PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

# GRAPH-BASED DISTRIBUTED MOTION PLANNING IN TIGHT MULTI-LANE PLATOONS

**GERMÁN EDUARDO PIZARRO LORCA**

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
FELIPE NÚÑEZ

Santiago de Chile, December 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

# GRAPH-BASED DISTRIBUTED MOTION PLANNING IN TIGHT MULTI-LANE PLATOONS

**GERMÁN EDUARDO PIZARRO LORCA**

Members of the Committee:
FELIPE NÚÑEZ
ALDO CIPRIANO
JUAN YUZ
MAURICIO LÓPEZ

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, December 2021

*Gratefully to my parents, sisters*
*and María Paz*

## ACKNOWLEDGEMENTS

To my advisor Felipe Núñez, who welcomed me in his research group. His constant support and advice is what made this thesis possible.

To my family for their unconditional support and blind faith in my work.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

LIST OF FIGURES vii

ABSTRACT ix

RESUMEN x

1. INTRODUCTION 1
   1.1. Motivation . . . . . 1
   1.2. Objectives, Methodology and Contributions . . . . . 4
   1.3. Organization . . . . . 4

2. PRELIMINARIES 6
   2.1. Notation . . . . . 6
   2.2. Vehicle Models . . . . . 6
      2.2.1. Vehicle State Representation . . . . . 6
      2.2.2. Road Area Occupied by the Vehicle . . . . . 8
   2.3. Platoon Representation . . . . . 10
   2.4. MPC . . . . . 11
   2.5. Centralized Motion Planning . . . . . 12
      2.5.1. Collision Avoidance Reformulation . . . . . 14

3. DISTRIBUTED MOTION PLANNING 17
   3.1. Distributed Motion Planning . . . . . 17
      3.1.1. NMPC . . . . . 18
      3.1.2. CA . . . . . 18
      3.1.3. Alternating Optimizations Scheme . . . . . 19
   3.2. Graph-based Distributed Motion Planning . . . . . 20
      3.2.1. Graph-based DNMPC . . . . . 21

3.2.2. Graph-based Alternating Optimizations Scheme . . . . . 22
3.3. Incremental Graph-based Distributed Motion Planning . . . . . 23
3.3.1. Incremental Graph-based Alternating Optimizations Scheme . . . . . 24
3.4. Reference Generation . . . . . 25
3.4.1. Graph Reference Generation . . . . . 26
3.4.2. Trajectory Reference Generation . . . . . 27

4. NUMERICAL EVALUATION 29
4.1. Scenarios . . . . . 29
4.1.1. Ideal Lane-change . . . . . 30
4.1.2. Platooning with Uncertainty in the States . . . . . 30
4.1.3. Internal Maneuvers in a Multi-lane Platoon . . . . . 30
4.1.4. Merging Lanes . . . . . 31
4.2. Evaluation Results for Scenario 1 . . . . . 33
4.3. Evaluation Results for Scenario 2 . . . . . 36
4.4. Evaluation Results for Scenario 3 . . . . . 40
4.5. Evaluation Results for Scenario 4 . . . . . 42
4.6. Discussion . . . . . 44

5. CONCLUSIONS 46
5.1. Open Challenges . . . . . 47

REFERENCES 49
A. Graph Reference Interpolation . . . . . 58
B. DNMPC Weight Matrices . . . . . 59
B.1. Distributed Motion Planner Weight Matrices . . . . . 59
B.2. Graph-based Distributed Planners Weight Matrices . . . . . 59

## LIST OF FIGURES


1.1 Common single-lane platoon connectivity topologies. (a) PF, (b) PFL, (c) BD, (d) BDL, (e) TPF, (f) TPFL (Li et al., 2017). . . . . 2

2.1 Kinematic Bicycle Model (Rajamani, 2011). . . . . 7

2.2 Two-dimensional polytope $\mathcal{P}$ describing the region occupied by a vehicle (Firoozi et al., 2021). . . . . 9

2.3 Dual interpretation of the shortest distance between two convex sets (Dax, 2006). . . . . 16

4.1 Scenario 3: Platoon formation for ten vehicles. The maneuvers involve vehicles 3 and 8 exchanging lanes. . . . . 31

4.2 Scenario 4: Lane merge formations of a six-vehicle group inside the platoon. 32

4.3 Results of scenario 1, lane change with enough longitudinal gap, for the Distributed Motion Planner. . . . . 33

4.4 Results of scenario 1, lane change with enough longitudinal gap, for the Graph-based Distributed Motion Planner. . . . . 34

4.5 Results of scenario 1, lane change with enough longitudinal gap, for the Incremental Graph-based Distributed Motion Planner. . . . . 34

4.6 Results of scenario 1, lane change with vehicles in parallel, for the Distributed Motion Planner. . . . . 35

4.7 Results of scenario 1, lane change with vehicles in parallel, for the Graph-based Distributed Motion Planner. . . . . 35

4.8 Results of scenario 1, lane change with vehicles in parallel, for the Incremental Graph-based Distributed Motion Planner. . . . . 36

4.9 Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Distributed Motion Planner. . . . . 37

4.10 Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Graph-based Distributed Motion Planner. . . . . 37

4.11 Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Incremental Graph-based Distributed Motion Planner. 38

4.12 Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Distributed Motion Planner. The dashed red line denotes a collision. . . . . 38

4.13 Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Graph-based Distributed Motion Planner. The dashed red line denotes a collision. . . . . 39

4.14 Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Incremental Graph-based Distributed Motion Planner. . . . . 39

4.15 Results of scenario 3, internal maneuvers, for the Distributed Motion Planner. 40

4.16 Results of scenario 3, internal maneuvers, for the Graph-based Distributed Motion Planner. . . . . 41

4.17 Results of scenario 3, internal maneuvers, for the Incremental Graph-based Distributed Motion Planner. . . . . 41

4.18 Results of scenario 4, merging lanes, for the Distributed Motion Planner. . . . 42

4.19 Results of scenario 4, merging lanes, for the Graph-based Distributed Motion Planner. . . . . 43

4.20 Results of scenario 4, merging lanes, for the Incremental Graph-based Distributed Motion Planner. . . . . 43

## ABSTRACT

Platooning of connected and automated vehicles (CAVs) has received extensive interest due to its potential to improve road traffic. In this context, multi-lane platoons have appeared as a generalized version of the classical train-like platoon structure used in the early platoon implementations from the 80s. Multi-lane platoons add structural flexibility to perform complex inter-vehicle maneuvers such as lane changes, coordination during bottlenecks, among others. Nonetheless, these advantages are at the price of more intricate internal dynamics. This work addresses motion planning in tight multi-lane platoons using distributed techniques that make use of a graph theoretical formulation of platoon formations, for reference generation, and distributed non-linear model predictive control, for vehicle command. The effectiveness of the proposed approaches is tested by simulating a variety of scenarios with progressive complexity, where it is shown that graph-based distributed approaches inherently promote cooperation within vehicles. Moreover, the second proposed approach, denominated incremental approach, is robust to sensor biases.



**Keywords**: Distributed control, Platooning, Motion planning

# RESUMEN

*Platooning* o coordinación en pelotones de vehículos conectados y automatizados (CAVs, por sus siglas en inglés) han recibido un gran interés debido a su potencial para mejorar el tráfico. En este contexto, los pelotones de múltiples carriles han aparecido como una versión generalizada de la estructura clásica de pelotón similar a un tren utilizada en las primeras implementaciones de los años 80. Los pelotones de múltiples carriles agregan flexibilidad estructural para realizar maniobras complejas entre vehículos, tales como cambios de carril, coordinación durante cuellos de botella, entre otros. No obstante, estas ventajas tienen el precio de una dinámica interna más intrincada. Este trabajo aborda la planificación del movimiento en pelotones estrechos de múltiples carriles utilizando técnicas distribuidas que hacen uso de una formulación teórica en base a grafos de formaciones de pelotones, para la generación de referencias, y el control predictivo distribuido no lineal, para el comando de vehículos. La efectividad de los enfoques propuestos se prueba mediante la simulación de una variedad de escenarios con complejidad progresiva, donde se muestra que los enfoques distribuidos basados en grafos promueven inherentemente la cooperación entre de los vehículos. Además, el segundo enfoque propuesto, denominado enfoque incremental, es robusto ante posibles sesgos de los sensores.

**Palabras Claves**: Control distribuido, Coordinación en pelotones, Planificación de movimiento

# 1. INTRODUCTION

## 1.1. Motivation

In the last decades, the automotive industry has been working intensively in developing the first fully functional autonomous vehicle. When operational, these vehicles, also known as connected and automated vehicles (CAVs), will communicate and collaborate, which is expected to improve drivers' efficiency, response, and comfort while enhancing safety and mobility. Furthermore, CAVs could improve road traffic, achieving efficiencies out of the reach of manually operated vehicle systems (Talebpour & Mahmassani, 2016).

In the context of CAVs, one of the simplest forms of collaboration is platooning. The earliest platooning implementation dates back to the PATH program in the 1980s (Shladover et al., 1991; S. E. Li et al., 2017) and has been studied since, in projects like CHAUFFEUR I and II (Harker, 2001), SARTRE (Chan, Gilhead, Jelinek, Krejci, & Robinson, 2012), and GCDC (Kianfar et al., 2012) in Europe, and EnergyITS in Japan (Tsugawa, Kato, & Aoki, 2011), where the potential of platooning to improve safety and traffic throughput was exposed. Recently, the potential of platooning to increase energy savings has also been observed (Wu, Bayen, & Mehta, 2018; Turri, Besselink, & Johansson, 2017; Smith et al., 2020).

The benefits of platooning are generally stated by understanding platoons as single lanes of vehicles, and its collective behavior is based on vehicles' mutual awareness of their states which is achieved by inter-vehicle sensing and communication. The information connectivity within CAVs affects the platoon behavior in terms of string stability (Swaroop & Hedrick, 1996; Seiler, Pant, & Hedrick, 2004; Wang & Nijmeijer, 2015), stability margin (Hao, Barooah, & Mehta, 2011; Zheng, Li, Li, & Wang, 2015), and coherence behavior (Lin, Fardad, & Jovanovic, 2011; Bamieh, Jovanovic, Mitra, & Patterson, 2012). Common communication topologies are presented in Fig. 1.1, which are Predecessor-Following (PF) and Bidiferectional (BD) topologies, Predecessor-following Leader (PFL), Bidirectional Leader (BDL) type, Two Predecessor-Following (TPF) and

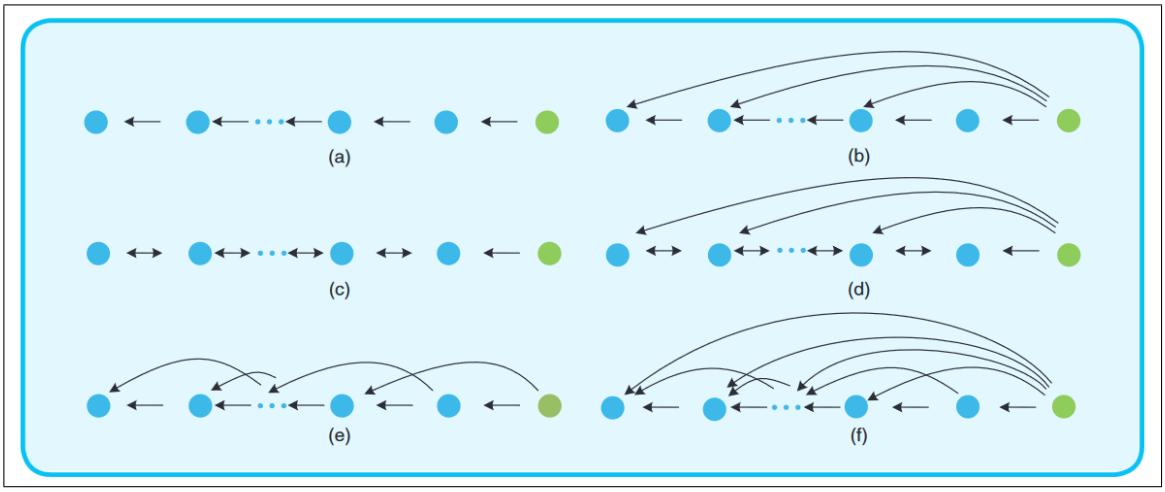


Figure 1.1. Common single-lane platoon connectivity topologies. (a) PF, (b) PFL, (c) BD, (d) BDL, (e) TPF, (f) TPFL (Li et al., 2017).

Two Predecessor-Following Leader (TPFL). (Zheng, Li, Wang, Cao, & Li, 2015; S. E. Li et al., 2017). For these train-like platoons, connectivity determines the control strategy, which is generally known as Adaptive Cruise Control (ACC) (Xiao & Gao, 2011; Wang & Nijmeijer, 2015; Dang, Wang, Li, & Li, 2015), and is mainly based on Model Predictive Control (Zheng, Li, Li, Borrelli, & Hedrick, 2016; Turri et al., 2017).

Previous works addressing complex maneuvers in platoons consider vehicles joining or leaving platoons (P. Liu, Kurt, & Ozguner, 2019; H. Liu, Zhuang, Yin, Tang, & Xu, 2018). These approaches are well suited for simple situations but insufficient when freeways operate at maximum capacity. Moreover, single-lane platoons are unable to handle lane-changes, which have been shown empirically as the primary source of traffic perturbations in multi-lane freeways (Ahn & Cassidy, 2007). This suggests that complex traffic maneuvers should be handled by multi-lane platoon approaches with collision avoidance (CA) in consideration.

Multi-lane platoons can be studied as multi robot coordination and formation control problems, where robots or vehicles are also denominated agents. Previous works addressing multi robot coordination include Dynamic window (Fox, Burgard, & Thrun, 1997), Priority Order (Čáp, Novák, Kleiner, & Selecký, 2015) and Scheduling (Bruni, Colombo,

& Del Vecchio, 2013). Nonetheless, these approaches do not consider the interaction within agents (Firoozi, Zhang, & Borrelli, 2021). Interaction within agents is investigated in formation control and classical works include Flocking and Particle Swarm Optimization (Olfati-Saber, 2006; Ren & Cao, 2011; Marjovi, Vasic, Lemaitre, & Martinoli, 2015), Sequence of Motion Primitives (Balch & Arkin, 1998), Potential Fields (Olfati-Saber & Murray, 2002; L. Gao, Chu, Cao, Lu, & Wu, 2019) and Learning-based methods (Chen, Liu, Everett, & How, 2017; Kreidieh, Wu, & Bayen, 2018; Fan, Long, Liu, & Pan, 2020; Busoniu, Babuska, & De Schutter, 2008). Flocking and Particle Swarm Optimization consider the interaction among agents as single points, which cannot model the CA within them. Sequence of Motion Primitives is difficult to mathematically analyze and solve. Potential Fields and Learning-based methods consider CA, however, it does not impose hard constraints on them. Another approach to study multi robot coordination and formation control is based on Constrained Optimization (Keviczky, Borrelli, Fregene, Godbole, & Balas, 2007; Alonso-Mora, Beardsley, & Siegwart, 2018; Firoozi et al., 2021; Firoozi, Ferranti, Zhang, Nejadnik, & Borrelli, 2020), nowadays, this approach is the only one that can handle explicit CA constraints with stability guarantees (Borrelli, Bemporad, & Morari, 2017). Up to date, the most advanced schemes targeting multi-lane platooning with constrained optimization are (Firoozi et al., 2021) and (Firoozi et al., 2020).

In (Firoozi et al., 2021), multi-lane platoons are considered from a centralized perspective where a non-linear model predictive control (NMPC) scheme is used to calculate control actions for each CAV, based on a set of global references for the platoon. Additionally, explicit CA constraints are incorporated by considering that the area occupied by each CAV should not intersect with any neighbor in the platoon. Although this approach shows good results in platoon reconfiguration and obstacle avoidance involving lane changes, its centralized formulation precludes scalability and is proposed only for small platoons.

Scalability is addressed in (Firoozi et al., 2020), with the so-called Distributed Motion Planner, in which the problem is solved by distributing the centralized NMPC problem into an alternating optimization scheme where distributed NMPC and CA are sequentially

solved, using a hand-crafted individual reference for each CAV. This approach for multi-lane platoons can effectively deal with lane-changes ensuring CA; yet, it does not exploit the shape of the platoon nor does it use information from neighbors in the cost function of the local NMPC problem, it does so only for CA. Moreover, the problem of generating the references for each CAV is not addressed in detail, adding an extra level of complexity.

## 1.2. Objectives, Methodology and Contributions

The main objective of this work is to propose a framework to solve the motion planning problem in multi-lane platoons, which must be scalable for a large number of vehicles, while ensuring collision free maneuvers without overestimating the vehicles sizes. To this end, we propose to tackle planning in tight multi-lane platoons using distributed formulations that explicitly consider the collaborative nature of the platoon.

Consequently, the main contributions of this work are two frameworks for distributed multi-lane motion planning. Both schemes use a graph-based description of the platoon to formulate a distributed NMPC problem (Constraint-based Optimization) that tracks references involving a set of neighboring CAVs in the platoon. First, an initial approach where each CAV takes states referenced to a global reference frame, and a second incremental scheme where each CAV takes states referenced to a local reference frame from its neighbors and uses information from local sensors to refer neighbors' information to its own local reference frame. A detailed simulation study compares the performance of the proposed techniques with a benchmark under a variety of scenarios, ranging from simple lane changes to motion planning in large-scale multi-lane platoons.

## 1.3. Organization

The rest of this thesis is organized as follows. In chapter 2, preliminaries on which this work is built are introduced. In chapter 3, the proposed graph-based distributed motion

planners are described in detail, as well as the benchmark planner. Chapter 4 summarizes the numerical evaluation performed. Finally, concluding remarks are given in chapter 5.

# 2. PRELIMINARIES

## 2.1. Notation

Throughout this manuscript, $\mathbb{R}$ denotes the real numbers, $\mathbb{R}^n$ the Euclidean space of dimension $n$, and $\mathbb{R}^{n\times m}$ the set of $n \times m$ matrices with real coefficients. Vectors and matrices are denoted in boldface, the latter also denoted in capital letters. For a vector $\mathbf{v}$, $\mathbf{v}^\top$ denotes its transpose. The standard Euclidean distance is denoted by $\|\cdot\|_2$, while $\|\mathbf{x}\|_Q := \sqrt{\mathbf{x}^\top \mathbf{Q}\mathbf{x}}$ is defined for $\mathbf{x} \in \mathbb{R}^n$ and a positive semi-definite matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$. For a countable set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality.

## 2.2. Vehicle Models

In the following sections, the two vehicle models used in this work are described. The first one corresponds to the state model that could be easily computed and has a wide operation range, while the second one corresponds to the region occupied by each vehicle and is used to avoid collisions.

### 2.2.1. Vehicle State Representation

In the literature, many state vehicle models are presented, ranging from low dimension point mass-models to high fidelity multi-body models (Dixit et al., 2018). These models are designed mainly to capture the longitudinal, lateral and yaw dynamics of the vehicles and have been documented in (Rajamani, 2011). For multi-lane platoons, in most cases re-configurations include lateral maneuvers. Therefore, a model with lateral dynamics should be considered.

Lateral models have been reviewed in detail (Snider et al., 2009; Rajamani, 2011; Amer, Zamzuri, Hudha, & Kadir, 2017; Sorniotti, Barber, & De Pinto, 2017; Rupp & Stolz, 2017) because the performance of a close loop tracking controller depends on the accuracy of the modelled system dynamics. Overall, the preferred ones are the dynamic

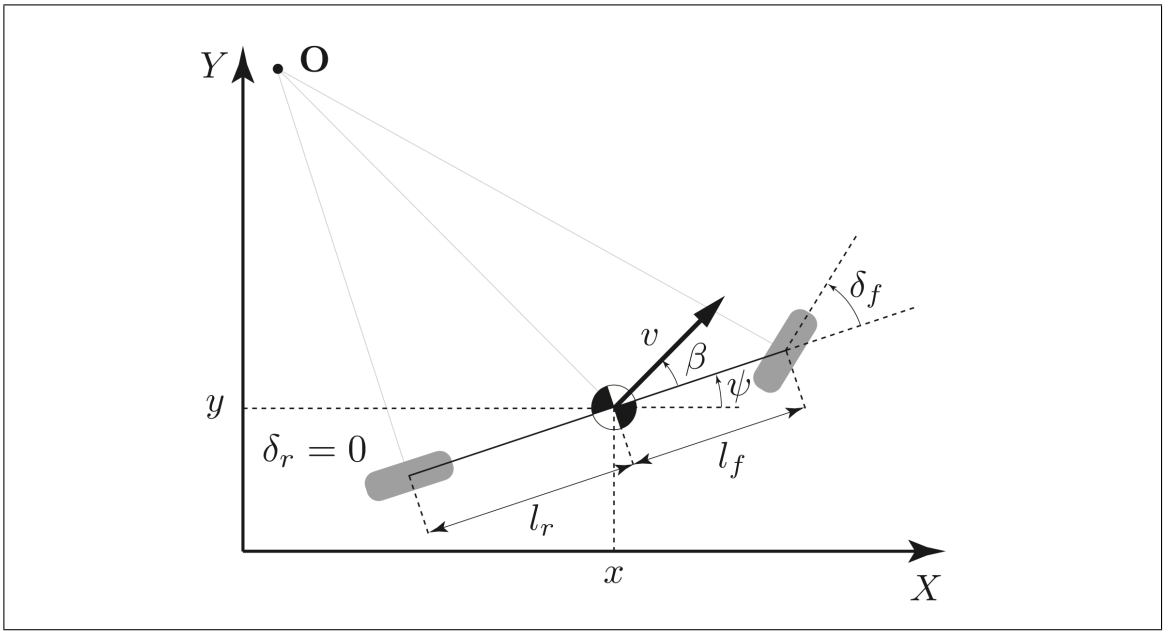


Figure 2.1. Kinematic Bicycle Model (Rajamani, 2011).

bicycle model and the kinematic bicycle model (KBM) since there is a good compromise between computational complexity and accuracy for control design related to highway driving applications (Kong, Pfeiffer, Schildbach, & Borrelli, 2015).

The dynamic bicycle model has high fidelity since it includes the tire and the road interaction, which means that a tracking controller could be developed with this model to perform well within the dynamic limits of the vehicle (i.e., lateral acceleration, vehicle side-slip, and yaw rate) (Dixit et al., 2018). Nonetheless, the KBM requires less computational power, it is easier because there are only two parameters to identify, and can be implemented over a wide range of vehicle speeds (Kong et al., 2015). Therefore, in this work vehicles are going to be modelled with the KBM. However, at higher speeds dynamic models should be used.

Accordingly, for a given vehicle labeled by the subscript $i$, the KBM defines a state vector as $\mathbf{z}_i = [x_i, y_i, \psi_i, v_i]^\top$, ruled by the following dynamic equations:

$$
\begin{aligned}
\dot{x}_i &= v_i \cos(\psi_i + \beta_i), & \dot{y}_i &= v_i \sin(\psi_i + \beta_i), \\
\dot{\psi}_i &= \frac{v_i \cos(\beta_i)}{l_{f,i} + l_{r,i}} \tan(\delta_i), & \dot{v}_i &= a_i,
\end{aligned} \tag{2.1}
$$

where $x_i$ and $y_i$ represent the longitudinal and lateral position, respectively; $\psi_i$ is the heading angle and $v_i$ is the velocity at the center of gravity (CG). The side slip angle is defined as $\beta_i := \arctan(\frac{l_{r,i}}{l_{f,i}+l_{r,i}} \tan(\delta_i))$, $l_{f,i}$ and $l_{r,i}$ are the longitudinal distance from the CG to front and rear tires, respectively. The control input is denoted by $\mathbf{u}_i = [a_i, \delta_i]^\top$, where $a_i$ is the vehicle's acceleration and $\delta_i$ is the steering angle. For implementation purposes, the model is discretized using Euler's method with sample time $\Delta t$ as:

$$
\begin{aligned}
x_i(t+1) &= x_i(t) + v_i(t) \cos(\psi_i(t) + \beta_i(t))\Delta t, \\
y_i(t+1) &= y_i(t) + v_i(t) \sin(\psi_i(t) + \beta_i(t))\Delta t, \\
\psi_i(t+1) &= \psi_i(t) + \frac{v_i(t) \cos(\beta_i(t))}{l_{i,f} + l_{i,r}} \tan(\delta_i)\Delta t, \\
v_i(t+1) &= v_i(t) + a_i(t)\Delta t.
\end{aligned} \tag{2.2}
$$

Finally, for notation compactness, the KBM is defined by the function $f : \mathbb{R}^4 \times \mathbb{R}^2 \to \mathbb{R}^4$ where

$$
\mathbf{z}_i(t+1) = f(\mathbf{z}_i(t), \mathbf{u}_i(t)) \tag{2.3}
$$

### 2.2.2. Road Area Occupied by the Vehicle

Modeling vehicles as point-masses with the KBM is useful for approximating the states. However, when using the KBM the spatial notion of the vehicle is lost, which is critical for motion planning in tight platoons where both the road (lane width) and platoon (longitudinal and lateral inter-vehicle spacing) geometries restrict the motion of the vehicles. Therefore, it is essential to consider vehicle dimensions with their exact sizes (Firoozi et al., 2021).

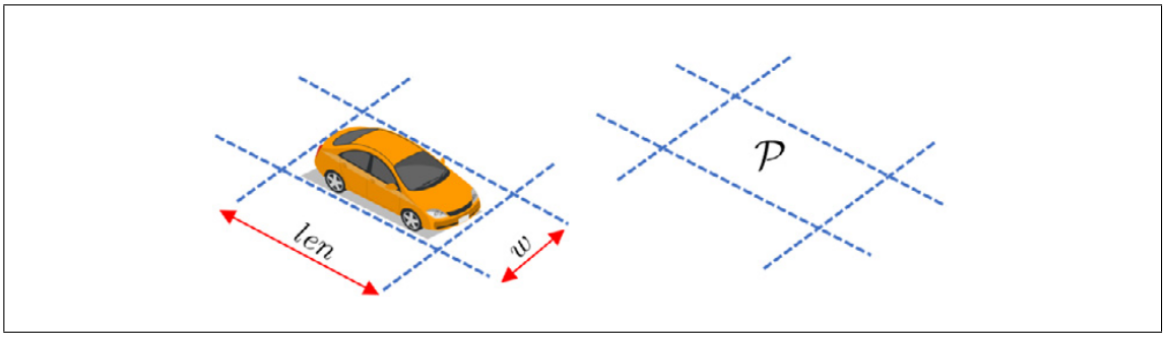


Figure 2.2. Two-dimensional polytope $\mathcal{P}$ describing the region occupied by a vehicle (Firoozi et al., 2021).

Following (Firoozi et al., 2021), we define the region occupied by each vehicle as a convex polytope $\mathcal{P}$ (see Fig. 2.2). Polytopes are described as the intersection of a set of half-spaces and formulated as a set of inequalities (Firoozi et al., 2020). In the platooning context, two-dimensional polytopes are used.

As the vehicle moves, the physical space it occupies naturally changes. Consider that each vehicle $i$ has an initial pose $\mathcal{P}_{i,0}$, when travelling along the road the polytope can be determined at any time instant $t$ from the vehicle's state vector $\mathbf{z}_i$ using affine transformations, as (Firoozi et al., 2021): $\mathcal{P}(\mathbf{z}_i(t)) = \mathbf{R}(\mathbf{z}_i(t))\mathcal{P}_{i,0} + \mathbf{t}(\mathbf{z}_i(t))$, where $\mathbf{R} : \mathbb{R}^4 \to \mathbb{R}^{2\times 2}$ is a matrix-valued function that outputs orthogonal rotation matrices and $\mathbf{t} : \mathbb{R}^4 \to \mathbb{R}^2$ is a translation mapping. For each $t$, the rotation matrix $\mathbf{R}(\mathbf{z}_i(t))$ is defined using the heading angle $\psi_i$,

$$\mathbf{R}(\mathbf{z}_i(t)) = \begin{bmatrix} \cos\psi_i(t) & -\sin\psi_i(t) \\ \sin\psi_i(t) & \cos\psi_i(t) \end{bmatrix}, \tag{2.4}$$

while the translation vector $\mathbf{t}(\mathbf{z}_i(t))$ depends on $x_i(t)$ and $y_i(t)$. The transformed polytope is given by (Firoozi et al., 2021):

$$\mathcal{P}(\mathbf{z}_i(t)) := \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 : \mathbf{A}(\mathbf{z}_i(t)) \begin{bmatrix} x \\ y \end{bmatrix} \leq \mathbf{b}(\mathbf{z}_i(t)) \right\}, \tag{2.5}$$

where $\mathbf{A}$ and $\mathbf{b}$ are defined as

$$\mathbf{A}(\mathbf{z}_i(t)) = \begin{bmatrix} \mathbf{R}(\mathbf{z}_i(t))^\top \\ -\mathbf{R}(\mathbf{z}_i(t))^\top \end{bmatrix},$$
$$\mathbf{b}(\mathbf{z}_i(t)) = [h_i/2, w_i/2, h_i/2, w_i/2]^\top + \mathbf{A}(\mathbf{z}_i(t))[x(t), y(t)]^\top,$$

with $h_i$ and $w_i$ being the length and the width of vehicle $i$, respectively.

### 2.3. Platoon Representation

A *platoon* is understood as a set $\mathcal{V} \coloneqq \{1, 2, \ldots, N\}$ of cooperative vehicles, where $N < \infty$ is the size of the platoon. Elements inside the platoon cooperate to fulfill a collective objective, as a common speed or safety distance constraints. For motion planning purposes, in this work we define the *platoon formation graph* as an undirected graph $\mathcal{G} \coloneqq \{\mathcal{V}, \mathcal{E}\}$, where the edge set $\mathcal{E}$ is such that $(i, j), (j, i) \in \mathcal{E}$ if there exists a set of constraints involving the internal states of vehicles $i$ and $j$. For each vehicle $i \in \mathcal{V}$, a neighbor set is defined as $\mathcal{N}_i := \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. $\mathcal{G}$ is assumed to be a connected graph.

In this work, we consider the presence of a virtual leader in the platoon. Therefore, we introduce the *platoon formation graph with virtual leader* as a directed graph $\bar{\mathcal{G}}_N \coloneqq (\bar{\mathcal{V}}_N, \bar{\mathcal{E}}_N)$, where a virtual leader node, labeled as node $0$, is included, yielding $\bar{\mathcal{V}}_N \coloneqq \{0\} \cup \mathcal{V}$ and $\bar{\mathcal{E}}_N \coloneqq \{(0, i)\} \cup \mathcal{E}$. $\bar{\mathcal{G}}_N$ is directed since edges $(0, i)$ imply that information is only transmitted from the virtual leader to the other vehicles in the platoon.

The *platoon formation graph with virtual leader* is finally enhanced with a reference set $\mathcal{D}_N$, which is such that $\mathcal{D}_N \coloneqq \{\mathbf{d}_{ij} \in \mathbb{R}^4 | (i, j) \in \bar{\mathcal{E}_N}\}$ where $\mathbf{d}_{ij}$ represents an equality constraint on the absolute value of the component-wise difference between the states of vehicles $i, j \in \bar{\mathcal{V}}_N$.

## 2.4. MPC

Model Predictive Control (MPC) refers to a class of control algorithms that utilize a dynamic optimization with an explicit model to predict the future response of a plant (Qin & Badgwell, 2003), for example

$$\mathbf{x}(k+1) = g(\mathbf{x}(k), \mathbf{u}(k)), \quad \mathbf{x}(0) = \mathbf{x}_0$$

that describes the evolution of the state $\mathbf{x}(k)$ with time, starting from the initial condition $\mathbf{x}(0)$, as it is affected by the manipulated input $\mathbf{u}(k)$. Here $g(\mathbf{x}(k), \mathbf{u}(k))$ could be a linear or nonlinear function, for the nonlinear case, we refer to the problem as a Nonlinear MPC (NMPC). The goal of MPC procedure is to find a sequence of manipulated inputs $\mathbf{u}_i(\cdot|t) = [\mathbf{u}_i(k|t), \ldots, \mathbf{u}_i(k+N_c-1|t)]$ such that the objective is optimized over some time horizon $N_c$, typically

$$\min_{\mathbf{u}_i(\cdot|t)} \sum_{k=t}^{t+N_c} q(\mathbf{x}(k), \mathbf{u}(k)) + p(\mathbf{x}(N))$$

The terms $q(\mathbf{x}(k), \mathbf{u}(k))$ and $p(\mathbf{x}(N))$ are referred as the *stage cost* and the *terminal cost*. Many practical problems can be put into this form and many algorithms and software packages are available to determine the optimal solution vector $\mathbf{u}_i(\cdot|t)$. The various algorithms exploit the structure of the particular problem, e.g., linearity and convexity, so that even large problems described by complex models and involving many degrees of freedom can be solved efficiently and reliably.

One difficulty with this idea is that, in practice, the sequence of $\mathbf{u}_i(\cdot|t)$, which is obtained by this procedure cannot be simply applied. The model of the system predicting its evolution is usually inaccurate and the system may be affected by external disturbances that may cause its path to deviate significantly from the one that is predicted. Therefore, it is common practice to measure the state after some time period, say one time step, and to solve the dynamic optimization problem again, starting from the measured state $\mathbf{x}(t+1)$

as the new initial condition. This feedback of the measurement information to the optimization endows the whole procedure with a robustness typical for closed-loop systems (Borrelli et al., 2017). What was described above is referred as the Receding Horizon Principle and it is a central part of the MPC formulation.

MPC formulation integrates optimal control, stochastic control, control of processes with dead time, multivariable control and future references when available (Camacho & Alba, 2013). Moreover, it has evolved to dominate the process industry, where it has been employed for thousands of problems (Qin & Badgwell, 2003). The popularity of MPC stems from the fact that the resulting operating strategy respects all the system and problem details, including interactions and constraints, something that would be very hard to accomplish in any other way. Indeed, often MPC is used for the regulatory control of large multivariable linear systems with constraints, where the objective function is not related to an economical objective, but is simply chosen in a mathematically convenient way, namely quadratic in the states and inputs, to yield a "good" closed-loop response. Again, there is no other controller design method available today for such systems, that provides constraint satisfaction and stability guarantees (Borrelli et al., 2017).

Concerning the path planning problem, MPC is suited for this class of problems since it can handle Multi-Input Multi-Output systems with input and state constraints while taking into account the nonlinear dynamics of the vehicle. The design task of path-generation and path-following is simplified, e.g., the reference path fed to the MPC controller does not need to be physically realizable, i.e respect to dynamic constraints, since the MPC optimization would find a solution that satisfy a feasible dynamic (Kong et al., 2015).

## 2.5. Centralized Motion Planning

Following (Firoozi et al., 2021), the centralized motion planner is formulated as an NMPC problem that computes a collision-free trajectory for $N$ vehicles. The NMPC

problem is formulated as

$$\min_{\mathbf{u}_i(\cdot|t)} \sum_{i=1}^{N} J_{1,i}(\mathbf{z}_i, \mathbf{u}_i) \tag{2.6a}$$

$$\text{s. t.} \quad \mathbf{z}_i(k+1|t) = f\big(\mathbf{z}_i(k|t), \mathbf{u}_i(k|t)\big) \tag{2.6b}$$

$$\mathbf{z}_i(k=t|t) = \mathbf{z}_i(t) \tag{2.6c}$$

$$\mathbf{z}_i(k|t) \in \mathcal{Z}_i,\ \mathbf{u}_i(k|t) \in \mathcal{U}_i \tag{2.6d}$$

$$\mathcal{P}(\mathbf{z}_i(k|t)) \cap \mathcal{P}(\mathbf{z}_j(k|t)) = \emptyset,$$
$$i \neq j\ \forall i \in \mathcal{V},\ j \in \mathcal{N}_i \tag{2.6e}$$

$$J_{1,i}(\mathbf{z}_i, \mathbf{u}_i) = \sum_{k=t}^{t+N_c} \|\mathbf{z}_i(k|t) - \mathbf{z}_{\text{i,Ref}}(k|t)\|^2_{\mathbf{Q}_z} + \sum_{k=t}^{t+N_c-1} \big(\|\mathbf{u}_i(k|t)\|^2_{\mathbf{Q}_u} + \|\Delta\mathbf{u}_i(k|t)\|^2_{\mathbf{Q}_{\Delta u}}\big), \tag{2.7}$$

where $\mathbf{u}_i(\cdot|t) = [\mathbf{u}_i(k|t), \ldots, \mathbf{u}_i(k+N_c-1|t)]$ is the control sequence optimized by the NMPC over the planning horizon $N_c$ for the $i$th vehicle in the platoon. $\mathbf{z}_i(\cdot|t) = [\mathbf{z}_i(k|t), \ldots, \mathbf{z}_i(k+N_c|t)]$ is the $i$th vehicle state sequence. The cost function $J_{1,i}(\mathbf{z}_i, \mathbf{u}_i) : \mathbb{R}^4 \times \mathbb{R}^2 \to \mathbb{R}$ is calculated for the $N$ vehicles in the platoon and penalizes: i) the deviation of the states $\mathbf{z}$ from a reference sequence $\mathbf{z}_{ref}$, which is not necessarily smooth and collision free, yet the optimal trajectory has these properties since they are ensured by the optimization; and ii) the input effort $\mathbf{u}$ and input rate effort $\Delta\mathbf{u}$. The three terms forming the cost function are weighted using positive semi-definite matrices $\mathbf{Q}_z, \mathbf{Q}_u$ and $\mathbf{Q}_{\Delta u}$, respectively. In terms of constraints, (2.6b) corresponds to the nonlinear vehicle model, which in this work is given by the KBM, (2.6c) defines the initial conditions for NMPC, (2.6d) defines the feasible state and input sets, given by road characteristics and the actuators and vehicles' operation range, and (2.6e) enforces CA by checking that the intersection between polytopes is empty.

### 2.5.1. Collision Avoidance Reformulation

Constraints (2.6e) use polytopes to address CA, which hinders solving (2.6) since in general (2.6e) are non-convex and non-differentiable. Hence, the condition is reformulated to preserve continuity and differentiability, which enables using existing gradient- and hessian-based optimization algorithms (Schulman et al., 2014; Zhang, Liniger, & Borrelli, 2020).

A common method of formulating CA is with the notion of distance between two polytopes, where each one is a closed convex set modeled using half-spaces.

$$\begin{aligned} \text{dist}(\mathcal{P}_i, \mathcal{P}_j) \coloneqq & \min_{\mathbf{x},\mathbf{y}} \|\mathbf{x}-\mathbf{y}\|_2 \\ & \text{s.t.} \quad \mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i, \mathbf{A}_j\mathbf{y} \leq \mathbf{b}_j. \end{aligned} \tag{2.8}$$

This formulation means that collisions are avoided when $\text{dist}(\mathcal{P}_i, \mathcal{P}_j) > 0$. In practice, to deal with uncertainties of physical models and stochastic measurement errors, the distance is restricted to a minimum safe distance as $\text{dist}(\mathcal{P}_i, \mathcal{P}_j) > d_{min}$, where $d_{min}$ is calibrated for each specific application.

The distance formulation (2.8) is per se an optimization problem that cannot be incorporated in (2.6) as a constraint. Nonetheless, the primal problem of minimum distance between two convex sets can be reformulated to its dual problem as the maximal "separation" between them (Dax, 2006). The strong dual problem was used in (Zhang et al., 2020; Firoozi et al., 2021, 2020) to include the reformulated CA into (2.6). To reformulate (2.8), the Karush–Kuhn–Tucker (KKT) conditions are used resulting in

$$\begin{aligned} \text{dist}(\mathcal{P}_i, \mathcal{P}_j) := & \max_{\boldsymbol{\lambda}_{ij},\boldsymbol{\lambda}_{ji},\mathbf{s}_{ij}} -\mathbf{b}_i^\top\boldsymbol{\lambda}_{ij} - \mathbf{b}_j^\top\boldsymbol{\lambda}_{ji} \\ & \text{s.t. } \mathbf{A}_i^\top\boldsymbol{\lambda}_{ij} + \mathbf{s}_{ij} = 0, \mathbf{A}_j^\top\boldsymbol{\lambda}_{ji} - \mathbf{s}_{ij} = 0, \\ & \|\mathbf{s}_{ij}\|_2 \leq 1, -\boldsymbol{\lambda}_{ij} \leq 0, -\boldsymbol{\lambda}_{ji} \leq 0, \end{aligned} \tag{2.9}$$

where $\boldsymbol{\lambda_{ij}}$, $\boldsymbol{\lambda_{ji}}$ and $\mathbf{s}_{ij}$ are the dual variables. Then, (2.9) is incorporated in (2.6) and the platoon motion planning results in

$$\min_{\substack{\mathbf{u}_i(\cdot|t),\\ \boldsymbol{\lambda}_{ij}(\cdot|t)\\ \boldsymbol{\lambda}_{j,i}(\cdot|t),\\ \mathbf{s}_{ij}(\cdot|t)}} \sum_{i=1}^{N} J_{1,i}(\mathbf{z}_i, \mathbf{u}_i)$$

s.t. (2.6b), (2.6c), (2.6d),

$$\mathbf{A}(\mathbf{z}_i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) + \mathbf{s}_{ij}(k|t) = 0, \tag{2.10a}$$

$$\mathbf{A}(\mathbf{z}_j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t) - \mathbf{s}_{ij}(k|t) = 0, \tag{2.10b}$$

$$\big( - \mathbf{b}(\mathbf{z}_i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) - \mathbf{b}(\mathbf{z}_j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t)\big) \geq d_{min}, \tag{2.10c}$$

$$\|\mathbf{s}_{ij}\|_2 \leq 1, -\boldsymbol{\lambda}_{ij} \leq 0, -\boldsymbol{\lambda}_{ji} \leq 0, \tag{2.10d}$$

$$\forall i \in \mathcal{V},\ j \in \mathcal{N}_i,$$

where (2.10a), (2.10b), (2.10c) and (2.10d) are solved for each vehicle $i \in \mathcal{V}$ with respect to each of its neighbors $j \in \mathcal{N}_i$. By adding sequences of dual variables to (2.10), more variables need to be optimized; yet, the CA constraint can be solved using gradient- and hessian-based optimization algorithms.

As mentioned earlier, the geometric interpretation behind the dual problem is that instead of calculating the shortest distance between two convex sets, the new formulation calculates the maximal "separation" between the sets, where the term "separation" refers to the distance between a pair of parallel hyperplanes that separates the two sets (see Fig. 2.3)(Dax, 2006).

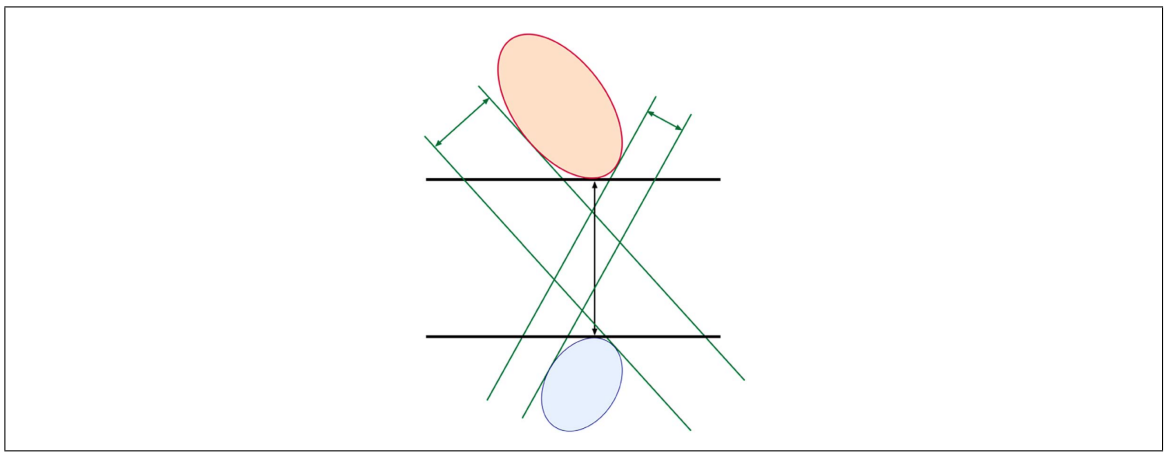

Figure 2.3. Dual interpretation of the shortest distance between two convex sets (Dax, 2006).

# 3. DISTRIBUTED MOTION PLANNING*

Centralized motion planning is fully capable of coordinating vehicles inside a platoon. However, its computational complexity increases proportionally to the number of vehicles in the platoon (Firoozi et al., 2020); therefore, using centralized planning in real time for a large platoon becomes infeasible. In (Firoozi et al., 2020) distributed motion planning was introduced by distributing (2.10) into two alternating optimizations, where each vehicle solves locally a sub-problem in parallel.

In the following, three distributed motion planning techniques are presented, starting with the approach introduced in (Firoozi et al., 2020), which will serve as inspiration and benchmark for the two novel graph-based approaches.

## 3.1. Distributed Motion Planning

To distribute (2.10), the vehicles should be decoupled from each other. Particularly in (2.10), vehicles are coupled through CA constraints (2.10a)-(2.10c). In (Firoozi et al., 2020), distribution is achieved by alternately solving two independent optimization problems. The first one consists of an NMPC where each vehicle solves its own states and inputs while keeping the neighbors dual variables fixed. The second problem solves for the CA dual variables while keeping the states fixed.

* Parts of the material presented in this chapter were published in: Pizarro, G., & Núñez, F. (2021). Graph-based distributed lane-change in tight multi-lane platoons. In *2021 IEEE Conference on Control Technology and Applications (CCTA)* (p. 1031-1036); and in Pizarro, G., & Núñez, F. (2021). Graph-based distributed motion planning in tight multi-lane platoons. *Control Engineering Practice*, under review.

### 3.1.1. NMPC

The first optimization is a Distributed Non-linear Model Predictive Control (DNMPC) problem formulated as

$$\min_{\mathbf{u}_i(\cdot|t)} J_{1,i}(\mathbf{z}_i, \mathbf{u}_i)$$

$$\text{s. t.} \quad \text{(2.6b), (2.6c), (2.6d),}$$

$$\big( - \mathbf{b}(\mathbf{z}_i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) - \mathbf{b}(\bar{\mathbf{z}}_j(k|t))^\top \bar{\boldsymbol{\lambda}}_{ji}(k|t)\big) \geq d_{min} \tag{3.1a}$$

$$\mathbf{A}(\mathbf{z}_i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) + \bar{\mathbf{s}}_{ij}(k|t) = 0 \tag{3.1b}$$

$$\forall j \in \mathcal{N}_i,$$

where constant variables are denoted by $\bar{(\cdot)}$. It can be seen that in the CA constraints (3.1a) and (3.1b), the dual variables and $\mathbf{b}$ matrices are fixed.

### 3.1.2. CA

Regarding CA, the corresponding optimization solves for the dual variables, while keeping the ego and neighbor states fixed, as follows

$$\max_{\substack{\boldsymbol{\lambda}_{ij}(\cdot|t),\\ \boldsymbol{\lambda}_{ji}(\cdot|t),\\ \mathbf{s}_{ij}(\cdot|t)}} - \mathbf{b}(\bar{\mathbf{z}}_i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) - \mathbf{b}(\bar{\mathbf{z}}_j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t)$$

$$\text{s. t.} \quad \mathbf{A}(\bar{\mathbf{z}}_i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) + \mathbf{s}_{ij}(k|t) = 0, \tag{3.2a}$$

$$\mathbf{A}(\bar{\mathbf{z}}_j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t) - \mathbf{s}_{ij}(k|t) = 0, \tag{3.2b}$$

$$\big( - \mathbf{b}(\bar{\mathbf{z}}_i(k|t))^\top \boldsymbol{\lambda}_{ij}(k|t) - \mathbf{b}(\bar{\mathbf{z}}_j(k|t))^\top \boldsymbol{\lambda}_{ji}(k|t)\big) \geq d_{min}, \tag{3.2c}$$

$$\|\mathbf{s}_{ij}\|_2 \leq 1, -\boldsymbol{\lambda}_{ij} \leq 0, -\boldsymbol{\lambda}_{ji} \leq 0, \tag{3.2d}$$

$$\forall j \in \mathcal{N}_i, \forall k \in \{1, 2, ..., N\}.$$

**Algorithm 1** Alternating Optimizations Scheme

Initialize $[\mathbf{s}_{ij}(t), \ldots, \mathbf{s}_{ij}(t+N_c|t)]$,
$[\boldsymbol{\lambda}_{ij}(t), \ldots, \boldsymbol{\lambda}_{ij}(t+N_c|t)]$,
$[\boldsymbol{\lambda}_{ji}(t), \ldots, \boldsymbol{\lambda}_{ji}(t+N_c|t)]$
**for** $t = 0, 1, \ldots, \infty$ **do**
  **for all** vehicle $i$, $i \in \mathcal{V}$ **do in parallel**
    Solve DNMPC Optimization Problem (3.1)
    Compute the shifted sequence $\mathbf{z}_i(\cdot|t) =$
    $[\mathbf{z}_i(t+1|t), \ldots, \mathbf{z}_i(t+N_c|t), \mathbf{z}_i(t+N_c|t)]$.
    Compute the polytopes of the sequence $\mathbf{A}\big(\mathbf{z}_i(\cdot|t)\big)$ and $\mathbf{b}\big(\mathbf{z}_i(\cdot|t)\big)$.
    Communicate the polytopes to agent $j, \forall j \in \mathcal{N}_i$ t
    Solve CA Optimization Problem (3.2)
    Apply $\mathbf{u}_i(t|t)$ obtained from (3.1) to move

With this formulation, the CA is a convex optimization problem and can be solved separately for each pair of neighbor vehicles. Therefore, it should be solved fast enough using a state-of-the-art convex solver.

### 3.1.3. Alternating Optimizations Scheme

Taking into account both optimizations, the Alternating Optimizations Scheme is executed following Algorithm 1. First, the dual variables are initialized and each vehicle solves its own NMPC Problem. Then, the optimal state sequences computed by the NMPC optimization are shifted one time step to address communication delays, followed by the calculation of the polytope sequence using the shifted state sequence. After that, the polytopes are communicated to all the agents in the neighbor set $\mathcal{N}_i$ and each vehicle solves in parallel the CA optimization. Finally, each vehicle applies the optimal input $\mathbf{u}_i(t|t)$ calculated in the first step and the algorithm is repeated (Firoozi et al., 2020).

Note that the distributed algorithm requires that the vehicles have their clocks synchronized, which is indispensable for a correct operation. The synchronization must be performed before Algorithm 1 begins and, it should be constantly corrected. Although the synchronization problem is not taken into account in this work, the distributed synchronization problem is well studied and, could be easily solved for connected graphs networks

(Núñez, Wang, & Doyle, 2014) like the platoons using an external synchronization loop that runs in parallel to Algorithm 1.

The presented distributed motion planning scheme provides an effective alternative to centralized motion planning. However, there are still some issues that must be addressed. First, individual references for each vehicle must be designed, which are then tracked individually by each vehicle without cooperating. Reference generation is left open in (Firoozi et al., 2020), where it is also stated that (2.10) might become infeasible and that persistent feasibility could be guaranteed by computing the reachable set. Infeasibility can be caused by long-term maneuvers that the fixed-length prediction horizon is not capable of foreseeing. Although references could be easily generated for each vehicle, finding a reachable reference for long-term maneuvers could become unscalable. A second issue is related to the shifted state sequence. Although shifting accounts for communication delays, at the end of the sequence the last state is just copied, which conceptually means that vehicles impulsively stop moving and could cause a stabilization at a speed lower than desired. Finally, a third issue is related to the neighbor set used in (Firoozi et al., 2020), where a neighbor is merely defined as a vehicle inside the transmission range. Under this setup, the problem grows in size as the platoon becomes larger, reducing the scalability.

### 3.2. Graph-based Distributed Motion Planning

To address the feasibility and scalablity issues that arise from reference generation, we propose a Graph-based Distributed Motion Planner. The proposed planner is conceptually different in the sense that the platoon tracks a formation, instead of each vehicle tracking an individual reference. Formation tracking enables a collaborative behavior, which promotes a smoother and faster solution.

In Graph-based Distributed Motion Planning, the platoon formulation takes relevance. Platoons are modeled as *platoon formation graphs with virtual leader*, which is particularly useful to model the interactions within vehicles and is essential to track references

collaboratively within the platoon. Based on the formation, vehicles track relative distances with respect to their neighbors, which makes vehicles aware of how close they are to their neighbors. In the graph-based setup, maneuvers are modeled by formation (graph) transitions, which gives freedom on how to generate references.

### 3.2.1. Graph-based DNMPC

Collaboration is achieved by adapting (3.1) to use the platoon formation. Constraints are the same as the vanilla case, while the main difference is in the cost function, which now consists of three parts, as follows,

$$
\begin{aligned}
J_{2,i}(\mathbf{z}_i, \mathbf{u}_i) = & \sum_{k=t}^{t+N_c-1} \left( \|\mathbf{u}_i(k|t)\|^2_{\mathbf{Q}_u} + \|\Delta\mathbf{u}_i(k|t)\|^2_{\mathbf{Q}_{\Delta u}} \right) \\
& + \sum_{j \in \mathcal{N}_i \setminus 0} \sum_{k=t}^{t+N_c} \frac{1}{|\mathcal{N}_i \setminus 0|} \|\mathbf{z}_i(k|t) - \mathbf{z}_j(k|t) - \mathbf{d}_{ij}(k|t)\|^2_{\mathbf{Q}_n} \\
& + \sum_{k=t}^{t+N_c} \|\mathbf{z}_i(k|t) - \mathbf{d}_{i0}(k|t)\|^2_{\mathbf{Q}_0},
\end{aligned} \tag{3.3}
$$

where the first term weights the the input effort $\mathbf{u}$ and input rate effort $\Delta\mathbf{u}$, similar to (2.7), the second term is in charge of tracking the distance references of vehicle $i$ with respect to its neighbor $j \in \mathcal{N}_i \setminus 0$. Here, $\mathbf{d}_{ij}(k|t) \in \mathcal{D}_N$ is the distance reference associated to the platoon formation $\bar{\mathcal{G}}_N$, and $|\mathcal{N}_i \setminus 0|$ is used to normalize the summation based on the number of neighboring vehicles. With these two terms, the platoon is able to track a formation $\bar{\mathcal{G}}_N$; nonetheless, since it is only tracking state differences, the formation can position itself at any place in the road or at any speed. To stick to an absolute reference, the third term is added, which propagates the states of a virtual leader, where $\mathbf{d}_{i0} \in \mathcal{D}$ plays the same role as the references in the second term; yet, this time referred to the virtual leader. This term is used to fix the platoon desired speed and the distance reference with respect to the road border. All the terms forming the cost function are weighted using positive semi-definite matrices $\mathbf{Q}_u, \mathbf{Q}_{\Delta u}, \mathbf{Q}_n$ and $\mathbf{Q}_0$, respectively.

Note that in the proposed formulation, vehicles should not track a global reference in the $x$ position of $\mathbf{d}_{i0}$. In case $\mathbf{Q}_0$ is not well defined, the DNMPC will track a global $x$ reference. To address this issue, the component of the diagonal matrix $\mathbf{Q}_0$ associated with the $x$ state is set to zero. As a consequence, vehicles only track $y$, $\psi$, and $v$ state references with respect to the virtual leader. Moreover, the steady-state velocity of the platoon and its lateral position are defined by these constraints in $\mathbf{d}_{0i}$.

Given the cost function (3.3), the Graph-based DNMPC is given by

$$\min_{\mathbf{u}_i(\cdot|t)} J_{2,i}(\mathbf{z}_i, \mathbf{u}_i) \tag{3.4}$$

$$\begin{aligned}
\text{s. t.} \quad & \text{(2.6b): } \mathbf{z}_i(k+1|t) = f\big(\mathbf{z}_i(k|t), \mathbf{u}_i(k|t)\big) \\
& \text{(2.6c): } \mathbf{z}_i(k=t|t) = \mathbf{z}_i(t) \\
& \text{(2.6d): } \mathbf{z}_i(k|t) \in \mathcal{Z}_i,\ \mathbf{u}_i(k|t) \in \mathcal{U}_i \\
& \text{(3.1a): } \big( - \mathbf{b}(\mathbf{z}_i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) - \mathbf{b}(\bar{\mathbf{z}}_j(k|t))^\top \bar{\boldsymbol{\lambda}}_{ji}(k|t)\big) \geq d_{min} \\
& \text{(3.1b): } \mathbf{A}(\mathbf{z}_i(k|t))^\top \bar{\boldsymbol{\lambda}}_{ij}(k|t) + \bar{\mathbf{s}}_{ij}(k|t) = 0, \\
& \quad \forall j \in \mathcal{N}_i
\end{aligned}$$

### 3.2.2. Graph-based Alternating Optimizations Scheme

The proposed graph-based scheme differs from the alternating optimization scheme presented in the previous section, yet, it still requires an external synchronization loop. In this case, the information communicated by each vehicle to the neighbors is different. First, the $\mathbf{A}$ and $\mathbf{b}$ sequences are communicated along with the state sequence obtained by the Graph-based DNMPC optimization to the neighbors. And secondly, the way how sequences are shifted to compensate for communication delays is changed, since when states are shifted as in Algorithm 1 ($[\mathbf{z}_i(t+1|t), \ldots, \mathbf{z}_i(t+N_c|t), \mathbf{z}_i(t+N_c|t)]$), the Graph-based DNMPC stabilizes the platoon at a speed value lower than the reference. This is a consequence of Algorithm 1 just copying the last state, which means that vehicles stop moving in just one time step. In the proposed graph-based formulation, this is solved

**Algorithm 2** Graph-based Alternating Optimizations Scheme

Initialize $[\mathbf{s}_{ij}(t), \ldots, \mathbf{s}_{ij}(t+N_c|t)]$
$[\boldsymbol{\lambda}_{ij}(t), \ldots, \boldsymbol{\lambda}_{ij}(t+N_c|t)]$
$[\boldsymbol{\lambda}_{ji}(t), \ldots, \boldsymbol{\lambda}_{ji}(t+N_c|t)]$
**for** $t = 0, 1, \ldots, \infty$ **do**
  **for all** vehicle $i$, $i \in \mathcal{V}$ **do in parallel**
    Solve Graph-based DNMPC Optimization Problem (3.4)
    Compute the shifted sequence $\mathbf{z}_i(\cdot|t) =$
    $[\mathbf{z}_i(t+1|t), \ldots, \mathbf{z}_i(t+N_c|t), f\big(\mathbf{z}_i(t+N_c|t), \mathbf{0}\big)]$
    Compute the polytopes of the sequence $\mathbf{A}\big(\mathbf{z}_i(\cdot|t)\big)$ and $\mathbf{b}\big(\mathbf{z}_i(\cdot|t)\big)$.
    Communicate the polytopes and vehicle state to agent $j, \forall j \in \mathcal{N}_i$
    Solve CA Optimization Problem (3.2)
    Apply $\mathbf{u}_i(t|t)$ obtained from (3.4) to move

by projecting the last state using the KBM, considering a zero-input, i.e., it is projected considering that vehicles keep a constant movement with respect to the last state. The proposed Graph-based Alternating Optimization Scheme is given in Algorithm 2.

The structure of graph-based planner makes platoons capable of tracking formations cooperatively using each vehicles' global states. Additionally, generating feasible and scalable references for complex maneuvers can be easily achieved by decoupling them into elemental longitudinal and lateral maneuvers. Nonetheless, the graph-based planner relies blindly on the information communicated by neighbors. It is of interest to formulate an approach that can make use of on-board sensors to estimate neighbors' states. This is addressed in the next section.

### 3.3. Incremental Graph-based Distributed Motion Planning

In this section, a robuster planner denominated the Incremental Graph-based Distributed Motion Planer is proposed, which exploits local proximity information using on-board sensors. This is achieved at each vehicle by re-framing the information received from neighbors with respect to the ego state. In practice, the re-framed information suppresses biases since it uses sensors that do not require integration of their measurements.

### 3.3.1. Incremental Graph-based Alternating Optimizations Scheme

The new optimization scheme uses the same structure as the Graph-based, since it is formulated in terms of the Graph-based DNMPC (Problem (3.4)) and CA (Problem (3.2)). Therefore, it also requires an external synchronization loop to operate correctly. Yet, it modifies the information communicated to neighbors and, incorporates the re-framing process. The modifications are described in the following.

Concerning the communicated information, vehicles only require to send the incremental state trajectory which is given by

$$\mathbf{z}_{i,0}(\cdot|t) = [\mathbf{z}_i(t+1|t), \ldots, \mathbf{z}_i(t+N_c|t), f(\mathbf{z}_i(t+N_c|t), \mathbf{0})] - \mathbf{z}_i(t+1|t) \tag{3.5}$$

where $\mathbf{z}_{i,0}(\cdot|t)$ refers to the trajectory of vehicle $i$ referred to the current state. The state sequence is obtained by the Graph-based DNMPC which is first projected with the KBM to account for communication delays, and then it is shifted.

To perform the neighbors' information re-framing, a state distance estimation $\mathbf{z}_{\Delta ij}(t)$ within vehicles $i$ and $j$ is used. $\mathbf{z}_{\Delta ij}(t)$ would give the notion of proximity respect to each neighbor and it is calculated at every time-step, avoiding problems like bias integration. In practice, the state distance estimation should be obtained using the local proximity information from on-board sensors like cameras and radars, among others.

When vehicle $i$ computes its Graph-based DNMPC and needs to track the relative state distance with respect to $j$, it uses $\mathbf{z}_{j,i}(\cdot|t)$, which corresponds to the state sequence of $j$ referred to $i$, given by

$$\mathbf{z}_{j,i}(\cdot|t) = \mathbf{z}_i(t) - \mathbf{z}_{\Delta ij}(t) + \mathbf{z}_{j,0}(\cdot|t) \tag{3.6}$$

Two points should be noted. First, $\mathbf{z}_{\Delta ij}(t)$ should be signed. And second, since the polytopes are not communicated anymore, these sequences must be calculated by each vehicle using the re-framed neighbor sequences $\mathbf{z}_{j,i}(\cdot|t)\ \forall j \in \mathcal{N}_i \backslash 0$. The resulting Incremental Graph-based Alternating optimization scheme is presented in Algorithm 3.

**Algorithm 3** Incremental Graph-based Alternating Optimization scheme

Initialize $[\mathbf{s}_{ij}(t), \ldots, \mathbf{s}_{ij}(t+N_c|t)]$
$[\boldsymbol{\lambda}_{ij}(t), \ldots, \boldsymbol{\lambda}_{ij}(t+N_c|t)]$
$[\boldsymbol{\lambda}_{ji}(t), \ldots, \boldsymbol{\lambda}_{ji}(t+N_c|t)]$
**for** $t = 0, 1, \ldots, \infty$ **do**
  **for all** vehicle $i$, $i \in \mathcal{V}$ **do in parallel**
    Estimate $\mathbf{z}_{\Delta ij}(t), \forall j \in \mathcal{N}_i$
    Calculate the re-framed sequence $\mathbf{z}_{j,i}(\cdot|t)$ and their polytopes $\mathbf{A}\big(\mathbf{z}_{j,i}(\cdot|t)\big)$ and $\mathbf{b}\big(\mathbf{z}_{j,i}(\cdot|t)\big)$, $\forall j \in \mathcal{N}_i$
    Solve CA Optimization Problem (3.2)
    Solve Graph-based DNMPC Optimization Problem (3.4)
    Compute the origin-referred state trajectory $\mathbf{z}_{i,0}(\cdot|t)$
    Communicate the origin-referred state trajectory $\mathbf{z}_{i,0}(\cdot|t)$ to agent $j, \forall j \in \mathcal{N}_i$
    Apply $\mathbf{u}_i(t|t)$ obtained from (3.4) to move

### 3.4. Reference Generation

Reference generation plays a fundamental role in the operation of the vanilla and graph-based planners to test their performance. Despite the planners use DNMPC to track a smooth trajectory from the reference, some references may end up in deadlock scenarios which may be unsolvable by the DNMPC due to finite horizon reference tracking. Therefore, references should be generated from a feasible set. In (Firoozi et al., 2020), reference generation was left as an open problem. In this work, reference generation is addressed based on hand-crafted rules and heuristics based on the platoon characteristic, the road structure and the desired speed.

A fair comparison between the vanilla distributed planners and the graph-based planners is necessary in order to achieve unbiased results about the performance of the planners. In order to achieve this, the graph-based reference were generated first and from them, the references for each individual vehicle used by the vanilla distributed planner were generated.

The design was generated for an open-loop controlled and known environment in a centralized fashion, which was sufficient to test the performance of the planners. Vehicles were considered point agents and bumper to bumper distances within vehicles are not

considered, but because the references were based on heuristic, they were designed with enough distances to avoid infeasible references.

### 3.4.1. Graph Reference Generation

The process of designing references is based on hand-crafted instructions as a function of the desired maneuver, the speed and time to perform it. The instructions consists on canonical movements and they are composed of four parameters: the type of movement (*longitudinal* or *lateral*); the time step $T_{vehicleid}$ in which the maneuver should start; the time window $\Delta T_{vehicleid}$ in which the maneuver should be performed; and the desired (signed) distance $\Delta D_{vehicleid}$ to move. This method allows decoupling complex platoon maneuvers as single-vehicle instructions. For example, a maneuver like a lane change is decoupled into longitudinal and lateral maneuvers, where initially vehicles perform the longitudinal instructions to make enough longitudinal gap, followed by the lateral instructions for the vehicles lane changing. Additionally, this method allows each vehicle to perform a different maneuver respect to the others. A set of instructions for three different vehicles would have the following structure.

$$
\begin{aligned}
&instructions(id1) = () \\
&instructions(id2) = ( \\
&\qquad\qquad (\prime Longitudinal\prime, TLo_{id2}, \Delta TLo_{id2}, \Delta DLo_{id2}) \\
&\qquad\qquad ) \\
&instructions(id3) = ( \\
&\qquad\qquad (\prime Longitudinal\prime, TLo_{id3}, \Delta TLo_{id3}, \Delta DLo_{id3}) \\
&\qquad\qquad (\prime Lateral\prime, TLa_{id3}, \Delta TLa_{id3}, \Delta DLa_{id3}) \\
&\qquad\qquad )
\end{aligned}
$$

where the first vehicle would not perform any instructions, the second one will perform a longitudinal displacement and the third one will perform longitudinal and a lateral displacements.

After the instructions are set for each vehicle, the simulation begins. Every one second, the instructions are checked for each vehicle and executed if they are in the time window to perform it. Since the algorithm operates in a centralized way, the instructions are executed by shifting the relative position of each vehicle inside the platoon, and then, the graph-based reference is calculated based on the relative positions of the vehicles. A detailed version of the procedure is presented in Algorithm 4. Ideally, instructions should be generated in a distributed fashion and updated in close-loop. Nonetheless, the method used had a suitable compromise between the capability of testing the planners' performance and the reference generation complexity.

Finally, the graph-based planners receive as input a sequence of distance references. Therefore, a final step is performed to calculate a smooth sequence of distance references, which are generated by linearly interpolating the current and the next graph, this procedure is described in Appendix A. Interpolating the graphs induces a smoother response.

### 3.4.2. Trajectory Reference Generation

Concerning the reference generation for the vanilla distributed planner, it was generated from the $positions$ used to obtain the graphs in order to perform the same maneuvers at the exact time steps as the graph-based references. In order to do so, the reference was generated before the simulation starts and each state was updated considering the sample time. The longitudinal state $x$ was obtained by integrating the velocity and adding the longitudinal variations in longitudinal position. The lateral state $y$ was updated only based on the lateral variations in the position. While the heading angle $\phi$ and the speed reference $v$ were kept constant. The details of the trajectory reference generation are presented in Algorithm 5.

**Algorithm 4** Graph-based Reference Generation

**for** $t = 0, 1, \ldots, \infty$ **do**
  **for** $id = 1, ..., N_{vehicles}$ **do**
    $inst = instructions(id)$
    # Check if vehicle id has instructions
    **if** isnotempty($inst$) **then**
      # Evaluate all possible instructions
      **for** $j = 1, ..., size(inst)$ **do**
        $type = inst(j, 1)$
        $T0 = inst(j, 2)$
        $\Delta t = inst(j, 3)$
        $\Delta d = inst(j, 4)$
        # Check if there is an instruction to execute in the current time step.
        **if** $T0 \leq t \leq T0 + \Delta t$ **then**
          # Check instruction type
          **if** $type ==' Longitudinal'$ **then**
            $positions(id) = positions(id) + [\Delta L, 0, 0, 0]^\top$
          **else**
            $positions(id) = positions(id) + [0, \Delta L, 0, 0]^\top$
  graph = []
  **for** $i = 1, ..., N_{vehicles}$ **do**
    **for** $j = 1, ..., N_{vehicles}$ **do**
      graph($i$, $j$) = $positions(i) - positions(j)$

**Algorithm 5** Trajectory Reference Generation

# Initialize states and references.
**for** $id = 1, ..., N_{vehicles}$ **do**
  $x(id) = x_0(id)$
  $y(id) = y_0(id)$
  $v(id) = v_{max}$
  $\phi(id) = 0$
  $ref(0)(id) = [x(id), y(id), v(id), \phi(id)]^\top$
# Calculate the trajectory of the experiment.
**for** $t = 0, 1, \ldots, \infty$ **do**
  # Calculate the current positions $pos(t)$ and the desired positions $pos(t + 1)$ with Algorithm 4.
  **for** $id = 1, ..., N_{vehicles}$ **do**
    $x(id) = x(id) + v_{max} * \Delta t + \frac{pos(t+1)(id,1) - pos(t)(id,1)}{\Delta t}$
    $y(id) = y(id) + \frac{pos(t+1)(id,2) - pos(t)(id,2)}{\Delta t}$
    $v(id) = v_{max}$
    $\phi(id) = 0$
    $ref(t + 1)(id) = [x(id), y(id), v(id), \phi(id)]^\top$

# 4. NUMERICAL EVALUATION

## 4.1. Scenarios

Four scenarios are setup to compare and analyze the performance of the three distributed planners. The scenarios consider possible situations in highways and the complexity progressively increases.

In each scenario, vehicles are modeled using the KBM in MATLAB, and the optimization problems are solved using YALMIP (Löfberg, 2004) in the same environment. DNMPC optimizations are solved using IPOPT (Wächter & Biegler, 2006), an interior-point solver for non-convex optimization, and CA is solved using Gurobi (Gurobi Optimization, 2021), a QP solver. Similar parameters to the ones defined in (Firoozi et al., 2020) and (Firoozi et al., 2021) are used. The controller's planning horizon is $0.75s$ using a time step of $\Delta t = 0.05s$ for the first two scenarios, while in the last two scenarios are $1s$ and $\Delta t = 0.2s$, respectively. CA minimum distance is $d_{min} = 0.3m$. Vehicles are considered homogeneous with dimensions $h = 4.5m$ and $w = 1.8m$. Vehicles' actuators are bounded by $a = \pm 4m/s^2$ and $\delta = \pm 1rad$, while their rates of change are bounded by $\pm 1m/s^2/s$ and $\pm 1rad/s$, respectively. Concerning the vehicles' states, these are modeled by the feasible set $\mathcal{Z}$ presented in (2.6d), which can be interpreted as the intersection of the vehicle limitations and the road constraints. In $\mathcal{Z}$, $y$ is bounded by the road borders, where each lane width is $3.7m$ (US standard for highways), while the speed is lower-bounded by zero and upper-bounded by $v_{max} = 19m/s$. The weight matrices of cost functions $J_{1,i}(\mathbf{z}_i, \mathbf{u}_i)$ and $J_{2,i}(\mathbf{z}_i, \mathbf{u}_i)$ are diagonal matrices and they are detailed in Appendices B.1 and B.2, respectively.

For each planner in each scenario, the resulting states and optimal inputs are shown in a single figure. Additionally, the longitudinal, or $x$ states, plots are depicted relative to a moving reference at the target platoon speed, which facilitates the comparison of the distance within vehicles and helps to show if the platoon is gaining or losing longitudinal distance compared to the moving at constant speed case.

### 4.1.1. Ideal Lane-change

The first scenario consists of a lane-change in a two-vehicle platoon. In this scenario, two cases are considered, one where the lane-changing vehicle is ahead enough to perform the maneuver without colliding, and a second one where both vehicles are travelling in parallel. Meanwhile in the first case the lane-change maneuver is direct, in the second case, vehicles need to create the necessary space, which is accomplished through the reference.

### 4.1.2. Platooning with Uncertainty in the States

The second scenario consists of a single-lane two-vehicle platoon with non-ideal state knowledge. Two cases are considered, in order to evaluate the influence of an additive zero-mean multivariate normal distributed noise and a bias. In the first case, both vehicles have state uncertainty modeled as additive zero mean noise with diagonal covariance given by $[0.01, 0.01, 0, 0.05]$, while in the second case, the rear vehicle additionally has a bias of $[-0.02, 0, 0, 0]$ in its state.

### 4.1.3. Internal Maneuvers in a Multi-lane Platoon

The third scenario consists of a ten-vehicle platoon performing internal maneuvers such that two vehicles exchange lanes. Uncertainties in the states are considered, with the same additive zero-mean multivariate normal distributed noise used in the previous scenario; moreover, actuator disturbances are included, modeled as a zero-mean multivariate noise with diagonal covariance matrix given by $[0.05, 0.0001]$. Vehicles are initialized in random positions; therefore, the first maneuver is a rearrange of the platoon into a steady state formation. To perform the lane-change, first vehicles rearrange longitudinally to make a sufficient gap, then, two vehicles perform the lateral movement and finally, vehicles rearrange longitudinally into the steady state formation again. The most relevant snapshots of the formation* are depicted in Fig. 4.1.

*Note that these Graph-based planners, references are referred to each vehicle CG's, nonetheless, they can be effectively re-framed to the vehicles' borders.

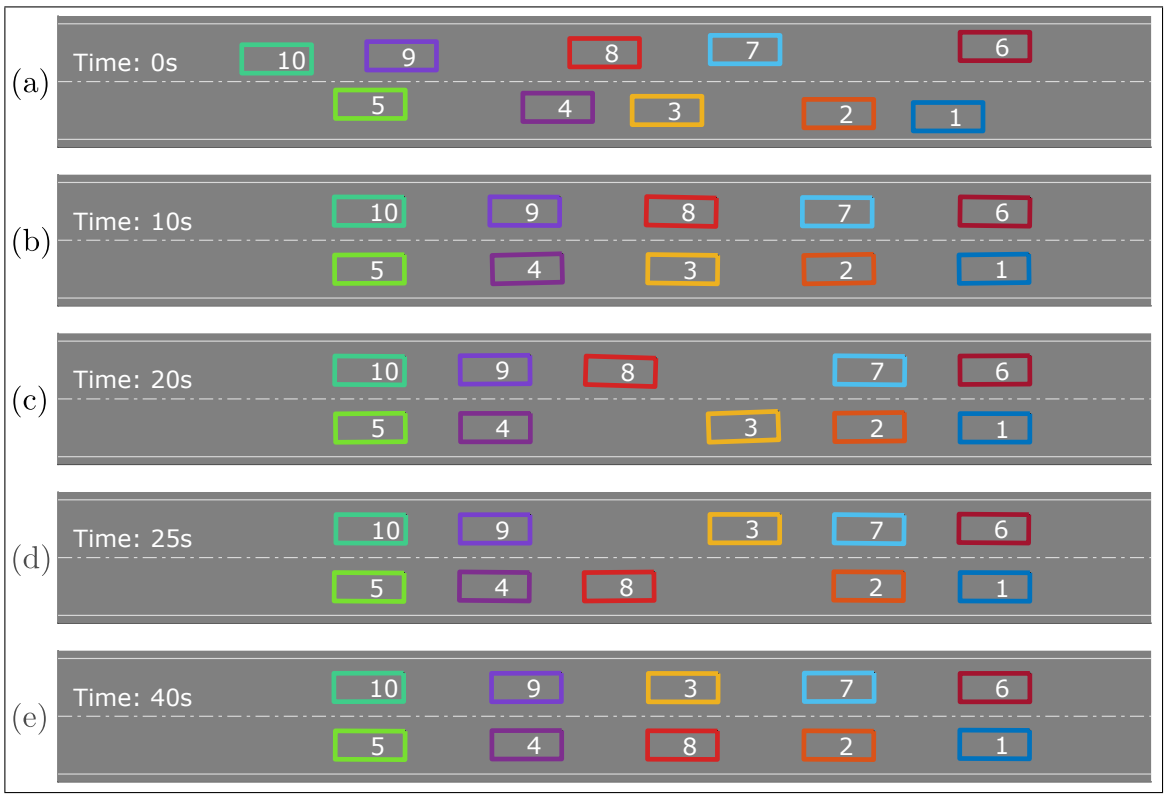


Figure 4.1. Scenario 3: Platoon formation for ten vehicles. The maneuvers involve vehicles 3 and 8 exchanging lanes.

Note that in this platoon, vehicles communicate with their surrounding neighbors in Fig. 4.1.b, which means that all-to-all communication is not used.

### 4.1.4. Merging Lanes

The last scenario considers a thirty-six-vehicle three-lane platoon that, as travels, has to rearrange into a two-lane platoon due to merging lanes. The scenario is simulated with the same parameters and non-idealities of the previous scenario and it also considers communication only with the surrounding neighbors in the platoon. In the experiment, the lane-merge maneuver was hand-designed such that groups of six vehicles are merging just in time and no shockwaves are created in the traffic flow. To perform the maneuver, each group of vehicles have $15s$ and the lanes finish merging in $T$ for the $k$ group. Therefore, let's suppose that the first vehicle $k+1$ of group $k$ inside the platoon reaches $T-15$, at

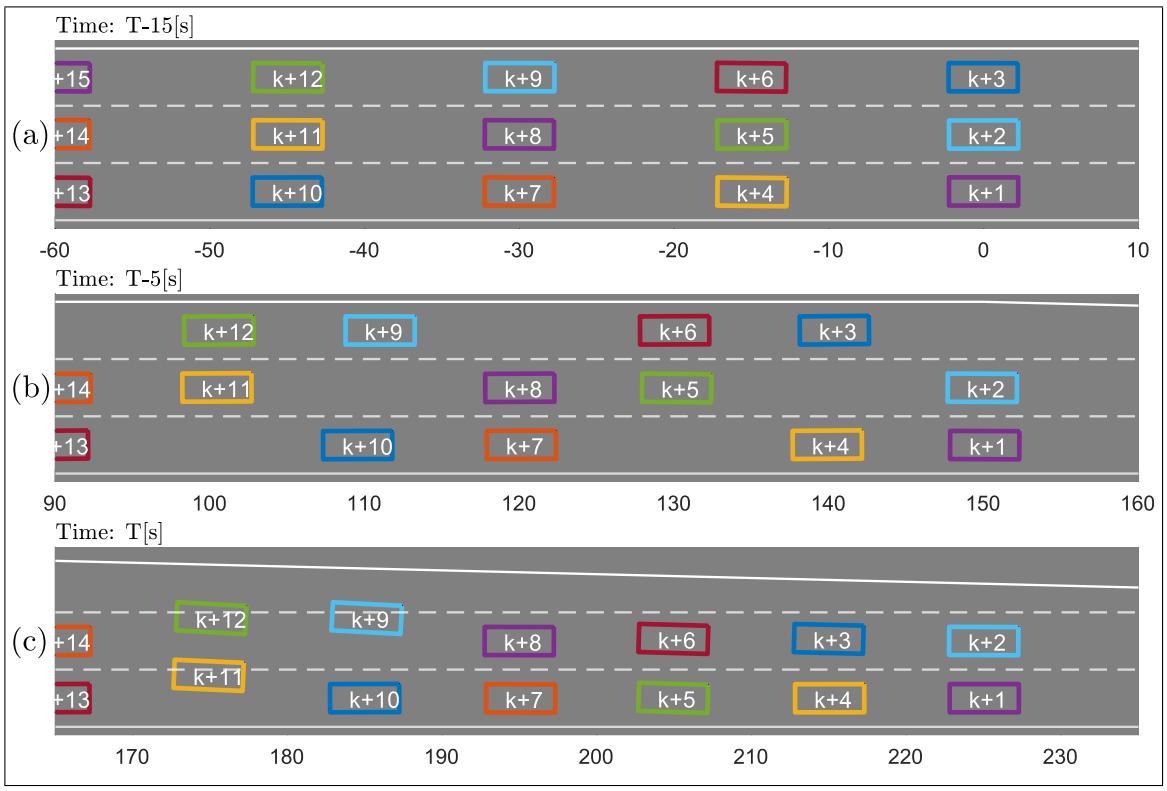


Figure 4.2. Scenario 4: Lane merge formations of a six-vehicle group inside the platoon.

this point, the group is in a three-lane steady state formation (Fig. 4.2.a). At this time, the group begins performing a longitudinal rearrange during the next 10 seconds (4.2.b) to make enough space for the lane-merge. At $T-5$, the group should be completely rearranged to perform the lane merge before $T$ (Fig. 4.2.c). Three points should be noted here, firstly, vehicles $k+1$ and $k+2$ do not move relative to the moves of the vehicle of the next group $k+7$ and $k+8$, which means that each group is performing and internal maneuver like in scenario 3. Secondly, due to the fact that groups are moving, each groups' maneuvers are overlapping in order to perform them in the desired time. Finally, the six-vehicle group structure was just used to design the formation graph references; nevertheless, vehicles are still in a thirty-six vehicle platoon tracking their states respect to their close neighbors, meaning that there are no restriction of vehicles communicating across the six-vehicle groups.

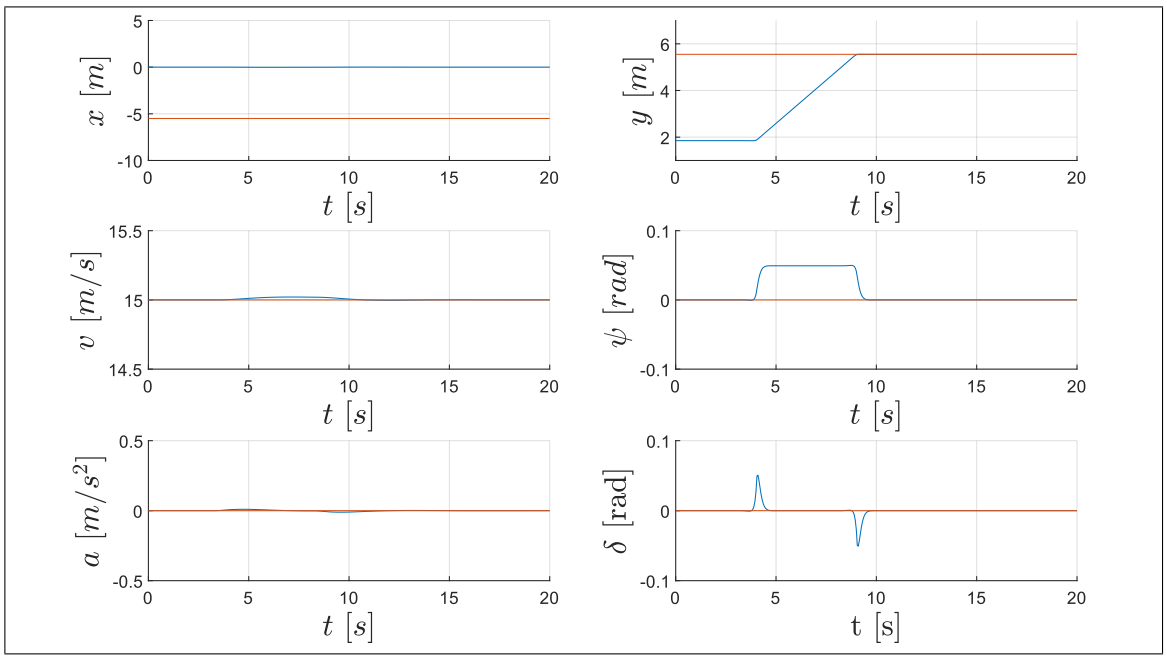


Figure 4.3. Results of scenario 1, lane change with enough longitudinal gap, for the Distributed Motion Planner.

## 4.2. Evaluation Results for Scenario 1

The results for scenario 1, with sufficient longitudinal gap within vehicles (case 1), for the Distributed Motion Planner, the Graph-based Distributed Motion Planner and the Incremental Graph-based Distributed Motion Planner are presented in Figs. 4.3, 4.4 and 4.5, respectively. It can be seen that when there is only lateral movements with a feasible reference (since there is enough longitudinal gap), the three planners perform almost identically, which is expected.

The results for the case where vehicles are travelling in parallel (case 2) are presented Figs. 4.6, 4.7 and 4.8. In this case, graph-based planners perform identically between them, with one vehicle accelerating and the other decelerating to accomplish the longitudinal relative distance within them. In contrast, in the vanilla distributed planner, only one vehicle accelerates while the other keeps tracking its constant reference.

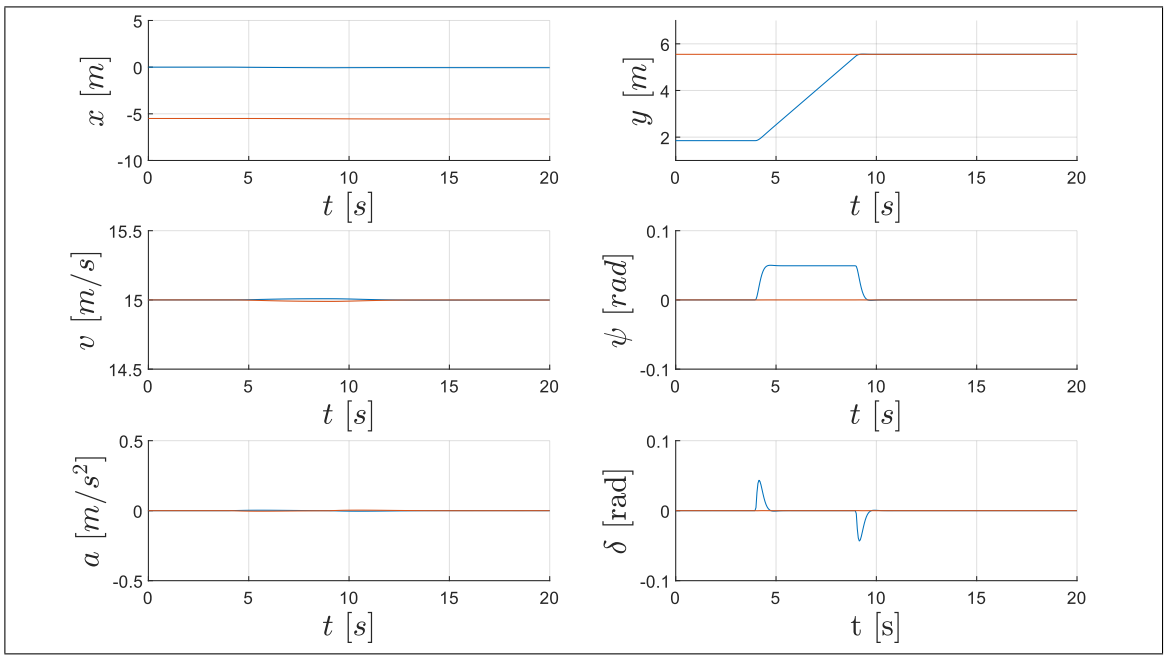


Figure 4.4. Results of scenario 1, lane change with enough longitudinal gap, for the Graph-based Distributed Motion Planner.

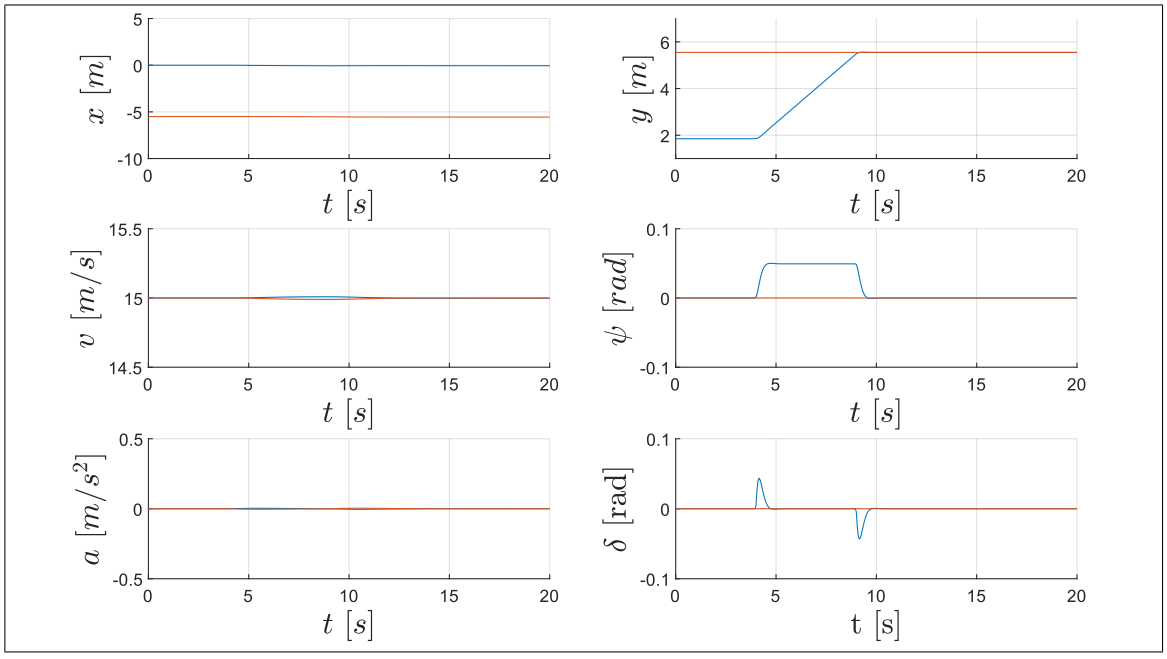


Figure 4.5. Results of scenario 1, lane change with enough longitudinal gap, for the Incremental Graph-based Distributed Motion Planner.

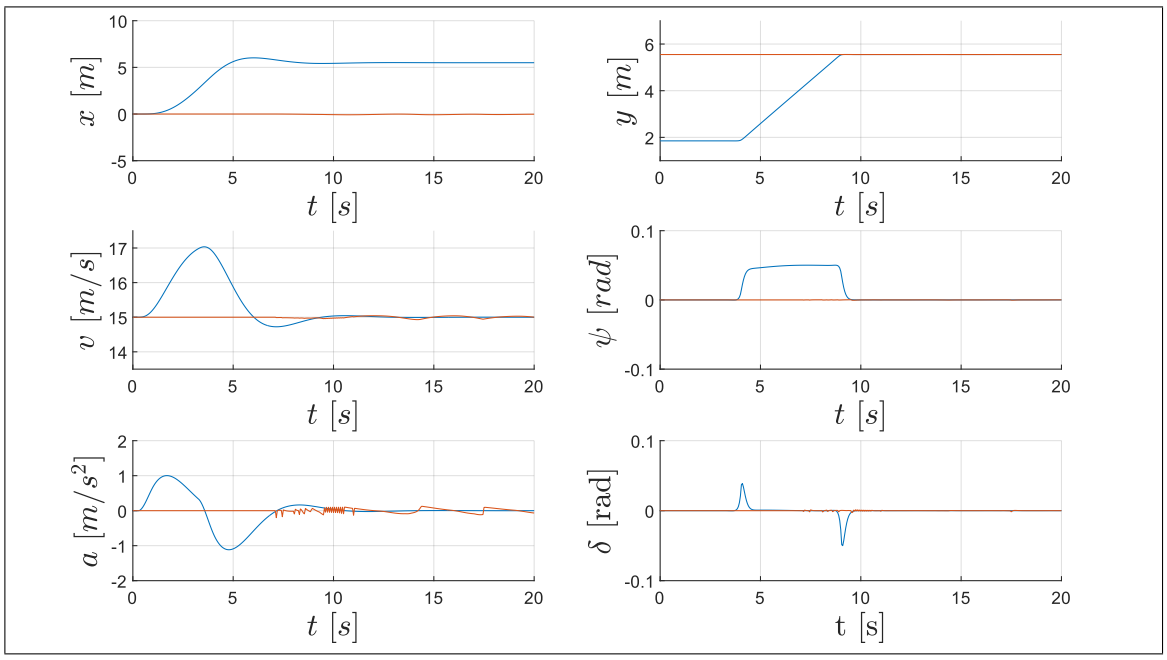


Figure 4.6. Results of scenario 1, lane change with vehicles in parallel, for the Distributed Motion Planner.

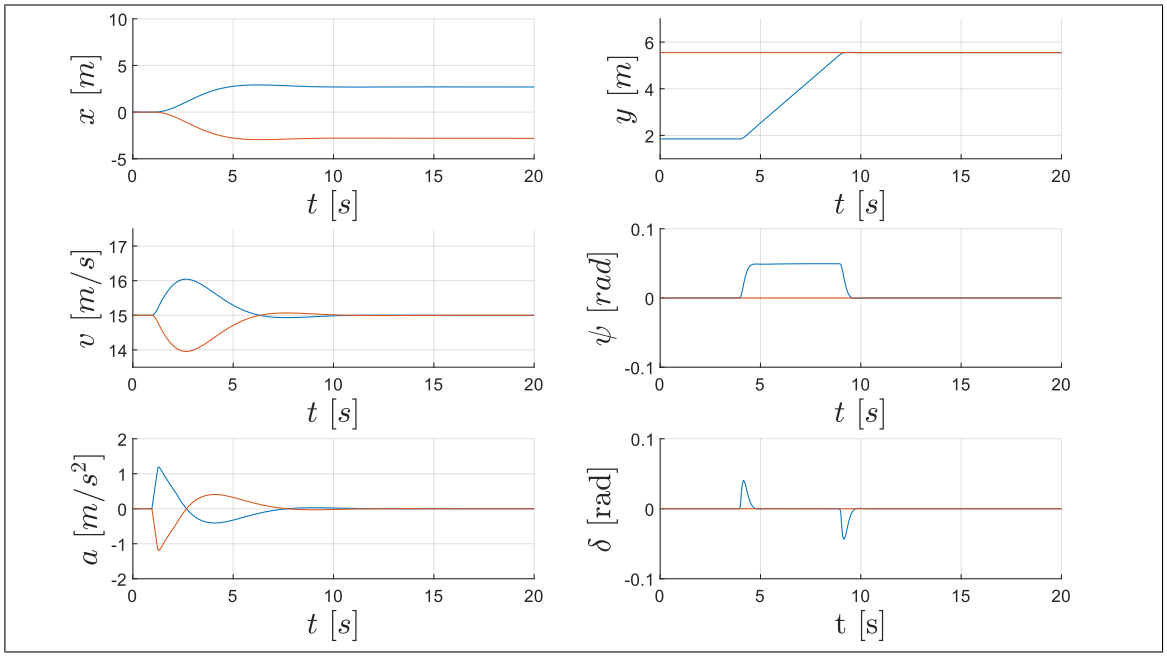


Figure 4.7. Results of scenario 1, lane change with vehicles in parallel, for the Graph-based Distributed Motion Planner.

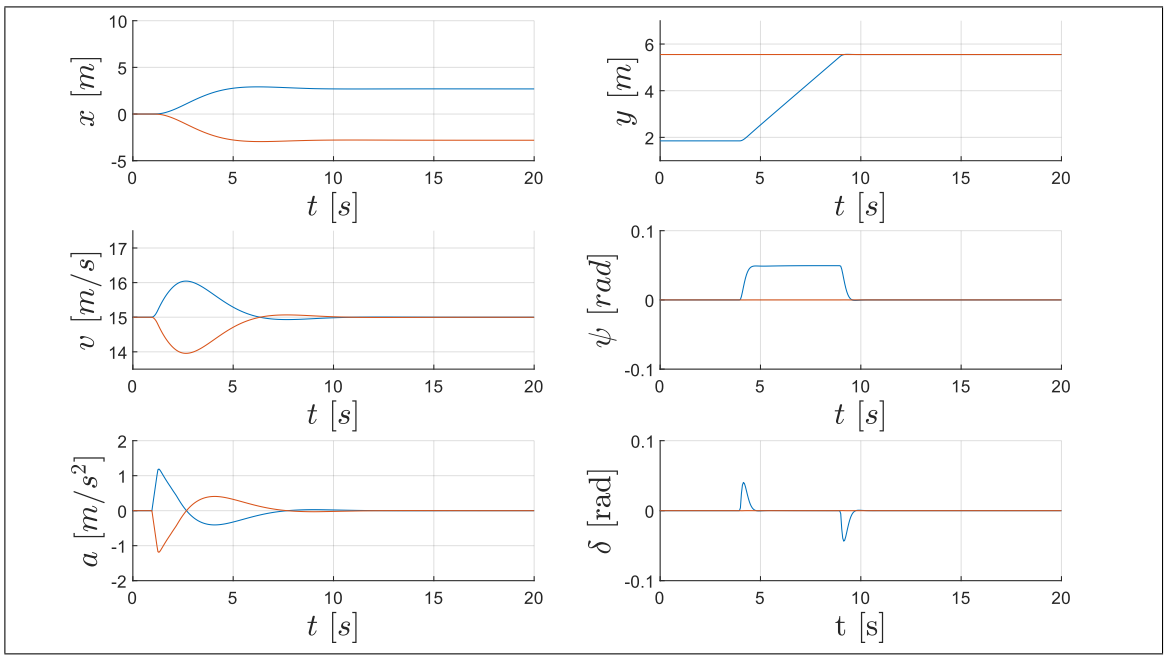


Figure 4.8. Results of scenario 1, lane change with vehicles in parallel, for the Incremental Graph-based Distributed Motion Planner.

### 4.3. Evaluation Results for Scenario 2

The results for scenario 2, with zero-mean multivariate normal distributed noise (case 1), for the planners are presented in Figs. 4.9, 4.10 and 4.11. Overall, measurement noise does not destabilize the platoon, although noise is observed in states and manipulated variables.

The results for scenario 2, when the rear vehicle has a state bias in the longitudinal position (case 2) are presented in Figs. 4.12, 4.13 and 4.14. It can be seen that only the Incremental Graph-based Motion planner is capable of accurately tracking the relative longitudinal distances. Moreover, even though CA constraints are explicitly formulated, the other two planners cannot avoid a collision, which is depicted by the dashed red line.

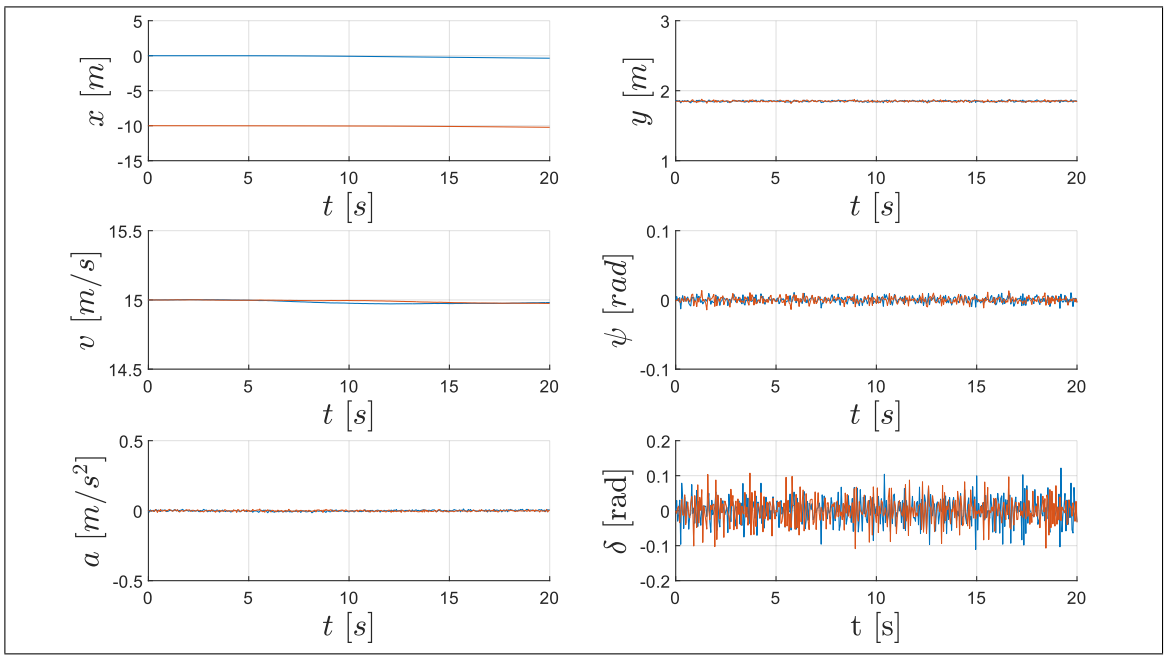


Figure 4.9. Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Distributed Motion Planner.

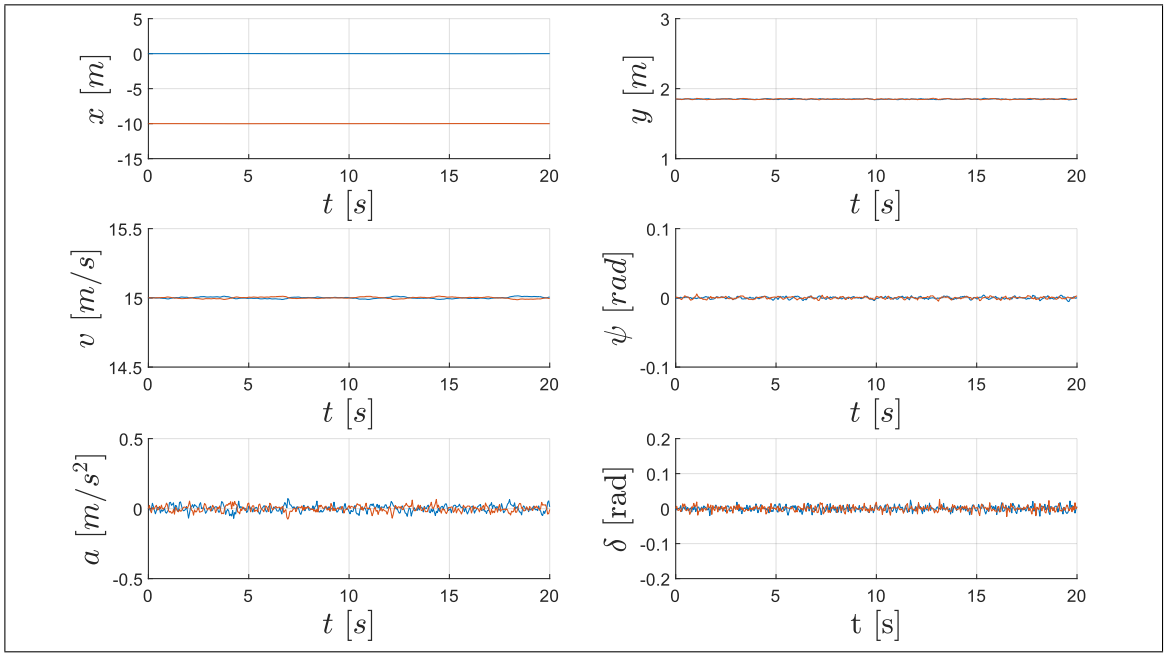


Figure 4.10. Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Graph-based Distributed Motion Planner.

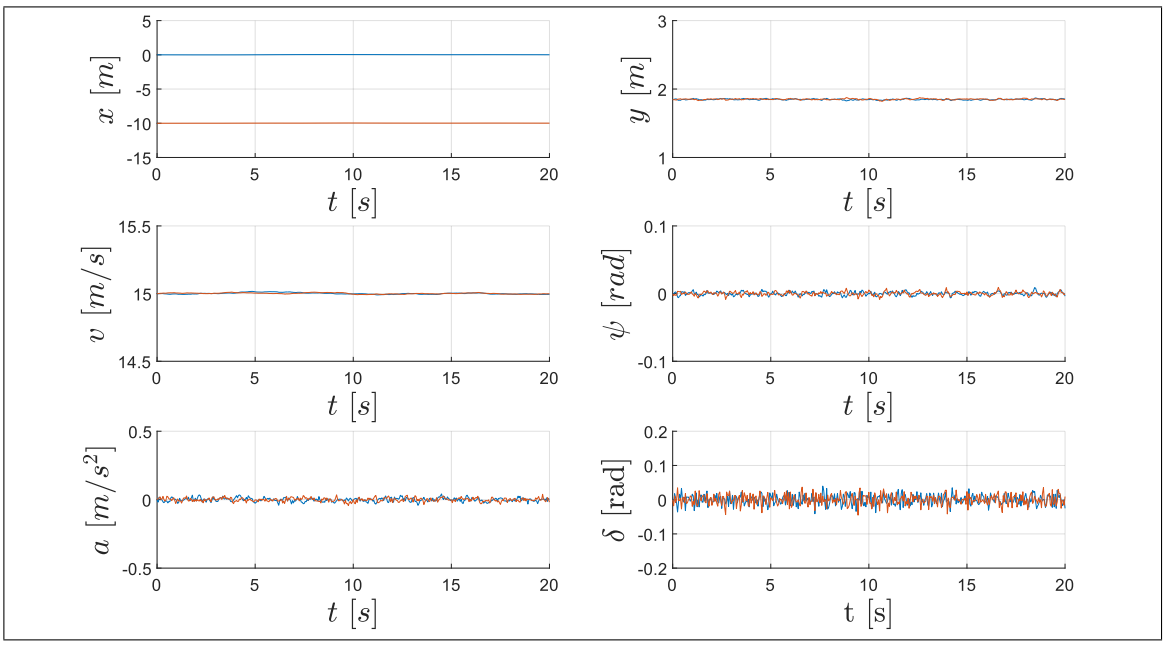


Figure 4.11. Results of scenario 2, state with additive zero-mean multivariate normal distributed noise, for the Incremental Graph-based Distributed Motion Planner.

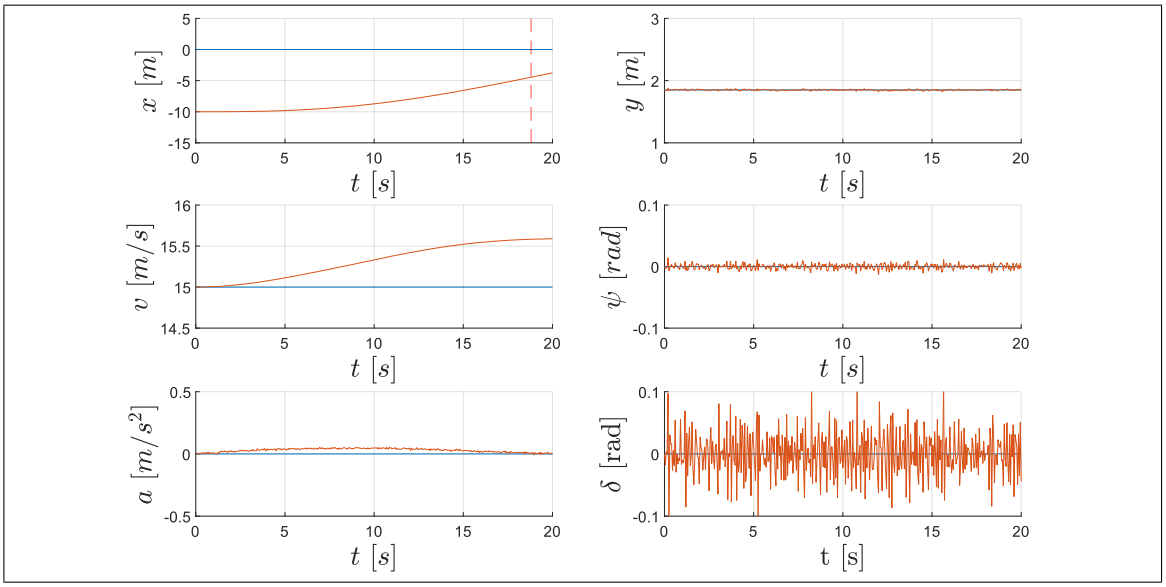


Figure 4.12. Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Distributed Motion Planner. The dashed red line denotes a collision.

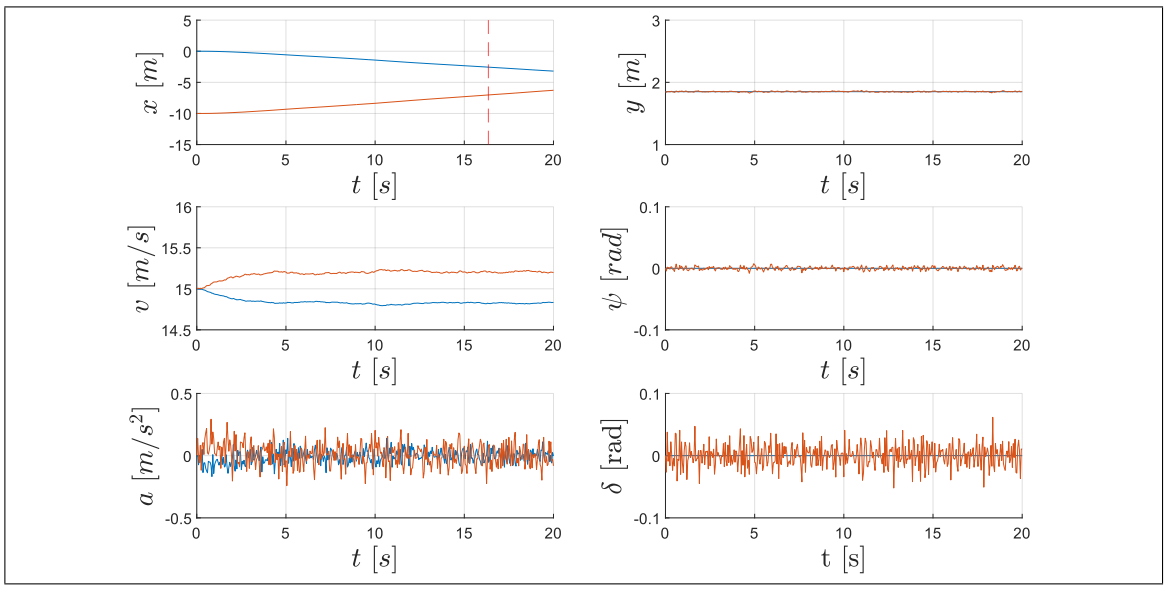


Figure 4.13. Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Graph-based Distributed Motion Planner. The dashed red line denotes a collision.

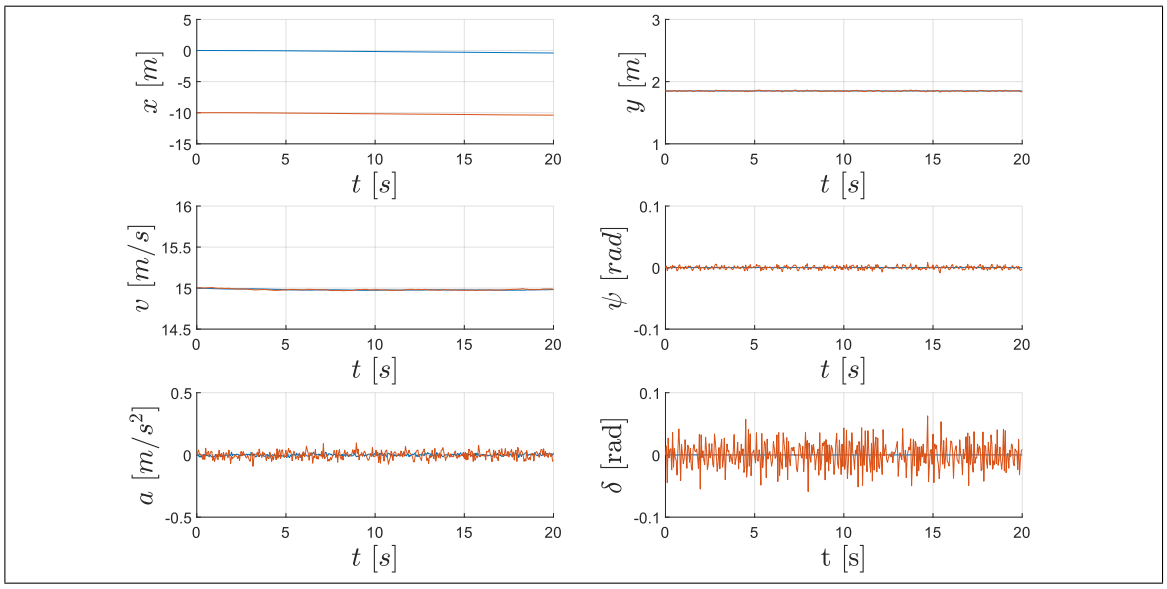


Figure 4.14. Results of scenario 2, state with additive biased multivariate normal distributed noise, for the Incremental Graph-based Distributed Motion Planner.

## 4.4. Evaluation Results for Scenario 3

The results for scenario 3 are presented in Figs. 4.15, 4.16 and 4.17. In this experiment, it can be seen that the three planners are capable of rearranging vehicles internally to perform a lane-change inside the platoon. Furthermore, the vehicles are not gaining or losing longitudinal distance.

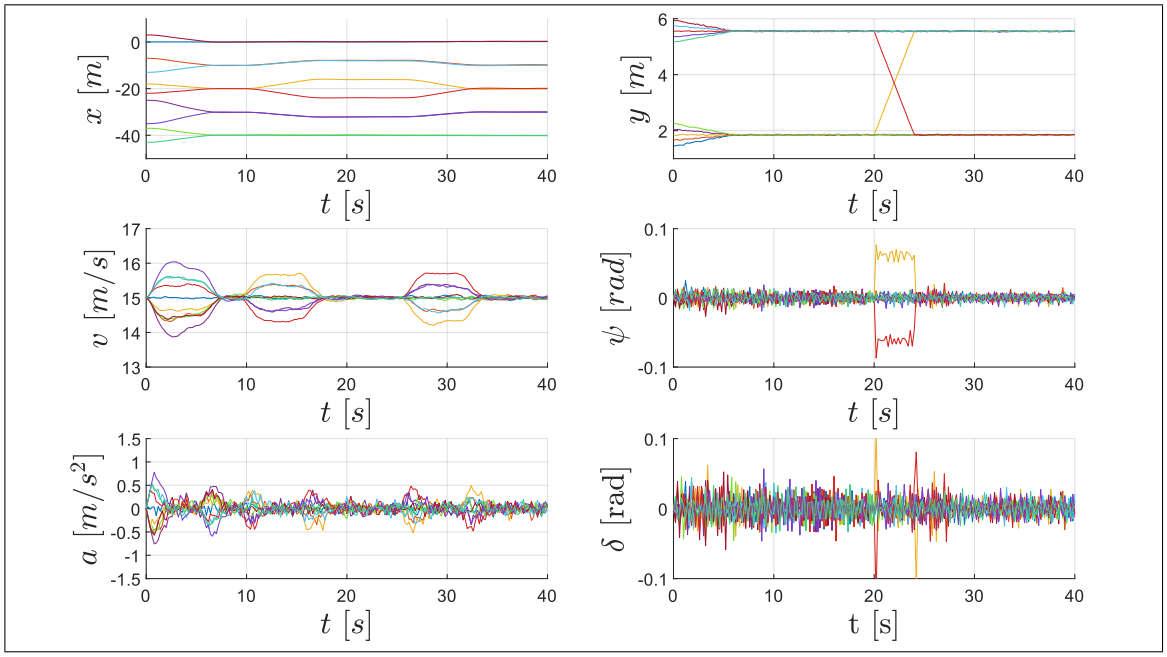


Figure 4.15. Results of scenario 3, internal maneuvers, for the Distributed Motion Planner.

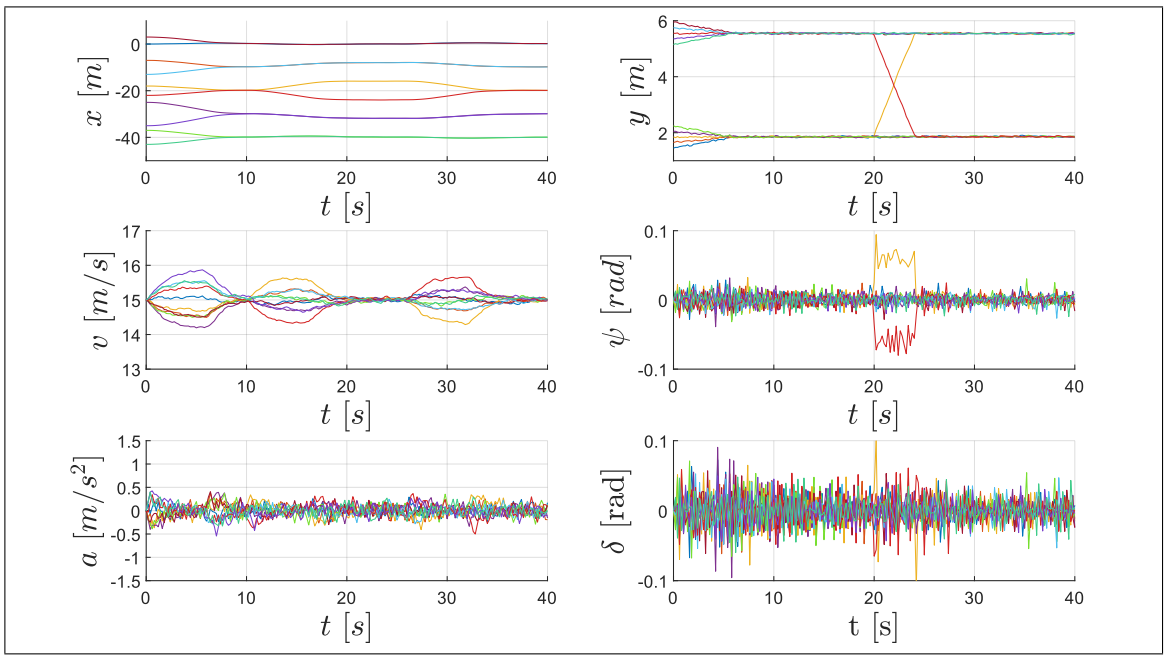


Figure 4.16. Results of scenario 3, internal maneuvers, for the Graph-based Distributed Motion Planner.

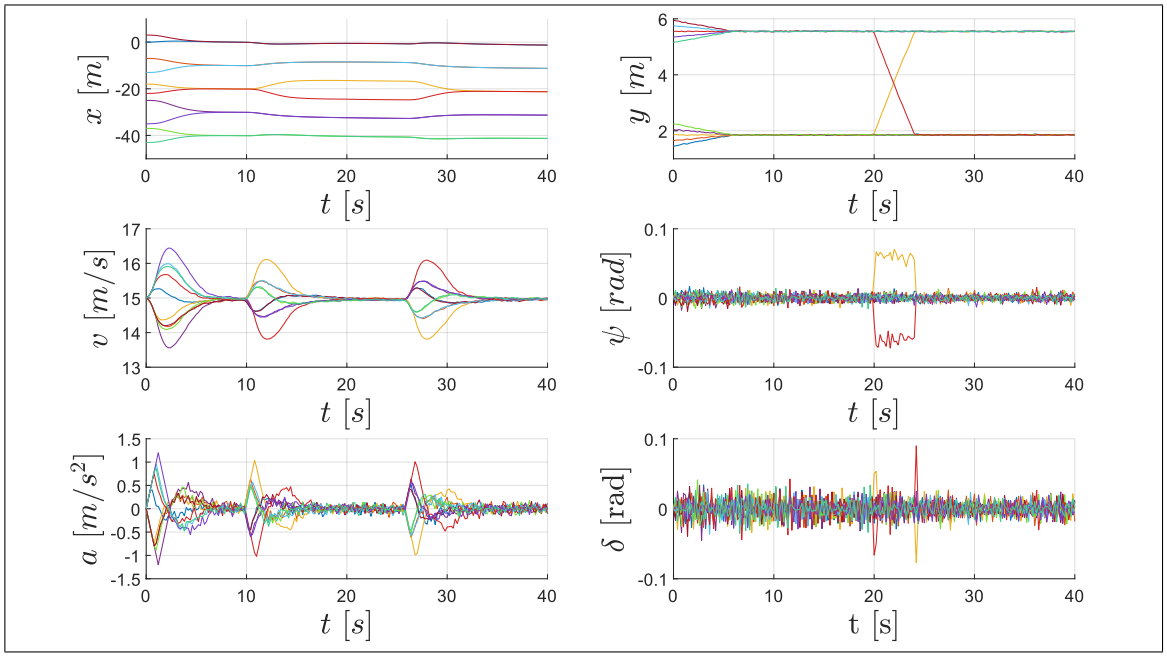


Figure 4.17. Results of scenario 3, internal maneuvers, for the Incremental Graph-based Distributed Motion Planner.

## 4.5. Evaluation Results for Scenario 4

The results for scenario 4 are presented in Fig. 4.18, 4.19 and 4.20. It can be seen that the thirty-six vehicle platoon can accurately perform the lane merging, without gaining or losing longitudinal distance. Moreover, vehicles inside the platoon do not need all-to-all communication. And finally, meanwhile the graph-based references are feasible, there is no restriction to have vehicles performing longitudinal maneuvers while others perform lateral ones.

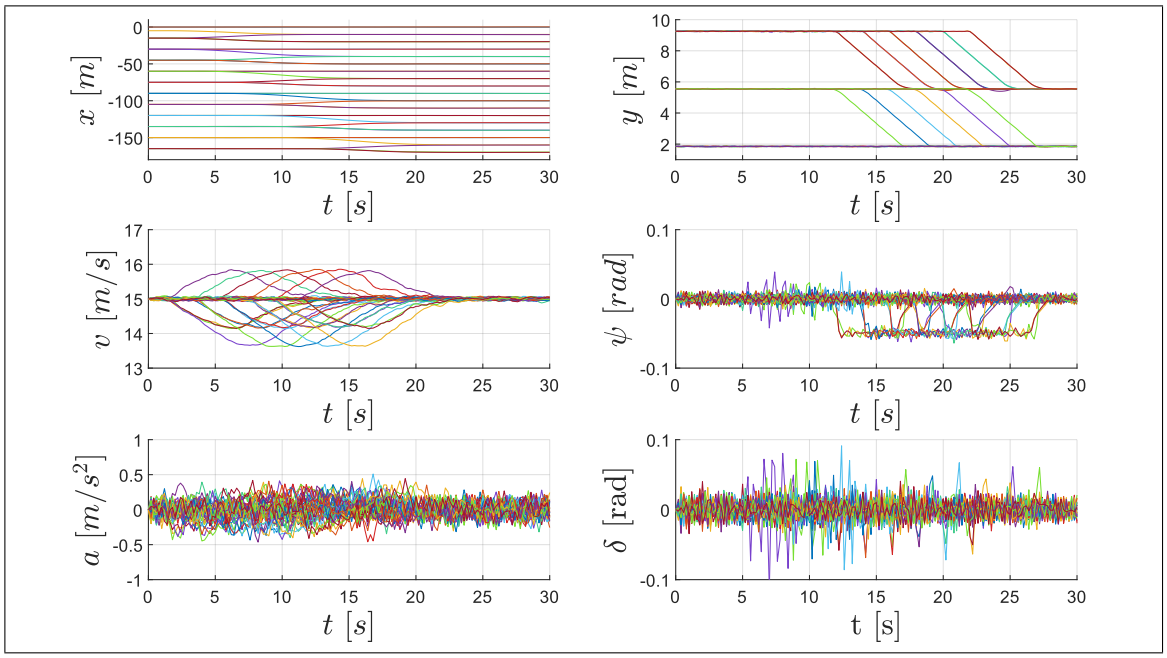


Figure 4.18. Results of scenario 4, merging lanes, for the Distributed Motion Planner.

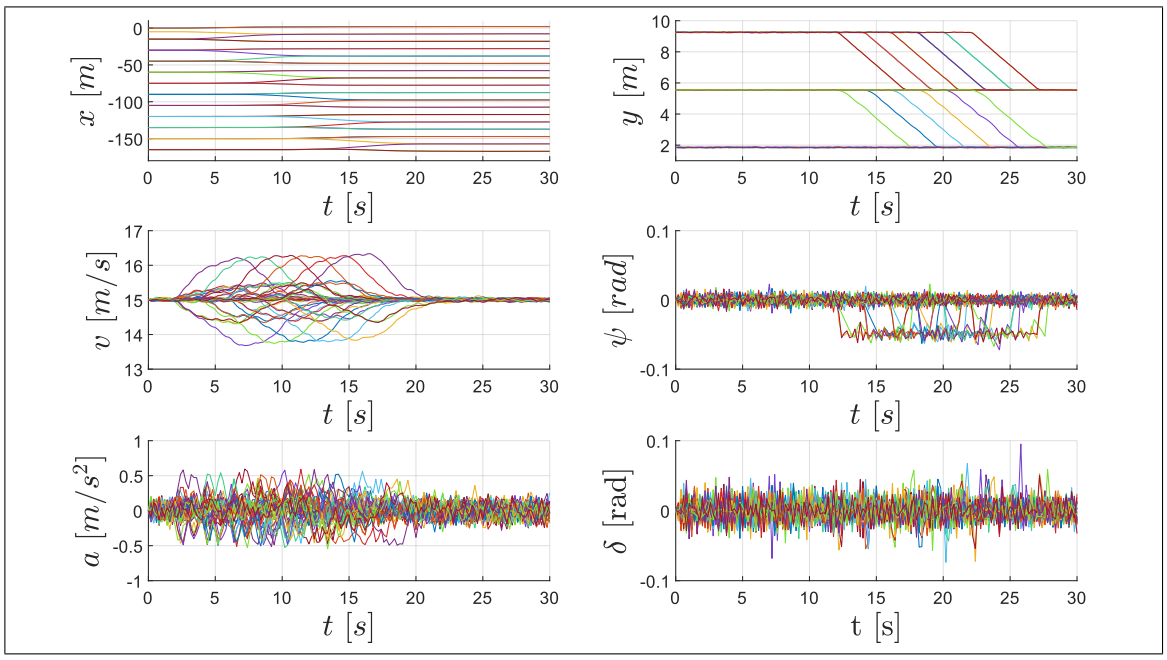


Figure 4.19. Results of scenario 4, merging lanes, for the Graph-based Distributed Motion Planner.

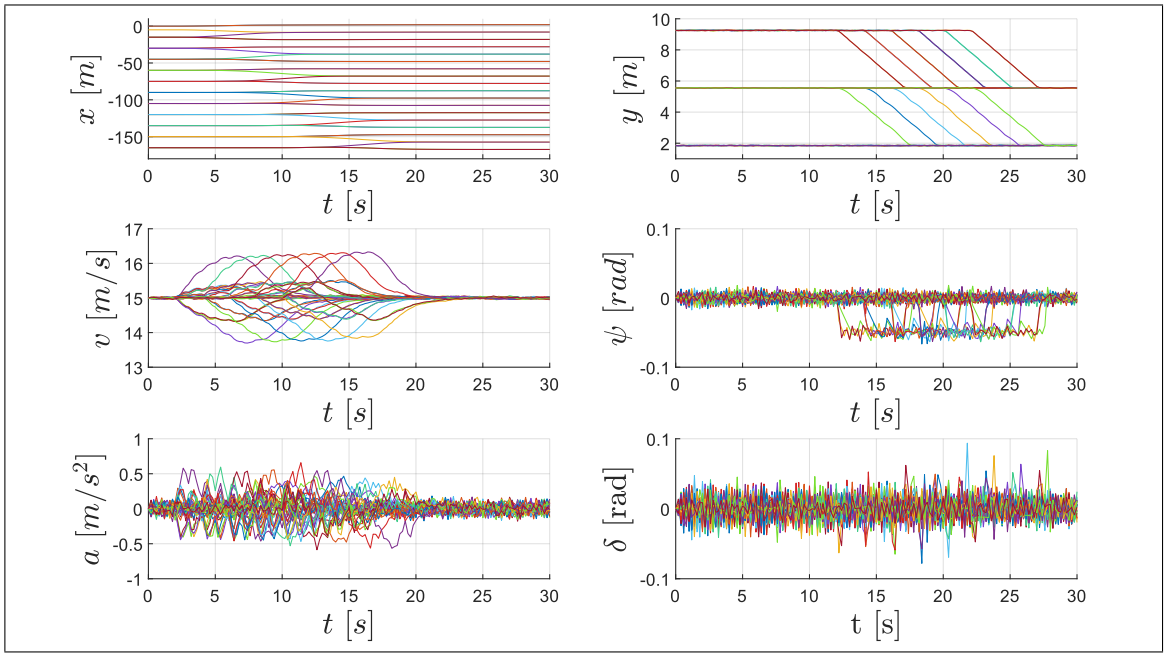


Figure 4.20. Results of scenario 4, merging lanes, for the Incremental Graph-based Distributed Motion Planner.

## 4.6. Discussion

The results presented show how the planners are capable of accomplishing almost all tasks in the different scenarios. Yet, there are some differences in their behavior and performance that need to be discussed.

In the first scenario, case 2, where longitudinal space to perform the lane change maneuver is not available, initially, vehicles need to create that gap. It can be seen in the results that in the graph-based solutions vehicles collaborate to accomplish the task, while in the vanilla distributed planner, only one vehicle is moving to make the gap. The reason behind this is that in the cost function of the graph-based planners, (3.3), both vehicles track the relative distance within them, while in the vanilla planner, each vehicle tracks its own global reference, which means that vehicles are not fully aware of how close they are to their neighbors. This does not imply that the vanilla distributed planner is not capable of achieving a cooperative-like behavior; yet, this must be explicitly hand-coded in the global reference. Therefore, graph-based planners show an advantage over the vanilla distributed planner since collaboration is intrinsic, which promotes the use of smoother actions to accomplish the same tasks.

Concerning scenario 2, both cases show important differences between the planners. In case 1, it must be noted that, due to the uncertainty in the state, the steering angle presents noise, particularly for the vanilla distributed planner, which shows a steering angle comparable with a lane change maneuver and much larger than graph-based planners. Despite the fact that MPC's weight matrices and cost functions are different, the main reason for this is that graph-based planners track more than one reference: neighbors and the road border, which naturally operates as disturbance rejection mechanism. Case 2 is also very relevant, since it shows that only the Incremental Graph-based Distributed Motion Planner is capable of handling biased states, by using local sensors that estimate the neighbors' states. To clearly illustrate the reason behind the malfunction of the other planners, suppose that vehicle $i$ has a state bias, analyzing (3.1) we notice that the optimization uses the

state-dependent polytopes $A(\mathbf{z}_i)$ and $b(\mathbf{z}_i)$. If even a mild bias is integrated, the polytopes will be biased and, as a consequence, both, vehicle $i$ and its neighbors $\mathcal{N}_i$ calculate their CA constraints using inputs that are inaccurate and may collide. State integration is common when using sensors like an IMU in the absence of a global positioning system, like a GNSS (for example under a tunnel or in an urban area), to correct the state estimation. In contrast, the Incremental Planner shifts neighbors' states using local sensors and then calculates their polytopes, which in practice solves this problem and increases robustness to biased state conditions.

In the third scenario, it is important to note that results seem similar, and it looks like the vanilla distributed planner promotes collaboration between vehicles. Nonetheless, vehicles do not cooperate in this planner and the reason of this seemingly cooperation is that individual references were carefully generated by hand, based on the graph designed for the graph-based planners. As mentioned when discussing the formulation, the authors of the vanilla planner do not address the generation of references, which must be feasible or the planner cannot accomplish the task, adding an extra degree of complexity. On the other hand, graph-based approaches work directly with graphs, which is a clear advantage over the vanilla distributed planner, in terms of complexity and scalability.

In the last scenario, the complexity was scaled up by combining the results from the previous scenarios. The results show that different maneuvers can be performed in parallel in different sections of the platoon, without propagating disturbances upwards or backwards. The findings of this scenario can be extrapolated to a theoretical infinite platoon across the road that maximizes road throughput and it is still capable of performing essential internal rearranges in any section of the platoon, without impacting road performance. Moreover the complexity of the algorithm does not increase, since only local communication is used.

# 5. CONCLUSIONS

In this work, graph-based distributed motion planners for tight multi-lane platoons are presented. Inspired by a vanilla distributed planner, two graph-based solutions are proposed: a formulation that tracks references referred to a global frame, and an incremental formulation. Among the advantages of graph-based planners over the vanilla formulation are: scalability, robustness to disturbances, the direct synthesis of references from graphs, and the promotion of a cooperative behavior between vehicles.

The Graph-based distributed planner was achieved by enhancing the vanilla distributed planner with a graph-based cost function, where neighbor vehicles track a reference among their states differences. The planner scales by the fact that each vehicle only requires the local information from their neighbors, which is referred to a global reference frame. Additionally, the planner is designed to perform complex maneuvers in tight environments, which is possible because it considers the (almost) exact area occupied by each vehicle through a convex polytope to solve the DNMPC problem with an explicit CA hard constraint.

The Incremental Graph-based distributed planner was developed directly from the Graph-based one, and its design was motivated to overcome the uncertainty errors of the information communicated by each vehicle neighbors, which is achieved by translating the neighbors states from a global to a local reference frame. As a consequence, it makes the algorithm robust to non-zero mean uncertainty conditions.

Evaluation in a variety of scenarios shows that graph-based planners can accomplish several maneuvers inside the platoon, ranging from simple lane-changes to complex rearranges due to line closures. To test the scenarios, all three approaches assume that: vehicles could be modeled using the KBM with known parameters; there are no external perturbations such as obstacles and human drivers; perfect synchronization within the vehicles to perform the planning algorithms. It was shown that the vanilla distributed planner (benchmark) cannot perform all the experiments. And although the results look

cooperative-like, this was only because it was imposed explicitly by the single-agent reference. Graph-based distributed planners were shown to outperform the vanilla distributed planner due to their natural cooperative behavior within agents due to the reference from graphs. Nonetheless, both planners were incapable of accomplishing the tracking experiment when information was biased. The experiments show that the Incremental Graph-based distributed planner was capable of overcoming all the experiments, including the biased ones, while keeping all the advantages of the previous planners.

Considering the evaluation in a 36-vehicle platoon, it can be concluded that the graph-based distributed planners are appealing schemes for coordinating multi-lane platoons due to the fact that they use explicit CA constraints for tight environments, they naturally cooperate while using a reference from graphs and the local neighbors' information, and finally, they are scalable and robust. In addition, the presented planners resulted in two papers, one conference paper (Pizarro & Núñez, 2021) presented in *2021 IEEE Conference on Control Technology Applications (CCTA)*, and a second paper in *Control Engineering Practice* under review.

## 5.1. Open Challenges

There are two open topics to keep improving the proposed framework. The first is to adapt the planner for vehicles joining or leaving the platoon without deteriorating the overall performance, while the second is associated to the reference generator from graphs. Currently, the reference generator creates graphs for platoons with homogeneous vehicles, under known highway topologies and in a centralized fashion. In order to exploit the full potential of graph-based planners, the reference generator should be upgraded to work under heterogeneous vehicles dimensions, under different road topologies and in a distributed fashion.

Concerning the former point, the scheme should be fully oriented to preserving the safeness of the platoon by accomplishing conditions for joining or leaving the platoon. The

minimum set of conditions should at least consider: the state and dimension interaction within joining/leaving vehicle and the surrounding vehicles; if there is a deadline to join or leave the platoon; and the road conditions. A possible approach for this should be through an explicit set of hand-crafted rules to keep a full control and explainability of the algorithm. Therefore, the program will only be executed when there are safe and known situations.

The second point to improve is the reference generator, which seems much complex than the first problem due to its three requirements: heterogeneous vehicles, different road topologies and distributed implementation. Approaching this problem through a hand-crafted set of rules could become unmanageable. Therefore, an alternative is to use learning-based techniques, more specifically, Deep Reinforcement Learning (DRL), which is the learning branch focused on Control and Decision Making. DRL has been gaining traction in recent years due to its outstanding results in this area, for example in learning to play Atari games (Mnih et al., 2013) and Robot Control from states or images (Haarnoja, Zhou, Abbeel, & Levine, 2018; Laskin, Srinivas, & Abbeel, 2020). Even though learning-based approaches do not consider explicit constraints, DRL has been used as a reference generator for MPC controllers with CA (Brito, Everett, How, & Alonso-Mora, 2021; Brito, Agarwal, & Alonso-Mora, 2021), where the learning-based approach generates a feasible trajectory, while the MPC smooths it. Particularly in the proposed application, DRL can be goal-conditioned (Ghosh et al., 2019; Lynch et al., 2020) and map-conditioned with encoded maps (J. Gao et al., 2020). Goal-conditioning could be used to impose desired distances within agents and map-conditioned to learn from different road topologies. Furthermore, recent approaches on multi-agent DRL are focusing on communications within agents and distributed applications (Q. Li, Gama, Ribeiro, & Prorok, 2020; Foerster, Assael, De Freitas, & Whiteson, 2016; Rashid et al., 2018). Finally, all these ideas could be implemented in a Mixed-Autonomy Traffic Framework like Flow (Wu, Kreidieh, Parvate, Vinitsky, & Bayen, 2021) by using a 100% of CAVs penetration.

# REFERENCES


Ahn, S., & Cassidy, M. J. (2007). Freeway traffic oscillations and vehicle lane-change maneuvers. In *Transportation and traffic theory 2007. papers selected for presentation at isttt17engineering and physical sciences research council (great britain) rees jeffreys road fundtransport research foundationtms consultancyove arup and partners, hong kongtransportation planning (international) ptv ag.*

Alonso-Mora, J., Beardsley, P., & Siegwart, R. (2018). Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics*, *34*(2), 404–420.

Amer, N. H., Zamzuri, H., Hudha, K., & Kadir, Z. A. (2017). Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges. *Journal of intelligent & robotic systems*, *86*(2), 225–254.

Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, *14*(6), 926–939.

Bamieh, B., Jovanovic, M. R., Mitra, P., & Patterson, S. (2012). Coherence in large-scale networks: Dimension-dependent limitations of local feedback. *IEEE Transactions on Automatic Control*, *57*(9), 2235–2249.

Borrelli, F., Bemporad, A., & Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.

Brito, B., Agarwal, A., & Alonso-Mora, J. (2021). Learning interaction-aware guidance policies for motion planning in dense traffic scenarios. *arXiv preprint arXiv:2107.04538*.

Brito, B., Everett, M., How, J. P., & Alonso-Mora, J. (2021). Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robotics and Automation Letters*, *6*(3), 4616–4623.

Bruni, L., Colombo, A., & Del Vecchio, D. (2013). Robust multi-agent collision avoidance through scheduling. In *52nd ieee conference on decision and control* (pp. 3944–3950).

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172.

Camacho, E. F., & Alba, C. B. (2013). *Model predictive control*. Springer science & business media.

Chan, E., Gilhead, P., Jelinek, P., Krejci, P., & Robinson, T. (2012). Cooperative control of sartre automated platoon vehicles. In *19th its world congressertico-its europeeuropean commissionits americaits asia-pacific*.

Chen, Y. F., Liu, M., Everett, M., & How, J. P. (2017). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 ieee international conference on robotics and automation (icra)* (pp. 285–292).

Dang, R., Wang, J., Li, S. E., & Li, K. (2015). Coordinated adaptive cruise control system with lane-change assistance. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2373–2383.

Dax, A. (2006). The distance between two convex sets. *Linear Algebra and its Applications*, *416*(1), 184–213.

Dixit, S., Fallah, S., Montanaro, U., Dianati, M., Stevens, A., Mccullough, F., & Mouzakitis, A. (2018). Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects. *Annual Reviews in Control*, *45*, 76–86.

Fan, T., Long, P., Liu, W., & Pan, J. (2020). Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, *39*(7), 856–892.

Firoozi, R., Ferranti, L., Zhang, X., Nejadnik, S., & Borrelli, F. (2020). A distributed multi-robot coordination algorithm for navigation in tight environments. *arXiv preprint arXiv:2006.11492*.

Firoozi, R., Zhang, X., & Borrelli, F. (2021). Formation and reconfiguration of tight multi-lane platoons. *Control Engineering Practice*, *108*, 104714.

Foerster, J. N., Assael, Y. M., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, *4*(1), 23-33. doi: 10.1109/100.580977

Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., & Schmid, C. (2020). Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11525–11533).

Gao, L., Chu, D., Cao, Y., Lu, L., & Wu, C. (2019). Multi-lane convoy control for autonomous vehicles based on distributed graph and potential field. In *2019 ieee intelligent transportation systems conference (itsc)* (pp. 2463–2469).

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., & Levine, S. (2019). Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*.

Gurobi Optimization, L. (2021). *Gurobi optimizer reference manual.* Retrieved from `http://www.gurobi.com`

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (pp. 1861–1870).

Hao, H., Barooah, P., & Mehta, P. G. (2011). Stability margin scaling laws for distributed

formation control as a function of network structure. *IEEE Transactions on Automatic Control*, *56*(4), 923–929.

Harker, B. (2001). Promote-chauffeur ii & 5.8 ghz vehicle to vehicle communications system.

Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., & Balas, G. J. (2007). Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on control systems technology*, *16*(1), 19–33.

Kianfar, R., Augusto, B., Ebadighajari, A., Hakeem, U., Nilsson, J., Raza, A., ... Wymeersch, H. (2012). Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge. *IEEE Transactions on Intelligent Transportation Systems*, *13*(3), 994-1007. doi: 10.1109/TITS.2012.2186513

Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 ieee intelligent vehicles symposium (iv)* (p. 1094-1099). doi: 10.1109/IVS.2015.7225830

Kreidieh, A. R., Wu, C., & Bayen, A. M. (2018). Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning. In *2018 21st international conference on intelligent transportation systems (itsc)* (pp. 1475–1480).

Laskin, M., Srinivas, A., & Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning* (pp. 5639–5650).

Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2020). Graph neural networks for decentralized multi-robot path planning. In *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 11785–11792).

Li, S. E., Zheng, Y., Li, K., Wu, Y., Hedrick, J. K., Gao, F., & Zhang, H. (2017). Dynamical modeling and distributed control of connected and automated vehicles: Challenges and

opportunities. *IEEE Intelligent Transportation Systems Magazine*, *9*(3), 46–58.

Lin, F., Fardad, M., & Jovanovic, M. R. (2011). Optimal control of vehicular formations with nearest neighbor interactions. *IEEE Transactions on Automatic Control*, *57*(9), 2203–2218.

Liu, H., Zhuang, W., Yin, G., Tang, Z., & Xu, L. (2018). Strategy for heterogeneous vehicular platoons merging in automated highway system. In *2018 chinese control and decision conference (ccdc)* (p. 2736-2740).

Liu, P., Kurt, A., & Ozguner, U. (2019). Distributed model predictive control for cooperative and flexible vehicle platooning. *IEEE Transactions on Control Systems Technology*, *27*(3), 1115-1128.

Löfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in matlab. In *In proceedings of the cacsd conference.* Taipei, Taiwan.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., & Sermanet, P. (2020). Learning latent plans from play. In *Conference on robot learning* (pp. 1113–1132).

Marjovi, A., Vasic, M., Lemaitre, J., & Martinoli, A. (2015). Distributed graph-based convoy control for networked intelligent vehicles. In *2015 ieee intelligent vehicles symposium (iv)* (pp. 138–143).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Núñez, F., Wang, Y., & Doyle, F. J. (2014). Synchronization of pulse-coupled oscillators on (strongly) connected graphs. *IEEE Transactions on Automatic Control*, *60*(6), 1710–1715.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, *51*(3), 401-420. doi: 10.1109/TAC.2005.864190

Olfati-Saber, R., & Murray, R. M. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. *IFAC Proceedings Volumes*, *35*(1), 495–500.

Pizarro, G., & Núñez, F. (2021). Graph-based distributed lane-change in tight multi-lane platoons. In *2021 ieee conference on control technology and applications (ccta)* (p. 1031-1036).

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, *11*(7), 733–764.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning* (pp. 4295–4304).

Ren, W., & Cao, Y. (2011). *Distributed coordination of multi-agent networks: emergent problems, models, and issues* (Vol. 1). Springer.

Rupp, A., & Stolz, M. (2017). Survey on control schemes for automated driving on highways. In *Automated driving* (pp. 43–69). Springer.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., ... Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, *33*(9), 1251–1270.

Seiler, P., Pant, A., & Hedrick, K. (2004). Disturbance propagation in vehicle strings. *IEEE Transactions on automatic control*, *49*(10), 1835–1842.

Shladover, S. E., Desoer, C. A., Hedrick, J. K., Tomizuka, M., Walrand, J., Zhang, W. ., ... McKeown, N. (1991). Automated vehicle control developments in the path program. *IEEE Transactions on Vehicular Technology*, *40*(1), 114-130.

Smith, S. W., Kim, Y., Guanetti, J., Li, R., Firoozi, R., Wootton, B., ... Arcak, M. (2020). Improving urban traffic throughput with vehicle platooning: Theory and experiments. *IEEE Access*, *8*, 141208-141223.

Snider, J. M., et al. (2009). Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*.

Sorniotti, A., Barber, P., & De Pinto, S. (2017). Path tracking for automated driving: A tutorial on control system formulations and ongoing research. *Automated driving*, 71–140.

Swaroop, D., & Hedrick, J. K. (1996). String stability of interconnected systems. *IEEE transactions on automatic control*, *41*(3), 349–357.

Talebpour, A., & Mahmassani, H. S. (2016). Influence of connected and autonomous vehicles on traffic flow stability and throughput. *Transportation Research Part C: Emerging Technologies*, *71*, 143–163.

Tsugawa, S., Kato, S., & Aoki, K. (2011). An automated truck platoon for energy saving. In *2011 ieee/rsj international conference on intelligent robots and systems* (p. 4109-4114). doi: 10.1109/IROS.2011.6094549

Turri, V., Besselink, B., & Johansson, K. H. (2017). Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *IEEE Transactions on Control Systems Technology*, *25*(1), 12-28.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, *106*(1), 25–57.

Wang, C., & Nijmeijer, H. (2015). String stable heterogeneous vehicle platoon using cooperative adaptive cruise control. In *2015 ieee 18th international conference on intelligent transportation systems* (p. 1977-1982). doi: 10.1109/ITSC.2015.320

Wu, C., Bayen, A. M., & Mehta, A. (2018). Stabilizing traffic with autonomous vehicles. In *2018 ieee international conference on robotics and automation (icra)* (p. 6012-6018).

Wu, C., Kreidieh, A. R., Parvate, K., Vinitsky, E., & Bayen, A. M. (2021). Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*.

Xiao, L., & Gao, F. (2011). Practical string stability of platoon of adaptive cruise control vehicles. *IEEE Transactions on intelligent transportation systems*, *12*(4), 1184–1194.

Zhang, X., Liniger, A., & Borrelli, F. (2020). Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 1-12. doi: 10.1109/TCST.2019.2949540

Zheng, Y., Li, S. E., Li, K., Borrelli, F., & Hedrick, J. K. (2016). Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies. *IEEE Transactions on Control Systems Technology*, *25*(3), 899–910.

Zheng, Y., Li, S. E., Li, K., & Wang, L.-Y. (2015). Stability margin improvement of vehicular platoon considering undirected topology and asymmetric control. *IEEE Transactions on Control Systems Technology*, *24*(4), 1253–1265.

Zheng, Y., Li, S. E., Wang, J., Cao, D., & Li, K. (2015). Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies. *IEEE Transactions on intelligent transportation systems*, *17*(1), 14–26.

Čáp, M., Novák, P., Kleiner, A., & Selecký, M. (2015). Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 835-849. doi: 10.1109/TASE.2015.2445780

# APPENDIX

## A. GRAPH REFERENCE INTERPOLATION

Reference generation is conditioned to two cases: a constant formation or a formation transition. In the first case, references are given by $\mathbf{d}_{ij}^{(m)}(k|t) = \mathbf{d}_{ij}^{(m)}, \forall k = t, \ldots, t + N_c$ with the distance constraints $\mathbf{d}_{ij}^{(m)} \in \mathcal{D}_N^{(m)}$ given by the formation $\bar{\mathcal{G}}_N^{(m)}$, since there are multiple graphs, the superindex $(m)$ is used to differentiate between them.

Concerning formation transitions a platoon formation with virtual leader from $\bar{\mathcal{G}}_N^{(m_1)}$ to $\bar{\mathcal{G}}_N^{(m_2)}$, a linear interpolation function is used for smoother transitions. As a design feature, formation transitions are restricted to either lateral or longitudinal maneuvers, not both, and in this way, collision-free formations are ensured and increase the reliability of the planning scheme.

The interpolation function is defined in Algorithm 6. The function is used for each distance reference associated to vehicles' $i$ and $j$. To generate the reference sequence $\mathbf{d}_{ij}^{(m)}(\cdot|t)$, an initial state difference reference $\mathbf{d}_{ij}^{(m_1)} \in \mathcal{D}^{(m_1)}$ is interpolated with a goal state difference reference $\mathbf{d}_{ij}^{(m_2)} \in \mathcal{D}^{(m_2)}$, each one associated with formations graphs $\bar{\mathcal{G}}_N^{(m_1)}$ and $\bar{\mathcal{G}}_N^{(m_2)}$, respectively. To execute the function, the following parameters are used: parameter $\tau \in [0, 1]$ which is saturated with the function $\text{sat}_0^1$ between $0$ and $1$. Parameters $t \in \mathbb{R}^+$ and $t_0 \in \mathbb{R}^+$ are the actual time and the start time of the maneuver, respectively. $T_s \in \mathbb{R}^+$ is the execution time, while $\Delta t \in \mathbb{R}^+$ and $N_c \in \mathbb{Z}^+$ are the controller time step and planning horizon.

**Algorithm 6**

**function** G($\mathbf{d}_{ij}^{m_1}, \mathbf{d}_{ij}^{m_2}, t, t_0, T_s, \Delta t, N_c$)
    sequence = []
    **for** $k = 0 : N_c$ **do**
        $\tau = sat_0^1((t_0 + T_s - t - k\Delta t)/Ts)$
        $\mathbf{d} = \tau \mathbf{d}_{ij}^{m_1} + (1 - \tau)\mathbf{d}_{ij}^{m_2}$
        sequence = [sequence, $\mathbf{d}$]
    **return** sequence
    **Note:** sequence corresponds to $\mathbf{d}_{ij}^{(m)}(\cdot|t)$

## B. DNMPC WEIGHT MATRICES

### B.1. Distributed Motion Planner Weight Matrices

The weight matrices used to evaluate the cost function $J_{1,i}(\mathbf{z}_i, \mathbf{u}_i)$ for the experiments carried out in Chapter 4 are presented here in a compact notation where diag$[\cdot]$ represents the values of the main diagonal of a square matrix.

$$\mathbf{Q}_z = \text{diag}[0.01, 10, 0.1, 0.01]$$

$$\mathbf{Q}_u = \text{diag}[0.1, 0.1]$$

$$\mathbf{Q}_{\Delta u} = \mathbf{0}_{2\times 2}$$

### B.2. Graph-based Distributed Planners Weight Matrices

The weight matrices used to evaluate the cost function $J_{2,i}(\mathbf{z}_i, \mathbf{u}_i)$ for the experiments carried out in Chapter 4 are presented here in a compact notation where diag$[\cdot]$ represents the values of the main diagonal of a square matrix.

$$\mathbf{Q}_z = \text{diag}[0.5, 0.05, 0, 0.25]$$

$$\mathbf{Q}_0 = \text{diag}[0, 10, 1, 0.01]$$

$$\mathbf{Q}_u = \text{diag}[0.01, 1]$$

$$\mathbf{Q}_{\Delta u} = \text{diag}[0.05, 5]$$