2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Reference-Free, Long-Horizon Trajectory Optimization for Aggressive Autonomous Driving in Milliseconds

Prayag Sharma[1], Jonathan Y.M. Goh[2], and Franck Djeumou[1]

***Abstract*— Autonomous vehicles must generate long-horizon and dynamically feasible trajectories in real time—even when operating at the limits of vehicle handling—to ensure safe operation in adverse conditions. However, existing work rarely quantifies the computational demands of generating such trajectories without prior references, warm starts and often defaults to low-fidelity models, compromising accuracy and control authority. We investigate the modeling and solver design choices that enable real-time solution of long-horizon, reference-free optimal control problems (OCPs) using full vehicle dynamics. To this end, we analyze vehicle stiffness properties to justify the OCP's integration scheme and show that lower-order A-stable methods consistently outperform alternatives, with solve time differences reaching two orders of magnitude. We show that robust nonlinear solver performance hinges on understanding barrier parameter update strategies and safeguarding techniques for Hessian indefiniteness, inherent in some interior point methods. Lastly, we propose a computationally efficient method for generating initial guesses using dynamic equilibrium, unlocking real-time performance and reducing initial infeasibility by up to four orders of magnitude. Extensive benchmarking and high-fidelity BeamNG simulation demonstrate compute times as low as 55 ms over a 260 m horizon, including high-speed obstacle avoidance scenarios where drifting emerges as a necessary component of feasible trajectory generation.**

I. INTRODUCTION

To ensure safety in all critical situations, autonomous vehicles must be engineered to exploit the absolute limits of their dynamic capabilities, pushing beyond the boundaries of normal vehicle operation. Existing work has established that intentionally pushing a vehicle beyond traditional operational limits not only expands the safety envelope but can often be the only feasible evasive maneuver [1]–[3]. Such maneuvers require long-horizon and dynamically feasible planning for linking immediate actions to their downstream consequences in unforeseen situations. Unlocking this capability requires a paradigm shift from conventional short-horizon reference tracking to solving a *reference-free, long-horizon* optimal control problem (OCP) in real time (5-10 Hz). This paper presents a systematic investigation into the fundamental design choices required to construct such a framework, thereby expanding the operational envelope of autonomous vehicles.

Prior approaches underscore the complexity of solving the full nonlinear OCPs from scratch, identifying the underlying problems as highly sensitive to initial guesses and too computationally intensive for real-time deployment [1], [3]–[6]. This has led to a shared strategy across both high-performance racing [7]–[9] and autonomous drifting [10]–[12]: decoupling the problem into offline reference generation and online tracking. Even though they are successful in handling even pop-up obstacles [7], [8], the core dependency of such approaches to an offline-generated reference precludes their adaptation to a dynamically changing course. Recent attempts toward real-time generation include a task-specific OCP to transition between drift equilibria [13], and reinforcement learning (RL) to generate drift trajectories [14], [15]. However, the former remains highly task-specific, while RL's reliance on training data makes its reliability in unfamiliar scenarios difficult to guarantee.

Fig. 1: Tracking of high-speed emergency collision avoidance trajectories in BeamNG simulation. Video available at https://tinyurl.com/trajopt

To enable online replanning in safety-critical scenarios, many existing approaches reduce model fidelity by solving online long-horizon Optimal Control Problems using simplified point-mass dynamics [6]. Although computationally tractable, these abstractions often produce trajectories that violate dynamic feasibility [5]. Alternatively, systems based on pre-computed motion primitives [4], [5] constrain the vehicle to a fixed library of maneuvers, limiting expressiveness and proving brittle in scenarios not anticipated during design. Both workarounds trade off solving the underlying OCPs for solutions that may either result in dynamic infeasibility or may fail to exploit the full vehicle capabilities.

Although existing work excels at reference tracking, the foundational challenge of real-time reference generation from scratch remains largely unaddressed, favoring offline or task-specific solutions. As a consequence, critical design choices that directly impact real-time performance, such as an accurate and efficient numerical integrator, the nonlinear solver strategy, and a strategy for a generalizable initial guess, are rarely examined. This paper addresses this gap by

*This work was funded by Toyota Research Institute (TRI)

[1]P. Sharma and F. Djeumou are with MANE Dept., Rensselaer Polytechnic Institute, Troy, NY, USA sharmp6 and djeumf2@rpi.edu

[2] J. Goh is with Toyota Research Institute, Los Altos, CA, USA jon.goh@tri.global

presenting a systematic design of a trajectory optimization framework for solving the full nonlinear, reference-free OCP in real time. Our contributions are as follows:

• Through extensive benchmarking, we show that, while interior point methods [16], [17] outperform sequential quadratic programming (SQP) approaches [18] for trajectory generation, commonly used solvers like IPOPT [16] are unsuitable for real-time use. Our findings reveal that robust and real-time performance emerges from carefully selected barrier update and numerical ill-conditioning handling strategies found in solvers like KNITRO [17].

• We analyze the stiffness properties of vehicle dynamics and establish a principled basis for selecting stable and computationally efficient integration schemes. Further, our evaluation of nine numerical integrators reveals computational performance differences of up to two orders of magnitude. Critically, we demonstrate that due to system stiffness, lower-order A-stable methods surprisingly outperform higher-order, non-A-stable methods in terms of accuracy.

• We propose a strategy to generate a high-quality, reusable initial guess for the OCP by solving a single-node, box-constrained least-square problem formulation. Such a guess is computationally cheap to obtain, is by construction dynamically feasible, reduces total initial feasibility error for the optimizer by three to four orders of magnitude compared to an ill-informed guess, while enabling real-time performance. We then show the robustness of this guess by solving OCPs spanning both time-optimal racing and collision avoidance.

• We validate our approach on a critical collision avoidance scenario, where drifting around the obstacles is the only feasible solution. Our framework generates a drifting trajectory at highway speeds (90 km/hr) in real time (72 ms), while constraining emerging drift behavior renders the problem infeasible. To confirm real-world viability, we demonstrate successful tracking of trajectories for both racing and obstacle avoidance in a high-fidelity BeamNG [19] simulation.

II. Problem Formulation

A. Vehicle Dynamics

We use a single-track vehicle model in curvilinear coordinates [10], considering the wheel speed and load transfer dynamics in a curvilinear coordinate system [12]. The vehicle's position is described by its arc length, s, and lateral error, e, relative to a reference path with curvature $k_{\text{ref}}(s)$. The equations of motion are given by:

$$\frac{d}{ds}\begin{bmatrix} r \\ V \\ \beta \\ V_{\omega r} \\ \Delta F_z \\ e \\ \Delta\psi \\ s \\ t \end{bmatrix} = \frac{1}{\dot{s}}\begin{bmatrix} (a(F_{xf}\sin\delta + F_{yf}\cos\delta) - bF_{yr})/I_z \\ (F_{xf}\cos(\delta-\beta) - F_{yf}\sin(\delta-\beta) + \\ F_{xr}\cos\beta + F_{yr}\sin\beta)/m \\ -r + (F_{xf}\sin(\delta-\beta) + F_{yf}\cos(\delta-\beta) \\ -F_{xr}\sin\beta + F_{yr}\cos\beta)/(mV) \\ R_w(T_{\text{comb}} - R_w F_{xr})/I_w \\ -c_L(\Delta F_z - h_{cg}F_{x,net}/L) \\ V\sin(\Delta\psi) \\ \dot{\beta} + r - k_{\text{ref}}(s)\dot{s} \\ 1 \\ 1 - k_{\text{ref}}(s)e/(V\cos(\Delta\psi)) \end{bmatrix} \tag{1}$$

where the state $\mathbf{x} = [r, V, \beta, V_{\omega r}, \Delta F_z, e, \Delta\psi, s, t]^T$, the control $\mathbf{u} = [\delta, T_{\text{comb}}, T_{\text{bf}}]^T$, and the independent variable s is such that $\dot{s} = V\cos(\Delta\psi)/(1 - k_{\text{ref}}(s)e)$. Here, $\Delta\psi$ is the velocity vector orientation error w.r.t. path, r, V, β are the yaw rate, vehicle speed, and side slip angle respectively, $V_{\omega r}$ is the rear wheel longitudinal speed, ΔF_z is the dynamic load transfer (front to rear), $F_{x,net} = F_{xr} + F_{xf}\cos\delta - F_{yf}\sin\delta$ is the net longitudinal force, and δ is steering angle. We combine the engine torque $T_{\text{eng}} \geq 0$ and rear brake torque $T_{\text{br}} \leq 0$ to T_{comb} such that $T_{\text{eng}} = \max(0, T_{\text{comb}})$ and $T_{\text{br}} = \min(0, T_{\text{comb}})$. As for the model parameters, m is the vehicle mass, I_z the inertia, a, b, L, h_{cg} geometric parameters of the vehicle (front and rear distances from the center of mass, wheelbase, CG height), and R_w, I_w are the wheel radius and rear wheel inertia, respectively.

To model tire forces $(F_{xf}, F_{yf}, F_{xr}, F_{yr})$, we use the isotropic coupled slip brush Fiala model [10], [12]

$$F_{yf} = -F_{\text{total},f}\tan(\alpha_f)/\sigma_f, \qquad F_{xf} = r_{\text{w}}T_{\text{br}},$$
$$F_{yr} = -F_{\text{total},r}\tan(\alpha_r)/\sigma_r, \qquad F_{xr} = F_{\text{total},r}\kappa_r/\sigma_r.$$

The magnitude of the tire forces $(F_{\text{total},f}, F_{\text{total},r})$ are

$$F_{\text{total}} = \begin{cases} C\sigma - \frac{C^2\sigma^2}{3F_{\text{max}}} + \frac{C^3\sigma^3}{27(F_{\text{max}})^2} & \text{if } |\sigma| < \sigma_{\text{sl}} \\ F_{\text{max}} & \text{if } |\sigma| \geq \sigma_{\text{sl}} \end{cases},$$

where (σ_f, σ_r) are the total tire slips, and $(\sigma_{\text{sl},f}, \sigma_{\text{sl},r})$ are the total slips as the tires begin fully sliding

$$\sigma = \sqrt{\tan(\alpha)^2 + \kappa^2}, \quad \sigma_{\text{slip}} = \tan^{-1}(3\mu F_z/C).$$

The tire loads (F_{zf}, F_{zr}) depend on the static tire loads $(F_{\text{nom},zf}, F_{\text{nom},zr})$ as $F_{zf} = F_{\text{nom},zf} - \Delta F_z$ and $F_{zr} = F_{\text{nom},zr} + \Delta F_z$. The maximal tire forces F_{max} are

$$F_{\text{max},f} = \sqrt{(\mu F_{zf})^2 - (r_{\text{w}}T_{\text{br}})^2}. \quad F_{\text{max},r} = \mu F_{zr},$$

The slip angles (α_f, α_r) and slip ratios (κ_f, κ_r) are

$$\tan(\alpha_f + \delta) = (V\sin\beta + ar)/(V\cos\beta),$$
$$\tan(\alpha_r) = \frac{V\sin\beta - br}{V\cos\beta}, \quad \kappa_{r,f} = \frac{V_{\omega_{r,f}} - V\cos\beta}{V\cos\beta}.$$

B. Minimum Time Discrete Optimal Control problem

We employ direct multiple shooting with both states and control as the decision variables of our optimization problem. The state and control variables are discretized for a constant $\Delta s_k := s_{k+1} - s_k$. $\mathcal{F}$ is defined as a general map $\mathcal{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_x}$ that approximates the next state x_{k+1}. Defining the lower and upper bounds on control $\underline{u}, \overline{u}$ and its rate $\underline{\dot{u}}, \overline{\dot{u}}$, the OCP is formulated as (1). We define a common objective of minimum time for both racing and collision avoidance studies as it serves well for both. Except for the variables fixed at boundary conditions(e.g., s, t, e), the above problem is a free initial $\mathbf{x}_0$ and final state $\mathbf{x}_f$ problem, which helps maintain a general structure. In addition to the track boundaries, we impose bounds on the vehicle's velocity, as this was empirically found to accelerate solver convergence. Furthermore, vehicle-specific control bounds and slew rate constraints are also enforced. The tire saturation

constraints prevent the wheels from going beyond their slip limits for racing and can be relaxed for collision avoidance scenarios to allow full dynamic range. A small constant ϵ is introduced in the tire saturation constraints to prevent infeasibility from brief, instantaneous saturation.

$$
\begin{aligned}
&\min_{\{x_k\}_{k=0}^{N},\{u_k\}_{k=0}^{N-1}} \quad t_N \\
&\text{s.t.} \quad x_{k+1} = \mathcal{F}(x_{k+1}, x_k, u_k; \Delta s_k), \quad k = 0, \dots, N-1, \\
&\text{(initial)} \quad \begin{cases} s_0 = 0,\ t_0 = 0,\ \Delta\psi_0 = 0, \\ e_{\min} \le e_0 \le e_{\max} \end{cases} \\
&\text{(track/path bounds)} \quad \begin{cases} e_{\min} \le e_k \le e_{\max}, \\ V_{\min} \le V_k \le V_{\max} \end{cases} \quad k = 0, \dots, N \\
&\text{(control bounds)} \quad \{\underline{u} \ \le \ u_k \ \le \ \overline{u} \qquad k = 0, \dots, N-1 \\
&\text{(control rates)} \quad \begin{cases} \dot{\underline{u}} \ \le \ \dfrac{u_k - u_{k-1}}{t_k - t_{k-1}} \ \le \ \dot{\overline{u}} \end{cases} \qquad k = 1, \dots, N-1 \\
&\text{(tire saturation)} \quad \begin{cases} \sigma_{f,sl} - |\sigma_{f_k}| < \epsilon, \\ \sigma_{r,sl} - |\sigma_{r_k}| < \epsilon \end{cases} \quad k = 0, \dots, N-1 \\
&\text{(terminal)} \quad \{s_N = N\Delta s_k,\ e_{\min} \le e_N \le e_{\max}
\end{aligned}
\tag{2}
$$

III. Interior Point Methods for Large Scale NLP

Although both sequential quadratic programming (SQP) and interior-point (IP) methods solve large-scale non-linear programming (NLP) problems [20] like (2), recent benchmarks show the superior performance of IP solvers such as `IPOPT` [16] and `KNITRO` [17] over SQP alternatives like `SNOPT` [21], which lies in-line with our findings in Sec. VI. However, the robust performance of IP methods is contingent upon several key factors: the strategy for handling non-convexity (i.e., an indefinite Hessian), the barrier parameter update rule, and the choice of globalization technique to ensure convergence [20]. Therefore, we present a brief overview of Interior-Point (IP) theory to accompany our discussion in Sec. VI on the various solver properties that help improve performance.

The optimization problem in (2) can be re-written as (3), where $\eta \in \mathbb{R}^{N \times n_x + n_u \times (N-1)}$ is the new stacked decision variable and c is the cost function. All equality constraints are stacked into c_E, representing general non-linear equality constraints. Similarly, inequality constraints (track/path bounds), (control, control rate bounds), (tire saturation) can be stacked into $c_I \in \mathbb{R}^{I_m}$ representing general nonlinear inequality constraints. Slack variables $s_i \in \mathbb{R}^{I_m}$, $s_i > 0$ and a barrier parameter μ are introduced. An IP algorithm solves a set of approximate barrier problems for a sequence of positive barrier parameters μ_k that converge to zero [17].

$$
\begin{aligned}
\min_{\eta, s} \quad & c(\eta) \ - \ \mu \sum\nolimits_{i=1}^{I_m} \log s_i, \\
\text{s.t.} \quad & c_E(\eta) = 0, \quad c_I(\eta) - s = 0
\end{aligned}
\tag{3}
$$

we write the KKT conditions [20] for problem (3) as:

$$
\begin{aligned}
\nabla c(\eta) - A_E^T(\eta) y - A_I^T(\eta) z = 0, \quad & -\mu e + Sz = 0, \\
c_E(\eta) = 0, \quad & c_I(\eta) - s = 0,
\end{aligned}
$$

where y and z are vectors of Lagrange multipliers, $e = (1, ..., 1)^T$, $S = \text{diag}(s_1, ..., s_m)$, A_E and A_I are the Jacobian matrices of constraints $c_E(\eta)$ and $c_I(\eta)$, respectively. Defining the Lagrangian $\mathcal{L}$, the merit function for measuring progress towards a feasible and optimal solution ϕ, and the corresponding Newton step to update the primal and dual variables η, s, y, z as follows:

$$
\mathcal{L}(\eta, s, y, z) = c(\eta) - y^T c_E(\eta) - z^T (c_I(\eta) - s), \tag{4}
$$

$$
\phi_\nu(\eta, s) = c(\eta) - \mu \sum_{i=1}^{m} \log s_i + \nu \|c_E(\eta)\|_2 + \nu \|c_I(\eta) - s\|_2, \tag{5}
$$

$$
\begin{bmatrix} \nabla^2_{\eta\eta}\mathcal{L} & 0 & -A_E^T(\eta) & -A_I^T(\eta) \\ 0 & Z & 0 & S \\ A_E(\eta) & 0 & 0 & 0 \\ A_I(\eta) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_\eta \\ d_s \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} A_E^T(\eta) y - \\ \nabla f(\eta) + \\ A_I^T(\eta) z \\ \mu e - Sz \\ -c_E(\eta) \\ s - c_I(\eta) \end{bmatrix} \tag{6}
$$

where $\nu > 0$. If the matrix in eq (6) is well defined in terms of inertia, then the step d is a descent direction for the merit function ϕ. Through backtracking line search [17], [20], step-length α can be computed, and the decision variables at the next iterate η^+, s^+, y^+, z^+ can be obtained using:

$$
\eta^+ = \eta + \alpha_s d_\eta, \qquad s^+ = s + \alpha_s d_s, \tag{7}
$$

$$
y^+ = y + \alpha_z d_y, \qquad z^+ = z + \alpha_z d_z. \tag{8}
$$

Our analysis in the results Sec. (VI) demonstrate how the treatment of bad inertia or indefiniteness of the Hessian matrix $\nabla^2_{\eta\eta}\mathcal{L}$ in the Newton step (6) and the barrier parameter (μ) update strategy, dictate solver performance.

IV. Stiffness and Integrator Study

Implicit and explicit integration techniques are known to perform well for stiff and non-stiff ODEs, respectively [22], [23]. To inform our choice of integrator, we first calculate the stiffness ratio at each node along a converged time-optimal trajectory for an oval track. Given the dynamics map f in (1), i.e. $\dot{x}(s) = f(x(s), u(s))$, the stiffness ratio SR can be defined as the ratio of the magnitude of the fastest-decaying mode (largest real $\Re$ negative eigenvalue λ_j magnitude) to the slowest-decaying mode (smallest real negative eigenvalue magnitude) of the Jacobian J_k of f:

$$
\delta\dot{x} = J_k\, \delta x, \quad J_k = \left.\frac{\partial f}{\partial x}\right|_{(x_k, u_k)}, k = 0, \dots, N \tag{9}
$$

$$
J_k v_j = \lambda_j v_j,\ j = 1, \dots, n_x, \quad \mathcal{S} = \{\, j : \Re\lambda_j < 0 \,\}, \tag{10}
$$

$$
r_j = |\Re\lambda_j|,\ j \in \mathcal{S}, \text{SR} = \frac{\max_{j\in\mathcal{S}} r_j}{\min_{j\in\mathcal{S}} r_j},\ if\ \min_j r_j > 0.
$$

We calculate a *participation matrix* using the right and left eigenvalues for the Jacobians J_k and backtrack the contribution of each state at every node in the stiffness ratio.

The results in Figure 2 show the system is conditionally stiff, with the SR peaking above 4500 during aggressive maneuvers, far exceeding the traditional stiff threshold of 10^3. This stiffness originates primarily from the fast dynamics of

the sideslip angle (β), which is consistent with the vehicle's physical behavior during high-speed cornering. This result implies the need to use implicit numerical integrators to prevent destabilization of iterations during optimization.

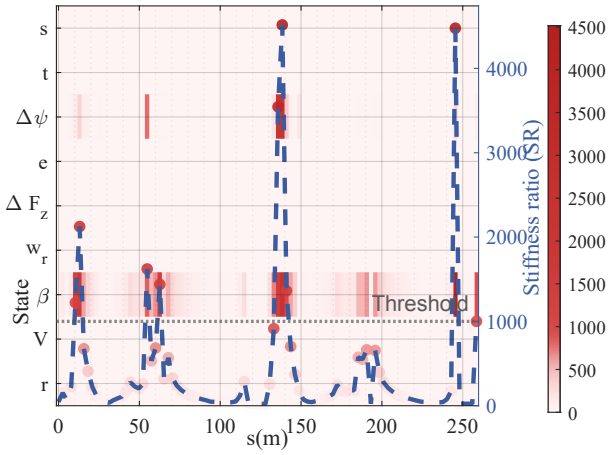

Fig. 2: Node-wise stiffness ratio computed around a time-optimal race trajectory with backtracking state contributions.

We study the impact of different integrators for the map $\mathcal{F}$ in (2) primarily on two metrics: (a) the solution accuracy compared to a ground truth, (b) the computational time required to solve the OCP (2). Each integrator is evaluated in combination with three solvers—`IPOPT`, `KNITRO`, and `SNOPT` [18]. The test problem is a time-optimal 260 m oval track trajectory ($N = 100$), initialized from a dynamically feasible centerline guess. This study is motivated by the need to match an integrator's properties, such as its stability and order, with the underlying system dynamics and its characteristics, such as stiffness. We benchmark nine numerical integrators: explicit methods (RK2, RK4) and implicit methods (Implicit Euler, Backward Differentiation Formula (BDF) 4, 5, 6, Crank-Nicolson, Adams-Moulton 3-stage, and Gauss-Legendre 2-stage). A key property for differentiating implicit schemes can be A-stability; an A-stable integrator guarantees that the numerical solution to a stable physical problem will not become unstable, regardless of the integration step size [22]. We note that among the methods tested, only the Implicit Euler, Crank-Nicolson, and Gauss-Legendre methods are A-stable. The ground truth solution was generated with a 6th-order A-stable Gauss-Legendre-3s method, verified against CVODES integration solver in Casadi [24]. Although they produced identical results, both were omitted from the benchmark study due to their prohibitive solve times.

V. Robust Initial Guess Formulation

A strategy for initializing complex OCPs is first to generate a dynamically consistent trajectory to serve as an initial guess. While a dynamically feasible guess can help initialize the main OCP (2), generating this guess itself requires solving another OCP of similar complexity. This approach effectively layers one complex optimization problem on top of another, increasing the overall computational burden and introducing further points of failure. To circumvent the complexity of generating a full trajectory guess, we propose a simple yet powerful initialization method.

We propose a box-constrained nonlinear least square problem minimizing the dynamic rates (subset of the full system state) given by $\dot{x}_{\text{dyn}} = [\dot{r}, \dot{V}, \dot{\beta}, \dot{\omega}_r, \dot{\Delta F}_z]^T$, which is solved to an optimality cost of zero, ensuring feasibility of the dynamic part of the equations of motion (1). We restrict ourselves to finding a single dynamically feasible set of state and control pairs for a given vehicle, making it computationally cheap, and initialize the dynamic components (x_{dyn}) at every node of the OCP with this pair. Since direct multiple shooting is employed, that is every node has its own state and control variables, this computed state-control pair should satisfy the $x_{\text{dyn}_{k+1}} = \mathcal{F}(x_{\text{dyn}_{k+1}}, x_{\text{dyn}_k}, u_k; \Delta s_k)$ at every node of the OCP (as rates are zero) bringing down the feasibility error of the guess. The kinematic components $e, \Delta\phi$ are initialized with zeros representing centerline position, s is known a priori, and t is initialized by using an average velocity guess V_{avg} and total path distance $s_{\text{end}} - s_0$. The initial guess is formulated as follows:

$$\min_{\{x_{\text{guess}}\},\{u_{\text{guess}}\}} \dot{r}^2 + \dot{V}^2 + \dot{V}_{\omega_r}^2 + \dot{\Delta F_z}^2 + \dot{\beta}^2 + (V - V_{\omega_r})^2$$
$$\text{(bounds)} \begin{cases} 0^- \leq r_{\text{guess}} \leq 0^+, \\ V_{\min} \leq V_{\text{guess}} \leq V_{\max}, \\ \underline{u} \leq u_{\text{guess}} \leq \overline{u} \end{cases} \tag{11}$$

We emphasize that the cost function and bounds in the above problem can be customized to produce different responses; we opted to obtain a straight line driving condition and introduce $(V - V_{\omega_r})^2$ to minimize slip. The guess once computed for a vehicle can be fixed and *is not required to be recomputed*.

VI. Results

We frame the OCP (2) for a Lexus LC 500 vehicle model [14] in MATLAB R2024b via CasADi [24] interfaced with `IPOPT` [16], `SNOPT` [18], and `KNITRO` [17] NLP solvers. The benchmarks were run on a desktop PC with a 5.7GHz AMD Ryzen 9 9950X processor, with Just-In-Time (JIT) compilation enabled for all results except those in Table I.

A. Integrator and Solver Tandem Study Results

TABLE I: CPU times (s) by integrator and solver. **Bold** = fastest optimal; red = suboptimal ; NS = No Solution.

Integrator	KNITRO	IPOPT	SNOPT
Implicit Euler	**0.087**	0.535	0.686
BDF5	**0.094**	0.280	NS
BDF6	**0.122**	0.719	1.120
Crank–Nicolson	**0.151**	0.283	0.885
BDF4	0.231	**0.201**	1.630
Adams–Moulton 3s	**0.251**	0.919	1.500
Gauss–Legendre 2s	**0.847**	1.212	3.070
RK4	**8.539**	NS	1.870
RK2	1.530	7.024	1.050

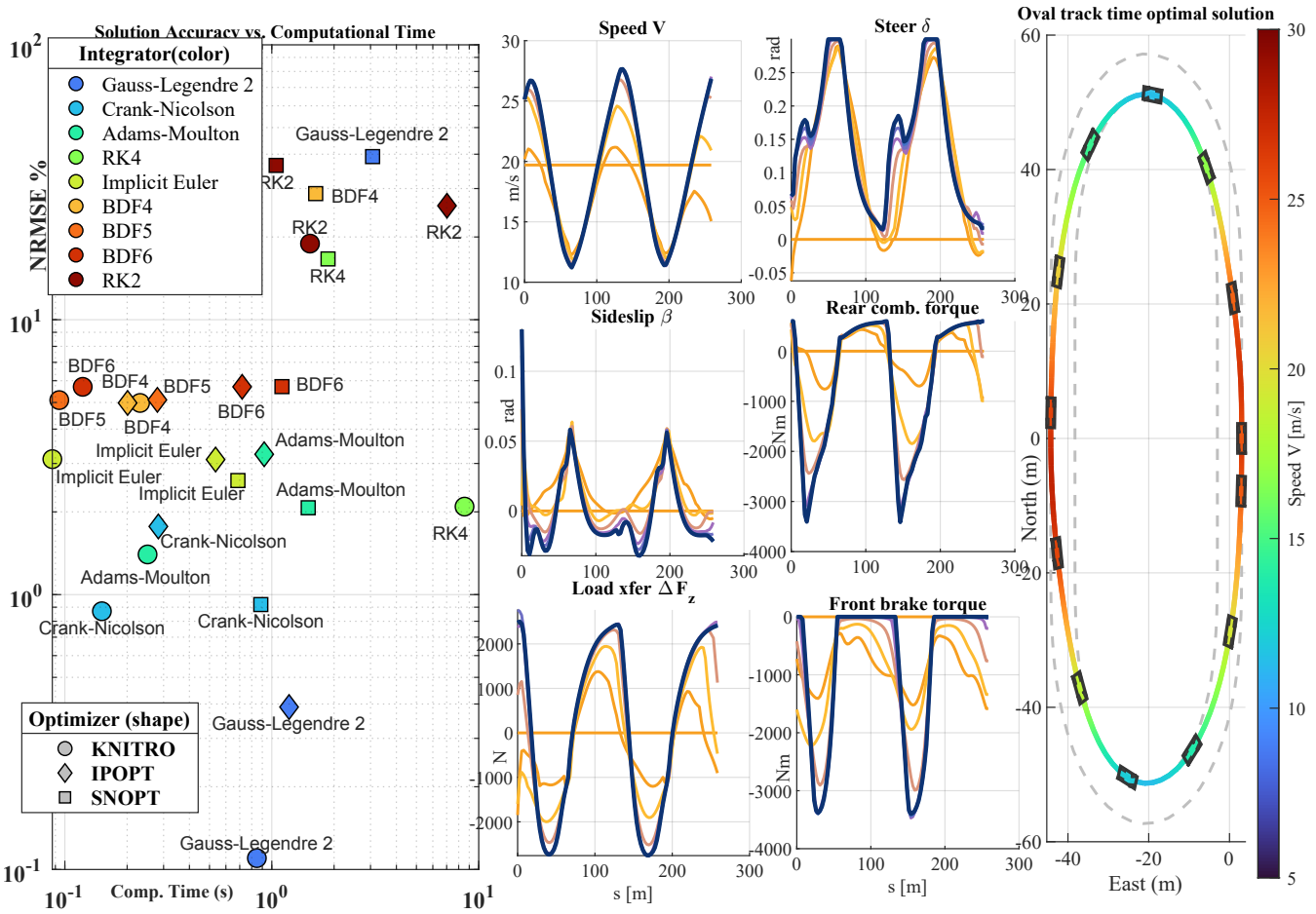

Fig. 3: (a) Different numerical integrators and optimization solver pairs plotted with NRMSE% depicting accuracy (Y-axis) and computation time (X-axis). (b) Iteration-wise converged time-optimal solution generated under $55ms$ (22) iterations for an oval track starting from the introduced guess strategy, illustrating optimal vehicle path, states, and control sequences.

The comprehensive results for the integrator study introduced in Sec. IV are summarized in Figure 3, and Table I. Solution accuracy is quantified by the normalized root mean square error (NRMSE) for each control input $u_j, j = 1, 2, 3$, which is the standard root mean square error between the computed trajectory u_j and the ground truth, u_j^*. Figure 3 illustrates the trade-off between solution accuracy and computational cost by plotting the maximum NRMSE % made in a control input against the CPU solve time for each integrator-solver pair (averaged over 10 consecutive solves of each combination). Table I provides a detailed breakdown of the CPU times for each integrator-solver pair. Note that rather than employing hidden internal Newton iterations to resolve the implicit dynamics at each step, the optimizer solves for the states, controls, and any multi-stage variables simultaneously as a single NLP problem. We summarize our findings as follows:

- **Implicit vs Explicit methods:** The results confirm the unsuitability of explicit integrators for this problem, aligning directly with our stiffness analysis (Sec. IV). The stiff dynamics impose severe penalties on explicit schemes, placing both RK2 and RK4 in the top-right quadrant of Figure 3(a), characterized by high computational cost and poor accuracy. None of the solvers converge to the optimal solution when using the single-stage RK2 method, as shown in Table I. Although KNITRO succeeds with the four-stage RK4, its solve time of 8.54 seconds is prohibitive for real-time use as it is nearly 100 *times slower* than the implicit Euler scheme.

- **Accuracy and A-stability:** A critical insight from Figure 3 (a) is that for this stiff dynamics, simply using a higher order integrator does not guarantee a more accurate solution. Instead, the property of A-stability emerges as the dominant factor. A clear pattern is visible among the A-stable implicit methods. As the order increases from the 1st-order implicit Euler (3–5% error) to the 2nd-order Crank-Nicholson ($<$ 1% error), and finally to the 4th-order Gauss-Legendre 2 (0.1% error), the solution accuracy consistently improves. Conversely, non-A-stable, higher-order methods like BDF 4-6 and Adams-Moulton 3-stage violate this trend, yielding larger errors than the second-order Crank-Nicholson.

- **Computational Efficiency:** To identify the best configuration for our problem, we fix an upper bound on the maximum NRMSE % to be 5%, and the upper bound on computational time to be 0.2 seconds. Within these bounds, the combination of the A-stable implicit Euler integrator and the KNITRO

solver performs best, achieving convergence in just 0.087 seconds (Table I) with an error margin of $3-4\%$.

- **Solver Performance:** As shown in Table I and Figure 3, the second-order IP solvers (KNITRO, IPOPT) exhibit superior robustness than the first-order SQP-based SNOPT, which frequently converges to suboptimal solutions. This performance gap underscores the importance of exact Hessian information for reliable convergence in highly nonlinear problems, where the quasi-Newton approximations employed by SNOPT appear insufficient. While both IP solvers demonstrate consistent robustness, KNITRO achieves notably faster solve times—often two to three times faster than IPOPT.

B. Robust Initial Guess Results

TABLE II: KNITRO per-segment performance: zero guess vs. robust initial guess. NS = evaluation/convergence failure

	Zero guess		Robust init		Comp time gain (×)
Segment	Iter	Time [s]	Iter	Time [s]	
S01	NS	NS	27	0.073	NS
S02	2005	7.674	40	0.098	×78.30
S03	NS	NS	40	0.105	NS
S04	807	2.967	62	0.153	×19.39
S05	NS	NS	62	0.139	NS
S06	1227	5.007	35	0.085	×58.90
S07	693	2.585	39	0.100	×25.85
S08	202	0.689	66	0.159	×4.34
S09	1675	6.542	38	0.091	×71.89
S10	937	3.763	49	0.136	×27.67

Using the optimal integrator-solver pair (Implicit Euler with KNITRO) we found for our problem, we evaluate the impact of our robust initialization strategy (Sec. V) against an ill-informed, zero guess. The feasibility error representing the maximum constraint violation is computed for the time-optimal OCP (2) around an oval track at the two guesses. The robust guess provides a starting point with an error of just 9×10^{-1}; conversely, the ill-informed guess is strongly infeasible, beginning with an error of 6.7×10^{3}. Moreover, when the same time-optimal problem is solved across the following robustness test for 10 different racing track segments [8], a similar difference in initial feasibility error is observed. This *four-order-of-magnitude decrease* in initial infeasibility effectively transforms the optimization landscape, providing a better starting point for the optimizer. Figure 3 (b) illustrates the iteration-wise descent (light yellow to dark blue) towards the optimal solution that took only 55 ms and 22 iterations to converge. Particularly, the dynamic states and controls are initialized with the same constant $x_{\text{guess}}, u_{\text{guess}}$ computed using the proposed guess formulation in Sec. V, from where the optimization proceeds towards an optimum.

Further, to isolate the impact of the robust initial guess from the robustness of the optimizer, keeping the integrator-solver pair intact, we solve a time-optimal OCP on 10 distinct 250 m ($N = 100$) segments of a racetrack [8], starting from the fixed robust guess and a zero guess (initializing all decision variables with 0). The results, presented in Table II, highlight three critical findings. First, the robust guess is essential for convergence. The zero guess fails to find a solution in 30% of the cases (S01, S03, S05). Second, the robust guess enables real-time performance. The zero guess never meets the 200 ms real-time target, with its fastest solve time being around 689 ms, whereas the robust guess consistently delivers solutions in under 160 ms. Finally, the proposed strategy is both efficient and generalizable; the same guess once computed works on all 10 diverse segments.

C. Optimization Solvers

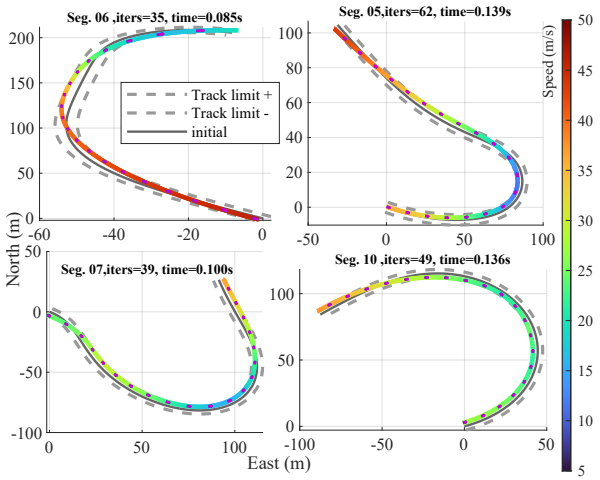

Fig. 4: Racing track segments time-optimal paths.

Since SNOPT consistently produced sub-optimal solutions (Table I) in the integrator test study, we limit our discussion to the parameters that are imperative to stabilize and produce better performance for the IP Methods.

- **Barrier Parameter Influence:** The barrier parameter update strategy is a critical hyperparameter in IP methods, directly impacting convergence and solution quality [17], [25]. Our tests show both IP solvers are highly sensitive to this choice, with a poor selection leading to suboptimal or failed solutions (Table III). For IPOPT, we found its default monotone strategy frequently failed to converge across different tests, in contrast to the adaptive strategy, which was found to be more robust. A similar effect was observed for KNITRO, where the advanced dampmpc strategy—a safeguarded predictor-corrector rule—solved the problem in just 22 iterations, while a basic monotone approach failed (Table III). Further details on various barrier strategies can be found in [25] (excluded here for brevity).

TABLE III: KNITRO barrier update strategies with iteration count for oval OCP

	dampmpc	probing	fullmpc	adaptive	quality	monotone
Iterations	**22**	29	29	35	50	289
Optimality	optimal	optimal	optimal	optimal	optimal	Sub-optimal

- **Solver Robustness, Real-Time Performance, and Convergence:** Building on our prior results, we next compare IPOPT and KNITRO on 10 time-optimal distinct race track segments. We use our robust guess to evaluate each solver's

capacity for real-time convergence (< 200 ms), reporting the results in Table IV. The benchmark results underscore KNITRO's robustness for real-time trajectory optimization, as it consistently meets the 200 ms performance target across all segments. In contrast, IPOPT proves unreliable, exceeding the time limit on 90% of the cases. This performance gap is starkly highlighted by segment $S06$, where IPOPT's solve time exceeds 10 seconds (due to a sharp increase in per-iteration cost)—over 126 times slower than KNITRO—rendering it unsuitable for time-critical tasks. Across all ten segments in Figure 5, KNITRO rapidly drives the optimality error $\|Z - Z_\star\|_2$ down by multiple orders of magnitude and then settles into a uniform linear-rate tail, reaching tolerance by < 70 iterations without oscillations. A few of the converged time-optimal path trajectories obtained from KNITRO in Table IV are visualized in Figure 4.

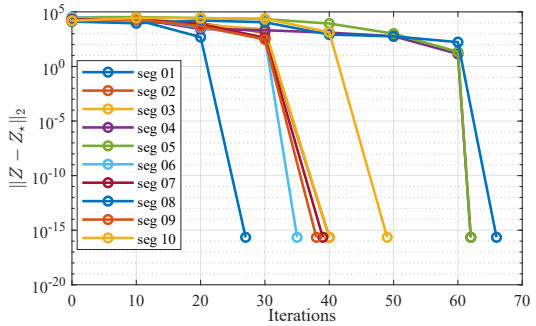

Fig. 5: Racing track segments convergence sequence.

TABLE IV: Robust guess with IPOPT (adaptive μ) & KNITRO (dampmpc μ). Times $>$ **0.2 s** in red, least in **bold**

	IPOPT (adaptive μ)		KNITRO (dampmpc μ)		Time gain (×)
Segment	Time [s]	Iter	Time [s]	Iter	
S01	0.406	82	**0.073**	27	×5.56
S02	0.178	36	**0.098**	40	×1.82
S03	0.415	102	**0.105**	40	×3.95
S04	0.485	111	**0.153**	62	×3.17
S05	0.461	101	**0.139**	62	×3.32
S06	10.714	129	**0.085**	35	×126.05
S07	0.783	90	**0.100**	39	×7.83
S08	0.323	76	**0.159**	66	×2.03
S09	0.324	61	**0.091**	38	×3.56
S10	0.570	81	**0.136**	49	×4.19

• **KNITRO and IPOPT: Globalization and Bad Inertia Strategies:** We specifically pay attention to the case of $S06$ where IPOPT takes about two orders of magnitude more time as compared to KNITRO, for just 129 iterations, attributing to a transient blow-up in per-iteration linear-algebra cost. Both solvers employ different globalization techniques, such as using a merit function like (5) in KNITRO or a filter-based method [16], that can influence how the iterates proceed. However, the primary difference stems from handling bad inertia in the Newton step or indefiniteness of the Hessian $\nabla^2_{\eta\eta}\mathcal{L}$ eq (6). On segment $S06$, IPOPT's time surge stems from its inertia-correction strategy: when the KKT system is indefinite or nearly singular, IPOPT repeatedly re-factorizes with regularization to enforce inertia. In contrast, KNITRO's hybrid IP algorithm safeguards by switching to a trust-region based conjugate gradient (CG) step when bad inertia is detected, stabilizing the step in a few CG iterations and achieving real-time performance.

D. Emergency Collision Avoidance Test

We put our architecture to test on two high-speed (90 km/h) emergency collision avoidance scenarios similar to those in [1]. Figure 6 highlights the framework's versatility, showing it can autonomously generate either a stable, low-sideslip maneuver (< 58 ms) for a forgiving scenario or a controlled drift (> 0.4 rad sideslip, < 72 ms) for a more critical one. We empirically prove this aggressive drift is the *only* viable solution, as reinstating the tire saturation constraints (2) renders the problem infeasible. Despite the complexity, both trajectories were generated in under 72 ms, confirming real-time performance. This showcases our framework's ability to act as a unified planner that obviates the need for specialized drift controllers while providing a real-time *certificate of feasibility*.

Further, we generated similar high-speed oval racing and collision avoidance trajectories for a test all-terrain vehicle (ATV) to validate real-world plausibility. A full-fidelity model of this vehicle is simulated by BeamNG, and the optimal trajectories are generated using a low-fidelity, data-driven model (similar to (1)) trained on BeamNG data. We track the reference trajectory using a short-horizon Model Predictive Control (MPC) running online.

Figure 1 and Figure 7 show successful double obstacle avoidance at high velocities 90 km/hr. The performance plots (Figure 7) show an initial velocity error due to a standing start and a maximum sideslip tracking error of up to ± 15 deg. The sideslip deviation is a deliberate consequence of the online tracking MPC's tuning, which prioritized path tracking over minimizing transient dynamic state errors to ensure safety. Future work aims to test these trajectories on the physical model of this ATV vehicle under development.

VII. CONCLUSIONS

We demonstrate that real-time, reference-free trajectory generation at the limits of vehicle handling is achievable through a principled initial guess strategy, integrator selection informed by system stiffness analysis, and tailored interior point solver techniques for barrier updates and nonconvexity handling. The framework was validated across multiple race track segments for time-optimal planning, and in a high-speed collision avoidance scenario that requires drift for feasibility. Future work will implement the proposed architecture on real-world race cars. We will investigate how the current stiffness analysis and equilibrium-based initial guess generalizes to other system dynamics.

REFERENCES

[1] D. Li, J. Zhang, and S. Lin, "Planning and control of drifting-based collision avoidance strategy under emergency driving conditions," *Control Engineering Practice*, vol. 139, p. 105625, 2023.

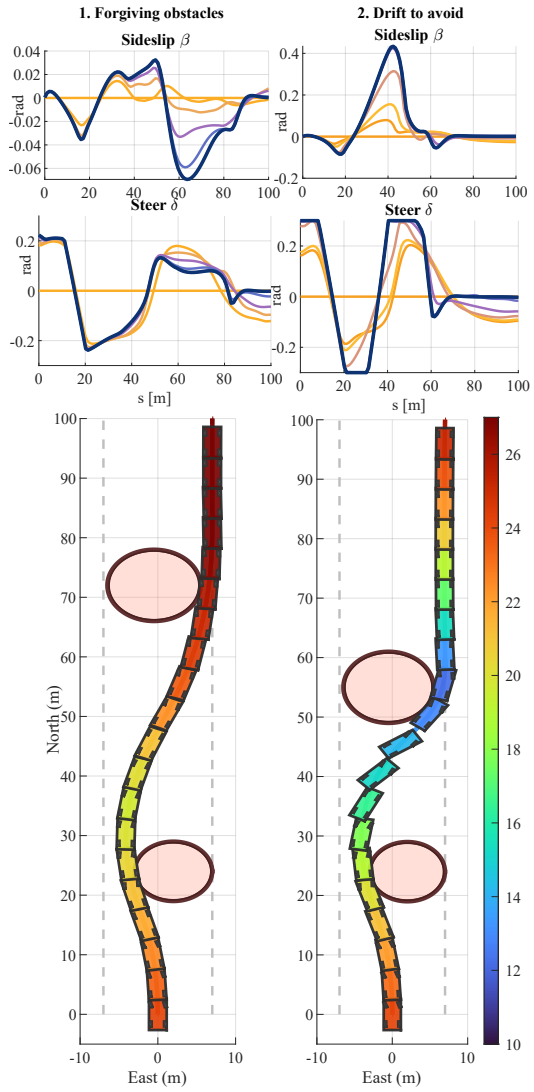

Fig. 6: Collision avoidance test trajectories: left (forgiving obstacle), right (requires drifting maneuver).

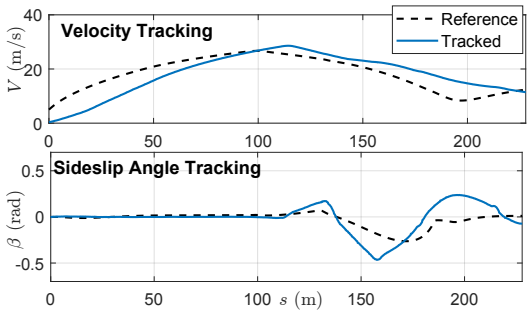

Fig. 7: Tracking performance velocity and sideslip for high-speed emergency collision avoidance in BeamNG.

[2] T. Zhao, E. Yurtsever, and G. Rizzoni, "Justifying emergency drift control for automated vehicles," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 141–148, 2022.

[3] B. Olofsson, K. Lundahl, K. Berntorp, and L. Nielsen, "An investigation of optimal vehicle maneuvers for different road conditions," *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 66–71, 2013.

[4] T. Zhao, E. Yurtsever, R. Chladny, and G. Rizzoni, "Collision avoidance with transitional drift control," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 907–914.

[5] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 4239–4244.

[6] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Dynamic systems and control conference*, vol. 44175, 2010, pp. 265–272.

[7] J. K. Subosits and J. C. Gerdes, "From the racetrack to the road: Real-time trajectory replanning for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 309–320, 2019.

[8] M. Thompson, J. Dallas, J. Y. Goh, and A. Balachandran, "Adaptive nonlinear model predictive control: maximizing tire force and obstacle avoidance in autonomous vehicles," *IEEE Transactions on Field Robotics*, 2024.

[9] V. A. Laurense and J. C. Gerdes, "Long-horizon vehicle motion planning and control through serially cascaded model complexity," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 1, pp. 166–179, 2021.

[10] J. Y. M. Goh, *Automated vehicle control beyond the stability limits*. Stanford University, 2019.

[11] J. Y. Goh, M. Thompson, J. Dallas, and A. Balachandran, "Beyond the stable handling limits: nonlinear model predictive control for highly transient autonomous drifting," *Vehicle System Dynamics*, vol. 62, no. 10, pp. 2590–2613, 2024.

[12] T. P. Weber and J. C. Gerdes, "Modeling and control for dynamic drifting trajectories," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 2, pp. 3731–3741, 2023.

[13] J. A. Talbot, *Optimal vehicle control under friction uncertainty-from driver assistance to drift transitions*. Stanford University, 2024.

[14] F. Djeumou, M. Thompson, M. Suminaka, and J. Subosits, "Reference-free formula drift with reinforcement learning: From driving data to tire energy-inspired, real-world policies," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 3610–3616.

[15] F. Domberg, C. C. Wembers, H. Patel, and G. Schildbach, "Deep drifting: Autonomous drifting of arbitrary trajectories using deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7753–7759.

[16] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[17] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59.

[18] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[19] BeamNG GmbH, "BeamNG.tech," https://www.beamng.tech/, 2025, version 0.35.0.0, Accessed on June 15, 2025.

[20] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[21] H. Mittelmann, "Ampl-nlp benchmark," Jun. 2025, updated June 28, 2025. Benchmarks for Optimization Software.

[22] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, ser. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 1996.

[23] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, ser. Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems. Springer, 1993.

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[25] J. Nocedal, A. Wächter, and R. A. Waltz, "Adaptive barrier update strategies for nonlinear interior methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1674–1693, 2009.