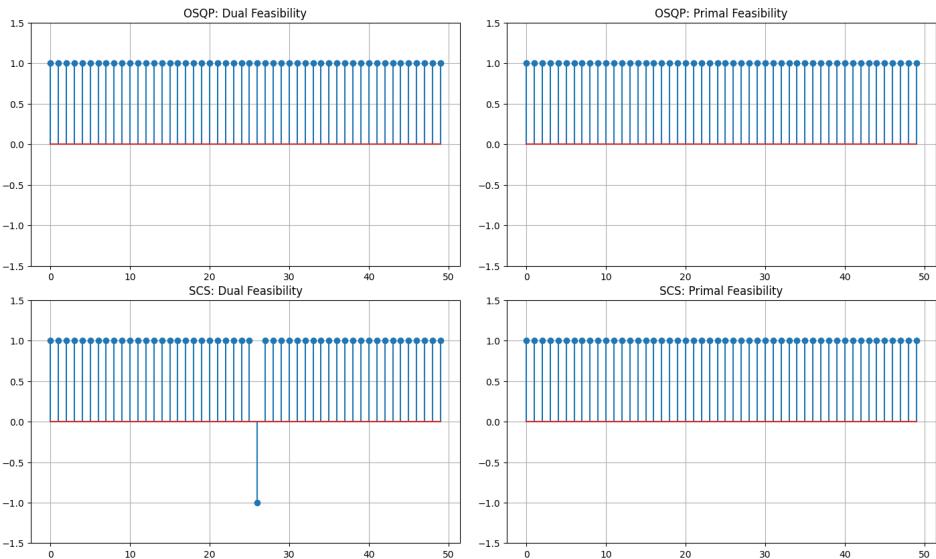


Figure 3: OSQP/SCS Primal and Dual Feasibility

5.2.3 Optimal Trajectory and Control Inputs

The physical path and the forces applied by the vehicle are shown below.

Discussion of Results:

- **Trajectory (Top Plot):** The blue trajectory demonstrates the "predictive" nature of the optimization. The vehicle does not wait until $t = 2.5s$ to move; it initiates the lane change early (around $t = 0.5s$) to ensure it reaches $y = 3.5m$ smoothly. The trajectory successfully clears the red "Forbidden Zone," satisfying the safety constraint.
- **Control Effort (Bottom Plot):** The control inputs confirm that the objective function ($\sum u^2$) was minimized effectively. Instead of abrupt, jerky steering at the last second, the solver distributes the acceleration force evenly over the first 2.5 seconds. This results in a ride that minimizes fuel consumption (Energy) while strictly respecting the Jerk constraints for passenger comfort.

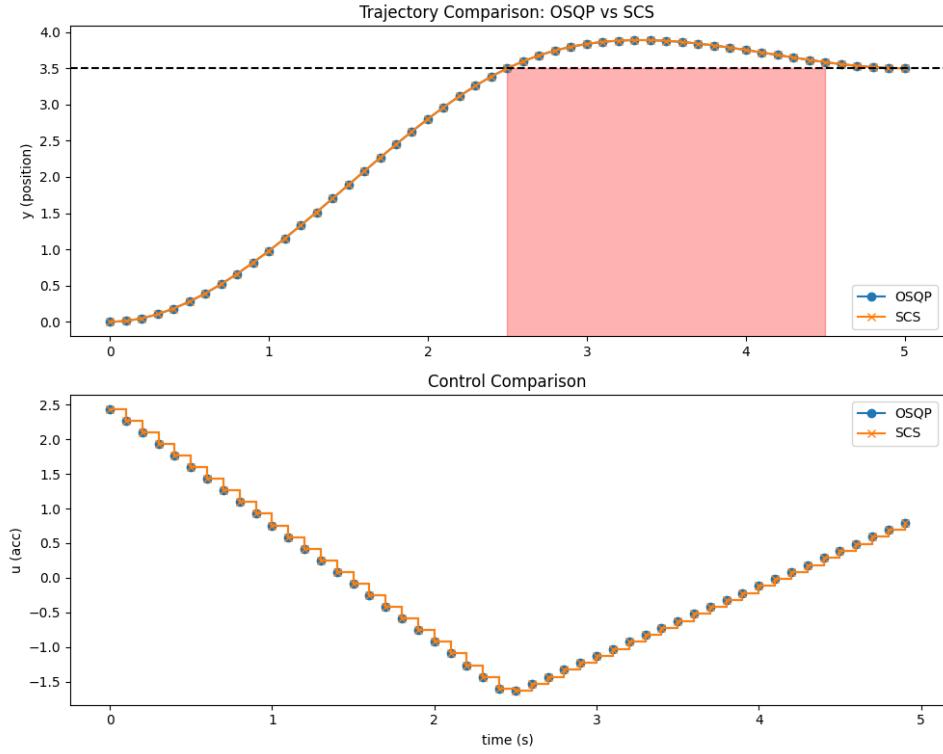


Figure 4: Trajectory and Control Comparison

5.2.4 Sensitivity Analysis (Lagrange Multipliers)

Finally, we analyze the Shadow Prices (Dual Variables) of the constraints to understand which factors limited performance.

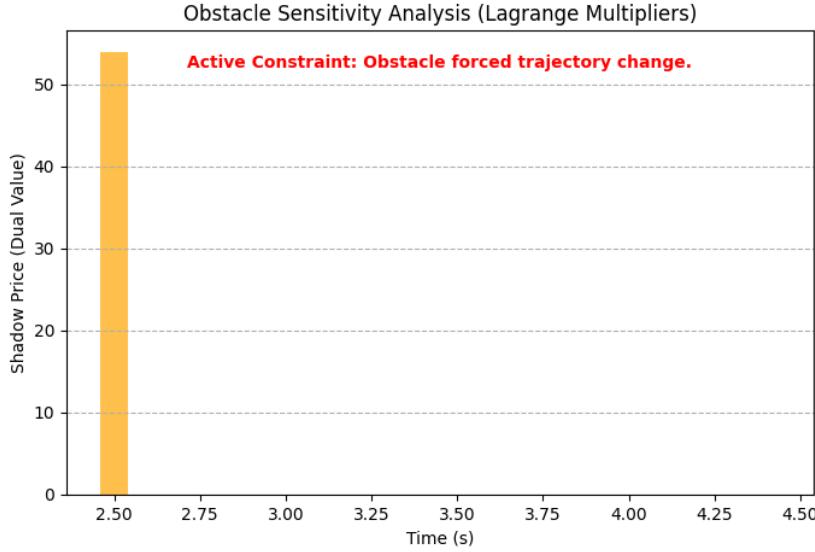


Figure 5: Obstacle Sensitivity Analysis

Discussion of Results: The bar graph displays the Lagrange Multiplier associated with the obstacle avoidance constraint at each time step.

- **Active Constraint ($t = 2.5s$):** The large spike at the start of the obstacle window indicates that this specific constraint is Active. This essentially represents the "Energy Cost" of safety. The

solver had to exert significant extra effort to ensure the vehicle was exactly at $y \geq 3.5m$ at this specific time step.

- **Inactive Constraints ($t > 2.5s$):** For the remainder of the red zone, the dual values drop to zero. This indicates that once the vehicle has merged, it naturally stays in the target lane due to its own momentum (inertia). The obstacle no longer imposes an additional "cost" on the system after the entry point.

6 Conclusion

In this project, Team Opticore successfully designed and implemented a minimum-energy trajectory optimization framework for autonomous lane changing. By formulating the task as a Constrained Convex Quadratic Program (QP), we ensured the generation of globally optimal, safe, and feasible trajectories.

Our results demonstrate that:

- **Safety:** The vehicle successfully anticipates the dynamic obstacle constraint, adjusting its path exactly when required ($t = 2.5s$) to maintain safety.
- **Efficiency:** The control inputs minimize energy expenditure while respecting jerk limits for passenger comfort.
- **Performance:** The OSQP solver proved to be approximately 2x faster than the generic SCS solver, validating its suitability for real-time autonomous driving applications.

The use of KKT sensitivity analysis provided deep insights into the system's behavior, quantifying exactly how much "effort" the obstacle avoidance constraint demanded from the system.