Department of Mechanical Engineering
& Sustainable Energy Technology

Experimental verification of a scalable consensus protocol applied to vehicle platooning

Internship Project

Student:
T.J.A. van Oorschot

ID-number:
1352725

TU/e supervisor:
dr. prof. E. Lefeber

Research group:
Dynamics & Control

Lund University supervisor:
dr. M. Jeeninga

Research group:
Department of automatic control

Eindhoven, July 4, 2024

Abstract

This report is concerned with a recently proposed scalable protocol for achieving consensus in networks, known as the *serial consensus* protocol. This achieves coordination of integrator systems of any order through a series connection of first-order conventional consensus protocols, and has been theoretically shown to have advantageous stability and performance properties.

We implement this protocol on a vehicle platoon of five robots in a lab setting, where it is subject to measurement noise. We present experimental verification that the system is scalably stable, which is in accordance with theoretical findings. Moreover, we show theoretically that the parameters for which scalable stability occurs can be relaxed when information of the communication topology is available, which is also verified experimentally. In parallel, a conventional consensus protocol is implemented, which is known to *not* be scalably stable, which we also demonstrate in experiments. Lastly, we implement experiments to demonstrate the string stable behavior of the the serial consensus protocol in directed vehicle platoons, as predicted by theory, which again does not hold for conventional consensus.

Contents

1 Introduction

Dynamical systems over networks are a fundamental area of study in engineering due to their extensive use and the complex behaviours they enable. We refer to these networks as multi-agent systems. Multi-agent systems are widely used in control applications, such as distributed sensor networks and autonomous vehicle coordination in traffic and multi-robot networks. In these applications, we often want to achieve synchronization of all agents in the network. Think of a vehicle platoon, where all vehicles should have the same velocity. We refer to this synchronization of the agents as *consensus*.

Consensus is a central topic in multi-agent systems which asks how and under which conditions a network of agents achieves synchronization through distributed control laws. In addition, multi-agent systems typically feature a degree of locality, typically described by the network-topological constraints of the system. For synchronization problems, it is natural to formulate a control law based on relative feedback, which means that only measurements of the differences between states such as positions, velocities and accelerations are available to the controller, for example implemented through radar measurements between neighbors. Jensen and Bamieh [1] give a thorough exposition on the topics of locality and relative feedback.

In many applications, the networks are becoming increasingly large-scale and complex. On these large scales, conventional consensus algorithms often exhibit poor dynamic behaviour. This includes notions of performance, but also instability. Swaroop et al. [2] and Seiler et al. [3] research the *string instability* of conventional consensus, highlighting the poor performance as the transient error in a network grows exponentially with the number of agents. The problem of instability is is highlighted by Tegling, Bamieh and Sandberg [4], showing that certain classes of graphs are subject to *scale fragilities* in networks with agent integrator dynamics of order two or higher. Scale fragility means that stability is lost when the number of agents in the network increases-e.g. see [5].

Recently a new consensus strategy has been developed by J. Hansson and E. Tegling [6], named serial consensus. The serial consensus algorithm is a linear and distributed controller using only local and relative measurements. The main advantage of serial consensus is its superior dynamic behaviour in large networks compared to conventional consensus. Hansson [6] shows that serial consensus is *scalably stable* for agent integrator dynamics of any order, meaning that it is not subject to scale fragilities. Furthermore, Hansson [7] reveals that serial consensus also guarantees a strong notion of string stability for networks with second order agent dynamics. This superior scalable performance of serial consensus comes at the cost of extra communication in the network. More specifically, the agents require relative information between the neighbours of all its neighbours of at most n steps away, where n is the order of the agent dynamics.

The new consensus strategy of Hansson and Tegling has only been considered in a theoretical framework. In reality, there are other relevant aspects such as measurement errors, transport delays, implementation on a multi-dimensional system, etcetera. Therefore, the main objective of this study is to provide an experimental verification of serial consensus and to compare its performance and robustness to a conventional consensus approach. To this end, the serial consensus algorithm is applied on a physical experimentation setup that mimics a vehicle platoon. In addition, we present a relaxation of the serial consensus algorithm, for which we mathematically prove and experimentally verify its scalable stability. It is also demonstrated that this relaxation, however, can lead to a loss of string stability.

The serial consensus protocol discussed in [6] only describes the behavior along the path that the platoon is following, and only applies to longitudinal control. To verify the protocol in a lab setting, both protocols are implemented on vehicles that follow a circular trajectory. Given the limited range of the sensory setup, this means that long-duration experiments may be performed. Furthermore, it allows to implement networks with circular communication. The agents of the network are represented by omnibots, which each have a maximum velocity. It is noted that the experimentation setup is subject to measurement noise and features a discrete-time implementation of the protocols, both of which were not covered by the theory in [6, 7]. We therefore argue that this provides a realistic implementation and verification of the protocol. Though the serial consensus algorithm guarantees scalable stability for any n^{th} order network, this study focuses second order agent dynamics. Vehicles driving with modern adaptive cruise control (ACC) can be modeled as second order integrators (see [8]), making this a realistic assumption in the context of vehicle platooning.

The remainder of this report is structured as follows. In Chapter 2 we specify the class of problems we are considering and give some definitions. In Chapter 3 we present some theoretical background, highlighting the poor scalable performance of conventional consensus. Chapter 4 introduces the novel serial consensus algorithm, summarizing the results of Hansson [6]. Chapter 5 shows how the conditions for scalable stability of the serial consensus algorithm can be relaxed, by introducing the *scissor condition*. Chapter 6 demonstrates that this relaxation does not guarantee string stability. Chapter 7 elaborates on the experimentation

setup, showing the control structure that keeps the omnibots on the circle and how the consensus algorithm is implemented along the path. Chapter 8 shows the results of the experiments on scalable stability, comparing the performance of serial consensus to conventional consensus. Chapter 9 presents the results of the experiments on string stability. Chapter 10 dives into a specific implementation of serial consensus proposed by Hansson [7] which requires less communication among agents. The report is concluded in Chapter 11.

2 Problem formulation

In this section, we specify the class of problems we are considering, and we give some definitions regarding consensus in networks. The main objective of this project is to provide an experimental verification of the serial consensus protocol introduced in [6] by implementing the protocol on a physical system. It is of interest how practical challenges such as measurement errors and time delays affect the performance and stability of the network dynamics. Furthermore, we want to compare the behaviour of the serial consensus protocol to the conventional consensus protocol. The conventional approach is is known to have very undesirable scalable behaviour, so we verify whether the serial consensus protocol improves this behaviour in a practical setting. The experimentation setup mimics vehicle platooning where the vehicles are following a circular trajectory. The serial consensus protocol theoretically guarantees scalable stability for agent dynamics of any order. Here, we focus on vehicles in the platoon that are modelled as second-order integrators. In the context of vehicle platooning this is a very relevant case, as a lot of vehicles can be modelled as second-order integrators (i.e. we can control the acceleration), see [8].

The experiments are performed on platoons with two different underlying networks, representing the communication structures. To verify the scalable stability, the vehicles can communicate over a directed circular network (see Figure 7.13). To verify the string stability, the vehicles can communicate over a directed linear network with a virtual leader agent (see Figure 7.14).

In the remainder of this section, first the network model and some definitions are defined, as well as the agent dynamics. Then the control structure considered throughout the project is introduced, making a distinction between the conventional consensus protocol and the serial consensus protocol. Finally, the definitions of scalable stability and string stability are introduced.

2.1 Network model and definitions

The vehicles in the platoon can communicate according to some communication topology. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a directed graph (or network) that encodes this communication topology. We denote the set of vehicles with $\mathcal{V}$, where $N = |\mathcal{V}|$ denotes the number of agents. We let the set $\mathcal{E}$ denote the edges (i.e., connections) in the graph. We say that if there is an edge from agent i to agent j, then agent i has access to the relative information of agent j. The graph can equivalently be represented by the communication adjacency matrix $W \in \mathbb{R}^{N\times N}$, where $W_{i,j} = 1 \Leftrightarrow (j,i) \in \mathcal{E}$ and $W_{ij} = 0$ otherwise. The graph is called undirected if $W^T = W$ and is called directed otherwise. We say that the graph contains a spanning tree if for some agent $i \in \mathcal{V}$ there is a directed path from i to any other agent $j \in \mathcal{V}$. For a *weighted* directed graph, we associate a positive weight $w_{ij} > 0$ to each edge $(i,j) \in \mathcal{E}$. The associated weighted graph Laplacian is defined as

$$L_{i,j} = \begin{cases} -w_{ij}, & \text{if } i \neq j \\ \sum_{k\neq i} w_{ik}, & \text{if } i = j \end{cases}. \tag{2.1}$$

The graph Laplacian L has a simple and unique eigenvalue at 0 and the remaining eigenvalues lie strictly in the right half plane (open RHP) if and only if the graph underlying the graph Laplacian contains a spanning tree. The smallest non-zero eigenvalue $\lambda_2(\mathcal{G})$ is called the algebraic connectivity. Following the approach in [4, 6, 7], we consider networks with a growing number of agents to test the scalable properties of the controllers. To this end, we define the *family* of graphs $\{\mathcal{G}_N\}$, where N is the size of the growing network.

2.2 Agent dynamics and consensus

We let the system be modelled as a network consisting of N vehicles with identical dynamics. The vehicles are numbered such that vehicle $i = 1$ is the first vehicle in the platoon, and $i = N$ is the last vehicle. We assume that all vehicles in the platoon adhere to the same path, and the goal is to control their position along this path. For each vehicle i we model its position along the path by the scalar x_i. Our aim is to control the vehicles in the platoon such that the inter-vehicular distance converges to a fixed value Δs^*. That is, we want to design the controllers such that the positions x_i converge to

$$x_i = x_{i+1} + \Delta s^* \quad \text{for } i = 1, \dots, N-1. \tag{2.2}$$

By defining the change of coordinates $\hat{x}_i(t) := x_i(t) - (i-1)\Delta s^*$, we observe that the objective in (2.2) is equivalent to

$$\hat{x}_i(t) = \hat{x}_{i+1}(t) \quad \text{for } i = 1, \dots, N-1. \tag{2.3}$$

We note from (2.3) that converging to a fixed inter-vehicular distance corresponds to achieving consensus among all vehicles in the shifted coordinates $\hat{x}_i$. Hence, vehicle platooning may be regarded as a consensus problem. We say that the multi-agent system achieves n^{th} order consensus if

$$\lim_{t\to\infty} |\hat{x}_i^k(t) - \hat{x}_j^k(t)| = 0 \quad \forall i, j \in \mathcal{V} \text{ and } \forall k \in \{0, 1, \dots, n-1\}. \tag{2.4}$$

We continue to model the dynamics of the vehicles in terms of $\hat{x}_i(t)$, and collect these shifted positions in the vector $\hat{x} \in \mathbb{R}^N$. To allow for more compact notation, the convention $\hat{x}_i^{(0)}(t) = \hat{x}_i(t)$ and $\hat{x}_i^{(k)}(t) = \frac{\mathrm{d}^k \hat{x}_i(t)}{\mathrm{d}t^k}$ are introduced to denote time derivatives. Since Δs^* is constant, we have that $\hat{x}_i^{(k)}(t) = x_i^{(k)}(t)$ for all $k \geq 1$. The dynamics of the agents are modelled as the n^{th} order integrator system

$$\frac{\mathrm{d}^n x_i(t)}{\mathrm{d}t^n} = u_i(t) \quad \text{for all} \quad i \in \mathcal{V}. \tag{2.5}$$

We focus on agents that have second-order integrator dynamics, meaning that $n = 2$. In this case, the dynamics of each agent read $\ddot{x}_i(t) = u_i(t)$.

2.3 Control structure

The proposed controller we consider is a linear state-feedback controller, which can be written as

$$u(t) = -\sum_{k=0}^{n-1} A_k \hat{x}^{(k)}(t) + u_{\text{ref}}(t), \tag{2.6}$$

where $u_{\text{ref}}(t) \in \mathbb{R}^N$ is a feedforward term, and $A_k \in \mathbb{R}^{N\times N}$ defines the feedback on the k^{th} derivative of the shifted positions $\hat{x}$. The class of controllers that we consider is restricted in three ways.

1. The controller can only use relative feedback. This translates to the condition that $A_k \mathbf{1}_N = 0$ for all k, i.e. the sum of elements on each row must be 0. Put differently, only the inter-vehicular measurements $x_i^{(k)} - x_j^{(k)}$ are available to the controller.
2. The controller must have a limited gain. This translates to enforcing that $||A_k||_\infty \leq c$.
3. The controller must only depend on the local neighbourhood of each agent. The local neighbourhood is quantified by q, enforcing that the controller only uses information that is at most q steps away. Therefore, if we want the controller to only use information that is at most q steps away, it should hold that $\sum_{k=0}^{q} W_{i,j}^k = 0 \Rightarrow [A_k]_{i,j} = 0$.

Condition (3) ensures that the serial consensus protocol can be implemented, as serial consensus requires an n-hop neighbourhood for n^{th} order agent dynamics. If a controller satisfies all three conditions, it is called a q-step implementable relative feedback controller w.r.t. the adjacency matrix W. The family of all q-step implementable relative feedback controllers is given by the set

$$\mathcal{A}^q(W, c) = \left\{ A \;\middle|\; \begin{array}{c} \left[\sum_{k=0}^{q} W^k\right]_{i,j} = 0 \Longrightarrow A_{i,j} = 0 \\ A\mathbf{1}_N = 0, \|A\|_\infty \leq c \end{array} \right\}. \tag{2.7}$$

2.3.1 Conventional consensus algorithm

The control input for agent i of conventional n^{th} order consensus is given by

$$u_i = -\sum_{k=0}^{n-1} a_k \sum_{j\in\mathcal{N}_j} w_{ij}(\hat{x}_i^{(k)} - \hat{x}_j^{(k)}), \tag{2.8}$$

where $a_k > 0$ are fixed gains on the k^{th} derivative of the states. This results in the stacked control input

$$u(t) = -\sum_{k=0}^{n-1} a_k L_k \hat{x}^{(k)}(t) + u_{\text{ref}}(t), \tag{2.9}$$

for some graph Laplacians L_k. Note that this conventional controller is a 1-step implementable relative feedback controller, as all $A_k = L_k \in \mathcal{A}^1(W, c)$. In many cases, the Laplacians L_k are assumed to be the same such that $L_k = b_k L$ for some graph Laplacian L and constants $b_k > 0$.

2.3.2 Serial consensus algorithm

For serial consensus, the stacked control input of the n^{th} order serial consensus algorithm is given by

$$u(t) = -\sum_{k=0}^{n-1} A_k \hat{x}^{(k)}(t) + u_{\text{ref}}(t), \tag{2.10}$$

where each $A_k = \sum_{\alpha \in \mathcal{I}_k} \prod_{j \in \alpha} L_j$ for all $k \in [0, n-1]$. $\mathcal{I}_k$ denotes all the ordered subsets of the range $[1, n]$ with size $n - k$. This is defined as $\mathcal{I}_k = \{\ \alpha \mid |\alpha| = n - k, \alpha \subset \{1, 2, \ldots, n\}, i < j \Rightarrow \alpha(i) < \alpha(j)\ \}$. Chapter 4 gives an example, showing the resulting A_k for $n = 3$. Note that this serial controller is an n-step implementable relative feedback controller, as all $A_k \in \mathcal{A}^n(W, c)$. This means that the favourable scalable properties of serial consensus come at the cost of extra communication in the network. The serial controller needs information of agents in its n-hop neighbourhood, whereas the conventional control only requires information of agents in its 1-hop neighbourhood. This requirement for extra communication is further elaborated in Chapter 4.

2.4 Scalable stability and string stability

It was shown in [4] that conventional consensus algorithms in networks where the agents have integrator dynamics of order two or higher are sensitive to scale fragilities in certain families of graphs, meaning that stability is lost as the network scales. To this end, we introduce the following notion of scalable stability as defined in [4]: A consensus control design is scalably stable if the resulting closed loop system achieves consensus over any graph in the family $\{\mathcal{G}_N\}$ independent of N, meaning that it is not subject to scale fragilities.

The notion of scalable stability of a controller presumes a modular design principle. This means that new agents are added to the network with pre-designed controller gains a_k satisfying $a_k \leq a_{\max} < \infty$, and that these gains cannot be re-tuned as the network grows. Furthermore, all nodes in the graph family $\{\mathcal{G}_N\}$ have a neighbourhood of at most q, where q is independent of N. Put differently, $|\mathcal{N}_i| \leq q\ \forall i \in \mathcal{V}_N$. Finally, the edge weights in $\{\mathcal{G}_N\}$ are finite, that is $w_{ij} \leq \omega_{\max} < \infty$ for all $(i, j) \in \mathcal{E}_N$ where ω_{max} is fixed and independent of N.

In addition, we introduce a notion of performance for agents with second order integrator dynamics as defined by Hansson [7]. To this end, the relative position $e_p(t)$ and velocity deviation $e_v(t)$ are introduced as

$$\begin{aligned} e_p(t) &:= d + Lx(t) \\ e_v(t) &:= \dot{x}(t) - v_{ref}\mathbf{1}, \end{aligned} \tag{2.11}$$

with $d \in \mathbb{R}^N$ being a vector of desired positional offsets (e.g. $d = \Delta s^* \mathbf{1}$) and $v_{ref} \in \mathbb{R}$ being the desired vehicle velocity. The vector e_p represents the local relative position errors. In the context of vehicle platooning, these need to remain small to avoid vehicle collisions. The vector e_v represents the deviation from the desired velocity, which needs to remain small to respect the speed limits of the vehicles. The multi-agent system (2.5) is said to be string stable if there exists a fixed and finite α that ensures that

$$\sup_{t \geq 0} \left\| \begin{bmatrix} e_p(t) \\ e_v(t) \end{bmatrix} \right\|_\infty \leq \alpha \left\| \begin{bmatrix} e_p(0) \\ e_v(0) \end{bmatrix} \right\|_\infty. \tag{2.12}$$

Note that the choice of the norm is important, as it bounds the worst-case behaviour by the initial maximum deviation. Although the disturbance amplification scenario also requires careful analysis as remarked by Besselink and Knorn [9], we leave this outside the scope of this project.

In the next section, we will give some theoretical background regarding conventional protocols that achieve consensus in networks. More specifically, we highlight the poor scalable behaviour of this conventional approach.

3 Theoretical background

Now, we highlight the poor scalable behaviour of the conventional consensus approach. The novel serial consensus algorithm introduced in [6] is mathematically proven to be scalably stable and string stable. This is in contrast to the conventional consensus algorithm in (2.9), which is neither scalably stable nor string stable. In this section, we explain how conventional consensus is sensitive to scale fragilities when the agents have integrator dynamics of order two or higher, showing its poor scalable behaviour. This is a summary of the results presented in [4].

3.1 Scale fragilities in high-order consensus

First we present the important result of [4], which states that conventional high-order consensus lacks scalable stability in graph families with a decreasing algebraic connectivity. More specifically, it is proven that if $n \geq 3$, then no conventional control of the form (2.9) is scalably stable in graph families where $\mathrm{Re}\{\lambda_2\{\mathcal{G}_N\}\} \to 0$ as $N \to \infty$.

To see this, we define the full state vector $\xi = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(n-1)} \end{bmatrix}^T \in \mathbb{R}^{Nn}$. The corresponding closed loop dynamics are

$$\frac{\mathrm{d}}{\mathrm{d}t}\xi = \underbrace{\begin{bmatrix} 0 & I_N & 0 & \dots & 0 \\ 0 & 0 & I_N & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 L & -a_1 L & -a_2 L & \dots & -a_{n-1} L \end{bmatrix}}_{\mathcal{A}} \xi, \tag{3.1}$$

with L being a graph Laplacian as defined in (2.1) and I_N being the $N \times N$ identity matrix. We define matrix T by taking its columns as the eigenvectors of L, meaning that T is an invertible $N \times N$ matrix such that $\Lambda = T^{-1}LT$ is in Jordan normal form. If the graph is undirected, then L is symmetric and therefore diagonizable. If the graph is directed, Λ consists of k Jordan blocks, with k being the number of linearly independent eigenvectors. We define the block-diagonal matrix $\mathbf{T} = \mathrm{diag}(T, T, \dots, T)$, and pre- and post-multiply $\mathcal{A}$ by $\mathbf{T}$ to obtain

$$\hat{\mathcal{A}} = \mathbf{T}^{-1}\mathcal{A}\mathbf{T} = \underbrace{\begin{bmatrix} 0 & I_N & 0 & \dots & 0 \\ 0 & 0 & I_N & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 \Lambda & -a_1 \Lambda & -a_2 \Lambda & \dots & -a_{n-1} \Lambda \end{bmatrix}}_{\hat{\mathcal{A}}}. \tag{3.2}$$

We pre- and post-multiply $\hat{\mathcal{A}}$ by an appropriate permutation matrix to rearrange its rows and columns into the matrix $\mathrm{diag}(\hat{\mathcal{A}}_1, \dots, \hat{\mathcal{A}}_k)$ with

$$\hat{\mathcal{A}}_l = \begin{bmatrix} 0 & I & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & I \\ -a_0 \Lambda_l & -a_1 \Lambda_l & \dots & -a_{n-1} \Lambda_l & -a_n \Lambda_l \end{bmatrix} \tag{3.3}$$

for $l = 1, \dots, k$. Due to the block-diagonal structure, the eigenvalues of $\mathcal{A}$ are given by the union of eigenvalues of all $\hat{\mathcal{A}}_l$. Since $\lambda_1 = 0$ is a simple and unique eigenvalue of L, $\hat{\mathcal{A}}_1$ is an $n \times n$ Jordan block with its eigenvalues at 0. To ensure scalable stability, it must therefore hold that all eigenvalues of all $\hat{\mathcal{A}}_l$ with $l = 2, \dots, k$ lie in the open left half plane for any N. The characteristic polynomial of each $\hat{\mathcal{A}}_l$ is denoted by $P_l(s)$ and reads

$$P_l(s) = \underbrace{\left(s^n + a_{n-1}\lambda_l s^{n-1} + \dots + a_1 \lambda_l s^1 + a_0 \lambda_l\right)}_{p_l(s)}{}^{r_l} \tag{3.4}$$

whose roots are given by the the roots of $p_l(s)$. The roots of $P_l(s)$ represent the eigenvalues of A. Therefore, we want these roots to have a negative real part. To see when this is the case, we apply the Routh-Hurwitz criteria. In general, the eigenvalues λ_l can be complex-valued. Therefore, we apply the Routh-Hurwitz criteria for polynomials with complex coefficients, which are recalled in the appendix of [4]. The first Routh-Hurwitz criterion applied to $p_l(s)$ reads that $a_{n-1}\text{Re}\{\lambda_l\} > 0$ for $l = 2, \dots, k$. This is always satisfied, as both $a_{n-1} > 0$ and $\text{Re}\{\lambda_l\} > 0$ for $l = 2, \dots, k$. The second criterion can be written as

$$a_{n-1}(\text{Re}\{\lambda_l\})^2(a_{n-1}a_{n-2}\text{Re}\{\lambda_l\} - a_{n-3}) + a_{n-2}(\text{Im}\{\lambda_l\})^2(a_{n-1}^2\text{Re}\{\lambda_l\} - a_{n-2}) > 0, \tag{3.5}$$

which must hold for all $l = 2, \dots, k$. This condition is violated when the expressions in brackets are negative (recall that all $a_k > 0$). The brackets are negative if $\text{Re}\{\lambda_2\} = \min_l \text{Re}\{\lambda_l\}$ is sufficiently small. Therefore, if the Routh-Hurwitz criterion (3.5) is evaluated for a graph family $\{\mathcal{G}_N\}$ where $\text{Re}\{\lambda_2(\mathcal{G}_N)\} \to 0$ as $N \to \infty$, it is eventually violated for a large (but finite) N. This means that at least one of the eigenvalues of A has a non-negative part and the serial consensus protocol is not scalably stable.

3.2 Scale fragility in second order consensus

Now, we present the important result on scale fragility of [4] for the specific case where the agents have second orde integrator dynamics. This case is especially relevant, as many formation control problems use second order dynamics to model the agents in their network (e.g. [8]). In second order consensus, scalable stability is guaranteed if the underlying graph family is undirected. However, this is not the case for directed networks. More specifically, conventional control is subject to scale fragilities in families of directed graphs where the real part of one or more eigenvalues of the Laplacian approaches zero as N grows and at least one of these eigenvalues has a sufficiently large imaginary part.

To see this, we consider the characteristic polynomial in (3.4) for $n = 2$, given by $p_l(s) = s^2 + a_1\lambda_l s + a_0\lambda_l$ for $l = 2, \dots, N$. To see when the roots of this characteristic polynomial have a non-negative real part, we look at the corresponding Routh-Hurwitz criterion

$$a_1^2\text{Re}\{\lambda_l\}\left[\text{Re}\{\lambda_l\}^2 + \text{Im}\{\lambda_l\}^2\right] - a_0\text{Re}\{\lambda_l\}^2 > 0 \quad \text{for all} \quad l = 2, \dots, N. \tag{3.6}$$

If the underlying network is undirected, then $L^T = L$ and all eigenvalues are real-valued. Clearly, this implies that the Routh-Hurwitz criterion is satisfied as $\text{Im}\{\lambda_l\} = 0$ and $\text{Re}\{\lambda_l\} > 0$. Therefore, scalable stability of the network with second order agent dynamics is guaranteed when the underlying network is undirected.

If the underlying network is directed, eigenvalues may be complex-valued. Recall that Laplacian eigenvalues only appear in complex conjugate pairs. In this case, we can re-formulate the Routh-Hurwitz criterion as

$$a_1^2\text{Re}\{\lambda_l\}\left[\left(\frac{\text{Re}\{\lambda_l\}}{\text{Im}\{\lambda_l\}}\right)^2 + 1\right] - a_0 > 0. \tag{3.7}$$

If the expression in brackets can be upper bounded by some constant, this condition is violated whenever $\text{Re}\{\lambda_l\}$ is sufficiently small. Therefore, if the condition is evaluated for a graph family $\{\mathcal{G}_N\}$ in which there are eigenvalues for which $\text{Re}\{\lambda_l\} \to 0$ as $N \to \infty$ and it holds that $\left(\frac{\text{Re}\{\lambda_l\{\mathcal{G}_N\}\}}{\text{Im}\{\lambda_l\{\mathcal{G}_N\}\}}\right)^2 \leq$ const. for at least one of them, then the condition is eventually violated and stability is lost. The condition $\left(\frac{\text{Re}\{\lambda_l\{\mathcal{G}_N\}\}}{\text{Im}\{\lambda_l\{\mathcal{G}_N\}\}}\right)^2 \leq$ const. is equivalent to saying that $\frac{\text{Re}\{\lambda_l\{\mathcal{G}_N\}\}}{\text{Im}\{\lambda_l\{\mathcal{G}_N\}\}}$ must be upper bounded for all $l = 2, \dots, N$ in the first quadrant of the complex plane.

This leads to the realization (visualized in Figure 3.1) that, if $n = 2$, then no control of the form (2.9) is scalably stable in graph families where, for a fixed index $l \in \{2, 3, \dots, \bar{l}\}$, it holds that both

1. $\text{Re}\{\lambda_{\bar{l}}(\mathcal{G}_N)\} \to \infty$ as $N \to \infty$, and
2. for at least one $l \in \{2, 3, \dots, \bar{l}\}$ it holds that $\arg\{\lambda_l(\mathcal{G}_N)\} > \Psi$ where $\Psi \in (0, \pi/2)$ is a constant angle independent of N.

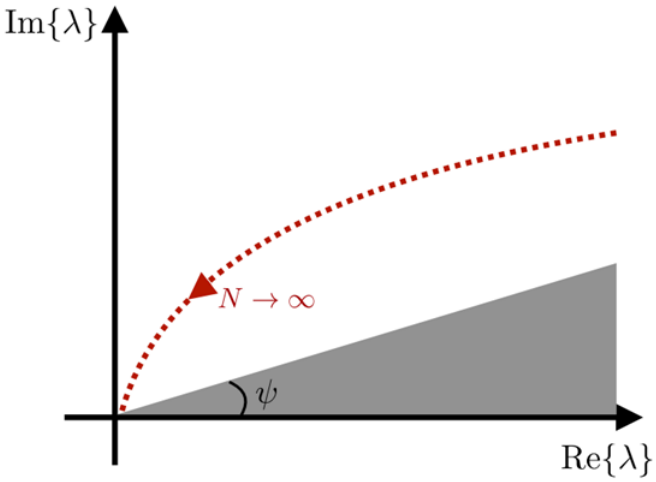

Figure 3.1: Illustration of condition 3.7. If a Laplacian eigenvalue approaches the origin at an angle greater than some Ψ as the network grows, the second order consensus lacks scalable stability. Figure is obtained from [4].

A particular graph family that fails to be scalably stable with second order agent integrator dynamics is the family of directed ring graphs. This was already observed in [5, 10, 11]. Therefore, this network is suitable to use in experiments to verify scalable stability.

Now, we have shown that the conventional control protocol causes very poor behaviour in large scale networks (i.e., networks with a growing number of agents) in the form of scale fragilities. In the next section, we present the novel control algorithm, referred to as serial consensus. We show theoretically that serial consensus has superior scalable behaviour compared to conventional consensus in the form of scalable stability. We do this by summarizing the results of [6].

4 Introduction to serial consensus

In this section, the novel control algorithm introduced by J. Hansson and E. Tegling [6] is summarized. The algorithm is referred to as serial consensus, as its closed loop dynamics of a network with n^{th} order integrators are equivalent to interconnecting n first order consensus systems in series. First, the closed-loop formulation of the serial consensus protocol is introduced. Next, it is shown that the serial consensus controller satisfies the constraints set for a n-step implementable relative feedback controller, meaning that the local controller only needs relative measurements of agents within its n-hop neighbourhood. Finally, it is shown that the controller satisfies scalable stability for the network.

4.1 Closed-loop form

In the Laplace domain, the closed-loop dynamics of the conventional controller are described by a matrix polynomial, resulting in counter-intuitive poles. To avoid this, the serial consensus controller is designed such that the closed-loop dynamics in the Laplace domain of the n^{th} order consensus system are given by

$$\left(\prod_{k=1}^{n}(sI+L_k)\right)\hat{X}(s)=\hat{U}_{ref}(s), \tag{4.1}$$

where L_k represent the weighted and directed graph Laplacian for all $k \in \{1,2,\dots,n\}$. The controller required to obtain these closed-loop dynamics is given by

$$\hat{U}(s)=\hat{U}_{ref}(s)+\left(s^nI-\prod_{k=1}^{n}(sI+L_k)\right)\hat{X}(s). \tag{4.2}$$

This formulation is referred to as the serial consensus formulation, because because the same closed-loop dynamics can also be achieved by interconnecting n first order consensus systems in series. This becomes clear when the dynamics are transformed to state-space form by introducing the alternative Laplace-domain variables $\hat{\Xi}_k$ with the corresponding time-domain states ξ_k. These states are related to $\hat{X}(s)$ through $\hat{\Xi}_1 = \hat{X}(s)$, $\hat{\Xi}_k = (sI+L_{k-1})\hat{\Xi}_{k-1}$ for $k \in \{2,\dots,n\}$ and $s\hat{\Xi}_n = -L_n\hat{\Xi}_n + \hat{U}_{ref}$. This leads to the following continuous-time state-space representation

$$\begin{bmatrix}\dot{\xi}_1\\ \dot{\xi}_2\\ \vdots\\ \dot{\xi}_{n-1}\\ \dot{\xi}_n\end{bmatrix}=\underbrace{\begin{bmatrix}-L_1 & I & \dots & 0 & 0\\ 0 & -L_2 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \dots & -L_{n-1} & I\\ 0 & 0 & \dots & 0 & -L_n\end{bmatrix}}_{A}\begin{bmatrix}\xi_1\\ \xi_2\\ \vdots\\ \xi_{n-1}\\ \xi_n\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ u_{ref}\end{bmatrix}. \tag{4.3}$$

4.2 Implementing serial consensus

Now we show that the n^{th} order serial consensus system in (4.1) can be achieved with an n-step implementable relative feedback controller w.r.t. the communication adjacency matrix W and a finite gain c' if each $L_k \in \mathcal{A}^1(W,c)$. To this end, we use that

$$A_1 \in \mathcal{A}^{q_1}(W,c_1),\ A_2 \in \mathcal{A}^{q_2}(W,c_2) \quad \Rightarrow \quad \begin{cases}(A_1+A_2) \in \mathcal{A}^{\max(q_1,q_2)}(W,c_1+c_2)\\ (A_1A_2) \in \mathcal{A}^{q_1+q_2}(W,c_1c_2)\end{cases}. \tag{4.4}$$

The derivation of this relation is presented in the appendix of [6]. The serial consensus controller in (4.2) can be expanded to the matrix polynomial

$$\hat{U}(s)=\hat{U}_{ref}(s)-\sum_{k=0}^{n-1}s^kA_k\hat{X}(s) \tag{4.5}$$

for some matrices A_k. To show that the serial consensus controller is indeed an n-step implementable relative feedback controller, it must be shown that $A_k \in \mathcal{A}^q(W,c')$ for all $k \in \{0,\dots,n-1\}$, with $q \leq n$ and $c' < \infty$. Let

$$\mathcal{I}_k = \{\alpha \quad | \quad |\alpha| = n-k, \alpha \subset \{1,2,\dots,n\}, i<j \Rightarrow \alpha(i)<\alpha(j)\} \tag{4.6}$$

denote all the ordered subsets of the range $[1,n]$ with size $n-k$. Then $A_k = \sum_{\alpha\in\mathcal{I}_k}\prod_{j\in\alpha} L_j$ for all $k\in[0,n-1]$. Since all $\alpha\in\mathcal{I}_k$ has $n-k$ elements we can show that $\prod_{j\in\alpha} L_j = B_\alpha \in \mathcal{A}^{n-k}\left(W, c^{n-k}\right)$ by applying (4.4) recursively. Now we have a sum

$$A_k = \sum_{\alpha\in\mathcal{I}_k} B_\alpha. \tag{4.7}$$

The number of ordered subsets of the range $[1,n]$ with size $n-k$ is given by the binomial coefficients and therefore the size of $|\mathcal{I}_k| = \binom{n}{n-k}$. Applying (4.4) recursively shows that $A_k \in \mathcal{A}^{n-k}(W, \binom{n}{n-k}c^{n-k})$. Clearly, we have that $n-k\leq n$ and $\binom{n}{n-k}c^{n-k} \leq \binom{n}{[n/2]}\max(c,c^n) < \infty$ for all k. Let $c' = \binom{n}{[n/2]}\max(c,c^n)$ and then we have that $A_k\in\mathcal{A}^n(W,c')$ for all k.

Example: 3^{rd} order serial controller

Consider the serial controller in the Laplace domain in (4.2) for $n=3$. Expanding this expression gives a controller of the form (2.10). Combining this with (4.4) shows that this controller is indeed a 3-step implementable relative feedback controller, as $A_0 = L_1L_2L_3 \in \mathcal{A}^3(W,c)$, $A_1 = L_1L_2+L_1L_3+L_2L_3 \in \mathcal{A}^2(W,c)$ and $A_2 = L_1+L_2+L_3\in\mathcal{A}^1(W,c)$:

$$\begin{aligned}
\hat{U}(s) &= \hat{U}_{ref}(s) + \left(s^3 I - (sI+L_1)(sI+L_2)(sI+L_3)\right)\hat{X}(s)\\
&= \hat{U}_{ref}(s) + \left(s^3 I - s^3 I + s^2(L_1+L_2+L_3) + s(L_1L_2+L_1L_3+L_2L_3) + L_1L_2L_3\right)\hat{X}(s)\\
&= \hat{U}_{ref}(s) + \left(s^2(L_1+L_2+L_3) + s(L_1L_2+L_1L_3+L_2L_3) + L_1L_2L_3\right)\hat{X}(s)\\
&= u_{ref} - \sum_{k=0}^{n-1} A_k x^{(k)}, \text{ where } A_0 = L_1L_2L_3, A_1 = L_1L_2+L_1L_3+L_2L_3, A_2 = L_1+L_2+L_3.
\end{aligned} \tag{4.8}$$

4.3 Scalable stability of serial consensus

We continue by proving that serial consensus is indeed scalably stable (this is a summary of the proof in [6]). To this end, consider the n^{th} order serial consensus system as defined in (4.1). We want to show the following two postulates: (i) The poles of the closed loop system (4.1) are given by the union of the eigenvalues of $-L_k$. Furthermore, (ii) the solution achieves n^{th} order consensus.

To see (i), recall that any square matrix can be unitarily transformed to upper triangular form by the Schur triangularization theorem. We define the transformation matrix T by choosing its columns as the eigenvectors of L. Therefore, $U_kL_kU_k^H = T_k$ is upper triangular, and pre- and post-multiplying A in (4.3) with the block diagonal matrix $U = \text{diag}(U_1,U_2,\dots,U_n)$ upper-triangularizes A. Therefore, its eigenvalues lie on the diagonal entries and are equal to the eigenvalues of each L_k.

To see (ii), we consider the closed dynamics of (4.3) in the Laplace domain, written as

$$\hat{X}(s) = \left(\prod_{k=n}^{1}(sI+L_k)^{-1}\right)\hat{U}_{ref}(s). \tag{4.9}$$

Since we design $\hat{U}_{ref}(s)$ to be stable, we can assume that the limit $\lim_{s\to 0}\hat{U}_{ref} = \hat{U}_{ref}(0)$ exists. To prove that the system achieves n^{th} order consensus, we want to show that $\lim_{t\to\infty} y(t) = \lim_{t\to\infty} C(s)\hat{X}(s) = 0$ for some transfer matrix $C(s)$ encoding the consensus states. We can simplify the problem by only considering impulse responses, which has the same transfer function as the initial value response where $\xi_n(0) = \hat{U}_{ref}(s)$. Therefore, without loss of generality, we can assume that $\hat{U}_{ref}(s) = 0$ and we consider an arbitrary initial condition $\xi(0) = [\xi_1(0)^T, \xi_2(0)^T, \dots, \xi_n(0)^T]^T$.

We can compute the solution of (4.3) by $e^{At}\xi(0) = Se^{Jt}S^{-1}\xi(0)$, where J is the Jordan normal form of A and S the corresponding invertible matrix with its columns being the eigenvectors of A. From (i) we know that all non-zero eigenvalues of A lie in the left half plane. We also know that the eigenvalue of each L_k at zero is simple, and therefore that these eigenvalues form a Jordan block of size n. To see this, let $\mathbf{e}_k$

denote the k^{th} 1-block vector (i.e., $\mathbf{e}_2 = [0_N, \mathbf{1}_N^T, 0_N, \ldots, 0_N]$). Then $\mathbf{e}_1$ is an eigenvector since $A\mathbf{e}_1 = 0$. For $k \in \{2, 3, \ldots, n\}$ we have $A\mathbf{e_k} = \mathbf{e_{k-1}}$ which implies that $A^k\mathbf{e}_k = 0$. This shows that there is a Jordan block of size n with an invariant subspace spanned by the vectors $\mathbf{e}_k$.

All other eigenvectors correspond to eigenvalues with a negative real part, meaning that these eigenvectors span an asymptotically converging invariant subspace. In other words, $\xi(t)$ converges to a solution in the span$(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$, and consequently $\lim_{t\to\infty} \xi_k(t) = \alpha_k(t)\mathbf{1}_N$. Furthermore, since $\dot{\xi}_k = -L_k\xi_k + \xi_{k+1} \to \xi_{k+1}$ as $t \to \infty$ for $k \in \{1, \ldots, n-1\}$, it follows that $\lim_{t\to\infty} x^{(k)}(t) = \alpha_{k+1}(t)\mathbf{1}_N$. This shows that the system achieves n^{th} order consensus.

It is interesting to note that the stability of the consensus for n^{th} order serial consensus can be reduced to verifying that the n first-order consensus systems $\dot{x} = -Lx$ achieve consensus. This, in turn, is equivalent to determining whether the graphs underlying each L_k contain a spanning tree. In the next section, we derive a relaxation of this stability condition, which we refer to as the scissor condition.

5 A relaxation of serial consensus and its scalability

In this section, we present a relaxation of the stability condition of the serial consensus algorithm. The serial consensus algorithm requires the underlying graph Laplacians L_k to contain a connected spanning tree to guarantee consensus. In [7], the graph Laplacians are taken as $L_k = b_k L$, where b_k are positive *real-valued* scalars. Now, we realize that these b_k can also be *complex-valued* under some conditions for the underlying graph Laplacian L. More specifically, we define the set $\mathcal{B}$ containing all b_k that achieve consensus of the closed loop dynamics. This set depends on the set $\mathcal{E}$, defined as the smallest conic region containing all eigenvalues of L. We refer to this theorem as the scissor condition. Since we want to have real-valued inputs u_k, we require that all b_k only appear as complex conjugated pairs.

First, we derive the scissor condition for networks where the agents have integrator dynamics of order any order n. Then, the scissor condition is analyzed for the specific case where $n = 2$. Here, we introduce the more intuitive feedback gains a_0 and a_1, and the set $\mathcal{B}$ that lives in $(b_1, b_2) \in \mathbb{C}^2$ is transformed to the set $\mathcal{K}$ that lives in $(a_0, a_1) \in \mathbb{R}^2$.

5.1 n^{th} order scissor condition

Following the approach in [7], we impose that the k^{th} graph Laplacian is given by $L_k = b_k L$ for some fixed graph Laplacian L. The closed-loop dynamics in the Laplace domain of the n^{th} order consensus system of (4.1) become

$$\left(\prod_{k=1}^{n}(sI + b_k L)\right)\hat{X}(s) = \hat{U}_{ref}(s). \tag{5.1}$$

In [6], it is proven that the closed-loop system is scalably stable for any $b_k \in \mathbb{R}_+$. For a more intuitive notation, we introduce the scalars a_k for all $k \in \{0, 1, \ldots, n-1\}$. These a_k are more intuitive as they have an interpretation. For example, in the second-order case, a_0 and a_1 can be interpreted as the stiffness and damping of the system respectively. We define a_k as the sum over all products of $n-k$ elements in the set $b \in \{b_1, b_2, \ldots, b_n\}$. For clarity, consider the controller for the case $n = 3$. Then the intuitive feedback gains become $a_0 = b_1 b_2 b_3$, $a_1 = b_1 b_2 + b_1 b_3 + b_2 b_3$ and $a_2 = b_1 + b_2 + b_3$. The control input corresponding to the closed-loop form in (5.1) is

$$u(t) = -a_0 L^n x(t) - a_1 L^{n-1} x^{(1)}(t) - \ldots - a_{n-1} L^2 x^{(n-2)}(t) - a_n L x^{n-1}(t), \tag{5.2}$$

which is of a very similar form to the control input of the conventional consensus algorithm, except for the exponents of the graph Laplacians L. The a_k gains are much more intuitive to tune than the b_k. To prove stability of the system we follow the approach of [4], and we consider the closed-loop state space

$$\dot{\xi} = \underbrace{\begin{bmatrix} 0 & I & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & I \\ -a_0 L^n & -a_1 L^{n-1} & \ldots & -a_{n-1} L^2 & -a_n L \end{bmatrix}}_{\mathcal{A}} \xi + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ u_{ref} \end{bmatrix} \tag{5.3}$$

with $\xi = \begin{bmatrix} x^{(0)} & x^{(1)} & \ldots & x^{(n-1)} \end{bmatrix}^T \in \mathbb{R}^{Nm}$. Now let T be an invertible $N \times N$ matrix such that $\Lambda = T^{-1} L T$ is of Jordan normal form. Introduce the block diagonal matrix $\mathbf{T} = \text{diag}(T, T, \ldots, T)$. By pre- and post-multiplying $\mathcal{A}$ with $\mathbf{T}$ we get

$$\mathbf{T}^{-1}\mathcal{A}\mathbf{T} = \underbrace{\begin{bmatrix} 0 & I & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & I \\ -a_0 \Lambda^n & -a_1 \Lambda^{n-1} & \ldots & -a_{n-1} \Lambda^2 & -a_n \Lambda \end{bmatrix}}_{\hat{\mathcal{A}}}, \tag{5.4}$$

where we use that $T^{-1}L^kT = \Lambda^k$. By pre- and post-multiplying with a suitable permutation matrix, the rows and columns of $\hat{\mathcal{A}}$ can be rearranged into the system matrix $\text{diag}(\hat{\mathcal{A}}_1, \dots, \hat{\mathcal{A}}_1)$ with

$$\hat{\mathcal{A}}_l = \begin{bmatrix} 0 & I & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & I \\ -a_0\Lambda_l^n & -a_1\Lambda_l^{n-1} & \dots & -a_{n-1}\Lambda_l^2 & -a_n\Lambda_l \end{bmatrix} \tag{5.5}$$

for all $l = 1, 2, \dots, k$. The eigenvalues of $\mathcal{A}$ are given by the eigenvalues of $\hat{\mathcal{A}}$, as writing the matrix in Jordan form does not change the eigenvalues. The eigenvalues of $\hat{\mathcal{A}}$ are given by the union of the eigenvalues of all $\hat{\mathcal{A}}_l$ due to the block diagonal structure. The characteristic polynomial of each $\hat{\mathcal{A}}_l$ is given by

$$p_l(s) = \sum_{i=0}^{n} a_i \lambda_l^{n-i} s^i, \quad \text{with } a_n = 1. \tag{5.6}$$

We denote the roots of this characteristic polynomial by $s'_l = \left\{ s'_{l,1}, s'_{l,2}, \dots, s'_{l,n} \right\}$, and we realize that these roots correspond the the eigenvalues of the closed-loop system. We introduce the new variable $\hat{s} := \frac{s}{\lambda_l}$. Substituting $s = \hat{s}\lambda_l$ into the characteristic polynomial yields a polynomial of $\hat{s}$ of the form

$$p_l(\hat{s}\lambda_l) = p_l(\hat{s}) = \sum_{i=0}^{n} a_i \lambda_l^n \hat{s}^i = \left[\sum_{i=0}^{n} a_i \hat{s}^i \right] \lambda_l^n, \quad \text{with } a_n = 1. \tag{5.7}$$

The roots of this polynomial are denoted by $\hat{s}'_l = \left\{ \hat{s}'_{l,1}, \hat{s}'_{l,2}, \dots, \hat{s}'_{l,n} \right\}$. We now realize that these roots of $p_l(\hat{s})$ are given by the $-b_k$ for all $k \in \{1, 2, \dots, n\}$. However, we are interested in the roots of the original characteristic polynomial $p_l(s)$. Using the relation $s = \hat{s}\lambda_l$, we get that

$$s'_l = \hat{s}'_l \lambda_l = -b_k \lambda_l, \quad \forall k \in \{1, 2, \dots, n\}. \tag{5.8}$$

We realize that these characteristic roots are equal to the eigenvalues (or poles) of $\mathcal{A}$. This means that these eigenvalues are equal to the product of the eigenvalues of the Laplacian L and the gains $-b_k$, which we can choose. We want to choose $-b_k$ in such a way that guarantees that all non-zero eigenvalues of $\mathcal{A}$ lie within the open left half plane to guarantee consensus. To this end, we note that all non-zero eigenvalues of the Laplacian L lie within the open right half plane, and that complex eigenvalues of L only appear in complex-conjugated pairs. This leads to the scissor theorem.

Theorem 1: scissor condition

The argument of the product of two complex numbers is equal to the sum of the arguments of the two complex numbers. This means that if we choose all $b_k \in \mathbb{R}_+$, the eigenvalues of L are flipped around the imaginary axis, meaning that they all lie in the closed left half plane, confirming the conclusion on stability in [6].

However, if we can define a conic bound on all eigenvalues of L (i.e. a bound on the arguments of the complex eigenvalues), we can also choose b_k as complex numbers within a certain conic bound. We denote the conic region containing the eigenvalues of L by $\mathcal{E}$, and define it as

$$\mathcal{E} = \left\{ e \in \mathbb{C}_+ \quad | \quad \left| \frac{\text{Im}(e)}{\text{Re}(e)} \right| \leq m_\lambda \right\}, \quad \text{where} \tag{5.9}$$

$$m_\lambda = \min \left\{ x \quad | \quad \left| \frac{\text{Im}(\lambda_l)}{\text{Re}(\lambda_l)} \right| \leq x \quad \forall\, l = 2, 3, \dots, N \right\} \tag{5.10}$$

denotes the minimum slope of the conic region which still ensures that it contains all eigenvalues. Imposing that the product of all $-b_k\lambda_l$ must lie within the open left half plane defines the maximum conic region $\mathcal{B}$ that contains all b_k that ensure consensus, given by

$$\mathcal{B} = \left\{ b \in \mathbb{C}_+ \quad | \quad \left| \frac{\mathrm{Im}(b)}{\mathrm{Re}(b)} \right| < \frac{1}{m_\lambda} \right\}. \tag{5.11}$$

We call this result the scissor theorem. The theorem defines the set $\mathcal{B}$ where we can choose our b_k for which consensus is guaranteed, which is a relaxation of the stability condition in [6]. This set depends on the set $\mathcal{E}$ containing all non-zero eigenvalues of L. The set $\mathcal{B}$ shrinks as the set $\mathcal{E}$ grows, similarly to a scissor mechanism. The scissor theorem is visualized in Figure 5.1, where the yellow and green region represent $\mathcal{E}$ and $-\mathcal{B}$ respectively.

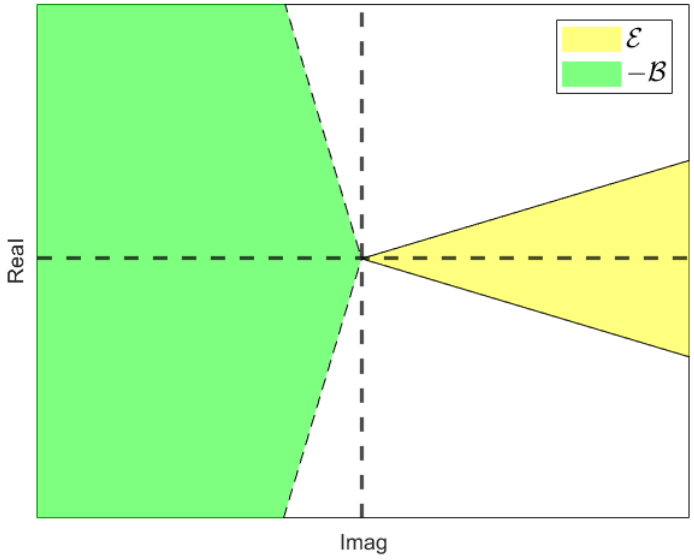

Figure 5.1: Visualization of scissor condition

It is interesting to note that the same result can also be realized by looking directly at the closed-loop dynamics in the Laplace domain in (4.1). The poles of the closed loop transfer function can be computed by solving

$$\det\left(\prod_{k=1}^{n} (sI + b_k L) \right) = 0 \tag{5.12}$$

for s. We realize that solving $\det(sI - L) = 0$ for s gives us the eigenvalues of L. Therefore, solving $\det(sI + b_k L) = 0$ for s gives us the eigenvalues of $-b_k L$. If we choose any $b_k \in \mathbb{R}_+$, the non-zero eigenvalues of $-b_k L$ therefore lie in the open left half plane, and the poles of the system are given by these eigenvalues. If we choose $b_k \in \mathbb{C}_+$, the scissor condition tells us if the system achieves consensus. Alternatively, this same result can also be realized by looking at the dynamics in closed loop form

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \vdots \\ \dot{\xi}_{n-1} \\ \dot{\xi}_n \end{bmatrix} = \begin{bmatrix} -b_1 L & I & \dots & 0 & 0 \\ 0 & -b_2 L & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -b_{n-1} L & I \\ 0 & 0 & \dots & 0 & -b_n L \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{n-1} \\ \xi_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ u_{ref} \end{bmatrix}, \tag{5.13}$$

which is a block-triangular form, meaning that the eigenvalues are given by the union of the eigenvalues of the blocks on the diagonal. As these diagonal blocks are $-b_k L$, we realize that the eigenvalues of the closed loop system are given by $-b_k \lambda_k$.

In many (if not all) physical systems, it is required to have real-valued control input. Therefore, we impose that complex b_k can only appear as complex conjugated pairs to ensure a real-valued control input. Imposing this ensures that each a_k and therefore also the control input is real-valued.

5.2 Second-order scissor condition

In this section, we apply the scissor condition to a second-order serial consensus system. Next to clarifying the condition, the second-order scissor condition also gives a lot of intuition. Again, we impose that the graph Laplacians are given by $L_k = b_k L$ for $k = 1, 2$. The closed-loop dynamics in the Laplace domain now become

$$(sI + b_1 L)(sI + b_2 L)\hat{X}(s) = \hat{U}(s). \tag{5.14}$$

For a more intuitive notation, we define the feedback gains $a_0 = b_1 b_2$ and $a_1 = b_1 + b_2$. The feedback gain a_0 on the position $x(t)$ can be interpreted as the stiffness of the system. The feedback gain a_1 on the velocity $\dot{x}(t)$ can be interpreted as the damping of the system. Its inverse relation is given by $b_{1,2} = \frac{1}{2}a_1 \pm \sqrt{\frac{1}{4}a_1^2 - a_0}$. The resulting control input can be written as

$$\begin{aligned} u(t) &= -\underbrace{b_1 b_2}_{a_0} L^2 x(t) - \underbrace{(b_1 + b_2)}_{a_1} L\dot{x}(t) \\ \rightarrow u(t) &= -a_0 L^2 x(t) - a_1 L\dot{x}(t) \end{aligned} \quad . \tag{5.15}$$

In order to prove stability of the closed-loop system, the characteristic second-order polynomial

$$P_l(s) = (s^2 + a_1\lambda_l s + a_0\lambda_l^2)^{r_l} = [p_l(s)]^{r_l} \tag{5.16}$$

is evaluated. The eigenvalues are given by the roots of $p_l(s)$. These roots are computed using the quadratic formula, and denoted by $s'_{l,(1,2)}$. We again realize that the roots of the characteristic polynomial are given by the product of the gains $-b_{1,2}$ and the eigenvalues λ_l.

$$s'_{l,1} = \lambda_l \underbrace{(-\frac{1}{2}a_1 \pm \sqrt{\frac{1}{4}a_1^2 - a_0})}_{-b_1} = -\lambda_l b_1(a_0, a_1) \quad \forall l = 1, 2, \ldots, N \tag{5.17}$$

$$s'_{l,2} = \lambda_l \underbrace{(-\frac{1}{2}a_1 \pm \sqrt{\frac{1}{4}a_1^2 - a_0})}_{-b_2} = -\lambda_l b_2(a_0, a_1) \quad \forall l = 1, 2, \ldots, N \tag{5.18}$$

Using the scissor theorem, we can now define the set $\mathcal{B}$ defining all possible b_1 and b_2 that guarantee consensus, based on the set $\mathcal{E}$ containing all eigenvalues of L. We want to relate this set $\mathcal{B}$ that lives in (b_1, b_2) to a set $\mathcal{K}$ that lives in (a_0, a_1) to gain some intuitive insight.

We know that if $a_1 \geq 2\sqrt{a_0}$, then $-b_{1,2}(a_0, a_1) \in \mathbb{R}_-$ lies on the negative real axis. When we assume that $a_1 < 2\sqrt{a_0}$, then $-b_{1,2}(a_0, a_1)$ lies in the open left half plane. To see which (a_0, a_1) correspond to a (b_1, b_2) within set $\mathcal{B}$, the slope of the complex numbers $b_{1,2}$ is computed in terms of $a_{0,1}$ to see when it exceeds $\frac{1}{m_\lambda}$.

$$\begin{aligned} \left|\frac{\mathrm{Im}(b_{1,2})}{\mathrm{Re}(b_{1,2})}\right| = \left|\frac{\sqrt{a_0 - \frac{1}{4}a_1^2}}{-\frac{1}{2}a_1}\right| &= \frac{\sqrt{a_0 - \frac{1}{4}a_1^2}}{\frac{1}{2}a_1} < \frac{1}{m_\lambda} \\ \sqrt{a_0 - \frac{1}{4}a_1^2} &< \frac{1}{2}\frac{1}{m_\lambda}a_1 \\ a_0 - \frac{1}{4}a_1^2 &< \frac{1}{4}\frac{1}{m_\lambda^2}a_1^2 \\ \frac{1}{4}\frac{1}{m_\lambda^2}a_1^2 + \frac{1}{4}a_1^2 - a_0 &> 0 \\ \frac{1}{4}(\frac{1}{m_\lambda^2} + 1)a_1^2 - a_0 &> 0 \\ (\frac{1}{m_\lambda^2} + 1)a_1^2 &> 4a_0 \\ a_1 &> \frac{2}{\sqrt{(\frac{1}{m_\lambda})^2 + 1}}\sqrt{a_0} \end{aligned} \tag{5.19}$$

Theorem 2: second-order scissor condition

This gives a condition on a_0 and a_1 that ensures that the corresponding b_1 and b_2 lie within set $\mathcal{B}$ (and hence ensures consensus). The set defining all (a_0, a_1) that yield (b_1, b_2) in set $\mathcal{B}$ is denoted by $\mathcal{K}$, and defined as

$$\mathcal{K} = \left\{ (a_0, a_1) \in \mathbb{R}^2 \quad | \quad a_1 > \frac{2}{\sqrt{(\frac{1}{m_\lambda})^2 + 1}} \sqrt{a_0} \right\}. \tag{5.20}$$

Example: directed ring graph with $N = 5$ agents with second order integrator dynamics

As an example, consider a directed ring graph with $N = 5$ agents (see Figure 7.13), with all edge weights equal to one. The four non-zero eigenvalues graph Laplacian L are visualized in Figure 5.2, along with the conic set $\mathcal{E}$ containing all eigenvalues and its corresponding set $-\mathcal{B}$. Set $\mathcal{K}$ is visualized in Figure 5.3. As the conic set $\mathcal{E}$ grows, the conic set $\mathcal{B}$ shrinks and the border of set $\mathcal{K}$ asymptotically converges to $a_1 = 2\sqrt{a_0}$.

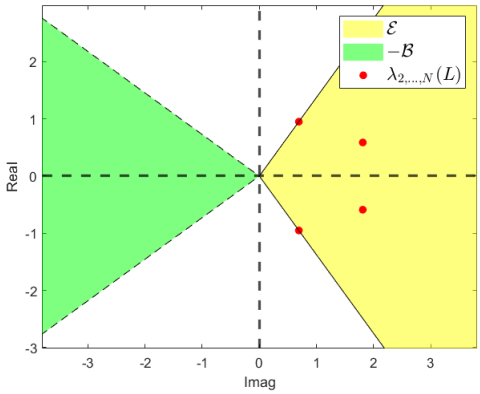

Figure 5.2: Visualization of scissor condition with 5 agents in a directed circular network.

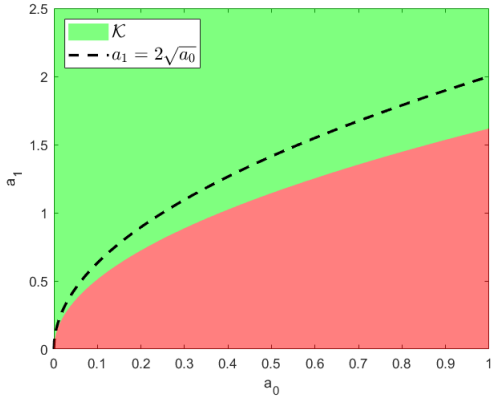

Figure 5.3: Visualization of scissor condition in (a_0, a_1) with 5 agents in a directed circular network.

In this section, we have presented a relaxation of the stability condition of the serial consensus protocol. In the next section, we investigate the implications of this relaxation on the string stability of serial consensus.

6 String instability of relaxed serial consensus

In this section, we investigate whether the relaxation of the scalable stability condition (i.e., the scissor condition) also guarantees string stability.

Apart from to scalable stability, the serial consensus algorithm as presented in [7] also guarantees string stability for second order agent dynamics, which is a strong notion of scalable performance. More specifically, the maximum relative positions and deviations in velocity can be bounded as in (2.12) when we impose that $L_k = b_k L$ for $k = 1, 2$ where b_1 and b_2 are *real-valued*. We show this by summarizing the result of [7]. Moreover, we also show that this result cannot be generalized to *complex-valued* b_1 and b_2. In addition, we express the string stability guarantee in terms of the more intuitive feedback gains a_0 and a_1 to provide some intuition.

The response of the closed loop dynamics in (5.13) for $n = 2$ can be solved analytically. After some algebra and applying a coordinate transformation, the response of the relative position $e_p(t)$ and velocity deviation $e_v(t)$ (see (2.11) for the definitions of e_p and e_v) can be written as

$$\begin{aligned} e_p(t) &= \frac{b_1 e^{-b_2 Lt} - b_2 e^{-b_1 Lt}}{b_1 - b_2} e_p(0) + \frac{e^{-b_2 Lt} - e^{-b_1 Lt}}{b_1 - b_2} e_v(0), \\ e_v(t) &= \frac{b_1 b_2 (e^{-b_1 Lt} - e^{-b_2 Lt})}{b_1 - b_2} e_p(0) + \frac{b_1 e^{-b_1 Lt} - b_2 e^{-b_2 Lt}}{b_1 - b_2} e_v(0). \end{aligned} \tag{6.1}$$

Taking the infinity norm $||\cdot||_\infty$ on both sides and applying the triangle inequality yields the following two bounds on $e_p(t)$ and $e_v(t)$.

$$\begin{aligned} ||e_p(t)||_\infty &\leq \left\| \frac{b_1 e^{-b_2 Lt} - b_2 e^{-b_1 Lt}}{b_1 - b_2} e_p(0) \right\|_\infty + \left\| \frac{e^{-b_2 Lt} - e^{-b_1 Lt}}{b_1 - b_2} e_v(0) \right\|_\infty \\ &\leq \frac{||b_1 e^{-b_2 Lt} - b_2 e^{-b_1 Lt}||_\infty}{|b_1 - b_2|} ||e_p(0)||_\infty + \frac{||e^{-b_2 Lt} - e^{-b_1 Lt}||_\infty}{|b_1 - b_2|} ||e_v(0)||_\infty \\ &\leq \frac{b_1 \cdot ||e^{-b_2 Lt}||_\infty + b_2 \cdot ||e^{-b_1 Lt}||_\infty}{|b_1 - b_2|} ||e_p(0)||_\infty + \frac{||e^{-b_2 Lt}||_\infty + ||e^{-b_1 Lt}||_\infty}{|b_1 - b_2|} ||e_v(0)||_\infty \\ ||e_v(t)||_\infty &\leq \left\| \frac{b_1 b_2 (e^{-b_1 Lt} - e^{-b_2 Lt})}{b_1 - b_2} e_p(0) \right\|_\infty + \left\| \frac{b_1 e^{-b_1 Lt} - b_2 e^{-b_2 Lt}}{b_1 - b_2} e_v(0) \right\|_\infty \\ &\leq \frac{||b_1 b_2 (e^{-b_1 Lt} - e^{-b_2 Lt})||_\infty}{|b_1 - b_2|} ||e_p(0)||_\infty + \frac{||b_1 e^{-b_1 Lt} - b_2 e^{-b_2 Lt}||_\infty}{|b_1 - b_2|} ||e_v(0)||_\infty \\ &\leq \frac{b_1 b_2 (||e^{-b_1 Lt}||_\infty + ||e^{-b_2 Lt}||_\infty)}{|b_1 - b_2|} ||e_p(0)||_\infty + \frac{b_1 \cdot ||e^{-b_1 Lt}||_\infty + b_2 \cdot ||e^{-b_2 Lt}||_\infty}{|b_1 - b_2|} ||e_v(0)||_\infty \end{aligned} \tag{6.2}$$

We now realize that $-b_{1,2} Lt$ is a Metzler matrix for all $b_1, b_2 \in \mathbb{R}_+$ and $t \geq 0$. A Metzler matrix is any matrix M that satisfies $M = (m_{ij})$ where $m_{ij} \geq 0 \; \forall i \neq j$ and $a_{ij} \in \mathbb{R}$ (see [12]). In other words, a Metzler matrix M requires all of its non-diagonal elements to be non-negative and real valued. A Metzler matrix has the property that $e^M \geq 0$ (see again [12]). Therefore, it holds that $e^{-p_{1,2} Lt} \geq 0$ for all $t \geq 0$. This allows the simplification of the matrix norms $||e^{-b_{1,2} Lt}||_\infty$ by realizing that

$$\begin{aligned} ||e^{-b_{1,2} Lt}||_\infty &= \max_i \left\{ |e^{-b_{1,2} Lt}| \mathbf{1} \right\} \\ &= \max_i \left\{ e^{-b_{1,2} Lt} \mathbf{1} \right\} \\ &= \max_i \left\{ \left(I + \sum_{k=1}^{\infty} \frac{1}{k!} (-b_{1,2})^k L^k t^k \right) \mathbf{1} \right\} \\ &= \max_i I \\ &= 1. \end{aligned} \tag{6.3}$$

It is important to note that a Metzler matrix requires that all its entries are real-valued. Therefore, the result on string stability does not hold for complex valued b_1 and b_2. However, it does not mean that it is string unstable, as there could be another derivation that does hold for complex valued $b_{1,2}$. The identity

$||e^{-b_{1,2}Lt}||_\infty = 1$ for real-valued b_1 and b_2 implies that the bounds on the relative position and velocity deviation can be written as

$$\begin{aligned} ||e_p(t)||_\infty &\leq \frac{b_1+b_2}{|b_1-b_2|}||e_p(0)||_\infty + \frac{2}{|b_1-b_2|}||e_v(0)||_\infty, \\ ||e_v(t)||_\infty &\leq \frac{2b_1b_2}{|b_1-b_2|}||e_p(0)||_\infty + \frac{b_1+b_2}{|b_1-b_2|}||e_v(0)||_\infty. \end{aligned} \tag{6.4}$$

Finally, by realizing that $||e_p||_\infty, ||e_v||_\infty \leq \left\| \begin{bmatrix} e_p \\ e_v \end{bmatrix} \right\|_\infty = \max\{||e_p||_\infty, ||e_v||_\infty\}$, we find the performance criterion for string stability

$$\sup_{t\geq 0} \left\| \begin{bmatrix} e_p(t) \\ e_v(t) \end{bmatrix} \right\|_\infty \leq \alpha \left\| \begin{bmatrix} e_p(0) \\ e_v(0) \end{bmatrix} \right\|_\infty, \text{ with } \alpha = \frac{1}{|b_1-b_2|}(b_1+b_2+2\max\{1, b_1b_2\}). \tag{6.5}$$

To provide some understanding into the string stability, the bound is expressed in terms of the intuitive feedback gains a_0 and a_1 using the relation $a_0 = b_1b_2$ and $a_1 = b_1 + b_2$. Writing (6.4) in terms of these gains and in a matrix notation gives

$$\begin{bmatrix} \|e_p(t)\|_\infty \\ \|e_v(t)\|_\infty \end{bmatrix} = \frac{1}{\sqrt{a_1^2-4a_0}} \begin{bmatrix} a_1 & 2 \\ 2a_0 & a_1 \end{bmatrix} \begin{bmatrix} \|e_p(0)\|_\infty \\ \|e_v(0)\|_\infty \end{bmatrix}. \tag{6.6}$$

Similarly, α can be written in terms of a_0 and a_1, see (6.7). This expression clearly shows that $a_1 > 2\sqrt{a_0}$ is required for a real-valued and finite α, which corresponds to choosing real-valued and $b_{1,2}$ with $b_1 \neq b_2$.

$$\alpha = \frac{1}{\sqrt{a_1^2-4a_0}}(a_1 + 2\max\{1, a_0\}) \tag{6.7}$$

We want to get some insight into the string stability guarantee as a function of the intuitive feedback gains a_0 and a_1. To this end, Figure 6.1 visualizes the value of α in the range $\alpha \in [0, 10]$ as a function of a_0 and a_1 to see how it evolves. This shows that α grows very large close to $a_1 = 2\sqrt{a_0}$. Choosing an a_0 and a_1 far away from this line gives a more strict guarantee on the relative position and velocity deviation.

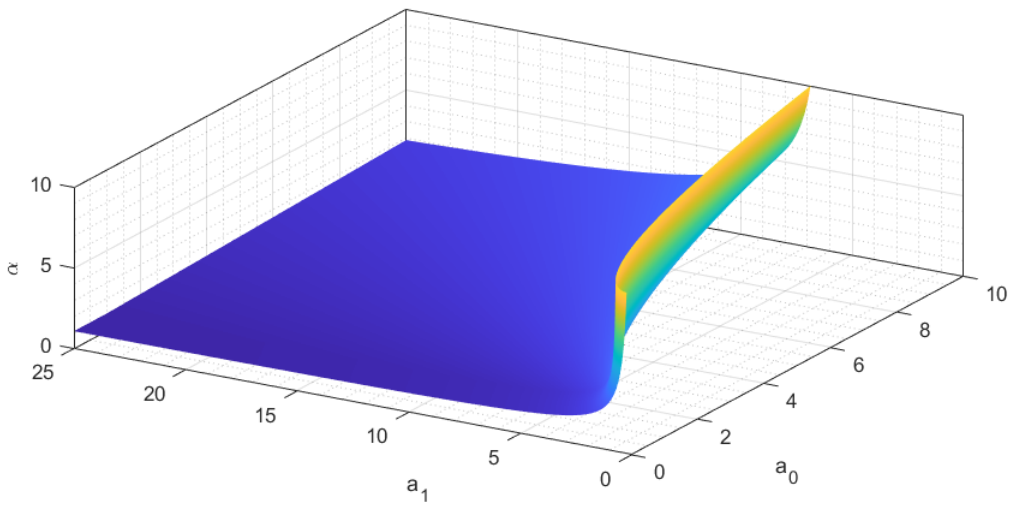

Figure 6.1: Visualization of α (in the range $\alpha \in [0, 10]$) as a function of a_0 and a_1

We have now presented theoretical results regarding the string stability and scalable stability of both the conventional and serial consensus protocol. In the next section, we elaborate on the design of the physical setup that we use to experimentally verify these theoretical results.

7 Experimentation Setup

In this section, we elaborate on the experimentation setup that we use to experimentally verify the conventional and serial consensus protocol. To compare serial consensus to conventional consensus, both consensus protocols are implemented on robotic vehicles in the same experimentation setup. The setup mimics vehicle platooning, as the agents move over a path along which they have to achieve consensus in cruising velocity while maintaining a constant inter-vehicular distance. For practical reasons, the path is chosen to be a circle with radius r^*. The vehicles, or agents, are represented by omnibots[1] (see Figure 7.3). The setup should be able to model each agent as an n^{th} order integrator. In this project, we focus on experiments where the vehicles have second-order dynamics, meaning that the input is its acceleration along the circle and the states are its velocity and travelled distance along the circular path. This requires additional low-level control, as we should make sure that each agent stays on the circle. We want to decouple the control, in the sense that any deviation of an omnibot from the circle does not influence its dynamics along the path. Figure 7.5 shows the omnibots travelling along the path. A playlist containing video recordings of several experiments can be found in [13].

The remainder of this section is organized as follows. First, we explain how the experimentation setup mimics vehicle platooning. Then, the omnibot kinematics are presented, and we explain how the control of each omnibot is decoupled by applying a change of coordinates. Next, it is explained how the control input is integrated numerically to mimic the behaviour of an n^{th} order integrator. Furthermore, the sensor system is elaborated. After this, the control structure design is verified by presenting some experimental results showing the lateral and rotational tracking errors. Finally, the graph structures that we use in the experiments are introduced.

7.1 Vehicle platooning

In the experimental setup, the agents (which are omnibots) have to achieve consensus in their shifted position $\hat{x}$ and their velocity. This can be interpreted as vehicle platooning, see Chapter 3. For our experimentation, there are generally two platooning regimes of interest and relevant for vehicle platooning. To verify string stability, the agents communicate according to a linear graph. Each vehicle has access to the relative information of its predecessor, and the first vehicle is a virtual leader agent, meaning that it is not a physical vehicle but its trajectory is computed in software. To verify the scalable stability, the agents communicate according to a circular graph. Again, each vehicle looks its predecessor, only now there is no leader agent. It is important to note that the algorithm tries to achieve consensus of the vehicles' travelled distance along a path and its derivatives. This means that the consensus algorithm is only applied in one dimension; the travelled distance along the path (see Figure 7.1). Each vehicle is referred to as an agent, and each agent is modelled as the n^{th} order integrator in (2.5). The travelled distance along a path is denoted by s. For practical reasons, the path is chosen to be a circle with radius r^* (see Figure 7.2). This allows the agents to travel freely along the path without being constrained by a limited length.

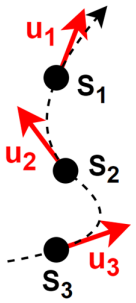

Figure 7.1: Visualization of travelled distance of 3 agents along any arbitrary path s

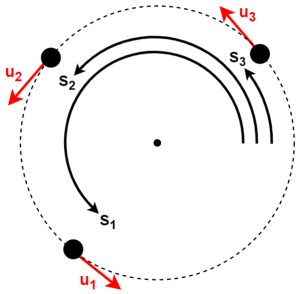

Figure 7.2: Visualization of travelled distance of 3 agents along a circular path

As the consensus algorithm is applied to the travelled distance along the path, additional control is required to ensure that each agent stays on its path. Ideally, these controllers are decoupled in the sense that any deviation from the path does not affect the dynamics along the travelled distance along the path. Therefore, a change of coordinates is applied, decoupling the lateral control, rotational control and longitudinal control. The lateral controller is responsible for minimizing the deviation of each agent to the path. The rotational

[1] Omnibots are mobile robots equipped with three omniwheels, allowing for decoupled control of its velocity.

controller is responsible for the orientation of each agent. The longitudinal controller steers the travelled distance along the path of each agent, e.g. the consensus algorithm. This approach of separating the lateral and longitudinal control is very common in vehicle platooning applications.

7.2 Omnibot kinematics

In the vehicle platooning experiments, the agents are represented by omnibots. An omnibot is a mobile robot equipped with three omniwheels. Omniwheels are wheels with small discs around the circumference (see Figure 7.3) which are perpendicular to the turning direction. By virtue of this design, the wheel can be driven with full force, but without the no-slip constraint in the lateral direction, meaning that it can slide sideways. Figure 7.4 shows one of the omnibots that are used in the experiment. Figure 7.5 depicts the top view of the experimentation setup, showing a platoon of omnibots driving over the circular path.

Figure 7.3: Picture of an omniwheel. The arrows show it does not have the holonomic constraint in lateral direction, meaning that it can slide sideways freely.

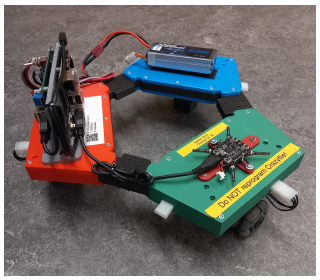

Figure 7.4: Picture of one of the omnibots that are used in the experiments.

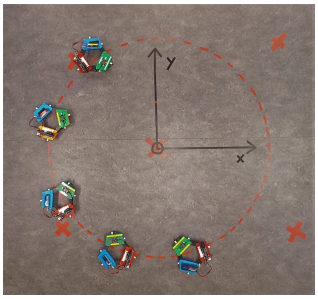

Figure 7.5: Top view of the experimentation setup. We see a platoon of omnibots driving over the circular path.

The control inputs of each omnibot are the velocities of its three omniwheels. The omniwheel velocities of agent i are denoted by $v_{\text{wheels}} = (v_{1,i}, v_{2,i}, v_{3,i})$ for omniwheel $(1, 2, 3)$ respectively. Figure 7.6 shows a schematic representation of an omnibot. The length of the arms is denoted by R. The vector (x_i, y_i, θ_i) represents the Cartesian position of omnibot i.

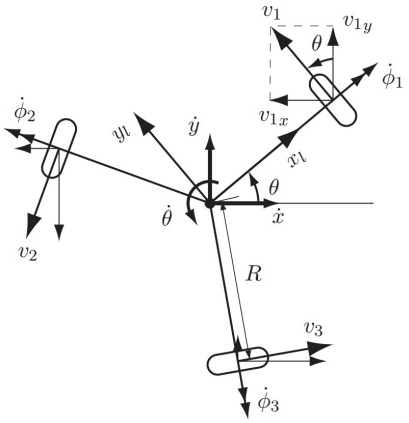

Figure 7.6: Schematic representation of an omnibot

The Cartesian velocities are denoted by $v_{\text{Cartesian}} = (v_{x,i}, v_{y,i}, \omega_i)$ and are related to the individual omniwheels velocities $(v_{1,i}, v_{2,i}, v_{3,i})$ through matrix $M(\theta_i)$, where θ_i represents the orientation of agent i in the Cartesian frame:

$$\begin{bmatrix} v_{x,i} \\ v_{y,i} \\ \omega_i \end{bmatrix} = M(\theta_i) \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ v_{3,i} \end{bmatrix}, \quad M(\theta_i) = \begin{bmatrix} -\frac{2}{3}\sin(\theta_i) & -\frac{2}{3}\cos(\theta_i + \frac{1}{6}\pi) & \frac{2}{3}\sin(\theta_i + \frac{1}{3}\pi) \\ \frac{2}{3}\cos(\theta_i) & -\frac{2}{3}\cos(\theta_i - \frac{1}{3}\pi) & -\frac{2}{3}\cos(\theta_i + \frac{1}{3}\pi) \\ \frac{1}{3R} & \frac{1}{3R} & \frac{1}{3R} \end{bmatrix}. \tag{7.1}$$

The determinant of $M(\theta_i)$ is $\det(M(\theta_i)) = \frac{0.3849}{R} \neq 0$ for all $R > 0$. Therefore, matrix $M(\theta_i)$ is invertible and $M^{-1}(\theta_i)$ can be used to compute required omniwheel velocities to obtain a desired Cartesian velocity. The required omniwheel velocities $(v_{1,i}, v_{2,i}, v_{3,i})$ to obtain the Cartesian velocities $(v_{x,i}, v_{y,i}, \omega_i)$ are given by

$$\begin{bmatrix} v_{1,i} \\ v_{2,i} \\ v_{3,i} \end{bmatrix} = M^{-1}(\theta_i) \begin{bmatrix} v_{x,i} \\ v_{y,i} \\ \omega_i \end{bmatrix}, \quad \text{where} \quad M^{-1}(\theta_i) = \begin{bmatrix} -\sin(\theta_i) & \cos(\theta_i) & R \\ -\sin(\theta_i + \frac{2}{3}\pi) & \cos(\theta_i + \frac{2}{3}\pi) & R \\ -\sin(\theta_i + \frac{4}{3}\pi) & \cos(\theta_i + \frac{4}{3}\pi) & R \end{bmatrix}. \tag{7.2}$$

7.3 Decoupling control by change of coordinates

Matrix $M^{-1}(\theta_i)$ gives the omniwheel velocities required to obtain a desired Cartesian velocity. This allows to assume that the control inputs of each agent are the Cartesian velocities $(v_{x,i}, v_{y,i}, \omega_i)$. The omnibot dynamics are now given by

$$\begin{cases} \dot{x}_i = v_{x,i} \\ \dot{y}_i = v_{y,i} \\ \dot{\theta}_i = \omega_i \end{cases}. \tag{7.3}$$

As the path to be followed by the agents is a circle, a polar coordinate frame (s_i, r_i, θ_i) is introduced, visualized in Figure 7.7. The scalar quantity $s_i = r^* \phi_i \in [0, 2\pi r^*]$ represents the travelled distance of agent i along the circle where $\phi_i = \text{atan2}(y_i, x_i) \mod 2\pi$ is the corresponding travelled angle. The radial coordinate $r_i = \sqrt{x_i^2 + y_i^2}$ represents the distance from agent i to the center of the circle. The polar velocities in tangential direction $v_{t,i}$ and radial direction $v_{r,i}$ are introduced, and we say that $v_{\text{polar}} = (v_{t,i}, v_{r,i}, \omega_i)$. These polar velocities are related to the Cartesian velocities through the rotation matrix $R(s_i)$ according to

$$\begin{bmatrix} v_{x,i} \\ v_{y,i} \\ \omega_i \end{bmatrix} = R(s_i) \begin{bmatrix} v_{r,i} \\ v_{t,i} \\ \omega_i \end{bmatrix}, \quad R(s_i) = \begin{bmatrix} \cos(\frac{s_i}{r^*}) & -\sin(\frac{s_i}{r^*}) & 0 \\ \sin(\frac{s_i}{r^*}) & \cos(\frac{s_i}{r^*}) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.4}$$

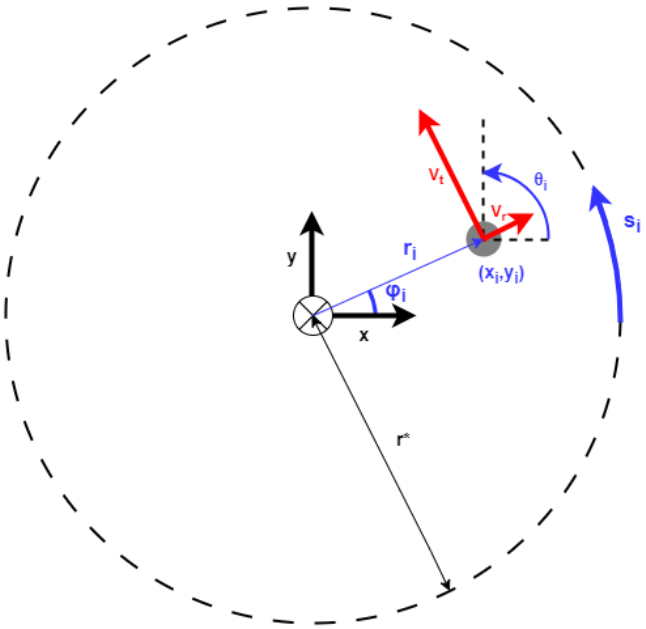

Figure 7.7: Schematic representation of omnibot with relevant coordinate definitions

$$\begin{cases} \dot{s}_i = \dfrac{r_i}{r*} v_{t,i} \\ \dot{r}_i = v_{r,i} \\ \dot{\theta}_i = \omega_i \end{cases}. \tag{7.5}$$

This change of coordinates from Cartesian to polar coordinates allows to write the dynamics of each agent in the form of (7.5). These dynamics are almost completely decoupled, except for the factor $\frac{r_i}{r*}$. This term represents the fact that an agent moves slower along the arc of a circle when it is outside of this circle, as it has a bigger radius and therefore has to travel a longer distance (and vice verse when it is inside the circle). To account for this, the new input $\hat{v}_{t,i} = \frac{r_i}{r^*} v_{t,i}$ is introduced to obtain input-output linearization (I/O-linearization). This results in the dynamics

$$\begin{cases} \dot{s}_i = \hat{v}_{t,i} \\ \dot{r}_i = v_{r,i} \\ \dot{\theta}_i = \omega_i \end{cases} \tag{7.6}$$

which are completely linear and decoupled. The new control inputs are $(\hat{v}_{t,i}, v_{r,i}, \omega_i)$. The lateral (or radial) velocity $v_{r,i}$ is used to keep each agent on the circle. The rotational velocity ω_i is used to control the orientation of each agent. The tangential velocity $\hat{v}_{t,i}$ is used in the consensus protocol.

Figure 7.8 shows the resulting control structure that is used on each agent in the form of a block diagram. It shows the coordinate transformation from the Cartesian coordinates to polar coordinates. A separate algorithm for rotational control, lateral control and longitudinal control compute the polar velocities. These velocities are transformed back to Cartesian velocities, which are subsequently transformed to the required omniwheel velocities of the omnibot. It also shows where the sensor noise is introduced into the control loop. The noise on the measured x, y and θ is denoted by n_x, n_y and n_θ respectively.

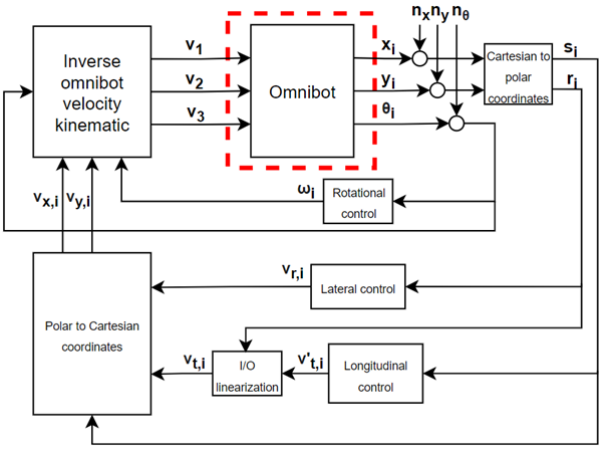

Figure 7.8: Control structure of each omnibot as a block diagram

It is important to note that this control structure models each agent as a first order integrator, as the control input of each omnibot is a velocity. In order to have higher-order agent dynamics, the control input should be acceleration, jerk, et cetera. As this would require to completely alter the software of the omnibots, the control input is integrated numerically $n-1$ times in software to obtain the velocity, which is then applied by the omnibot. This results in a system that is very similar to an n^{th} order integrator, but not equivalent. The differences are discussed in Section 7.4.

7.3.1 Lateral control

The state space in (7.6) shows that the lateral dynamics of each agent are given by $\dot{r}_i = v_{r,i}$. As this is a single integrator system, a proportional feedback controller of the form $v_{r,i} = \gamma_i(r_i - r^*)$ is sufficient to stabilize the system and drive r_i asymptotically to r^*. Here, $\gamma_i > 0$ is a positive feedback gain. In continuous time, any feedback gain $\gamma_i > 0$ suffices. However, the controller is applied in discrete-time with a finite sampling frequency. In addition, the sensors introduce sensor noise. Choosing α_i too low results in poor reference tracking, but choosing α_i too high results in noise amplification or even instability due to the finite sampling time. Therefore, the feedback gain γ_i is tuned in the frequency domain using conventional loopshaping techniques.

7.3.2 Rotational control

The state space in (7.6) shows that the rotational dynamics of each agent are given by $\dot{\theta}_i = \omega_i$. This is again a single integrator system, meaning that a proportional feedback controller of the form $\omega_i = -\beta_i(\theta_i - \theta_{i,ref})$ is stabilizing and asymptotically drives the system to the reference orientation $\theta_{i,ref}$. Here, $\beta_i > 0$ is a positive feedback gain. Similarly to the lateral control algorithm, the feedback gain $\beta_i > 0$ is tuned in the frequency domain using conventional loopshaping techniques to account for the finite sampling time and measurement noise.

Due to the fully decoupled dynamics in (7.6), the rotational control does not have any influence on the dynamics of s_i and r_i. This would make the choice for the reference orientation θ_{ref} arbitrary. However, the servos that actuate the omniwheels have a maximum velocity, after which they saturate. This means that if one of the omniwheels reaches its maximum velocity, the desired control velocities $(\hat{v}_{t,i}, v_{r,i}, \omega_i)$ are not obtained. Therefore, it is important that we maximize the tangential velocity at which this happens.

The distribution of velocities among each omniwheel depends on the orientation of the omnibot along the circle. Therefore, it is important to choose θ_{ref} with care. To this end, the optimal orientation of an omnibot along a circle is determined that maximizes the tangential velocity at which the omniwheels saturate. The orientation of the omnibot along the circle is denoted by $\delta := \theta - \phi$, and the optimal orientation depends on the system geometry $\beta := \frac{R}{r*}$. Both the optimal orientation and the corresponding maximum tangential velocity (normalized w.r.t. the maximum omniwheel velocity c) are derived analytically in Appendix A. The results are presented below:

$$\delta^*(\beta) = \begin{cases} \arccos(2\beta) & \text{if} \quad \beta \leq \frac{1}{4} \\ \frac{1}{3}\pi & \text{if} \quad \beta > \frac{1}{4} \end{cases}, \tag{7.7}$$

$$\frac{v_t^*}{c}(\beta) = \begin{cases} \frac{2\sqrt{3}}{3\sqrt{1-4\beta^2}} & \text{if} \quad \beta \leq \frac{1}{4} \\ \frac{2}{2\beta+1} & \text{if} \quad \beta > \frac{1}{4} \end{cases}. \tag{7.8}$$

The result is visualized in Figure 7.9, showing both the optimal orientation δ^* and normalized maximum velocity $\frac{v_t^*}{c}$ as a function of β. The highest tangential velocity can be reached at $\beta = \frac{1}{4}$, i.e., when the radius of the circle is four times the length of the omnibot arms.

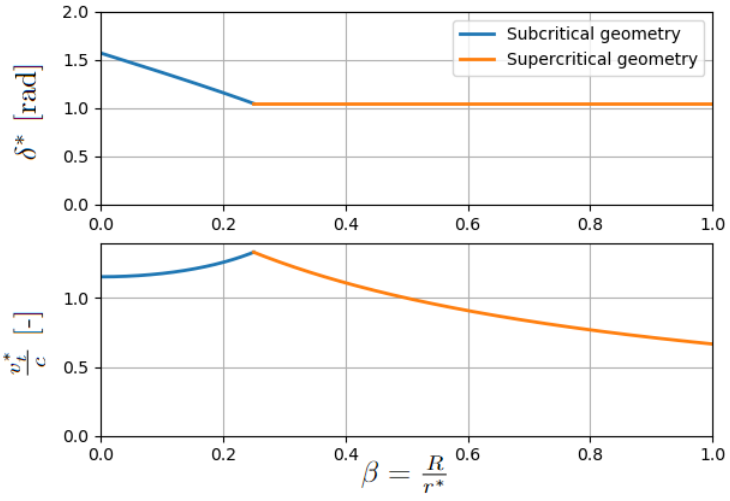

Figure 7.9: Optimal orientation and corresponding tangential velocity as a function of the system geometry $\beta = \frac{R}{r^*}$

7.3.3 Reference offset between agents

In the experimental setup, a reference offset in the position is introduced to avoid collisions between the omnibots when trying to achieve consensus. To this end, we define $\hat{s}_i(t) = s_i(t) - (i-1)\Delta s^*$. This is the same transformation as in (2.2), meaning we can still aply the consensus algorithm. Similarly, we do the same for the travelled angle as $\hat{\phi}_i(t) = \phi_i(i) - (i-1)\frac{\Delta s^*}{r^*}$. The consensus algorithms use the transformed state $\hat{s}$.

7.4 Artificial integration

The longitudinal dynamics of each agent i are $\dot{s}_i = \hat{v}_{t,i}$. This is a first order integrator systems, whereas the agents should be n^{th} order integrators for any $n \in \mathbb{N}_+$. To also perform experiments with higher-order dynamics, the input signal u_i is (numerically) integrated $n-1$ times to obtain $\hat{v}_{t,i}$ before applying the velocity to the omnibot. The resulting system is very similar to an n^{th} order integrator, but they are not equivalent. To see this, consider the second-order integrator. As the controller is working with a finite sampling frequency, the discrete-time state space of the second-order integrator

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau(k) + \frac{1}{2}u(k)\tau(k)^2 \\ \hat{v}_t(k+1) &= \hat{v}_t(k) + u(k)\tau(k) \end{aligned} \tag{7.9}$$

is considered, where $\tau(k)$ denotes the sampling time between iteration k and $k+1$. However, this is not what is happening on the omnibot. Here, the control input $u(k)$ is integrated to obtain the velocity $\hat{v}_t(k+1)$. This velocity is then used to compute the next position $s(k+2)$. The corresponding discrete-time state space is given by

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau(k) \\ \hat{v}_t(k+1) &= \hat{v}_t(k) + u(k)\tau(k). \end{aligned} \tag{7.10}$$

This clearly shows that the update step for $s(k+1)$ does not directly take the control input $u(k)$ into account. Instead, it is only indirectly influenced by $u(k-1)$ through $v_t(k)$. The error in the update step $s(k+1)$ of the physical system with respect to a pure n^{th} order integrator is denoted by $E_n(k)$. For this second-order example, the error is $E_2(k) = \frac{1}{2}u(k)\tau(k)^2$. This shows that the error decreases for small control inputs $u(k)$ and a small sampling time $\tau(k)$. In other words, we can assume that the omnibot dynamics can be modelled as a pure integrator for sufficiently small control inputs and a high sampling frequency. The derivation of the error $E_n(k)$ for any n is presented in Appendix D.

The omniwheels driving the omnibots each have a maximum velocity (in both directions), denoted by c. The results in (7.7) and (7.8) allow us to impose the optimal orientation of the omnibots with respect to the circle, meaning that each omnibot has a maximum tangential velocity, denoted by $\hat{v}_{\mathrm{t,max}}$. This is taken into account when numerically integrating the control input by enforcing that

$$\begin{aligned} s(k+1) &= s(k) + \hat{v}_t(k)\tau(k) \\ \hat{v}_t(k+1) &= \min[\max[\hat{v}_t(k) + u(k)\tau(k), \hat{v}_{\mathrm{t,max}}], -\hat{v}_{\mathrm{t,max}}]. \end{aligned} \tag{7.11}$$

7.5 State observation

The omnibots are equipped with a deck that emits a signal. This signal is detected by four lighthouses that compute the Cartesian position of each omnibot, denoted by (x_i, y_i, θ_i). These Cartesian coordinates are transformed to the polar coordinates (s_i, r_i, θ_i), using and $s_i = r^* \operatorname{atan2}(y_i, x_i) \mod 2\pi$ and $r_i = \sqrt{x_i^2 + y_i^2}$. It is important to note that this gives a travelled distance s_i in the range $[0, 2\pi r^*]$, i.e. it 'resets' after a full rotation. Therefore, the relative position between agents cannot be computed by simply taking the difference. Instead, this relative position should be wrapped to a range. We choose to wrap the relative position between agents to the range $[-\pi, \pi]$, using the function wrap().

$$\Delta s_{ij} = \mathrm{wrap}(\phi_i - \phi_j) = \begin{cases} r^*(\phi_i - \phi_j + 2\pi) & \text{if } \phi_i - \phi_j < -\pi \\ r^*(\phi_i - \phi_j - 2\pi) & \text{if } \phi_i - \phi_j > \pi \\ r^*(\phi_i - \phi_j) & \text{else} \end{cases} \tag{7.12}$$

The control input signal is integrated numerically to obtain the velocity. We assume that these numerical signals are sufficiently similar to the physical higher order states of each agent (velocity, acceleration, jerk, etc...). This eliminates the need for a sensor system for these higher order states.

We also wish to remark that, though the lighthouse system gives absolute measurements, the controllers only use the relative positions and velocities. This emulates relative measurements that could have been obtained with ,e.g., lidar measurements had that equipment been available.

7.6 Verification

An experiment is performed in order to verify the control structure design. The experiment is performed with five omnibots, and each omnibot starts with a random initial position and orientation and has a constant tangential velocity $\hat{v}_t = 0.1$ [m/s]. The radius of the circular reference path is $r^* = 1$ [m]. Both the lateral and rotational feedback gains are $\beta = \gamma = 0.5$. The optimal orientation of the omnibots with respect to the circle is $\delta^* = 1.25$ [rad], meaning that the reference is $\theta_{ref}(t) = \phi(t) + 1.25$ [rad]. The control algorithm operates at a sampling frequency of 2 [Hz]. Figure 7.10 visualizes the resulting trajectories in the Cartesian plane, showing that each omnibot indeed converges towards the circle after which it follows the path.

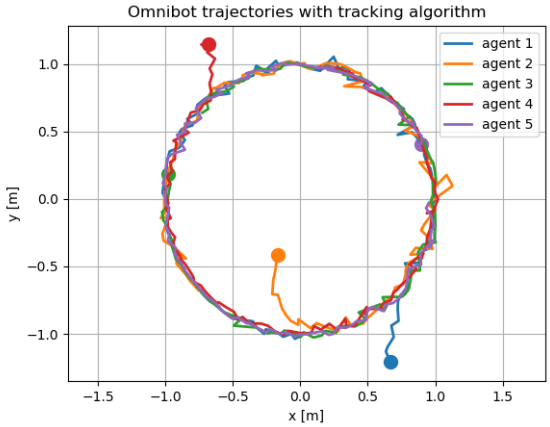

Figure 7.10: Trajectories of verification experiment, visualized in the Cartesian (x, y)-plane. Each omnibot indeed converges to the circular path.

To further analyze the control design, Figure 7.11 and Figure 7.12 show the lateral and rotational tracking performance respectively. The lateral position r indeed converges to the reference radius r^*. However, the rotational orientation error $\theta - \theta_{ref}$ does not converge to zero. Instead, it has a steady-state offset of approximately 5 [°], which can be attributed to the time-varying reference $\theta_{ref}(t) = \phi(t) + \delta^*$ in combination with the finite sampling frequency of the control algorithm. This steady-state error is assumed to be sufficiently small to be negligible.

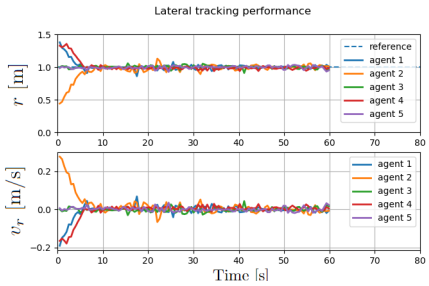

Figure 7.11: Lateral tracking performance of the controller design, showing that the lateral position indeed converges to the reference r^*

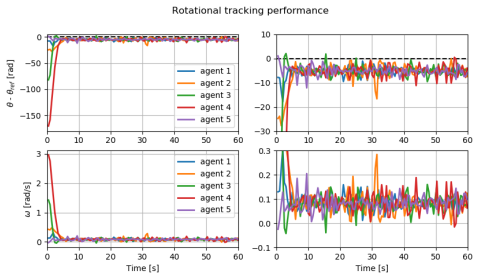

Figure 7.12: Longitudinal tracking performance of the controller design. The left plot shows the complete temporal trajectory, whereas the right plot is zoomed in the y-direction to show the steady-state error.

7.7 Graph structures and consensus protocols

In this section, we introduce the graphs (representing the communication structure of the network) that we use in the experiments. The setup should be able to perform experiments regarding scalable stability and string stability. To this end, we use two different graphs. The graph that is used in the experiments regarding scalable stability is a circular directed graph with $N = 5$, with the agents labeled 1 to 5 (Figure 7.13). The graph that is used in the experiments regarding string stability is a linear directed graph with $N = 6$, with the agents labeled 0 to 5 (Figure 7.14). Agent 0 is a virtual leader, meaning that it is not a physical omnibot but it has a trajectory that is computed in software.

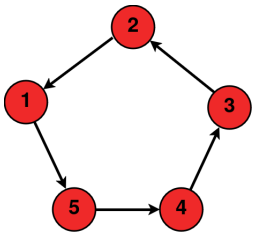

Figure 7.13: Graph of the leaderless directed network

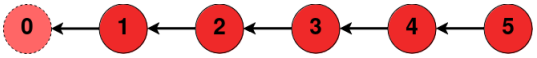

Figure 7.14: Graph of the leader-follower directed network. Agent 0 represents the virtual leader.

Note that an arrow pointing from agent i to agent j corresponds to agent i measuring its relative state with respect to agent j. Both graphs contain a connected spanning tree. Agent 0 in the linear graph does not measure any relative state, and therefore this graph is a leader-follower network. The communication adjacency matrices corresponding to the circular and linear graph read

$$W_{circular} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad W_{linear} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{7.13}$$

respectively. It is assumed that the weight of all edges is equal to 1. This gives the directed Laplacian matrices $L_{circular}$ and L_{linear} as

$$L_{circular} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \quad L_{linear} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{7.14}$$

For the leader-follower graph L_{linear}, the grounded Laplacian $\bar{L}_{linear} \in \mathbb{R}^{N-1 \times N-1}$ is obtained by deleting the first row and first column of the Laplacian. The circular graph has eigenvalues at $\lambda_1 = 0$, $\lambda_{2,3} = 0.69 \pm 0.95i$, $\lambda_{4,5} = 1.81 \pm 0.59i$. If the number of agents in this graph increases, the real part of the algebraic connectivity converges to zero ($Re(\lambda_2) \to 0$ as $N \to \infty$), making it a suitable graph structure to perform experiments on scalable stability. The linear graph has all its non-zero eigenvalues at 1, independent of its network size N (this can easily be deducted from the lower-triangular form).

It is interesting to note that in the linear directed graph, the localized control of agent i only depends on preceding vehicles (labeled $k < i$). Following vehicles (labeled $k > i$) do not influence agent i. This means that the trajectories of the agents during an experiment can be saved and used as the virtual leader for a next experiment. By "connecting" the linear graphs with $N = 6$ of consecutive experiments, we can emulate an experiment with $N > 6$. This means that there is no limit on the amount of agents that we can use in the experiment despite only having five omnibots to work with.

The control input using the conventional control algorithm (introduced in Chapter 2) is denoted by $u_{\text{conventional}}(t)$. The control input using the serial control algorithm (introduced in Chapter 4) is denoted by $u_{\text{serial}}(t)$. The feedback gains of the serial control algorithm are $a_0 = b_1 b_2 > 0$ and $a_1 = b_1 + b_2 > 0$.

$$\begin{aligned} u_{\text{conventional}}(t) &= -a_0 L \hat{s}(t) - a_1 L \dot{\hat{s}}(t) \\ u_{\text{serial}}(t) &= -a_0 L^2 \hat{s}(t) - a_1 L \dot{\hat{s}}(t) \end{aligned} \tag{7.15}$$

We have now elaborated on the design of the experimentation setup. In the next two sections, we will show the experimental results of both the scalable stability and string stability.

8 Experimental results of scalable stability

In this section, experiments are performed using the directed circular graph of (7.13) with five agents to verify the scalable stability of both protocols. As this type of graph has an algebraic connectivity converging to zero for an increasing N, it is suitable to verify the scalable stability of the algorithms. We measure the practical stability bound of the consensus algorithm ($a_{1,\text{practical}}$) and compare this to the theoretical bound ($a_{1,\text{theoretical}}$). We run the experiment for different values of a_0 and a_1, and we observe whether the system achieves consensus of becomes unstable, giving us the practical stability bound.

In the physical setup, the system is subject to measurement noise, and it operates with a finite sampling frequency. Therefore, we expect expect that the stability condition in practice is more conservative than the theoretical stability bound, both for the conventional and the serial consensus protocol.

To illustrate how we have determined instability in these experiments, we consider the following two examples, Figure 8.1 shows the measured unstable response of serial consensus for a certain combination of feedback gains, since it is observed that the velocities steadily blow up. Figure 8.2 shows that the response is stable when a_1 (i.e. the damping) is sufficiently increased, since the states remain neatly bounded. We therefore conclude that the practical stability bound lies between $a_1 = 0.47$ and $a_1 = 0.48$, i.e. $a_{1,practical} = 0.475 \pm 0.005$.

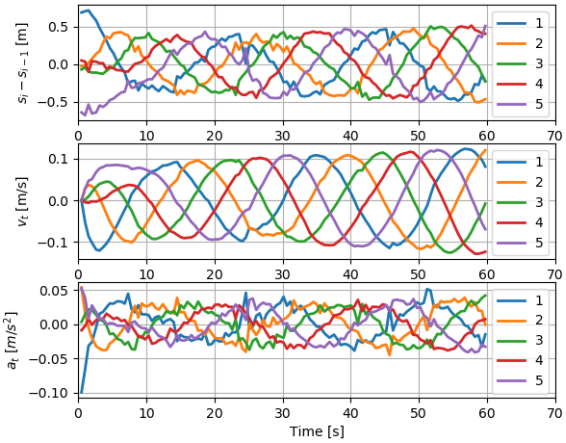

Figure 8.1: Measured vehicle trajectories of experiment using serial consensus with feedback gains $a_0 = 0.075$ and $a_1 = 0.47$. The amplitude of the oscillation increases, and therefore we conclude that this combination of (a_0, a_1) is unstable.

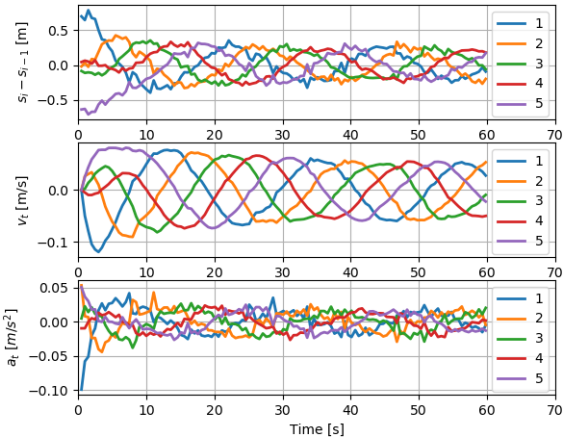

Figure 8.2: Measured vehicle trajectories of experiment using serial consensus with feedback gains $a_0 = 0.075$ and $a_1 = 0.48$. The amplitude of the oscillation decreases, and therefore we conclude that this combination of (a_0, a_1) is stable.

8.1 Conventional consensus algorithm

The theoretical stability condition of conventional consensus in (3.7) for the circular network reads $a_1 > 0.9732\sqrt{a_0}$. This theoretical stability bound is verified experimentally by measuring the practical stability bound. The results are presented in Table 8.1. The measured responses used to determine these stability bounds are presented in Section B.1.

	$a_0 = 0.025$	$a_0 = 0.050$	$a_0 = 0.075$	$a_0 = 0.100$	$a_0 = 0.125$	$a_0 = 0.150$
$\mathbf{a_{1,theoretical}}$	0.154	0.218	0.267	0.308	0.344	0.377
$\mathbf{a_{1,practical}}$	0.1825	0.2625	0.3425	0.4025	0.4825	0.5425

Table 8.1: Measured stability bound for the conventional consensus algorithm

The same results are visualized in Figure 8.3. We observe that the practical stability bound is more conservative than the theoretical stability bound, in the sense that a higher a_1 (i.e. higher damping) is required to achieve consensus. This is in accordance with our expectations. Furthermore, it appears that the bound becomes more conservative for increasing a_0.

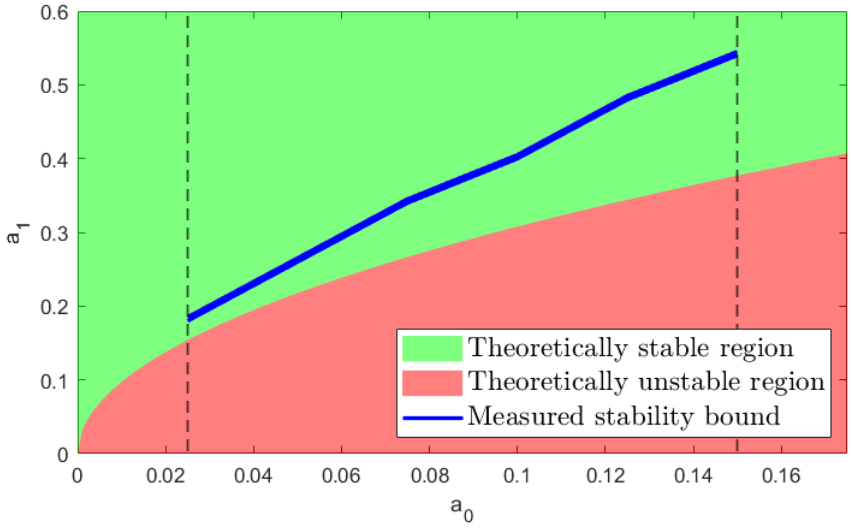

Figure 8.3: Theoretical and measured practical stability bounds of conventional consensus protocol

8.2 Serial consensus algorithm

Next, experiments are performed using the serial consensus algorithm. Serial consensus is guaranteed to be scalably stable when $a_1 \geq 2\sqrt{a_0}$ (which corresponds to real-valued b_1 and b_2). We denote this bound with $a_{1,real}$. Furthermore, the scissor condition in Section 5.2, (5.20) gives the relaxed stability bound $a_1 > 1.6180\sqrt{a_0}$. This theoretical stability bound is again verified experimentally by measuring the practical stability bound. The results are presented in Table 8.2. The measured responses used to determine these stability bounds are presented in Section B.2.

	$a_0 = 0.025$	$a_0 = 0.050$	$a_0 = 0.075$	$a_0 = 0.100$	$a_0 = 0.125$	$a_0 = 0.150$
$\mathbf{a_{1,real}}$	0.316	0.447	0.548	0.632	0.707	0.775
$\mathbf{a_{1,theoretical}}$	0.256	0.362	0.443	0.512	0.572	0.627
$\mathbf{a_{1,practical}}$	0.265	0.385	0.475	0.555	0.615	0.695

Table 8.2: Measured stability bounds for the serial consensus algorithm

The same results are visualized in Figure 8.4. We again observe that the practical stability bound is more conservative than the theoretical stability bound. This is again in line with our expectations.

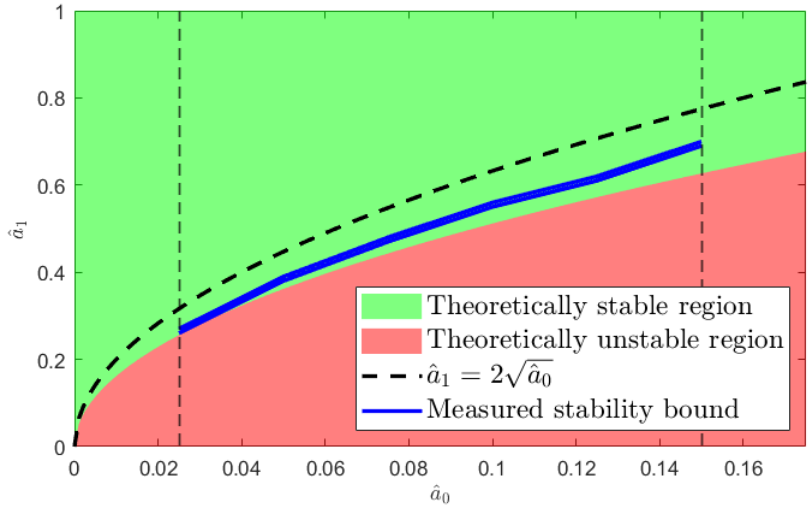

Figure 8.4: Theoretical and measured practical stability bounds of serial consensus protocol

8.3 Comparison between conventional and serial protocol

The practical stability bound of both algorithms are more conservative than their theoretical bound, in the sense that a higher a_1 is required to achieve consensus. This makes sense when we realize that the feedback gains a_0 and a_1 can be the interpreted as the stiffness and damping of the system respectively. In practice, a higher damping is required to guarantee asymptotically stable consensus dynamics.

To visualize how much the unstable region is magnified, the ratio of the practical stability bound over the theoretical stability bound $\frac{a_{1,\text{practical}}}{a_{1,\text{theoretical}}}$ is visualized in Figure 8.5. This shows that the practical stability bound becomes more conservative for increasing a_0. Furthermore, it shows how the factor of conventional consensus is generally larger than the factor of serial consensus. From this, we conclude that serial consensus is more robust to practical implementation than conventional consensus.

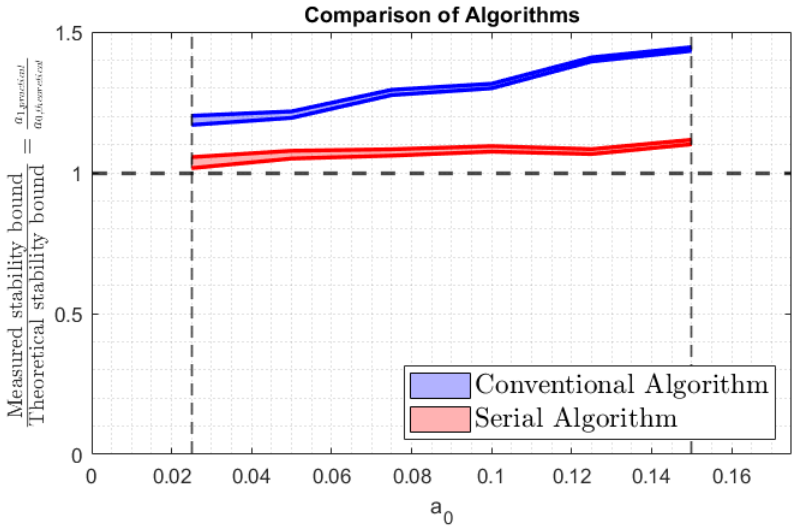

Figure 8.5: Comparison between theoretical and practical stability bounds, both for conventional and serial consensus. The figure shows that serial consensus is more robust to practical implementation.

9 Experimental results of string stability

In this section, experiments are performed using the linear graph of (7.13) to verify the string stability of both protocols. Due to the structure of this graph, there is no limit to the amount of agents we can use in the experiment, as we can emulate a string of vehicles by saving trajectories of consecutive experiments (see Section 7.7). All agents start with consensus in position and a zero velocity, except the leading virtual agent (denoted by agent 0). This agent has a constant (non-zero) reference velocity.

Theory shows that the conventional consensus protocol is string unstable. Therefore, we expect the response of the experiment with the conventional controller to be string unstable too. In Chapter 6 we show that the serial consensus protocol is string stable when $a_1 > 2\sqrt{a_0}$. Therefore, we expect the response of the experiment with the serial controller to be string stable too when $a_1 > 2\sqrt{a_0}$. Next to this, Chapter 6 also says that the string stability guarantee cannot be generalized to feedback gains with $a_1 < 2\sqrt{a_0}$. However, this does not necessarily mean that the system is string unstable. Simulations suggest that the response is string unstable, so therefore we also expect the experiment to be string unstable when $a_1 < 2\sqrt{a_0}$.

First, we demonstrate that for the conventional consensus protocol, the transient error indeed grows exponentially with the number of agents. This is in line with our expectations. We then show that serial consensus is string stable in the sense that worst case behaviour is bounded for $a_1 > 2\sqrt{a_0}$, highlighting its superior scalable performance. We also show, however, that the relaxation in the form of the scissor condition does not guarantee string stability in the practical setting.

9.1 Conventional consensus algorithm

The results of the conventional consensus experiment with feedback gains $a_0 = 0.1$ and $a_1 = 0.6$ are visualized in Figure 9.1. The leading agent has a constant velocity of 0.05 [m/s], and the 40 following agents try to achieve the same velocity while maintaining the inter-vehicular distance. Each vehicle has a maximum tangential velocity (enforced using the protocol in (7.11)). A simulation with the same parameters is performed, and its trajectories are presented in Appendix C.

The measurement noise of the sensors make it hard to see the exponential growth of the relative position. However, we do see the exponential growth in the overshoot of the following vehicles w.r.t. the leader reference velocity. Each consecutive following vehicle has a bigger overshoot until the velocity saturation $\hat{v}_{\mathrm{t,max}} = 0.18$ [m/s] is reached. The behaviour of conventional consensus indeed scales very poorly, confirming our expectations.

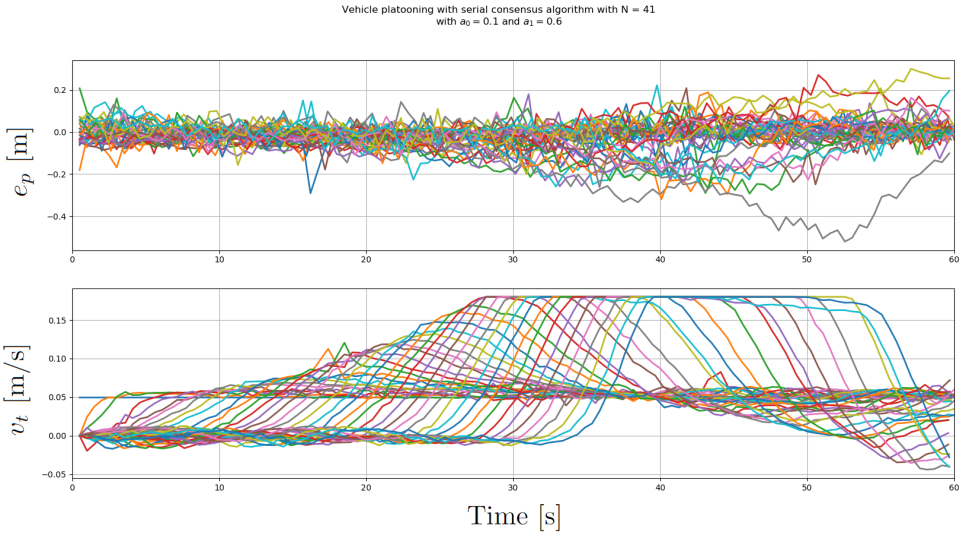

Figure 9.1: Visualization of the exponential growth of transient error for N of conventional consensus algorithm

9.2 Serial consensus algorithm

In this subsection, we first perform an experiment using the serial consensus protocol with $a_1 > 2\sqrt{a_0}$ (corresponding to real-valued b_1 and b_2), to verify the string stability guarantee of Chapter 6. We then perform an experiment with $a_1 < 2\sqrt{a_0}$ (corresponding to complex-valued b_1 and b_2) to demonstrate that this causes the vehicle platoon to become string unstable.

9.2.1 Serial consensus with $a_1 > 2\sqrt{a_0}$

The results of the serial consensus experiment with feedback gains $a_0 = 0.1$ and $a_1 = 0.8$ are visualized in Figure 9.2. The leading agent has a constant velocity of 0.10 [m/s], and the 30 following vehicles try to achieve the same velocity while maintaining a constant inter-vehicular distance. A simulation with the same parameters is performed, and its trajectories are presented in Appendix C.

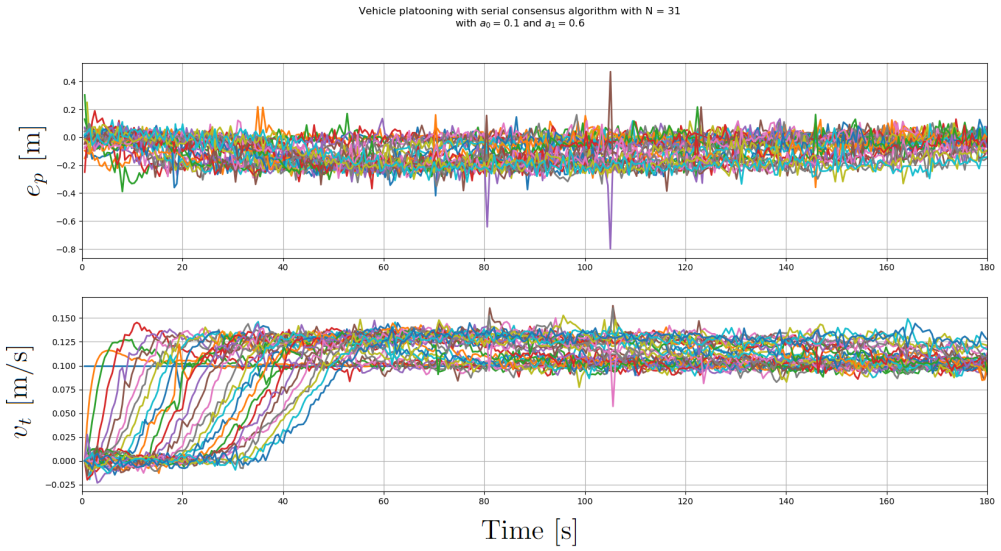

Figure 9.2: Experimental results of serial consensus protocol. We see that the relative position and velocity overshoot can be bounded, meaning that the system is string stable.

The results clearly show that now the relative position and overshoot in velocity with respect to the leader agent do not grow unboundedly or become saturated. Instead, the transient behaviour can be bounded by the initial maximum deviation according to (2.12), independent of the amount of vehicles in the platoon. This shows that the behaviour of serial consensus indeed scales very favourably, especially compared to a conventional consensus approach. This confirms our expectation.

By using the parameters in $a_0 = 0.1$ and $a_1 = 0.8$ with $||e_p(0)||_\infty = 0$ [m] and $||e_v(0)||_\infty = 0.1$ [m/s] in the performance criterion in (6.6) gives the bounds $||e_p(t)||_\infty = 0.4082$ [m] and $||e_v(t)||_\infty = 0.1633$. The relative position e_p and velocity deviation e_v in the experiment can indeed be bounded by these infinity norms, except for the position of the experiment at approximately 81 [s] and 105 [s]. We regard this is as an outlier in the position due to measurement noise.

9.2.2 Serial consensus with $a_1 > 2\sqrt{a_0}$

The results of the serial consensus experiment with feedback gains $a_0 = 0.1$ and $a_1 = 0.6$ are visualized in Figure 9.3. The leading agent has a constant velocity of 0.10 [m/s], and the 40 following vehicles try to achieve the same velocity while maintaining the inter-vehicular distance. A simulation with the same parameters is performed, and its trajectories are presented in Appendix C.

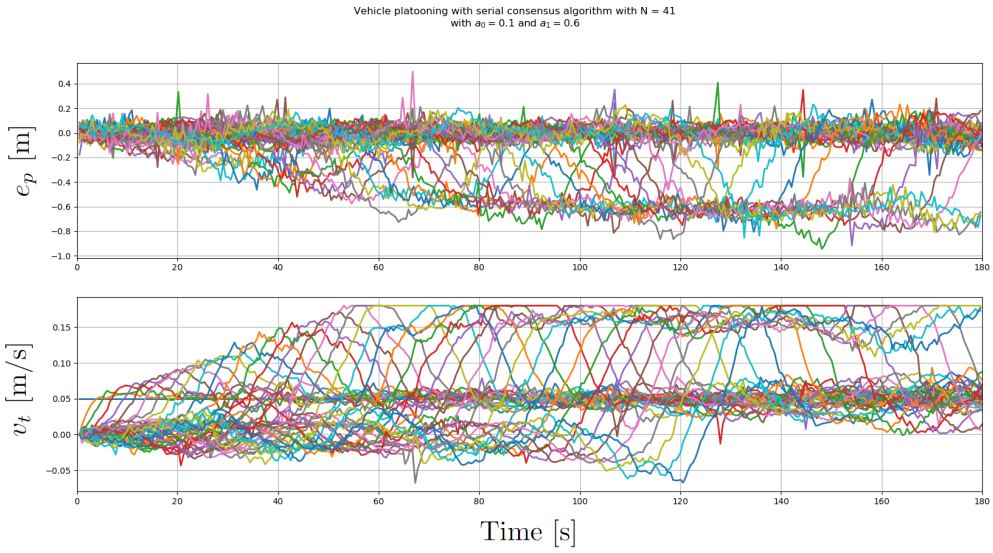

Figure 9.3: Visualization exponential growth of transient error for increasing N of conventional consensus algorithm

The results clearly show that the relative position and overshoot in velocity with respect to the leader agent keep on growing with the agents until the agents reach their velocity limit. Therefore, we conclude that the system is string unstable. In other words, the relaxation of the scalable stability condition does not guarantee string stability, which is in accordance with our expectation.

We have now experimentally verified the poor scalable behaviour of the conventional consensus protocol. This is in contrast to the results of the serial consensus protocol, which shows very favourable scalable behaviour in the same experiments. Chapter 4 shows that this comes at the cost of extra communication in the network. In the next section, we explore the possibilities of a specific implementation of serial consensus, which omits the need for this extra communication.

10 Serial consensus using only direct measurements

In this section, we explore the possibilities of a specific implementation of serial consensus. In this implementation, we carefully choose the graph Laplacians L_1 and L_2 such that no extra communication is required, as proposed by Hansson in [7].

The novel serial consensus algorithm can be achieved with an n-step implementable relative feedback controller, as shown in Section 4.2. This requires an extra communication step with respect to the communication topology of the network, encoded by the adjacency matrix W. In some applications this might not be desirable, for instance in applications where realizing this extra communication is challenging or expensive. Therefore, Hansson [7] proposes a specific implementation of serial consensus for networks with second-order agent dynamics that eliminates the need for extra communication. Consider a circular undirected network with second order agent integrator dynamics, with the corresponding communication adjacency matrix

$$W = \begin{bmatrix} 0 & 1 & 0 & \dots & 1 \\ 1 & 0 & 1 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{10.1}$$

Now, instead of imposing that $L_k = b_k L$ for $k = 1, 2$, we relax this to imposing that $L_1 = b_1 L$ and $L_2 = b_2 L^T$ with L being an underlying graph Laplacian encoding the communication topology of the network. By choosing L and L^T as the graph Laplacians, their product $LL^T \in \mathcal{A}^1(W, c)$. In other words, we ensure that the resulting controller is 1-step implementable w.r.t. the communication adjacency matrix W. This is obtained by choosing one Laplacian where agents only use relative information of the agent ahead of them, and one Laplacian where agents only use the relative information of the agent behind them, as

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & -1 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \text{ and } L^T = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{10.2}$$

The resulting controller can be written as $u(t) = -b_1 b_2 LL^T x(t) - (b_1 L + b_2 L^T)\dot{x}(t)$. Note that this is a different expression than the controller in (5.15) due to the different Laplacians. To see this, consider the the control input

$$u(t) = -b_1 b_2 \begin{bmatrix} 2 & -1 & 0 & \dots & -1 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 2 \end{bmatrix} x(t) - \begin{bmatrix} b_1 + b_2 & -b_2 & 0 & \dots & -b_1 \\ -b_1 & b_1 + b_2 & -b_2 & \dots & 0 \\ 0 & -b_1 & b_1 + b_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -b_2 & 0 & 0 & \dots & b_1 + b_2 \end{bmatrix} \dot{x}(t). \tag{10.3}$$

First of all, we see that the resulting controller is indeed a 1-step implementable relative feedback controller with respect to W. Furthermore, from this notation it is clear that in order to have a real valued control input $u(t)$ it is no longer sufficient to impose b_1 and b_2 to be a *complex conjugate* pair. Instead, it is required that b_1 and b_2 are *real valued*. We again define $a_0 = b_1 b_2$ and $a_1 = b_1 + b_2$, which can again be interpreted to be proportional to the stiffness and damping of the system respectively. As we impose that $u(t)$ is real valued, it must hold that $a_1 \geq 2\sqrt{a_0}$. Choosing $a_1 = 2\sqrt{a_0}$ means that $b_1 = b_2$. A higher a_1 (i.e. a high damping) means that b_1 and b_2 diverge further away from each other. We introduce the new parameters b_0 and Δb related to b_1 and b_2 through $b_1 = b_0 - \Delta b$ and $b_2 = b_0 + \Delta b$. The control input $u(t)$ in terms of b_0 and Δb reads

$$u(t) = -(b_0^2 - \Delta b^2) \begin{bmatrix} 2 & -1 & 0 & \dots & -1 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 2 \end{bmatrix} x(t) - \begin{bmatrix} 2b_0 & -(b_0 + \Delta b) & 0 & \dots & -(b_0 - \Delta b) \\ -(b_0 - \Delta b) & 2b_0 & -(b_0 + \Delta b) & \dots & 0 \\ 0 & -(b_0 - \Delta b) & 2b_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -(b_0 + \Delta b) & 0 & 0 & \dots & 2b_0 \end{bmatrix} \dot{x}(t). \tag{10.4}$$

This notation gives us another intuitive interpretation of the algorithm. Choosing a high b_0 corresponds to choosing high feedback gains. Choosing a high Δb means choosing the $b_{1,2}$ far away from each other,

corresponding to a high damping. However, it also says something about the "asymmetry" of the controller between looking at the agents ahead and agents behind. Choosing $\Delta b > 0$ means that the localized control depends more on the relative velocity of the agents behind. Equivalently, $\Delta b < 0$ means that the localized control depends more on the relative velocity of the agents ahead.

To demonstrate the serial consensus algorithm using only direct measurements, an experiment is performed with a circular undirected network with $N = 5$ agents. The communication structure W of the network is visualized in Figure 10.1. The graph Laplacians L_1 and L_2 are visualized in Figure 10.2 and Figure 10.3 respectively, where all edge weight are equal to one. L corresponds to the agents looking forward, and L^T corresponds to the agents looking backwards.

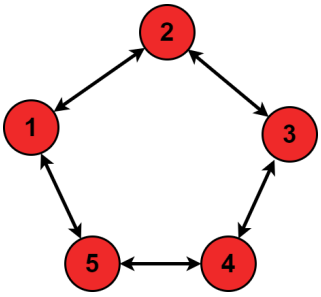

Figure 10.1: Visualization of communication structure W used in the experiment.

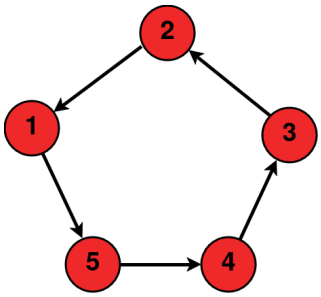

Figure 10.2: Visualization of Laplacian L where each vehicle uses relative measurement of the vehicle ahead.

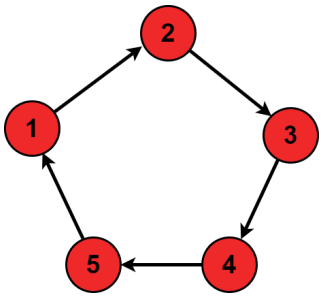

Figure 10.3: Visualization of Laplacian L^T where each vehicle uses relative measurement of the vehicle behind.

First, the experiment is performed with $b_1 = b_2 = 0.316$. This corresponds to $a_1 = 2\sqrt{a_0}$, or equivalently $b_0 = 0.316$ and $\Delta b = 0$. This means that the system is symmetric in the sense that the localized control has the same feedback gains on relative velocity for vehicles ahead and behind. The measured relative positions and velocities are visualized in Figure 10.4.

Next, the experiment is performed with $b_1 = 0.18$ and $b_2 = 0.55$. This corresponds to $a_1 > 2\sqrt{a_0}$, or equivalently $b_0 = 0.365$ and $\Delta b = 0.185 > 0$. The system is asymmetric in the sense that the localized control has higher feedback gain on relative velocity for vehicles behind than vehicles ahead. The measured relative positions and velocities are visualized in Figure 10.5.

Finally, the experiment is performed with $b_1 = 0.55$ and $b_2 = 0.18$. This again corresponds to $a_1 > 2\sqrt{a_0}$. Equivalently, $b_0 = 0.365$ and $\Delta b = -0.185 < 0$. The system is now asymmetric in the sense that the localized control has higher feedback gain on relative velocity for vehicles ahead than vehicles behind. The measured relative positions and velocities are visualized in Figure 10.6.

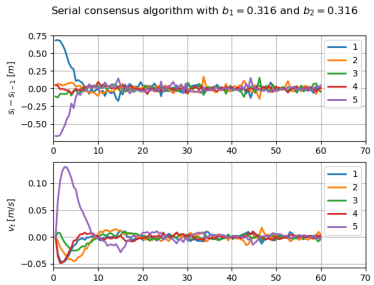

Figure 10.4: Serial consensus experiment with $b_1 = b_2 = 0.316$

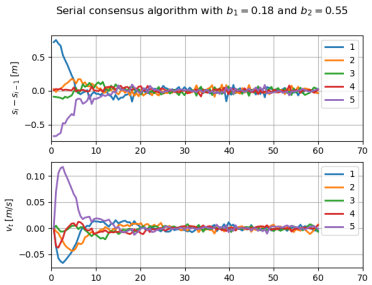

Figure 10.5: Serial consensus experiment with $b_1 = 0.18$ and $b_2 = 0.55$

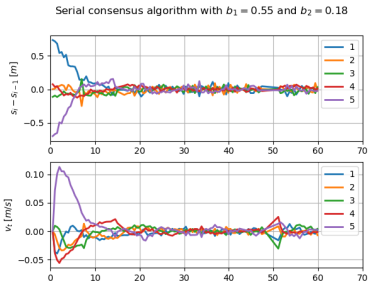

Figure 10.6: Serial consensus experiment with $b_1 = 0.55$ and $b_2 = 0.18$

The results show that this implementation of serial consensus indeed achieves consensus, without the need for any additional communication. In future work, it might be interesting to further investigate the stability conditions of this implementation, as avoiding the additional communication can be very desirable when implementing the algorithm on a physical system.

11 Conclusion and directions for future work

We now summarize the most important achievements of this project. First of all, we have developed a physical experimentation setup in a lab setting with five robots, mimicking vehicle platooning. The agents in the platoon network can be modelled as n^{th} order integrators. Their control is decoupled, allowing for a separate lateral controller that keeps the agents on the path, and a consensus protocol that drives the agents along this path. The setup can be used for future experiments to verify consensus algorithms.

We used this experimentation setup to implement the serial consensus protocol on networks containing agents with second order integrator dynamics. The implementation of the discretised protocol was subject to measurement noise. For the serial consensus protocol, it was mathematically proven in [6,7] that it is scalably stable and string stable. In this report we have presented experimental verification that these results also hold in a real-life implementation. In parallel, a conventional consensus protocol was implemented, which was demonstrated to not be scalably stable and string stable, underlining the superiority of the serial consensus protocol. Moreover, we have introduced a relaxation of the serial consensus protocol, and demonstrated that scalable stability is still satisfied, but that the relaxation is subject to string instability.

Next to this, we have verified a specific implementation of serial consensus for networks with second-order agent dynamics that eliminates the need for additional communication in the network, as suggested in [7], and we tried to develop some intuition into this approach.

There are several interesting directions for future work. First of all, the experimentation setup could be used to verify serial consensus for networks with higher-order agent dynamics ($n \geq 3$). One could investigate how the practical stability bound for these systems relates to the theoretical stability bound, and compare this with a conventional consensus approach. A second direction is to investigate the effect of velocity saturations of individual agents on the behaviour of the serial consensus algorithm. Simulations suggest that this might not only have an effect on the performance, but also on stability. The topic of velocity saturation is very relevant in the context of vehicle platooning, as vehicles (e.g. cars) in a platoon generally have to respect some maximum velocity. The same can be investigated for saturations of higher-order signals, like maximum accelerations. Finally, investigating the stability conditions of second order serial consensus that uses only direct measurements (introduced in Chapter 10) is an interesting next step. This too is a relevant direction, as avoiding the additional step of communication can be very desirable when implementing the algorithm on a physical system.

References

[1] E. Jensen and B. Bamieh, "On Structured-Closed-Loop versus Structured-Controller Design: the Case of Relative Measurement Feedback," pp. 1–16, 2020. [Online]. Available: http://arxiv.org/abs/2008.11291

[2] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *Proceedings of the American Control Conference*, vol. 3, no. September, pp. 1806–1810, 1995.

[3] P. Seiler, A. Pant, and K. Hedrick, "! 1 ! 1," vol. 49, no. 10, pp. 1835–1841, 2004.

[4] E. Tegling, B. Bamieh, and H. Sandberg, "Scale fragilities in localized consensus dynamics," *Automatica*, vol. 153, p. 111046, 2023. [Online]. Available: https://doi.org/10.1016/j.automatica.2023.111046

[5] S. Stüdli, M. M. Seron, and R. H. Middleton, "Vehicular Platoons in cyclic interconnections with constant inter-vehicle spacing," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2511–2516, 2017.

[6] J. Hansson and E. Tegling, "A Closed-Loop Design for Scalable High-Order Consensus," *Proceedings of the IEEE Conference on Decision and Control*, pp. 7388–7394, 2023.

[7] ——, "Closed-loop design for scalable performance of vehicular formations," pp. 1–8, 2024. [Online]. Available: http://arxiv.org/abs/2402.15208

[8] G. Gunter, D. Gloudemans, R. E. Stern, S. McQuade, R. Bhadani, M. Bunting, M. L. Delle Monache, R. Lysecky, B. Seibold, J. Sprinkle, B. Piccoli, and D. B. Work, "Are Commercially Implemented Adaptive Cruise Control Systems String Stable?" *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6992–7003, 2021.

[9] B. Besselink and S. Knorn, "Scalable Input-to-State Stability for Performance Analysis of Large-Scale Networks," *IEEE Control Systems Letters*, vol. 2, pp. 507–512, 2018.

[10] C. E. Cantos, J. J. Veerman, and D. K. Hammond, "Signal velocity in oscillator arrays," *European Physical Journal: Special Topics*, vol. 225, no. 6-7, pp. 1115–1126, 2016.

[11] I. Herman, "Scaling in vehicle platoons," no. May, p. 210, 2016.

[12] Wikipedia, "Metzler matrix." [Online]. Available: https://en.wikipedia.org/wiki/Metzler_matrix

[13] T. v. Oorschot, "Serial consensus verification playlist." [Online]. Available: https://www.youtube.com/playlist?list=PLFCGpbpO3e3TsO2fvl4uOkMnRioEAE6_o

A Maximizing omnibot velocity along circle

This appendix shows the derivation of the results in (7.7) and (7.8) in Chapter 7. We derive the optimal orientation of an omnibot with respect to a circular path and the corresponding maximum velocity (normalized to the maximum omniwheel velocity).

The relation between the velocities of the individual wheels of the omnibot and its velocities in the global Cartesian frame is given by $v_{\text{Cartesian}} = M(\theta)v_{\text{wheels}}$. The relation between these Cartesian velocities and the velocities in the polar frame is given by $v_{\text{Cartesian}} = R(\phi)v_{\text{polar}}$. $M(\theta)$ is referred to as the omnibot velocity kinematics matrix, and $R(\phi)$ is referred to as the rotation matrix. The relation between the velocities of the individual omniwheels and the polar velocities is

$$\begin{bmatrix} v_r \\ v_t \\ \omega \end{bmatrix} = R^{-1}(\phi)M(\theta) \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = M^{-1}(\theta)R(\phi) \begin{bmatrix} v_r \\ v_t \\ \omega \end{bmatrix}, \tag{A.1}$$

where v_r, v_t and ω represent the radial, tangential and angular velocity of the omnibot, whereas v_1, v_2 and v_3 represent the velocity of omniwheel 1, 2 and 3 respectively. The expressions for the omnibot velocity kinematics matrix $M^{-1}(\theta)$ and the rotation matrix $R(\phi)$ read

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } M^{-1}(\theta) = \begin{bmatrix} -\sin(\theta-\xi_1) & \cos(\theta-\xi_1) & R \\ -\sin(\theta-\xi_2) & \cos(\theta-\xi_2) & R \\ -\sin(\theta-\xi_3) & \cos(\theta-\xi_3) & R \end{bmatrix}, \text{ where } \xi_{1,2,3} = \left\{0, \tfrac{4}{3}\pi, \tfrac{2}{3}\pi\right\}. \tag{A.2}$$

It is interesting to note that the rows of M^{-1} are equivalent except for the value of ξ_i, representing the different angle offsets of the arms of the omnibot. We realize that the inverse omnibot velocity kinematics matrix $M^{-1}(\theta)$ can also be interpreted as a rotation matrix, pre-multiplied by a certain row vector. This is shown by selecting an individual row of $M^{-1}(\theta)$ using the vector e_i and realizing that the same row can be constructed by pre-multiplying the rotation matrix $R(\xi_i - \theta)$ by the row vector $\begin{bmatrix} 0 & 1 & R \end{bmatrix}$.

$$e_i^T M^{-1}(\theta) = \begin{bmatrix} -\sin(\theta-\xi_i) & \cos(\theta-\xi_i) & R \end{bmatrix} \tag{A.3}$$

$$\begin{bmatrix} 0 & 1 & R \end{bmatrix} R(\xi_i - \theta) = \begin{bmatrix} \sin(\xi_i-\theta) & \cos(\xi_i-\theta) & R \end{bmatrix} = \begin{bmatrix} -\sin(\theta-\xi_i) & \cos(\theta-\xi_i) & R \end{bmatrix} \tag{A.4}$$

$$\rightarrow e_i^T M^{-1}(\theta) = \begin{bmatrix} 0 & 1 & R \end{bmatrix} R(\xi_i - \theta) \tag{A.5}$$

Next, the expression is post-multiplied by the rotation matrix $R(\phi)$. This results in a multiplication of two rotation matrices, which is equivalent to applying one rotation matrix with the sum of the rotations. This notation allows to compute each individual row of $M^{-1}(\theta)R(\phi)$, meaning that also the complete matrix can be computed.

$$e_i^T M^{-1}(\theta)R(\phi) = \begin{bmatrix} 0 & 1 & R \end{bmatrix} R(\xi_i - \theta)R(\phi) = \begin{bmatrix} 0 & 1 & R \end{bmatrix} R(\phi - \theta + \xi_i) \tag{A.6}$$

$$\rightarrow M^{-1}(\theta)R(\phi) = \begin{bmatrix} \sin(\phi-\theta+\xi_1) & \cos(\phi-\theta+\xi_1) & R \\ \sin(\phi-\theta+\xi_2) & \cos(\phi-\theta+\xi_2) & R \\ \sin(\phi-\theta+\xi_3) & \cos(\phi-\theta+\xi_3) & R \end{bmatrix} \tag{A.7}$$

This gives an easier notation of the relationship between the desired polar velocities of the omnibot and the required individual omniwheel velocities. This simplified relationship reads

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \sin(\phi-\theta+\xi_1) & \cos(\phi-\theta+\xi_1) & R \\ \sin(\phi-\theta+\xi_2) & \cos(\phi-\theta+\xi_2) & R \\ \sin(\phi-\theta+\xi_3) & \cos(\phi-\theta+\xi_3) & R \end{bmatrix} \begin{bmatrix} v_r \\ v_t \\ \omega \end{bmatrix}. \tag{A.8}$$

We now want to optimize the velocity of the omnibot along the circle, while not saturating the velocities of the individual omniwheels. To this end, v_t is maximized, assuming that the omnibot is driving on the circle,

meaning that $v_r = 0$. Furthermore, it is assumed that the orientation of the omnibot w.r.t. the tangent to the circle is constant, i.e. the omnibot rotates along with the circle arc. Therefore, it is imposed that $\omega = \frac{v_t}{r}$, and that $\phi - \theta$ is constant. This results in the following expression for the required individual omniwheel velocities to obtain a desired v_t.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (\cos(\theta - \phi + \xi_1) + \frac{R}{r})v_t \\ (\cos(\theta - \phi + \xi_2) + \frac{R}{r})v_t \\ (\cos(\theta - \phi + \xi_3) + \frac{R}{r})v_t \end{bmatrix} \tag{A.9}$$

It is assumed that the individual omniwheel velocities saturate at a value of c. This gives the constraint that $|v_i| \leq c$ for $i = 1, 2, 3$. Next, the variables $x := \frac{c}{v_t}$, $\beta := \frac{R}{r}$ and $\delta := \theta - \phi$ are introduced for easier notation. The scalar x is the ratio of the maximum omniwheel velocity over the tangential velocity of the omnibot. Note that maximizing v_t corresponds to minimizing x. The scalar β defines the geometrical ratio of the system, defined as the length of the arms of the omnibot over the radius of the reference circle. The scalar δ defines the optimal orientation of the omnibot w.r.t. the reference path. The resulting optimization problem comes down to finding the optimal δ that allows for a maximum v_t while subject to the following three constraints

$$-x \leq \cos(\delta + \xi_i) + \beta \leq x, \quad \forall i = 1, 2, 3. \tag{A.10}$$

The optimization problem is visualized in Figure A.1, which shows that the problem comes down to finding the δ for which the smallest x can be chosen that ensures that the three red dots still lie between the dotted red lines. The figure shows that the solution of the optimization problem depends on the system geometry β. More specifically, we can distinguish three cases depending on the value of β. In the first case, all three points lie exactly on the dotted red lines. We denote the β for which this happens by the critical system geometry β^*. For $\beta < \beta^*$, only two points lie on the dotted lines; one on the right line and on the left line. For $\beta > \beta^*$, only two points lie on the dotted right line on the right.

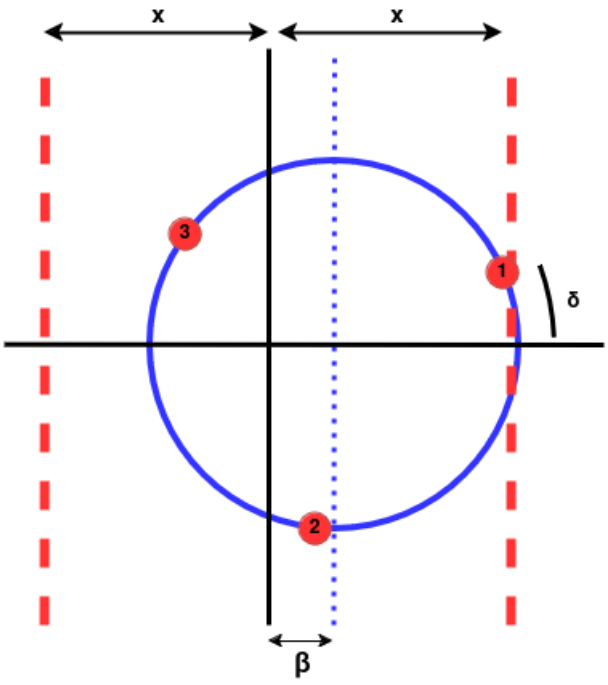

Figure A.1: Visualization of optimization problem

Critical system geometry: $\beta = \beta_{crit}$

The critical system geometry β_{crit} is defined as the β where each constraint is active, i.e. where each of the wheels reaches its maximum velocity. From the visualized geometry, it is clear that the optimal orientation is $\delta^* = \frac{1}{3}\pi$. As each constraint is active, it holds that

$$\begin{cases} \cos(\frac{1}{3}\pi) + \beta = x \\ \cos(\frac{1}{3}\pi + \frac{2}{3}\pi) + \beta = -x \\ \cos(\frac{1}{3}\pi + \frac{4}{3}\pi) + \beta = x \end{cases} \rightarrow \begin{cases} \frac{1}{2} + \beta = x \\ -1 + \beta = -x \\ \frac{1}{2} + \beta = x \end{cases}. \tag{A.11}$$

This gives the equality $\frac{1}{2} + \beta = 1 - \beta$. Solving for β gives $\beta^* = \frac{1}{4}$. The corresponding x reads $x^* = \frac{1}{2} + \beta^* = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$. However, we are interested in the maximum tangential velocity, which we can now write as $v_t^* = \frac{4}{3}c$.

Subcritical system geometry: $\beta < \beta_{crit}$

In the subcritical case, the optimal configuration is when two constraints are active; one at $-x$ and one at x. These two active constraints give that

$$\begin{cases} \cos(\delta^* + \frac{4}{3}\pi) + \beta = x \\ \cos(\delta^* + \frac{2}{3}\pi) + \beta = -x \end{cases} \tag{A.12}$$

We solve these two equations for δ^* by summing them, giving $-\cos(\delta) + 2\beta = 0$ (here, we use that $\cos\delta + \cos\delta + \frac{2}{3}\pi + \cos\delta + \frac{4}{3}\pi = 0$). Therefore, we know that $\cos(\delta^*) = 2\beta$ and consequently $\delta^* = \arccos(2\beta)$. The corresponding x is obtained by filling in the optimal orientation δ^* in one of the equations. We simplify this equation as

$$\begin{aligned} x &= \cos(\arccos(2\beta) + \frac{4}{3}\pi) + \beta \\ x &= \cos(\delta + \frac{4}{3}\pi) \\ x &= \cos(\delta)\cos(\frac{4}{3}\pi) - \sin(\delta)\sin(\frac{4}{3}\pi) \\ &\cos(\delta) = 2\beta \text{ and } \sin(\delta) = \sqrt{1-\cos(\delta)^2} = \sqrt{1-4\beta^2} \\ x &= 2\beta\cos(\frac{4}{3}\pi) - \sqrt{1-4\beta^2}\sin(\frac{4}{3}\pi) + \beta \\ x &= -\beta + \frac{1}{2}\sqrt{3}\sqrt{1-4\beta^2} + \beta \\ x &= \frac{1}{2}\sqrt{3}\sqrt{1-4\beta^2}. \end{aligned} \tag{A.13}$$

We now know that $x^* = \frac{c}{v_t^*} = \frac{1}{2}\sqrt{3}\sqrt{1-4\beta^2}$. Therefore, we can write the maximum omnibot velocity as $v_t^* = \frac{2\sqrt{3}}{3\sqrt{1-4\beta^2}}c$.

Supercritical system geometry: $\beta > \beta_{crit}$

In the supercritical case, the optimal configuration is when two constraints are active at x. Clearly, this gives that the optimal orientation is $\delta^* = \frac{1}{3}\pi$. Filling in this optimal orientation in one of the equations gives that $\cos(\delta^*) = \cos(\frac{1}{3}\pi) + \beta = x$. Simplifying leads to $\frac{1}{2} + \beta = x^* = \frac{c}{v_t}$, and solving for v_t^* gives $v_t^* = \frac{2}{2\beta+1}c$.

B Scalable stability experimental results

This appendix visualizes the measured trajectories that we considered to determine the stability bounds in Chapter 8. It illustrates how the consensus dynamics are unstable when a_1 is chosen too low, and that they become stable when a_1 is increased. In Section B.1 we used the conventional consensus algorithm, and in Section B.2 we used the serial consensus algorithm.

B.1 Conventional consensus algorithm

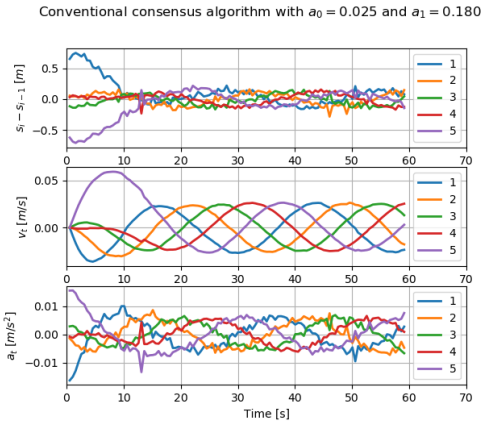

Figure B.1

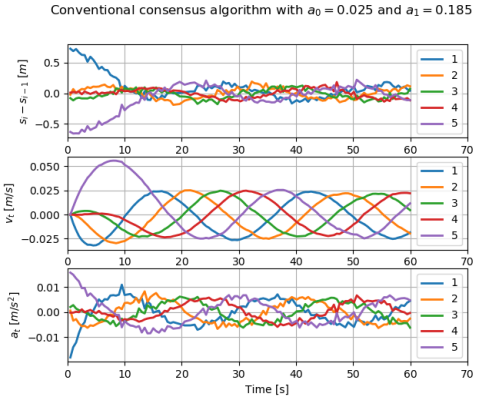

Figure B.2

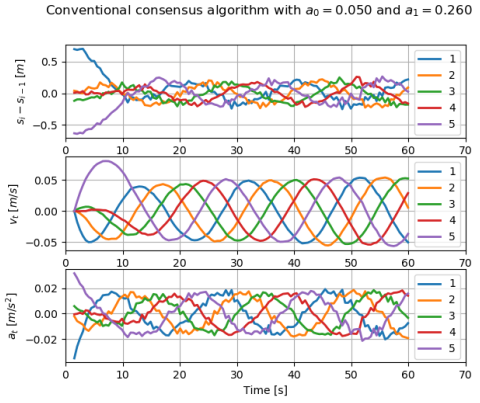

Figure B.3

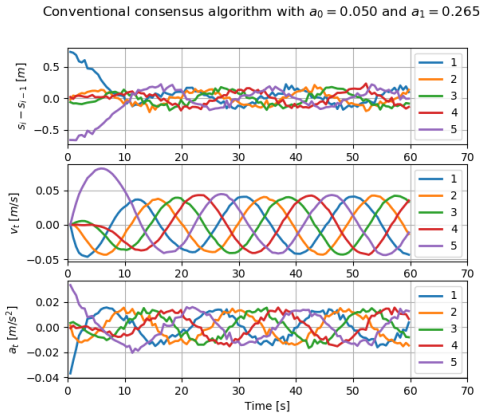

Figure B.4

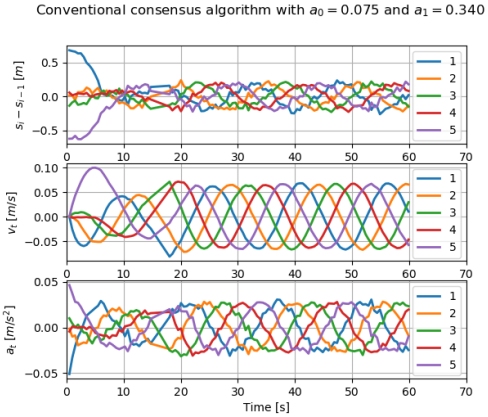

Figure B.5

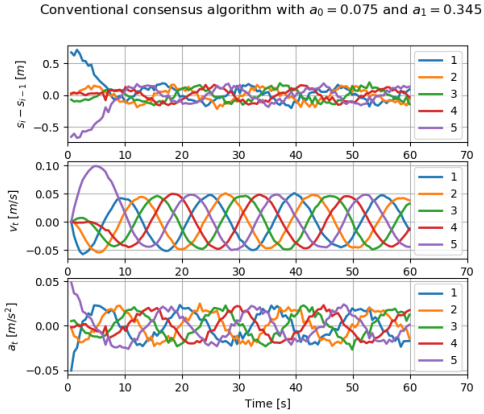

Figure B.6

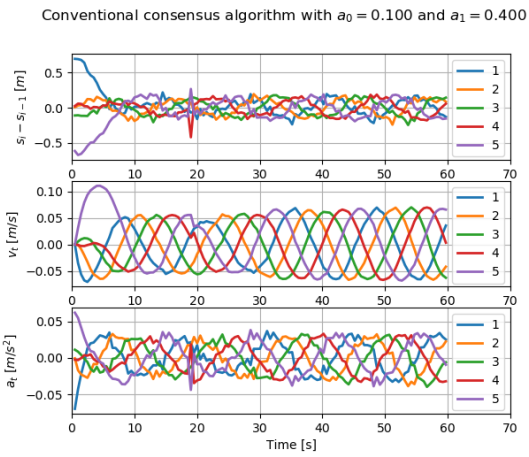

Figure B.7

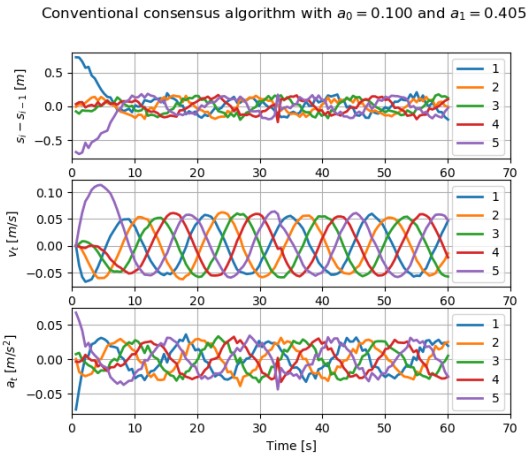

Figure B.8

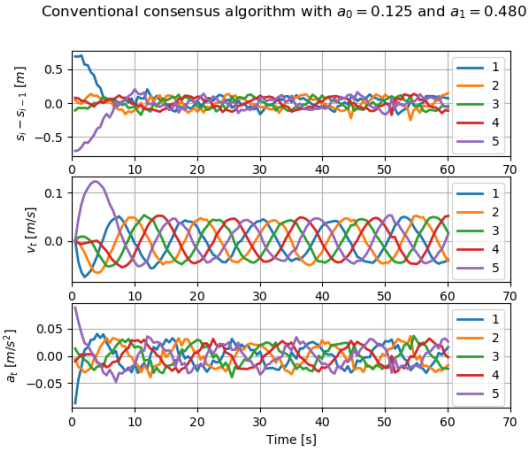

Figure B.9

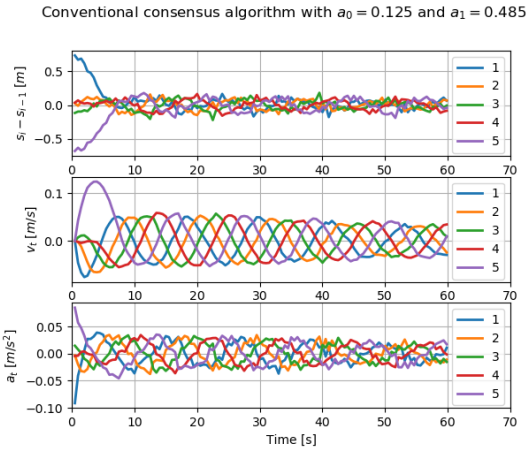

Figure B.10

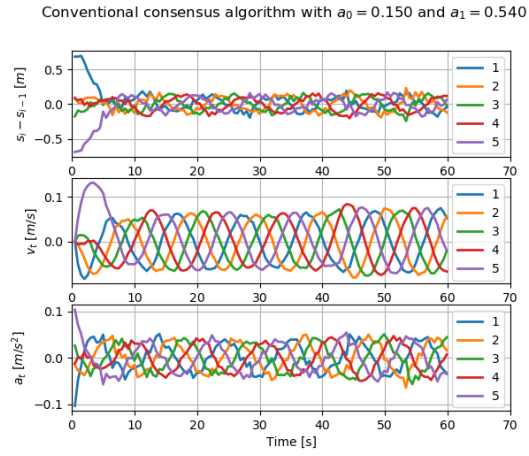

Figure B.11

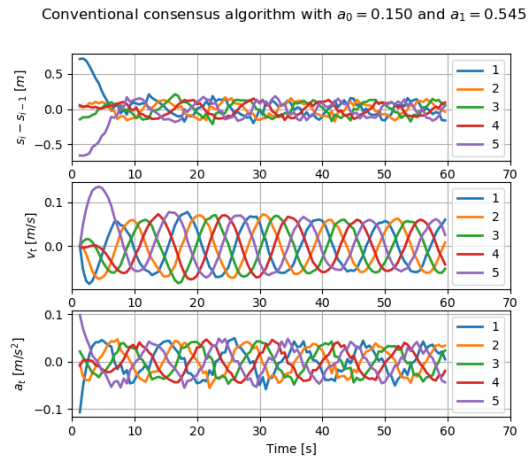

Figure B.12

B.2 Serial consensus algorithm

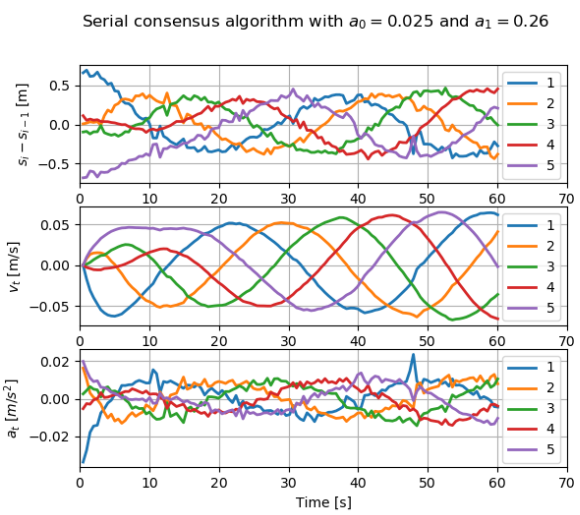

Figure B.13

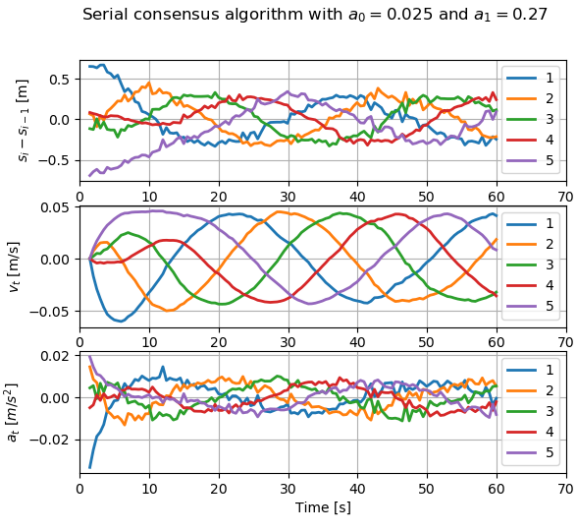

Figure B.14

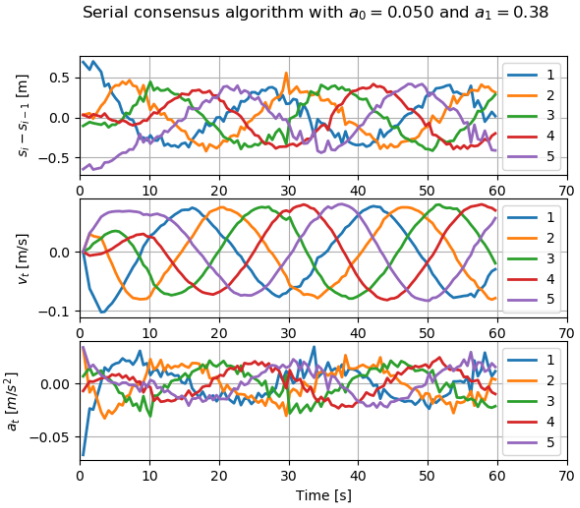

Figure B.15

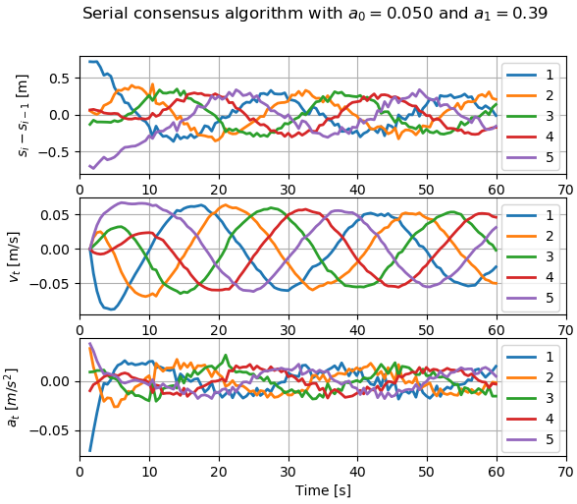

Figure B.16

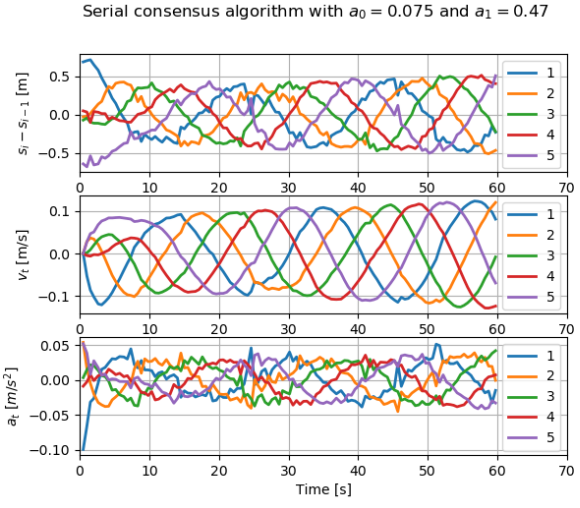

Figure B.17

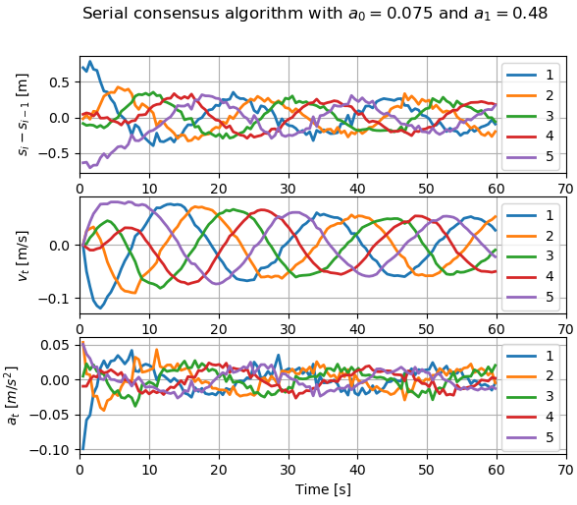

Figure B.18

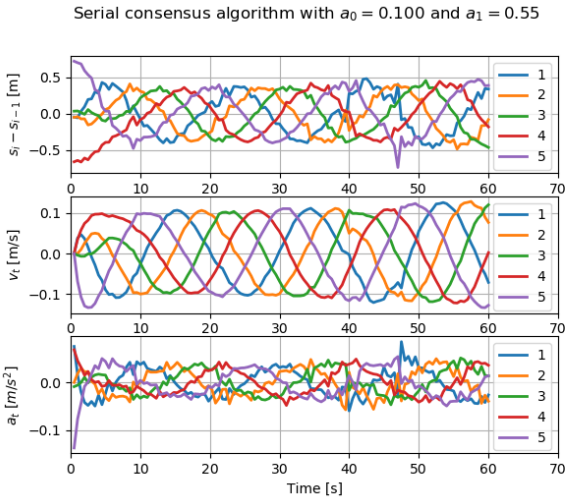

Figure B.19

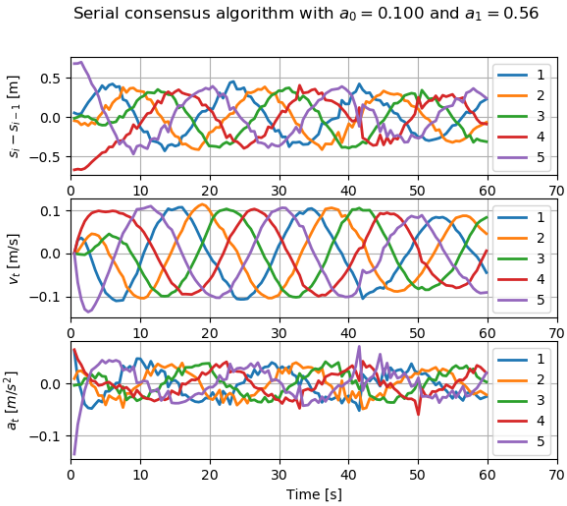

Figure B.20

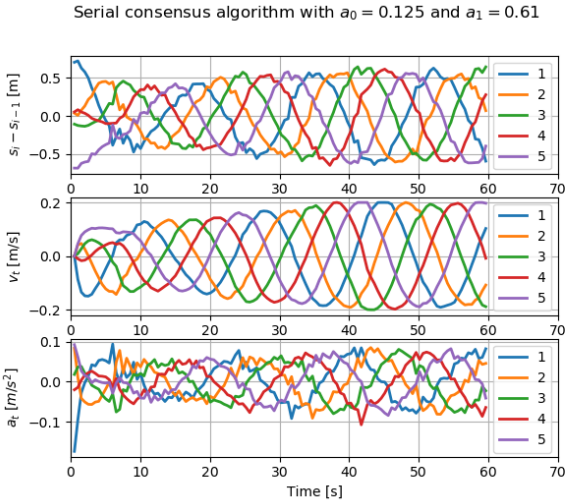

Figure B.21

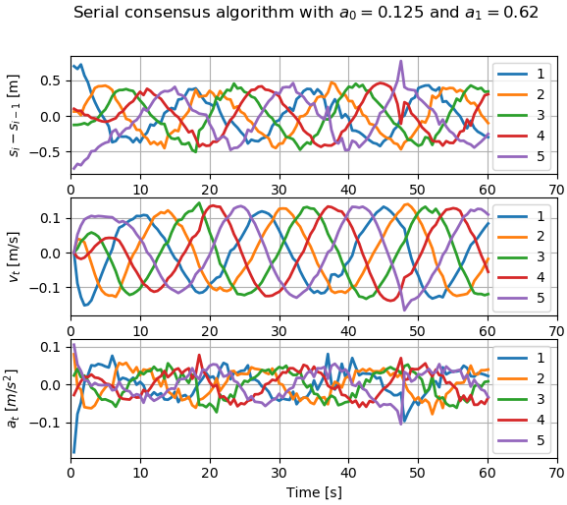

Figure B.22

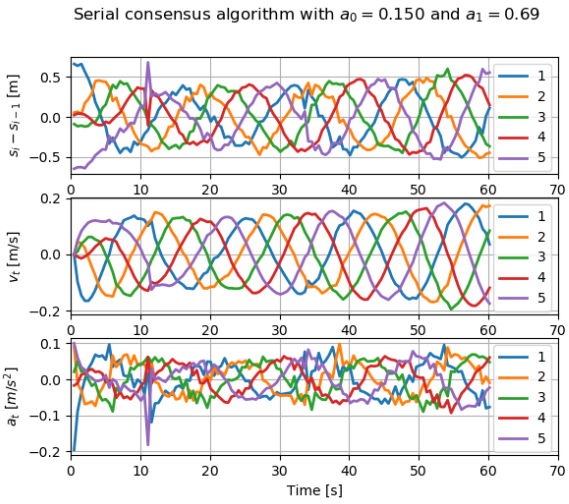

Figure B.23

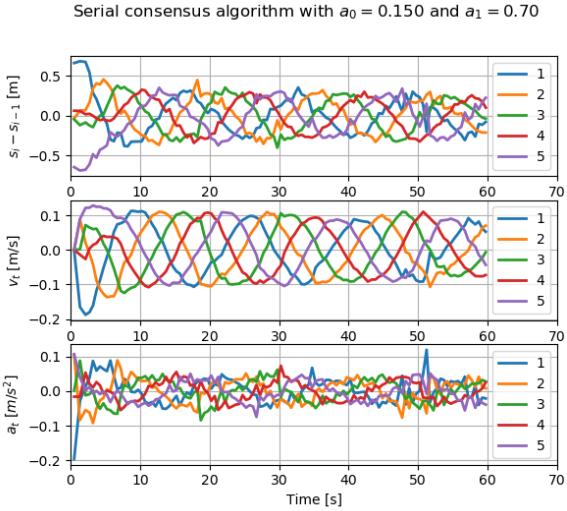

Figure B.24

C String stability simulation results

This appendix visualizes the simulated trajectories to demonstrate the string (in)stability of both protocols that we refer to in Chapter 9. Figure C.1 illustrates how the conventional protocol is not string stable. Furthermore, we show in Figure C.2 that the serial consensus protocol is string stable when $a_1 > 2\sqrt{a_0}$. However, this sting stability is lost when $a_1 < 2\sqrt{a_0}$, which is illustrated by Figure C.3.

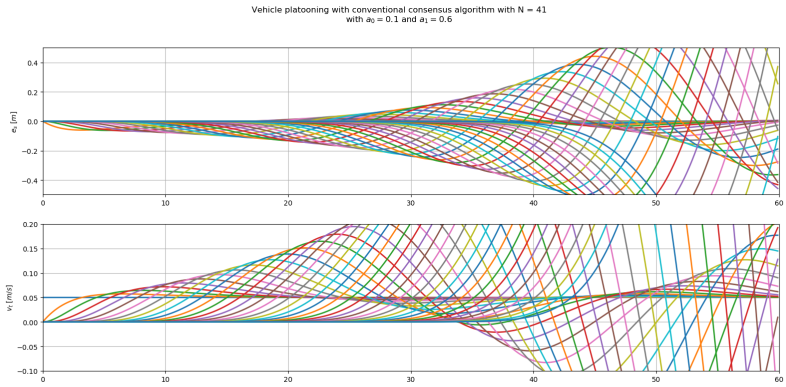

Figure C.1: Simulation of vehicle platoon with conventional consensus protocol, where $a_0 = 0.1$ and $a_1 = 0.6$. The trajectories show that the system is not string stable.

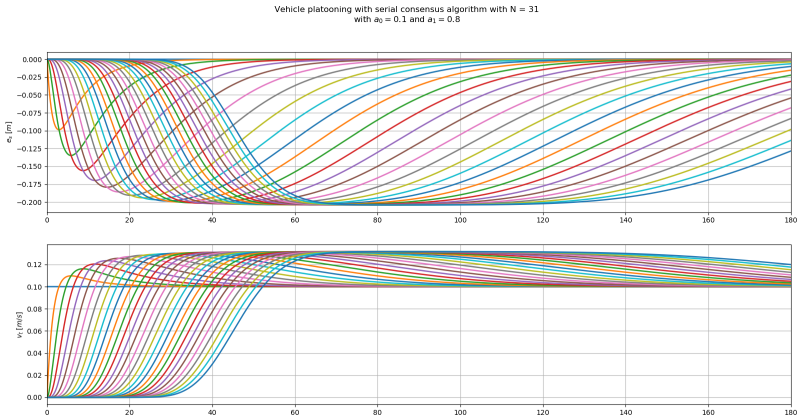

Figure C.2: Simulation of vehicle platoon with serial consensus protocol, where $a_0 = 0.1$ and $a_1 = 0.8$. The trajectories show that the relative position and velocity overshoot remain bounded, independent of the number of agents N.

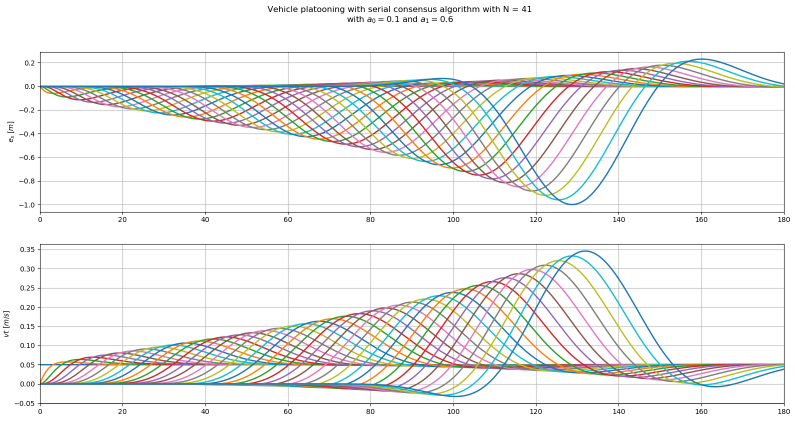

Figure C.3: Simulation of vehicle platoon with serial consensus protocol, where $a_0 = 0.1$ and $a_1 = 0.6$. The trajectories show that the relative position and velocity overshoot do not remain bounded, i.e. the system is not string stable.

D Artificial integration

This appendix illustrates how the artificial integration of the control input (described in Section 7.4) leads to agent dynamics similar to that of an n^{th} order integrator, but not equivalent. More specifically, we derive the error in the update step for the n^{th} order integrator $E_k(k)$. This is a generalization of the result derived in Section 7.4 for the second-order agent dynamics.

We consider the dynamics of a pure n^{th} order integrator, given by $x^{(n)}(t) = u(t)$. The corresponding continuous-time state space reads

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \\ \vdots \\ \dot{\xi}_n \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}}_{A} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{B} u. \tag{D.1}$$

The system is discretized with $\tau(k)$ being the time-dependent sampling time between iteration k and $k+1$. We note that the system matrix A is an $n \times n$ Jordan block with n eigenvalues at zero. Therefore, we can write the discrete-time state space of the pure n^{th} order integrator as

$$\begin{bmatrix} \xi_1(k+1) \\ \xi_2(k+1) \\ \xi_3(k+1) \\ \vdots \\ \xi_n(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \tau(k) & \frac{1}{2}\tau(k)^2 & \dots & \frac{1}{(n-1)!}\tau(k)^{(n-1)} \\ 0 & 1 & \tau(k) & \dots & \frac{1}{(n-2)!}\tau(k)^{(n-2)} \\ 0 & 0 & 1 & \dots & \frac{1}{(n-3)!}\tau(k)^{(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \xi_1(k) \\ \xi_2(k) \\ \xi_3(k) \\ \vdots \\ \xi_n(k) \end{bmatrix} + \begin{bmatrix} \frac{1}{(n-1)!n}\tau^n \\ \frac{1}{(n-2)!(n-1)}\tau^{(n-1)} \\ \frac{1}{(n-3)!(n-2)}\tau^{(n-2)} \\ \vdots \\ \tau \end{bmatrix} u(k). \tag{D.2}$$

However, this is not what is happening physically on the omnibot. On the omnibot, the control input $u(k)$ is integrated $n-1$ times to obtain the velocity $\hat{v}_t(k+n-1)$. This velocity is then used on the omnibot to get the updated position $s(k+n)$. The discrete-time state space describing these dynamics reads

$$\begin{bmatrix} \xi_1(k+1) \\ \xi_2(k+1) \\ \xi_3(k+1) \\ \vdots \\ \xi_n(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \tau(k) & 0 & \dots & 0 \\ 0 & 1 & \tau(k) & \dots & \frac{1}{(n-2)!}\tau(k)^{(n-2)} \\ 0 & 0 & 1 & \dots & \frac{1}{(n-3)!}\tau(k)^{(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \xi_1(k) \\ \xi_2(k) \\ \xi_3(k) \\ \vdots \\ \xi_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{(n-2)!(n-1)}\tau^{(n-1)} \\ \frac{1}{(n-3)!(n-2)}\tau^{(n-2)} \\ \vdots \\ \tau \end{bmatrix} u(k). \tag{D.3}$$

We denote the error in the update step $s(k+1)$ of the physical system w.r.t. a pure n^{th} order integrator by $E_n(k)$. Comparing the discrete-time state space of the physical system and a pure integrator reveals that this error is given by

$$E_k(k) = \frac{1}{(n-1)!n}\tau^n + \sum_{l=2}^{n-1} \frac{1}{l!}\xi_i(k)\tau(k)^l. \tag{D.4}$$