Multibody System Dynamics
https://doi.org/10.1007/s11044-023-09932-6

RESEARCH

A consensus-based alternating direction method of multipliers approach to parallelize large-scale minimum-lap-time problems

L. Bartali[1] · E. Grabovic[1] · M. Gabiccini[1]

Received: 20 April 2023 / Accepted: 1 September 2023

Abstract

Minimum-lap-time planning (MLTP) problems, which entail finding optimal trajectories for race cars on racetracks, have received significant attention in the recent literature. They are commonly addressed as optimal control problems (OCPs) and are numerically discretized using direct collocation methods. Subsequently, they are solved as nonlinear programs (NLPs). The conventional approach to solving MLTP problems is *serial*, whereby the resulting NLP is solved all at once. However, for problems characterized by a large number of variables, distributed optimization algorithms, such as the alternating direction method of multipliers (ADMM), may represent a viable option, especially when multicore CPU architectures are available.

This study presents a consensus-based ADMM approach tailored to solving MLTP problems through a distributed optimization algorithm. The algorithm partitions the problem into smaller subproblems based on different sectors of a track, distributing them among multiple processors. ADMM is then used to ensure consensus among the distributed computational processes. In particular, here the term "consensus" denotes the requirement for each subproblem to achieve mutual agreement across the junction areas. The paper also outlines specific strategies leveraging domain knowledge to improve the convergence of the distributed algorithm. The ADMM approach is validated against the *serial* approach, and numerical results are presented for both single-lap and multilap scenarios. In both cases, the ADMM approach proves superior for problem dimensions of 70k+ variables compared to *serial* methods. In planning scenarios with complex vehicle models on long track horizons, i.e., for problems with 1M+ variables, the efficiency gain of the ADMM approach is substantial, and it becomes the only viable option to maintain computational times within acceptable limits.

Keywords Minimum-lap-time simulation · Large-scale problems · Parallel optimal trajectory planning · Consensus-based ADMM

✉ M. Gabiccini
marco.gabiccini@unipi.it

[1] Dipartimento di Ingegneria Civile e Industriale, Università di Pisa Largo Lucio Lazzarino 1, 56122 Pisa, Italy

1 Introduction

Despite being a classical subject, minimum-lap-time planning (MLTP) still stirs a very lively research activity. This is indeed well documented in the recent extensive survey paper [1]. Here, besides approaches based on quasi-steady-state (QSS) and transient vehicle models, also fundamentals on nonlinear optimal control problems (OCPs), road modeling, and vehicle positioning are well summarized.

In the recent literature the approaches that tackle MLTP as the solution of an OCP seem the most widely adopted. Among these, *direct methods* [2–4], based on multiple shooting or collocation, are usually preferred over *indirect ones*, which typically involve the utilization of Pontryagin's maximum principle, as in [5, 6]. Within the OCP context, the variations proposed mainly consist in the model simplifications tolerated. These can be broadly classified as follows: a priori description of the vehicle's trajectory [7–9] (as opposed to leaving the trajectory free [3, 10]), quasi-steady-state (QSS) models [8, 10, 11] (as opposed to dynamic models [6, 12]), and simplified tyre and aerodynamics models [3, 13] (as opposed to adoption of elaborate tire models [14] and aerodynamic maps [15]). A further computational classification criterion is the particular OCP suite adopted, like MUSCOD-II [16], GPOPS-II [17], ICLOCS2 [18], or CasADi [19] and the specific backend solver, which handles the resulting NLP. This choice is typically restricted among SNOPT [20], IPOPT [21], or WORHP [22].

The common characteristic to all of the above approaches is that the resulting NLP is solved as a *single problem*. This inevitably imposes an upper bound to the combined effect of growing vehicle model complexity and increasing the track length, mainly due to the conceivably high-memory footprint and the large number of variables in the resulting NLP. In the present paper, such standard approaches are designated as *serial* methods. Even if some computational steps in the solution of the NLP can take advantage of multicore CPU architectures, no specific MLTP problem reformulation is devised from the outset to profitably spread the computational load across different processors according to a *divide-and-conquer* strategy.

For problems characterized by the number of variables growing at an unprecedented scale, distributed optimization algorithms seem the only viable solution [23]. In particular, the alternating direction method of multipliers (ADMM) has proved to tackle efficiently, and in a distributed manner, global consensus optimization problems in different research contexts, such as control of microgrid electricity networks and smart grids [24, 25], estimation problem by a network of agents [26], and optimal asset utilization in power market operation [27], to mention only a few.

In the consensus-based ADMM method, the original cost function $f(\boldsymbol{x})$ is split additively as $\sum f_i(\boldsymbol{x}_i)$, where $f_i(\boldsymbol{x}_i)$ pertain to a specific agent (or worker) who retains a local representation $\boldsymbol{x}_i$ of the global variable $\boldsymbol{x}$. Specifically, in a distributed computing environment, the agents represent the CPU-cores performing the optimization in parallel. and the term "consensus" denotes the requirement for each worker to achieve an agreement with its neighbors. One of the nodes, called the master, is responsible for updating the so-called *consensus variable* $\boldsymbol{z}$. Each worker seeks to minimize its local objective (based on its data subset) but is forced by a mechanism to reach a consensus with the other workers. After completing the local optimization, each worker sends its updated local copy $\boldsymbol{x}_i$ to the master. The master, in turn, updates the consensus variable $\boldsymbol{z}$ so that the $\boldsymbol{x}_i$ reach a consensus and then distributes the updated value back to the workers. The process is then repeated until convergence.

Even if convergence proofs of the ADMM algorithm are not available except under very special conditions [28], in practice, ADMM has proved successful in countless scenarios, and clever adjustments have been proposed in the literature to increase its robustness and to speed up its convergence [29–33]. Many studies also investigated the influence on the convergence speed and robustness of relevant parameters such as the topology of the connections between workers [34] and the effect of synchronous/asynchronous flow of information among workers [35].

In the context of trajectory optimization, which is the broad area of our work, there are previous contributions. However, they pertain to different domains like human-Mars entry trajectory for a landing mission [36], multiagent/multirobot trajectory synthesis [37, 38], obstacle avoidance for kinematic vehicles models [39], legged locomotion [40], and multicontact model predictive control for robotic manipulation [41]. However, it is worth noting that, to the best of our knowledge, no prior contribution can be counted among distributed approaches for solving MLTPs with reliable dynamic models from a vehicle engineer's perspective.

In the present paper, in contrast to prior work on MLTP, we focus on the specific structure of MLTPs and exploit it via ADMM [23]. In more detail, the optimal trajectory planning problem on a long track is broken into several segments, each pertaining to a given sector of the track. Each segment becomes a subproblem with a reduced number of variables and is discretized using the *direct collocation* method. The subproblems are solved classically, but *in parallel* and on distinct CPU cores. This ensures reduced memory footprint for each subproblem and increased robustness thanks to the reduced dimensionality. The solutions are iteratively forced to adhere to a single trajectory via properly devised *consensus constraints*, which progressively enforce continuity of state and input variables across segments via an augmented Lagrangian mechanism. This scheme offloads the overall problem complexity by distributing it among segments (and CPU cores) and enables synthesizing optimal trajectories on very long tracks with dense discretizations. It is worth noting that the proposed formulation is amenable to any existing (serial) trajectory optimizer for solving each subproblem.

Finally, it is worth remarking that MLTPs are of great importance in the automotive field. The capability to effectively address the minimum-lap-time challenge equips the automotive industry with the means to systematically assess different vehicle configurations and determine the most efficient option while providing guidelines for drivers. Furthermore, in the motorsport context, MLTP can be used as a powerful tool for investigating the impact of specific vehicle parameters on the driving behavior of the pilot, as evidenced in [3]. Here we want to establish the groundwork for extending these benefits to encompass large-scale problem domains.

2 Vehicle model

In this section, we introduce the dynamic vehicle model that serves as a benchmark to assess the potential of the ADMM methodology in solving MLTP problems. The proposed model is described in detail in [42], whereas here, to ensure clarity, we provide only the main highlights.

While the vehicle model used here has been previously applied in solving an MLTP on the Nordschleife circuit, as documented in [42], our current work introduces an innovative algorithm tailored for parallelizing the solution of extensive MLTPs, particularly in multilap scenarios. In contrast to [42], where the primary focus was on describing the vehicle model, the central objective of this paper is to present an original parallel approach. This approach is highlighted as crucial for enhancing computational efficiency when dealing with large-scale MLTPs.

Although our vehicle model is relatively accurate, its complexity falls in-between a classic double-track and a fully fledged multibody system. By utilizing mathematical tools mostly used in robotics, which are based on Lie-group and Lie-algebra methods [43, 44], and recursive algorithms [45] to express the dynamics, the equations are systematically structured in the form of a serial kinematic chain. In particular, the recursive algorithm described in [42] is employed, which is based upon the *articulated body algorithm* in [45]. Although our model does not possess a huge number of degrees of freedom, the ABA offers us a systematic approach while enjoying an algorithmic complexity of $O(n)$, which scales linearly with the number of degrees of freedom. This leads to a significant reduction in the volume of the algebra during the assembly of the dynamic equations. In contrast, the classical Lagrange equation-based approach, with its complexity of $O(n^3)$, is not considered in this analysis due to its inferior performances.

2.1 Track parameterization

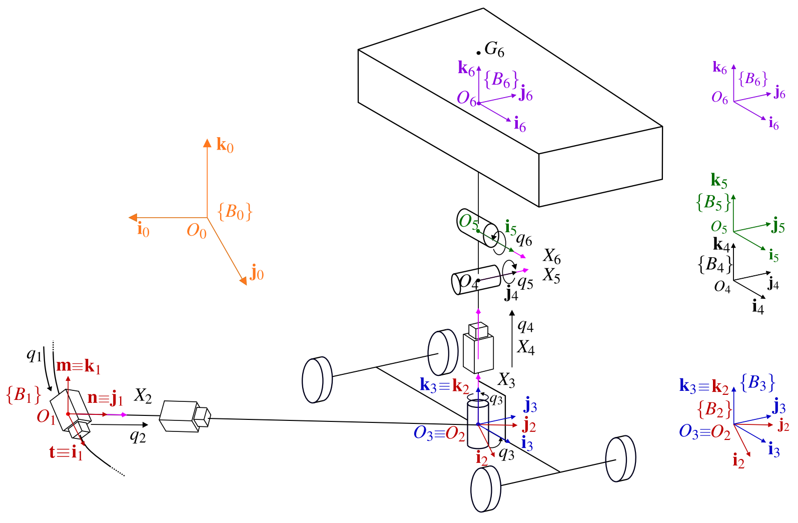
The first ingredient of the proposed model is the track parameterization, which is schematically illustrated in Fig. 1b. The track centerline $\boldsymbol{c}(q_1) = \begin{bmatrix} x_c(q_1) & y_c(q_1) & z_c(q_1) \end{bmatrix}^T \in \mathbb{R}^3$ with $q_1 \in [0, 1]$ is described by a NURBS curve as a function of the parameter q_1, which not necessarily coincides with the curvilinear abscissa. The local frame $\{B_1\}$, which follows the curve, is obtained using a special orthonormal parameterization, which also takes into account the track banking and slope. Specifically, once the tangent vector $\boldsymbol{t}$ is computed, its projected counterpart $\boldsymbol{t}_{\Pi_{\boldsymbol{k}_0}}$ on the (xy)-plane $\Pi_{\boldsymbol{k}_0}$ of the ground-fixed frame $\{B_0\}$ is employed to derive the intermediate normal vector $\boldsymbol{v} = \boldsymbol{k}_0 \times \boldsymbol{t}_{\Pi_{\boldsymbol{k}_0}}$. The final normal and binormal vectors are obtained by rotating $\boldsymbol{k}_0$ and $\boldsymbol{v}$ by the banking angle ν around $\boldsymbol{t}$. This formulation, in contrast to the classical Frenet–Serret formulas, in addition to account for banking, also avoids undesirable direction changes of $\boldsymbol{n}$.

In the context of our model, the pose transformation from $\{B_0\}$ to $\{B_1\}$ represents the effect of a special nonelementary joint, contained in the overall kinematic chain that represents the vehicle. This joint accounts for the sliding motion of the vehicle along the track centerline and has an associate twist (generalized velocity) induced by the curvilinear path constraint.

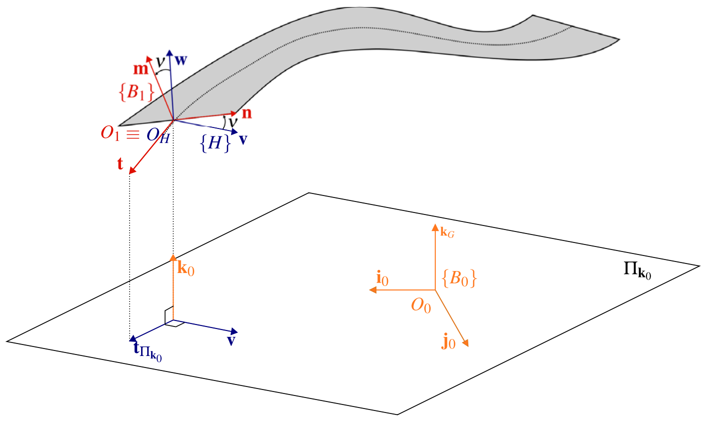
2.2 Vehicle parameterization

The model has six degrees of freedom (DoFs), associated with variables $\boldsymbol{q}$, and is composed of two main parts: a kart-like part, which includes the unsprung masses and the wheels, and a chassis body, which represents the sprung masses. The schematics are shown in Fig. 1a. The first three virtual joints of the kinematic chain, configured as two prismatic joints followed by a revolute joint, constrain the position and orientation of the kart along the track surface. The remaining three joints, consisting of a prismatic joint followed by two revolute joints, connect the kart to the sprung chassis. These last joints also capture the first-order information of the overall suspension geometry as they contain springs and dampers equivalent to bounce k, pitching k_θ, and rolling k_φ stiffness coefficients (and their corresponding damping coefficients c, c_θ, and c_φ).

Overall, we define seven reference frames denoted as $\{B_i\}$ $(i = 0, \ldots, 6)$, attached to the corresponding ith link. The frames $\{B_0\}$ and $\{B_1\}$ are the already mentioned ground-fixed frame and the track reference frame.



(a) Vehicle parameterization



(b) Track parameterization

Fig. 1 Schematics of the vehicle model embedded into the track (Color figure online)

The relative 1-DoF motions (with the exception of the first one) are kinematically described by the unitary twists $X_i \in \mathbb{R}^6$ $(i = 2, \dots, 6)$. These are associated with standard joints (either prismatic or revolute), which are used in the *local product-of-exponentials* (POE) formula [44] to define each relative homogeneous transformation matrix $g_{i-1,i}(\boldsymbol{q}) \in SE(3)$ $(i = 2, \dots, 6)$, between the links of the chain. Specifically, the last five joints represent (i) the lateral displacement q_2 of the vehicle relative to the track centerline, (ii) the *yaw angle* q_3 between the tangent vector of the track centerline and the kart frame $\{B_3\}$, (iii) the vertical displacement q_4 of the chassis relative to the ground, (iv) the *pitch angle* q_5 of the chassis, and (v) the *roll angle* q_6 of the chassis.

Among all the body velocities in the chain, the distal twist of $\{B_3\}$, denoted by $V_3^3 = [v_{3_x}^3\ v_{3_y}^3\ v_{3_z}^3\ \omega_{3_x}^3\ \omega_{3_y}^3\ \omega_{3_z}^3]^T$,[1] is of special importance when computing tire slips and aerodynamic forces. Furthermore, $v_{3_x}^3$ and $v_{3_y}^3$ represent the classical longitudinal and lateral velocities, usually analyzed in telemetry inspections. The calculation of V_3^3 is achieved during a recursive forward propagation of body poses and twists along the chain, which is needed in the subsequent derivation of the dynamic equations.

2.3 Dynamic equations

The dynamic equations are obtained adapting Featherstone's *articulated body algorithm* (ABA) [45], which provides an efficient factorization of the direct dynamics of a kinematic chain. Including also the *reconstruction equations*, the algorithm gives

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}}_v \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_v \\ \text{ABAdyn}(\boldsymbol{q}, \boldsymbol{q}_v, f_{xa}, f_{xb}, \delta, \boldsymbol{w}), \end{bmatrix} = F(\boldsymbol{x}). \tag{1}$$

Here the state is defined as $\boldsymbol{x} = [\boldsymbol{q}^T\ \boldsymbol{q}_v^T]^T$, and $\boldsymbol{q}_v$ and $\dot{\boldsymbol{q}}_v$ represent the joint velocities and accelerations, respectively. By f_{xa} and f_{xb} we denote the total accelerating and braking forces exchanged between the tires and the road, respectively. The steer angle δ is an additional input to the system. To solve the algebraic loops that arise during the calculation of the load transfers, we introduce additional algebraic variables $\boldsymbol{w} \in \mathbb{R}^7$, whose terms are detailed later.

The interactions between the tire and track, globally represented by the resultant wrench $W_3^3 \in \mathbb{R}^6$ (generalized force acting on $\{B_3\}$), can be split into two contributions, the *in-plane* (plane locally tangent to the road) $W_{3_E}^3 = [f_{3_x}^3\ f_{3_y}^3\ 0\ 0\ 0\ m_{3_z}^3]^T$ and *out-of-plane* $W_{3_J}^3 = [0\ 0\ f_{3_z}^3\ m_{3_x}^3\ m_{3_y}^3\ 0]^T$ components. Since the vehicle is parameterized as a serial kinematic chain, the *out-of-plane* components arise, in the ABA framework, from the structural constraints exerted by the third (virtual) joint on $\{B_3\}$, whereas the *in-plane* components act as external "driving" forces. It is worth remarking that $W_3^3 = W_{3_E}^3 + W_{3_J}^3$ has its *in-plane* components evaluated as the resultants of the forces exchanged between the road and each tire, encoded in f_{ij_x}, f_{ij_y}, and f_{ij_z} for the ijth wheel.[2]

Assuming that a rear-wheel drive vehicle is equipped with an open differential, the longitudinal tire forces can be expressed as

$$f_{11_x} = f_{12_x} = \frac{1}{2} f_{xb} k_b, \tag{2}$$

$$f_{21_x} = f_{22_x} = \frac{1}{2} f_{xb}(1 - k_b) + \frac{1}{2} f_{xa}, \tag{3}$$

where, k_b is the braking ratio.

The lateral forces $f_{ij_y}(f_{ij_z})$ are derived from the Pacejka's magic formula [46] as functions of the vertical forces f_{ij_z} on each wheel.

2.4 Additional algebraic equations

The *out-of-plane* components present the issue of giving rise to an algebraic loop: the tire vertical forces f_{ij_z} depend on the overall system variables $\boldsymbol{q}$ and their derivatives $\boldsymbol{q}_v$ and $\dot{\boldsymbol{q}}_v$,

[1] Here subscript refers to the number of the link (body), whereas the superscript denotes the reference frame in which its components are expressed.

[2] $i = 1, 2$ denotes the front/rear axle, whereas $j = 1, 2$ refers to the left/right wheel.

which, in turn, depend on the *in-plane* forces. In the present implementation, we employ the following algebraic variables $\boldsymbol{w}$ to conveniently cut open this loop:

$$\boldsymbol{w} = \begin{bmatrix} f_{11_z} & f_{12_z} & f_{21_z} & f_{22_z} & f^3_{3_x} & f^3_{3_y} & m^3_{3_z} \end{bmatrix}^T, \tag{4}$$

where $f^3_{3_x}$ and $f^3_{3_y}$ are the *in-plane* resultant forces (longitudinal and lateral), and $m^3_{3_z}$ is the corresponding resultant moment along $\boldsymbol{k}_3$ (yaw moment), which are the nonzero components of $W^3_{3_E}$.

Therefore seven additional algebraic equations are required. The first four equations are associated with the vertical loads and are expressed as follows:

$$f_{ij_z} = f_{z_{i0}} + f_{z_{ia}} + \Delta f_z + (-1)^j \Delta f_{z_i}, \tag{5}$$

where the notation used is borrowed from [47]. They are, in order, the *static load*, the *aerodynamic force*, and the two longitudinal and lateral *load transfers*. These terms are linked to the kinematic quantities and structural reaction loads of the third joint, as well as the *in-plane* forces, and can be easily obtained during the unfolding of the ABA algorithm. The remaining three equations expressing the resultant of the *in-plane* components are

$$f^3_{3_x} = (f_{11_x} + f_{12_x})\cos\delta - (f_{11_y} + f_{12_y})\sin\delta + f_{21_x} + f_{22_x}, \tag{6}$$

$$f^3_{3_y} = (f_{11_y} + f_{12_y})\cos\delta + (f_{11_x} + f_{12_x})\sin\delta + f_{21_y} + f_{22_y}, \tag{7}$$

$$m^3_{3_z} = \left[(f_{11_y} + f_{12_y})\cos\delta + (f_{11_x} + f_{12_x})\sin\delta\right]a_1 - (f_{21_y} + f_{22_y})a_2. \tag{8}$$

The quantities a_1 and a_2 in (8) represent the longitudinal distances of G_6 from the front and rear axles, respectively.

Additional constraints are also imposed during the MTLP construction to account for power limits, adherence, complementarity constraints between the accelerating and braking forces, and path constraints required to remain within the track bounds, as detailed in Sect. 4. More details of the model employed are described in [42].

3 ADMM approach to the solution of MLTPs

The scope of our work is to solve an MLTP problem on a given track with a specific vehicle model. The formulation of a discretized optimal control problem takes the form of a general nonlinear programming problem (NLP) and is described by the following equations:

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}), \tag{9a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}) = 0, \tag{9b}$$

$$\boldsymbol{h}(\boldsymbol{x}) \le 0, \tag{9c}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ are the optimization variables, $f(\boldsymbol{x}) \in \mathbb{R}$ is the cost function, $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{N_g}$ is the set of equality constraints, and $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^{N_h}$ is the set of inequality constraints.

Among the many techniques available to discretize and solve OCPs (see, e.g., [48]), in this study, we use the *direct collocation method* (more details will be provided in Sect. 4.2). Clearly, the problem dimension can increase dramatically when the MLT planning horizon grows large as in the case of long race tracks. The maximum size of the MLTP that can be solved is limited by both the complexity of the dynamic model and the performance of the hardware used. Within such limits, the problem can be solved as a single, separate one. In this case, we refer to such an approach as a *serial approach* and to its corresponding solution as a *serial solution*.

When we cross these limits or go well beyond, a parallel approach becomes mandatory. In this case, we refer to such an approach as a *parallel approach* and to its corresponding solution as a *parallel solution*. In the latter perspective, an asset of multicore processors can provide not only an effective computational improvement, but may represent the only way to handle huge problems, as demonstrated in many applications; see, e.g., [24–27]. Among the most successful approaches to tackle large-scale optimization problems in parallel, we count the *alternating direction method of multipliers* (ADMM). This is the preferred choice in the present paper due its relative ease of implementation and intrinsic robustness. The main steps of the ADMM and the original adaptations specific to our OCP are described hereafter.

3.1 Partition of the variables in the parallel approach

The original problem is divided into N_p subproblems. The discretized states, controls, and algebraic variables are organized so that, *internally* to the ith sector, the sequence of states $\boldsymbol{s}_i = \{\boldsymbol{s}_{i,0}\ \boldsymbol{s}_{i,1}\ \cdots\ \boldsymbol{s}_{i,n_i}\}$ consists in $n_i + 1$ discretized samples of $\boldsymbol{s} \in \mathbb{R}^{n_s}$, the sequence of controls $\boldsymbol{u}_i = \{\boldsymbol{u}_{i,0}\ \boldsymbol{u}_{i,1}\ \cdots\ \boldsymbol{u}_{i,n_i-1}\}$ represents n_i discretized samples of $\boldsymbol{u} \in \mathbb{R}^{n_u}$, and the sequence of algebraic variables $\boldsymbol{w}_i = \{\boldsymbol{w}_{i,0}\ \boldsymbol{w}_{i,1}\ \cdots\ \boldsymbol{w}_{i,n_i-1}\}$ stands for n_i samples of $\boldsymbol{w} \in \mathbb{R}^{n_w}$. Therefore $\boldsymbol{s}_i \in \mathbb{R}^{N_{s_i}}$, $\boldsymbol{u}_i \in \mathbb{R}^{N_{u_i}}$, and $\boldsymbol{w}_i \in \mathbb{R}^{N_{w_i}}$, where $N_{s_i} = (n_i + 1)n_s$, $N_{u_i} = n_i n_u$, and $N_{w_i} = n_i n_w$. For convenience, we cast the *internal augmented sequence* in the ith sector as $\boldsymbol{x}_i = \{\boldsymbol{s}_i\ \boldsymbol{u}_i\ \boldsymbol{w}_i\} \in \mathbb{R}^{N_i}$, where $N_i = (n_i + 1)n_s + n_i(n_u + n_w)$.

To account for the states, controls, and algebraic variables at the boundaries between neighboring sectors, it is convenient to define the *head* (h) and *tail* (t) sequences. With reference to the ith sector, assuming that o is the length of the sequence of states in the overlapping area, we define the *head* quantities $\boldsymbol{s}_{i,h} = \{\boldsymbol{s}_{i,-o}\ \boldsymbol{s}_{i,-o+1}\ \cdots\ \boldsymbol{s}_{i,0}\}$, $\boldsymbol{u}_{i,h} = \{\boldsymbol{u}_{i,-o}\ \boldsymbol{u}_{i,-o+1}\ \cdots\ \boldsymbol{u}_{i,-1}\}$, and $\boldsymbol{w}_{i,h} = \{\boldsymbol{w}_{i,-o}\ \boldsymbol{w}_{i,-o+1}\ \cdots\ \boldsymbol{w}_{i,-1}\}$. Similarly, we define the *tail* quantities $\boldsymbol{s}_{i,t} = \{\boldsymbol{s}_{i,n_i}\ \boldsymbol{s}_{i,n_i+1}\ \cdots\ \boldsymbol{s}_{i,n_i+o}\}$, $\boldsymbol{u}_{i,t} = \{\boldsymbol{u}_{i,n_i}\ \boldsymbol{u}_{i,n_i+1}\ \cdots\ \boldsymbol{u}_{i,n_i+o-1}\}$, and $\boldsymbol{w}_{i,t} = \{\boldsymbol{w}_{i,n_i}\ \boldsymbol{w}_{i,n_i+1}\ \cdots\ \boldsymbol{w}_{i,n_i+o-1}\}$. It comes handy to cast the augmented head and tail sequences in the ith sector as $\boldsymbol{x}_{i,h} = \{\boldsymbol{s}_{i,h}\ \boldsymbol{u}_{i,h}\ \boldsymbol{w}_{i,h}\}$ and $\boldsymbol{x}_{i,t} = \{\boldsymbol{s}_{i,t}\ \boldsymbol{u}_{i,t}\ \boldsymbol{w}_{i,t}\}$, respectively.

With reference to Fig. 2, considering as an example the situation at the boundary between sector i and $i + 1$, it is worth observing that $\boldsymbol{x}_{i,t}$ and $\boldsymbol{x}_{i+1,h}$ are local representations of the augmented states for the ith and $(i + 1)$th sectors in the transition area. Then, to gradually enforce coherence in a consensus fashion, the *consensus* augmented states z_i are introduced, whose role is to negotiate possibly conflicting requirements of adjacent sectors. In particular, with reference to the transition between sector i and $i + 1$, coherence is registered if the following conditions are met: $\boldsymbol{x}_{i,t} = \boldsymbol{x}_{i+1,h} = z_i$.

We also introduce the *extended head* (H) and *extended tail* (T) sequences. Assuming that e is the length of the sequence of states in the extended areas, we define the *extended head* quantities $\boldsymbol{s}_{i,H} = \{\boldsymbol{s}_{i,-e}\ \boldsymbol{s}_{i,-e+1}\ \cdots\ \boldsymbol{s}_{i,-1}\}$, $\boldsymbol{u}_{i,H} = \{\boldsymbol{u}_{i,-e}\ \boldsymbol{u}_{i,-e+1}\ \cdots\ \boldsymbol{u}_{i,-1}\}$, and $\boldsymbol{w}_{i,H} = \{\boldsymbol{w}_{i,-e}\ \boldsymbol{w}_{i,-e+1}\ \cdots\ \boldsymbol{w}_{i,-1}\}$. Similarly, we define the *extended tail* quantities $\boldsymbol{s}_{i,T} = \{\boldsymbol{s}_{i,n_i+1}\ \boldsymbol{s}_{i,n_i+2}\ \cdots\ \boldsymbol{s}_{i,n_i+e}\}$, $\boldsymbol{u}_{i,T} = \{\boldsymbol{u}_{i,n_i}\ \boldsymbol{u}_{i,n_i+1}\ \cdots\ \boldsymbol{u}_{i,n_i+e-1}\}$, and $\boldsymbol{w}_{i,T} = \{\boldsymbol{w}_{i,n_i}\ \boldsymbol{w}_{i,n_i+2}\ \cdots\ \boldsymbol{w}_{i,n_i+e-1}\}$. For brevity, both extended augmented head and tail sequences in the ith sector are casted as $\boldsymbol{x}_{i,H} = \{\boldsymbol{s}_{i,H}\ \boldsymbol{u}_{i,H}\ \boldsymbol{w}_{i,H}\}$ and $\boldsymbol{x}_{i,T} = \{\boldsymbol{s}_{i,T}\ \boldsymbol{u}_{i,T}\ \boldsymbol{w}_{i,T}\}$, respectively. It is worth noting that incorporating segments of variables that extend into neighboring sectors can aid in promoting agreement among independent solutions that arise from adjacent sectors right from the initial ADMM iteration, as detailed in Sect. 4.3.2.

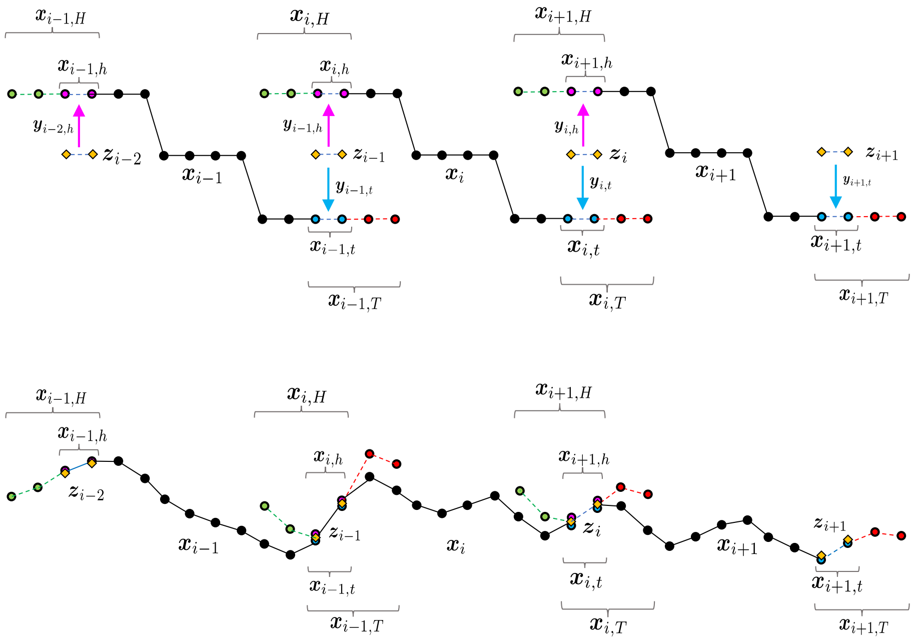


Fig. 2 Conceptual scheme for the allocation of variables among the adjacent sectors (subproblems running in parallel) in the ADMM setting. The upper panel shows a *start* condition. In the lower panel a typical situation obtained *upon convergence* is depicted. Here adjacent states and corresponding consensus variables have reached an agreement (Color figure online)

It is convenient also to introduce $\hat{\boldsymbol{x}}_i = \{\boldsymbol{x}_{i,H}\ \boldsymbol{x}_i\ \boldsymbol{x}_{i,T}\}$ and $\check{\boldsymbol{x}}_i = \{\boldsymbol{z}_{i-1}\ \boldsymbol{x}_i\}$. It is worth noting that while building $\check{\boldsymbol{x}}_i$, the duplication of $s_{i,0}$, contained in both $\boldsymbol{z}_{i-1}, \boldsymbol{x}_i$, is avoided by counting it only once during the concatenation. Hereafter, for concision, both the augmented sequences $\boldsymbol{x}_i$ and the extended augmented sequences $\hat{\boldsymbol{x}}_i$ will be equivalently referred to as *variables* ($\boldsymbol{x}_i$) and *extended variables* ($\hat{\boldsymbol{x}}_i$), respectively.

3.2 A naive parallel approach

Having partitioned the variables $\boldsymbol{x}$ as described above, the solution of original problem (9a)–(9c) can be recovered by properly piecing together the solutions for all sectors. In principle, for the ith sector, the solution $\hat{\boldsymbol{x}}_i^*$ can be computed independently from the others as follows:

$$\hat{\boldsymbol{x}}_i^* = \underset{\hat{\boldsymbol{x}}_i}{\text{argmin}}\ f_i(\hat{\boldsymbol{x}}_i), \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{i,h} - \boldsymbol{z}_{i-1} = 0, \tag{10b}$$

$$\boldsymbol{x}_{i,t} - \boldsymbol{z}_i = 0, \tag{10c}$$

$$\boldsymbol{g}(\hat{\boldsymbol{x}}_i) = 0, \tag{10d}$$

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_i) \le 0, \tag{10e}$$

where (10b) and (10c) are, as explained above, the equality constraints necessary to reestablish coherence in the transition between sectors $i - 1$ and i and between sectors i and

$i+1$. Hereafter, for concision, these constraints will be simply referred to as *consensus constraints*. Note, however, that the optimal values of consensus variables z_i and z_{i-1} are not known in advance.[3] Therefore a mechanism to make them progress toward the *unknown* optimal transition points should be devised.

3.3 A consensus-based ADMM parallel approach

With reference to [23], to solve *in parallel* problems of the form (10a)–(10e), one clever mechanism is provided by the *alternating direction method of multipliers* algorithm. It is worth remarking that the method presented hereafter is our original adaptation of the classical algorithm found in [23].

At a generic kth iteration of the ADMM algorithm, the following *augmented Lagrangian* is introduced for the ith sector:

$$L_i(\hat{\boldsymbol{x}}_i; \boldsymbol{z}_{i-1}^k, \boldsymbol{z}_i^k; \boldsymbol{y}_{i,t}^k, \boldsymbol{y}_{i-1,h}^k) = f_i(\hat{\boldsymbol{x}}_i) + \phi(\boldsymbol{x}_{i,t}, \boldsymbol{x}_{i,h}, \boldsymbol{z}_{i-1}^k, \boldsymbol{z}_i^k; \boldsymbol{y}_{i,t}^k, \boldsymbol{y}_{i-1,h}^k), \tag{11}$$

where we defined the auxiliary function $\phi(\cdot)$ as follows:

$$\phi(\cdot) = \boldsymbol{y}_{i,t}^{kT}(\boldsymbol{x}_{i,t} - \boldsymbol{z}_i^k) + \frac{\rho_{i,t}^k}{2}\|\boldsymbol{x}_{i,t} - \boldsymbol{z}_i^k\|_2^2 + \boldsymbol{y}_{i-1,h}^{kT}(\boldsymbol{x}_{i,h} - \boldsymbol{z}_{i-1}^k) + \frac{\rho_{i,h}^k}{2}\|\boldsymbol{x}_{i,h} - \boldsymbol{z}_{i-1}^k\|_2^2. \tag{12}$$

The vectors $\boldsymbol{y}_{i,t}^k$, $\boldsymbol{y}_{i-1,h}^k$ are the dual variables (multipliers), and $\rho_{i,t}^k, \rho_{i,h}^k > 0$ are penalty parameters. This new cost is composed by the partial cost $f_i(\hat{\boldsymbol{x}}_i)$, and the term $\phi(\cdot)$ that penalizes the mismatch at the boundaries of sector i, i.e., between $\boldsymbol{x}_{i,t}$ and $\boldsymbol{z}_i^k$ and between $\boldsymbol{x}_{i,h}$ and $\boldsymbol{z}_{i-1}^k$, via an augmented Lagrangian strategy.

One ADMM iteration is composed by three fundamental steps. As the *first step*, the following problem is solved:

$$\hat{\boldsymbol{x}}_i^{k+1} = \underset{\hat{\boldsymbol{x}}_i}{\operatorname{argmin}}\, L_i(\hat{\boldsymbol{x}}_i; \boldsymbol{z}_{i-1}^k, \boldsymbol{z}_i^k; \boldsymbol{y}_{i,t}^k, \boldsymbol{y}_{i-1,h}^k), \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\hat{\boldsymbol{x}}_i) = 0, \tag{13b}$$

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_i) \leq 0. \tag{13c}$$

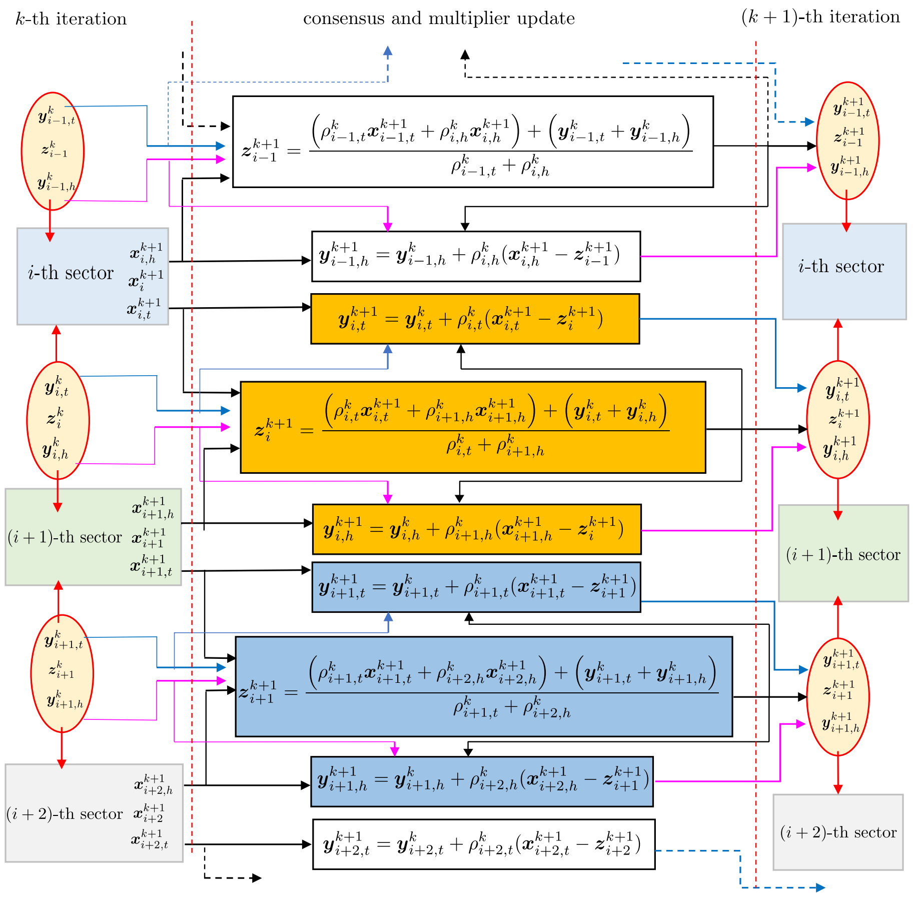
Let us denote by ${}^\sharp G_i^k(\boldsymbol{z}_i) = {}^\sharp L_i^k(\boldsymbol{z}_i) + {}^\sharp L_{i+1}^k(\boldsymbol{z}_i)$ the sum of those cost functions where the ith consensus variable $\boldsymbol{z}_i$ comes into play. The notation $^\sharp$ implies that the evaluation of the function $G_i^k(\boldsymbol{z}_i)$ takes place using quantities at the kth ADMM iteration except for the argument $\hat{\boldsymbol{x}}_i^k$, which is replaced with its best update $\hat{\boldsymbol{x}}_i^{k+1}$, computed from (13a)–(13c). Then, as the *second step* of the ADMM procedure, the best *a posteriori* update for the consensus $\boldsymbol{z}_i^{k+1}$ is computed as follows:

$$\boldsymbol{z}_i^{k+1} = \underset{\boldsymbol{z}_i}{\operatorname{argmin}}\, {}^\sharp G_i^k(\boldsymbol{z}_i). \tag{14}$$

An analytical solution for (14) can be derived, which defines the following simple update rule:

$$\boldsymbol{z}_i^{k+1} = \frac{\left(\rho_{i,t}^k \boldsymbol{x}_{i,t}^{k+1} + \rho_{i+1,h}^k \boldsymbol{x}_{i+1,h}^{k+1}\right) + \left(\boldsymbol{y}_{i,t}^k + \boldsymbol{y}_{i,h}^k\right)}{\rho_{i,t}^k + \rho_{i+1,h}^k}. \tag{15}$$

[3] Moreover, known and constant values for z_i and z_{i-1} are required when dealing with problem (10a)–(10e). This ensures that all sub-problems can be solved independently and in parallel.



k-th iteration
consensus and multiplier update
$(k+1)$-th iteration
i-th sector
$(i+1)$-th sector
$(i+2)$-th sector
$z_{i-1}^{k+1} = \frac{(\rho_{i-1,t}^k x_{i-1,t}^{k+1} + \rho_{i,h}^k x_{i,h}^{k+1}) + (y_{i-1,t}^k + y_{i-1,h}^k)}{\rho_{i-1,t}^k + \rho_{i,h}^k}$
$y_{i-1,h}^{k+1} = y_{i-1,h}^k + \rho_{i,h}^k (x_{i,h}^{k+1} - z_{i-1}^{k+1})$
$y_{i,t}^{k+1} = y_{i,t}^k + \rho_{i,t}^k (x_{i,t}^{k+1} - z_i^{k+1})$
$z_i^{k+1} = \frac{(\rho_{i,t}^k x_{i,t}^{k+1} + \rho_{i+1,h}^k x_{i+1,h}^{k+1}) + (y_{i,t}^k + y_{i,h}^k)}{\rho_{i,t}^k + \rho_{i+1,h}^k}$
$y_{i,h}^{k+1} = y_{i,h}^k + \rho_{i+1,h}^k (x_{i+1,h}^{k+1} - z_i^{k+1})$
$y_{i+1,t}^{k+1} = y_{i+1,t}^k + \rho_{i+1,t}^k (x_{i+1,t}^{k+1} - z_{i+1}^{k+1})$
$z_{i+1}^{k+1} = \frac{(\rho_{i+1,t}^k x_{i+1,t}^{k+1} + \rho_{i+2,h}^k x_{i+2,h}^{k+1}) + (y_{i+1,t}^k + y_{i+1,h}^k)}{\rho_{i+1,t}^k + \rho_{i+2,h}^k}$
$y_{i+1,h}^{k+1} = y_{i+1,h}^k + \rho_{i+2,h}^k (x_{i+2,h}^{k+1} - z_{i+1}^{k+1})$
$y_{i+2,t}^{k+1} = y_{i+2,t}^k + \rho_{i+2,t}^k (x_{i+2,t}^{k+1} - z_{i+2}^{k+1})$

Fig. 3 Block-diagram representation of our ADMM approach tailored to the solution of MLTPs. From left to right the three ADMM steps described in Eqs. (13a)–(13c), (14), and (16a)–(16b) are illustrated. From top to bottom it is possible to see how the consensus variables and multipliers are shared between adjacent sectors and how they interact with each other in the unfolding computations (Color figure online)

As the *third and last step* of the ADMM procedure, the update of the dual variables is performed following an integral law as follows:

$$y_{i,h}^{k+1} = y_{i,h}^k + \rho_{i+1,h}^k (x_{i+1,h}^{k+1} - z_i^{k+1}), \tag{16a}$$

$$y_{i,t}^{k+1} = y_{i,t}^k + \rho_{i,t}^k (x_{i,t}^{k+1} - z_i^{k+1}). \tag{16b}$$

Hence it is worth underlining that the ADMM algorithm consists of the steps described in Eqs. (13a)–(13c), (15), and (16a)–(16b). The algorithm proceeds until certain convergence criteria are met. In particular, in Sect. 3.4 a stopping criterion for ADMM algorithm is explained. A general idea of the flow of information involved in the update steps in Eq. (16a)–(16b) can be elicited from Fig. 3.

3.4 Proposed stopping criterion

Necessary and sufficient optimality conditions for the ADMM problem are presented by Boyd et al. [23], along with a reasonable termination criterion to determine ADMM convergence. Their criterion has been widely used and tested in the literature; see, e.g., [27, 31], or [39].

Here, inspired by their approach, we propose an adaptation, which is more practical in the context of MLTP solutions.

The convergence of ADMM is characterized in terms of the residuals

$$\boldsymbol{r}_{i,h}^{k} = \boldsymbol{x}_{i,h}^{k} - \boldsymbol{z}_{i-1}^{k}, \qquad \boldsymbol{r}_{i,t}^{k} = \boldsymbol{x}_{i,t}^{k} - \boldsymbol{z}_{i}^{k}, \qquad \boldsymbol{d}_{i}^{k} = \boldsymbol{z}_{i}^{k+1} - \boldsymbol{z}_{i}^{k}, \tag{17}$$

where $\boldsymbol{r}_{i,h}^{k}$ and $\boldsymbol{r}_{i,t}^{k}$ are the *head* and *tail primal residual*, respectively, and $\boldsymbol{d}_{i}^{k}$ is the *dual residual*. It is worth pointing out a peculiarity of the *first step* of the proposed formulation. In the original formulation proposed by Boyd et al. [23], the constraints (13b) and (13c) are restricted to be only linear equality constraints. They are treated with a penalty approach, incorporating them into the cost function. Then each of them comes to play a role when computing the *primal residual*. However, in our present study, we take a distinct approach. In fact, since constraints (13b) and (13c) are nonlinear and lack a direct dependence on the consensus variables, these are fulfilled at each iteration of the ADMM algorithm by separate interior-point solver (IPOPT) instances that solve problems (13a)–(13c) (in parallel). As a consequence, the *primal residuals* are computed only as the error between the states $\boldsymbol{x}_{i,h}^{k}$ and $\boldsymbol{x}_{i,t}^{k}$, which always comply with constraints (13b) and (13c) and their corresponding consensus variables $\boldsymbol{z}_{i-1}^{k}$ and $\boldsymbol{z}_{i}^{k}$.

As discussed in [23], a reasonable termination criterion is that the primal and dual residuals must be small. Considering that in our problem states, controls, and algebraic parameters have precise physical meanings for the vehicle engineer, it makes more sense to declare ADMM convergence when the following inequalities hold for each sector:

$$|\boldsymbol{r}_{i,h}^{k}| \leq \boldsymbol{\epsilon}_{r}, \qquad |\boldsymbol{r}_{i,t}^{k}| \leq \boldsymbol{\epsilon}_{r}, \qquad |\boldsymbol{d}_{i}^{k}| \leq \boldsymbol{\epsilon}_{d}, \tag{18}$$

where $\boldsymbol{\epsilon}_r$ and $\boldsymbol{\epsilon}_d$ are the vectors of tolerance values (possibly with different components), and $|\cdot|$ returns a vector with the absolute values of each component of the original entry; $\boldsymbol{\epsilon}_r$ is defined as the *primal residual tolerance*, and $\boldsymbol{\epsilon}_d$ as the *dual residual tolerance*. This allows a finer grain treatment of the convergence for all state, control, and algebraic variables.

4 Optimal control problem formulation

In this section, we outline the key aspects of the minimum lap-time problem formulation, including (i) the components of the cost function, (ii) the structure of the inequality, equality, and terminal constraints, and (iii) the initial guess and variable scaling. These considerations are essential for achieving a practical and efficient solution.

4.1 States, controls, and algebraic variables

With reference to the model illustrated in Sect. 2, the states, controls, and algebraic parameters of the dynamic model are related to the vectors of variables introduced in Sect. 3 as follows:

$$\boldsymbol{s} = [q_1, q_2, q_3, q_4, q_5, q_6, \dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6] \in \mathbb{R}^{12}, \tag{19}$$

$$\boldsymbol{u} = [f_{xa}, f_{xb}, \delta] \in \mathbb{R}^3, \tag{20}$$

$$\boldsymbol{w} = [f_{11_z}, f_{12_z}, f_{21_z}, f_{22_z}, f^3_{3_x}, f^3_{3_y}, m^3_{3_z}] \in \mathbb{R}^7. \tag{21}$$

It is important to highlight that our MLTP is formulated in the spatial domain using the track coordinate q_1 (defined in Sect. 2) as the independent variable, as thoroughly described in our previous work [49]. Therefore the vehicle position q_2 and orientation q_3 (along with the motions q_4, q_5, and q_6 of bounce, pitch, and roll) with respect to the track reference frame should depend on the track coordinate q_1, highlighted in Fig. 1. To reconcile the differential equations obtained through the ABA algorithm, which naturally depend on time t as the independent variable, we need to translate the equations as functions of the independent track coordinate. The spatial formulation of dynamics can be easily recovered by computing $\boldsymbol{s}_{,q_1} = d\boldsymbol{s}/dq_1$ as follows:

$$\boldsymbol{s}_{,q_1}(q_1) = \dot{\boldsymbol{s}}/\dot{q}_1 = F(\boldsymbol{s}(q_1), \boldsymbol{u}(q_1), \boldsymbol{w}(q_1))/\dot{q}_1, \tag{22}$$

where $F(\cdot)$ is the dynamic vector field defined in Eq. (1), in which the reconstruction equations $\dot{\boldsymbol{q}} = \boldsymbol{q}_v$ are pieced together with the accelerations $\dot{\boldsymbol{q}}_v$ obtained through the *articulated body algorithm*. The explicit dependence on the variable q_1 instead of time t is remarked.

4.2 OCP discretization via direct collocation

The generic OCP written for the ith sector is discretized following a *direct collocation* strategy, as described in [3]. Its peculiarity is that the original OCP is transformed into a large (but sparse) *nonlinear program* (NLP).

We now focus on the structure of the ith subproblem (corresponding to the ith sector of the track). First, an equally spaced grid of track coordinates (n_i mesh intervals) is sampled such that, within sector i, $q_{1_j} = jh_q$ $(j = 0, \ldots, n_i)$ with $h_q = (q_{1_{n_i}} - q_{1_0})/n_i$, and q_{1_0} and $q_{1_{n_i}}$ are the starting and final values, respectively. However, according to the definitions in Sect. 3.1, the discretized *augmented* states $\hat{\boldsymbol{x}}_i$ include extended head and tails instances, which serve the purpose of easing convergence offering to the ith problem a landscape beyond its boundaries. Therefore we have $\hat{\boldsymbol{x}}_i = \{\hat{\boldsymbol{x}}_{i,j} \,|\, j = -e, \ldots, n_i + e\}$, where $\hat{\boldsymbol{x}}_i(q_{1_j}) = \hat{\boldsymbol{x}}_{i,j}$ is the discretized set of the optimization variables associated with the jth node. In agreement with the dimension of controls, states, and algebraic vectors stemming from the dynamic model in Sect. 2, each $\hat{\boldsymbol{x}}_{i,j} \in \mathbb{R}^{22}$.

The *first step* in (13a)–(13c) of the ADMM, at its kth iteration, can now be reshaped as

$$\hat{\boldsymbol{x}}_i^{k+1} = \underset{\hat{\boldsymbol{x}}_i}{\operatorname{argmin}} \left[\phi(\boldsymbol{x}_{i,t}, \boldsymbol{x}_{i,h}, \boldsymbol{z}^k_{i-1}, \boldsymbol{z}^k_i; \boldsymbol{y}^k_{i,t}, \boldsymbol{y}^k_{i-1,h}) + \sum_{j=-e}^{n_i-1+e} f_{i,j}(\hat{\boldsymbol{x}}_{i,j}, \hat{\boldsymbol{x}}_{i,j+1}, \hat{\boldsymbol{v}}_{i,j}) \right] \tag{23a}$$

$$\text{s.t.} \quad \ldots$$

$$\boldsymbol{g}(\hat{\boldsymbol{x}}_{i,j}, \hat{\boldsymbol{x}}_{i,j+1}, \hat{\boldsymbol{v}}_{i,j}) = 0, \tag{23b}$$

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{i,j}, \hat{\boldsymbol{x}}_{i,j+1}, \hat{\boldsymbol{v}}_{i,j}) \leq 0. \tag{23c}$$

$$\ldots$$

It is worth noting that, as usual, a set of d *collocation points* $q_{1_{j,m}}$ $(m = 1, \ldots, d)$ can be chosen in each interval $[q_{1_j}; q_{1_{j+1}}]$, allowing for a dth-degree polynomial representation of the state trajectory within each jth interval. Therefore the collocation states have dimension $\hat{\boldsymbol{v}}_{i,j} \in \mathbb{R}^{22(d+1)}$.

The equality constraints $g(\cdot)$ include the dynamic equations (22) and the path algebraic equations (here omitted for brevity) involving variables $\boldsymbol{w}$ and $\boldsymbol{u}$. It is worth noting that this set of constraints contains also the *complementary constraint* $f_{xa} f_{xb} = 0$, which prevents the traction and braking forces from acting simultaneously.

The inequalities $\boldsymbol{h}(\cdot) \leq 0$ (23c) involve all path constraints limiting states, controls, and algebraic parameters. Power limits, adherence constraints, and bounds on the lateral displacement q_2, necessary to remain within track bounds, are all included in this form. Finally, the cost function is approximated in each interval by a quadrature formula. The typical stage cost $f_{i,j}(\cdot)$ for the ith sector takes the form

$$f_{i,j} = (h_q/\dot{q}_{1_{i,j}})^2 + K_\delta(\delta_{i,j+1} - \delta_{i,j})^2, \tag{24}$$

where the first term penalizes lap time, and the second terms prevents abrupt variations of the steering angle weighted by the coefficient K_δ. It is worth noting that comparing (11) with (23a)–(23c), $f_i(\cdot)$ is simply $\sum f_{i,j}$ $(j = -e, \ldots, n_i - 1 + e)$.

4.3 Choice of the ADMM parameters

We shift now our attention to the calibration of the ADMM parameters, whose choice plays an important role on the convergence performance. The parameters are (i) the number of subproblems N_p in which the original MLTP problem is divided, (ii) the number of discretization intervals n_i in each sector, (iii) the length of the overlapping areas measured by the number of samples o (see Sect. 3), (iv) the lengths of the *extended head* and the *extended tail* measured by the number of samples e, (v) the penalty parameters $\rho_{i,t}$ and $\rho_{i,h}$ for the head and tail of sector i, and (vi) the tolerance vectors $\boldsymbol{\epsilon}_r$ and $\boldsymbol{\epsilon}_d$.

4.3.1 Number of subproblems N_p

It is useful to highlight that in the MLTP framework, when the optimization of a single lap of a given track is considered, the number of subproblems N_p corresponds to the number of sectors N_s in which the track is divided, i.e., $N_p = N_s$. Instead, when considering multiple laps N_{lap}, the subproblems are $N_p = N_s N_{\text{lap}}$. The sectors may or may not be of the same length, depending on the user settings. For of simplicity, in this paper, we divide the track into sectors of equal length.

A small value of N_p is the obvious choice when performing the computations on a laptop with a low CPU core count. This results in high-dimensional NLP subproblems (13a)–(13c), which can be slow to solve. On the other hand, a small value of N_p generally requires few ADMM iterations to converge since consensus among a small number of interfaces (i.e., few sectors) is more likely to be achieved (see conditions in (18)). On the contrary, a high value of N_p is the obvious choice on a cluster with many CPU cores since this allows us to drastically cut the size of the NLP subproblems (13a)–(13c), thus promoting their fast convergence. However, the introduction of many consensus variables may require many ADMM iterations for convergence (18). Therefore, as it is clear from the results discussed in Sect. 5, the optimal trade-off is to be decided on a case-by-case basis. When the dimensionality of a problem makes it difficult to solve as a whole, splitting it into smaller parts becomes the only viable solution.

4.3.2 Discretization intervals n_i, length of overlapping areas o, and length of extended head/tail e

The proposed algorithm has three setup parameters, n_i, o, and e. In our approach, a uniform value of n_i is set for all subproblems. To determine the appropriate value of n_i for a given track, a convergence analysis of the optimal time is performed through a serial solution.

The value of o, which indicates the number of points in the mesh grid to be considered for coherence between sectors, is set to 1 to ensure fast convergence while maintaining continuity. The positions of overlapping areas along the track depends on N_p and n_i. In practice, they should be located at key points for better convergence properties, for example, in the straights. Future work should investigate the optimal location for the overlapping areas to ensure sector coherence where vehicle dynamics are smoother, as this can improve ADMM convergence. It is worth noting that we did not implement any specific strategy for the location of overlapping areas to test the robustness of our framework. This aspect remains open for future investigation.

Our work also makes an original contribution through the incorporation of *extended heads* and *tails* of length e in the algorithm. This parameter does not appear in classical ADMM algorithms [23], but its presence in our work is motivated by a speed up in the convergence process. By incorporating segments of variables that extend into neighboring sectors can aid in promoting agreement among independent solutions that arise from adjacent sectors right from the first ADMM iteration. The key observation is that, considering neighboring solutions of sectors i and $i+1$, if the *extended tail* $\boldsymbol{x}_{i,T}$ and *head* $\boldsymbol{x}_{i+1,H}$ extend sufficiently beyond the *overlapping areas* $\boldsymbol{x}_{i,t}$ and $\boldsymbol{x}_{i+1,h}$, then at the solution they will be near to the (unique) global optimum and hence already close to each other. Therefore the two consecutive subproblems i and $i+1$ will quickly reach consensus at their interface. However, a good compromise between the length o of the overlapping areas and the subproblem extremities e is necessary for fast convergence. Choosing a small value of e may not ensure this distance, whereas an excessive value may increase the problem dimension and computational time unnecessarily.

4.3.3 Update rules for the adaptive penalty parameters $\rho_{i,t}$ and $\rho_{i,h}$

Nonconstant penalty parameters and adaptive laws to update them are already available in the literature [23, 31, 33]. In the present work, due to the particular structure of the connection graph between subproblems, two distinct values $\rho_{i,t}$ and $\rho_{i,h}$ have been introduced in (12) with the goal of improving convergence in practice. Their update strategy is reminiscent of that proposed in [23], and for the ith subproblem, we pose

$$\rho_{i,j}^{k+1} = \begin{cases} \tau\rho_{i,j}^{k} & \text{if} \quad \|r_{i,j}^{k}\|_2 > \mu\|d_i^k\|_2, \\ \rho_{i,j}^{k}/\tau & \text{if} \quad \|d_i^k\|_2 > \mu\|r_{i,j}^{k}\|_2, \\ \rho_{i,j}^{k} & \text{otherwise,} \end{cases} \tag{25}$$

where $\mu > 1$ and $\tau > 1$ are parameters, and $j = \{t, h\}$ corresponds to the *tail* or *head* penalty, respectively. Here we set $\mu = 10$ and $\tau = 2$. This approach guarantees that the primal and dual norms are kept within a factor of μ of one another while they converge to zero.

Finally, the tolerance vectors $\boldsymbol{\epsilon}_r$ and $\boldsymbol{\epsilon}_d$ have been assigned for each optimization variable based on sensible engineering accuracy.

4.4 Warm start & scaling

The success of the ADMM algorithm is closely tied to the convergence of each subproblem during the kth iteration, as missing even one of these convergences can cause the algorithm to fail. Therefore it is crucial to properly "warm start" the ADMM. In this work, this is achieved through a *homotopy approach* [50]. The key idea of this method is to set up a sequence of optimal control problems that are easier to solve, gradually transforming them into the original problem until the original NLP is solved. In this process the previous solution provides a "warm start" to initialize the subsequent problem. Note that while the homotopy technique increases robustness, it also increases the computational time required to obtain the optimal solution. In this work, we perform three homotopy iterations before starting the ADMM. It is worth noting that the homotopy technique has not been employed to warm start the *serial* solution, as our numerical tests revealed that this was not necessary. Including it anyway would have added unnecessary ballast to the serial solution and lead to an unfair comparison with the *parallel* (ADMM) solution.

As said, before starting the actual ADMM iterations, three homotopy steps are performed where the Lagrangian function $L_i^k(\cdot)$ of the ith subproblem in Eq. (11) is substituted with

$$\tilde{f}_i(\hat{\boldsymbol{x}}_i) = \lambda f_i(\hat{\boldsymbol{x}}_i) + (1-\lambda) J_i(\hat{\boldsymbol{x}}_i), \tag{26}$$

where $J_i(\hat{\boldsymbol{x}}_i) = (v_{3_x}^3 - \bar{v}_{3_x}^3)^2 + q_2^2$. According to the model notation (see Sect. 2), minimizing cost function $J_i(\cdot)$ would lead to a solution where the vehicle has to maintain a constant speed, $\bar{v}_{3_x}^3$ while progressing along the track centerline. The scalar parameter $\lambda \in [0, 1]$ is such that when $\lambda = 0$, Eq. (26) defines the artificial problem whose solution is easy to obtain, whereas $\lambda = 1$ is associated with the original cost function. It is worth remarking that while performing these iterations, the term $\phi(\cdot)$ defined in (12) related to the *consensus constraints* in (11) is removed, whereas $\phi(\cdot)$ is reintroduced coherently from the first ADMM iteration onward.

Another variation with respect to problem (13a)–(13c) is that also the *complementary constraint* is relaxed through a tolerance $\epsilon_{ab} \geq 0$ such that $-\epsilon_{ab} \leq f_{xa} f_{xb} \leq \epsilon_{ab}$. Therefore our three homotopy iterations start with $\lambda = 0$ and $\epsilon_{ab} > 0$ and end with $\lambda = 1$ and $\epsilon_{ab} = 0$.

In line with the strategy adopted for the *serial problem*, during the first homotopy iteration, the provided initial guess assumes a constant speed of the vehicle along the track centerline. The guess for the remaining optimization variables (controls and algebraic variables) are estimated via inverse dynamics. The final solutions, obtained after this preliminary process, are used to initialize the first ADMM iteration. In this way, a very robust *warm start* is provided.

To conclude, it is worth mentioning that also scaling is crucial for avoiding numerical issues and improving the convergence rate in NLPs as they contain states, controls, and algebraic variables with different ranges. Therefore a normalization is performed on all discretization points using the corresponding expected maximum values, resulting in the scaled variables falling within the range $[-1, 1]$.

5 Results

The proposed consensus-based ADMM algorithm is validated and put on a test on the *Nurburgring* race track, which is one of the longest and most famous circuits in the motorsport context. In particular, the MLTP is solved for the *Nordschleife* version of *Nurburgring*, whose length is $\simeq 21$ km.

The optimal solution of the MLTP is obtained and discussed for a formula SAE vehicle. The choice was driven by the availability, for this setup, of accurate model parameters. For more detail on vehicle parameters, omitted here for brevity, we refer the interested reader to our previous work [42].

The ADMM algorithm is first validated against the *serial solution* on one lap of *Nordschleife*. In the serial solution the problem is solved as a single problem and is considered here as the ground-truth solution since its reliability has already been shown in [42].

Then, to evaluate the performance of the ADMM approach in solving high-dimensional optimization problems, a series of multilap problems are formulated and solved. The aim of these problems is to investigate whether the ADMM approach offers a more efficient solution when faced with a significant increase in problem dimensionality compared to the traditional serial approach.

Finally, a single lap is solved multiple times with different ADMM settings to investigate its performances when the number of subproblems N_p is increased but the size of the overall problem is not excessively large.

The validation process is performed on a laptop with 2.30 GHz (boosted at 4.5 GHz) Intel(R) Core(TM) i7-10875H CPU and 32 GB RAM representative of a standard portable personal computer. Instead, the multilap problems are solved, and the lap splitting is analyzed on a cluster workstation equipped with four sockets containing an Intel(R) Xeon(R) Platinum 8260L CPU @ 2.40 GHz (boosted at 3.4 GHz) each, and 3.70 TB RAM made available by the Sistema Informatico Dipartimentale (SID) of the Università di Pisa. The latter has overall 96 CPU cores and was chosen as the ideal platform for exploiting distributed computations. Hereafter, for concision, we refer to the above hardware settings as *laptop* and *cluster*, respectively.

The optimal control problems are coded in a scripting environment using the MATLAB interface to the open-source CasADi framework [51], which provides building blocks to efficiently formulate and solve large-scale optimization problems. To solve each NLP in (13a)–(13c), the IPOPT [21] solver is used.

5.1 Validation of the ADMM approach

5.1.1 Mesh size calibration

The validation process begins with a convergence analysis necessary to accurately determine the appropriate mesh size in the discretization of the track. This is accomplished by solving the MLTP on *Nordschleife* as a *single problem* while gradually increasing the number of mesh intervals. According to the ADMM notation, this involves setting $N_p = 1$ and increasing the number of discretization points n_1. The results of this analysis, which was conducted on the *laptop* configuration, are presented in Table 1.

During the analysis, various key performance indicators (KPIs) are monitored as n_1 is increased. These include (i) the optimal time t_{opt}, required for the vehicle to complete one lap, (ii) the computational time T_{s}, required by IPOPT to solve the problem, (iii) the number Iter of IPOPT iterations to convergence, (iv) the average amount of seconds per iteration s/Iter, which gives an insight on the speed of the solution process, and (v) the total number of NLP variables, which is given by $N_{\text{var}} = n_1(n_s(d+1) + n_u + n_z) + n_s$ and is helpful in tracking the problem dimension.

Table 1 shows how the optimal time t_{opt}, along with other KPIs, varies as a function of the size of the mesh. The best compromise between the speed of the solution process, measured by s/Iter, and its quality, measured by t_{opt}, is given by setting $n_1 = 1500$. This

Table 1 Convergence analysis with respect to the overall number n_1 of discretization intervals on a single lap of the *Nordschleife* circuit, solved as a single (serial) problem. Hardware configuration: *laptop*. Whereas the optimal time t_{opt} does not change significantly while increasing n_1, the computational time T_{S} and the number of iterations vary substantially. The mesh $n_1 = 1500$ is the best compromise between the speed of the solution process, measured by s/Iter, and its quality, measured by t_{opt}. This value is therefore chosen to perform the subsequent comparisons between the serial and ADMM solutions

n_1	$t_{\mathrm{opt}}(s)$	$T_{\mathrm{S}}(s)$	Iter	s/Iter	N_{var}
1000	518.1	287.7	71	4.05	46012
1500	**517.7**	**516.1**	85	6.07	69012
2000	517.7	671.6	84	8.0	92012
2500	517.7	584.4	57	10.3	115012
3000	517.6	542.6	45	12.1	138012
3500	517.6	719.2	51	14.1	161012
4000	517.6	852.3	52	16.4	184012

will be fixed in all the tests performed hereafter for the validation and testing of ADMM on the Nordschleife track. This choice does not trade accuracy for efficiency, like $n_1 = 1000$, for which an error in t_{opt} of 0.5 s (518.1 vs 517.6 s) may not be acceptable. On the other end, it would require to select $n_1 = 3000$, thereby almost doubling the number of variables N_{var} (from 69012 to 138012) and almost doubling s/Iter (from 6.07 to 12.1), to improve by only 0.1 s (0.02%) the value of t_{opt}. With the chosen value $n_1 = 1500$, the mesh size corresponds to one discretization point every 14 meters along the track.

5.1.2 ADMM parameters employed

Turning our attention to the setup of the ADMM parameters, the optimal settings for the *Nordschleife* track have been determined through numerical tests by the authors. These settings consist of $N_p = 4$, the number of sector elements n_i coherent with the selected mesh for the track, and $e = 40$ points, so that both $\boldsymbol{x}_{i,H}$ and $\boldsymbol{x}_{i,T}$ span approximately 560 meters beyond the consensus interface. The distance pertaining to each ith subproblem and corresponding to the variable $\hat{\boldsymbol{x}}_i$ is computed by dividing the track length by N_p and adding the extended head and tail lengths. These *internal* sectors corresponding to $\hat{\boldsymbol{x}}_i$ ($i = 2, \ldots, N_p - 1$) have an optimized distance of approximately 6300 meters. Instead, the first ($i = 1$) and last ($i = N_p$) subproblems, which have only either an extended tail or an extended head, respectively, present an optimized distance of approximately 5750 meters.

5.1.3 Discussion of the validation process

Using the above-discussed setup parameters, the ADMM results are presented in a concise format in Table 2. This table encompasses the KPIs that pertain specifically to the *parallel* approach. These comprise (i) the optimal time t_{opt}, required for the vehicle to complete one lap, (ii) the computational time T_{ADMM}, required to execute the ADMM iterations, (iii) the homotopy time T_h, i.e., the time spent in performing the three homotopy iterations necessary to warm start the problem, (iv) the total computational time T_p of the *parallel* approach, with $T_p = T_h + T_{\mathrm{ADMM}}$, (v) the number of iterations of the ADMM algorithm, denoted as Iter (or k), (vi) the average amount of seconds per ADMM iteration s/Iter, computed as T_{ADMM}/k, and (vii) the total number of NLP variables N_{var}, given by $\sum_{i=1}^{N_p} N_{\mathrm{var},i}$. Here the number of

Table 2 Numerical results obtained through the ADMM algorithm for an overall number of discretization intervals $n_1 = 1500$. Hardware configuration: *laptop*. The overall computational time T_p for the parallel approach encompasses the homotopy time T_h and the ADMM time T_{ADMM}, i.e., $T_p = T_h + T_{\mathrm{ADMM}}$. Although the ADMM algorithm is very efficient, requiring only three iterations $k = 3$, the overhead of a warm start process via homotopy causes the parallel approach to be slower than the serial approach

N_p	$t_{\mathrm{opt}}(s)$	$T_{\mathrm{ADMM}}(s)$	$T_{\mathrm{h}}(s)$	$T_{\mathrm{p}}(s)$	Iter (k)	s/Iter	N_{var}
4	**517.7**	357.8	224	**581.8**	3	119.3	80088

variables of the ith subproblem is computed as $N_{\mathrm{var},i} = (n_i + n_e e)(n_s(d+1) + n_u + n_z) + n_s$, where $n_e = 2$ ($i = 2, \ldots, N_p - 1$), and $n_e = 1$ (for $i = 1$ and $i = N_p$, i.e., the first and last subproblems), which have only either an *extended tail* or an *extended head*, respectively.

Upon comparing Table 2 with the second row of Table 1, two key observations are in order. The first one is that for the same overall track discretization points, the optimal laptime value t_{opt} provided by the ADMM algorithm and the *serial* solution are exactly the same, i.e., $t_{\mathrm{opt}} = 517.7$ s. This indeed validates the ADMM algorithm against the classical solution. The second observation is that the parallel approach requires a computational time $T_p = 581.8$ s (see the 5th column of Table 2), larger than $T_s = 516.1$ s required by the *serial* solution.

The latter result can be attributed mainly to three different factors: (i) In the first place, the *homotopy approach*, used to warm start the ADMM algorithm, but absent in the serial solution, increases the total computational time T_p since $T_p = T_{\mathrm{ADMM}} + T_h$; however, this is inevitable at this stage to ensure that a fair comparison is performed between two equally robust approaches to the solution; (ii) The *laptop* configuration employed in the validation process may not be very efficient in managing parallel computations due to its hardware architecture, thereby favoring the serial computation at the expense of the solution of the ADMM; note in support of this particular reason that when the *cluster* configuration is employed, the situation is reversed (see results in Table 3 and the corresponding discussion in Sect. 5.2); (iii) The NLP dimension considered for the validation ($N_{\mathrm{var}} = 69012$ for the serial case; see Table 1) is not large enough to cause the *serial* approach either to max out the computational resources of the *laptop* configuration or to fail completely.

As said, there is no way out for the additional computational burden due to point (i), since the *homotopy approach* is required to increase ADMM robustness. However, to demonstrate the computational advantages of the *parallel* approach when run on an appropriate hardware configuration, issues raised in points (ii) and (iii) will be explored in greater detail in Sects. 5.2 and 5.3. In particular, in Sect. 5.2, the NLP dimension is increased by optimizing on multilap horizon, whereas the *parallel* and *serial* optimizations are both performed using the *cluster* configuration.

To demonstrate the consistency of the solutions provided by the *serial* and *parallel* solutions in the validation case, in addition to obtaining the same optimal lap time t_{opt}, a compelling comparison is reported in Figs. 4 and 5. Specifically, Fig. 4 depicts the optimal longitudinal speed profiles of the vehicle for the *serial* and ADMM solutions. Here the red curve represents the serial solution, whereas the marker-style thick lines (in the background) refer to the ADMM solution. The different colors refer to the distributed solutions computed in parallel for the $N_p = 4$ sectors. It is worth noting that the solutions correctly match at the interfaces where both the primal and dual residual vectors $\boldsymbol{r}_{i,h}^k$, $\boldsymbol{r}_{i,t}^k$, and $\boldsymbol{d}_i^k$ satisfy conditions (18).

In Fig. 4, also a scaled version of the profile of the longitudinal speed error e_{v_x} between *serial* and ADMM solutions is visible. The small ripples at the interfaces, with max value in the order of 5 mm/s, foster the statement of a perfect agreement, from a practical standpoint,

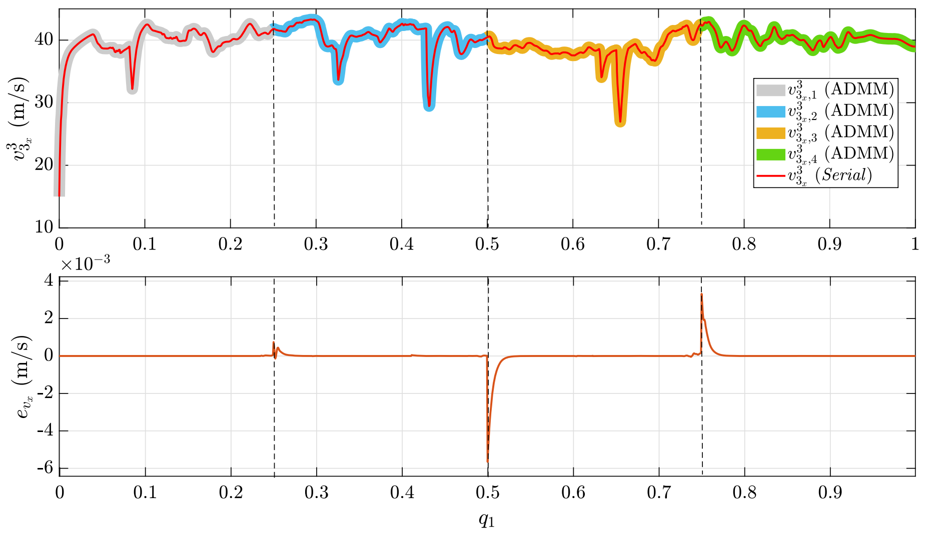


Fig. 4 [Upper panel] Comparison of optimal speed profiles computed as the *serial solution* (thin red line) and *parallel (ADMM) solution* (marker-style thick lines). Hardware configuration: *laptop*. Different colors refer to different sectors computed in parallel in the ADMM setting. The additional vertical dashed lines mark the interfaces between adjacent sectors. [Lower panel] The error profile e_{v_x}, computed as the difference of the *serial* and ADMM solutions (thin orange line), even considering the small ripples at the interfaces (with negligible max value in the order of 5 mm/s), demonstrates a perfect agreement between the two solutions everywhere (Color figure online)

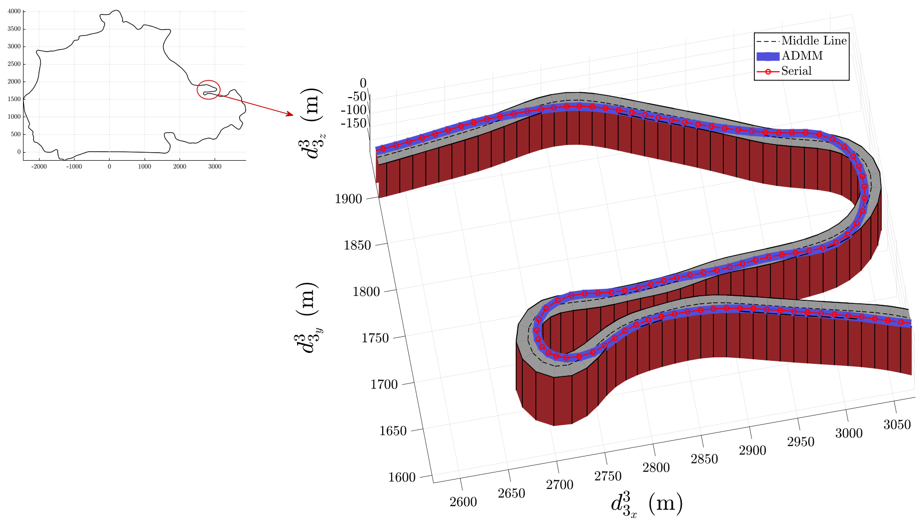


Fig. 5 Optimal trajectory comparison of *serial solution* (red) and *parallel solution* (blue). The trajectories are perfectly overlapped, underling the equivalence of the two solutions (Color figure online)

between the two solutions everywhere. It is worth noting, in passing, that the range in which the speed error exponentially vanished may hold significant implications. These intervals

Table 3 Results for MLTPs on multilap scenarios for both serial and ADMM approaches. Hardware configuration: *cluster*. The hardware architecture allows the ADMM algorithm to be more efficient with respect to the *serial* approach already on $N_{\text{lap}} = 1$ lap. As the number of laps N_{lap} increases and, as a result, the problem size N_{var} grows, the ADMM algorithm outperforms the serial one and represents the sole feasible option to maintain computational times within acceptable bounds

N_{lap}	Serial				ADMM				
	t_{opt} (s)	T_s (s)	Iter	N_{var}	t_{opt} (s)	T_p (s)	N_p	Iter (k)	N_{var}
1	517.7	**633.8**	85	69012	517.7	**567.1**	4	3	80088
2	1034	1620	111	138012	1034	612.3	8	3	163856
3	1550	2232	94	207012	1550	624.3	12	3	247624
4	2067	2710	95	276012	2067	656.9	16	3	331392
8	4132	5308	92	552012	4132	738.2	32	3	666464
16	8262	47640	397	1104012	8262	926	64	3	1336608

may represent a form of *extinction lengths*, which indicate how deeply perturbations at the sector's boundary affect the solution inward the sector. Note that neither the error entity nor the length remains constant across different interfaces, as the local curvature, inclination, and banking of the track at the boundaries may have a notable influence. Consequently, additional research is required in the future to fully elucidate the true significance of this phenomenon.

Additionally, Fig. 5 provides a qualitative comparison of the optimal trajectories for a particular track sector of the *Nurburgring* circuit, underscoring again the equivalence of the two solutions.

As a result, the suggested findings serve to verify the efficacy of the ADMM algorithm in providing an identical solution to that of a traditional serial approach.

5.2 Comparison of ADMM and serial solutions in multilap scenarios

In this section, we set out to quantitatively verify whether the ADMM-based approach delivers on the promises of increased efficiency when faced with solving problems of very large dimensions. To this sake, starting from the usual horizon of one lap ($N_{\text{lap}} = 1$), a series of multilap problems ($N_{\text{lap}} = 2, 3, 4, 8, 16$) are formulated and solved either in a *serial* or *parallel* fashion via the ADMM approach.

With reference to Sect. 4.3, in the ADMM framework, each lap is divided into $N_s = 4$ sectors, the number of sector elements n_i is coherent with the selected mesh for the track, and $e = 40$ points, so that both $\boldsymbol{x}_{i,H}$ and $\boldsymbol{x}_{i,T}$ span approximately 560 meters. With reference to Table 3, the number of ADMM subproblems is $N_p = N_s N_{\text{lap}}$, so that $N_p = 4, 8, 12, 16, 32, 64$.

Table 3 shows the results of the tests run on the *cluster* configuration for both *serial* and ADMM approaches. The first row corresponds to a single-lap MLTP ($N_{\text{lap}} = 1$), whereas rows 2–6 correspond to multilap MLTP problems ($N_{\text{lap}} = 2, 3, 4, 8, 16$). The optimal solutions obtained from both methods are consistent, as seen from the corresponding (minimum-time) t_{opt} columns. Even for a single-lap problem, when multicore CPU architectures are employed, the ADMM approach beats the serial one already for $N_{\text{lap}} = 1$ with $T_p = 567.1$ s vs $T_s = 633.8$ s.

Furthermore, it is apparent that as the number of laps increases, the ADMM outperforms the serial approach significantly. Comparing the T_s and T_p columns, T_s increases almost

Table 4 The table shows the results for the lap-splitting analysis. Here the planning horizon is fixed to $N_{\text{lap}} = 1$ lap, and the number of subproblems N_p is increased. Hardware configuration: *cluster*. A high value of N_p leads to a rise in the number of iterations k to convergence while the computational time required for each ADMM iteration, encoded in s/Iter, decreases

N_p	t_{opt} (s)	T_{ADMM} (s)	T_h (s)	T_p (s)	Iter (k)	s/Iter	N_{var}
2	517.7	1618	317	1935	4	404.5	71048
4	517.7	2158	197	2355	13	166	75120
8	517.7	2218	437	2655	17	130.5	83126
16	517.7	1968	178	2146	28	70.3	99736
32	517.7	2684	109	2793	53	50.6	132312

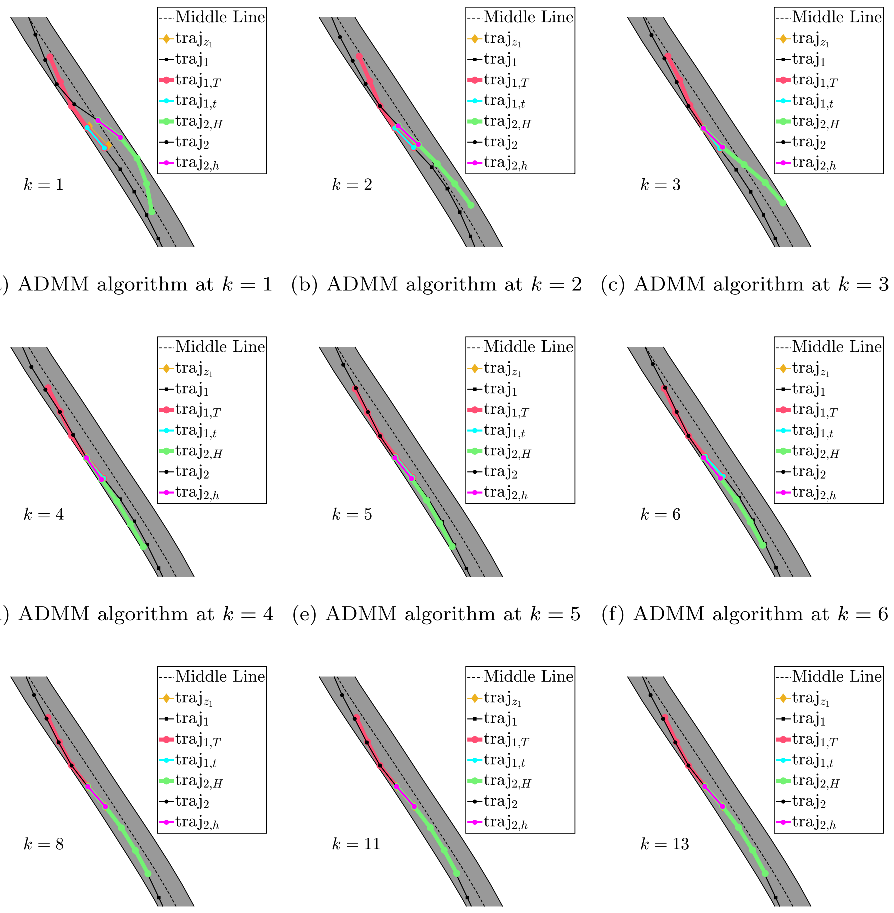
linearly with N_{lap} until $N_{\text{lap}} = 8$, peaking for $N_{\text{lap}} = 16$ (i.e., $N_{\text{var}} = 1104012$ variables), when a huge $T_s \simeq 13$ h and 14 min is registered. In contrast, T_p remains almost constant with increasing N_{lap} until $N_{\text{lap}} = 8$, with a slight increase for $N_{\text{lap}} = 16$, where $T_p \simeq 15$ min, which still denotes a remarkable performance. This problem contains $N_{\text{var}} = 1336608$ variables. The increase of T_p as N_{lap} increases, which is not theoretically expected due to the availability of a large number of CPU cores, is mostly due to inter-CPU communication overheads and is unavoidable. However, the key takeaway is that for large-scale problems (over 1 million variables), which could also be the result of planning on shorter horizons but with larger dynamic models, the proposed distributed approach is the only viable solution to keep computational times within acceptable limits.

5.3 Testing ADMM performances for $N_{\text{lap}} = 1$ with varying N_p

In this section, we set out to evaluate the performance of ADMM by increasing the number of subproblems while the optimization horizon is kept constant. For this purpose, we select a fixed optimization horizon of one lap ($N_{\text{lap}} = 1$) and solve the associated MLTP by dividing the track into $N_p = 2, 4, 8, 16, 32$ sectors. By doing so we can analyze the effect of increasing the number of subproblems on ADMM performance without changing the optimization horizon on a small-scale problem.

For these test cases, the number of sector elements n_i is coherent with the selected mesh for the track, i.e., one discretization point every 14 meters (similarly to the choice in Sect. 5.1.1). The length of the *extended tail* and *head* is the same for each test case, and it is set according to the length of the shortest subproblem. For example, considering that for $N_p = 32$, the distance pertaining to the ith subproblem (corresponding to variable $\check{\boldsymbol{x}}_i$) is approximately 647 meters, we set $e = 22$. Hence both $\boldsymbol{x}_{i,H}$ and $\boldsymbol{x}_{i,T}$ span approximately 305 meters. With these settings, the length of the *internal* sectors corresponding to $\hat{\boldsymbol{x}}_i$ ($i = 2, \ldots, N_p - 1$) have an optimization distance of approximately 11000 meters when $N_p = 2$ and of 1260 meters when $N_p = 32$. It is worth remarking that the optimization distance for the other cases (i.e., $N_p = 4, 8, 16$) assumes intermediate values in that range.

Table 4 presents the results of the lap-splitting analysis in a condensed format. From the table four key aspects of the parallel approach are apparent. The first one is highlighted in the 2nd row, where $N_p = 4$. A comparison of this row with the ADMM outcome in the 1st row of Table 3 reveals the impact of the parameter e. Reducing the length of the *extended tail* and *head* by half causes a four-fold increase in both the computational time T_p and the number of iterations k. This finding suggests that the chosen value of e may not provide a sufficient distance beyond the overlapping area to ensure a quick consensus convergence (refer to Sect. 4.3.2).



(a) ADMM algorithm at $k = 1$ (b) ADMM algorithm at $k = 2$ (c) ADMM algorithm at $k = 3$

(d) ADMM algorithm at $k = 4$ (e) ADMM algorithm at $k = 5$ (f) ADMM algorithm at $k = 6$

(g) ADMM algorithm at $k = 8$ (h) ADMM algorithm at $k = 11$ (i) ADMM algorithm at $k = 13$

Fig. 6 An highlight on ADMM convergence process for optimal trajectory when $N_p = 4$ (Color figure online)

In Fig. 6 a pictorial representation of the convergence process with $N_p = 4$ is shown. For clarity, the *extended tail* and *head* have been trimmed in the plots. With reference to Figs. 6a, 6b, 6c, and 6d, we can observe that the most significant changes in trajectory occur during the initial four iterations. From a careful observation we can note that the *extended head* $\mathrm{traj}_{2,H}$ is displaced from the right of traj_1 (at $k = 1$, Fig. 6a) to its left (at $k = 4$, Fig. 6d). Then the variations in the shape of the trajectories are reduced gradually, as evident from the sequence of Figs. 6e–6i for $k = 8, \ldots, 13$. During this phase, ADMM addresses the strict constraints on the primal and dual residuals, leading the consensus to its optimal configuration.

A second key observation can be elicited from the 6th column of Table 4. Increasing the number of subproblems leads to an increase in the number of ADMM iterations required for convergence. This is because a larger value of N_p leads to an increased number of consensus interfaces, which may require more iterations to reach an agreement. This phenomenon can be attributed to two factors. Firstly, increasing the number of consensus interfaces raises the likelihood of them being located in areas with nonsmooth vehicle dynamics, such as the center of a turn. Secondly, as N_p increases, the consensus variables associated with the *head* and *tail* interfaces become closer. Subsequent changes to the consensus variables during ADMM iterations can significantly impact the optimal solution $\boldsymbol{x}_i$.

The third pretty intuitive aspect is confirmed from the 7th column of Table 4: when parallelization is increased (N_p increases), the computational time per iteration s/Iter decreases. As discussed in Sect. 4.3.1, high values of N_p can drastically cut the size of the NLP subproblems.

However, a fourth observation is in order: in terms of overall computational time T_p, it is not always clear which is the most influential factor between the increased numbers of ADMM iterations (Iter) required to reach consensus and the reduction in the computational time per iteration (s/Iter) as N_p increases. In fact, from the 3rd column in Table 4, in this particular test case, the ADMM computational time T_p does not show a clear trend as N_p increases. This is observed when comparing the results for $N_p = 16$, for which $T_p = 2146$ s, with those for $N_p = 8$ and 32, where $T_p = 2655$ s and $T_p = 2793$ s are registered, respectively. As a general rule, however, we can state that when the problem dimension is small, as in the case of Table 4, using ADMM is not necessarily a wise choice, as the serial approach may be more efficient in providing satisfactory results. Based on our numerical test cases, the ADMM approach should be devoted either to large- and huge-scale problems or to more complex models. Specifically, when we mention a more complex model, we are referring to advanced multibody vehicle models. These models include features like individual wheel handling, suspension system dynamics, aerodynamic maps, and comprehensive Pacejka or thermo-mechanical tire models. Additionally, this complexity may extend to cover hybrid or electric powertrain systems, including considerations for battery dynamics. In those cases the presented distributed optimization becomes more of a necessity.

6 Conclusions

In this paper, we presented a *parallel* approach to address minimum-lap-time problems characterized by a number of variables growing at an unprecedented scale. Starting from the classical *serial* formulation, in which the resulting NLP is solved as a single problem, we presented a consensus-based alternating direction method of multipliers (ADMM) approach to solve MLTPs. It seems the first time that MLTPs involving accurate dynamic models, from a vehicle engineer's perspective, are solved in a distributed fashion.

The aim of our work is to demonstrate the convenience and, in some cases, the necessity of using the parallel approach. To this sake, various tests are presented involving a race-car on the *Nurburgring* track taking into account both slope and banking.

First and foremost, the ADMM approach is shown to successfully converge in locating optimal consensus interfaces and adhering to user-defined tolerances. As a matter of fact, a comparison between the *parallel* and *serial* optimization, employed as the reference solution, yields identical outcomes. However, although the *parallel* and *serial* approaches achieve the same optimal trajectories, they show a significant difference in their computational performances.

Based on the validation procedure (Sect. 5.1) and analysis of the multilap scenarios (Sect. 5.2), two interwoven conditions favor the convenience of the ADMM over the serial approach. The first one, evident from the multilap analysis (see Table 3), is the problem dimension. For large-scale problems (over 1 million variables), which could also occur when dealing with shorter horizons but with larger dynamic models, the proposed distributed approach outperforms the serial one and allows us to keep computational time within acceptable limits. The second one is the availability of multicore CPU architectures, in which the distributed algorithm can be efficiently deployed.

The analysis involving the lap-splitting scenario (Sect. 5.3) brings about the importance of two parameters that must be carefully addressed when employing the ADMM approach. These are the number of subproblems N_p and the extended horizon beyond the consensus interface e. A high degree of parallelization, associated with a large value of N_p, should be devoted to both large- and huge- scale problems. Indeed, increasing the number of subproblems in other situations may lead to a higher number of ADMM iterations and larger computational times with respect to a standard *serial* approach. A special attention should be reserved also to the preview length associated with e. Reducing the length of the *extended tail* or *head* may drastically increase the *parallel* algorithm computational time, as clearly shown when comparing the single-lap results with $N_p = 4$ obtained in Sect. 5.2 with those in Sect. 5.3.

To sum up, the key finding of this study is that the ADMM is particularly beneficial for MLTP problems when planning for long horizons or using complex models. In such cases the ADMM approach with multicore CPU architectures can drastically reduce the computational time. When pushing the limits to very large- and huge-scale problems, resorting to a distributed approach like ADMM becomes mandatory. However, a final *caveat* is in order: when utilizing the proposed *parallel* approach, careful attention must be paid to its setup parameters. Further insights in this direction are left for future work.

Author contributions All authors contributed equally to the manuscript.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement.

Declarations

Disclosure statement No potential conflict of interest was reported by the authors.

Competing interests The authors declare no competing interests.

References

1. Massaro, M., Limebeer, D.J.N.: Minimum-lap-time optimisation and simulation. Veh. Syst. Dyn. **59**(7), 1069–1113 (2021)
2. Massaro, M., Lovato, S., Veneri, M.: An optimal control approach to the computation of g-g diagrams. Veh. Syst. Dyn., 1–15 (2023)
3. Gabiccini, M., Bartali, L., Guiggiani, M.: Analysis of driving styles of a GP2 car via minimum lap-time direct trajectory optimization. Multibody Syst. Dyn. **53**(1), 85–113 (2021)
4. Christ, F., Wischnewski, A., Heilmeier, A., Lohmann, B.: Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. Veh. Syst. Dyn., 1–25 (2019)

5. Dal Bianco, N., Bertolazzi, E., Biral, F., Massaro, M.: Comparison of direct and indirect methods for minimum lap time optimal control problems. Veh. Syst. Dyn. **57**(5), 665–696 (2019)
6. Dal Bianco, N., Lot, R., Gadola, M.: Minimum time optimal control simulation of a GP2 race car. Proc. Inst. Mech. Eng., Part D, J. Automob. Eng. **232**(9), 1180–1195 (2018)
7. Lovato, S., Massaro, M.: Three-dimensional fixed-trajectory approaches to the minimum-lap time of road vehicles. Veh. Syst. Dyn. **60**(11), 3650–3667 (2021)
8. Brayshaw, D.L., Harrison, M.F.: A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location. Proc. Inst. Mech. Eng., Part D, J. Automob. Eng. **219**(6), 725–739 (2005)
9. Brayshaw, D.L., Harrison, M.F.: Use of numerical optimization to determine the effect of the roll stiffness distribution on race car performance. Proc. Inst. Mech. Eng., Part D, J. Automob. Eng. **219**(10), 1141–1151 (2005)
10. Lovato, S., Massaro, M.: A three-dimensional free-trajectory quasi-steady-state optimal-control method for minimum-lap-time of race vehicles. Veh. Syst. Dyn. **60**(5), 1512–1530 (2021)
11. Veneri, M., Massaro, M.: A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles. Veh. Syst. Dyn. **58**(6), 933–954 (2020)
12. Lot, R., Dal Bianco, N.: The significance of high-order dynamics in lap time simulations. In: The Dynamics of Vehicles on Roads and Tracks – Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics, IAVSD 2015, pp. 553–562 (2016)
13. Biniewicz, J., Pyrz, M.: A quasi-steady-state minimum lap time simulation of race motorcycles using experimental data. Veh. Syst. Dyn., 1–23 (2023)
14. Kelly, D.P., Sharp, R.S.: Time-optimal control of the race car: influence of a thermodynamic tyre model. Veh. Syst. Dyn. **50**(4), 641–662 (2012)
15. Perantoni, G., Limebeer, D.J.N.: Optimal control for a formula one car with variable parameters. Veh. Syst. Dyn. **52**(5), 653–678 (2014)
16. Leineweber, D.B., Schäfer, A., Bock, H.G., Schlöder, J.P.: An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Comput. Chem. Eng. **27**(2), 167–174 (2003)
17. Patterson, M.A., Rao, A.V.: GPOPS - II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. ACM Trans. Math. Softw. **41**(1), 1–37 (2014)
18. Nie, Y., Faqir, O., Kerrigan, E.C.: ICLOCS2: try this optimal control problem solver before you try the rest. In: 2018 UKACC 12th International Conference on Control (CONTROL). IEEE, Sheffield (2018)
19. Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: CasADi: a software framework for nonlinear optimization and optimal control. Math. Program. Comput. **11**(1) (2019)
20. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**(4), 979–1006 (2002)
21. Wächter, A., Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
22. Kuhlmann, R., Büskens, C.: A primal–dual augmented Lagrangian penalty-interior-point filter line search algorithm. Math. Methods Oper. Res. (2017)
23. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2010)
24. Braun, P., Grune, L., Kellett, C.M., Weller, S.R., Worthmann, K.: A distributed optimization algorithm for the predictive control of smart grids. IEEE Trans. Autom. Control **61**(12), 3898–3911 (2016)
25. Braun, P., Faulwasser, T., Grüne, L., Kellett, C.M., Weller, S.R., Worthmann, K.: Hierarchical distributed ADMM for predictive control with applications in power networks. IFAC J. Syst. Control **3**, 10–22 (2018)
26. Ling, Q., Ribeiro, A.: Decentralized dynamic optimization through the alternating direction method of multipliers. IEEE Trans. Signal Process. **62**(5), 1185–1197 (2014)
27. Wang, Y., Wu, L., Li, J.: A fully distributed asynchronous approach for multi-area coordinated network-constrained unit commitment. Optim. Eng. **19**(2), 419–452 (2018)
28. Wang, Y., Yin, W., Zeng, J.: Global convergence of ADMM in nonconvex nonsmooth optimization. J. Sci. Comput. **78**(1), 29–63 (2018)
29. Chang, X., Liu, S., Zhao, P., Li, X.: Convergent prediction–correction-based ADMM for multi-block separable convex programming. J. Comput. Appl. Math. **335**, 270–288 (2018)
30. Buccini, A., Dell'Acqua, P., Donatelli, M.: A general framework for ADMM acceleration. Numer. Algorithms **85**(3), 829–848 (2019)
31. Ghadimi, E., Teixeira, A., Shames, I., Johansson, M.: Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. IEEE Trans. Autom. Control **60**(3), 644–658 (2015)

32. Goldstein, T., O'Donoghue, B., Setzer, S., Baraniuk, R.: Fast alternating direction optimization methods. SIAM J. Imaging Sci. **7**(3), 1588–1623 (2014)
33. Song, C., Yoon, S., Pavlovic, V.: Fast ADMM algorithm for distributed optimization with adaptive penalty. In: 30th AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 753–759 (2016)
34. Franca, G., Bento, J.: Distributed optimization, averaging via ADMM, and network topology. Proc. IEEE **108**(11), 1939–1952 (2020)
35. Zhang, R., Kwok, J.T.: Asynchronous distributed ADMM for consensus optimization. In: 31st International Conference on Machine Learning, ICML 2014, vol. 5, pp. 3689–3697 (2014)
36. Pei, C., Wan, C., Dai, R., Rea, J.R.: A hybrid ADMM for six-degree-of-freedom entry trajectory optimization based on dual quaternions. IEEE Trans. Aerosp. Electron. Syst., 1–16 (2022)
37. Rastgar, F., Masnavi, H., Shrestha, J., Kruusamae, K., Aabloo, A., Singh, A.K.: GPU accelerated convex approximations for fast multi-agent trajectory optimization. IEEE Robot. Autom. Lett. **6**(2), 3303–3310 (2021)
38. Ni, R., Pan, Z., Gao, X.: Robust multi-robot trajectory optimization using alternating direction method of multiplier. IEEE Robot. Autom. Lett. **7**(3), 5950–5957 (2022)
39. Wang, C., Bingham, J., Tomizuka, M.: Trajectory splitting: a distributed formulation for collision avoiding trajectory optimization. In: IEEE/RSJ Intelligent Robots and Systems (IROS), pp. 8113–8120 (2021)
40. Zhou, Z., Zhao, Y.: Accelerated ADMM based trajectory optimization for legged locomotion with coupled rigid body dynamics. In: American Control Conference, pp. 5082–5089 (2020)
41. Aydinoglu, A., Posa, M.: Real-time multi-contact model predictive control via ADMM. In: IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, pp. 3414–3421 (2022)
42. Bartali, L., Gabiccini, M., Grabovic, E., Guiggiani, M.: A Lie group-based race car model for systematic trajectory optimization on 3d tracks (2023). https://arxiv.org/abs/2302.09879. https://doi.org/10.48550/ARXIV.2302.09879
43. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
44. Mueller, A., Maisser, P.: A Lie-group formulation of kinematics and dynamics of constrained MBS and its application to analytical mechanics. Multibody Syst. Dyn. **9**(4), 311–352 (2003)
45. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, Heidelberg (2007)
46. Pacejka, H.: Tyre and Vehicle Dynamics. Tyre and Vehicle Dynamics. Butterworth-Heinemann, Oxford (2002)
47. Guiggiani, M.: The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars. Springer, Berlin (2018)
48. Betts, J.T.: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming. Advances in Design and Control. SIAM, Philadelphia (2010)
49. Lot, R., Biral, F.: A curvilinear abscissa approach for the lap time optimization of racing vehicles. IFAC Proc. Vol. **47**(3), 7559–7565 (2014)
50. Nocedal, J., Wright, S.J.: Numerical Optimization, 2e edn. Springer, New York (2006)
51. Andersson, J.: A general-purpose software framework for dynamic optimization. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT-SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium (2013)