IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Real-time Velocity Profile Optimization for Time-Optimal Maneuvering with Generic Acceleration Constraints

Mattia Piazza[1], Mattia Piccinini[2], Sebastiano Taddei[1,3], Francesco Biral[1], and Enrico Bertolazzi[1]

***Abstract*—The computation of time-optimal velocity profiles along prescribed paths, subject to generic acceleration constraints, is a crucial problem in robot trajectory planning, with particular relevance to autonomous racing. However, the existing methods either support arbitrary acceleration constraints at high computational cost or use conservative box constraints for computational efficiency. We propose FBGA, a new Forward-Backward algorithm with Generic Acceleration constraints, which achieves both high accuracy and low computation time. FBGA operates forward and backward passes to maximize the velocity profile in short, discretized path segments, while satisfying user-defined performance limits. Tested on five racetracks and two vehicle classes, FBGA handles complex, non-convex acceleration constraints with custom formulations. Its maneuvers and lap times closely match optimal control baselines (within 0.11%-0.36%), while being up to three orders of magnitude faster. FBGA maintains high accuracy even with coarse discretization, making it well-suited for online multi-query trajectory planning. Our open-source C++ implementation is available at: https://github.com/DRIVEWISE/FBGA.**

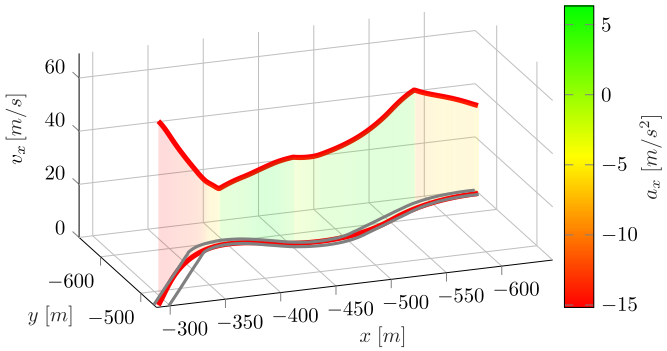
***Index Terms*—Optimization, Optimal Control, Velocity Planning.**

I. INTRODUCTION

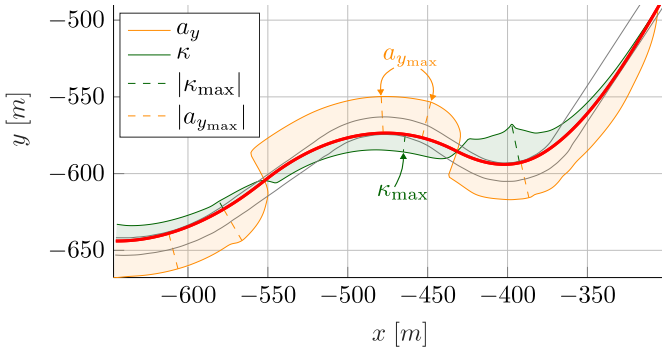
TIME-OPTIMAL velocity planning under acceleration constraints is a key problem in robotics, with applications in autonomous racing (Fig. 1) [1]–[3], drone flight [4], manipulators [5], and mobile robot navigation [6], [7]. These applications use point-mass models to compute time-optimal velocity profiles along fixed paths, subject to acceleration limits. Fast online generation of such profiles is essential in dynamic environments, where trajectory planners must evaluate many candidate maneuvers, often with graph- or sampling-based frameworks [1], [4], [8], [9].

However, the existing methods for minimum-time velocity planning face a trade-off: Optimal Control Problems (OCPs) [10] and Quasi-Steady-State (QSS) [11], [12] offer high accuracy, but their high computational cost does not allow online multi-query planning. Conversely, semi-analytical Forward-Backward (FB) and sequential methods [2], [7] are faster but limited to box-shaped acceleration bounds, which are overly conservative for racing vehicles. Accurate performance modeling of racing vehicles requires complex g-g-v diagrams (Fig. 2a) [11], [13]–[15], describing the speed-dependent coupling of the lateral and longitudinal acceleration limits.

This paper introduces a real-time Forward-Backward method with Generic Acceleration constraints (FBGA), matching the accuracy of OCPs while being up to three orders of magnitude faster, making it a suitable building block for online sampling-based trajectory planners.



(a) Path, velocity (v_x) and longitudinal acceleration (a_x).



(b) Lateral acceleration (a_y) and curvature (κ), with their local maxima.

Fig. 1. (a) Time-optimal velocity and longitudinal acceleration profiles computed by our FBGA along the first two corners of the Catalunya circuit. (b) Lateral acceleration and curvature profiles. Unlike QSS [16], FBGA does not assume that peak lateral accelerations ($a_{y_{\max}}$) occur at curvature peaks ($\kappa_{\max}$): in the second corner, $\kappa_{\max}$ happens between two $a_{y_{\max}}$ peaks.

A. Related Work

In robotics and autonomous racing, the methods to compute minimum-time velocity profiles on a given path fall into three families: (1) OCP with direct and indirect methods, (2) QSS, and (3) FB or related semi-analytical strategies.

Manuscript received: July, 11th, 2025; Revised September, 19th, 2025; Accepted November, 19th, 2025.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments.

[1] Department of Industrial Engineering, University of Trento, 38123 Trento, Italy, name.surname@unitn.it.

[2] Professorship of Autonomous Vehicle Systems, Technical University of Munich, 85748 Garching, Germany; Munich Institute of Robotics and Machine Intelligence (MIRMI), mattia.piccinini@tum.de.

[3]Department of Electrical and Information Engineering, Politecnico di Bari, 70125 Bari, Italy, s.taddei@phd.poliba.it.

This work was partly supported by the European Union - Next Generation EU - under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 1 Investment 3.4 - Decree No. 351 of Italian Ministry of University and Research - Concession Decree No. 2152 of the Italian Ministry of University and Research, Project code D93C22000500001, within the Italian National Program PhD Programme in Autonomous Systems (DAuSy).

Digital Object Identifier (DOI): see top of this page.

1) Optimal Control Methods: Minimum-time OCPs are typically solved using direct [17] or indirect [18] methods to jointly optimize path and velocity. They are used in offline lap-time minimization with multibody models [19], and in online planning with point-mass models for autonomous racing [10], [17], [20]. However, even with simplified models, OCPs remain computationally intensive and are not suitable for online evaluation of multiple trajectories in dynamic, non-convex race scenarios [3].

2) Quasi-Steady-State Methods: QSS methods [16] solve the same problem addressed in this paper: minimum-time velocity planning along a fixed path under g-g-v acceleration constraints. QSS splits the path at the curvature apexes and solves a sequence of nonlinear programs (NLPs) to compute the optimal velocity profiles. While effective, QSS methods are computationally expensive and unsuitable for online planning. For instance, [21] applied QSS to a point-mass motorcycle model with jerk bounds for offline lap-time optimization, and [11], [12] used QSS in free-trajectory planning with point-mass car models, yet their computational times were too high for online planning. Moreover, QSS critically depends on apex selection for numerical convergence, and assumes that the maximum lateral acceleration is at the path apexes, which is not always the time-optimal maneuver (Fig. 1).

3) Forward-Backward, Semi-Analytical and Alternative Methods: In race car trajectory optimization, two-step methods combining forward-backward (FB) velocity planning and minimum-curvature path generation were proposed in [22], [23], while [24] applied a similar iterative approach along a fixed path. However, their methods required tenths of seconds to minutes, and could not allow online re-planning. Conversely, the semi-analytical FB algorithm in [2] enabled online time-optimal speed planning, yet with box-constrained accelerations (rectangular g-g diagram). Similar speed planning problems were solved with range reduction [25], sequential line search [7], and convexification [26], adding acceleration rate and jerk limits. However, the box acceleration constraints used by [2], [7], [25], [26] remained the main limiting factor, being very conservative for racing vehicles, whose performance envelope is typically defined by speed-dependent, diamond-shaped g-g-v diagrams [14], [17], [27]. Still, such simplified constraints were used in sampling-based planners for autonomous racing with cars [8] and drones [4], where computational efficiency is prioritized over accurate performance modeling. We aim to extend these methods to support generic acceleration constraints while maintaining computational efficiency.

4) Critical Summary: To the best of our knowledge, existing methods are limited by at least one of the following:

1) OCP and QSS methods are not suitable for online multi-query planning due to their relatively high computational cost [10], [11], [21]. Also, QSS assumes that the maximum lateral acceleration is reached at the corner apexes [16], which may be non-time-optimal.
2) Semi-analytical FB [2], sequential and convexified methods [7], [25], [26] are computationally efficient for online speed planning but limited to rectangular g-g acceleration constraints, which poorly represent the true performance envelope of vehicles and robots.

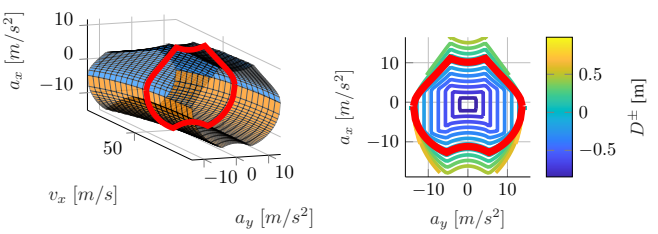


Fig. 2. (a) g-g-v diagram of the racing motorcycle model [28] used in Sec. III. (b) Output of the $D^{\pm}$ signed distance function (Algorithm 1) for the red slice in (a). $D^{\pm}$ ranges from -1 (inside the g-g-v envelope) to $+\infty$ (outside). The red line denotes the envelope boundary, where $D^{\pm} = 0$.

B. Contributions

To address the limitations of existing methods, we propose the following contributions:

1) FBGA: a new method for time-optimal velocity planning along a given path. For the first time, FBGA enables real-time planning with generically-shaped acceleration constraints (g-g-v diagrams).
2) We validate FBGA on different racetracks, using both race car and motorcycle examples, and show that it can accurately compute time-optimal velocity profiles for different vehicle classes and acceleration constraints.
3) We compare our approach against OCPs, achieving lower computational cost (up to three orders of magnitude) with the same level of accuracy, showing the potential for online planning in autonomous racing.

II. METHODOLOGY

A. Time-Optimal Velocity Planning Problem

We consider the following velocity planning problem along a given path:

$$\min_{a_x} \quad T \tag{1a}$$
$$\text{s.t.} \quad \dot{s} = v_x, \quad \dot{v}_x = a_x, \tag{1b}$$
$$\Gamma_y^-(v_x) \le a_y = \kappa(s)\, v_x^2 \le \Gamma_y^+(v_x), \tag{1c}$$
$$\Gamma_x^-\left(a_y, v_x\right) \le a_x \le \Gamma_x^+\left(a_y, v_x\right), \tag{1d}$$
$$v_x(0) = v_{\text{ini}},\ s(0) = 0,\ s(T) = L. \tag{1e}$$

Where T is the maneuver time to be minimized. The system dynamics (1b) is a double integrator, where the states $\{s, v_x\}$ are the curvilinear abscissa along the path and the longitudinal velocity, while the control input a_x is the longitudinal acceleration. The control a_x is bounded by the acceleration constraints (1c)-(1d), where $\kappa(s)$ is the curvature of the path as a function of s, $a_y = \kappa(s)\, v_x^2$ is the lateral acceleration, and $\Gamma_x^{\pm}(\cdot)$ and $\Gamma_y^{\pm}(\cdot)$ are functions providing the lower and upper bounds of the lateral and longitudinal accelerations. Finally, the boundary conditions (1e) specify the initial velocity v_{ini} and the final travelled distance L. In (1), all the variables are time-dependent, and the notation $\dot{\text{x}}$ denotes the time derivative of the variable x.

Assuming the lateral acceleration $a_y = \kappa(s)\, v_x^2$ places the model in the quasi-steady-state class, along a given path. Our formulation (1) condenses all dynamic nonlinearities in the acceleration constraints (1c)-(1d), which, for vehicles, define the g-g-v diagram. This diagram captures key physical effects, including tire saturation, actuation limits, aerodynamic drag, load transfer, and front/rear wheel lift in motorcycles, and expresses the maximum performance envelope in terms of center-of-mass accelerations.

Variants of the problem (1) were solved with optimal control for vehicle motion planning [10], [29], but their high computational cost hinders online planning of multiple trajectories in dynamic scenarios. This paper introduces FBGA, a novel algorithm that solves (1) with high accuracy and significantly lower computation time.

1) Solution with Acceleration Constraints at the Borders: Let us start by considering the problem (1) with the acceleration constraints (1c)-(1d) enforced only at borders ($s = 0$ and $s = L$). Since the maneuver time T in (1a) is unknown, to solve the problem we perform a change of independent variable, using the curvilinear abscissa s instead of the time t. This yields:

$$\min_{a_x} \quad \int_0^L \frac{\mathrm{d}s}{v_x(s)} \tag{2a}$$

$$\text{s.t.} \quad v_x'(s)\, v_x(s) = a_x(s), \tag{2b}$$

$$v_x(0) = v_{\text{ini}}\,, \text{ with (1c)-(1d) at } s = 0 \text{ and } s = L \tag{2c}$$

where $v_x'(s) = \mathrm{d}v_x(s)/\mathrm{d}s$. Using the Pontryagin's Maximum Principle, the problem (2) yields an analytical solution, which simplifies considering a constant longitudinal acceleration:

$$v_x(s) = \sqrt{2\int_0^s a_x(\zeta)\,\mathrm{d}\zeta + v_{\text{ini}}^2}\,\Big|_{a_x(\zeta)=a_x} = \sqrt{2sa + v_{\text{ini}}^2}\,. \tag{3}$$

Furthermore, the resulting maneuver time is:

$$T = \int_0^L \frac{\mathrm{d}s}{v_x(s)} = \frac{-v_{\text{ini}} + \sqrt{2aL + v_{\text{ini}}^2}}{a}\,. \tag{4}$$

Sec. II-C will use (3)-(4) on short discretized segments of the path, where a_x will be assumed piecewise constant and on the border of the acceleration constraints (1c)-(1d).[1] This assumption introduces discontinuities in a_x at the segment boundaries, which is accepted for real-time trajectory planning, but is not representative of the real vehicle dynamics. Jerk constraints could be enforced by increasing the algorithm's complexity, and will be part of future work.

B. Signed Distance from the g-g-v Envelope

Before describing the FBGA algorithm, we need to define an auxiliary function that computes the signed distance of a point from the g-g-v acceleration envelope. The g-g-v is defined by the functions $\Gamma_y^{\pm}(v_x)$ and $\Gamma_x^{\pm}(a_y, v_x)$ in (1c)-(1d), which represent the bounds of the lateral and longitudinal accelerations. An example of g-g-v shape is given in Fig. 2a, where a slice at constant speed is shown in red. Algorithm 1 introduces a new function, named $D^{\pm}(\cdot)$, to compute the signed distance of a point $P = (a_x, a_y, v_x)$ from the g-g-v. As shown in Fig. 2, $D^{\pm}(P)$ yields a positive value if P is outside the g-g-v envelope, and a negative value otherwise. The function is exactly zero if P is along the envelope.

Algorithm 1 $\mathrm{D}^{\pm}$ signed distance function

1: **Input:** point $P = (a_x, a_y, v_x)$
2: **Output:** signed distance D of P from the g-g-v envelope
3: $a_{y_{\text{clip}}} \leftarrow \text{CLIP}(a_y, \Gamma_y^-(v_x), \Gamma_y^+(v_x))$;
4: $a_{x_{\min}} \leftarrow \Gamma_x^-(a_{y_{\text{clip}}}, v_x)$; $\quad a_{x_{\max}} \leftarrow \Gamma_x^+(a_{y_{\text{clip}}}, v_x)$;
5: $D \leftarrow \Lambda\Big(\Phi\big(a_x, a_{x_{\min}}, a_{x_{\max}}\big), \Phi\big(a_y, \Gamma_y^-(v_x), \Gamma_y^+(v_x)\big)\Big)$;

Algorithm 1 starts by clipping the lateral acceleration a_y of P to the g-g-v bounds (line 3). Then, it retrieves the longitudinal acceleration limits $\{a_{x_{\min}}, a_{x_{\max}}\}$ for the clipped a_y and the given v_x (line 4). Finally, it computes the signed distance D using the functions $\Lambda(\cdot)$ and $\Phi(\cdot)$ (line 5). $\Lambda(\cdot)$ has the shape of a pyramid pointing downwards, forming a square trace on the horizontal plane:

$$\Lambda(x, y) = \max\big(x - 1, -1 - x, y - 1, -1 - y\big) \tag{5}$$

The function $\Phi(\cdot)$ maps a given range $[\mathrm{x}_{\min}, \mathrm{x}_{\max}]$ to $[-1, +\infty)$, enabling a consistent evaluation of the $\Lambda(\cdot)$ function:

$$\Phi(\mathrm{x}, \mathrm{x}_{\min}, \mathrm{x}_{\max}) = 2\,\frac{\mathrm{x} - \mathrm{x}_{\min}}{\mathrm{x}_{\max} - \mathrm{x}_{\min}} - 1 \tag{6}$$

The resulting function $D^{\pm}(\cdot)$ is plotted in Fig. 2b.

C. FBGA Algorithm

As depicted in Fig. 3, the proposed FBGA method solves the problem (1) using the following inputs:

1) The path to be driven, in the form of vectors of curvilinear abscissas $\boldsymbol{s} = [s_1, \ldots, s_N]$ and corresponding curvatures $\boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_N]$.
2) The initial speed v_{ini} and the top speed $v_{\max}$.
3) The g-g-v acceleration constraints, expressed by the arbitrary function $\Gamma_y^{\pm}(v_x)$ and $\Gamma_x^{\pm}(a_y, v_x)$ in (1c)-(1d), which can represent any shape of the g-g-v envelope.

FBGA outputs arrays of speed and acceleration profiles $\{\boldsymbol{v}_x, \boldsymbol{a}_x, \boldsymbol{a}_y\}$[2], and the maneuver time T. Algorithm 2 and Fig. 3 provide an overview of the FBGA method. For each of the $N-1$ path segments, Algorithm 2 performs the following three steps: calculation of the maximum saturated speed given the lateral acceleration bounds (line 3), forward pass (line 4), and backward pass (line 5). The total maneuver time T is obtained as the sum of the individual segment durations (line 7), each computed via (4). The lateral acceleration vector $\boldsymbol{a}_y$ is finally given by the Hadamard product of the curvature $\boldsymbol{\kappa}$ and the squared velocity profile $\boldsymbol{v}_x^2$ (line 7).

Fig. 4 illustrates the three main phases of the FBGA on an example scenario. These phases, detailed in Algorithms 3-5, share a custom root-finding routine, named SOLVE, which computes function zeros with a custom non-derivative method.

[1] Assuming a constant longitudinal acceleration within short path segments is consistent with direct optimal control methods, where the control inputs are typically held constant in discretized mesh intervals.

[2] In post-processing, FBGA returns $v_x(t)$, $a_x(t)$, and $a_y(t)$ $\forall t \in [0, T]$.

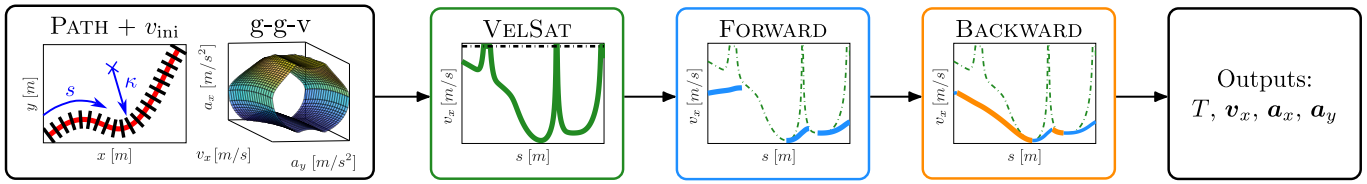


Fig. 3. Main phases of our FBGA, described in Algorithm 2. FBGA takes as input the path (vectors of curvilinear abscissas $\boldsymbol{s}$ and curvatures $\boldsymbol{\kappa}$), the initial speed v_{ini}, and the g-g-v acceleration constraints. It outputs vectors with the speed and acceleration profiles $\{\boldsymbol{v}_x, \boldsymbol{a}_x, \boldsymbol{a}_y\}$, and the maneuver time T.

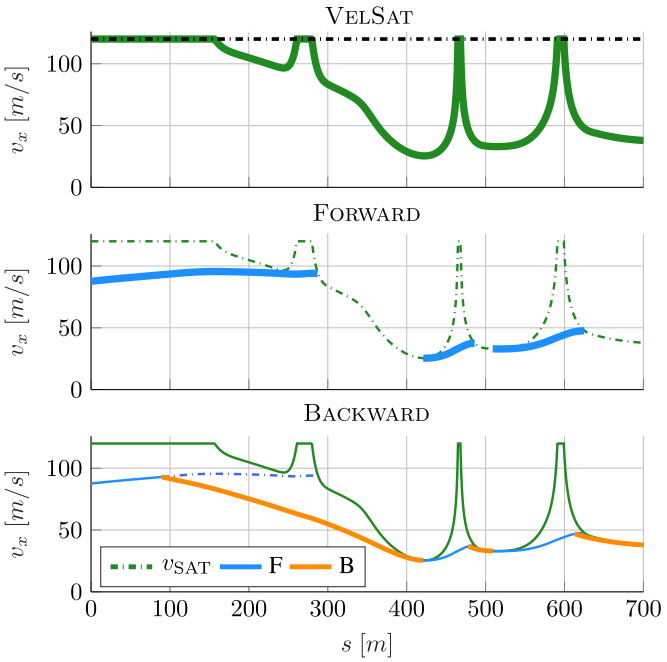


Fig. 4. Phases of the FBGA method on an example scenario: maximum speed given the lateral acceleration bounds (top plot), forward pass (center plot, with blue lines when successful), and backward pass (bottom plot, with orange lines when editing invalid forward pass segments).

Optimality proofs of our FBGA with generic acceleration constraints are an open research question. However, optimality proofs exist for convex or rectangular g-g shapes and linear dynamics (in the control) [2], [30].

Algorithm 2 FBGA method

1: **Input:** $\boldsymbol{s}$, $\boldsymbol{\kappa}$, v_{ini}, $v_{\max}$, Γ_y^-, Γ_y^+, Γ_x^-, Γ_x^+
2: **Output:** $\boldsymbol{v}_x$, $\boldsymbol{a}_x$, $\boldsymbol{a}_y$, T
3: $\boldsymbol{v}_{\text{sat}} \leftarrow$ VELSAT($\boldsymbol{s}$, $\boldsymbol{\kappa}$, $v_{\max}$, Γ_y^-, Γ_y^+);
4: $\boldsymbol{v}_x, \boldsymbol{a}_x \leftarrow \mathrm{F}(\boldsymbol{s}, \boldsymbol{\kappa}, v_{\text{ini}}, \Gamma_y^-, \Gamma_y^+, \Gamma_x^-, \Gamma_x^+, \boldsymbol{v}_{\text{sat}})$;
5: $\boldsymbol{v}_x, \boldsymbol{a}_x \leftarrow \mathrm{B}(\boldsymbol{s}, \boldsymbol{\kappa}, \Gamma_y^-, \Gamma_y^+, \Gamma_x^-, \Gamma_x^+, \boldsymbol{v}_{\text{sat}}, \boldsymbol{v}_x, \boldsymbol{a}_x)$;
6: $N \leftarrow \text{size}(\boldsymbol{s})$;
7: $T \leftarrow \sum_{i=1}^{N-1} \frac{-\boldsymbol{v}_{x_i} + \sqrt{2\boldsymbol{a}_{x_i}(\boldsymbol{s}_{i+1} - \boldsymbol{s}_i) + \boldsymbol{v}_{x_i}^2}}{\boldsymbol{a}_{x_i}}$; $\boldsymbol{a}_y \leftarrow \boldsymbol{\kappa} \odot \boldsymbol{v}_x^2$;

1) Maximum Speed given the Lateral Acceleration Bounds: As shown in the top plot of Fig. 4, FBGA first computes a vector of saturated velocities $\boldsymbol{v}_{\text{sat}}$, which are the maximum speeds to comply with the lateral acceleration bounds $\Gamma_y^{\pm}(v_x)$. This calculation is done by the VELSAT function, provided in Algorithm 3. The function loops over the path points (line 4), and computes the corresponding saturated speed $\bar{v}$ by finding the zero of $H(v_x) = \kappa v_x^2 - \Gamma_y^{\pm}(v_x)$ (line 8). If no root is found, $\bar{v}$ is set to the top speed $v_{\max}$ (line 9). The function stores $\bar{v}$ in the vector $\boldsymbol{v}_{\text{sat}}$ (line 10), which is then used in the forward and backward passes to compute the final speed and acceleration profiles.

Algorithm 3 VELSAT function

1: **Input:** $\boldsymbol{s}$, $\boldsymbol{\kappa}$, $v_{\max}$, $\Gamma_y^-(v_x)$, $\Gamma_y^+(v_x)$
2: **Output:** $\boldsymbol{v}_{\text{sat}}$
3: $N \leftarrow \text{size}(\boldsymbol{s})$;
4: **for all** $i \in [1, N]$ **do**
5: $\quad \kappa \leftarrow \boldsymbol{\kappa}(i)$; $v_x \leftarrow \boldsymbol{v}_x(i)$;
6: $\quad$ **if** $\kappa \geq 0$ **then** $H(V) := \kappa V^2 - \Gamma_y^+(V)$;
7: $\quad$ **else** $H(V) := \kappa V^2 - \Gamma_y^-(V)$;
8: $\quad \bar{v} \leftarrow$ SOLVE$(H(v_x) = 0, [0, v_{\max}])$;
9: $\quad$ **if** $\bar{v} = \varnothing$ **then** $\bar{v} \leftarrow v_{\max}$;
10: $\quad \boldsymbol{v}_{\text{sat}}(i) \leftarrow \bar{v}$;

2) Forward Pass: The middle plot of Fig. 4 shows the forward pass (F) in Algorithm 4, which computes the maximum feasible longitudinal accelerations satisfying the g-g-v constraints (1c)-(1d). For each path segment, the algorithm extracts the initial/final curvatures $\{\kappa_0, \kappa_1\}$ and the initial velocity v_0 (line 6). It then computes the acceleration bounds $\{a_{x_{\min}}, a_{x_{\max}}\}$ (line 8) and checks whether applying $a_{x_0} = a_{x_{\max}}$ satisfies the g-g-v constraints at the segment end (lines 9-10). If not, it solves a root-finding problem to find the maximum feasible a_{x_0} that brings the end state to the g-g-v boundary (line 11). The final velocity v_1 is then set to the minimum of $\boldsymbol{v}_{\text{sat}}$ and the velocity from forward integration of a_{x_0} via (3) (line 13). This v_1 becomes the initial velocity for the next segment (line 14). If instead no feasible a_{x_0} is found at line 11, a_{x_0} is set to $\varnothing$, and the next segment starts from the saturated velocity $\boldsymbol{v}_{\text{sat}}$ (line 12). The function finally returns the vectors of velocity $\boldsymbol{v}_x$ and acceleration $\boldsymbol{a}_x$.

3) Backward Pass: The backward pass, described in Algorithm 5 and Fig. 4 (bottom plot), computes the braking manoeuvers for all path segments where the forward pass did not yield valid solutions. It returns the final velocity and acceleration vectors, $\boldsymbol{v}_x$ and $\boldsymbol{a}_x$. The algorithm iterates through the segments in reverse order (starting from the end, line 4) and proceeds according to the following three cases.

In the first case, if the forward solution is still valid—*i.e.*, the final segment velocity v_1 is reachable from the initial state v_0 using the acceleration a_{x_0}—the algorithm proceeds to the next segment without modification (lines 7-8).

In the second case, the initial velocity v_0 is still reachable from the updated final speed v_1, within the acceleration bounds (line 10). Here, the algorithm computes the average acceleration $a_{x_{\text{avg}}}$ (line 11) and checks whether it lies within the g-g-v envelope at both segment endpoints (lines 15-17). If

Algorithm 4 FORWARD (F) function

1: **Input:** $\boldsymbol{s}$, $\boldsymbol{\kappa}$, v_{ini}, Γ_y^-, Γ_y^+, Γ_x^-, Γ_x^+, $\boldsymbol{v}_{\text{sat}}$
2: **Output:** $\boldsymbol{v}_x, \boldsymbol{a}_x$
3: $\boldsymbol{v}_x, \boldsymbol{a}_x \leftarrow \varnothing$;
4: $\boldsymbol{v}_x(1) \leftarrow v_{\text{ini}}$; $v_0 \leftarrow v_{\text{ini}}$; $N \leftarrow \text{size}(\boldsymbol{s})$;
5: **for all** $i \in [1, N-1]$ **do**
6: $\quad \kappa_0 \leftarrow \boldsymbol{\kappa}(i)$; $\kappa_1 \leftarrow \boldsymbol{\kappa}(i+1)$; $L \leftarrow \boldsymbol{s}(i+1) - \boldsymbol{s}(i)$;
7: $\quad a_{y_{\text{clip}}} \leftarrow \text{CLIP}(\kappa_0 v_0^2, \Gamma_y^-(v_0), \Gamma_y^+(v_0))$;
8: $\quad a_{x_{\min}} \leftarrow \Gamma^-(a_{y_{\text{clip}}}, v_0)$; $\quad a_{x_{\max}} \leftarrow \Gamma^+(a_{y_{\text{clip}}}, v_0)$;
9: $\quad G(A) := \text{D}^{\pm}\Big(A, \underbrace{\kappa_1\left(2LA + v_0^2\right)}_{a_{y_1}}, \underbrace{\left(2LA + v_0^2\right)^{1/2}}_{v_1}\Big)$;
10: $\quad$ **if** $G(a_{x_{\max}}) \le 0$ **then** $a_{x_0} \leftarrow a_{x_{\max}}$;
11: $\quad$ **else** $a_{x_0} \leftarrow \text{SOLVE}(G(a_{x_0}) = 0, [a_{x_{\min}}, a_{x_{\max}}])$;
12: $\quad$ **if** $a_{x_0} = \varnothing$ **then** $v_1 \leftarrow \boldsymbol{v}_{\text{sat}}(i+1)$;
13: $\quad$ **else** $v_1 \leftarrow \min\left(\sqrt{2La_{x_0} + v_0^2}, \boldsymbol{v}_{\text{sat}}(i+1)\right)$;
14: $\quad \boldsymbol{a}_x(i) \leftarrow a_{x_0}$; $\quad \boldsymbol{v}_x(i+1) \leftarrow v_1$; $\quad v_0 \leftarrow v_1$;

Algorithm 5 BACKWARD (B) function

1: **Input:** $\boldsymbol{s}$, $\boldsymbol{\kappa}$, Γ_y^-, Γ_y^+, Γ_x^-, Γ_x^+, $\boldsymbol{v}_{\text{sat}}$, $\boldsymbol{v}_x$, $\boldsymbol{a}_x$
2: **Output:** $\boldsymbol{v}_x$, $\boldsymbol{a}_x$
3: $N \leftarrow \text{size}(\boldsymbol{s})$;
4: **for all** $i \in [N, 2]$ **do**
5: $\quad \kappa_0 \leftarrow \boldsymbol{\kappa}(i-1)$; $\kappa_1 \leftarrow \boldsymbol{\kappa}(i)$; $v_0 \leftarrow \boldsymbol{v}_x(i-1)$; $v_1 \leftarrow \boldsymbol{v}_x(i)$;
6: $\quad a_{x_0} \leftarrow \boldsymbol{a}_x(i-1)$; $L \leftarrow \boldsymbol{s}(i) - \boldsymbol{s}(i-1)$;
7: $\quad \text{is\_valid\_forward} \leftarrow a_{x_0} \neq \varnothing$ **and** $\left(v_0^2 + 2La_{x_0}\right)^{1/2} = v_1$;
8: $\quad$ **if** is_valid_forward **then continue**
9: $\quad a_{y_{\text{clip}}} \leftarrow \text{CLIP}(\kappa_1 v_1^2, \Gamma_y^-(v_1), \Gamma_y^+(v_1))$;
10: $\quad a_{x_{\min}} \leftarrow \Gamma^-(a_{y_{\text{clip}}}, v_1)$; $\quad a_{x_{\max}} \leftarrow \Gamma^+(a_{y_{\text{clip}}}, v_1)$;
11: $\quad a_{x_{\text{avg}}} \leftarrow (v_1^2 - v_0^2)/(2L)$;
12: $\quad v_{0_{\text{REACH}_{\max}}} \leftarrow \left(v_1^2 - 2La_{x_{\min}}\right)^{1/2}$;
13: $\quad v_{0_{\text{REACH}_{\min}}} \leftarrow \left(v_1^2 - 2La_{x_{\max}}\right)^{1/2}$;
14: $\quad \text{is\_valid\_}v_0 \leftarrow (v_0 \le v_{0_{\text{REACH}_{\max}}})$ **and** $(v_0 \ge v_{0_{\text{REACH}_{\min}}})$;
15: $\quad \text{is\_valid\_}a_x \leftarrow (a_{x_{\text{avg}}} \le a_{x_{\max}})$ **and** $(a_{x_{\text{avg}}} \ge a_{x_{\min}})$;
16: $\quad G(A) := \text{D}^{\pm}\Big(A, \underbrace{\kappa_0(v_1^2 - 2LA)}_{a_{y_0}}, \underbrace{\left(v_1^2 - 2LA\right)^{1/2}}_{v_0}\Big)$
17: $\quad$ **if** is_valid_a_x **and** is_valid_v_0 **and** $G(a_{x_{\text{avg}}}) \le 0$ **then**
18: $\quad\quad \boldsymbol{a}_x(i-1) \leftarrow a_{x_{\text{avg}}}$;
19: $\quad$ **else**
20: $\quad\quad$ **if** $G(a_{x_{\min}}) \le 0$ **then** $a_{x_0} \leftarrow a_{x_{\min}}$;
21: $\quad\quad$ **else** $a_{x_0} \leftarrow \text{SOLVE}(G(a_{x_0}) = 0, [a_{x_{\min}}, a_{x_{\max}}])$;
22: $\quad\quad \boldsymbol{v}_x(i-1) \leftarrow \left(v_1^2 - 2La_{x_0}\right)^{1/2}$; $\boldsymbol{a}_x(i-1) \leftarrow a_{x_0}$;

so, $a_{x_{\text{avg}}}$ is accepted (line 18), and the algorithm continues to the next segment.

In the final case, either $a_{x_{\text{avg}}}$ is infeasible or v_0 is no longer reachable from v_1 (line 19). The algorithm then applies the maximum deceleration $a_{x_{\min}}$, if it remains within the g-g-v envelope (line 20), or computes the deceleration a_{x_0} that brings v_0 to the g-g-v envelope boundary (line 21). Finally, the updated velocity and acceleration are stored (line 22). The backward pass always yields a feasible solution, and overrides the requested initial velocity v_{ini} if it is infeasible.

III. RESULTS

A. Simulation Setup, Implementation and Benchmarking

We validate the proposed FBGA with the following simulation setup, implementation code, and benchmarking.

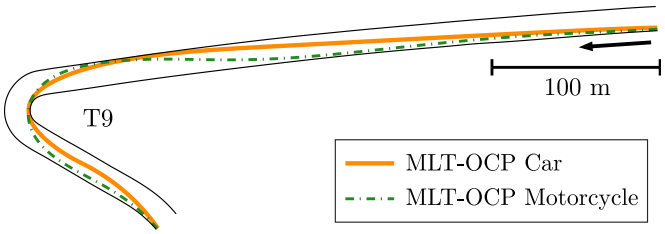


Fig. 5. Comparison of the MLT-OCP maneuvers for the racing car and motorcycle models of this paper, on the turn n.9 of the Sepang circuit.

1) Vehicle Types and Racetracks: FBGA could be applied for velocity planning with different robotic systems (Sec. I), and in this paper, we focus on the challenging case of racing vehicles, whose g-g-v acceleration envelopes can have complex non-convex shapes. We consider two vehicle types: a racing car, whose g-g-v envelope is taken from [10], and a racing motorcycle, whose g-g-v is derived from [28]. For each vehicle type, we apply FBGA on five racetracks: Catalunya, Valencia, Misano, Sepang, and Palm Beach.

2) Implementation: Our code is implemented in `C++` and is available open source at https://github.com/DRIVEWISE/FBGA. Our numerical experiments are run on a laptop with an M2 Max Apple Silicon chip.

3) Benchmarking: We benchmark FBGA against $\text{OCP}_{\text{bench}}$, which is a numerical solution of the same fixed-path problem (1), obtained using the state-of-the-art indirect optimal control solver Pins [18].

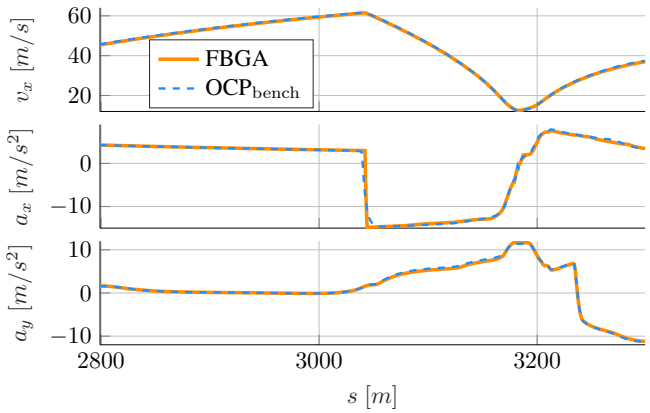
B. Preprocessing: Path Generation

Our FBGA computes the time-optimal speed profile along a given path. For each racetrack and vehicle (Sec. III-A1), the path is generated by solving a free-trajectory minimum-lap-time OCP (MLT-OCP), using the formulation in [10] and the solver Pins [18]. The MLT-OCP uses a point-mass model with longitudinal and lateral dynamics, and enforces the same g-g-v constraints as in (1). From the resulting minimum-time lap, we extract the curvilinear abscissas $\boldsymbol{s}$ and curvatures $\boldsymbol{\kappa}$, used for both our FBGA and $\text{OCP}_{\text{bench}}$. We remark that the path can be obtained with other methods according to the application (*e.g.*, from experimental data, or geometrical optimization), and does not need to be generated by an OCP.

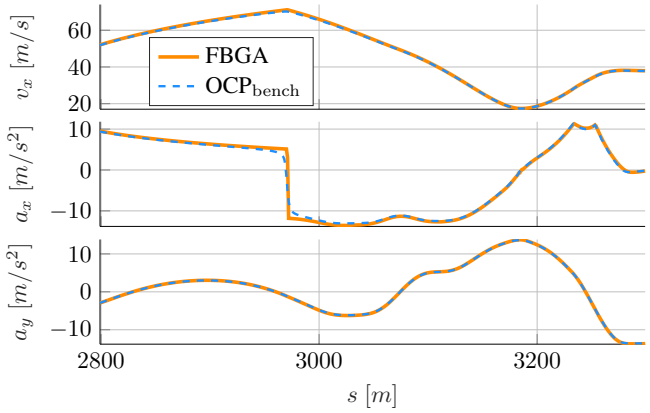
C. Maneuver Analysis

We now compare the results of our FBGA with the benchmark $\text{OCP}_{\text{bench}}$, on the five circuits and the race car and motorcycle models described in Sec. III-A1. The two vehicles have markedly different g-g-v acceleration constraints, shown in Fig. 2a (motorcycle) and Fig. 7 (car). The motorcycle's envelope is notably non-convex due to the front and rear wheel lift during acceleration and braking, which limits the peak longitudinal accelerations [14], [28].

Fig. 5-6 shows the minimum-time maneuvers (path, speed, and accelerations) at turn 9 of the Sepang circuit, for both the car and motorcycle models. The paths in Fig. 5 are computed via the MLT-OCP preprocessing step, which accounts for

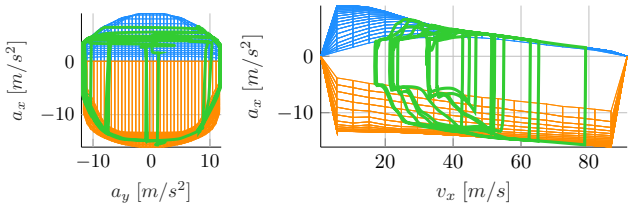


(a) Results for the racing car model.



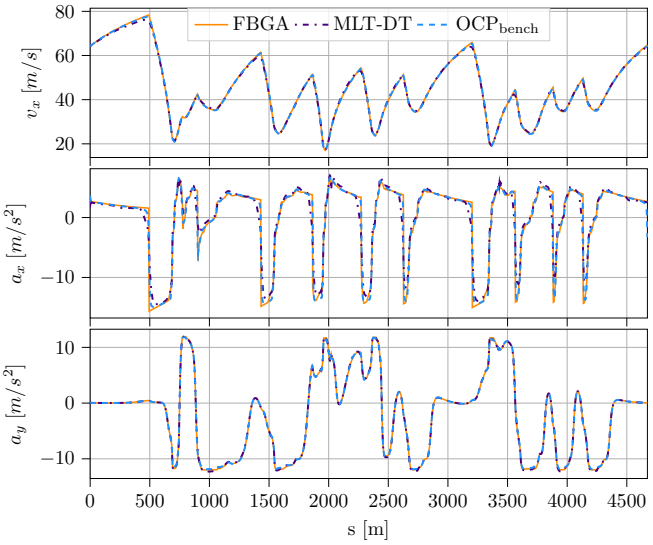
(b) Results for the racing motorcycle model.

Fig. 6. Velocity, longitudinal and lateral acceleration profiles of our FBGA and the benchmark OCP$_{\text{bench}}$, for a racing car (a) and motorcycle (b) at turn 9 of the Sepang circuit (Fig. 5).



(a) Top view (b) Lateral view

Fig. 7. g-g-v envelope (blue-orange lines) for the race car model, and solution computed by our FBGA (green line) on the Catalunya circuit.



FBGA	MLT-DT	OCP$_{\text{bench}}$
113.96 s	114.05 s	114.16 s

Fig. 8. Comparing our FBGA, the OCP$_{\text{bench}}$ and the MLT-DT OCP, which is solved with a double-track (DT) vehicle model, on the Catalunya circuit.

the g-g-v constraints and vehicle transient dynamics (Sec. III-B). Due to their distinct g-g-v envelopes, the resulting time-optimal paths differ significantly: the car brakes on a straight before the turn, while the motorcycle brakes along a curved path to reach higher longitudinal accelerations by avoiding rear-wheel lift. Fig. 6 plots the corresponding speed and acceleration profiles. The braking and acceleration phases vary notably between the car (Fig. 6a) and motorcycle (Fig. 6b), again reflecting their differences in the g-g-v diagrams. FBGA's results closely match those of OCP$_{\text{bench}}$, with minor discrepancies in transition regions where FBGA switches instantaneously between acceleration and braking. Conversely, OCP$_{\text{bench}}$ filters the control a_x with a fast first-order dynamics, to prevent numerical oscillations of the solution around the g-g-v boundaries. Fig. 7a-7b plots the g-g-v diagram of the racing car model, together with the solution computed by FBGA (green line) on the Catalunya circuit. The figures highlight that the FBGA solution moves along the g-g-v envelope, while satisfying the constraints up to the desired solver's precision. To further validate our FBGA, we compare it with a minimum-lap-time OCP solved using a high-fidelity double-track vehicle model (MLT-DT), formulated as in [31]. This model captures high-speed race car dynamics, including a Pacejka MF5.2 tire model, real engine torque curves, and a limited-slip differential. We apply FBGA on the path generated by MLT-DT on the Catalunya circuit, and the g-g-v envelope is identified following [10]. As shown in Fig. 8, our approach closely matches both MLT-DT and OCP$_{\text{bench}}$, with minor deviations only during acceleration-braking transitions, where jerk is limited in the double-track model. Despite this, the lap time difference between FBGA and MLT-DT is only 73 ms, relatively small given the model complexity and the different optimization methods.

D. Lap Times

Table I compares the lap times of FBGA and OCP$_{\text{bench}}$ for the race car and motorcycle models, across five racetracks. The lap times of FBGA and OCP$_{\text{bench}}$ are very close, with differences ranging from 0.094 to 0.449 s (0.11% – 0.36% of the total lap time). These deviations are comparable to those observed among professional race drivers [32]. FBGA slightly underestimates the lap times due to instantaneous traction-braking transitions, while professional drivers have a limited actuation rate. Nonetheless, this simplification boosts the computational efficiency while still providing accurate lap time estimates. Indeed, FBGA is designed for high-level

Circuit (length)	N. mesh points	Vehicle	Lap time [s]			CPU time [ms]	
			OCP_{bench}	**FBGA** (ours)	Δ (FBGA $-$ OCP_{bench})	OCP_{bench}	**FBGA** (ours)
Catalunya (4.66 km)	4660	Car	112.461	112.204	-0.257	8017.15	9.86
		Motorcycle	105.381	104.999	-0.382	693.91	3.31
Valencia (4.00 km)	4000	Car	104.742	104.520	-0.222	7411.99	8.85
		Motorcycle	96.752	96.434	-0.318	830.82	3.40
Misano (4.16 km)	4160	Car	107.740	107.451	-0.289	6805.87	9.66
		Motorcycle	98.814	98.497	-0.317	602.19	3.37
Sepang (5.52 km)	5520	Car	135.480	135.086	-0.394	8239.47	11.44
		Motorcycle	125.483	125.034	-0.449	1282.58	4.30
Palm Beach (3.17 km)	3170	Car	79.963	79.869	-0.094	4819.43	6.38
		Motorcycle	74.790	74.537	-0.253	706.43	2.39

TABLE I

COMPARISON OF LAP TIMES AND CPU TIMES BETWEEN OUR FBGA AND THE BENCHMARK OCP_{bench}, BOTH SOLVED WITH THE SAME MESH, ON AN M2 MAX APPLE SILICON CHIP. THE TABLE SHOWS THE RESULTS FOR THE RACING CAR AND MOTORCYCLE MODELS ON 5 RACETRACKS.

trajectory planning [1], [4], where the computational time is key, and the acceleration rate bounds are typically handled by downstream motion controllers.

E. Computational Times

1) Full-Lap Computational Times: Table I reports the computational (CPU) times of the FBGA and OCP_{bench} methods when computing the speed profiles over full laps. The CPU times of FBGA range in $2.39-11.44$ ms, and they are $2-3$ orders of magnitude lower[3] than OCP_{bench}. The CPU times of both methods are influenced by the path curvature values[4] and the implementation of the g-g-v constraints (1c)-(1d). For the car, these constraints are modeled with bilinear splines, while for the motorcycle, we use analytical functions, which are computationally cheaper.

These results show that our FBGA algorithm supports any formulation of the g-g-v acceleration constraints, independently of their complexity, without the need for convexity or differentiability assumptions.

2) Short-Horizon Computational Times: We evaluate the performance of our FBGA for short-horizon planning over a 300 m segment, covering the first two corners of the Catalunya circuit (Fig. 1). This horizon length is typically adopted in sampling-based planners for high-speed autonomous racing [1], [8]. Table II reports the results for different mesh sizes used to discretize the path. It compares our FBGA with OCP_{bench}, which uses a fixed number of 300 mesh points. With the same mesh, FBGA is computationally faster by a factor of 550 (0.177 ms vs 97.533 ms), with only a 0.19% difference in maneuver time (8.7765 s vs 8.7936 s). Reducing the FBGA's mesh size to 100 points further increases the speed-up to 3 orders of magnitude, with negligible impact on the maneuver time. These analyses show that our FBGA is robust to the mesh resolution and suitable for real-time planning of multiple time-optimal maneuvers, as required in sampling-based planners for autonomous racing [1].

F. Sensitivity Analysis

Fig. 9 shows the sensitivity of our FBGA algorithm to the number of path discretization segments, analyzing both

[3]As a remark, the CPU times are obtained with single runs of our FBGA, without averaging over multiple runs.

[4]A CPU time analysis of our FBGA shows that roughly 20% of the time is spent in the VELSAT function, 60% in the FORWARD and 20% in the BACKWARD functions. These ratios highly depend on the provided path.

OCP_{bench}		FBGA (ours)		
T [s]	**CPU [ms]**	**N. points**	**T [s]**	**CPU [ms]**
8.7936	97.533	300	8.7765	0.177
		200	8.7830	0.131
		100	8.8029	0.062

TABLE II

COMPARING THE MANEUVER TIME T AND CPU TIMES OF OUR FBGA AND OCP_{bench}, ON A 300 M HORIZON. OCP_{bench} USES 300 MESH POINTS, WHILE FBGA IS SOLVED WITH A VARIABLE MESH.

the lap time (Catalunya circuit) and the total CPU time. As expected, the CPU time scales linearly with the number of segments[5], since the algorithm processes each segment sequentially. Minor oscillations appear at coarse resolutions, where segment-level curvature variations can affect the number of iterations of our root-finding procedure. In contrast, the lap time decreases hyperbolically, and changes by only 0.07 s (0.06%) when increasing the segment count by 10 times.

These results confirm that FBGA is robust to the mesh resolution, and can be applied to different use cases: fine meshes maximize the solution accuracy and fully exploit the acceleration limits, while coarse meshes offer significant speed-ups with a negligible impact on the maneuver time.

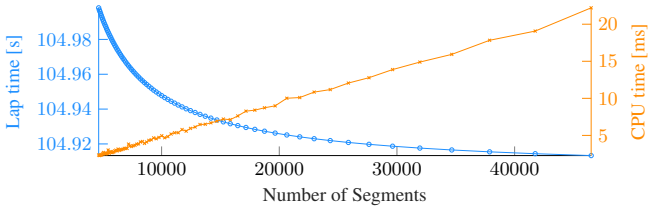


Fig. 9. Lap times and CPU times of FBGA as a function of the number of mesh segments, with the racing motorcycle model on the Catalunya circuit.

IV. CONCLUSIONS

We introduced FBGA, a new forward-backward method for real-time computation of time-optimal velocity profiles along prescribed paths, supporting arbitrary acceleration constraints. FBGA discretizes the path into short segments and performs forward and backward integrations to maximize the velocity while satisfying user-defined acceleration limits.

FBGA was validated in the context of autonomous racing,

[5]Our results are obtained by randomizing the number of segments at each run, to avoid caching optimization of subsequent calls of the same functions.

with race car and motorcycle models on five circuits. It produced velocity and acceleration profiles closely matching those of a benchmark optimal control problem ($\text{OCP}_{\text{bench}}$), with lap time deviations below 0.449 s (0.36%) and computational speed-ups of up to three orders of magnitude. These results held across tracks and vehicle types, with no assumptions on convexity, smoothness, or differentiability of the acceleration constraints. In short-horizon planning, FBGA achieved maneuver times within 0.19% of $\text{OCP}_{\text{bench}}$, while maintaining a three orders of magnitude speed-up and a high solution quality even with coarse path discretization.

Overall, FBGA can be applied to accurate full-lap velocity optimization, warm-starting of complex OCPs, and real-time short-horizon planning. It is expected to be an effective building block for multi-query time-optimal trajectory planning in dynamic environments, extending sampling-based planners [4], [8] beyond conservative box acceleration constraints, and unlocking new capabilities for autonomous racing.

Future work will enhance the FBGA method to address its limitations and extend its scope. We plan to improve the feasibility of the acceleration-braking transitions by extending Algorithms 4-5 to deal with jerk limits. Also, formal guarantees on the solution's optimality and convergence will be better investigated. Finally, application-oriented papers will integrate FBGA into multi-query trajectory planners for diverse fields of robotics, including vehicles and drone flight.

REFERENCES

[1] L. Ögretmen, M. Rowold, A. Langmann, and B. Lohmann, "Sampling-based motion planning with online racing line generation for autonomous driving on three-dimensional race tracks," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024, pp. 811–818.

[2] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles," *Automatica*, vol. 86, pp. 18–28, 2017.

[3] M. Piccinini, S. Gottschalk, M. Gerdts, and F. Biral, "Computationally efficient minimum-time motion primitives for vehicle trajectory planning," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 5, pp. 642–655, 2024.

[4] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

[5] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[6] M. Benko Loknar, G. Klančar, and S. Blažič, "Minimum-time trajectory generation for wheeled mobile systems using bézier curves with constraints on velocity, acceleration and jerk," *Sensors*, vol. 23, no. 4, 2023.

[7] L. Consolini, M. Locatelli, and A. Minari, "A sequential algorithm for jerk limited speed planning," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3192–3209, 2022.

[8] M. Piazza, M. Piccinini, S. Taddei, and F. Biral, "MPTree: A sampling-based vehicle motion planner for real-time obstacle avoidance," *IFAC-PapersOnLine*, vol. 58, no. 10, pp. 146–153, 2024.

[9] R. Trauth, K. Moller, G. Würsching, and J. Betz, "Frenetix: A high-performance and modular motion planning framework for autonomous driving," *IEEE Access*, 2024.

[10] M. Piccinini, S. Taddei, E. Pagot, E. Bertolazzi, and F. Biral, "How optimal is the minimum-time manoeuvre of an artificial race driver?" *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–28, 2024.

[11] M. Veneri and M. Massaro, "A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles," *Vehicle System Dynamics*, vol. 58, no. 6, pp. 933–954, 2020.

[12] F. Rodegher, S. Lovato, and M. Massaro, "The effect of load-dependent tyre friction and aerodynamic cross-flow forces on the gg maps and minimum-lap-time of motorcycles," *Vehicle System Dynamics*, pp. 1–19, 2024.

[13] D. Limebeer, M. Massaro, and A. Dollar, "Asymmetric vehicle performance assessment using gg diagrams," *Vehicle System Dynamics*, pp. 1–27, 2024.

[14] M. Piccinini, S. Taddei, M. Piazza, and F. Biral, "Impacts of ggv constraints formulations on online minimum-time vehicle trajectory planning," *IFAC-PapersOnLine*, vol. 58, no. 10, pp. 87–93, 2024.

[15] F. Werner, S. Sagmeister, M. Piccinini, and J. Betz, "A quasi-steady-state black box simulation approach for the generation of g-g-g-v diagrams," in *2025 IEEE Intelligent Vehicles Symposium (IV)*, 2025, pp. 2503–2509.

[16] D. Brayshaw and M. Harrison, "A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location," *Proceedings of The Institution of Mechanical Engineers Part D-journal of Automobile Engineering*, vol. 219, pp. 725–739, 06 2005.

[17] M. Rowold, L. Ögretmen, U. Kasolowsky, and B. Lohmann, "Online time-optimal trajectory planning on three-dimensional race tracks," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–8.

[18] F. Biral, E. Bertolazzi, and P. Bosetti, "Notes on numerical methods for solving optimal control problems," *IEEJ Journal of Industry Applications*, vol. 5, no. 2, pp. 154–166, 2016.

[19] R. Lot and N. D. Bianco, "Lap time optimisation of a racing go-kart," *Vehicle System Dynamics*, vol. 54, no. 2, pp. 210–230, 2016.

[20] M. Piccinini, S. Taddei, J. Betz, and F. Biral, "Kineto-dynamical planning and accurate execution of minimum-time maneuvers on three-dimensional circuits," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 1–7.

[21] J. Biniewicz and M. Pyrz, "A quasi-steady-state minimum lap time simulation of race motorcycles using experimental data," *Vehicle System Dynamics*, vol. 62, no. 2, pp. 372–394, 2024.

[22] N. R. Kapania, J. Subosits, and J. Christian Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091005, 2016.

[23] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, 2020.

[24] B. Lenzo and V. Rossi, "A simple mono-dimensional approach for lap time optimisation," *Applied Sciences*, vol. 10, no. 4, p. 1498, 2020.

[25] F. Cabassi, L. Consolini, and M. Locatelli, "Time-optimal velocity planning by a bound-tightening technique," *Computational Optimization and Applications*, vol. 70, no. 1, pp. 61–90, May 2018.

[26] L. Consolini and M. Locatelli, "Is time-optimal speed planning under jerk constraints a convex problem?" *Automatica*, vol. 169, p. 111864, 2024.

[27] A. Langmann, L. Ögretmen, F. Werner, and J. Betz, "Online velocity profile generation and tracking for sampling-based local planning algorithms in autonomous racing environments," in *2025 IEEE Intelligent Vehicles Symposium (IV)*, 2025, pp. 632–639.

[28] F. Biral and R. Lot, "An interpretative model of g-g diagrams of racing motorcycle," in *Proceedings of the 3rd ICMEM International Conference on Mechanical Engineering and Mechanics*, Beijing, China, 2009.

[29] T. Novi, A. Liniger, R. Capitani, and C. A. and, "Real-time control for at-limit handling driving on a predefined path," *Vehicle System Dynamics*, vol. 58, no. 7, pp. 1007–1036, 2020.

[30] E. Velenis and P. Tsiotras, "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation," *Journal of Optimization Theory and Applications*, vol. 138, no. 2, pp. 275–296, 2008.

[31] E. Pagot, "A study of a robust and accurate framework for minimum-time optimal control of high-performance cars: from coaching professional drivers to autonomous racing," Ph.D. dissertation, University of Trento, Italy, 2023.

[32] J. C. Kegelman, L. K. Harbott, and J. C. Gerdes, "Insights into vehicle trajectories at the handling limits: analysing open data from race car drivers," *Vehicle System Dynamics*, vol. 55, no. 2, pp. 191–207, 2017.