# The optimal racing line

Minimum-lap-time optimal control by direct collocation

Muhammet Emir Fil

B.Sc. Computational Engineering Science, RWTH Aachen • Technical report, June 21, 2026

**Aim.** Compute the minimum-lap-time racing line as the solution of an optimal-control problem rather than drawing it by hand, and verify the result against closed-form optima. The goal is a trajectory-optimisation solver, built from the vehicle model up, in which the racing line emerges from the tyre-grip limit and the lap time is checked against analytical references. The point mass on a friction circle is a deliberate baseline.

**Abstract**

Given a track and a tyre-grip limit, the fastest way round is the solution of a minimum-lap-time optimal-control problem: find the controls (how to brake, turn, and accelerate) that minimise the time to traverse a fixed arc length, subject to the vehicle dynamics and the friction limit. This report formulates that problem, removes its free final time by switching the independent variable to arc length, transcribes it to a nonlinear program by direct collocation with hand-derived analytic Jacobians, and solves it. On a 2.4 km circuit the optimal line is 8.1 % faster than the geometric centreline and rides the grip limit for nearly the whole lap, and its shape (straight-line braking, late apex, power-down on exit) emerges without being imposed. The result is verified against closed-form optima; the core is numpy and scipy.

## 1 The problem

A lap time is a functional of how the car is driven. Minimising it is a free-final-time optimal-control problem: the final time is the quantity to minimise, which makes the time horizon itself an unknown. The standard reformulation removes this difficulty by changing the independent variable from time $t$ to arc length $s$ along the track centreline. The finish is then a fixed boundary, $s = L$, and the lap time becomes a definite integral to minimise. The vehicle model is a baseline by choice: the single-track (bicycle) model is the natural next layer and would warm-start from this solution.

## 2 Formulation

**States and controls.** In the Frenet frame attached to the centreline the car has three states as functions of $s$: lateral offset $n$, heading relative to the track tangent $\xi$, and speed $v$. The controls are longitudinal and lateral acceleration $(a_t, a_n)$, limited by the friction circle

$$a_t^2 + a_n^2 \le (\mu g)^2. \tag{1}$$

**Dynamics.** With track curvature $\kappa(s)$ and $D = 1 - n\kappa$, the arc-length dynamics are

$$\frac{dn}{ds} = D\tan\xi, \qquad \frac{d\xi}{ds} = \frac{D\,a_n}{v^2\cos\xi} - \kappa, \qquad \frac{dv}{ds} = \frac{D\,a_t}{v\cos\xi}, \qquad \underbrace{\frac{dt}{ds} = \frac{D}{v\cos\xi}}_{\text{lap-time integrand}}. \tag{2}$$

The objective is the lap time $T = \int_0^L (dt/ds)\, ds$. The track is enforced as a bound $|n| \le \frac{1}{2} w(s)$, and the lap is closed by periodic boundary conditions $\mathbf{x}(0) = \mathbf{x}(L)$, which is what makes the optimiser solve the corners coupled (the line through one corner depends on the next).

**Transcription.** The continuous problem is discretised by trapezoidal direct collocation on $N$ intervals: states and controls at the nodes are the unknowns $\mathbf{z}$, the dynamics (2) become equality defect constraints between adjacent nodes,

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \tfrac{1}{2}\,\Delta s_k (\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0}, \tag{3}$$

and the objective is the trapezoidal sum of $dt/ds$. This yields a nonlinear program (NLP) solved with scipy's SLSQP. The solver is handed exact analytic Jacobians of the objective gradient, the defects (3), and the friction constraint (1), derived by hand from Eq. (2) rather than finite-differenced.

**Conditioning.** A closed-circuit NLP is stiff: the states span two orders of magnitude and the periodic constraint couples the whole lap. Three ingredients make it converge cleanly: variable scaling to dimensionless decision variables; a quasi-steady-state warm start (the $g$–$g$-limited speed profile on the centreline, $v = \sqrt{a_{\max}/|\kappa|}$ with a forward acceleration pass and a backward braking pass, so the NLP only has to find the line); and mesh continuation.

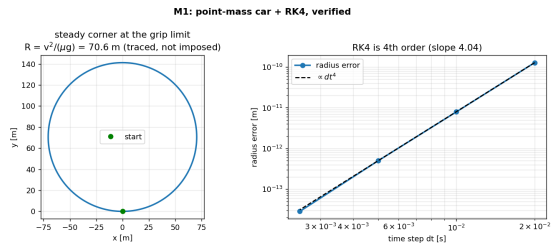
## 3 Verification

Tight defects alone only prove the NLP satisfied its own constraints. Correctness is established against independent references at two levels, the forward model and the optimiser (Table 1).

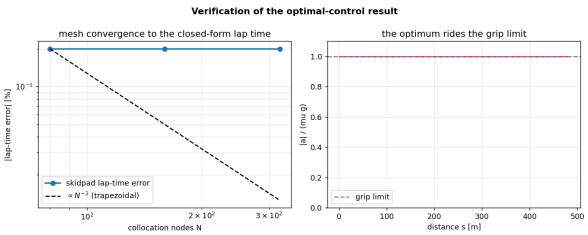
**Table 1:** Verification of the forward car and the optimal-control solver.

| Check | Reference | Result |
|---|---|---|
| Analytic Jacobians vs finite difference | exact derivatives | $< 1.2 \times 10^{-6}$ |
| Steady corner radius | $v^2 = \mu g R$ | exact |
| RK4 integration order | 4th order | slope 4.04 |
| Skidpad lap time (closed form) | $T = 2\pi\sqrt{R/\mu g}$ | matches to $-0.2\,\%$ |
| Mesh convergence of lap time | trapezoidal $\sim \Delta s^2$ | converges |
| Friction-circle saturation | optimal control rides the limit | about $100\,\%$ at limit |

The two non-trivial checks: the analytic Jacobians match finite differences to better than $1.2 \times 10^{-6}$; and the converged lap time matches the closed-form skidpad optimum $T = 2\pi\sqrt{R/\mu g}$ to $-0.2\,\%$, with the residual explained by the finite track width. Mesh refinement confirms the trapezoidal $\mathcal{O}(\Delta s^2)$ convergence.




**(a)** Forward car: energy conservation, steady corner $v^2 = \mu g R$, RK4 order 4.04.




**(b)** Optimiser: skidpad closed-form match, mesh convergence, friction saturation.

**Figure 1:** Verification at both levels: the forward dynamics and the optimal-control solve.

## 4 Result: the line emerges

On a stylised 2.4 km Grand Circuit ($\mu = 1.5$), the optimal lap is 38.76 s, against 42.19 s for the same car constrained to the geometric centreline. The optimal racing line is therefore 3.43 s (8.1 %) faster. The maximum defect is about $5 \times 10^{-11}$.

From minimising time under Eqs. (1)–(2) alone, the solver finds that the fast way round is to brake in a straight line, use the full track width to straighten the corner (a larger effective radius buys a higher apex speed), trail-brake to a late apex, and get back to power on exit. The $g$–$g$ diagram (Fig. 2) shows the car visiting every point on the friction circle and sitting on it for nearly the whole lap, which is the signature of a time-optimal trajectory.

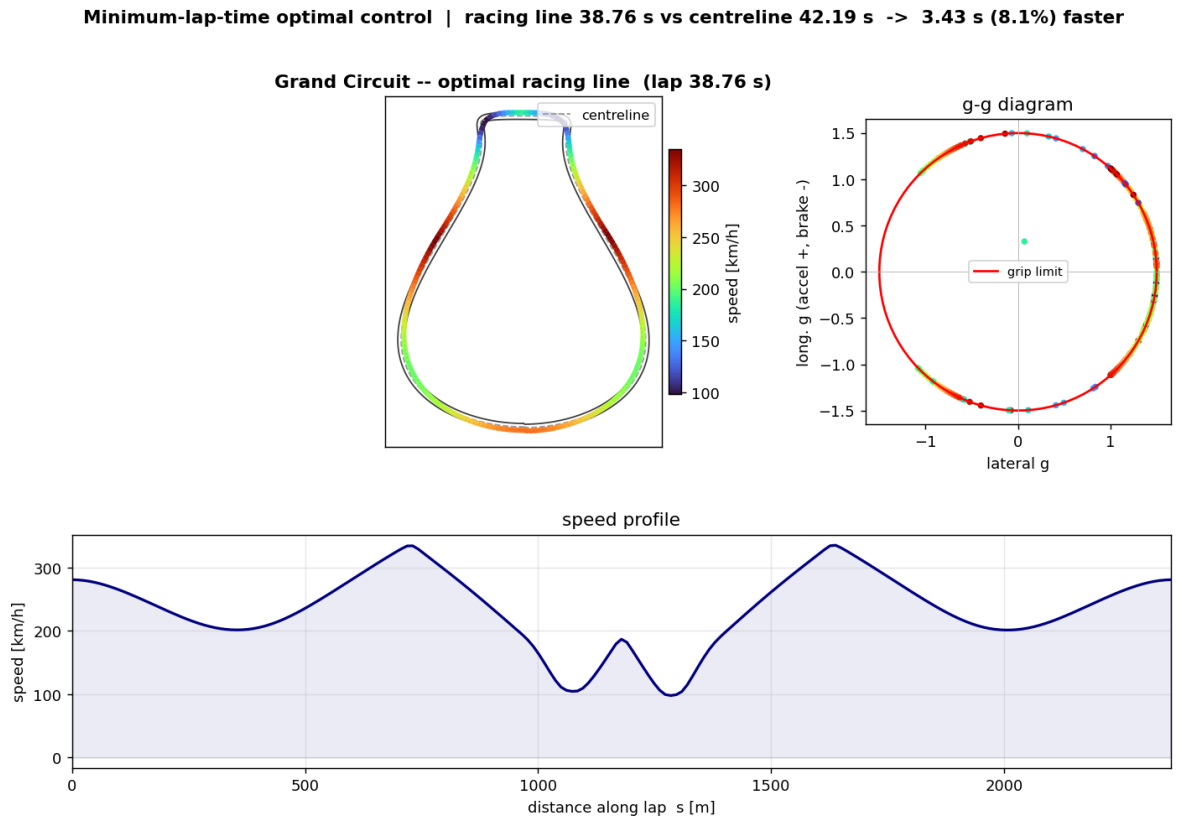



**Figure 2:** The optimal lap. Left: the racing line coloured by speed, straightening the corners against the centreline. Middle: the $g$–$g$ diagram saturating the friction circle. Right: the speed profile (brake, apex, accelerate).

## 5 Real car physics on the same core

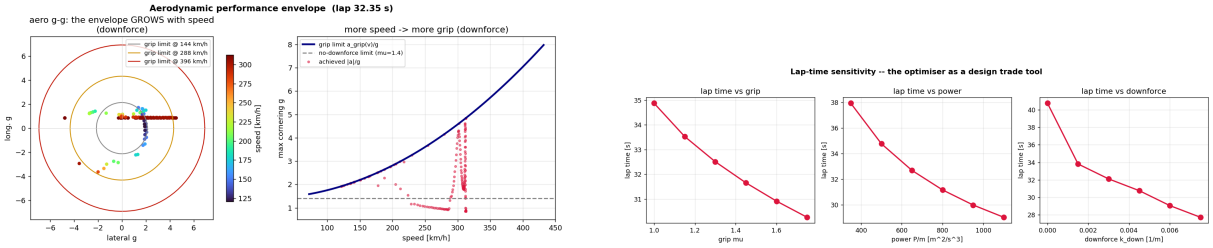
Each real-car effect is one extra term in the dynamics plus its analytic Jacobian row, and all default off so the baseline stays the pure friction circle: an engine power limit $a_t \le P/(mv)$, aerodynamic drag $a_{\text{drag}} = \frac{1}{2}\rho C_d A\, v^2/m$, and downforce, which makes the grip limit grow with speed,

$$a_{\text{grip}}(v) = \mu(g + \tfrac{1}{2}\rho C_l A\, v^2/m). \tag{4}$$

The $g$–$g$ circle then becomes a speed-dependent performance envelope (Fig. 3): fast corners pull more $g$ than slow ones (up to $4.7g$ at 310 km/h).

# 6 The optimiser as a design tool

Because the solve is cheap and warm-starts across parameters, re-solving the lap while sweeping a parameter turns the model into a trade study (Fig. 3, right): lap time versus grip (wet to slicks), versus power (diminishing returns), and versus downforce (40.8 s to 27.7 s across the swept range).




**(a)** Speed-dependent $g$–$g$ envelope with downforce.

**(b)** Lap-time sensitivity: grip, power, downforce.

**Figure 3:** The same optimal-control core, used for real-car physics and as a design tool.

# 7 Limitations and outlook

- The vehicle is a point mass on a friction circle: no load transfer, no yaw dynamics, no combined-slip tyre. This is a deliberate, hand-checkable baseline.
- The natural next layer is the single-track (bicycle) model with a combined-slip tyre, which would warm-start directly from this solution.
- SLSQP is a dense SQP and does not scale past $N \sim 250$ nodes; for much finer meshes an interior-point solver (IPOPT) exploiting sparsity is the right tool.

# 8 Reproducibility

The core is numpy and scipy only. Each result is reproduced by a script that prints its numbers and saves the figure shown here: `verify_car.py` (forward car + RK4), `verify_jac.py` (analytic Jacobians vs finite difference), `run_lap.py` (the lap, the $g$–$g$ diagram, and the centreline gain), `verify_lap.py` (skidpad, mesh convergence, saturation), and `sweeps.py` (aero envelope and sensitivity). An interactive web demo animates the car running the optimal lap with a live $g$–$g$ dot.

# References


[1] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed., SIAM, 2010.

[2] M. Kelly, "An introduction to trajectory optimization," *SIAM Review*, 59(4), 2017.

[3] G. Perantoni & D. J. N. Limebeer, "Optimal control for a Formula One car with variable parameters," *Vehicle System Dynamics*, 52(5), 2014.

[4] W. F. Milliken & D. L. Milliken, *Race Car Vehicle Dynamics*, SAE International, 1995.

[5] D. Kraft, "A software package for sequential quadratic programming," DFVLR-FB, 1988. (The SLSQP algorithm.)