CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed eco-driving control of a platoon of electric vehicles through Riccati recursion

Distributed eco-driving control of a platoon of electric vehicles through Riccati recursion

Rémi Lacombe, Sébastien Gros, Nikolce Murgovski, and Balázs Kulcsár

***Abstract*—This paper presents a distributed optimization procedure for the cooperative eco-driving control problem of a platoon of electric vehicles subject to safety and travel time constraints. Individual optimal trajectories are generated for each platoon member to account for heterogeneous vehicles and for the road slope. By rearranging the problem variables, the Riccati recursion can be applied along the chain-like structure of the platoon and be used to solve the problem by repeatedly transmitting information up and down the platoon. Since each vehicle is only responsible for its own part of the computations, the proposed control strategy is privacy-preserving and could therefore be deployed by any group of vehicles to form a platoon spontaneously while driving. The energy efficiency of this control strategy is evaluated in numerical experiments for platoons of electric trucks with different masses and rated motor powers.**

***Index Terms*—Distributed optimal control, Riccati recursion, platooning, eco-driving, electric vehicles, nonlinear programming.**

I. Introduction

CONTRARY to other sectors, the greenhouse gases emissions of the transport sector have continued to increase in recent years, reaching 24% of direct global CO_2 emissions in 2018 [1]. At the same time, the freight demand is expected to triple between 2015 and 2050 [2]. In this report, the International Transport Forum emphasizes the need for a massive deployment of zero-emission propulsion for heavy-duty vehicles, such as with electric batteries, in order to reach the internationally agreed emission targets. Energy efficiency improvements are also listed as a significant mitigation measure [2].

In this context, truck platooning offers a promising way to reduce the energy consumption of road freight transport without requiring any structural change. By driving in close succession, the overall drag force acting on the trucks can be decreased. This has the effect of reducing the energy needed to maintain a certain speed, and field experiments indicate that sizeable energy savings can be achieved as a result [3]–[5]. However, safety must be guaranteed when driving vehicles at a close distance, meaning that the trade-off between drag reduction and collision avoidance should be carefully addressed when operating vehicle platoons.

The paper has been supported by the Swedish Energy Agency through the project "Operational Network Energy Management for Electrified Buses" (46365-1) and IRIS with Transport Area of Advance (Chalmers University of Technology). *(Corresponding author: Rémi Lacombe.)*

R. Lacombe, N. Murgovski, and B. Kulcsár are with the Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: lacombe@chalmers.se; nikolce.murgovski@chalmers.se; kulcsar@chalmers.se).

S. Gros is with the Department of Engineering Cybernetics, NTNU, Trondheim, Norway (e-mail: sebastien.gros@ntnu.no).

Control methods for vehicle platoons have historically focused on string stability, i.e. the attenuation of deviations on position and speed along a string of vehicles, rather than energy or fuel optimality [6]–[8]. The latter has sometimes been addressed indirectly by using vehicle proximity as a proxy for energy efficient operation [9]. But as emphasized in this last work, the road slope can significantly impact the energy savings obtained through platooning for heavy-duty vehicles. In a hilly terrain, tracking a constant spacing or time gap between successive vehicles is not energy-optimal if it requires some of the vehicles to use their friction brakes in order to avoid collisions.

Optimal driving profiles can be generated for one or several vehicles by forming and solving an *optimal control problem* (OCP) including external factors like the road slope, a procedure generally known as *eco-driving* [10]. In the case of a truck platoon, the overall energy consumption can be minimized by either following a single common optimal trajectory for the entire platoon [11], or by computing and tracking the individual optimal trajectory of each member [12], [13]. Whichever approach is used, the optimal control problem is always solved centrally in the works cited above. In fact, there are currently no works in the literature proposing fully distributed algorithms to solve the eco-driving control problem of a truck platoon, to the best of our knowledge.

To remedy this, we take a different perspective on this problem in this work and use the concept of *Riccati recursion* to solve it in a distributed fashion. The Riccati recursion is a matrix factorization scheme traditionally deployed for solving unconstrained linear-quadratic (LQ) OCPs faster [14]–[16], i.e. problems with linear dynamics and a quadratic cost function. Such problems have a particular stage-wise structure and often arise in *model predictive control* (MPC) [17].

If inequality constraints are also considered, which is the case in most MPC applications, then constrained LQ OCPs must be solved, a general form of which is given in [18] for instance. In the rest of this paper, we refer to the particular structure of this family of problems as the *standard LQ OCP structure*. Constrained LQ OCPs are often solved with second-order iterative methods in practice, where an unconstrained LQ OCP needs to be solved at each iteration [14]–[16]. Solving these intermediate LQ OCP subproblems is in fact often the computational bottleneck of these solution methods, which is why the Riccati recursion is today routinely used in specialized solvers for LQ OCPs [19].

In this paper, we make the argument that the backward and forward sweeps needed in the Riccati recursion can be carried out between successive agents instead of successive

stages when solving unconstrained LQ OCPs. By doing so, we show that second-order solution methods for problems with a standard LQ OCP structure can be fully distributed over agents organized in a chain-like structure.

Second-order optimization methods are known for their strong convergence properties: they can achieve a quadratic convergence rate when taking full Newton steps [20]. As a result, they require less iterations than first-order methods, which results in reduced communication needs in the case of distributed algorithms. In settings where communications are slow or expensive this grants a clear advantage to second-order methods. However, second-order methods are in general much more difficult to distribute fully since they usually require specific problem structures with sparse coupling, see e.g. [21], [22], and references therein.

The main contribution of this paper is to propose a distributed solution procedure for the cooperative eco-driving control problem of a vehicle platoon. This procedure combines the greater convergence speed of second-order optimization methods with fully distributed computations. The absence of centralized computation requirements means that every vehicle is ultimately responsible for computing its own optimal trajectory and needs not trust a third party with it. The local problem information of each vehicle can be kept private as it does not have to be shared directly with the other vehicles, meaning that this control method is *privacy-preserving*. Therefore, in the case of heavy-duty vehicles, this method could even be deployed on vehicles belonging to competing transportation companies. In addition, the proposed procedure exploits the chain-like structure of a platoon since it only requires each vehicle to communicate with its direct neighbors. This opens the door to the use of high-bandwidth communications (e.g. optical or microwave communication) while also reducing the risk of interference or packet loss. The distributed aspect of the solution procedure is achieved through the use of the Riccati recursion. To the best of our knowledge, no other study has applied the Riccati recursion scheme to the end of physically distributing computations between agents organized in succession.

The originality of the proposed procedure is also that it solves a detailed formulation of the eco-driving platoon control problem. Namely, this formulation includes the influence of both the road slope and aerodynamic drag reduction on the energy consumption of the vehicles. Furthermore, each vehicle is allowed to compute its own optimal trajectory, so that headways between successive vehicles are not constrained to a fixed target value but allowed to vary along the route in order to save energy and to account for heterogeneous vehicles. This paper is the first work that we are aware of in the literature which proposes a fully distributed solution procedure for a formulation of the cooperative eco-driving platoon control problem including all the features described above. It is also the first one to explicitly study the eco-driving problem for a platoon of heavy-duty electric vehicles. Note that even though the present study focuses on truck platoons, the proposed method could in principle be deployed on platoons of different vehicle types, such as cars or buses.

The article is organized as follows. The vehicle platoon model is presented in Section II, together with the formulation of the eco-driving control problem. Section III introduces a *sequential quadratic programming* (SQP) algorithm to solve this problem, and Section IV a *primal-dual interior point* (PDIP) algorithm based on the Riccati recursion to solve the SQP subproblems. These two algorithms are detailed further in Section V. In Section VI, simulation results are shown and commented, while some key properties of the proposed control method are discussed in Section VII. Finally, the paper closes on some concluding remarks in Section VIII.

II. Modeling

Consider a platoon of n electric trucks traveling on a hilly highway, where the platoon leader has index 1. These trucks need not have the same physical properties, i.e. the platoon can be heterogeneous. The goal is to minimize the overall energy consumption of the platoon over a spatial horizon of length s_f. This horizon is assumed to start at the current position of the platoon leader, where $s = 0$, and to be common to all the trucks in the platoon. The problem of controlling the trucks before they reach $s = 0$ is left outside the scope of this paper.

A. Longitudinal Dynamics

In this paper, the dynamics of the vehicles are modeled in the space domain, i.e. as functions of the position s. For instance, for any truck with index i in the platoon, its velocity and travel time at position s are written $v_i(s)$ and $t_i(s)$, respectively. Note that this is a common practice in the eco-driving literature as it removes some nonlinearities from the vehicle longitudinal dynamics equations, compared with a formulation in the time domain [12], [13], [23]–[25].

It is assumed that the platoon has a constant preferred cruising speed $\bar{v}$ when traveling [12]. However, due to limited motor power and large vehicle mass, some of the trucks may not be able to reach $\bar{v}$ in steep uphill sections of the road. Instead, a reference speed function $v_{\text{ref},i}$ is used for each truck i of the platoon. This function takes value $\bar{v}$ on most of the horizon, with the exception of steep uphills where it is the maximum speed that truck i can achieve.

Instead of simply tracking the reference speed, each truck i has the freedom to adjust its velocity around $v_{\text{ref},i}$, in order to minimize the energy consumption of the platoon. The speed constraints for each truck i can be formulated as:

$$v_{\text{min},i}(s) \leq v_i(s) \leq v_{\text{max},i}(s), \quad \forall s \in [0, s_f], \tag{1}$$

where the bounds $v_{\text{min},i}$ and $v_{\text{max},i}$ can encode information on the legal speed limit or the minimum allowed speed for example. Note that these bounds can be designed independently from the reference speed, but they are assumed to satisfy $v_{\text{min},i}(s) \leq v_{\text{ref},i}(s)$ and $v_{\text{ref},i}(s) \leq v_{\text{max},i}(s)$ at each position s.

In order to not slow down the trucks too much during their driving mission, they are required to reach the end of the horizon as fast as they would by following their reference speed. Let $t_{\text{ref},i}$ be the reference travel time for truck i:

$$t_{\text{ref},i} = \int_0^{s_f} \frac{ds}{v_{\text{ref},i}(s)}. \tag{2}$$

The constraint on the terminal travel time of truck i can then be written as:

$$t_i(s_f) \leq t_i(0) + t_{\text{ref},i}, \tag{3}$$

where $t_i(0)$ denotes the time at which truck i reaches the start of the horizon. In the case of a heterogeneous platoon, it can happen that a powerful vehicle trails behind a less powerful one. In this case, the travel time constraints must be modified to avoid infeasibility. Therefore, for every index pair $\{i-1, i\}$, $i \in \mathbb{I}_{[2,n]}$, the reference travel time $t_{\text{ref},i}$ is set to $t_{\text{ref},i-1}$ in case $t_{\text{ref},i} < t_{\text{ref},i-1}$.

In addition, the platoon vehicles should never be too close to each other on the road in order to avoid any possible collision. These safety constraints are enforced by specifying a minimum time headway t_h at which trucks are allowed to be from each other [12]:

$$t_i(s) - t_{i-1}(s) \geq t_h, \quad \forall s \in [0, s_f], \tag{4}$$

for every truck with index $i \geq 2$.

In order to remove additional nonlinearities from the longitudinal dynamics of truck i, the kinetic energy $E_i(s) = (1/2)m_i v_i(s)^2$ is used as a state variable, together with the travel time t_i. This is also a common modeling choice when working in the space domain, and a complete treatment of the derivation of the longitudinal equations in that case can be found in [23]. We now directly state these equations for a single truck i:

$$\frac{\mathrm{d}E_i(s)}{\mathrm{d}s} = F_{m,i}(s) - F_{b,i}(s) - F^0_{\text{drag},i}(E_i) - m_i g(\sin\theta(s) + c_r \cos\theta(s)), \tag{5a}$$

$$\frac{\mathrm{d}t_i(s)}{\mathrm{d}s} = \sqrt{\frac{m_i}{2E_i(s)}}, \tag{5b}$$

where m_i is its mass, $F_{m,i}$ is the longitudinal force, $F_{b,i}$ is the braking force from the mechanical friction brakes, $F_{\text{drag},i}$ is the aerodynamic drag, θ is the road gradient, and c_r is the rolling resistance coefficient. We define the state vector of truck i as $x_i = [E_i, t_i]^\top$ and the control input vector as $u_i = [F_{m,i}, F_{b,i}]^\top$.

It is assumed that the braking force is bounded below by a constant value $F_{b,\min}$, such that:

$$F_{b,\min} \leq F_{b,i}(s) \leq 0, \quad \forall s \in [0, s_f]. \tag{6}$$

Likewise, the longitudinal force $F_{m,i}$ is bounded above and below by functions of the vehicle speed, due to the power limitations of the electric motor [26]. These constraints can in general be expressed with non-convex functions $F_{m,\min}$ and $F_{m,\max}$ as:

$$F_{m,\min}(E_i) \leq F_{m,i}(s) \leq F_{m,\max}(E_i), \quad \forall s \in [0, s_f]. \tag{7}$$

The longitudinal force constraints of the electric motor (EM) model used in this paper are displayed in Fig. 1. Notice that the regenerative braking feature of the EM is modeled by the negative longitudinal force values in this figure.

In case truck i is not surrounded by any vehicle, the aerodynamic drag acting on it has the form:

$$F^0_{\text{drag},i}(E_i) = \frac{\rho_a c_d A_{f,i}}{m_i} E_i(s), \tag{8}$$

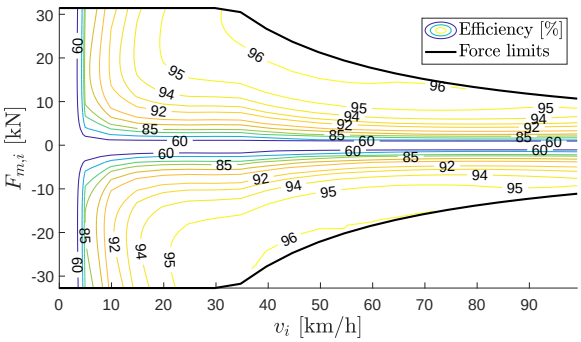


Fig. 1: Efficiency map of the EM model used, given as a function of vehicle speed and longitudinal force. The black lines denote the longitudinal force constraints.

where ρ_a is the air density, c_d is the aerodynamic drag coefficient, and $A_{f,i}$ is the frontal area of truck i [26].

B. Drag Reduction Model

In the case of a truck platoon, each platoon member is surrounded by other vehicles in its lane. Consequently, the drag force experienced by each truck is reduced. It is assumed here that the drag reduction is only caused by the preceding truck, since it is the one which contributes the most. As a result, the platoon leader does not benefit from any drag reduction, and the drag force acting on it can be modeled by (8) directly: $F_{\text{drag},1}(E_1) = F^0_{\text{drag},1}(E_1)$. For the other trucks, the magnitude of the drag reduction depends primarily on the distance with the preceding truck. Using a simple drag reduction model [27], the drag force acting on each truck i, with $i \geq 2$, can be modeled as:

$$F_{\text{drag},i}(E_i, d_i) = F^0_{\text{drag},i}(E_i)(1 - f_d(d_i)), \tag{9}$$

where the function f_d is:

$$f_d(d_i) = \frac{c_1}{c_2 + d_i(s)}. \tag{10}$$

In these expressions, c_1 and c_2 are drag reduction coefficients, and d_i is the distance between truck i and the preceding truck in the platoon. However, the inter-vehicular distances cannot be obtained directly in our modeling framework, since the truck positions are not state variables. An alternative is to approximate d_i as an affine function of the travel time. Under the assumption that the truck speeds do not deviate significantly from $\bar{v}$, we have:

$$d_i(s) \approx \bar{v}(t_i(s) - t_{i-1}(s)) - L_{i-1}, \tag{11}$$

where L_{i-1} is the length of the truck with index $i-1$, which is the truck preceding truck i in the platoon. The complete derivation of this approximated expression, together with a discussion of its validity, can be found in [12]. In what follows, this approximation is used when computing the drag reduction. The complete drag force terms $F_{\text{drag},i}$ are now used in the longitudinal dynamics (5) instead of $F^0_{\text{drag},i}$.

C. Optimal Control Formulation

The cooperative eco-driving problem for the truck platoon can then be formulated as:

$$\min_{x(s),u(s)} \quad \sum_{i=1}^{n} \int_0^{s_f} \frac{P_{b,i}(E_i, F_{m,i})}{v_i(s)} ds, \tag{12a}$$

$$\text{s.t.} \quad x_i(0) = x_i^0, \quad \forall i \in \mathbb{I}_{[1,n]}, \tag{12b}$$

$$\frac{\mathrm{d}x_1(s)}{\mathrm{d}s} = f(x_1, u_1), \tag{12c}$$

$$h(x_1, u_1) \leq 0, \tag{12d}$$

$$\forall i \in \mathbb{I}_{[2,n]}:$$

$$\frac{\mathrm{d}x_i(s)}{\mathrm{d}s} = f(x_i, u_i, x_{i-1}), \tag{12e}$$

$$h(x_i, u_i, x_{i-1}) \leq 0, \tag{12f}$$

where $P_{b,i}$ is the internal battery power of truck i, and x_i^0 gathers its initial kinetic energy and initial travel time. The right-hand side of the longitudinal dynamics (5) has been noted as a function f, for the sake of simplicity. Similarly, all the inequality constraints defined above, i.e. (1), (3), (4), (6), (7), have been gathered in a common inequality constraint function h.

Note that the equality and inequality constraints for the platoon leader have been written separately from those of the other trucks in (12). This is to emphasize the special role of the leader in the coupling structure of the platoon OCP. Indeed, the platoon leader does not benefit from any drag reduction in our model, nor does it have to maintain a minimum headway with preceding platoon members. As a result, there is no inter-truck coupling in equations (12c) and (12d), contrary to (12e) and (12f). This observation becomes important in the next section, when the problem has to be solved in a distributed fashion.

The expression for the internal battery power in (12) can be derived by assembling an EM and a battery model, as is done in [13] for instance. In this paper, we proceed along similar lines, and the efficiency map of the EM model used is presented in Fig. 1 for reference.

Remark 1: In most practical applications, the horizon length s_f in OCP (12) will likely be shorter than the total length of the driving mission of the platoon, mostly due to the numerical complexity of solving this problem over long horizons. In this case, the cooperative eco-driving problem must be solved repeatedly in a receding horizon fashion, i.e. as part of an MPC. OCP (12) would then be solved online at each sampling instant of the MPC. Since solving optimization problems tends to be the bottleneck in any MPC implementation, we focus our attention on the problem of solving OCP (12) in the rest of this paper.

Remark 2: Solving OCP (12) only provides energy-optimal reference trajectories for the vehicles, meaning that a lower control layer might be needed in a practical implementation in order to reject disturbances and track these trajectories. A distributed MPC scheme such as the one proposed in [8] could for example be deployed to that end. In this context, some of the constraints in (12) should be modeled as soft constraints instead, in order to avoid running into feasibility issues. In particular, the lower speed bound in (1) and the terminal travel time constraint (3) may need to be relaxed. The design of this lower control layer is left outside the scope of this paper, however.

D. Formation of an NLP

In order to be able to treat (12) numerically, a direct method is deployed to assemble a *nonlinear program* (NLP) which approximates the OCP. This is done through a piecewise input parametrization and a multiple shooting formulation of the dynamics [28]. Assume that the spatial horizon $[0, s_f]$ is partitioned into N intervals of equal size, and that the state and control input vectors of truck i at the k-th grid point are noted $x_{i,k}$ and $u_{i,k}$, respectively, with $x_{i,k} = [E_{i,k}, t_{i,k}]^\top$ and $u_{i,k} = [F_{m,i,k}, F_{b,i,k}]^\top$. The following NLP can then be formulated:

$$\min_{X,U} \quad \sum_{i=1}^{n} \sum_{k=0}^{N-1} J_i(x_{i,k}, u_{i,k}) \tag{13a}$$

$$\text{s.t.} \quad x_{i,0} = x_i^0, \quad \forall i \in \mathbb{I}_{[1,n]}, \tag{13b}$$

$$\forall k \in \mathbb{I}_{[0,N-1]}:$$

$$x_{1,k+1} = F_k(x_{1,k}, u_{1,k}), \tag{13c}$$

$$h_k(x_{1,k}, u_{1,k}) \leq 0, \tag{13d}$$

$$\forall i \in \mathbb{I}_{[2,n]}:$$

$$x_{i,k+1} = F_k(x_{i,k}, u_{i,k}, x_{i-1,k}), \tag{13e}$$

$$h_k(x_{i,k}, u_{i,k}, x_{i-1,k}) \leq 0, \tag{13f}$$

where J_i carries out the numerical integration of the power consumed by truck i on each interval, and F_k the numerical integration of the dynamics (12c) or (12e) of each truck between the k-th and $(k+1)$-th grid points. Any numerical integrator may be chosen to implement these functions, but the classical Runge-Kutta method (RK4) has been used in this work due to its good accuracy. In addition, the initial conditions (12b) are expressed as (13b), while the function h_k is used to express the inequality constraints (12d) and (12f) at the k-th grid point.

Note that (13) is a non-convex NLP with the modeling choices above. In what follows, a solution procedure is proposed to solve this NLP in a distributed fashion.

III. Presentation of the Distributed SQP Algorithm

In this section, we show how NLP (13) can be solved by deploying the SQP method [20]. However, note that the procedure presented below could in principle be adapted to other types of second-order optimization methods, like interior point methods, since it boils down to a particular factorization of the KKT systems which appear when using such methods. The crux of this procedure is to organize the problem agent-by-agent instead of stage-by-stage as in a typical optimal control application, while keeping a standard LQ OCP structure. This way, standard methods like the Riccati recursion can still be deployed to solve it. This is the topic of next section.

In the rest of this paper, the subscript i denotes local information to which only vehicle i has access, except if explicitly stated otherwise. The rationale for indexing variables

in this way is to specify exactly which information needs to be exchanged between the vehicles when solving the cooperative eco-driving problem in a distributed fashion.

A. Sequential Quadratic Programming

The SQP method operates by iteratively forming a local *quadratic program* (QP) approximation of (13) around the current iterate of the primal-dual variables [20], which are gathered in the vector Y_i hereafter. The Newton direction at the current iterate can be computed by solving this quadratic subproblem, and the next iterate is generated by taking a step along this direction. Note that the issue of distributed backtracking is left outside the scope of this paper, and it is assumed that full steps are taken by the SQP algorithm. Furthermore, Hessian regularization is deployed to improve the quality of these steps [20].

In order for the Riccati recursion to be deployed when solving the SQP subproblems, these problems must have the standard LQ OCP structure. However, the inter-truck coupling of some of the state variables in the constraints (13e) and (13f) introduces some non-diagonal blocks in the Hessian of the Lagrangian of (13), which is used to generate the subproblems. This results in a non-separable Hessian and in subproblems which lack the desired sparse structure.

For this reason, we will now focus on a modified version of the quadratic subproblems where the Hessian coupling blocks have been removed by being set to zero. The modified Hessian is then block diagonal, and each of its block can be regularized independently by each vehicle. Note that, in practice, the coupling blocks have been observed to have a small magnitude compared with that of the diagonal blocks of the Hessian. For instance, the maximum matrix 2-norm observed for an individual coupling block in the simulation section was around 10^{-1}, whereas the norm of individual diagonal blocks was consistently in the order of 10^2. This ratio 3 in the order of magnitude of the different Hessian block types indicates that the modified Hessian likely constitutes a very good approximation of the full Hessian matrix. Therefore, it can be expected that the SQP iterates should not differ significantly from those obtained with the full Hessian.

Since the optimization problem (13) is a non-convex NLP in general, no global convergence result can be stated for the SQP method. However, the SQP method is known to have robust convergence properties, even from remote starting points [20]. It should also be noted that taking steps with the modified Hessian instead of the full Hessian does not impact the convergence of the SQP method locally. The iterates will converge to the same local minimum as with the full Hessian, provided that the method is initialized close enough to the local minimum in question [20]. This, in turn, can usually be achieved with a sensible initial guess.

B. Reorganization of the SQP subproblems

In the rest of this section, we discuss how a search direction for the SQP algorithm can be computed distributively at each iteration by solving the quadratic subproblem described above. At the current iterate, this subproblem can be written as:

$$\min_{\Delta X,\Delta U} \sum_{i=1}^{n}\sum_{k=0}^{N-1} \frac{1}{2}\begin{bmatrix}\Delta x_{i,k}\\ \Delta u_{i,k}\end{bmatrix}^\top \begin{bmatrix} q_{i,k} & m_{i,k}\\ m_{i,k}^\top & r_{i,k}\end{bmatrix}\begin{bmatrix}\Delta x_{i,k}\\ \Delta u_{i,k}\end{bmatrix} + \begin{bmatrix} l_{x,i,k}\\ l_{u,i,k}\end{bmatrix}^\top \begin{bmatrix}\Delta x_{i,k}\\ \Delta u_{i,k}\end{bmatrix} \tag{14a}$$

$$\text{s.t.}\quad \Delta x_{i,0} = c_i^0, \quad \forall i \in \mathbb{I}_{[1,n]}, \tag{14b}$$

$$\forall k \in \mathbb{I}_{[0,N-1]}:$$

$$\Delta x_{1,k+1} = a_{1,k}\Delta x_{1,k} + b_{1,k}\Delta u_{1,k} + c_{1,k}, \tag{14c}$$

$$h_{x,1,k}\Delta x_{1,k} + h_{u,1,k}\Delta u_{1,k} + d_{1,k} \le 0, \tag{14d}$$

$$\forall i \in \mathbb{I}_{[2,n]}:$$

$$\Delta x_{i,k+1} = a_{i,k}\Delta x_{i,k} + b_{i,k}\Delta u_{i,k} + c_{i,k} + \hat{a}_{i,k}\Delta x_{i-1,k}, \tag{14e}$$

$$h_{x,i,k}\Delta x_{i,k} + h_{u,i,k}\Delta u_{i,k} + d_{i,k} + \hat{h}_{x,i,k}\Delta x_{i-1,k} \le 0, \tag{14f}$$

where the decision variables $\Delta x_{i,k}$ and $\Delta u_{i,k}$ are the search directions in the state and input variables of truck i, respectively. All the parameters appearing in the objective function and in the constraints are obtained in a traditional SQP fashion [20], and are not discussed further here.

In order to be able to exploit the chain-like structure of the platoon and solve the problem in a distributed fashion, QP (14) is rewritten in a more compact form by arranging the decision variables over successive trucks rather than successive grid points:

$$\min_{X,U} \sum_{i=1}^{n} \frac{1}{2}\begin{bmatrix}X_i\\ U_i\end{bmatrix}^\top \begin{bmatrix} Q_i & M_i\\ M_i^\top & R_i\end{bmatrix}\begin{bmatrix}X_i\\ U_i\end{bmatrix} + \begin{bmatrix} L_{x,i}\\ L_{u,i}\end{bmatrix}^\top \begin{bmatrix}X_i\\ U_i\end{bmatrix} \tag{15a}$$

$$\text{s.t.}\quad X_1 = B_1U_1 + C_1, \tag{15b}$$

$$H_{x,1}X_1 + H_{u,1}U_1 + D_1 \le 0, \tag{15c}$$

$$\forall i \in \mathbb{I}_{[2,n]}:$$

$$X_i = A_iX_{i-1} + B_iU_i + C_i, \tag{15d}$$

$$H_{x,i}X_i + H_{u,i}U_i + D_i + \hat{H}_{x,i}X_{i-1} \le 0. \tag{15e}$$

The decision variables for truck i have been gathered in the vectors X_i and U_i, such that $X_i = [\Delta x_{i,0}, ..., \Delta x_{i,N}]^\top$ and $U_i = [\Delta u_{i,0}, ..., \Delta u_{i,N-1}]^\top$. Note that Δ has been dropped from these notations for the sake of simplicity. The weights in the objective function have been assembled into the matrices Q_i, M_i, R_i and the vectors $L_{x,i}$, $L_{u,i}$ for truck i, and the inequality constraints into the matrices $H_{x,i}$, $H_{u,i}$, $\hat{H}_{x,i}$ and the vector D_i. By using the stage-wise dynamics (14c), (14e), and the initial conditions (14b), the state vector of truck i can be expressed as a function of the control inputs of i and of the state variables of the preceding truck. The matrices A_i, B_i and the vector C_i are used to write the dynamics of truck i in this compact form.

Contrary to a typical MPC-type problem, which is usually arranged from one grid point to the next, QP (15) has been arranged in a truck-by-truck fashion, and has a structure which

is slightly different from a standard LQ OCP structure as a result. First, state vectors with different indices appear in the inequality constraints (15e). Second, there is some coupling between X_i and U_i in the objective function, whereas only coupling between X_{i-1} and U_i would appear in a standard LQ OCP structure. This last point might be harder to see given our choice of indices, and it is useful to remember here that X_{i-1} appears together with U_i in the right-hand side of the dynamics (15d): the states of truck i are determined by its own control inputs and by the states of the preceding truck $i-1$.

Fortunately, QP (15) can easily be transformed into a QP with a standard LQ OCP structure. This is done by injecting the dynamics (15d) both in the inequality constraints (15e) and in the objective coupling terms mentioned earlier in (15a). Likewise, the leader dynamics (15b) are injected in (15c) and in (15a). By following these steps, a modified QP with a standard LQ OCP structure is obtained:

$$\begin{aligned}\min_{X,U}\ & \sum_{i=2}^{n}\left(\frac{1}{2}\begin{bmatrix}X_{i-1}\\U_i\end{bmatrix}^\top\begin{bmatrix}\tilde{Q}_{i-1} & \tilde{M}_i\\ \tilde{M}_i^\top & \tilde{R}_i\end{bmatrix}\begin{bmatrix}X_{i-1}\\U_i\end{bmatrix}\right.\\ &\qquad\left.+\begin{bmatrix}\tilde{L}_{x,i-1}\\\tilde{L}_{u,i}\end{bmatrix}^\top\begin{bmatrix}X_{i-1}\\U_i\end{bmatrix}\right)+\frac{1}{2}U_1^\top\tilde{R}_1U_1\\ &\qquad+\tilde{L}_{u,1}^\top U_1+\frac{1}{2}X_n^\top\tilde{Q}_nX_n+\tilde{L}_{x,n}^\top X_n \qquad (16a)\\ \text{s.t.}\ & X_1=B_1U_1+C_1, \qquad (16b)\\ & \tilde{H}_{u,1}U_1+\tilde{D}_1\le 0, \qquad (16c)\\ & \forall i\in\mathbb{I}_{[2,n]}:\\ & X_i=A_iX_{i-1}+B_iU_i+C_i, \qquad (16d)\\ & \tilde{H}_{x,i}X_{i-1}+\tilde{H}_{u,i}U_i+\tilde{D}_i\le 0, \qquad (16e)\end{aligned}$$

where:

$$\begin{aligned}&\tilde{Q}_i=Q_i,\quad \tilde{R}_i=R_i+B_i^\top M_i+M_i^\top B_i, && i\ge 1, && (17a)\\ &\tilde{M}_i=A_i^\top M_i, && i\ge 2, && (17b)\\ &\tilde{L}_{x,i}=L_{x,i},\quad \tilde{L}_{u,i}=L_{u,i}+M_i^\top C_i, && i\ge 1, && (17c)\\ &\tilde{H}_{x,i}=\hat{H}_{x,i}+H_{x,i}A_i, && i\ge 2, && (17d)\\ &\tilde{H}_{u,i}=H_{u,i}+H_{x,i}B_i, && i\ge 1, && (17e)\\ &\tilde{D}_i=D_i+H_{x,i}C_i, && i\ge 1. && (17f)\end{aligned}$$

The following proposition establishes a relation between the optimal solutions of these two QPs. Its proof is given in Appendix A.

Proposition 1: The modified QP (16) has the same primal solution as the original QP (15). In addition, the dual solution of (15) can be obtained as:

$$\begin{aligned}\mu_i^* &= \mu_i^{\text{mod}}, && (18a)\\ \lambda_i^* &= \lambda_i^{\text{mod}}+H_{x,i}^\top\mu_i^{\text{mod}}+M_iU_i^*, && (18b)\end{aligned}$$

where λ_i^{mod} and μ_i^{mod} are the optimal dual variables associated with the equality and inequality constraints of (16) corresponding to truck i, respectively. The optimal dual variables λ_i^* and μ_i^* are defined in a similar way for (15).

From Proposition 1, it follows that SQP search directions can be obtained by repeatedly solving (16). Solving this QP in a distributed fashion is the topic of the next section.

IV. Presentation of the Distributed PDIP Algorithm

In this section, we present how a PDIP method can be deployed to solve QP (16). We consider a standard PDIP algorithm where the search direction is computed at each iteration by solving the perturbed KKT conditions of (16) [20]. It is shown how the resulting KKT system can be solved in a distributed fashion with the Riccati recursion. In particular, we detail the exact computations and information exchanges needed to do so.

For each truck i, the primal-dual variables at the j-th PDIP iteration are noted $(U_i^{[j]}, X_i^{[j]}, \lambda_i^{[j]}, \mu_i^{[j]})$ and the slack variables $\sigma_i^{[j]}$. All these variables are assembled in $Z_i^{[j]}$ hereafter. The '+' superscript is used to denote the PDIP search direction.

A. Construction of the KKT System

The KKT system to solve at each PDIP iteration can be modified to have the same structure as that of an unconstrained LQ OCP. By applying a block-elimination procedure, which is standard in interior point methods, the slack variables and the dual variables for the inequality constraints can be successively removed from the KKT system. More details on how this compact form is obtained can be found in [18]. At the j-th PDIP iteration, the search direction can then be computed by solving the compact KKT system:

$$T^{[j]}\bar{Z}^{+[j]}=-\bar{r}^{[j]}, \qquad (19)$$

where:

$$T^{[j]}=\begin{bmatrix}\bar{R}_1^{[j]} & B_1^\top & & & & & \\ B_1 & & -I & & & & \\ & -I & \bar{Q}_1^{[j]} & \bar{M}_2^{[j]} & A_2^\top & & \\ & & \bar{M}_2^{[j]\top} & \bar{R}_2^{[j]} & B_2^\top & & \\ & & A_2 & B_2 & & -I & \\ & & & & -I & \ddots & \ddots \\ & & & & & \ddots & \bar{Q}_n\end{bmatrix} \qquad (20a)$$

$$\bar{Z}^{+[j]}=\begin{bmatrix}U_1^{+[j]} & \lambda_1^{+[j]} & X_1^{+[j]} & \dots & X_n^{+[j]}\end{bmatrix}^\top \qquad (20b)$$

$$\bar{r}^{[j]}=\begin{bmatrix}\bar{r}_{u,1}^{[j]} & \bar{r}_{\lambda,1} & \bar{r}_{x,1}^{[j]} & \dots & \bar{r}_{x,n}\end{bmatrix}^\top \qquad (20c)$$

with:

$$\begin{aligned}&\bar{Q}_n=\tilde{Q}_n,\quad \bar{r}_{x,n}=\tilde{L}_{x,n}, && && (21a)\\ &\bar{Q}_i^{[j]}=\tilde{Q}_i+\tilde{H}_{x,i+1}^\top\Sigma_{i+1}^{[j]}\tilde{H}_{x,i+1}, && i\le n-1, && (21b)\\ &\bar{M}_i^{[j]}=\tilde{M}_i+\tilde{H}_{x,i}^\top\Sigma_i^{[j]}\tilde{H}_{u,i}, && i\ge 2, && (21c)\\ &\bar{R}_i^{[j]}=\tilde{R}_i+\tilde{H}_{u,i}^\top\Sigma_i^{[j]}\tilde{H}_{u,i}, && i\ge 1, && (21d)\\ &\bar{r}_{x,i}^{[j]}=\tilde{L}_{x,i}+\tilde{H}_{x,i+1}^\top\big(\tau^{[j]}\Lambda_{i+1}^{[j]}\\ &\qquad+\Sigma_{i+1}^{[j]}\tilde{D}_{i+1}+\mu_{i+1}^{[j]}\big), && i\le n-1, && (21e)\\ &\bar{r}_{u,i}^{[j]}=\tilde{L}_{u,i}+\tilde{H}_{u,i}^\top\big(\tau^{[j]}\Lambda_i^{[j]}\\ &\qquad+\Sigma_i^{[j]}\tilde{D}_i+\mu_i^{[j]}\big), && i\ge 1, && (21f)\\ &\bar{r}_{\lambda,i}=C_i,\quad \Lambda_i^{[j]}=\operatorname{diag}\big(\sigma_i^{[j]}\big)^{-1}, && i\ge 1, && (21g)\\ &\Sigma_i^{[j]}=\operatorname{diag}\big(\mu_i^{[j]}\big)\Lambda_i^{[j]}, && i\ge 1, && (21h)\end{aligned}$$

where $\tau^{[j]}$ is the barrier parameter at the j-th iteration [18]. The Newton step components for the variables which have previously been eliminated can be recovered as:

$$\mu_1^{+[j]} = \mu_1^{[j]} + \tau^{[j]}\Lambda_1^{[j]}e + \Sigma_1^{[j]}\big(\tilde{H}_{u,1}U_1^{+[j]} + \tilde{D}_1\big), \quad (22a)$$

$$\mu_i^{+[j]} = \mu_i^{[j]} + \tau^{[j]}\Lambda_i^{[j]}e + \Sigma_i^{[j]}\big(\tilde{H}_{x,i}X_i^{+[j]} + \tilde{H}_{u,i}U_i^{+[j]} + \tilde{D}_i\big), \ i \geq 2, \quad (22b)$$

$$\sigma_i^{+[j]} = \sigma_i^{[j]} - \big(\Sigma_i^{[j]}\big)^{-1}\big(\mu_i^{+[j]} - \tau^{[j]}\Lambda_i^{[j]}e\big), \quad i \geq 1, \quad (22c)$$

where $e = [1, 1, ..., 1]^\top$.

It can be observed that the problem matrices and vectors in (21) have been modified compared with their original form in (16)-(17) in order to obtain the compact KKT system (19) where the KKT matrix (20a) is banded. This system now has the same structure as an unconstrained LQ OCP, and the Riccati recursion can be deployed to solve it.

Note that the KKT matrix (20a) is non-singular if the matrices:

$$\begin{bmatrix} \bar{Q}_{i-1}^{[j]} & \bar{M}_i^{[j]} \\ \bar{M}_i^{[j]\top} & \bar{R}_i^{[j]} \end{bmatrix}, \quad i \geq 2, \quad (23)$$

and $\bar{Q}_n^{[j]}$ are positive semidefinite, and the matrices $\bar{R}_i^{[j]}, i \geq 1$, are positive definite [16]. These conditions generally hold in practice due to the Hessian regularization carried out before each SQP iteration, but an extra regularization step may be applied to enforce them if needed. This guarantees that system (19) has a unique solution.

Remark 3: Due to the bandwidth of the banded KKT matrix (20a), the complexity of solving the KKT system (19) with the Riccati recursion is $\mathcal{O}(N^3 n)$. This difference with the $\mathcal{O}(Nn^3)$ complexity usually given for the Riccati recursion in the literature is explained by the fact that the variables of the linear system to be solved here have been arranged in a truck-by-truck fashion instead of a stage-by-stage fashion [18]. While the former structure enables a distributed truck-by-truck solution approach to be designed, it replaces the linear complexity of the Riccati recursion in the number of stages N with a cubic complexity when implemented in a naive way. This problem may perhaps be avoided in an implementation where the sparse stage-by-stage structure of the initial QP formulation (14) is exploited, but we believe this to be a research topic in its own right and this is not further discussed here.

B. The Riccati Recursion

The Riccati recursion for an unconstrained QP OCP is conceptually very similar to the *dynamic programming* (DP) solution detailed in standard textbooks [17], [29]. Just like the DP solution, the Riccati recursion consists of a backward sweep where recursive expressions such as the *discrete-time Riccati equation* are applied, and a forward sweep where the dynamics are propagated forward. The complete derivation of the Riccati recursion equations for a KKT matrix with the form (20a) can be found in [18]. Here, this procedure is used to solve the KKT system (19) and compute the PDIP search direction in a distributed fashion. Algorithm 1 presents the computations and information exchanges required to carry out the Riccati recursion in the platoon.

Algorithm 1: Riccati Recursion. The superscript j has been dropped for the sake of simplicity. The truck index at which computations take place is given in each line.

1 **procedure** RICCATIRECURSION($Z_1, ..., Z_n, \tau$)
2 n: $P_n \leftarrow \bar{Q}_n, \quad \psi_n \leftarrow -\bar{r}_{x,n}$
`// Backward recursion`
3 **for** $i = n, ..., 2$ **do**
4 i: $P_{i-1} \leftarrow \tilde{H}_{x,i}^\top \Sigma_i \tilde{H}_{x,i} + A_i^\top P_i A_i - (A_i^\top P_i B_i + \bar{M}_i)(\bar{R}_i + B_i^\top P_i B_i)^{-1}(B_i^\top P_i A_i + \bar{M}_i^\top)$
5 i: $\psi_{i-1} \leftarrow -\tilde{H}_{x,i}^\top(\tau\Lambda_i + \Sigma_i\tilde{D}_i + \mu_i) - A_i^\top P_i \bar{r}_{\lambda,i} + A_i^\top\psi_i - (A_i^\top P_i B_i + \bar{M}_i)(\bar{R}_i + B_i^\top P_i B_i)^{-1}(B_i^\top\psi_i - \bar{r}_{u,i} - B_i^\top P_i \bar{r}_{\lambda,i})$
6 i: send $\tau, P_{i-1}, \psi_{i-1}$ to $i-1$
7 $i-1$: $P_{i-1} \leftarrow P_{i-1} + \tilde{Q}_{i-1}$
8 $i-1$: $\psi_{i-1} \leftarrow \psi_{i-1} - \tilde{L}_{x,i-1}$
9 **end**
10 1: solve $\begin{bmatrix} \bar{R}_1 & B_1^\top & \\ B_1 & & -I \\ & -I & P_1 \end{bmatrix} \begin{bmatrix} U_1^+ \\ \lambda_1^+ \\ X_1^+ \end{bmatrix} = - \begin{bmatrix} \bar{r}_{u,1} \\ \bar{r}_{\lambda,1} \\ -\psi_1 \end{bmatrix}$
11 1: compute μ_1^+, σ_1^+ from (22a),(22c)
12 1: $\alpha \leftarrow \min_t(t\mu_1^+ + (1-t)\mu_1 > 0, t\sigma_1^+ + (1-t)\sigma_1 > 0)$
13 1: $Z_1^+ \leftarrow \begin{bmatrix} U_1^+ & X_1^+ & \lambda_1^+ & \mu_1^+ & \sigma_1^+ \end{bmatrix}^\top$
`// Forward recursion`
14 **for** $i = 2, ..., n$ **do**
15 $i-1$: send α, X_{i-1}^+ to i
16 i: $U_i^+ \leftarrow (\bar{R}_i + B_i^\top P_i B_i)^{-1}(B_i^\top\psi_i - \bar{r}_{u,i} - B_i^\top P_i \bar{r}_{\lambda,i} - (B_i^\top P_i A_i + \bar{M}_i^\top)X_{i-1}^+)$
17 i: $X_i^+ \leftarrow A_i X_{i-1}^+ + B_i U_i^+ + C_i$
18 i: $\lambda_i^+ \leftarrow P_i X_i^+ - \psi_i$
19 i: compute μ_i^+, σ_i^+ from (22b),(22c)
20 i: $\alpha_i \leftarrow \min_t(t\mu_i^+ + (1-t)\mu_i > 0, t\sigma_i^+ + (1-t)\sigma_i > 0)$
21 i: $\alpha \leftarrow \min(\alpha_i, \alpha)$
22 i: $Z_i^+ \leftarrow \begin{bmatrix} U_i^+ & X_i^+ & \lambda_i^+ & \mu_i^+ & \sigma_i^+ \end{bmatrix}^\top$
23 **end**

In this algorithm, a block factorization scheme is first applied to the KKT matrix (20a) in Lines 2-9, followed by the computation of the search direction in Lines 10-23. Both steps involve recursive equations between consecutive platoon members, where Line 4 is the discrete-time Riccati equation [29]. These equations are used to compute the vectors ψ_i and matrices P_i in the backward recursion step, and the primal-dual search directions Z_i^+ and maximum step size α in the forward recursion step. Note that backward recursion is to be understood in the sense of truck indices here, and not in terms of the direction in which the platoon is moving.

Each truck is only responsible for computations involving its own local matrices when applying the Riccati recursion. Therefore, ψ_{i-1} and P_{i-1} need to be partially computed by

Algorithm 2: PDIP algorithm for solving QP (16).

1 **procedure** PDIPSOLVER($Y_1^{\text{guess}}, ..., Y_n^{\text{guess}}$)
2 $j \leftarrow 1, \quad \tau^{[1]} \leftarrow \tau_{\text{ini}}, \quad \beta \leftarrow$ true
3 $\forall i$: $Z_i^{[1]} \leftarrow \begin{bmatrix} Y_i^{\text{guess}} & 0 \end{bmatrix}^\top$
// Main PDIP loop
4 **while** $\beta =$ true **or** $\tau^{[j]} > \text{tol}_\text{P}$ **do**
5 $Z_1^{+[j]}, ..., Z_n^{+[j]}, \alpha_{\max}^{[j]} \leftarrow$ RICCATIRECURSION($Z_1^{[j]}, ..., Z_n^{[j]}, \tau^{[j]}$)
// Backward recursion
6 **for** $i = n, ..., 1$ **do**
7 i: $Z_i^{[j+1]} \leftarrow \alpha_{\max}^{[j]} Z_i^{+[j]} + (1 - \alpha_{\max}^{[j]}) Z_i^{[j]}$
8 **if** $i \geq 2$ **then**
9 i: send $\alpha_{\max}^{[j]}$ to $i-1$
10 **end**
11 **end**
12 1: $\beta \leftarrow$ false, $\quad \phi_0^{[j+1]} \leftarrow 0$
// Forward recursion
13 **for** $i = 1, ..., n$ **do**
14 **if** $\beta =$ false **then**
15 i: $r_i^{[j+1]} \leftarrow$ KKTCONDITIONS($Z_i^{[j+1]}, \phi_{i-1}^{[j+1]}, \tau^{[j]}, i$)
16 **if** $\|r_i^{[j+1]}\|_\infty > \text{tol}_\text{P}$ **then**
17 i: $\beta \leftarrow$ true
18 **else if** $i \leq n-1$ **then**
19 i: $\phi_i^{[j+1]} \leftarrow \tilde{Q}_i X_i^{[j+1]} + \tilde{L}_{x,i} - \lambda_i^{[j+1]}$
20 i: send $\phi_i^{[j+1]}$ to $i+1$
21 **end**
22 **end**
23 **if** $i \leq n-1$ **then**
24 i: send β to $i+1$,
25 **end**
26 **end**
27 **if** $\beta =$ false **then**
28 n: $\tau^{[j+1]} \leftarrow \gamma \tau^{[j]}$
29 **else**
30 n: $\tau^{[j+1]} \leftarrow \tau^{[j]}$
31 **end**
32 $j \leftarrow j + 1$
33 **end**
34 $\forall i$: $U_i^* \leftarrow U_i^{[j+1]}, \quad X_i^* \leftarrow X_i^{[j+1]}$
35 $\forall i$: $\lambda_i^{\text{mod}} \leftarrow \lambda_i^{[j+1]}, \quad \mu_i^{\text{mod}} \leftarrow \mu_i^{[j+1]}$

truck i first and are then sent to truck $i-1$ in Line 6. Likewise, the search direction in the variables of any follower is computed by passing forward the search direction in the state variables of the previous truck in Line 15. A parameter α is also transmitted, which indicates the maximum step size value such that the PDIP constraints $\mu_i > 0$ and $\sigma_i > 0$ hold at the next iteration for all preceding trucks.

V. SUMMARY OF THE DISTRIBUTED SOLUTION PROCEDURE

This section offers an overview of the complete procedure proposed to solve the platoon eco-driving NLP (13). The algorithms assembled in the two previous sections are detailed further, and the exact information that needs to be sent along the platoon for solving (13) in a distributed fashion is given.

Algorithm 3: Subroutine to compute the part of the perturbed KKT conditions of QP (16) corresponding to truck i. The superscript j has been dropped for the sake of simplicity.

1 **procedure** KKTCONDITIONS(Z_i, ϕ_{i-1}, τ, i)
2 **if** $i = 1$ **then**
3 $r_i \leftarrow \begin{bmatrix} \tilde{R}_1^\top U_1 + \tilde{L}_{u,1} + B_1^\top \lambda_1 + \tilde{H}_{u,1}^\top \mu_1 \\ B_1 U_1 + C_1 - X_1 \\ \tilde{H}_{u,1} U_1 + \tilde{D}_1 + \sigma_1 \\ \text{diag}(\mu_1)\sigma_1 - \tau e \end{bmatrix}$
4 **else**
5 $r_i \leftarrow \begin{bmatrix} \tilde{M}_i U_i + A_i^\top \lambda_i + \tilde{H}_{x,i}^\top \mu_i + \phi_{i-1} \\ \tilde{M}_i^\top X_{i-1} + \tilde{R}_i^\top U_i + \tilde{L}_{u,i} + B_i^\top \lambda_i + \tilde{H}_{u,i}^\top \mu_i \\ A_i X_{i-1} + B_i U_i + C_i - X_i \\ \tilde{H}_{x,i} X_{i-1} + \tilde{H}_{u,i} U_i + \tilde{D}_i + \sigma_i \\ \text{diag}(\mu_i)\sigma_i - \tau e \end{bmatrix}$
6 **end**
7 **if** $i = n$ **then**
8 $r_i \leftarrow \begin{bmatrix} r_i \\ \tilde{Q}_n X_n + \tilde{L}_{x,n} - \lambda_n \end{bmatrix}$
9 **end**

A. PDIP Algorithm

Algorithm 2 presents the complete PDIP algorithm which is deployed to solve QP (16). After running the Riccati recursion, an additional step of backward and then forward recursion is needed for each PDIP iteration. The former, in Lines 6-11, is used to transmit the maximum feasible step size $\alpha_{\max}^{[j]}$, which is originally only available to the last truck of the platoon, to all trucks. Each truck can then take a Newton-type step on its local variables.

The forward recursion detailed in Lines 13-26 is used to check the termination criterion for a given barrier parameter $\tau^{[j]}$. Namely, the norm of the perturbed KKT conditions of QP (16) must be below a certain tolerance threshold. This condition can be checked in a distributed fashion since the perturbed KKT conditions vector is almost separable. For each follower, only a term $\phi_i^{[j+1]}$ needs to be computed and passed forward by the previous truck in Line 20. The local part $r_i^{[j+1]}$ of the KKT conditions vector can then be computed, as detailed in Algorithm 3.

If the termination criterion is not met for one of the trucks, the Boolean variable β can be transmitted with a positive value to indicate that another PDIP iteration must be carried out, and that there is no need to further compute the KKT conditions vector. Note that it can be assumed that each follower i holds a local copy of $X_{i-1}^{[j+1]}$ when running Algorithm 3. Indeed, the step size $\alpha_{\max}^{[j]}$ is the same for all trucks, and the search direction in the state variables of $i-1$ has previously been

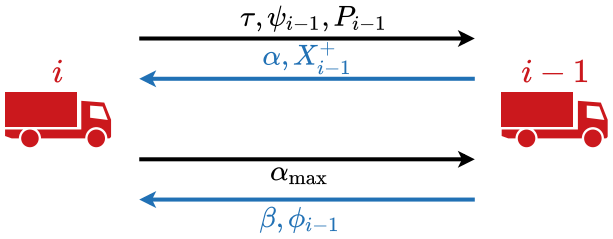


Fig. 2: Information exchange required between successive trucks during one PDIP iteration. The black and blue arrows denote backward and forward recursions, respectively.

transmitted to i during the backward recursion step of the Riccati recursion.

Finally, a simple update procedure is used for the barrier parameter in Line 28 of Algorithm 2, where $\gamma \in]0, 1[$. Since the update is carried out by the last truck of the platoon, the new value $\tau^{[j+1]}$ has to be transmitted backward at the beginning of the next PDIP iteration, which can be done during the Riccati recursion.

The total information exchange between successive trucks during each PDIP iteration is summarized in Fig. 2. It can be seen that the trucks do not need to transmit directly the matrices associated with their local problem in this algorithm. Note that even though all computations involved in the PDIP algorithm are distributed, the iterates are exactly the same as in a centralized version of the algorithm since no approximation has been introduced in the distributed solution approach.

B. SQP Algorithm

Once the modified subproblem (16) has been solved, a search direction for the SQP algorithm can be obtained. The complete distributed SQP procedure is detailed in Algorithm 4. Note that the Δ notation is briefly readopted there in order to emphasize the role of the solution of QP (16) as a search direction.

Each vehicle i starts by recovering the primal-dual solution of the original subproblem (15) through (18). Then, a Newton-type step on the local primal-dual variables Y_i is taken with the search direction ΔY_i. As was mentioned previously, it is assumed here that only full steps are taken in the SQP algorithm. The vector of the KKT conditions of NLP (13) is then computed at the new SQP iterate to check for optimality. Finally, each truck can start forming their local matrices for the next version of subproblem (16).

Note that all the computations described in the previous paragraph can be carried out in parallel aboard each vehicle. Contrary to the backward and forward recursion steps of the PDIP algorithm, where each computation is dependent on some information that has to be transmitted by a neighboring truck, the vehicles only need to exchange a Boolean variable β in Algorithm 4. The communication overhead needed for the SQP method is therefore almost zero.

As before, β is used to decide whether another SQP iteration is needed, and is transmitted through backward recursion in order to avoid unnecessary computations in the event that the local KKT conditions of one of the vehicles do not hold. Once convergence to the desired tolerance has been reached for the SQP method, a final forward recursion may be needed to notify the rest of the platoon that the problem has been solved.

Algorithm 4: SQP algorithm for solving NLP (13).

$Y_1, ..., Y_n \leftarrow Y_1^{\text{ini}}, ..., Y_n^{\text{ini}}, \quad \beta \leftarrow$ true
$\forall i$: form local matrices of QP (16) at Y_i
`// Main SQP loop`
while $\beta =$ true **do**
  $\{\Delta U_i^*, \Delta X_i^*, \lambda_i^{\text{mod}}, \mu_i^{\text{mod}}\}_{i=1,...,n} \leftarrow$ PDIPSOLVER$(Y_1, ..., Y_n)$
  $\forall i$: compute λ_i^*, μ_i^* from (18)
  $\forall i$: $\Delta Y_i \leftarrow [\Delta U_i^*, \Delta X_i^*, \lambda_i^*, \mu_i^*]^\top$
  $\forall i$: $Y_i \leftarrow$ TAKESTEP(ΔY_i)
  $\forall i$: form local matrices of QP (16) at Y_i
  n: $\beta \leftarrow$ false
  `// Backward recursion`
  for $i = n, ..., 1$ **do**
    if $\beta =$ false **then**
      i: $r_i \leftarrow$ KKTCONDITIONSSQP(Y_i)
      if $\|r_i\|_\infty >$ tol$_{\text{S}}$ **then**
        i: $\beta \leftarrow$ true
      end
    end
    if $i \geq 2$ **then**
      i: send β to $i-1$
    end
  end
end

In order to validate that the proposed distributed solution method can satisfactorily solve the eco-driving NPL (13), the problem is also solved centrally using the open source symbolic framework CasADi [30] and the solver IPOPT, known for its robust convergence features [31]. In each case tested, the two solution approaches converge to the same minimum when starting from the same initial point, and the iterates converge to that same minimum for all initial points tested.

VI. SIMULATIONS

This section presents the results obtained for a platoon of electric trucks. One of the main advantages of solving the cooperative eco-driving problem in a distributed fashion is that this can easily be deployed on the go for trucks belonging to different transportation companies. The main purpose of this case study is therefore to give some indications on the practical implications of deploying the proposed distributed algorithms. Since the trucks may have very different properties in case they belong to different companies, we focus mainly on heterogeneous platoons in what follows.

A. Setup

The following control methods are deployed on the platoon:

- *Cooperative eco-driving:* The trucks cooperate to solve NLP (13) by using the distributed solution method proposed in this paper.

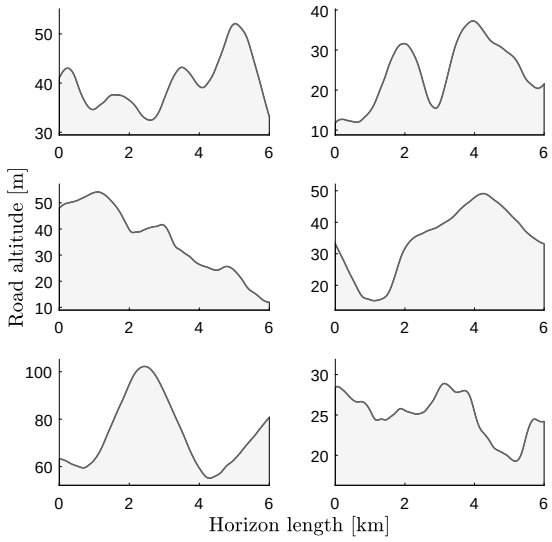


Fig. 3: Driving cycles used in the simulations.

- *Non-cooperative eco-driving:* Each truck greedily solves its own eco-driving problem with the aim of minimizing its own energy consumption. It is assumed that the individual truck problems are solved sequentially, starting with the leader of the platoon. The optimal state trajectory obtained by each truck is then transmitted to its direct follower, which uses this information when solving its own problem.
- *Reference speed tracking:* The platoon leader tracks the reference speed $v_{\text{ref},1}$ over the entire horizon. Each follower then tracks a headway of t_h with the preceding truck in order to form a tight platoon and benefit from the air drag reduction.

Note that the non-cooperative controller presented here should be seen as a 'best-case scenario' of the performances non-cooperating trucks could achieve in practice. In more realistic settings, trucks would probably not have access to the perfect state information of the truck preceding them, and would most likely keep larger headways than they do here for safety reasons, thus benefiting less from drag reduction.

The tracking controller is meant to represent a scenario where all the trucks do not have the computing resources necessary to solve eco-driving optimization problems onboard, but rather rely on tracking simpler references. This controller is based on the observation that a tight platoon formation is energy-optimal most of the time.

In total, 6 driving cycles of 6 km each are used, and displayed in Fig. 3. They are extracted from the Södertälje-Norrköping highway in Sweden, which has many uphill and downhill segments [24]. These driving cycles have been chosen to cover the principal features of the road. Each of them should be considered as the horizon of the eco-driving problem at one sampling instant in an online MPC implementation. In order to avoid unnecessary conversion of kinetic energy to battery energy, the vehicles are made to have a speed of $\bar{v}$ when reaching the end of the horizon.

TABLE I: SIMULATION PARAMETERS

Horizon length	$s_f = 6$ km
Cruising speed	$\bar{v} = 80$ km/h
Allowed speed deviation	$\Delta v = 10$ km/h
Rolling resistance coefficient	$c_r = 0.006$
Air density	$\rho_a = 1.184$ kg/m^3
Frontal truck area	$A_f = 10$ m^2
Aerodynamic drag coefficient	$c_d = 0.6$
Drag reduction coefficients	$c_1 = 12.8$ m^{-1}, $c_2 = 19.7$ m
Truck length	$L = 18$ m
Minimum time headway	$t_h = 1.35$ s
Number of samples	$N = 75$
PDIP tolerance	$\text{tol}_\text{P} = 10^{-4}$
SQP tolerance	$\text{tol}_\text{S} = 10^{-3}$

The simulation parameters are given in Table I, and the vehicle models used in the simulations are the same as the ones presented earlier in the modeling section. For heterogeneous platoons, the truck masses are assumed to be uniformly spaced within 27-44 t and the rated electric motor powers within 200-330 kW. Electric motors are modeled by linearly scaling an original model of a 300 kW motor. In all the simulations, it is assumed that the trucks are initially at 3 times the minimum allowed time headway from each other. In addition, the speed of each truck is only allowed to vary in a window of size $2\Delta v$ centered around the reference trajectory of this truck.

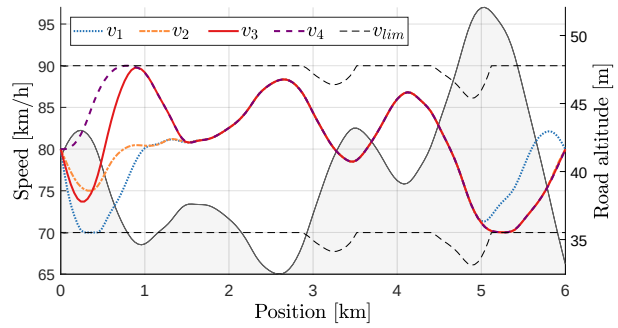
B. Optimal Trajectories

We start by considering a heterogeneous platoon in which 4 trucks are organized in a decreasing motor power order, with the motor and mass characteristics specified above. Fig. 4 presents the optimal solution of the cooperative eco-driving NLP (13) when solved with the proposed method. The speed limits v_{lim} displayed in Fig. 4a are those that apply to the last truck of the platoon. Being the least powerful vehicle of the group, it is the one with the most restrictive speed limits in steep uphill sections. Similarly, the longitudinal force limits of the last truck are computed from its optimal speed trajectory and displayed in Fig. 4c as F_{lim}.

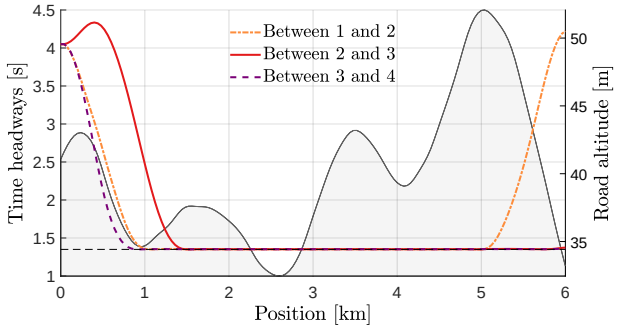
It can be observed in this figure that the longitudinal force demanded never comes close to the lower bound. Consequently, the last truck of the platoon never needs to use its friction brakes since regenerative braking can always be used, meaning that the braking force F_b remains zero. This observation holds in fact for all the trucks and across all the simulation scenarios studied in this paper. Under our modeling assumptions, electric trucks can avoid dissipating energy through braking altogether, even during steep downhills. The direct conclusion is that it is always energy-optimal for a platoon of electric trucks to adopt the tightest possible formation, regardless of the road gradient.

Interestingly, this result stands in sharp contrast to what is commonly observed for platoons of conventional or hybrid electric vehicles. In such cases, the use of friction brakes cannot usually be avoided and the vehicles have an incentive to momentarily increase their time headways in downhill sections in order to avoid energy losses due to braking [12], [13].

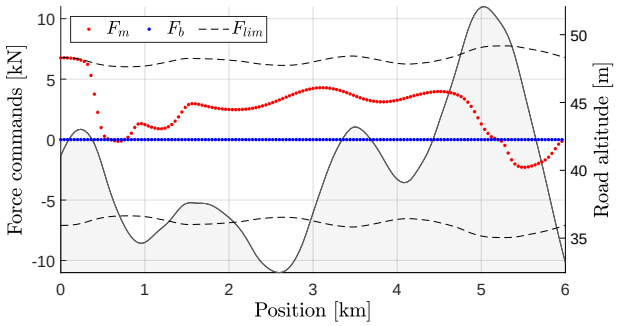
The optimal speed trajectories obtained by the cooperative and non-cooperative eco-driving methods are shown in Fig. 4a and Fig. 5, respectively. They are very similar to each



(a) Speed profiles for all trucks.



(b) Time headways of successive trucks.



(c) Control input trajectories for the last truck of the platoon.

Fig. 4: Optimal trajectories of the cooperative eco-driving control method for a heterogeneous platoon of 4 trucks.

other since the trucks adapt their driving behavior to the road gradient. On the other hand, the simpler reference tracking controller has the leader remain at the reference speed over the entire horizon, as shown in Fig. 6. The other trucks go through a strong initial acceleration phase in order to reach headways of t_h with the truck in front of them, at which point they drive at the reference speed until the end of the horizon.

In the case of the cooperative eco-driving control method, it can be seen in Fig. 4b that the trucks travel in a tight platoon formation when their speeds coincide, starting shortly after the first kilometer. For the other two control methods, however, a tight platoon is formed only after the second kilometer. The non-cooperative controller is the latest to achieve a tight platoon formation due to the high initial speed of the platoon leader. On the other hand, the platoon leader purposefully slows down initially with the cooperative controller in order to wait for the trailing vehicles to catch up. This then results in

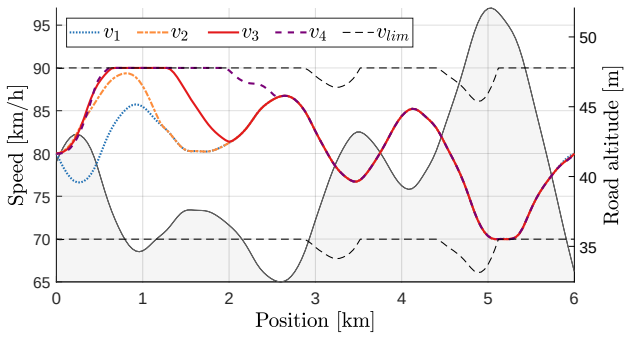


Fig. 5: Optimal speed profiles of the non-cooperative eco-driving control method for a heterogeneous platoon of 4 trucks.

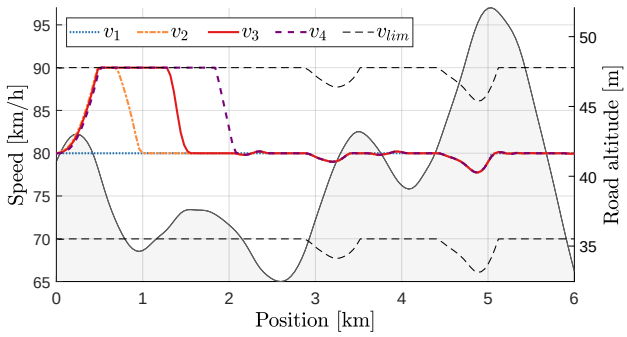


Fig. 6: Optimal speed profiles of the reference speed tracking control method for a heterogeneous platoon of 4 trucks.

the leader needing to accelerate shortly before the end of the horizon and leave the platoon in order to satisfy its terminal travel time constraint, as can be seen in the last kilometer in Fig. 4a.

C. Energy Savings

Next, we present quantitative estimates of the energy savings that each control method can achieve. Note that all the results in this section are averaged over the driving cycles presented in Fig. 3.

In Fig. 7, energy saving performances for the platoon control methods are given for different platoon sizes. The values given are percentages which represent the energy savings over a reference scenario without platooning, i.e. where the vehicles travel individually without benefiting from any drag reduction. It is assumed that each vehicle follows an eco-driving trajectory in this scenario.

Regardless of the control method chosen, traveling as a platoon seems to always be preferable in terms of energy consumption. The same can be said about using slope information to compute eco-driving trajectories, as the control methods which do so perform better than the simpler tracking baseline. The best performances are achieved by the cooperative eco-driving method though, which displays greater overall energy savings than the non-cooperative one.

In addition, it can be observed in Fig. 7 that the energy savings of all control methods increase significantly with the platoon size, for platoons of up to 4 vehicles. For larger platoons, the energy savings over individual trucks remain almost

TABLE II: ENERGY SAVINGS OF THE PLATOON CONTROL METHODS OVER A SCENARIO WITHOUT PLATOONING FOR HETEROGENEOUS 4-TRUCK PLATOONS WITH DIFFERENT COMPOSITIONS. THE AVERAGE ENERGY CONSUMPTION PER DRIVING CYCLE OF THE VEHICLES IN THE REFERENCE SCENARIO IS DISPLAYED IN THE RIGHTMOST COLUMN OF THE TABLE. THE LEADER OF THE PLATOON IS THE RIGHTMOST TRUCK.

				Tracking	**Non-cooperative**	**Cooperative**	**No platoon**
220 kW 27 t	257 kW 33 t	293 kW 38 t	330 kW 44 t	8.5%	9.5%	11.0%	26.8 kWh
330 kW 44 t	293 kW 38 t	257 kW 33 t	220 kW 27 t	8.5%	9.7%	11.2%	26.7 kWh
220 kW 40 t	257 kW -	293 kW -	330 kW -	4.8%	7.2%	9.1%	28.2 kWh
330 kW 40 t	293 kW -	257 kW -	220 kW -	6.9%	8.5%	9.7%	27.6 kWh
300 kW 27 t	- 33 t	- 38 t	- 44 t	8.5%	9.6%	10.9%	27.0 kWh
300 kW 44 t	- 38 t	- 33 t	- 27 t	7.2%	8.3%	10.7%	27.2 kWh

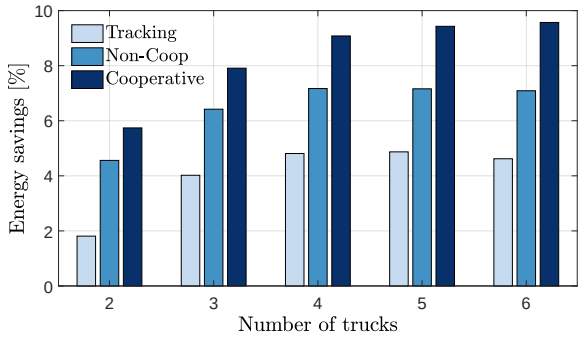


Fig. 7: Energy savings of the platoon control methods over a scenario without platooning for different platoon sizes. All trucks are assumed to have a mass of 40 t and rated electric motor powers spaced uniformly within 330-200 kW.

constant. Therefore, it would be unpractical to organize a large group of electric trucks into a single platoon, when similar energy savings could be achieved by forming several smaller platoons for which only problems with a lower complexity would have to be solved.

We now study different compositions for a heterogeneous platoon of 4 vehicles. The energy saving performances of the different control methods for each platoon composition are given in Table II. In this table, the platoons differ in the mass and rated motor power of the trucks, and in how the trucks are distributed within the platoon.

One can observe that the same performance pattern as in Fig. 7 emerges, and that the cooperative eco-driving control method proposed in this paper systematically outperforms the other methods implemented.

The energy savings over individual trucks are the least significant overall when the trucks are on average heavier, as displayed in rows 3 and 4 of the table, where each truck weighs 40 tons and no controller can achieve savings above 10%. This can be explained by the observation that a tight platoon is formed later in that case, due to the larger inertia of the vehicles. In fact, as discussed previously, the energy efficiency of a platoon of electric trucks seems to be strongly correlated with its ability to adopt and maintain a tight formation, thanks to the regenerative braking ability of the trucks.

This last point is further motivated by the smaller relative variations in energy savings of the cooperative control method compared with the other control methods. Indeed, when the trucks cooperate, the leading trucks tend to wait initially for the other trucks to catch up. The time at which a tight platoon is formed is consequently not greatly affected by the characteristics of the trucks. However, the same cannot be said of the other control methods, where the trucks act individually. For these methods, the energy savings get noticeably worse in settings where the trucks at the back of the platoon struggle to catch up with the rest due to e.g. lower motor power over mass ratios, such as in row 3 of Table II.

To conclude, the cooperative eco-driving method proposed in this paper achieves significant energy savings compared with a situation where the trucks would refuse to merge and travel together as a platoon. For electric trucks, the savings in question have been found to be quite consistent across various platoon compositions, meaning that independent trucks would greatly benefit from spontaneously forming a platoon on the go regardless of their individual characteristics.

D. Comments on Computation Times

We want to draw the reader's attention to the fact that the implementation of the distributed cooperative eco-driving procedure used in this section should only be seen as a proof of concept. This is why the focus of this case study has been on the practical implications of having a fully distributed algorithm, such as the potential energy savings, and not on benchmarking a mature implementation of this algorithm against alternatives.

That being said, we give here a few orders of magnitude for the runtime of different parts of the implementation used in the case study to complete the picture. Note that these runtime values represent pessimistic upper bounds since they are obtained from a prototype version of the algorithm. A platoon of 4 vehicles with $N = 75$ samples is considered, in which case the algorithm solves an NLP with 1508 variables. The communication overheads are not included.

In our prototype implementation, the most computationally demanding step is the backward sweep of the Riccati recursion, which can take up to 100 milliseconds to complete in the worst case. The forward sweep is faster as it only requires

up to 30 milliseconds. We refer the reader to Algorithm 1 for a detailed description of these steps. The computation of the perturbed KKT conditions and of the next PDIP step in Algorithm 2 and Algorithm 3 takes up to 50 milliseconds in total. Finally, computing the KKT conditions and the next SQP step takes up to 30 milliseconds in Algorithm 4. Note that these values scale linearly with the number of vehicles due to the distributed nature of the procedure, where all computations are arranged vehicle-by-vehicle.

In comparison, a few hundred milliseconds are required to solve each individual vehicle problem with both the non-cooperative and tracking control methods. Such methods would likely be implemented very differently in practice, however. For instance, the tracking method is simulated here by solving an NLP for the sake of consistent comparison with the other methods, whereas adaptive cruise controllers would normally be used for fixed headway tracking. Any direct runtime comparison between these methods should therefore be drawn cautiously as they do not reflect accurately what a real-life implementation would look like.

In fact, our prototype implementation of the proposed distributed solution procedure uses exclusively Matlab's internal linear algebra routines. While this is enough for a proof of concept, significant improvements in computation times could be obtained by using a dedicated solver instead. For instance, the HPIPM software [19] contains hardware-tailored linear algebra routines specifically designed for the real-time execution of MPC algorithms.

VII. Discussion

The focus of this paper has been on proposing distributed algorithms to solve the cooperative eco-driving control problem when expressed as NLP (13). As explained before, this problem would probably need to be embedded in an MPC implementation and solved repeatedly in order to be applied to real-time settings. We think that this would only require little extra work, however, and that frequent updates could in fact be achieved for this MPC through the use of a real-time iteration scheme [32]. A low-level tracking layer could then be implemented independently for each vehicle in order to follow the resulting optimal trajectories.

When solving NLP (13), improvements in performance could likely be gained by taking better steps in the SQP and PDIP algorithms by using a backtracking line-search on a merit function [20], and even more importantly by carefully warm-starting the SQP method. The latter could easily be implemented in a receding horizon context, where the shifted previous solution could be used as an initial guess. In addition, a lot of the computations could likely be parallelized thanks to the distributed aspect of the solution procedure. The algebraic operations in Algorithm 1 and Algorithm 3 could for example be precomputed in parallel by the vehicles, and potentially reused across PDIP iterations. We believe that this would result in significant runtime improvements.

The proposed distributed solution method for NLP (13) could then likely be carried out in real-time with the performance gains mentioned above. Indeed, with properly optimized algorithms, large-scale NLPs can be solved extremely fast with the SQP method, as was observed both in simulations and on real systems [33]. In the case of automotive applications, experimental results suggest that vehicle-to-vehicle communication delays may be more of a bottleneck than actually solving NLPs [34].

In the case of a practical implementation of our method, trucks belonging to different transportation companies might be dissuaded from spontaneously forming platoons since the truck acting as platoon leader does not usually benefit from it. A way to incentivize platoon formation on the go is to make the role of platoon leader rewarding, which could be achieved through a simple compensation mechanism. Once NLP (13) has been solved, this mechanism could estimate how much energy the leader would have saved had it been trailing behind another platoon member instead. This difference could be materialized as an energy 'price' P_{leader} for being the leader of the platoon instead of a follower. This price could serve as the basis of this compensation mechanism whereby the other platoon members would 'owe' P_{leader} to the leader. Appendix B details how such a mechanism could be implemented.

VIII. Conclusion

In this paper, the cooperative eco-driving of a platoon of electric vehicles has been formulated as an optimal control problem, and a solution procedure has been proposed to solve it. This procedure exploits the chain-like structure of the platoon, and can be carried out in a distributed fashion as a result. This is achieved by solving the problem with the SQP method, and by solving each intermediate SQP subproblem with a PDIP algorithm. In this algorithm, the particular structure of the subproblems results in banded KKT matrices, which can be factorized in a truck-by-truck fashion with the Riccati recursion. It follows from this factorization method that vehicles only need to exchange information with their direct neighbors, so that high-bandwidth communications could potentially be used. In addition, the complete procedure is privacy-preserving since the vehicles do not need to share directly their local problem information in these algorithms.

The performances of the proposed cooperative eco-driving control have been evaluated in numerical experiments, where the benefits of forming and keeping tight platoons for electric trucks in a hilly terrain, due to the efficiency of regenerative braking, have been shown. Several heterogeneous platoon compositions have been studied in order to model the spontaneous formation of a platoon by trucks belonging to different transportation companies.

The next logical step in this line of research would now be to focus on a computationally effective implementation of the prototype algorithms presented in this study. Their online implementation could be carried out with MPC, but the issue of delay between successive updates would have to be investigated further. This real-time implementation could later be studied in a real truck system in order to assess the potential of the proposed control method for a future deployment in commercial systems. A complementary future research direction is to investigate how this method could fit into a higher-level framework aiming to decide when and

where it is optimal for truck platoons to form or split on the go.

APPENDIX

A. Proof of Proposition 1

First, let us show that both problems have the same primal solution. Recall that the modified QP (16) has been obtained from the original QP (15) by injecting the linear dynamics in the objective function and the inequality constraints. Consequently, the objective function and inequality constraints of both QPs are the same at any point which satisfies the dynamics equations, i.e. the equality constraints (15b) and (15d) (which are the same for both QPs). Since every point in the feasible set of either QP must satisfy these equality constraints, it follows that (15) and (16) have the same feasible set and that their objective functions have the same value everywhere on this set. Therefore, both QPs have the same primal solution, which is noted $W^* = [U_1^*, X_1^*, ..., U_n^*, X_n^*]^\top$ hereafter. Note that W^* is the global solution of QP (15). This follows from the strict convexity of (15), thanks to the Hessian regularization step carried out in the SQP algorithm [20].

Then, let us show that the dual variables of both problems are linked through (18) for every truck $i \in \mathbb{I}_{[1,n]}$. To do so, let the Lagrange function of QP (15) be defined as:

$$\mathcal{L}(W^*, \lambda^*, \mu^*) = J(W^*) + \lambda^{*\top} c_{\mathrm{g}}(W^*) + \mu^{*\top} c_{\mathrm{h}}(W^*), \quad (24)$$

with $\lambda^* = [\lambda_1^*, ..., \lambda_n^*]^\top$ and $\mu^* = [\mu_1^*, ..., \mu_n^*]^\top$. In this expression, J denotes objective function (15a), while c_{g} gathers equality constraints (15b) and (15d), and c_{h} gathers inequality constraints (15c) and (15e).

Similarly, the Lagrange function of QP (16) can be defined as:

$$\tilde{\mathcal{L}}(W^*, \lambda^{\mathrm{mod}}, \mu^{\mathrm{mod}}) = \tilde{J}(W^*) + \lambda^{\mathrm{mod}\top} \tilde{c}_{\mathrm{g}}(W^*) + \mu^{\mathrm{mod}\top} \tilde{c}_{\mathrm{h}}(W^*), \quad (25)$$

where $\lambda^{\mathrm{mod}} = [\lambda_1^{\mathrm{mod}}, ..., \lambda_n^{\mathrm{mod}}]^\top$ and $\mu^{\mathrm{mod}} = [\mu_1^{\mathrm{mod}}, ..., \mu_n^{\mathrm{mod}}]^\top$, and where $\tilde{J}$ denotes objective function (16a). Functions $\tilde{c}_{\mathrm{g}}$ and $\tilde{c}_{\mathrm{h}}$ gather equality constraints (16b) and (16d), and inequality constraints (16c) and (16c), respectively.

Next, the stationarity part of the KKT conditions of both QPs [20] can be combined into:

$$\nabla_W \mathcal{L}(W^*, \lambda^*, \mu^*) - \nabla_W \tilde{\mathcal{L}}(W^*, \lambda^{\mathrm{mod}}, \mu^{\mathrm{mod}}) = 0. \quad (26)$$

By taking the rows corresponding to the set of variables $\{U_i, X_i\}$, $i \in \mathbb{I}_{[1,n-1]}$, in this equation, we obtain the following equation system:

$$\begin{aligned} &M_i^\top X_i^* + R_i U_i^* - \tilde{M}_i^\top X_{i-1}^* - \tilde{R}_i U_i^* + L_{u,i} - \tilde{L}_{u,i} \\ &\quad + B_i^\top (\lambda_i^* - \lambda_i^{\mathrm{mod}}) + H_{u,i}^\top \mu_i^* - \tilde{H}_{u,i}^\top \mu_i^{\mathrm{mod}} = 0, \quad (27a) \\ &Q_i X_i^* + M_i U_i^* - \tilde{Q}_i X_i^* - \tilde{M}_{i+1} U_{i+1}^* + L_{x,i} - \tilde{L}_{x,i} \\ &\quad - (\lambda_i^* - \lambda_i^{\mathrm{mod}}) + H_{x,i}^\top \mu_i^* + A_{i+1}^\top (\lambda_{i+1}^* - \lambda_{i+1}^{\mathrm{mod}}) \\ &\quad + \hat{H}_{x,i+1}^\top \mu_{i+1}^* - \tilde{H}_{x,i+1}^\top \mu_{i+1}^{\mathrm{mod}} = 0. \quad (27b) \end{aligned}$$

Using (17), this system is simplified to:

$$\begin{aligned} &B_i^\top (\lambda_i^* - \lambda_i^{\mathrm{mod}}) - B_i^\top M_i U_i^* + H_{u,i}^\top \mu_i^* - \tilde{H}_{u,i}^\top \mu_i^{\mathrm{mod}} = 0, \quad (28a) \\ &M_i U_i^* - A_{i+1}^\top M_{i+1} U_{i+1}^* - (\lambda_i^* - \lambda_i^{\mathrm{mod}}) + H_{x,i}^\top \mu_i^* \\ &\quad + A_{i+1}^\top (\lambda_{i+1}^* - \lambda_{i+1}^{\mathrm{mod}}) + \hat{H}_{x,i+1}^\top \mu_{i+1}^* - \tilde{H}_{x,i+1}^\top \mu_{i+1}^{\mathrm{mod}} = 0. \quad (28b) \end{aligned}$$

Similarly, the rows of (26) which correspond to the variables $\{U_n, X_n\}$ can be written as:

$$\begin{aligned} &B_n^\top (\lambda_n^* - \lambda_n^{\mathrm{mod}}) - B_n^\top M_n U_n^* + H_{u,n}^\top \mu_n^* - \tilde{H}_{u,n}^\top \mu_n^{\mathrm{mod}} = 0, \quad (29a) \\ &M_n U_n^* - (\lambda_n^* - \lambda_n^{\mathrm{mod}}) + H_{x,n}^\top \mu_n^* = 0. \quad (29b) \end{aligned}$$

We are now ready to prove (18) by induction:

- For the base case, here performed at index n, we start by rewriting (29b) as:

  $$\lambda_n^* - \lambda_n^{\mathrm{mod}} = M_n U_n^* + H_{x,n}^\top \mu_n^*. \quad (30)$$

  By using (30) in (29a), and using (17) again, we obtain:

  $$\tilde{H}_{u,n}^\top (\mu_n^* - \mu_n^{\mathrm{mod}}) = 0. \quad (31)$$

  Under the assumption that linear independent constraint qualification (LICQ) holds at the optimal solution, the gradients of the active constraints are linearly independent [20]. In particular, for QP (16), this means that matrix $\tilde{H}_{u,i}$, $\forall i \in \mathbb{I}_{[1,n]}$, is full row-rank for the active constraints. Consequently, it results from (31) that $\mu_n^* = \mu_n^{\mathrm{mod}}$. By substituting μ_n^* with μ_n^{mod} in (30), we obtain the desired expression for λ_n^*.
- For the induction step, let us consider any index i, $i \in \mathbb{I}_{[1,n-1]}$, and assume that (18) holds at index $i+1$, i.e.:

  $$\begin{aligned} &\mu_{i+1}^* = \mu_{i+1}^{\mathrm{mod}}, \quad (32a) \\ &\lambda_{i+1}^* = \lambda_{i+1}^{\mathrm{mod}} + H_{x,i+1}^\top \mu_{i+1}^{\mathrm{mod}} + M_{i+1} U_{i+1}^*. \quad (32b) \end{aligned}$$

  By using (32) in (28b), we obtain that:

  $$\lambda_i^* - \lambda_i^{\mathrm{mod}} = M_i U_i^* + H_{x,i}^\top \mu_i^*, \quad (33)$$

  which can again be used in (28a), resulting in:

  $$\tilde{H}_{u,i}^\top (\mu_i^* - \mu_i^{\mathrm{mod}}) = 0. \quad (34)$$

  At this point, we can simply follow the same line of reasoning as above to conclude that (18) holds at index i. This concludes the proof by induction. □

B. Leader Compensation Mechanism

Let a scalar parameter p_k be added to the leader dynamics (13c) at each grid point:

$$x_{1,k+1} = F_k(x_{1,k}, u_{1,k}) + \begin{bmatrix} p_k \\ 0 \end{bmatrix}, \quad \forall k \in \mathbb{I}_{[0,N-1]}, \quad (35)$$

with $p = [p_0, 0, p_1, ..., p_{N-1}, 0]^\top$. Each parameter p_k can be thought of as a virtual subtraction to the drag force acting upon the leader at the k-th grid point. These parameters are meant to model a scenario in which drag reduction would also

apply for the leader. Note that other choices for the parametric reformulation of constraints (13c) are possible, but an affine model has been chosen here for the sake of simplicity.

Replacing the leader dynamics (13c) with their parametrized version (35) in (13) defines a parametric NLP [35]. Let $J^*(p)$ be the parametric optimal cost of this parametric NLP, and let $\mathcal{L}$ denote its Lagrange function. Note that the case $p = 0$ corresponds to the original NLP (13).

A first-order Taylor approximation of the optimal cost for a given parameter p yields:

$$J^*(p) = J^*(0) + \nabla_p J^*(0)^\top p + \mathcal{O}(\|p\|^2), \tag{36}$$

where $J^*(0)$ is the value of the cost function in (13) at optimality. By noting Y^* the primal-dual solution of (13), the sensitivity of J^* to the parameters is given as:

$$\nabla_p J^*(0) = \nabla_p \mathcal{L}(Y^*), \tag{37}$$

under the assumptions that LICQ and second-order sufficient conditions (SOSC) hold at Y^* [35]. Since p only appears linearly in the leader dynamics, we have:

$$\nabla_p \mathcal{L}(Y^*) = \lambda_1^*, \tag{38}$$

where λ_1^* is the set of optimal dual variables corresponding to the dynamics of the leader. Consequently, the price for being the platoon leader can be expressed as:

$$P_{\text{leader}} = J^*(0) - J^*(p) = -\lambda_1^{*\top} p, \tag{39}$$

and is therefore proportional to some of the Lagrange multipliers of NLP (13) as well as to the virtual drag reduction modeled through p. The latter may be estimated in different ways directly from the solution of (13), and thus requires very few extra computations. For instance, p could be based on the maximum drag reduction observed among platoon members at each grid point. In this case, P_{leader} would be slightly overestimated, which could act as an additional compensation for the leader for having a delayed reward.

Once the platoon members agree on a value for P_{leader}, the proposed compensation mechanism could split the cost equally among the followers, which would then have to compensate the leader in some way at the end of the trip.

REFERENCES

[1] International Energy Agency, "Tracking transport 2020," IEA, Paris. https://www.iea.org/reports/tracking-transport-2020.

[2] International Transport Forum, *ITF Transport Outlook 2019*. OECD Publishing, 2019.

[3] C. Bonnet and H. Fritz, "Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing," SAE technical paper, Tech. Rep., 2000.

[4] A. Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *Proc. IEEE 13th Intel. Transp. Syst. Conf. (ITSC)*, 2010, pp. 306–311.

[5] S. Tsugawa, S. Jeschke, and S. E. Shladover, "A review of truck platooning projects for energy savings," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 68–77, 2016.

[6] W. Levine and M. Athans, "On the optimal error regulation of a string of moving vehicles," *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 355–361, 1966.

[7] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 349–357, 1996.

[8] W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: Stability and string stability," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 620–633, 2011.

[9] A. Alam, B. Besselink, V. Turri, J. Mårtensson, and K. H. Johansson, "Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency," *IEEE Control Systems Magazine*, vol. 35, no. 6, pp. 34–56, 2015.

[10] A. Sciarretta, G. De Nunzio, and L. L. Ojeda, "Optimal ecodriving control: Energy-efficient driving of road vehicles as an optimal control problem," *IEEE Control Systems Magazine*, vol. 35, no. 5, pp. 71–90, 2015.

[11] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 12–28, 2016.

[12] N. Murgovski, B. Egardt, and M. Nilsson, "Cooperative energy management of automated vehicles," *Control Engineering Practice*, vol. 57, pp. 84–98, 2016.

[13] M. Hovgard, O. Jonsson, N. Murgovski, M. Sanfridson, and J. Fredriksson, "Cooperative energy management of electrified vehicles on hilly roads," *Control Engineering Practice*, vol. 73, pp. 66–78, 2018.

[14] D. Axehill and M. Morari, "An alternative use of the riccati recursion for efficient optimization," *Systems & Control Letters*, vol. 61, no. 1, pp. 37–40, 2012.

[15] G. Frison and J. B. Jørgensen, "Efficient implementation of the riccati recursion for solving linear-quadratic control problems," in *IEEE Inter. Conf. on Control Appl. (CCA)*, 2013, pp. 1117–1122.

[16] I. Nielsen, "Structure-exploiting numerical algorithms for optimal control," PhD thesis, Linköping University, 2017.

[17] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing, Madison, WI, 2017, vol. 2.

[18] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.

[19] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[20] J. Nocedal and S. Wright, *Numerical optimization*. Springer, 2006.

[21] R. Hult, M. Zanon, S. Gros, and P. Falcone, "A semidistributed interior point algorithm for optimal coordination of automated vehicles at intersections," *IEEE Transactions on Control Systems Technology*, 2021.

[22] S. Khoshfetrat Pakazad, A. Hansson, M. S. Andersen, and I. Nielsen, "Distributed primal–dual interior-point methods for solving tree-structured coupled convex problems using message-passing," *Optimization Methods and Software*, vol. 32, no. 3, pp. 401–435, 2017.

[23] S. Uebel, N. Murgovski, B. Bäker, and J. Sjöberg, "A two-level mpc for energy management including velocity control of hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5494–5505, 2019.

[24] A. Hamednia, N. K. Sharma, N. Murgovski, and J. Fredriksson, "Computationally efficient algorithm for eco-driving over long look-ahead horizons," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[25] R. Lacombe, S. Gros, N. Murgovski, and B. Kulcsár, "Bilevel optimization for bunching mitigation and eco-driving of electric bus lines," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[26] L. Guzzella and A. Sciarretta, *Vehicle propulsion systems*. Springer-Verlag, 2013, vol. 3.

[27] V. Turri, "Look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning," PhD thesis, KTH Royal Institute of Technology, 2018.

[28] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[29] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA, 2012, vol. 1.

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

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

[32] M. Diehl, "Real-time optimization for large scale nonlinear processes," PhD thesis, University of Heidelberg, 2001.

[33] M. Vukov, S. Gros, G. Horn, G. Frison, K. Geebelen, J. B. Jørgensen, J. Swevers, and M. Diehl, "Real-time nonlinear mpc and mhe for a large-scale mechatronic application," *Control Engineering Practice*, vol. 45, pp. 64–78, 2015.
[34] R. Hult, M. Zanon, G. Frison, S. Gros, and P. Falcone, "Experimental validation of a semi-distributed sequential quadratic programming method for optimal coordination of automated vehicles at intersections," *Optimal Control Applications and Methods*, vol. 41, no. 4, pp. 1068–1096, 2020.
[35] G. Still, "Lectures on parametric optimization: An introduction," *Optimization Online*, 2018.

Rémi Lacombe received the M.S. degree in applied mathematics jointly from ENSTA Paris, France, and from KTH Royal Institute of Technology, Sweden, in 2019. He is now pursuing the Ph.D. degree at the Department of Electrical Engineering, Chalmers University of Technology, Sweden. His main research interests include optimal control, nonlinear optimization, and their application to transportation systems.

Sébastien Gros received his Ph.D. degree from EPFL, Switzerland, in 2007. After a journey by bicycle from Switzerland to the Everest base camp in full autonomy, he joined a R&D group hosted at Strathclyde University focusing on wind turbine control. In 2011, he joined the university of KU Leuven, where his main research focus was on optimal control and fast NMPC for complex mechanical systems. He joined the Department of Signals and Systems at Chalmers University of Technology, Göteborg, in 2013, where he became associate Prof. in 2017. He is now full Prof. at NTNU, Norway, and affiliate Prof. at Chalmers. His main research interests include numerical methods, real-time optimal control, reinforcement learning, and the optimal control of energy-related applications.

Nikolce Murgovski Nikolce Murgovski received the M.S. degree in software engineering from University West, Trollhättan, Sweden, in 2007, and the M.S. degree in applied physics and the Ph.D. degree in systems and control from the Chalmers University of Technology, Gothenburg, Sweden, in 2007 and 2012, respectively. He is currently an Associate Professor with the Department of Electrical Engineering, Chalmers University of Technology. His research interest includes optimization and optimal control in the automotive area.

Balázs Kulcsár received the M.Sc. degree in traffic engineering and the Ph.D. degree from Budapest University of Technology and Economics (BUTE), Budapest, Hungary, in 1999 and 2006, respectively. He has been a Researcher/Post-Doctor with the Department of Control for Transportation and Vehicle Systems, BUTE, the Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN, USA, and with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. He is currently a Professor with the Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden. His main research interest focuses on traffic flow modeling and control.