EXPERIMENTAL VALIDATION OF AN $\mathcal{L}_1$ CONTROLLER ON A SINGLE ROBOTIC MANIPULATOR ON A MOVING PLATFORM AND A ROBOTIC COOPERATIVE NETWORK

BY

ANDRES RODRIGUEZ REINA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Harry Dankowicz

Abstract

This thesis reports on laboratory and in-field experimental results for a single robotic manipulator on a moving platform with unmodeled dynamics that aim to validate theoretical predictions for the dependence on control parameters of an $\mathcal{L}_1$ adaptive control strategy. The experiments verify expected bounds on the tracking error in terms of the bandwidth of a filter introduced in the control loop. Moreover, the results provide insight into different discretizations of the continuous-time formulation, suggesting that a partial discretization introduced by Cao and Hovakimyan is most suitable for a hardware implementation. A second set of experimental results, obtained from an implementation of the $\mathcal{L}_1$ control framework for synchronization and consensus in networks of robotic manipulators, similarly validate theoretical predictions on the sensitivity to network communication delays and network topology.

To Julia.

Acknowledgments

This project would not have been possible without the support of many people. Many thanks to my advisor, Harry Dankowicz, who read my numerous revisions and helped make some sense of the confusion, as well as Kim Nguyen, who developed most of the theory I use and got me started in understanding it. Also thanks to my fellow lab students, Yang Li and Christopher Marry, who offered support and guidance whenever I got lost in my journey.

Special thanks to *Fundación "La Caixa"* and their graduate fellowship program for their financial support that has made this research possible, and their strong commitment to achieve a better society for all to live in.

And finally, thanks to Julia, my parents, and numerous friends who endured this long process with me, always offering support and love.

This work was supported by National Institute of Food and Agriculture, U.S. Department of Agriculture, grant number 2014-67021-22109.

Table of Contents

List of Figures

Chapter 1

Introduction and Definitions

1.1 Literature review

Robotic manipulators on dynamic platforms have been proposed for use in agricultural applications such as harvesting cucumbers (van Henten et al., 2009; van Henten et al., 2010), radicchios (Foglia and Reina, 2006), melons (Edan et al., 2000), and peat moss (Johnson et al., 2009). In planetary exploration, autonomous robotic platforms with manipulators cooperate to build structures on unmodeled planetary surfaces (Huntsberger et al., 2003; Stroupe et al., 2005). In the construction field, assistive robots are used to install the hazardous parts of the external wall of tall buildings (Lee et al., 2007). Mobile robots employ vision systems for discriminating weeds from crops (Slaughter et al., 2008), in some cases using manipulators that require an appropriate control.

The technology transition from controlled to uncertain environments requires the use of obstacle detection systems, path planning algorithms, and robust controllers that overcome the uncertain dynamics induced in the platform. For example, work by (Ball et al., 2016) uses vision-based obstacle detection and navigation in farm environments. Extensive reviews of systems for guidance of agricultural autonomous vehicles can be found in the recent literature (Wilson, 2000; Reid et al., 2000; Slaughter et al., 2008; Cheein and Carelli, 2013). Path planning research includes sample-based algorithms for obstacle-dense environments (Karaman and Frazzoli, 2011; Frazzoli et al., 2002; LaValle, 1998) and also exploration algorithms (Yamauchi, 1997). Care must be taken in the control of the moving platform, for example to avoid tipping over in inclined terrain (Morales et al., 2013). This thesis focuses on the adaptive control of manipulators installed on dynamic

platforms that move across uncertain terrain.

The literature on adaptive control of manipulators includes designs based on MRAC (DesForges, 1979; Nicosia and Tomei, 1984; Tomizuka, 1986), the linear-in-parameter property (Craig et al., 1987; Slotine and Li, 1987; Ortega and Spong, 1989), and the passivity of rigid robot dynamics (Niemeyer and Slotine, 1991; Berghuis and Nijmeijer, 1993; Ortega et al., 2002). Control strategies designed principally for either fixed-base or free-floating manipulators exist, e.g., (Antonelli and Leonessa, 2008). This thesis focuses on the adaptive control of manipulators installed on dynamic platforms that are disturbed by the interaction with the environment.

The first part of this thesis is based on (Nguyen and Dankowicz, 2015) which designs a continuous-time $\mathcal{L}_1$ adaptive control scheme to achieve predictable performance with guaranteed robustness for a manipulator mounted on a dynamic platform with unknown dynamics. An important property of this adaptive control scheme is that detailed knowledge of the model of the manipulator-platform is not needed, nor is measurement of the platform motion. The thesis describes the implementation in actual hardware and reports on an extensive data collection aiming to validate predicted performance bounds for the continuous-time control scheme, as well as an alternative discrete-time implementation from (Hovakimyan and Cao, 2010). Other existing experimental research includes (Bennehar et al., 2015), which presents a parallel manipulator using the $\mathcal{L}_1$ controller.

Cooperative control of networked robots is of increasing relevance to industry. The range of applications is vast and includes cooperative robots in planetary missions building structures (Huntsberger et al., 2003; Stroupe et al., 2005), underwater docking of multiple vehicles (Majid et al., 2014), and swarms of terrestrial and aerial vehicles used for inspection of natural disasters or polluted areas (Zhou et al., 2015). Research in cooperative control of networked manipulators includes formulations based on convergence in contraction theories (Chung and Slotine, 2007; Chung and Slotine, 2009), neural networks (Wang, 2013b; Wang, 2013c; Wang, 2013a; Wang, 2014), or adaptive sliding-mode control schemes (Chen, 2015; Zhao et al., 2015). When the cooperating manipulators are mounted on a dynamic platform, the overall system model is complicated

and large perturbations are introduced, making results for fixed platforms challenging to apply.

The second part of this thesis aims to validate the theory in (Nguyen and Dankowicz, 2016) (see also (Nguyen and Dankowicz, 2014; Nguyen and Dankowicz, 2015)), which develops an adaptive controller using the $\mathcal{L}_1$ architecture for the cooperative control of networked manipulators on a moving platform in the presence of communication delays. This controller again needs no detailed knowledge of the system model, while guaranteeing robustness and fast adaptation. Both consensus and synchronization problems are considered. Although the platforms are not translated, and therefore there is no perturbation from the terrain, the manipulators are still influenced by the platform motion induced by the robot actuators.

The remainder of this chapter discusses the $\mathcal{L}_1$ controller formulation, two different implementations of this formulation in hardware, and the experimental setup. Chapter 2 contains a few simulation results showing the effects of discretization of the $\mathcal{L}_1$ controller. Chapter 3 focuses on experimental results for a single robot on a dynamic platform, and experimental validation of the performance bounds. The network implementation and associated experimental results are discussed in Chap. 4. Finally, Chap. 5 closes with some concluding comments.

1.2 $\mathcal{L}_1$ adaptive controller architecture and discretization

This section summarizes the definition and key properties of a family of $\mathcal{L}_1$ controllers. It describes two different discretization of the architecture suitable for implementation in physical hardware. All the controller formulations shown in this section have been developed in the references (Hovakimyan and Cao, 2010) and (Nguyen and Dankowicz, 2015), and are included here for completeness.

1.2.1 Physical system definition

As defined in (Nguyen and Dankowicz, 2015), we assume a physical system consisting of a fully actuated manipulator on an unactuated moving platform governed by the equation

$$\begin{bmatrix} M_{aa}(q) & M_{ap}(q) \\ M_{ap}^T(q) & M_{pp}(q) \end{bmatrix} \ddot{q} + \begin{bmatrix} N_{aa}(q,\dot{q}) \\ N_{pp}(q,\dot{q}) \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix} + \begin{bmatrix} D_{aa}(q,t) \\ D_{pp}(q,t) \end{bmatrix}, \tag{1.1}$$

where q_a represents the generalized degrees of freedom ot robotic manipulator, q_p represents the platform degrees of freedom and $q^T = \left[q_a^T, q_p^T\right]$. The matrices M_{aa}, M_{ap}, and M_{pp} are all positive-definite, symmetric, and bounded. The column vectors N_{aa} and N_{pp} contain the corresponding nonlinearities, the components of u are the time-dependent control input, and D_{aa} and D_{pp} represent the bounded time-dependent unknown disturbances to the manipulator and the platform.

Given a desired trajectory $q_d(t)$ for the system, define a sliding variable

$$r = \dot{q}_a - \dot{q}_d + \lambda(q_a - q_d). \tag{1.2}$$

If the controller is successful in driving $r \longrightarrow 0$, then

$$q_d(t) - q_a(t) \sim e^{-\lambda t}. \tag{1.3}$$

Equation (1.1) can be used to write

$$\dot{r} = A_m r + u + \eta(t, q, \dot{q}, \ddot{q}_p, u), \quad r(0) = r_0. \tag{1.4}$$

where A_m is a Hurwitz matrix that may be designed to achieve a desired behavior of the controlled system, and $\eta(t, q, \dot{q}, \ddot{q}_p, u)$ contains nonlinearities, time dependent disturbances, and the inertias of the system. The objective of the $\mathcal{L}_1$ controller is that the control input u cancels the contributions of η.

1.2.2 $\mathcal{L}_1$ adaptive controller

Like other adaptive controllers, the $\mathcal{L}_1$ adaptive controller is composed of a state predictor, one or several adaptive laws, and a control law (c.f. Fig. 1.1). The state predictor estimates the state of the system, the adaptive laws update the adaptive parameters based on the error and in the state predictor estimation, and the control law computes the control inputs based on the adaptive parameters.

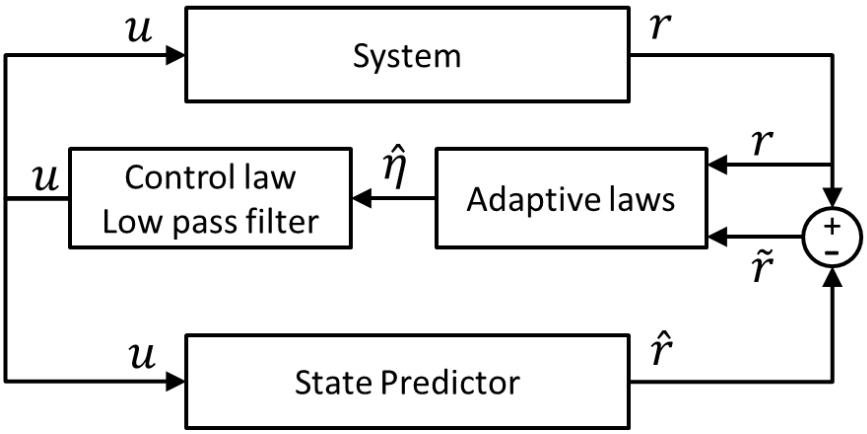


Figure 1.1: Block diagram for an adaptive $\mathcal{L}_1$ controller.

The $\mathcal{L}_1$ adaptive controller is characterized by the use of a low-pass filter in the control law, removing the highest frequencies from the control input. This makes it possible to have very high gains in the adaptation laws, achieving quick adaptation without letting high frequency signals reach the physical system.

The $\mathcal{L}_1$ controller is originally defined as set of continuous-time differential equations and projection operators, as described in Section 1.2.3. These equations need to be discretized in order to be implemented on a microcontroller in an implementation. The reader should realize that a naive discretization is very likely to give numerical problems as the adaptation laws have very high gains. A different approach is given in Section 1.2.5, where the time history of the adaptive parameters over one time step is estimated a priori, eliminating the need to integrate the corresponding equations. Rigorous proofs of the stability of the continuous-time controller is given in (Nguyen and Dankowicz, 2015), and for the discretization from Section 1.2.5 is given in

(Hovakimyan and Cao, 2010), respectively. Key results are summarized in the next sections.

1.2.3 Continuous formulation

This section presents a control framework for the stabilization of the dynamics of a manipulator operating on a moving platform as described in Section 1.2.1, following the $\mathcal{L}_1$ controller architecture described in (Nguyen and Dankowicz, 2015).

The system state is first estimated by the state predictor

$$\dot{\hat{r}} = A_m r + u + \hat{\theta}\|r_{[0,t]}\|_{\mathcal{L}_\infty} + \hat{\sigma} + A_{sp}\tilde{r}, \;\; \hat{r}(0) = \hat{r}_0, \tag{1.5}$$

where $\hat{r}$ is the state estimate, $\tilde{r} = \hat{r} - r$ is the estimation error, A_{sp} is a Hurwitz matrix can be designed to shape the convergence, and $\|r_{[t_0,t_1]}\|_{\mathcal{L}_\infty}$ denotes the $\mathcal{L}_\infty$ norm of $r(t)$ on the interval $[t_0, t_1]$. The adaptive estimates $\hat{\theta}$ and $\hat{\sigma}$ are obtained from the projection-based adaptive laws

$$\dot{\hat{\theta}} = \Gamma\, \mathbf{Proj}\big(\hat{\theta}, -P\tilde{r}\|r_{[0,t]}\|_{\mathcal{L}_\infty}; \theta_b, \varepsilon\big), \;\; \hat{\theta}(0) = \hat{\theta}_0, \tag{1.6}$$

$$\dot{\hat{\sigma}} = \Gamma\, \mathbf{Proj}\big(\hat{\sigma}, -P\tilde{r}; \sigma_b, \varepsilon\big), \;\; \hat{\sigma}(0) = \hat{\sigma}_0, \tag{1.7}$$

in terms of the adaptive gain Γ, the projection tolerance ε, and the positive-definite matrix P, obtained as the solution to the Lyapunov equation $A_{sp}^\top P + P A_{sp} = -Q$, in terms of some chosen positive definite matrix Q. The projection operator $\mathbf{Proj}(\cdot, \cdot\,; \cdot, \cdot)$, as defined in the Appendix of (Hovakimyan and Cao, 2010) and shown below, ensures that $\|\hat{\theta}(t)\|_\infty \leq \theta_b$ and $\|\hat{\sigma}(t)\|_\infty \leq \sigma_b$, provided that $\hat{\theta}_0$ and $\hat{\sigma}_0$ satisfy these same bounds.

$$\mathbf{Proj}(a, y; a_b, \epsilon) = \begin{cases} y & \text{if} \quad f(a) \leq 0 \quad \text{or} \quad \nabla f^\top y \leq 0 \\ y - \frac{\nabla f(a)}{\|\nabla f(a)\|^2}(\nabla f)^\top y(a)) f(a) & \text{if} \quad f(a) > 0 \quad \text{and} \quad \nabla f^\top y > 0 \end{cases} \tag{1.8}$$

where

$$f(a) = \frac{(\epsilon + 1)a^\top a - a_b^2}{\epsilon\, a_b^2} \tag{1.9}$$

Finally, the control law is

$$\dot{u} = -k\big(u + \hat{\theta}\|r_{[0,t]}\|_{\mathcal{L}_\infty} + \hat{\sigma}\big), \;\; u(0) = 0, \tag{1.10}$$

where k is the bandwidth of the low-pass filter rejecting high frequency components from the adaptation laws. For large values of k, u cancels the contribution of $\hat{\theta}\|r_{[0,t]}\|_{\mathcal{L}_\infty} + \hat{\sigma}$, which means that the state predictor is approximately given by $\dot{\hat{r}} = A_m r + u + \hat{\theta}\|r_{[0,t]}\|_{\mathcal{L}_\infty} + \hat{\sigma} + A_{sp}\tilde{r}, \;\; \hat{r}(0) = \hat{r}_0$. If in addition $\hat{\theta}\|r_{[0,t]}\|_{\mathcal{L}_\infty} + \hat{\sigma}$ is a close approximation of η then r would be driven to 0 as desired. This is the trade-off, between perfect cancellation and the introduction of noise in the system and unstabilize the system.

As an illustration of a possible application, Fig. 1.2 shows the framework for using this controller to stabilize the end effector of a robotic manipulator. The platform orientation is measured, and a desired trajectory in joint space is computed to compensate for changes to this orientation. This trajectory is fed into the $\mathcal{L}_1$ controller through the input r. The *Control Law* and *Low-pass Filter* blocks correspond to Eq. (1.10), obtaining u, while the *Predictor* obtains $\hat{r}$ in (1.5), *Adaptation* computes $\hat{\theta}$ and $\hat{\sigma}$ in Eqs. (1.6) and (1.7), and *Tracking Error* does so for r in Eq. (1.2).

The reference (Nguyen and Dankowicz, 2015) establishes a bound on the performance of the controller, under certain operating conditions. To this end, a non-adaptive reference system with states r_{ref} and control input u_{ref} is defined, assuming no model uncertainty. The stability of this reference system follows from Theorem 3.1 of (Nguyen and Dankowicz, 2015). For sufficiently large k, there exists a positive scalar ρ_{ref}, such that $\|r_{ref}(0)\|_\infty < \rho_{ref}$ implies that $\|r_{ref}\|_{\mathcal{L}_\infty} < \rho_{ref}$ and $\|u_{ref}\|_{\mathcal{L}_\infty}$ is finite. Lemma 3.1 of (Nguyen and Dankowicz, 2015) then establishes

$$\left\|r_{ref}(t) - e^{A_m t} r_0\right\|_{\mathcal{L}_\infty} = \mathcal{O}\left(k^{-1}\right), \tag{1.11}$$

where $r_0 = r_{ref}(0)$. This means that, for large enough values of τ_0 or when $r_0 = 0$, $\left\|r_{ref,[\tau_0,\infty]}\right\|_{\mathcal{L}_\infty}$ decreases as $\mathcal{O}\left(k^{-1}\right)$.

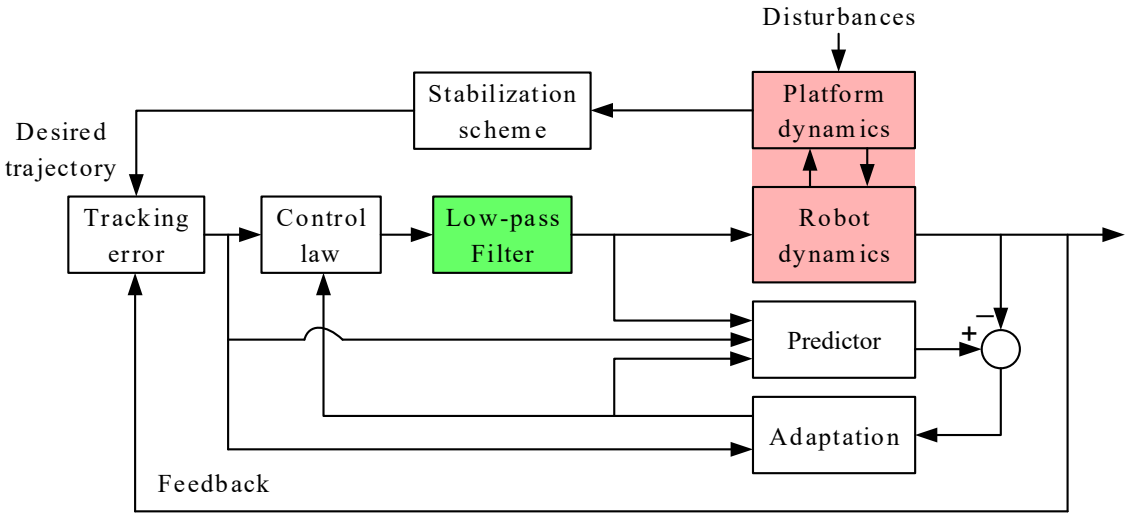


Figure 1.2: Block diagram for the end effector control of a moving-base manipulator.

Similarly, the results in Theorem 4.1 of (Nguyen and Dankowicz, 2015) establish that for the k and r_0 already chosen, and $\lambda \geq 1$, exist θ_b, σ_b, and ϵ such that for sufficiently large Γ, $\|r_{ref} - r\|_{\mathcal{L}_\infty}$ and $\|u_{ref} - u\|_{\mathcal{L}_\infty}$ are order $\mathcal{O}(1/\sqrt{\Gamma})$, proving the stability of the adaptive controller under those conditions. Similarly, Theorem 2 in (Nguyen et al., 2016) suggests that under equivalent conditions, there exist β_1, β_2, and β_3 such that

$$\|(r_{\text{ref}} - r)_{[\tau_0,\infty]}\|_{\mathcal{L}_\infty} \leq \beta_1 \|\tilde{r}_0\| \mathrm{e}^{-\beta_2 \tau_0} + \beta_3/\sqrt{\Gamma} \tag{1.12}$$

for arbitrary τ_0, and nonzero initialization errors $\tilde{r}_0$.

In the case that $\hat{r}_0 = r_0 = 0$, Eqs. (1.11) and (1.12) imply the following asymptotic estimate is obtained:

$$\|r\|_{\mathcal{L}_\infty} \leq \mathcal{O}(1/k) + \mathcal{O}(1/\sqrt{\Gamma}). \tag{1.13}$$

This result will be investigated in the experiments in the following chapters.

1.2.4 Euler-like discrete formulation

For an experimental implementation of the proposed control architecture, consider the following discretization of Eqs. (1.5-1.10) in terms of the sampling interval T:

$$\hat{r}^{(i+1)} = \hat{r}^{(i)} + T\big(A_m r^{(i+1)} + u^{(i)} + \hat{\theta}^{(i)} \max_{0\le j\le i} \|r^{(j+1)}\|_\infty + \hat{\sigma}^{(i)} + A_{sp}(\hat{r}^{(i)} - r^{(i+1)})\big), \tag{1.14}$$

$$\hat{\theta}^{(i+1)} = \mathbf{Clamp}\Big(\hat{\theta}^{(i)} + T\Gamma\,\mathbf{Proj}\big(\hat{\theta}^{(i)}, -P(\hat{r}^{(i+1)} - r^{(i+1)}) \max_{0\le j\le i} \|r^{(j+1)}\|_\infty; \theta_b, \varepsilon\big), \theta_b\Big), \tag{1.15}$$

$$\hat{\sigma}^{(i+1)} = \mathbf{Clamp}\Big(\hat{\sigma}^{(i)} + T\Gamma\,\mathbf{Proj}\big(\hat{\sigma}^{(i)}, -P(\hat{r}^{(i+1)} - r^{(i+1)}); \sigma_b, \varepsilon\big), \sigma_b\Big), \tag{1.16}$$

$$u^{(i+1)} = u^{(i)} - Tk\big(u^{(i)} + \hat{\theta}^{(i+1)} \max_{0\le j\le i} \|r^{(j+1)}\|_\infty + \hat{\sigma}^{(i+1)}\big) \tag{1.17}$$

where $\hat{r}^{(0)} = \hat{r}_0$, $\hat{\theta}^{(0)} = \hat{\theta}_0$, $\hat{\sigma}^{(0)} = \hat{\sigma}_0$, and $u^{(0)} = 0$, and the clamping operator

$$\mathbf{Clamp}(x, x_b) := \begin{cases} x & \text{if} \quad \|x\| \le x_b \\ x_b\, x/\|x\| & \text{if} \quad \|x\| > x_b \end{cases} \tag{1.18}$$

ensures that the implementation respects the bounds on $\hat{\theta}$ and $\hat{\sigma}$.

Note that although this discretization may appear similar to the explicit Euler discretization, it differs in two ways. First, the measured value of $r^{(i+1)}$ is used to update the control state to time $i+1$. Secondly, the equations are updated sequentially: $\hat{r}^{(i+1)}$ is used to update $\theta^{(i+1)}$ and $\sigma^{(i+1)}$, and these are next used to update $u^{(i+1)}$.

The experimental results in Chap. 3 show that the bound in Eq. (1.13) applies to this implementation, although the discretization error renders the system unstable for large values of k.

1.2.5 The Cao and Hovakimyan discretization

The reference (Cao and Hovakimyan, 2009) (see also Section 3.3 of (Hovakimyan and Cao, 2010)) introduces an alternative formulation of the differential adaptive laws in terms of a piecewise constant signal with explicit updates that depend only on the prediction error. This removes all the

complexity introduced by the high adaptation gain and the projection operators. A possible implementation is as follows, in terms of the sampling interval T:

$$\hat{r}^{(i+1)} = \hat{r}^{(i)} + T\big(A_m r^{(i+1)} + u^{(i)} + \hat{\sigma}^{(i)}\big), \tag{1.19}$$

$$\hat{\sigma}^{(i+1)} = (A_m^{-1}(e^{-A_m T} - 1))^{-1} e^{-A_m T}(\hat{r}^{(i+1)} - r^{(i+1)}), \tag{1.20}$$

$$u^{(i+1)} = u^{(i)} - Tk\big(u^{(i)} + \hat{\sigma}^{(i+1)}\big), \tag{1.21}$$

where $\hat{r}^{(0)} = \hat{r}_0$, $\hat{\sigma}^{(0)} = 0$ and $u^{(0)} = 0$. We refer to this discretization below as the Cao and Hovakimyan discretization.

Note that the term $(A_m^{-1}(e^{-A_m T} - 1))^{-1} e^{-A_m T}$ is a result of an assumed relationship between Γ and T. The proof of stability and the performance bounds in Section 3.3 of (Hovakimyan and Cao, 2010) generalize the observations made for the continuous-time formulation. For example, Theorem 3.3.1 establishes that $\|r - r_{ref}\|_{\mathcal{L}_\infty}$ can be made arbitrarily small by picking sufficiently small T and sufficiently large k.

As will be observed in the sections describing the physical experiments, this implementation presents several advantages when implemented on a real system. These include reduced complexity, fewer parameters to adjust and the less numerical stiffness.

1.2.6 Network of manipulators

Following (Nguyen and Dankowicz, 2016), a network of manipulators is defined by a weighted directed graph. Each manipulator i is represented by a node of the graph. It feeds its measured data α_i to the manipulators at the other ends of its outgoing edges. Let the set of incoming edges of each manipulator i be denoted by $\mathcal{S}_i$. Then,the signal β_i, given by

$$\beta_i(t) = \sum_{j \in \mathcal{S}_i} a_{j,i}(\pm\alpha_j^*(t) - \alpha_i(t)), \tag{1.22}$$

is available for the controller governing the ith manipulator. Here, $a_{j,i}$ is the weight of the edge from j to i. In the case of a bipartite consensus, edges can be positive or negative, as denoted by the symbol $\pm$. Fig. 1.3 shows an example of a positive-only network. For convenience, let $a_{i,j} = 0$ if there is no edge from manipulator i to j.

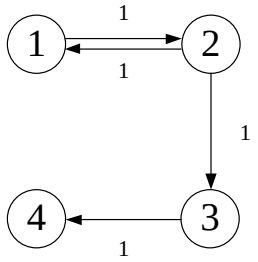


Figure 1.3: Example of a manipulator network, depicted as a directed graph. Numbers next to edges represent weights.

The notation $\alpha_j^*(t)$ denotes a signal received from the ith robot at time t. This corresponds to an earlier measurement whose later reception is caused by an potentially unknown network communication delay. In (Nguyen and Dankowicz, 2016), this delay is assumed to be the same in all edges.

1.2.7 Extension to synchronization before tracking

The synchronization-before-tracking strategy in (Nguyen and Dankowicz, 2016) can be implemented using the Cao and Hovakimyan discretization of the $\mathcal{L}_1$ controller from Section 1.2.5,

together with Eq (1.2):

$$r_i^+ = \dot{q}_{a,i}^+ - \dot{q}_{d,i}^+ + \lambda(q_{a,i}^+ - q_{d,i}^+), \tag{1.23}$$

$$\hat{r}_i^+ = \hat{r}_i + T\big(A_{m,i} r_i^+ + u_{a,i} + \hat{\sigma}_i\big), \tag{1.24}$$

$$\hat{\sigma}_i^+ = (A_{m,i}^{-1}(e^{-A_{m,i}T} - 1))^{-1} e^{-A_{m,i}T}(\hat{r}_i^+ - r_i^+), \tag{1.25}$$

$$u_{a,i}^+ = u_{a,i} - Tk\big(u_{a,i} + \hat{\sigma}_i^+\big) \tag{1.26}$$

$$u_{s,i}^+ = \sum_{j \in \mathcal{S}_i} a_{j,i} \left(r_j^* - r_i^+\right), \tag{1.27}$$

$$u_i^+ = u_{a,i}^+ + u_{s,i}^+ \tag{1.28}$$

for each manipulator i, where a $+$ superscript represents the new values of each variable in the currently computed time step, r_j^* is the delayed signal received from robot j, and $\hat{r}_i^{(0)} = \hat{r}_{0,i}$ and $u_{a,i}^{(0)} = 0$. The synchronization signal $u_{s,i}^+$ (c.f. Eq. (1.22), with $\alpha_j = r_j$ and using only positive edges) is added to the control input $u_{a,i}^+$ to obtain the total control input u_i^+.

For the corresponding continuous-time system, Theorems 1 and 2 and Corollary 1 in (Nguyen and Dankowicz, 2016) establish a bound similar to (1.13):

$$\|r - \bar{r}(t, r_0)\|_{\mathcal{L}_\infty} \leq \mathcal{O}(1/k) + \mathcal{O}(1/\sqrt{\Gamma}) \tag{1.29}$$

where $\bar{r}(t, r_0)$ is bounded by an exponentially decaying function. The experiments in this thesis will confirm that tracking can happen even with large network delays. Proposition 1 in (Nguyen and Dankowicz, 2016) shows that, for the case of perfect cancellation, it is possible to design $A_{m,i}$ so that $r = 0$ is globally asymptotically stable independently of the communication delay. Proposition 2 in the same paper shows that, for the case with no delay, it is possible to design $A_{m,i}$ to achieve synchronization before tracking. In other words, $r_i \to r_j$ faster than $r_i \to 0$, for all pairs of robots i, j, and generic initial conditions. In the experiments in Chap. 4 several combinations of $A_{m,i}$ values will be tested to validate this predictions. Finally, a stability map similar to that

reported in (Nguyen and Dankowicz, 2016) will be experimentally characterized.

1.2.8 Extension to consensus

Next, consider the consensus strategy in (Nguyen and Dankowicz, 2016). In this case, Eq. (1.23) is replaced by

$$r_i^+ = \dot{q}_{a,i}^+ - \sum_{j \in \mathcal{S}_i} a_{j,i} \left(\pm q_{a,j}^* - q_{a,i}^+ \right) + c_i (q_d - q_{a,i}) \tag{1.30}$$

and Eq. (1.28) is replace by

$$u_i^+ = u_{a,i}^+. \tag{1.31}$$

This notation is consistent with (1.22) with $\alpha_j = q_{a,j}$. The coefficients $c_i = 0$ for all i, except for the leader manipulator that seeks to track the desired trajectory q_d. When there is no leader, $c_i = 0$ for all i and the manipulators will seek a dynamically determined consensus.

In the case of the continuous-time formulation, Theorems 3 and 4 in (Nguyen and Dankowicz, 2016) show that $A_{m,i}$ and k exist so that

$$\| r_i - \bar{r}(t, r_0) \|_{\mathcal{L}_\infty} \leq \mathcal{O}(1/k) + \mathcal{O}(1/\sqrt{\Gamma}) \tag{1.32}$$

where $\bar{r}(t, r_0)$ is bounded by an exponentially decaying function, and $\| u_i \|_{\mathcal{L}_\infty}$ is bounded for all robots. This relationship will be explored for different network topologies in the experiments in Chap. 4.

Eq. (1.30) allows for the possibility of negative edges. As predicted by (Nguyen and Dankowicz, 2016) and as will be explored experimentally, a bipartite consensus can occur is such a graph if it is structurally balanced. In this case, there are two groups of nodes with all inter-connections negative and all intra-connections positive. The consensus values of one group are the negatives

of the consensus values of the other group.

1.3 Implementation

1.3.1 Hardware setup

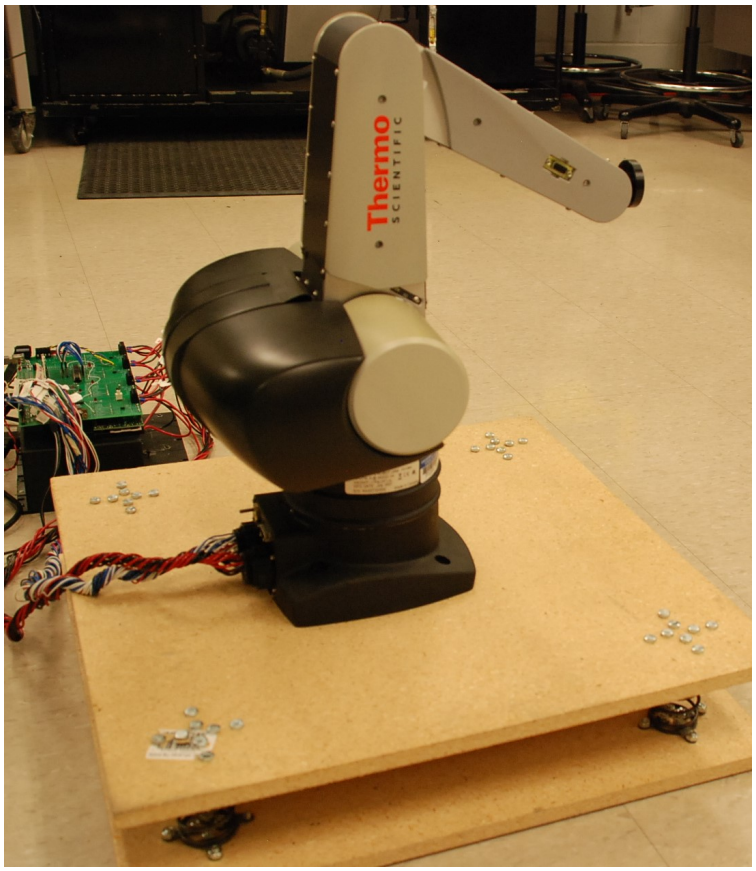


Figure 1.4: One of the robotic manipulators used in the experiments is mounted on a wooden platform that is suspended relative to the floor via four coil springs at its corners.

The manipulator used in all experimental setups is a CataLyst-5 from Thermo Scientific, mounted on different platforms, such as the one in Fig. 1.4. In the network setups, several identical manipulators are used, mounted on different independent platforms. The CataLyst-5 manipulator is a standard five-degrees-of-freedom robotic arm, which uses three degrees of freedom for determining the position of the end effector and three for its orientation (the first degree of freedom

affects both). The motor control of the robotic arm is realized using chain and gear mechanisms driven by DC motors, as shown in Fig. 1.5. The torques generated by the motors are modulated by a PWM duty cycle of the input current, parameterized by u. Optical encoders measure rotations of the chain sprockets corresponding to the angles of each link relative to an absolute, base-fixed reference frame, rather than relative angles between links at each joint (see Section 1.3.2).

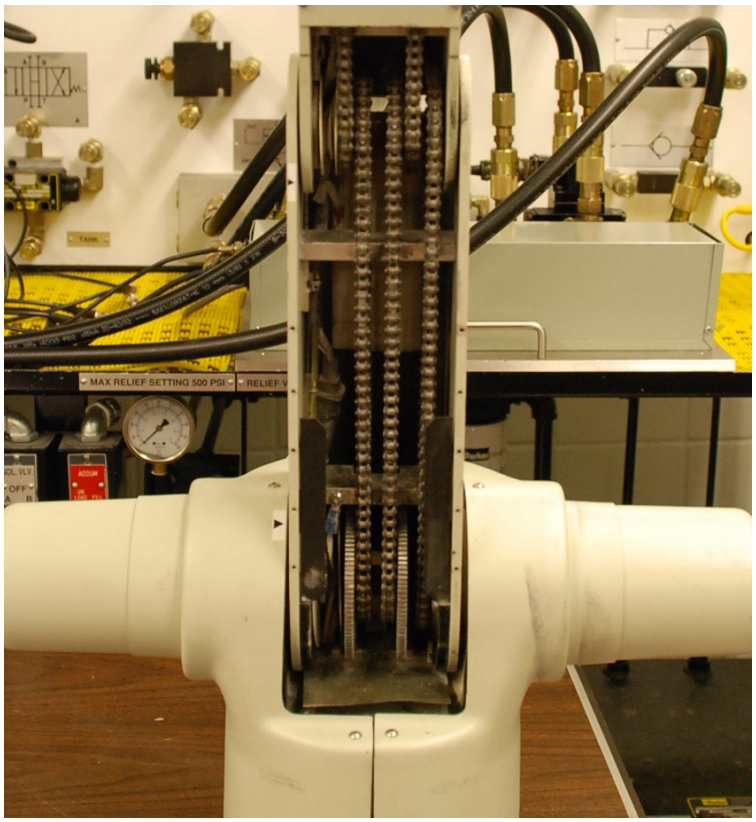


Figure 1.5: Chain and gear mechanism that transmits the motor torques to the manipulator joints.

A control PCB is designed with a Texas Instruments TMS320F28335 microprocessor and 5 Advanced Motion Controls ZBDC12A PWM amplifiers to control the motors. The measurements from the optical encoders are acquired through an I^2C bus for use in the control feedback. The sampling and control frequency of the microprocessor is set to 1000 Hz, implying that the sampling interval $T \geq 0.001$ s in all experiments. The system is also outfitted with an ADXL335 3-axis accelerometer (as a part of a Sparkfun SEN-10736 Inertial Measurement Unit), which computes the platform orientation.

In the experimental implementation of the adaptive control design, the geometry and mass distribution of the manipulator are assumed to be unknown, as is also the case with the platform dynamics. The transfer function from the duty cycle to the actual torque at the motor shafts is also unmodeled.

1.3.2 Robotic arm kinematics

The CataLyst-5 manipulator is designed so that its degrees of freedom are as independent from each other as possible. For this purpose, the actuated and measured degrees of freedom of the robot are attached to five chains that run inside the links. Specifically, with reference to Fig. 1.6, let α_i, β_i, and γ_i be the ZYX Euler angles of link i with respect to the base frame (i.e., the platform), corresponding to yaw, pitch and roll, respectively, for each of the three links. Then $q_{(a,1)} = \alpha_1$, $q_{(a,2)} = \pi/2 + \beta_1$ and $q_{(a,3)} = \beta_2$. Indeed, this implies that the third joint's relative-to-previous-link angle, which results from the difference of chains 3 and 2, equals $q_{(a,3)} - q_{(a,2)} + \pi/2$. Similarly, $q_{(a,4)} = -\beta_3$, so that the fourth joint's relative-to-previous-link angle is $-q_{(a,4)} - q_{(a,3)}$. Finally, $q_{(a,5)} = \gamma_3 - 2\beta_3$.

Due to this arrangement of the degrees of freedom, the end effector orientation only depends on three degrees of freedom, namely $q_{(a,1)}$, $q_{(a,4)}$, and $q_{(a,5)}$. Notably, the yaw, pitch, and roll angles of the end effector relative to the base-fixed reference frame are given by $q_{a,1}$, $q_{a,4}$, and $q_{a,5} - 2q_{a,4}$, respectively.

1.3.3 Network setup

Three CataLyst-5 manipulators were available for the network experiments. The network was initially designed over TCP-IP WiFi, using a Raspberry PI 3 as a serial port relay for each of the robots. A Python3 program serves the relay function, together with another Python3 program running in the operation control laptop, which computes the data sent to each manipulator.

One of the main limitations in the system is the data transmission rate achievable by the serial

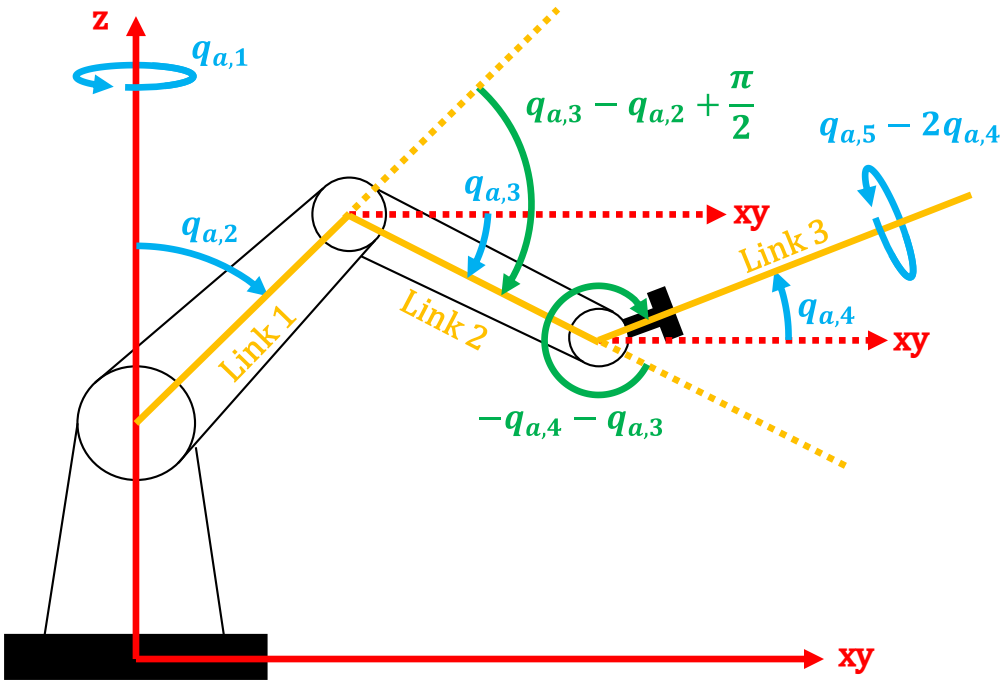


Figure 1.6: Schematic of the robot links and the degrees of freedom as measured and actuated by the hardware. Note that all the angles are absolute, not referred to the previous link.

port through an FTDI system, which is limited to approximately 10 real values every 16 milliseconds using the maximum available baud rate. This maximum throughput has been determined experimentally; sending data at another frequency makes it arrive at multiples of 16 ms. Sending more data or sending it more often results in the system collecting large chunks of data and sending them in one transaction, every few hundred milliseconds, rendering the system unsuitable for real time application. In practice, this means that only two to three values per joint can be transmitted every 16 ms and the minimum delay is unknown and on the order of 16 ms. To reduce the amount of data transmission needed, the network data is preprocessed by the operation control laptop and only the sun in Eq. (1.22) is transmitted. The processing pattern in the operation control laptop simulates the different network topologies, by adding up and sending only the inbound connections for each of the manipulators.

Another limitation in the network implementation comes from the use of TCP-IP as communication protocol. Although TCP provides reliable communication, due to its design, the possible network delays are not bounded, and experiments carried out in the lab show that the nominal net-

work delay is around 20 ms, but it can be larger in some cases of normal operation, depending on the network load; it can even be several seconds if connection is lost and needs to be reestablished. A major issue with TCP is the protocol overhead when sending small messages very often that can collapse the transmission medium. Standard algorithms to avoid this are active in the TCP stack, such as Nagle's algorithm (Nagle, 1984), which has the terrible side effect of accumulating a certain number of messages before sending a TCP packet with them, as seen in (Minshall et al., 2000). This can be disabled by using the `TCP_NODELAY` flag, reducing the trip time but increasing the network load. Although the full network implementation has been tested, in the experiments in Chap. 4 the TCP part of the network has been bypassed by using a direct USB connection to a single computer, avoiding the uncertainty in the delay produced by it in the experiments.

1.3.4 Graphical user interface

A Graphical User Interface (GUI) has been designed for use in the network experiments, allowing the modification of parameters just by clicking in buttons and typing in a textbox. Figure 1.7 depicts the design, in which the left panels contain information about the status of the network and the controllers, and the right side allow for the modification of parameters and sending commands to the robots.

The interface has been designed in Python3 with clearly labeled controls. The top toolbar allows for easy execution of sequential experiments and logging of data for subsequential analysis. The data can be plotted instantaneously with a single click. The safety features include the ability to stop all robots with one click, as well as to trigger an automatic shutdown if connection is lost to the receiver.

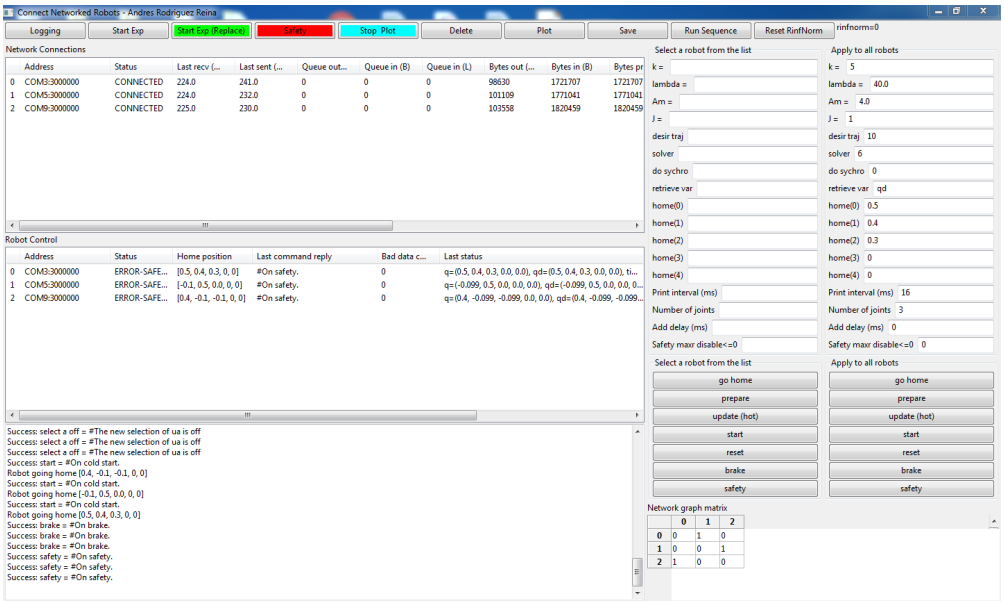


Figure 1.7: Graphical User Interface (GUI) for the network configuration and data collection.

Chapter 2

Simulation

This chapter explores some issues related to the controller discretization introduced in Chap. 1.1. The analysis may be useful for building intuition in advance of the experimental study reported in later chapters. A simplified second-order plant with only one degree of freedom is used. Simulations are performed in Simulink and the implementation is based on the examples provided in (Hovakimyan, 2017).

2.1 Choosing a discretization

The choice of discretization is critical for the solution of systems of ordinary differential equations (ODEs). There is extensive literature about numerical integration of ODEs and the different solvers available (Moin, 2010). There are two relevant properties of a numerical solver, *precision* and *stability*. High precision does not always imply stability and, conversely, some stable methods might not be very precise. The usual choice for the solution of ODEs is the Dormand-Prince (4,5) pair, based on a fifth-order explicit Runge-Kutta formula (Dormand and Prince, 1980). This is implemented in the function *ode45* in MATLAB, which uses an adaptive step size. In most cases, this function delivers very accurate results. It has been used to generate the *continuous solution* referenced in this chapter by solving for the time history of the internal state of the controller, as well as the state of the robot mechanism.

In a hardware implementation, the state of the robot mechanism is an exogenous signal used to drive the computation of the control input. In the typical case of the discretization of an ODE, the value of the exogenous signal at time $t - \delta t$ and the state at that time are used to compute

the state at time t. In contrast, in the discretization of a controller applied to a physical system, a measurement of the plant at time t and the internal state of the controller at $t - \delta t$ may be used to compute the internal state of the controller at time t and the control input at that time. The control input is the held fixed until the next time step.

In the presence of an exogenous signal, the implementation of an ODE solver for a controller in real hardware is constrained by the hardware measurement sampling rate. For instance, consider the state predictor in Eq. (1.5). Here, the exogenous signal r contains the joint angle measurements and velocity estimates, which are only available every 1 ms, when a new encoder measurement happens. Evaluation of the ODE between measurement times requires interpolation which may have an effect on the precision of the integrator. For this reason, we restrict ourselves to fixed-step-size solvers with no evaluation between measurements. Even in this case, numerical stability problems may arise when the derivatives are large compared to the step size.

As seen in the experimental results reported in this thesis, the combination of the ODE integration method and the controlled hardware system may results unstable dynamics. Two very different instability phenomena observed in the experiments. The first instability occurs when the value of k is too small. This is a possible violation of the stability condition in Theorem 3.1 of (Nguyen and Dankowicz, 2015), which requires k to be larger than some critical value. In this case, the system becomes unresponsive, with occasional large overshoots that eventually bring the robot out of safe operation. The second instability occurs when either k or T is too large. This instability is not predicted by the analysis of the continuous system. It is likely due to the effects of the discretization, as the derivatives are very large in these cases, compared to the sampling interval. The system progressively becomes oscillatory as k or T increase, making loud noises and eventually shaking the platform.

Although not reported here, other methods of discretization that those introduced in chapter 1 have been implemented and tested in experiments. These include the explicit Euler method, and the Bogacki-Shampine third-order explicit Runge-Kutta method from (Bogacki and Shampine, 1989), both with constant step size. The Euler method gave very similar results to those obtained

with the discretization in Section 1.2.4. To evaluate the ODE in between measurement sampling times in the Bogacki-Shampine, both interpolated measurements and a step size equal to three times the sampling rate were used. No notable qualitatively improvement was observed in spite of the increase in the implementation complexity.

2.2 Continuous formulation and Euler discretization

Figure 2.1 shows the Simulink model used for the simulation of the formulation in Section 1.2.3. The blocks *Control Law*, *Adaptive Law* and *Predictor* correspond to the Eqs. (1.5 - 1.10). The block *Sliding Window* is Equation (1.2) and *RinfNorm* computes $\|r_{[0,t]}\|_{\mathcal{L}_\infty}$. The block *Platform* is depicted in Fig. 2.2 and corresponds to

$$\ddot{q} = -0.05\ \dot{q} - 0.1\ q + u. \tag{2.1}$$

The desired trajectory is a sinusoid with amplitude 2 and frequency $1\ \mathrm{rad}/s$. The values used for the controller parameters are $A_m = -10$, $A_{sp} = -0.1$, $\lambda = 1$, $P = 1$, $\Gamma = 2000$ and $\sigma_b = \theta_b = 60$. The simulation was run for the continuous-time case (integrated using *ode45*) and in two discrete cases, using the explicit Euler, with time steps of 0.001 and 0.0005, respectively. In each case, the integrator was used for both the platform and the controller.

Figure 2.3 shows the dependence of $\|r\|_{\mathcal{L}_\infty}$ on the bandwidth k for each discretization. We observe that, in the case of integration with *ode 45*, $\|r\|_{\mathcal{L}_\infty}$ always decreases with increasing values k. In contrast, for the Euler integrator, the error quickly diverges beyond a critical value of k. As the step size increases, this critical value of k decreases. A similar divergence is observed in the experimental results in the next chapters.

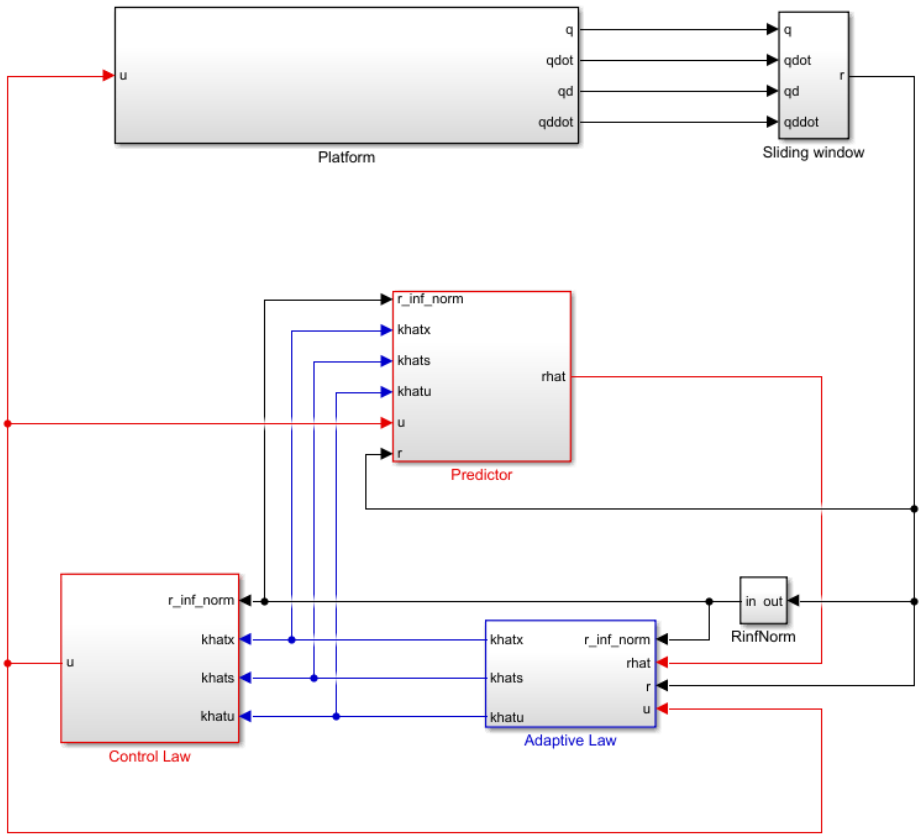


Figure 2.1: Schematic of the Simulink simulation model used.

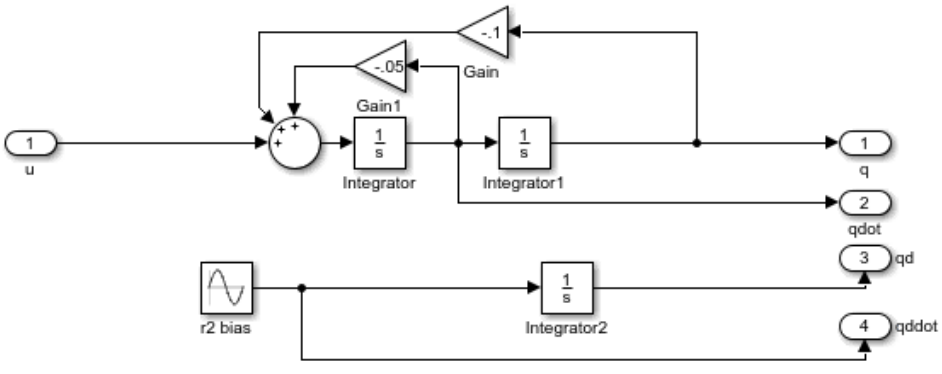


Figure 2.2: Schematic of the simulated Platform block.

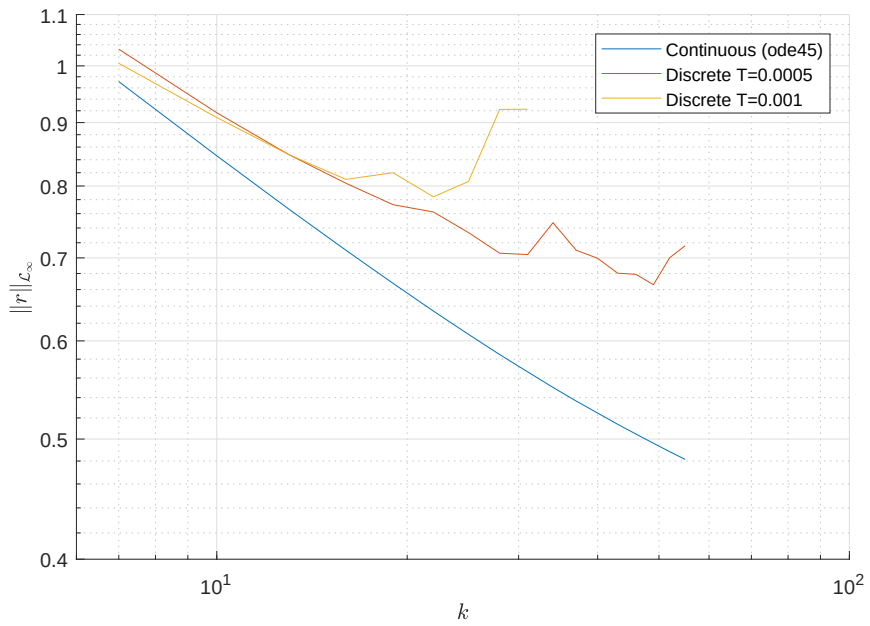


Figure 2.3: Simulation results. The value of $\|r\|_{\mathcal{L}_\infty}$ is plotted for each value of k. Each curve represents a different discretization method, including *ode45* and Euler with integrating time steps of 0.001 and 0.0005. In the discrete cases, the curve terminates at the value of k for which the response diverges.

2.3 Cao and Hovakimyan formulation

Figure 2.4 shows the Simulink model used for the simulation of the formulation in Section 1.2.5. The blocks *Control Law*, *Adaptive Law*, and *Predictor* correspond to the Eqs. (1.19 - 1.21), and the block *Sliding Window* is Eq. (1.2). The block *Platform* and the desired trajectory are the same as in the previous section. The values used for the controller parameters are $A_m = -10$ and $\lambda = 1$. The simulation was run for three discrete cases, with integrating time steps of $T = 0.001$, 0.005 and 0.010.

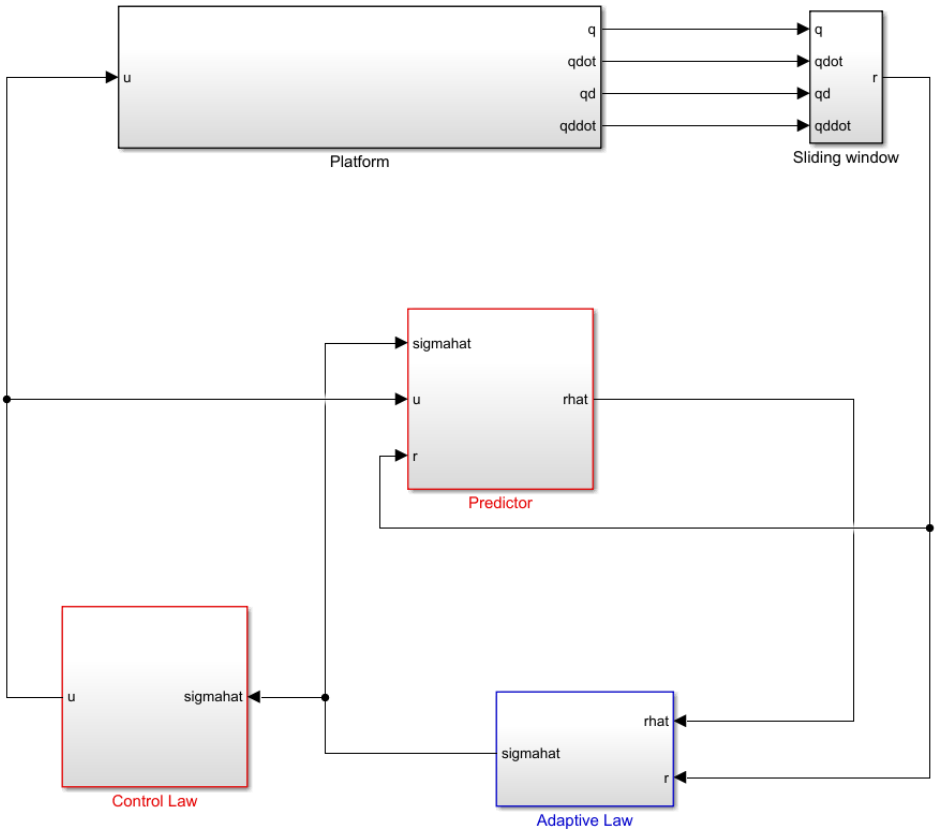


Figure 2.4: Schematic of the Simulink simulation model used for the Cao and Hovakimyan discretization.

Figure 2.5 shows the dependence of $\|r\|_{\mathcal{L}_\infty}$ with k and each discretization. We observe that the error decreases with increasing values of k until a critical value is reached, beyond which the error quickly raises and diverges. The reader should note that the Cao and Hovakimyan formulation

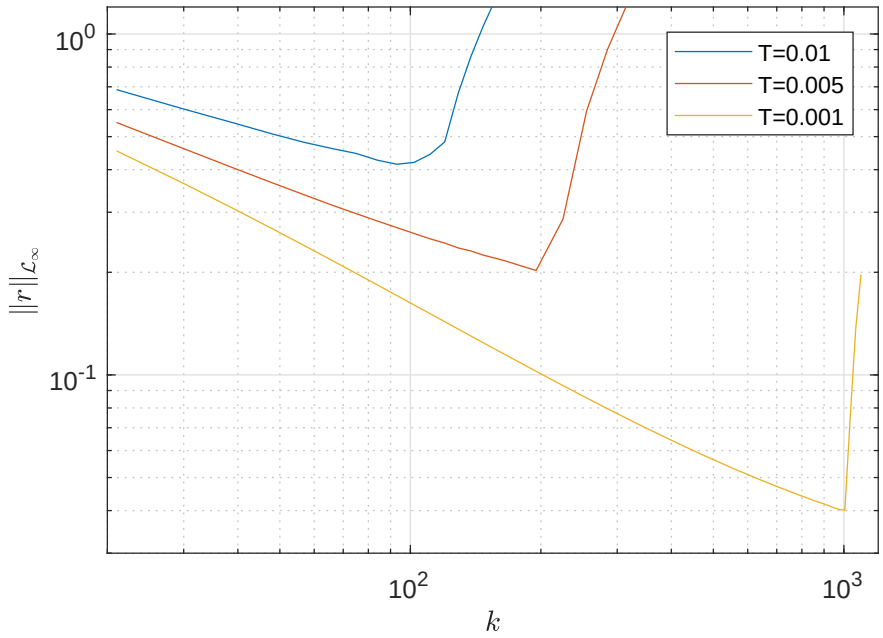


Figure 2.5: Simulation results. The value of $\|r\|_{\mathcal{L}_\infty}$ is plotted for each value of k. Each curve represents a different discretization time step.

remains stable for values of k and time step size T ten times higher than the Euler discretization, which suggests that this method is more suited to the integration of the $\mathcal{L}_1$ controller in fixed-time-step-size systems.

Chapter 3

Field Robotics Experimental Results

A possible application of robotic manipulators installed on dynamic platforms is manipulation of equipment or interaction with the environment while driving over uneven terrain. For example, a vehicle designed for crop inspection could carry a manipulator with the inspection equipment installed on the end effector. The experiments in this section have been designed to validate the control strategies for such an application. Section 3.1 explores the implementation and performance of the Euler-like controller discretization defined in Section 1.2.4, while Section 3.2 does so with the Cao and Hovakimyan discretization defined in Section 1.2.5, better suited for digital control applications. Finally, Section 3.3 compares the two proposed strategies and discusses the lessons learned.

3.1 Validation of the Euler-like discretization scheme

In the following subsections, we use the Euler-like discretization of the continuous controller from Section 1.2.4 to experimentally validate the performance bounds expected for the continuous formulation of the $\mathcal{L}_1$ controller from Section 1.2.3. As shown below, the experimental results validate the theory within a particular region of parameter space, while discretization and measurement errors result in a violation of the expected bounds outside of this region.

3.1.1 Experimental set-up

Three experiments were carried out both in the field and in the lab, comparing the controller capabilities in each case. In these experiments, the control framework is applied to the five-degree-

of-freedom manipulator, described in Section 1.3.2, mounted on a wheeled platform as shown in Fig. 3.1. In the lab, the platform is stationary. When the experiments are carried out in the field, an operator pulls and pushes the platform in a continuous movement along an approximately straight line across an 8 meter long sloping and bumpy terrain (Fig. 3.2). This motion of the platform disturbs the dynamics of the manipulator.

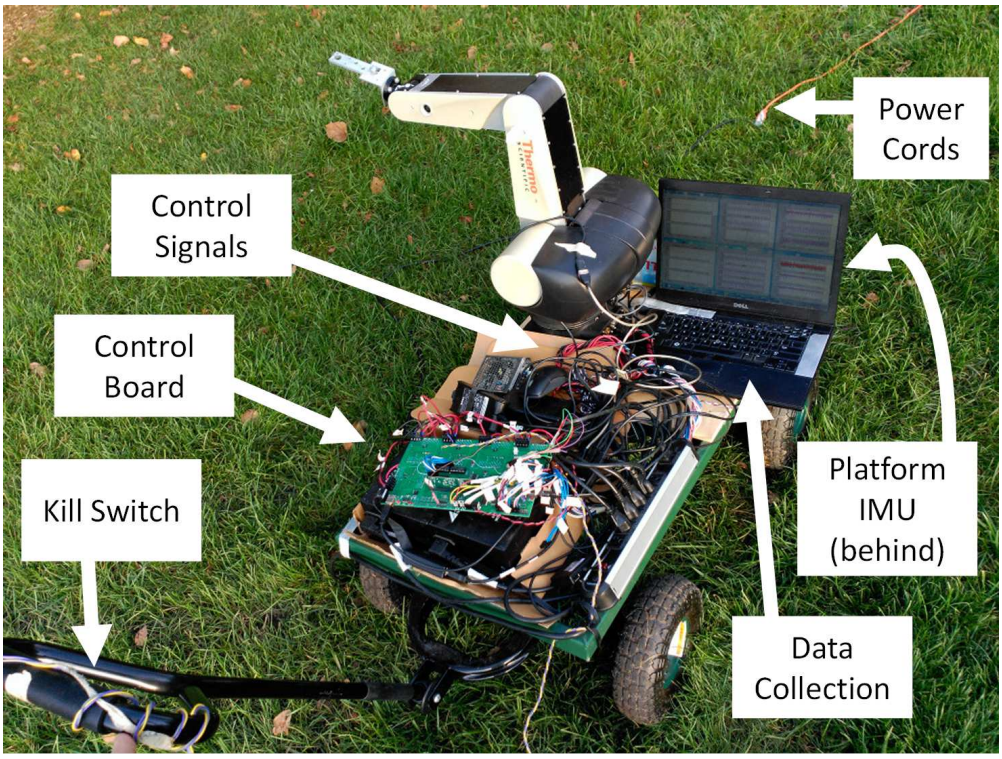


Figure 3.1: The robotic manipulator used in the experiments in Chapter 3 is mounted on a wheeled platform. Here, measurements from the platform IMU and the robot optical encoders are processed by the control board and used to compute the desired trajectory $q_d(t)$.

A subset of the experiments requires measuring the platform orientation in order to compute a desired configuration of the end effector. In these cases, the orientation is measured every 20 ms by an attached Sparkfun SEN-10736 9DOF Razor Inertial Measurement Unit (IMU). The IMU is composed of a triple-axis accelerometer, a triple-axis gyroscope, and a triple-axis magnetometer. The IMU communicates with the manipulator control board via a level converter board that interfaces between the UART port on the IMU and the RS-232 port on the manipulator control board.

Figure 3.2: An operator pulls and pushes the cart continuously in a straight line through a sloped and bumpy grass field about 8 meters long. In the image the sloped terrain and the cart are visible.

Calibration of the magnetometer is crucial. The estimation of the magnetic field is highly sensitive to the surrounding magnetic disturbances, such as parts made of iron and electronic components. We apply the sensor-fusion algorithm presented in (Mahony et al., 2008), and available in the software package Razor AHRS v1.4.2, to estimate the yaw, pitch, and roll angles of the platform. For large angles of rotation, uncorrected offsets in the gyro readings on the order of a few degrees per second result in drifts during integration of the angular velocity signal. To compensate for such drifts, a PI control algorithm may be used to drive the predicted yaw, pitch, and roll angles to values that are consistent with estimated values from the magnetometer and accelerometer. In our implementation, we use PI gains equal to a tenth of their default values in the Razor AHRS v1.4.2 package.

3.1.2 Proposed adaptive controller implementation

The discrete implementation of the proposed adaptive controller is shown in Eqs. (1.14 - 1.17) using the tracking error

$$r^{(i)} = \dot{q}_a^{(i)} - \dot{q}_d^{(i)} + \lambda(q_a^{(i)} - q_d^{(i)}), \tag{3.1}$$

where $q_d^{(i)} = q_d\big(iT\big)$ and $\dot{q}_d^{(i)} = \dot{q}_d\big(iT\big)$. Here $q_a^{(i)} = \left(q_{a,1}^{(i)},\, q_{a,2}^{(i)},\, q_{a,3}^{(i)},\, q_{a,4}^{(i)},\, q_{a,5}^{(i)}\right)^\top$ consists of the responses of five identical first-order low-pass filters with bandwidth 500 Hz to inputs obtained from each of the optical encoders associated with the motor degrees of freedom of the robotic arm. The filter attempts to mitigate the adverse effects of low sensor resolution on the approximation of derivatives. The value of $\dot{q}_a^{(i)}$ in Eq. (3.1) is obtained from the finite difference approximation

$$\dot{q}_a^{(i)} = \frac{q_a^{(i)} - q_a^{(i-1)}}{T}. \tag{3.2}$$

The controller is initialized with $\hat{r}_0 = 0$, $\hat{\sigma}_0 = 0$ and $\hat{\theta}_0 = 0$.

In the experiments described in this chapter, $T = 0.001$ s, $\theta_b = 30$, $\sigma_b = 30$, $\epsilon = 0.3$, $A_{sp} = 850\mathbb{I}$, $Q = 510\mathbb{I}_3$, $\lambda = 1$, and $A_m = -10\mathbb{I}_3$. These values were tuned for the in-lab experiments and maintained constant through all the experiments. This allows easy comparison of the different results. Only the value of k is changed between experiments in order to evaluate the dependence on the filter bandwidth.

3.1.3 Sinusoidal desired trajectory in platform frame

In this experiment, the desired trajectory is sinusoidal in the platform frame, independently of the platform motion, and given by

$$q_d = \begin{pmatrix} 0.5(1-\cos(t)) \\ 0.5(1-\cos(0.5t)) \\ 0.3(1-\cos(0.75t)) \\ -0.3(1-\cos(0.75t)) \\ 0.3(1-\cos(0.5t)) \end{pmatrix}. \tag{3.3}$$

Note that the period of this motion is $8\pi\ s = 25.1327\ s$.

Although the control loop runs with a sampling period of 1 ms, it is not possible to send the state of the system from the control board to the computer at such a rate through the serial connection. For short runs, it may be possible to store all relevant data in the microcontroller itself. As an example, a characteristic time history for $\|r(t)\|_\infty$, sampled every 1 ms, is shown in Fig. 3.3. The erratic nature of the signal results in a rapid rise of the measured $\|r_{[0,t]}\|_{\mathcal{L}_\infty}$ norm. The stepwise increase of the discretization of the error norm $\|r_{[0,t]}\|_{\mathcal{L}_\infty}$ is mostly contained in the first half of each experiment. In order to capture the predicted asymptotic trend, we use the discretization of $\|r_{[\mathrm{per}/2,\mathrm{per}]}\|_{\mathcal{L}_\infty}$ as a representation of the controller performance, where the notation $[\mathrm{per}/2, \mathrm{per}]$ refers to the second half of the period. In addition, to address the limitations of the data collection system and to impose a heuristic filtering of the norm signal, the reported data uses a sampling rate of 50 ms.

The full experiment was conducted over the course of 52 periods. Initially, and after each period, the counter i is set to 0. At the beginning of the experiment $\hat{\theta}^{(0)}$, $\hat{\sigma}^{(0)}$, $\hat{r}^{(0)}$, and $u^{(0)}$ are initialized to 0. Each time a new period begins, they are initialized to their final values in the previous time series. The value of k is held fixed during four consecutive periods of the desired trajectory, allowing for sets of data to be collected for each of 13 different values of k. The results

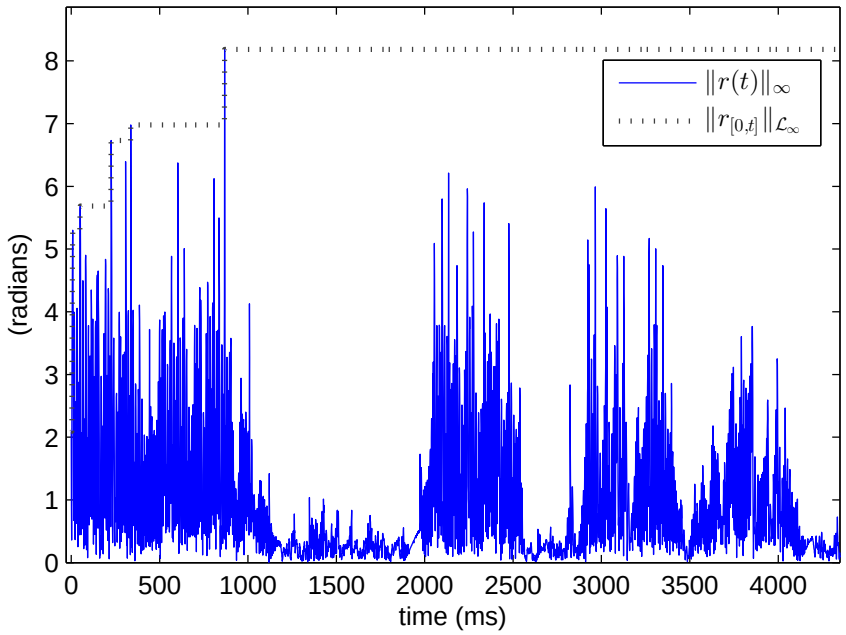
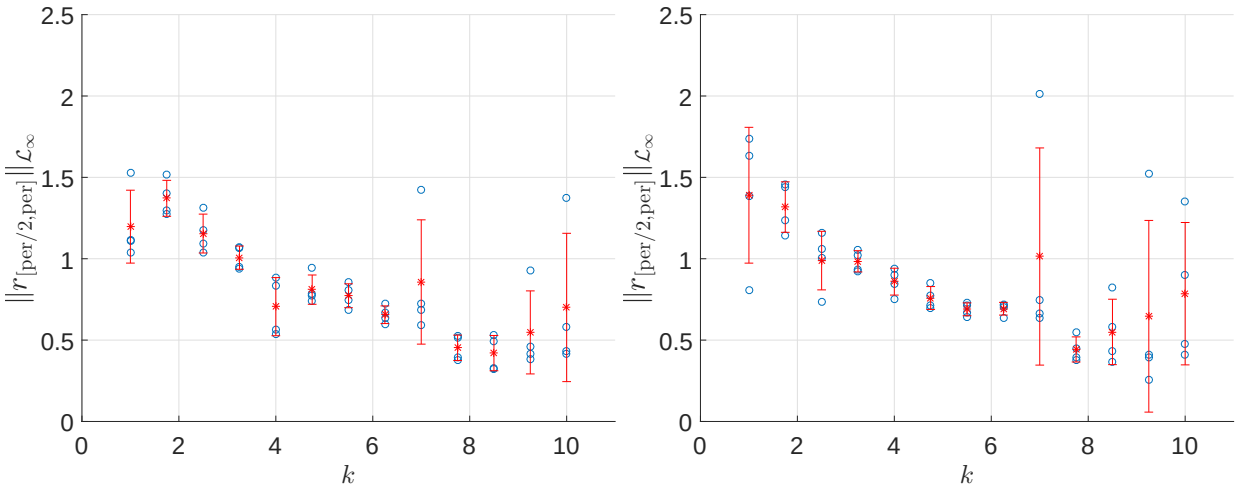


Figure 3.3: Characteristic time history for $\|r(t)\|_\infty$ (solid) and corresponding discretization of $\|r_{[0,t]}\|_{\mathcal{L}_\infty}$ (dotted), using data sampled every 1 ms. The setup corresponds to Section 3.1.4, with $k = 9.25$.

of laboratory and field tests are shown in Fig. 3.4, where different values of the discretization of the error norm $\|r_{[\mathrm{per}/2,\mathrm{per}]}\|_{\mathcal{L}_\infty}$ for a given k correspond to data collected during the second half of each period of the desired trajectory, as explained above.

We observe that the error norm at first decreases when k is increased, analogously to the bound in Eq. (1.13). The similarity between the data collected in the laboratory and those from field experiments show that the controller is able to handle the perturbations from the uneven terrain without a big increase in the tracking error. As indicated in the figure, larger values of k are associated with greater variability in the measured error norms. Interestingly, the analysis in (Nguyen et al., 2015) suggests a similar destabilizing effect beyond some critical value of k due to time delays and unmodeled dynamics.



(a) (b)

Figure 3.4: Variations in the discretization of the error norm $\|r_{[\mathrm{per}/2,\mathrm{per}]}\|_{\mathcal{L}_\infty}$ for different values of the filter bandwidth k for the desired trajectory given in Eq. (3.3). The bars represent the mean and standard deviation across four data points for each value of k. (a) Lab experiment with stationary platform; (b) Field experiment with platform pushed and pulled across uneven terrain.

3.1.4 Sinusoidal desired orientation in the inertial ground frame

In this section, the objective is to make the end effector follow a desired trajectory in an inertial frame. To this end, let

$$y_d(t) = \frac{\pi}{9}\sin\left(\frac{2\pi}{3}t\right), \quad p_d(t) = \frac{\pi}{9}\sin\left(\frac{4\pi}{3}t\right), \quad r_d(t) = \frac{\pi}{6}\sin\left(\frac{2\pi}{3}t\right) \tag{3.4}$$

be the yaw, pitch, and roll angles of the end effector in the ground frame. Let $y_p^{(i)}$, $p_p^{(i)}$ and $r_p^{(i)}$ denote the yaw, pitch, and roll angles of the platform with respect to the stationary frame (as updated by the IMU in the cart every 20 iterates, cf. Fig. 3.1). Assuming that $q_{d,2}^{(i)} = q_{d,3}^{(i)} = 0$ for all i, the discretized desired trajectory $q_d^{(i)}$ can be analytically obtained from the following equality

of the rotation matrices:

$$\underbrace{R_1\left(q_{d,5}^{(i)} - 2q_{d,4}^{(i)}\right) R_2\left(q_{d,4}^{(i)}\right) R_3\left(q_{d,1}^{(i)}\right)}_{\text{Rotation to end effector from cart}} =$$

$$= \underbrace{R_1\left(r_d^{(i)}\right) R_2\left(p_d^{(i)}\right) R_3\left(y_d^{(i)}\right)}_{\text{Rotation to end effector from ground}} \underbrace{\left(R_1(r_p^{(i)}) R_2(p_p^{(i)}) R_3(y_p^{(i)})\right)^\top}_{\text{Rotation to ground from cart}}, \quad (3.5)$$

where $R_i(\theta)$ denotes the matrix corresponding to a rotation from frame A to frame B by an angle θ about the common i-th basis vector, such that $\vec{v}_B = R_i(\theta)\vec{v}_A$ for all $\vec{v}$. An analytical expression for the desired velocities $\dot{q}_d^{(i)}$ is also derived from the previous expression.

As was discussed in the previous section, occasional large peaks in $\|r(t)\|_\infty$ may result from measurement and discretization errors, for example those produced in the velocities (3.2) by the low resolution of the measurement of $q_{a,4}^{(i)}$ and $q_{a,5}^{(i)}$, or those introduced by new measurements of the IMU every 20 ms. This is evident in the time history in Fig. 3.5 with data recorded every 1 ms. Although the omission of such data, as a result of collecting data only every 50 ms, might skew the results, the sharpness of these peaks shows that their source is the sensor design rather than the control architecture, whose performance we seek to quantify.

During full data collection, a series of 52 independent 24-seconds-long experiments were run, four for each of 13 values of the bandwidth k. The controller is initialized at the beginning of each experiment with $i = 0$, $\hat{\theta}^{(0)} = 0$, $\hat{\sigma}^{(0)} = 0$, $\hat{r}^{(0)} = 0$ and $u^{(0)} = 0$. The results of laboratory and field tests are shown in Fig. 3.6.

We again observe a critical value of the bandwidth, $k_{crit} \approx 5$, below which $\|r_{[\mathrm{per}/2,\mathrm{per}]}\|_{\mathcal{L}_\infty}$ decreases as k is increased, consistent with the bound given in Eq (1.13). For values of k below k_{crit}, the trend of the in-lab experiments matches that in the field. For values of k above k_{crit}, however, the controller is frequently found to behave poorly, having relatively larger tracking errors.

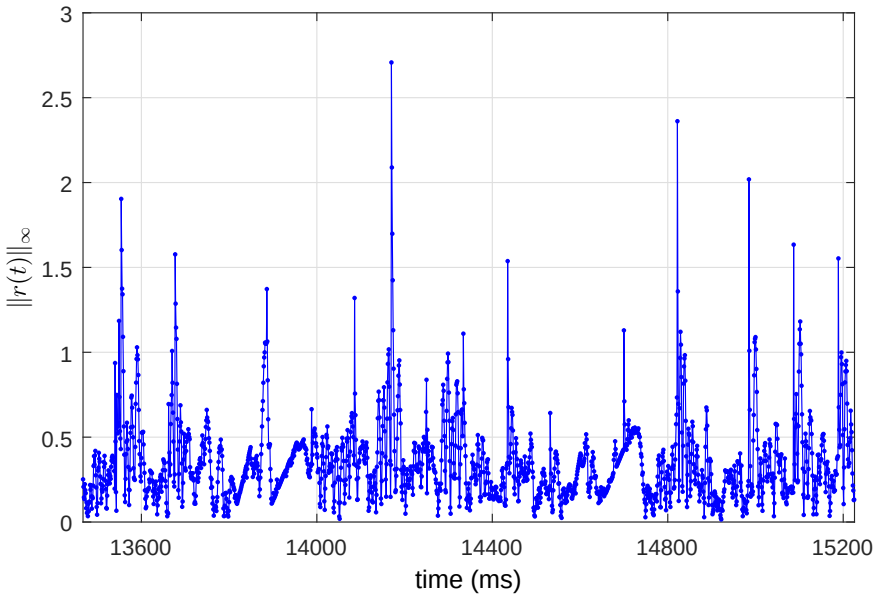


Figure 3.5: With IMU readings sampled approximately every 20 ms, sharp peaks are induced in $\|r(t)\|_\infty$, here sampled every 1 ms. The setup corresponds to Section 3.1.4, with $k = 5.5$.

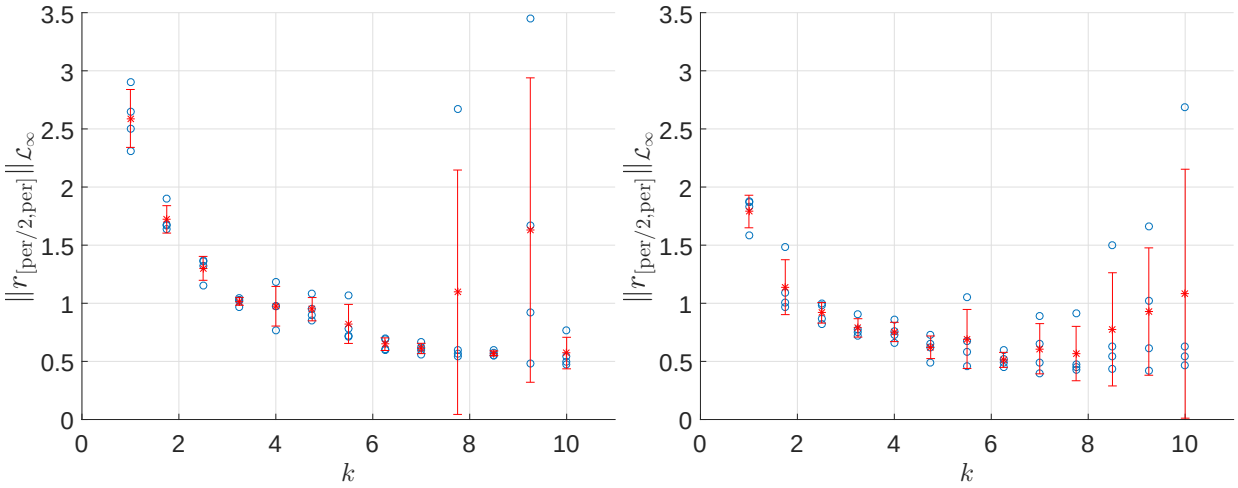


Figure 3.6: Variations in the discretization of the error norm $\|r_{[\mathrm{per}/2,\mathrm{per}]}\|_{\mathcal{L}_\infty}$ for different values of the filter bandwidth k for the desired trajectory given in Eq. (3.4). The bars represent the mean and standard deviation across four data points for each value of k. (a) Lab experiment with stationary platform; (b) Field experiment with platform pushed and pulled across uneven terrain.

3.1.5 Maintaining a constant orientation in the inertial ground frame

In the third and final experiment, the platform is translated across sloped, uneven terrain along an 'S'-shaped path while the controller attempts to maintain constant end effector orientation relative

to the inertial ground frame, with $y_d(t) = p_d(t) = r_d(t) = 0$, for some fixed k. Here, the controller is initialized with $\hat{\theta}^{(0)} = 0$, $\hat{\sigma}^{(0)} = 0$, $\hat{r}^{(0)} = 0$ and $u^{(0)} = 0$, and data collection is initiated after 1 minute. Fig. 3.7 shows a typical comparison between the yaw, pitch, and roll angles of the platform and the end effector, respectively, relative to the inertial ground frame. As the peak-to-peak variations in the platform yaw angle are considerably larger than in the other degrees of freedom, the controller's ability to compensate for the platform dynamics is more pronounced for the end effector yaw angle. Nevertheless, the control scheme appears able to maintain minimal rotary movements of the end effector at all times.

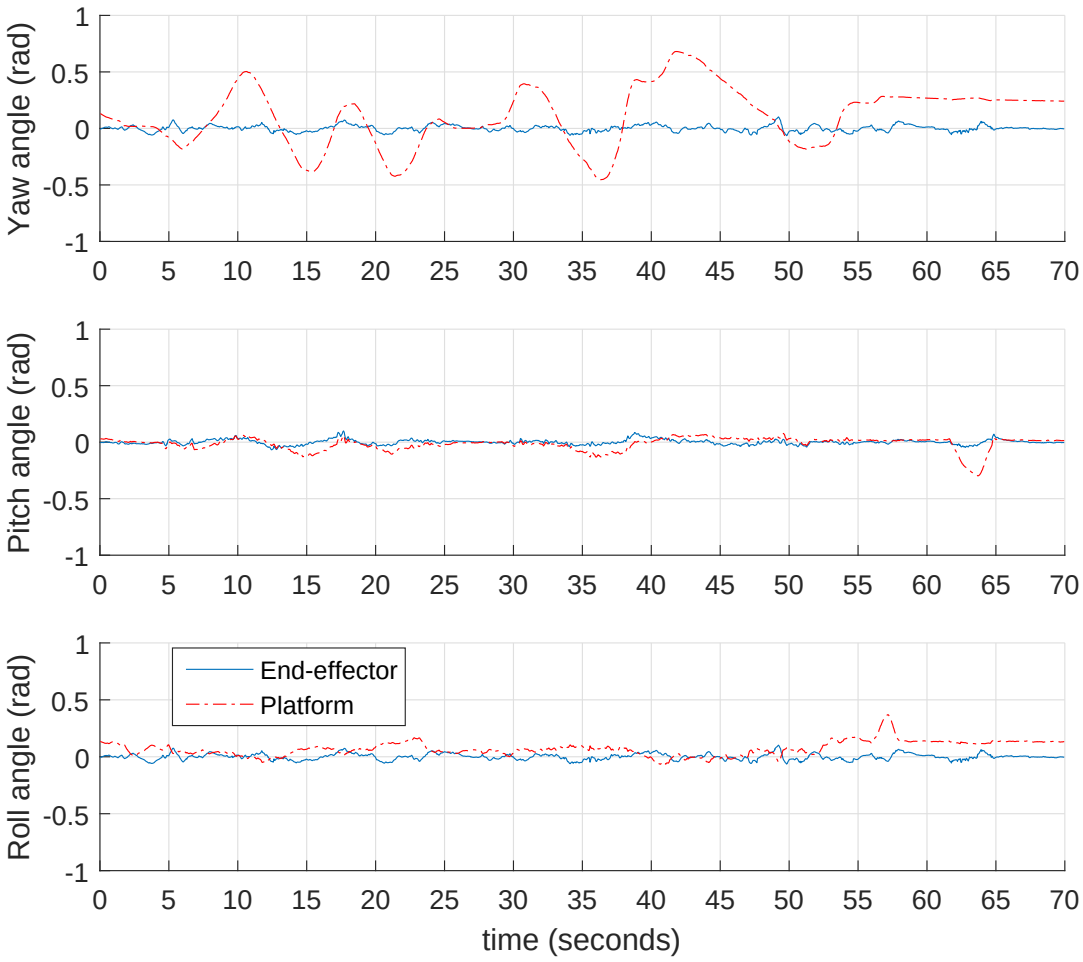


Figure 3.7: Estimated yaw, pitch, and roll angles for the platform and the end effector. The platform orientation is obtained directly from the IMU, while the end effector orientation is obtained by composition of the output of the platform IMU and the joint angles measured by the optical encoders.

3.2 Comparison with Cao and Hovakimyan discretization

A different implementation of the controller, following the Cao and Hovakimyan discretization in Section 1.2.5, is shown in this section. This controller is simpler to analyze, has fewer parameters to tune, and is better suited for discrete implementations with fixed controller sampling interval.

3.2.1 Experimental set-up and adaptive controller definition

For the sole purpose of comparison, the experiments described here were only carried out at the lab. The experiment is analogous to that in Section 3.1.3, where the robot is given a sinusoidal desired trajectory in the platform frame and the platform remains stationary.

The controller implements the discretization in Eqs (1.19-1.21) with $\lambda = 50$, $A_m = 5\mathbb{I}_3$ and $k = 1$. The controller parameters were manually adjusted to minimize the tracking error and oscillation in the control signal u. Since there are relatively fewer parameters to adjust (λ, A_m and k), the search was very quick and a low value of the tracking error was achieved within one hour of tuning (versus the several weeks it took for the controller in Section 3.1).

3.2.2 Methodology for data acquisition

An automated configuration utility was implemented in the robot controller, so that several values of the parameters could be sequentially tested. This allowed for an extensive and quick two-dimensional sweep, varying both the controller sampling interval T and the filter bandwidth k.

Before each run, the controller was reset to $i = 0$, $u^{(0)} = 0$, $\hat{r}^{(0)} = 0$, $\hat{\sigma}^{(0)} = 0$, and the robot was then driven to $q_a = 0$, leaving the robot there long enough for it to stop completely. Controller instability was automatically detected using an empirical condition $\left\|\sigma^{(i)}\right\|_\infty > \sigma_{max}$, and the robot status was reset to the initial position for the next experiment.

3.2.3 Results for the experiment using all joints

In the first set of experiments, the desired trajectory was given by

$$q_d = \begin{pmatrix} 0.5(1-\cos(0.6\pi t)) \\ 0.5(1-\cos(0.3\pi t)) \\ 0.3(1-\cos(0.45\pi t)) \\ -0.3(1-\cos(0.45\pi t)) \\ 0.3(1-\cos(0.3\pi t)) \end{pmatrix}. \tag{3.6}$$

Each experiment was executed for $11.669\ s$ with $\sigma_{max} = 1200$.

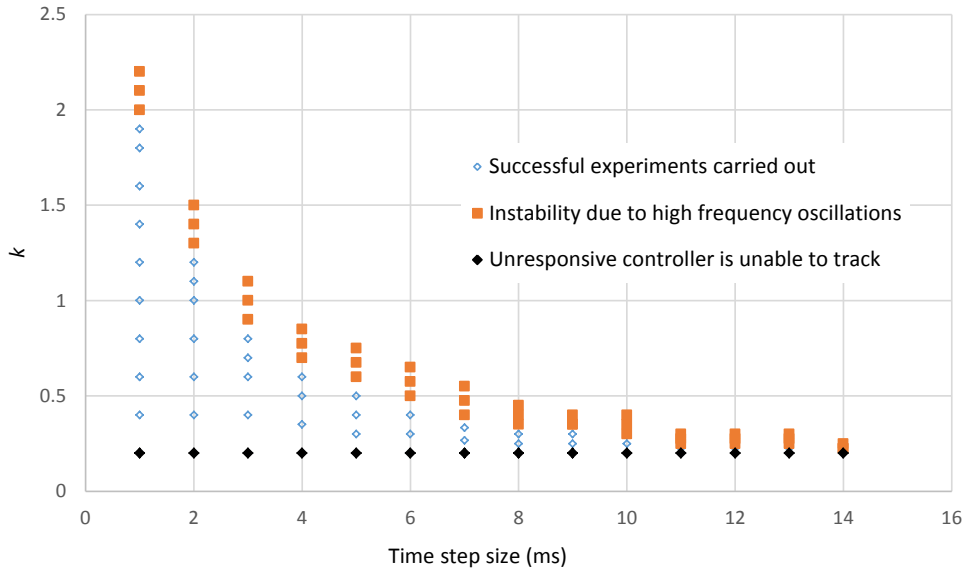


Figure 3.8: Values of k and T used in the experiments in Section 3.2.3 with a qualitative evaluation of each run. High frequency oscillations are observed through huge noise or vibration. Unresponsive controller means stationary robot for some time and sudden overshoots.

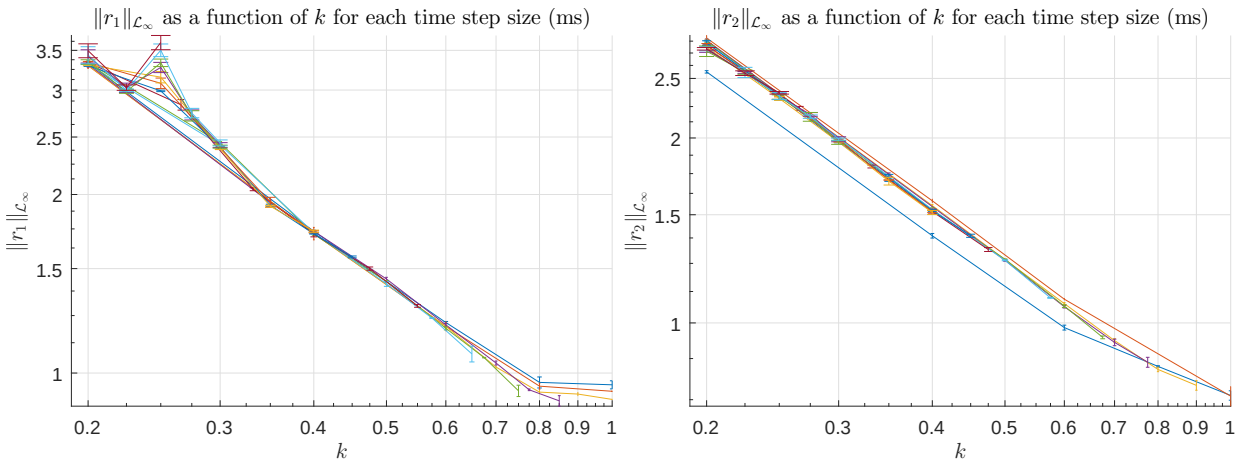
The values of k and T used in the experiments are depicted in Fig. 3.8, together with a qualitative observation on whether the experiment ran successfully. For high values of k, high frequency oscillations are observed as the robot makes a loud noise or vibrates visibly. For very low values of k, the controller is unresponsive, with the arm staying still for some period of time and suddenly overshooting. The experiment was run 3 times for each tuple (k, T).

Figure 3.9 shows the discretized $\mathcal{L}_\infty$ norm for each of the five components of r. Prior to the onset of instability, the first four joints show a $1/k$ dependence of the tracking error on the bandwidth. Instability is in all cases triggered by the fifth joint, which exhibits a less-clearly-defined initial decrease with k. The critical value of the bandwidth, k_{crit}, decreases with the sampling interval T. For $T \lesssim 14$ ms, there is no value of k for which $\|r\|_{\mathcal{L}_\infty}$ is bounded.

3.2.4 Results for the experiment using only the first joint

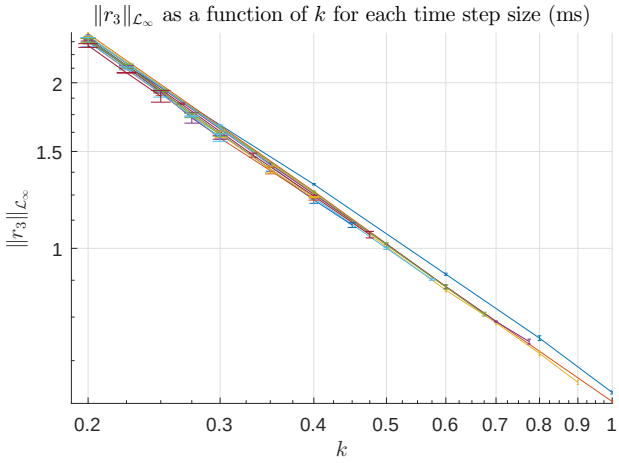
To further characterize the dependence on k and T, an experiment using only the first joint was carried out. This joint has the best motor and encoder and thus suffers less from perturbations and poor measurement resolution. The desired trajectory is the sinusoid $q_{d,1} = 0.5(1-\cos(1.8\pi t))$, and the duration of each experiment was two periods. The values attempted are depicted in Fig. 3.10 and $\sigma_{max} = 120$.

The results in Fig. 3.11 again confirm a $1/k$ dependence of the norm $\|r_1\|_{\mathcal{L}_\infty}$ as a function of the bandwidth k, prior to the onset of instability. The critical value of the bandwidth, k_{crit}, quickly decreases as T increases, leaving no value of k for which stable behavior is observed once $T \gtrsim 40$ ms. Notably, the critical value of T is significantly higher than found for the fifth joint in the previous section. Figure 3.12 shows the dependence of $\|r_1\|_{\mathcal{L}_\infty}$ on T for $k = 0.35$.

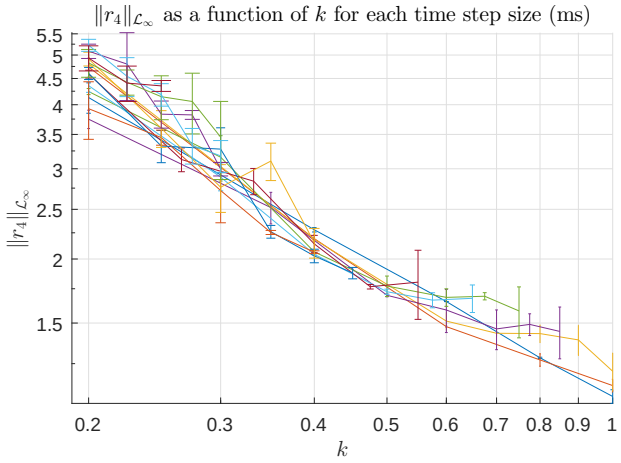


(a) Joint 1.

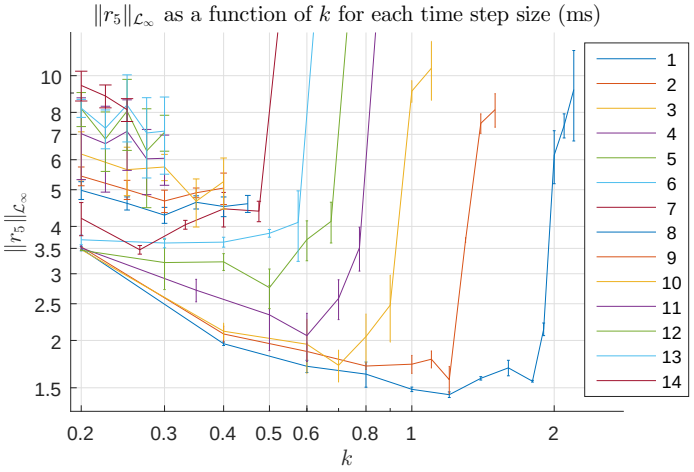
(b) Joint 2.



(c) Joint 3.



(d) Joint 4.



(e) Joint 5.

Figure 3.9: Results of $\|r_{i,[0,t_{exp}]}\|_{\mathcal{L}_\infty}$ for each joint as a function of the filter bandwidth k and the controller sampling interval T in milliseconds (colors). Error bars show the standard deviation, while the points are at the mean.

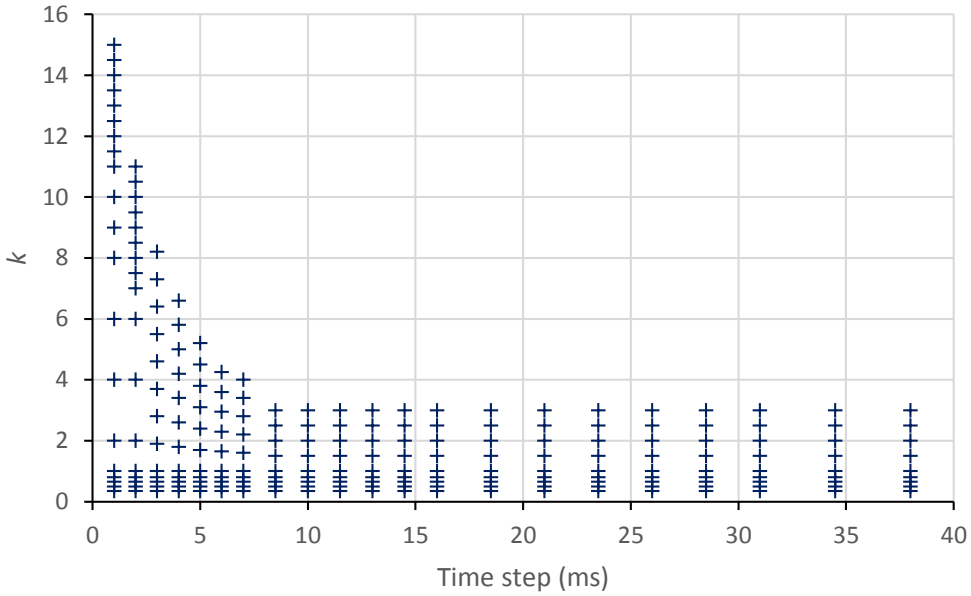


Figure 3.10: Values of k and T used in the experiments in Section 3.2.4.

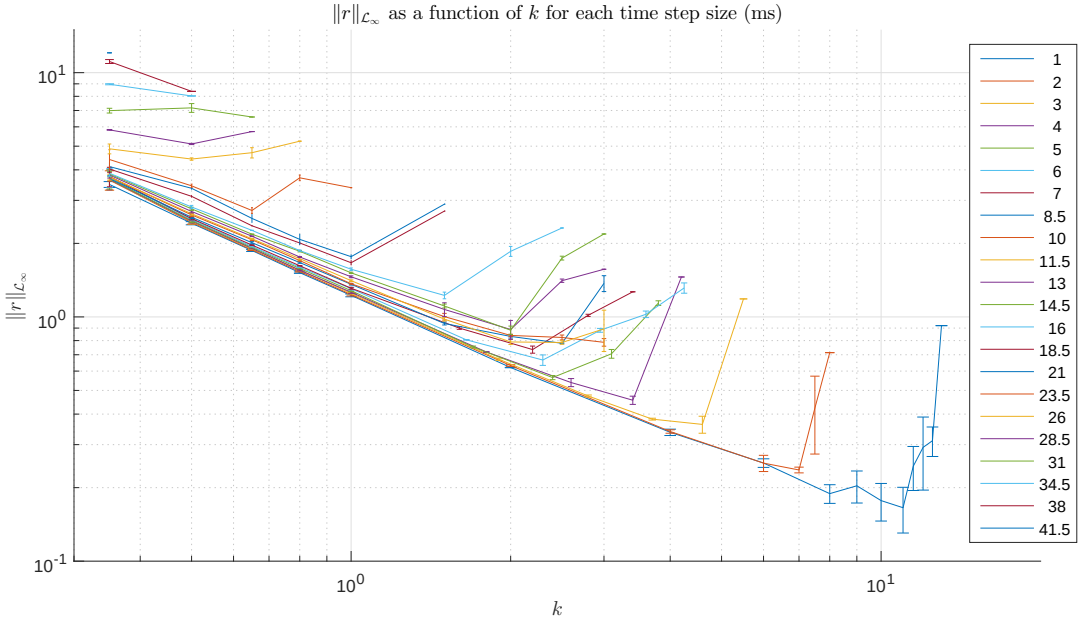


Figure 3.11: Results of $\|r_{1,[0,t_{exp}]}\|_{\mathcal{L}_\infty}$, when using only the first joint, as a function of the filter bandwidth k and the controller sampling interval T in milliseconds (colors). Error bars show the standard deviation, while the points are at the mean.

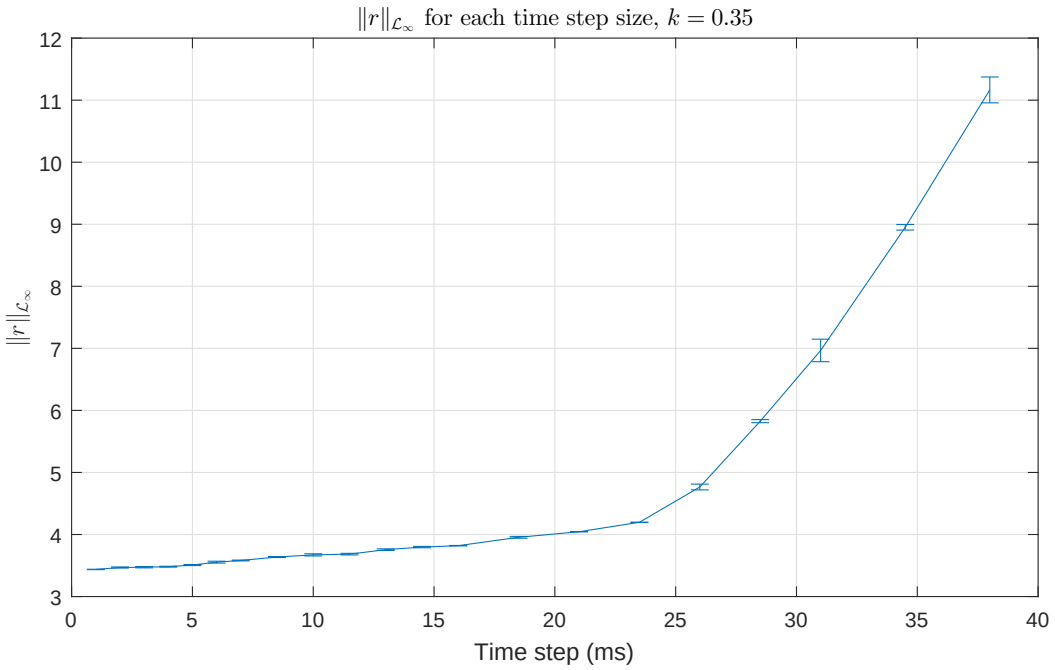


Figure 3.12: Results of $\|r_{1,[0,t_{exp}]}\|_{\mathcal{L}_\infty}$, for $k = 0.35$, when using only the first joint, as a function of the controller sampling interval T in milliseconds (colors). Error bars show the standard deviation, while the points are at the mean.

3.3 Discussion

This chapter has proposed and experimentally validated a stabilization scheme for the end effector of a manipulator, which is mounted on a mobile platform with unknown dynamics. Effective stabilization supports the use of this mechanism, for example, in drive-by inspection and treatment for agricultural applications.

The dependence of the performance bound on the bandwidth of a low-pass filter and the controller sampling interval were studied both in the lab and in the field. The limitation in the controller performance for very high values of the low-pass filter bandwidth was shown in the experimental results. Increasing this parameter above a critical value leads to a quick degradation in the performance, no longer following the expected bounded behavior.

Two discretizations of the $\mathcal{L}_1$ controller were compared. The experiments performed in the lab show that the Cao and Hovakimyan discretization in Section 3.2 has less variability (lower standard deviation) than the Euler-like discretization in Section 3.1. This together with the fact that it has many fewer parameters makes the Cao and Hovakimyan discretization much easier to implement and tune. This discretization allows for a sampling interval up to 7 ms, before becoming unstable, having a very smooth motion for $T = 1$ ms. Other discretizations, such as the third order scheme in (Bogacki and Shampine, 1989), were tested with no significant improvements.

The most important lesson learned in these experiments is that it is highly advisable to use the Cao and Hovakimyan discretization in all experimental setups, as hardware behaves better using this formulation. The added complexity of the Euler-like formulation generates significant implementation problems. The fact that the $\|r_{[0,t]}\|_{\mathcal{L}_\infty}$ norm is used in the controller means that any unexpected perturbation will degrade the controller performance for the remainder of the execution time. Additionally, an initial condition mismatch, which has no effects in the far future in the Cao and Hovakimyan formulation, will damage the whole future performance in the Euler-like formulation.

For future experiments, measuring the joint velocity directly would remove much of the diffi-

culty related to the low resolution of the joint position measurement, which introduced significant noise in the finite difference approximation of the velocity.

Chapter 4

Robot Networks in the Presence of Communication Delays

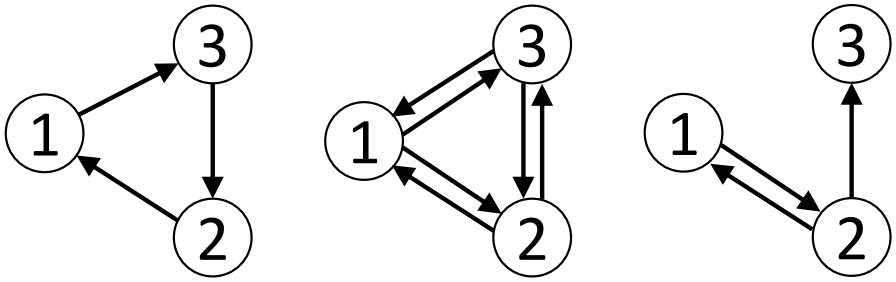
The next sections show experimental results for a network of manipulators, as described in (Nguyen and Dankowicz, 2016), using the formulations developed in Sections 1.2.7 and 1.2.8. The network was implemented as described in Section 1.3.3, with certain limitations on the minimum delay achievable, around 16 ms. The dependence of the network performance on the delay was explored, by adding a known ammount of extra delay (on top of the minimum unknown delay around 16 ms) to all links. The robots are mounted in three different platforms. Robot ① is mounted in a cart that is not allowed to roll. Robot ② is mounted in a platform suspended by four springs, allowing some oscillatory movement induced by the arm motion. Robot ③ is mounted in a cart with two wheels unlocked, allowing small movements.

Two cooperation strategies were implemented with different network topologies. Synchronization before tracking, described in Section 4.1, tries to make the robots quickly synchronize their respective joint angles while they all slowly approach a given desired trajectory. Consensus, in Section 4.2, tries to make the robots approach a common configuration, with no predefined desired trajectory. In both cases, the results of hardware experiments were compared to the numerical results in (Nguyen and Dankowicz, 2016).

A computer Graphical User Interface (GUI) was developed to assist with running these experiments in an easy and safe manner (see Section 1.3.4). The interface allows the user to select desired trajectories and values for the parameters without re-flashing the code to the robot controller boards on each run, since this would take a prohibitively long time, given that there are three robots and only one programmer available. The GUI also provides an easy shutdown of all robots with one click, and an automatic safety shutoff in case of a lost or broken connection.

4.1 Synchronization

The controller defined in Section 1.2.7 was used in several experiments, seeking to reproduce the numerical results in (Nguyen and Dankowicz, 2016). Only the first three degrees of freedom of each manipulator were controlled. This avoids the extra data communication, as well as problems with low sensor resolution in the last two degrees of freedom. The different network topologies used are depicted in Fig. 4.1. The network weights used for synchronization are all equal to 1.



(a) Circular. (b) Fully connected. (c) ① ⇄ ② → ③.

Figure 4.1: Different network topologies used in synchronization before tracking and consensus without leader.

4.1.1 Achievability of synchronization and tracking

For homogeneous networks with identical robots, (Nguyen and Dankowicz, 2016) predicts that synchronization before tracking is possible if the matrices $A_{m,i}$ are the same for all robots and if there are no communication delays. To verify this prediction, let the initial conditions for each of three robots be given by

$$\begin{aligned} q_a^{(1)}(0) &= \{0.5, 0.4, 0.3\} && \text{for robot ①,} \\ q_a^{(2)}(0) &= \{-0.1, 0.5, 0.0\} && \text{for robot ②,} \\ q_a^{(3)}(0) &= \{0.4, -0.1, -0.1\} && \text{for robot ③,} \end{aligned} \tag{4.1}$$

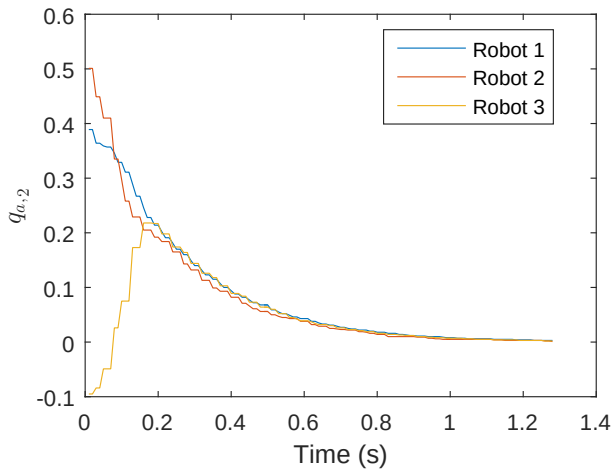
and consider the circular network topology in Fig. 4.1 (a). Let the desired trajectory be given by $q_d = 0$, and let the controller parameters be given by $T = 1\ ms$, $\lambda = 40$, $k = 5$ and $A_{m,i} = a_i \mathbb{I}_3$ for different combinations of a_i. Here no additional delay is added to the network.

For each experiment, data was collected once, and the experiment was manually stopped once the joint angles were qualitatively close to the desired trajectory. Figure 4.2 shows a comparison of the time series for the first joint of the networked manipulators. When $a_1 = a_2 = a_3$, synchronization is achieved before tracking. However, in the two cases where $a_1 \neq a_2 \neq a_3$, synchronization does not happen.

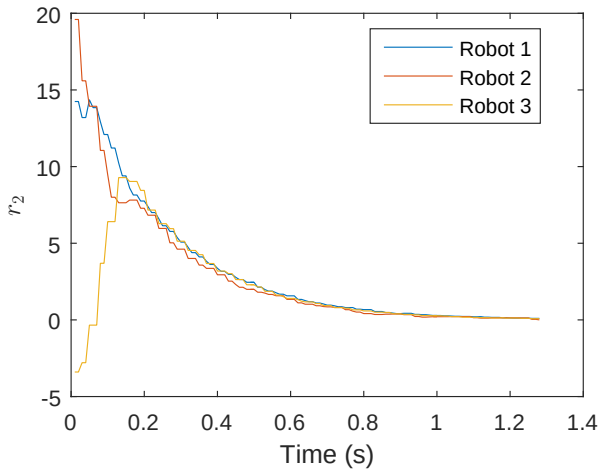
To evaluate the tracking capabilities of the networked controller for different values of k, the experiment was run with the desired trajectory given by

$$q_d = \begin{pmatrix} 0.5(1 - \cos(3\pi t/5)) \\ 0.5(1 - \cos(3\pi t/10)) \\ 0.3(1 - \cos(9\pi t/20)) \end{pmatrix}, \tag{4.2}$$

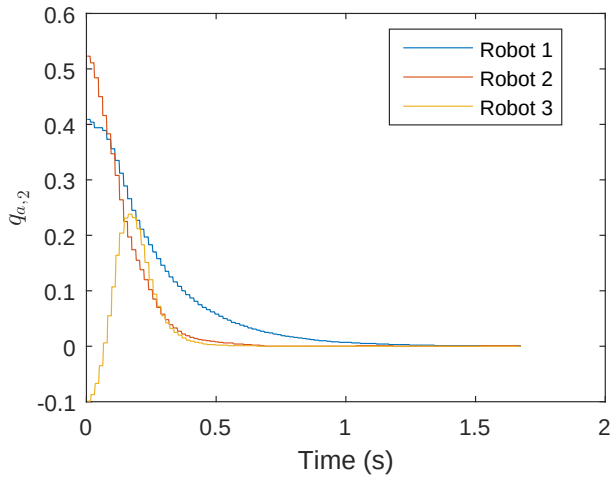
with no additional communication delay, and no tracking error in the starting condition, $q_a(0) = q_d(0)$. The network topology is again circular and the controller parameters are $T = 1$ ms, $\lambda = 40$, and $A_{m,i} = 4\mathbb{I}_3$. The results in Fig. 4.3 show a similar trend to that in Fig. 3.11 for a single robot. For low values of k the tracking error decays as $1/k$, matching the numerical results in Fig. 8 in (Nguyen and Dankowicz, 2016) and the expected behavior from Eq. (1.29). However, for large values of k the error induced by the discretization is very large. The flattening on the rightmost side of Fig. 8 in (Nguyen and Dankowicz, 2016) is not visible in this experiment, although it might appear if a lower controller sampling interval T was used, as this might allow higher values of k to be used before diverging. The current hardware available does not support a smaller sampling interval.



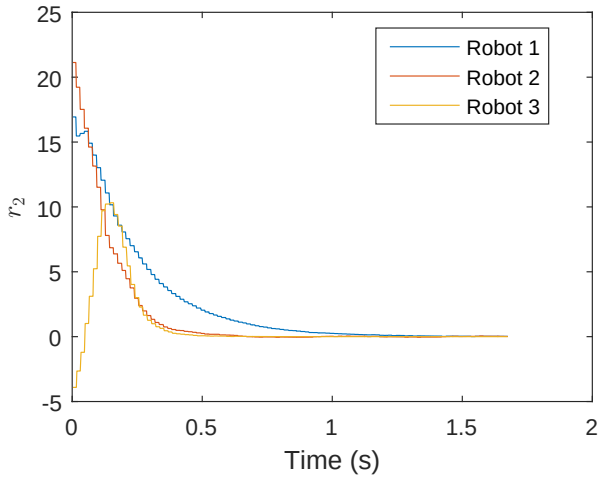
(a) Joint 2 for the case $a_1 = a_2 = a_3 = 4$.



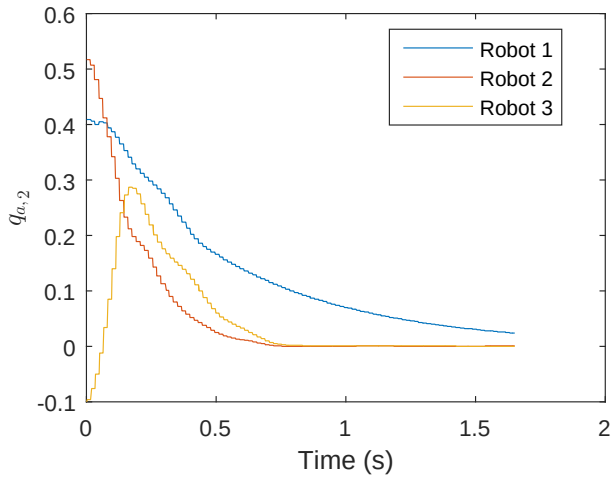
(b) Joint 2 for the case $a_1 = a_2 = a_3 = 4$.



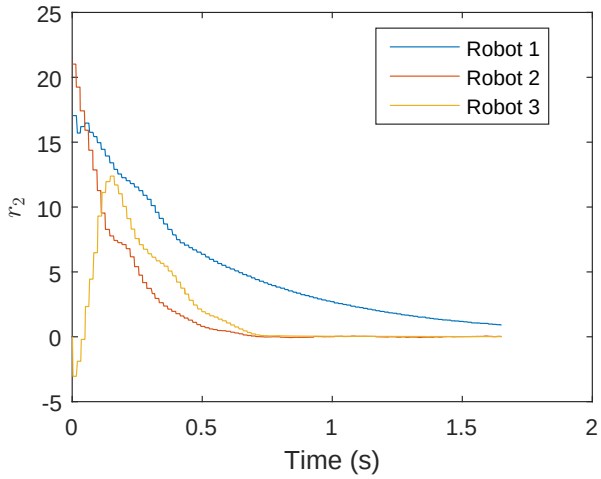
(c) Joint 2 for the case $a_1 = 4$, $a_2 = 10$, $a_3 = 20$.



(d) Joint 2 for the case $a_1 = 4$, $a_2 = 10$, $a_3 = 20$.



(e) Joint 2 for the case $a_1 = 1$, $a_2 = 8$, $a_3 = 50$.



(f) Joint 2 for the case $a_1 = 1$, $a_2 = 8$, $a_3 = 50$.

Figure 4.2: Time series for a circular network topology with $q_d = 0$, in the synchronization before tracking case.

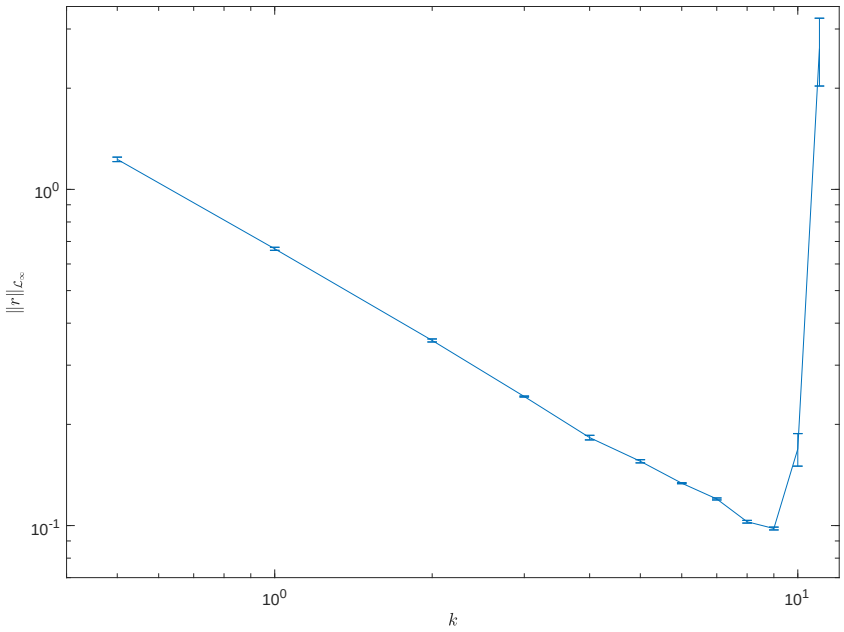


Figure 4.3: Dependence of the tracking error on the filter bandwidth for a circular network with sinusoidal desired trajectory.

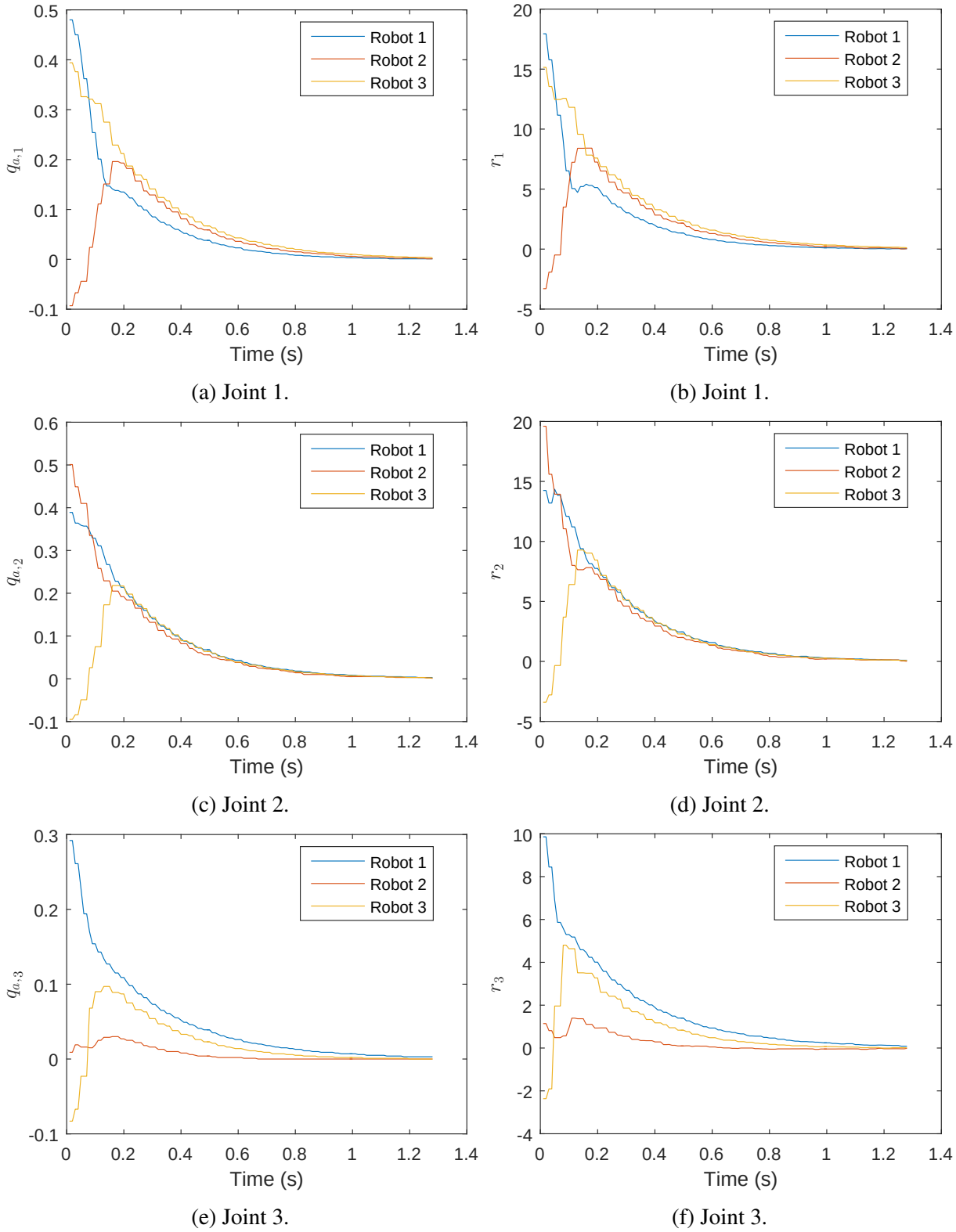
4.1.2 Effect of different network topologies and network delays

The experiments described in this section again consider a constant $q_d = 0$ with $T = 1\,ms$, $\lambda = 40$, $k = 5$, identical $A_{m,i} = 4\mathbb{I}_3$, and using the fixed initial condition in (4.1).

The results for a circular network topology are shown in Fig. 4.4; for the fully connected network in Fig. 4.5 and for the ① ⇄ ② → ③ network in Fig. 4.6. The reader should note that synchronization happens before tracking in all the cases, with little influence from the network topology. As all the incoming connections add up their influences in u_s, the synchronization controller u_s has more influence, relative to the tracking controller u_a, for robots with more incoming connections. This can be observed in the better synchronization performance of the fully connected topology (Fig. 4.5). Differences in the behaviors of each joint may be a result of different mechanical properties (motors and inertias) and the fact that joints 2 and 3 are influenced by gravity.

Different network delays have an effect in synchronization. Nguyen and Dankowicz (2016) predicts that synchronization should take place only with a network delay close to 0. It is clear that very large delays will not allow synchronization as each robot will not get information from the other robots before converging to the desired trajectory. To illustrate the effects of different network delays, Fig. 4.7 shows the time series for the first joint for different values of the added network delay $\delta T_s \in \{0, 100, 500\}$ ms. These confirm the theoretical predictions. When the delay is large, it generates notable perturbations in the tracking, but the system remains stable.

Similar results are obtained for the case of a sinusoidal desired trajectory as shown in Fig. 4.8. Synchronization is clearly achieved before tracking, as expected. As seen in Fig. 4.9 the controller remains stable for large network delays, although tracking is severely compromised during the first seconds.



(a) Joint 1.

(b) Joint 1.

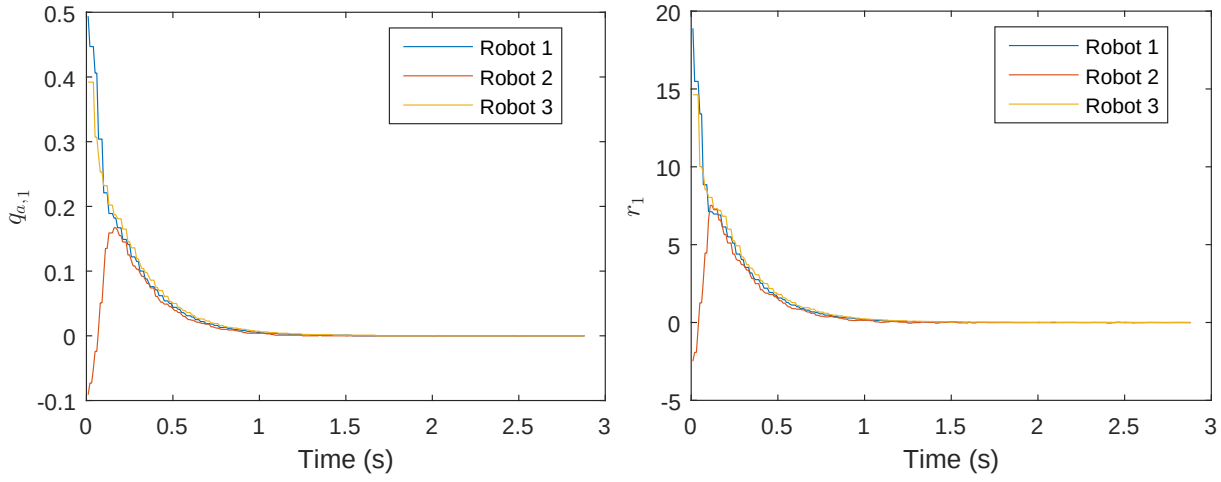
(c) Joint 2.

(d) Joint 2.

(e) Joint 3.

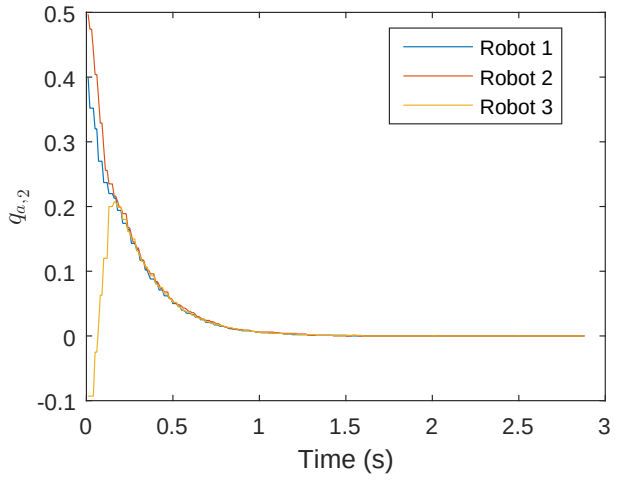
(f) Joint 3.

Figure 4.4: Time series for the circular network topology with $q_d = 0$, in the synchronization before tracking case, with identical parameters as were used to generate Figs. 4.2 (a) and (b).

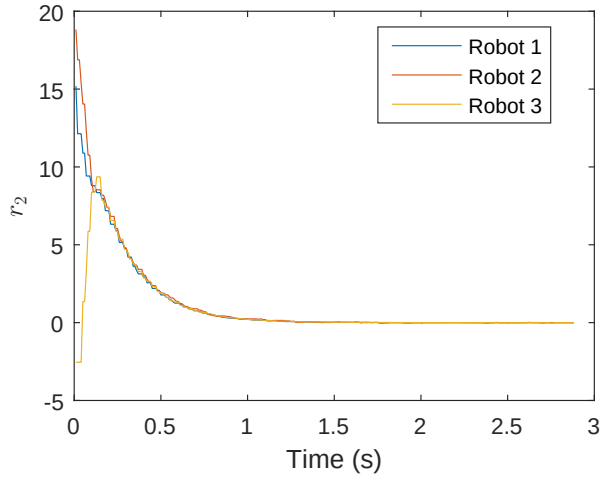


(a) Joint 1.

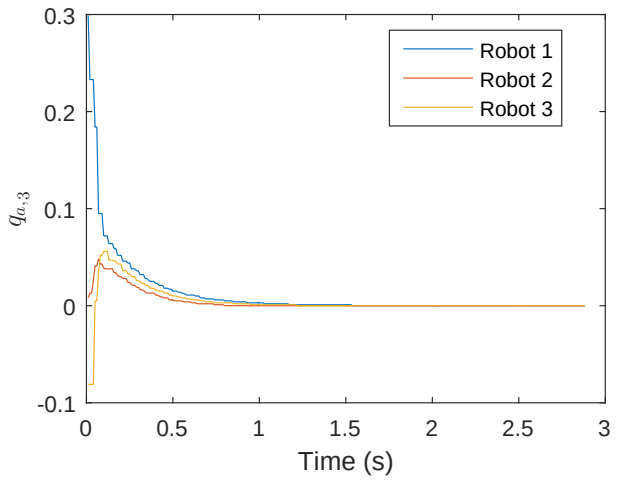
(b) Joint 1.



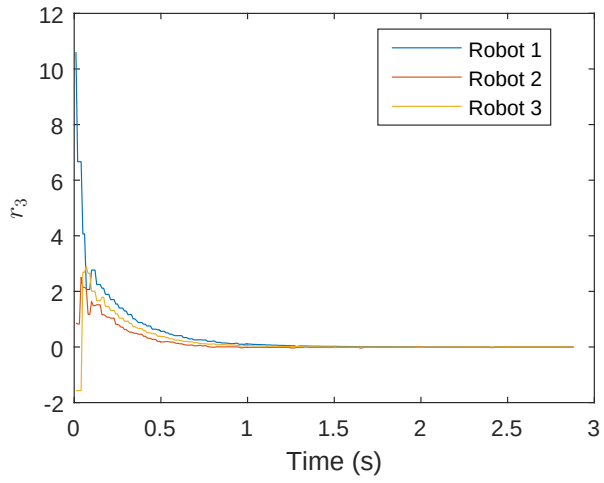
(c) Joint 2.



(d) Joint 2.

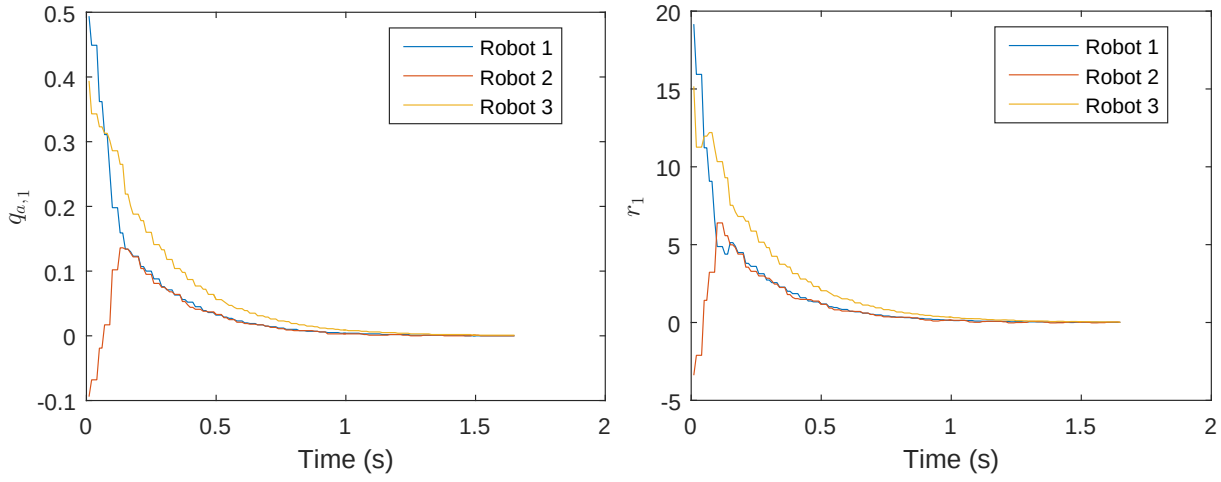


(e) Joint 3.



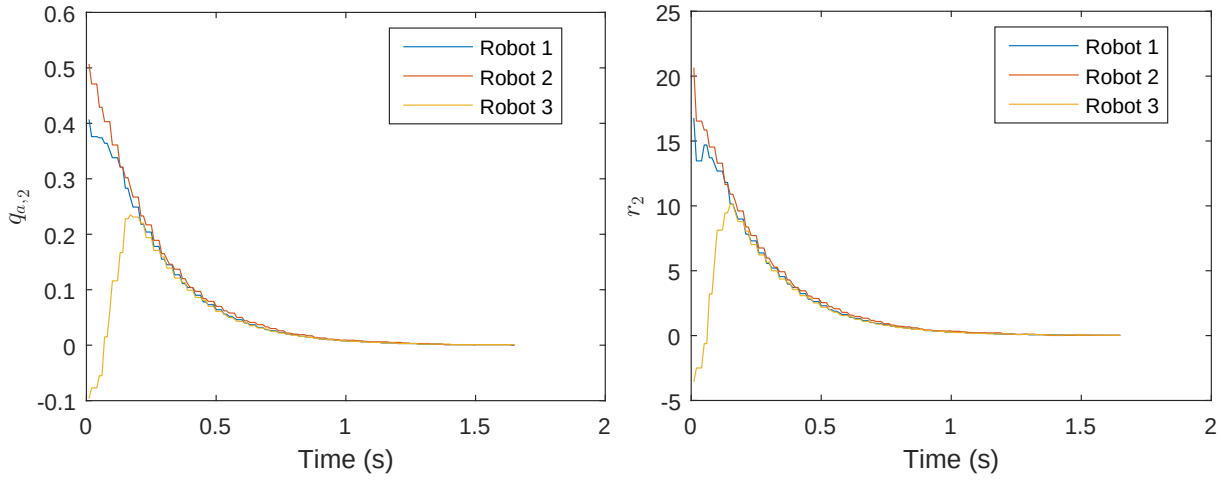
(f) Joint 3.

Figure 4.5: Time series for the fully connected network topology with $q_d = 0$, in the synchronization before tracking case, with identical parameters as were used to generate Figs. 4.2 (a) and (b).



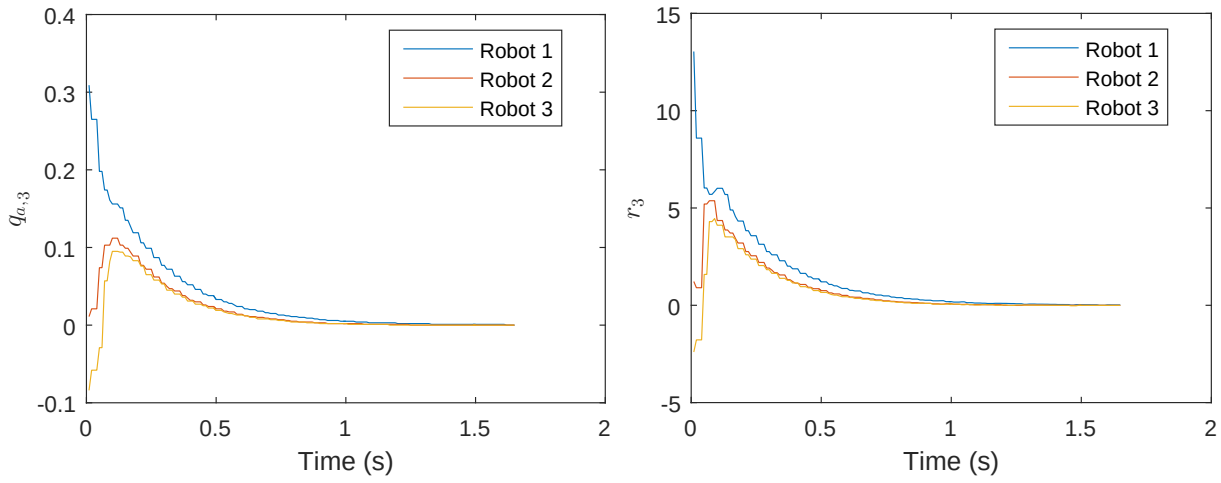
(a) Joint 1.

(b) Joint 1.



(c) Joint 2.

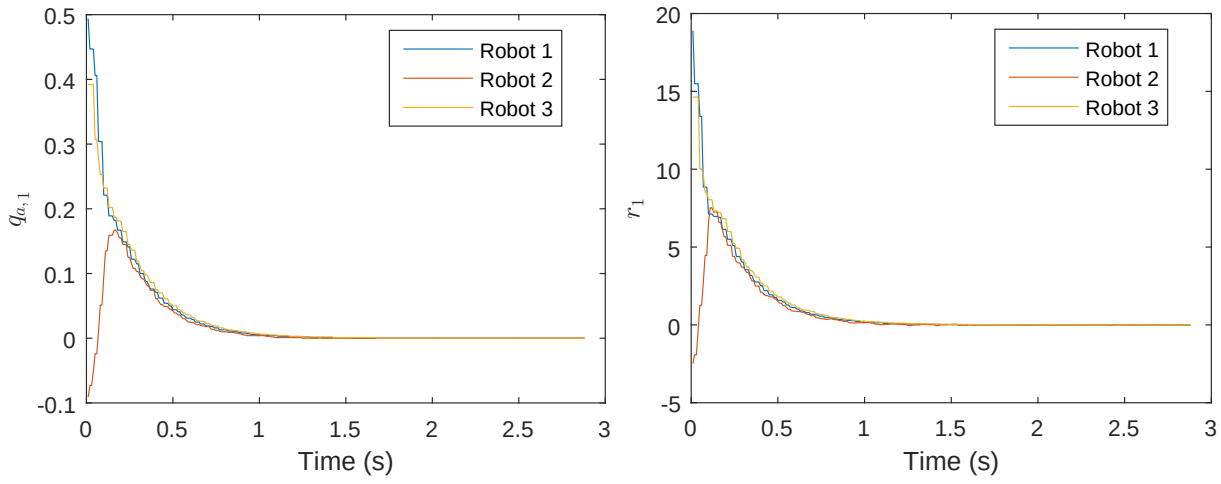
(d) Joint 2.



(e) Joint 3.

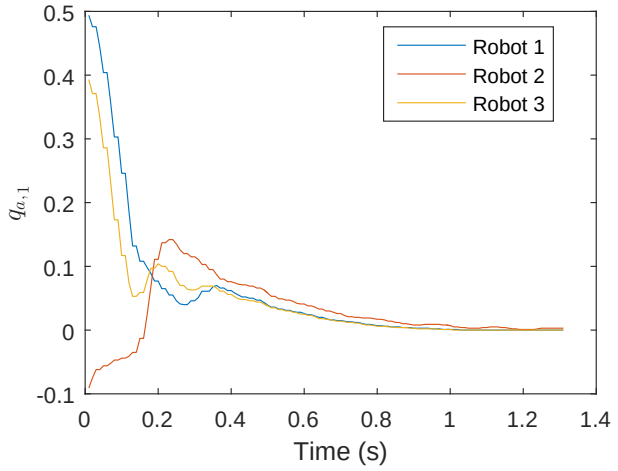
(f) Joint 3.

Figure 4.6: Time series for the ① ⇄ ② → ③ network topology with $q_d = 0$, in the synchronization before tracking case, with identical parameters as were used to generate Figs. 4.2 (a) and (b).

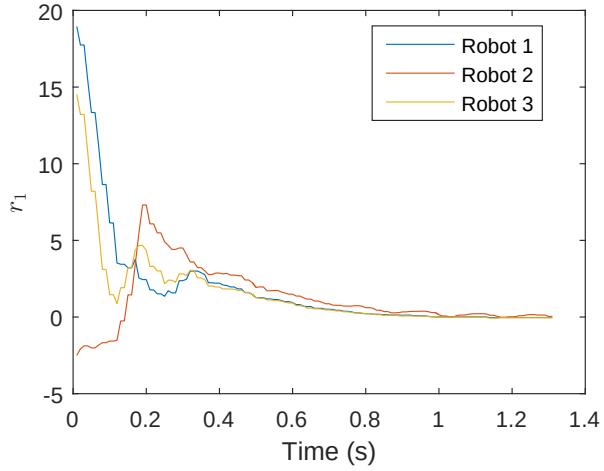


(a) Network delay smaller than 16 ms.

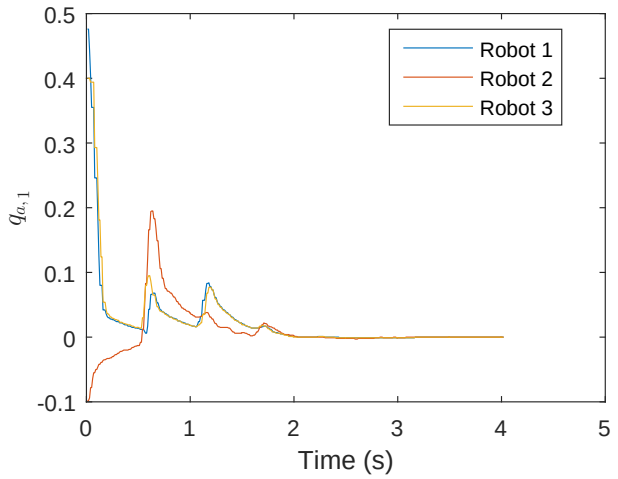
(b) Network delay smaller than 16 ms.



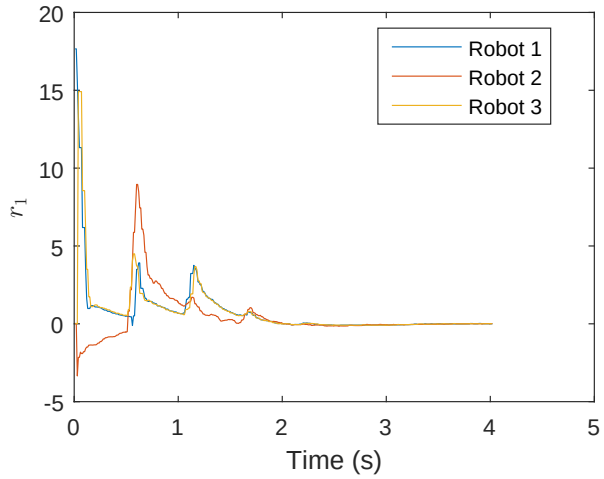
(c) Network delay of 100 ms.



(d) Network delay of 100 ms.

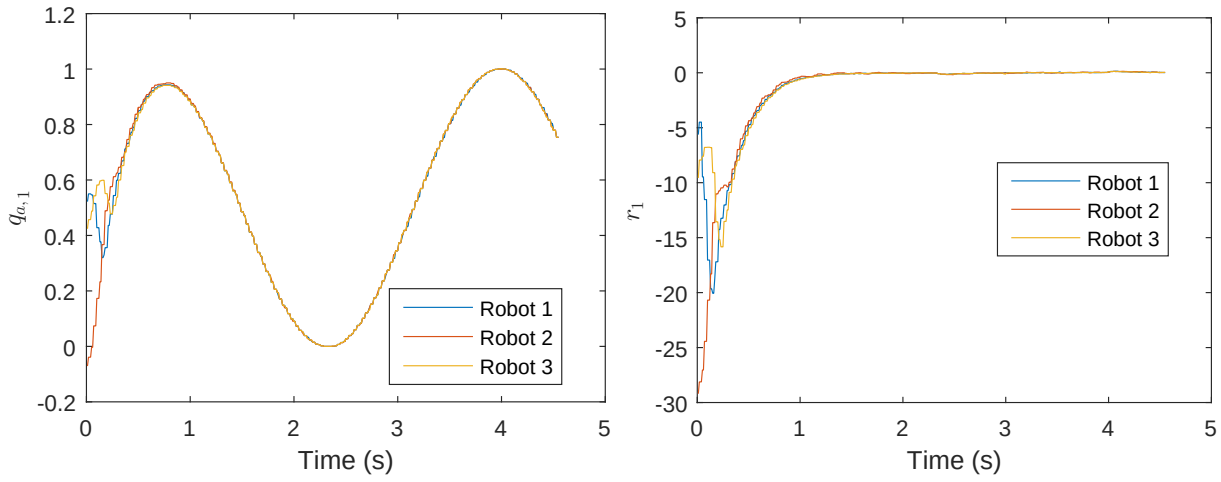


(e) Network delay of 500 ms.



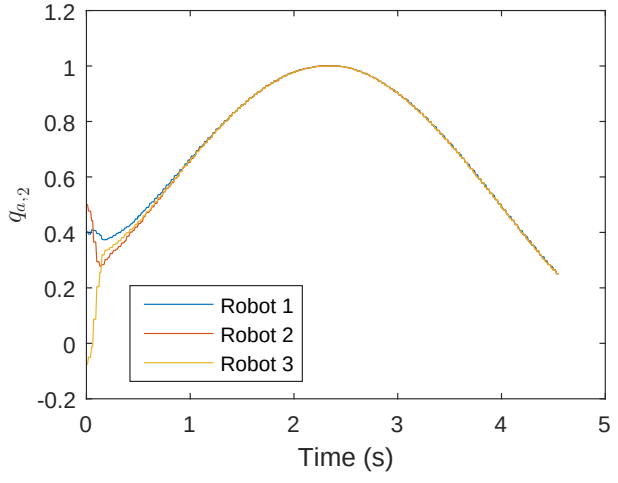
(f) Network delay of 500 ms.

Figure 4.7: Time series for the first joint in the fully connected network topology with $q_d = 0$, for different values of the network delay, in the synchronization before tracking case.

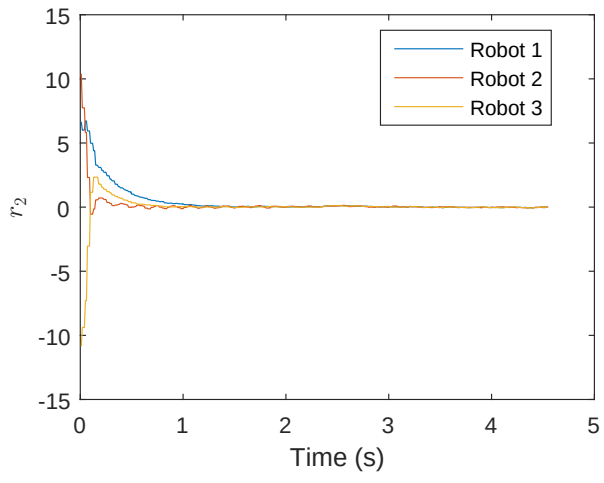


(a) Joint 1.

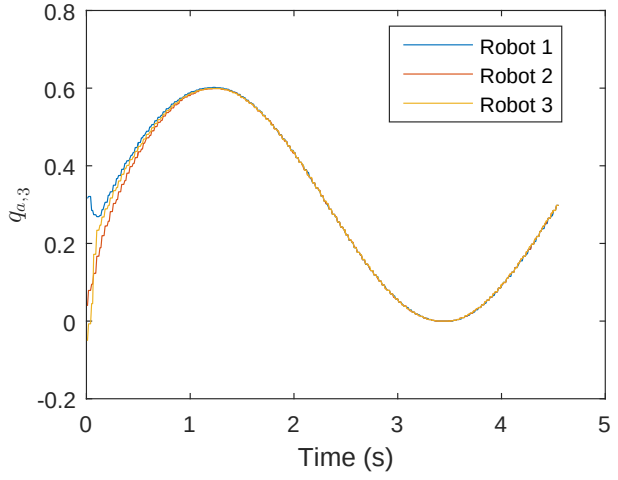
(b) Joint 1.



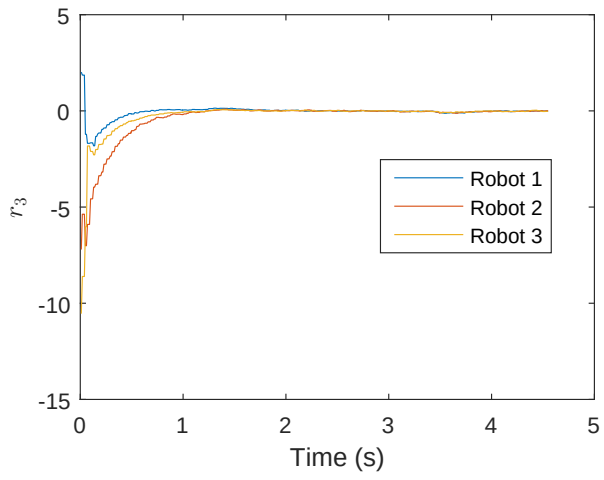
(c) Joint 2.



(d) Joint 2.

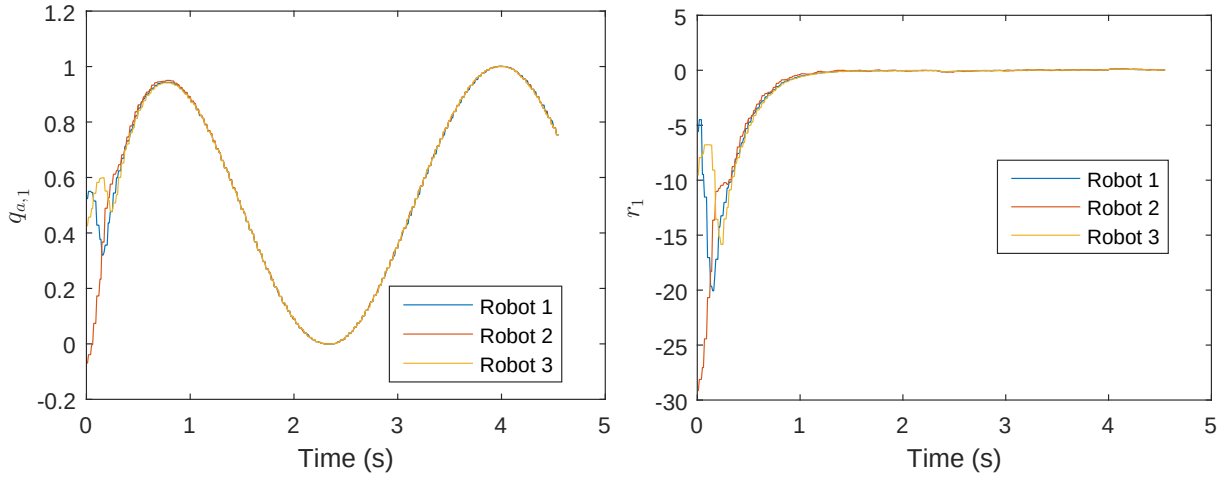


(e) Joint 3.



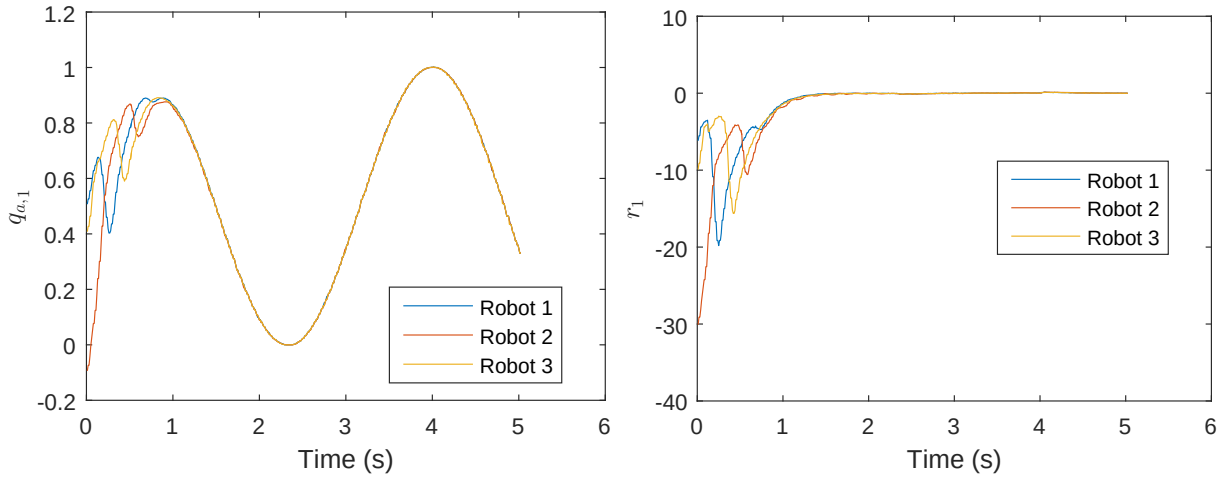
(f) Joint 3.

Figure 4.8: Time series for the circular network topology in the synchronization before tracking case, using the same parameters as in Fig. 4.3, for $k = 5$.



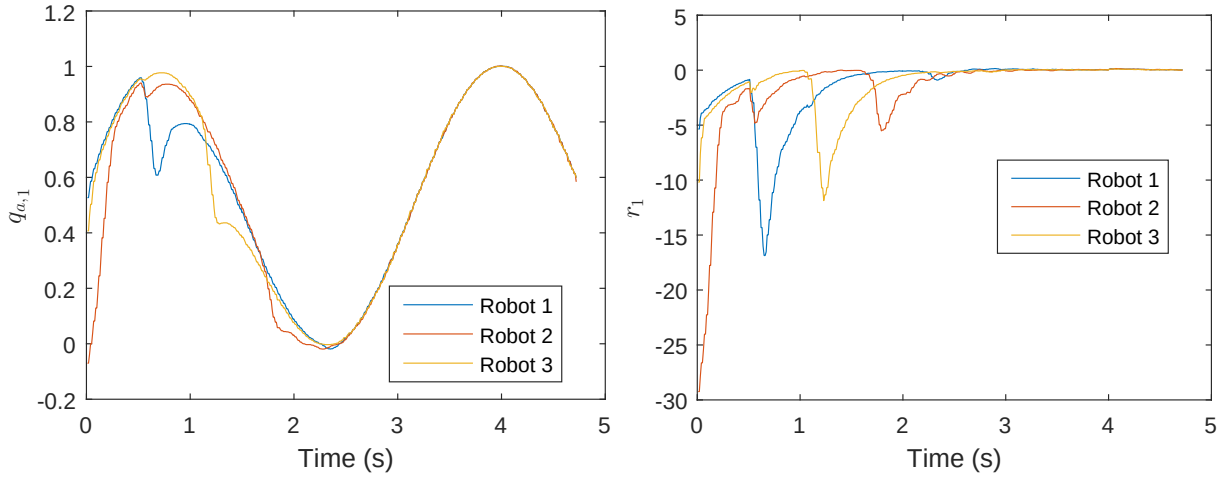
(a) Network delay smaller than 16 ms.

(b) Network delay smaller than 16 ms.



(c) Network delay of 100 ms.

(d) Network delay of 100 ms.



(e) Network delay of 500 ms.

(f) Network delay of 500 ms.

Figure 4.9: Time series for the first joint in the circular network topology, for different values of the network delay, using the same parameters as in Fig. 4.8, in the synchronization before tracking case.

4.1.3 Stability map

To construct a stability map for the synchronization before tracking strategy, experiments were performed on the circular topology with q_d, T and λ identical to the values used to generate Fig. 4.3, with no initial tracking error, and with identical $A_{m,i} = a\mathbb{I}_3$ for a range of values of a. Specifically, for $k = 1$ and $k = 8$, respectively, and for different values of the added network delay δT_s, a was initialized at 5 and, after letting the motion settle for 10 seconds, increased or decreased by multiplication by 0.75 or 1.25. The experiment was stopped when the motion was no longer stable, generating the stability boundary shown in Fig. 4.10.

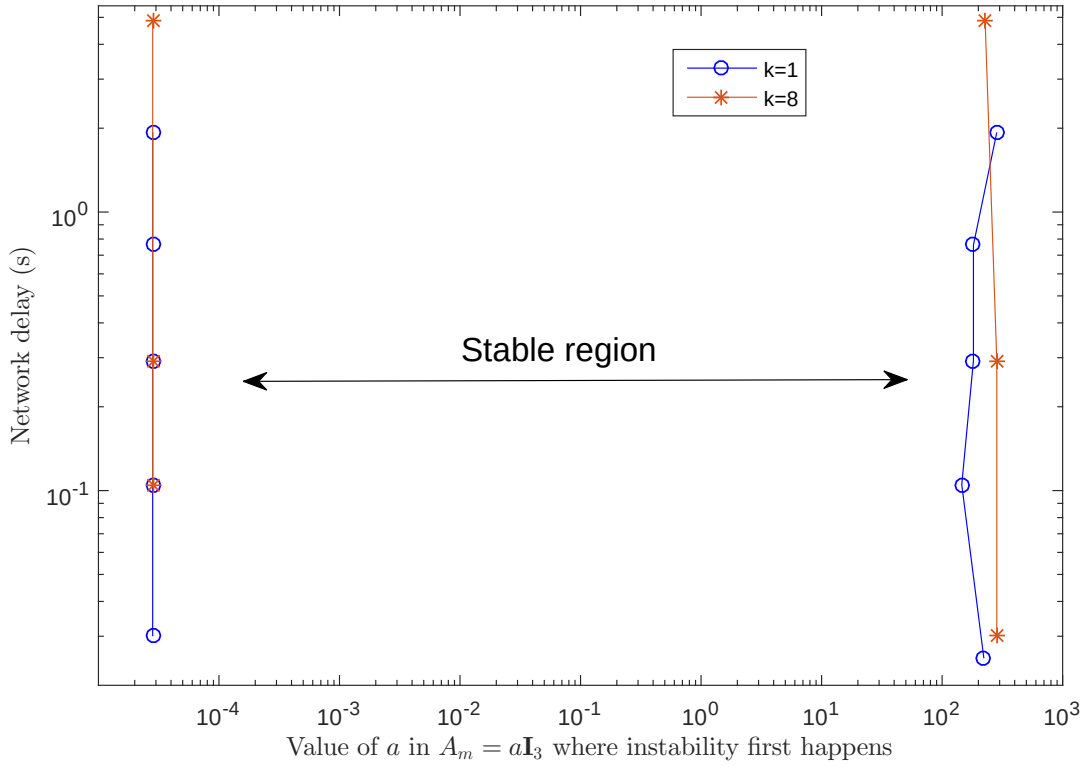


Figure 4.10: Stability map obtained experimentally with a network of three robots in the synchronization before tracking case. The center region between the two boundaries corresponds to parameter values with stable response.

Note that neither k nor the network delay T_s have any noticeable effect on the stability, as was also seen in the numerical results in (Nguyen and Dankowicz, 2016).

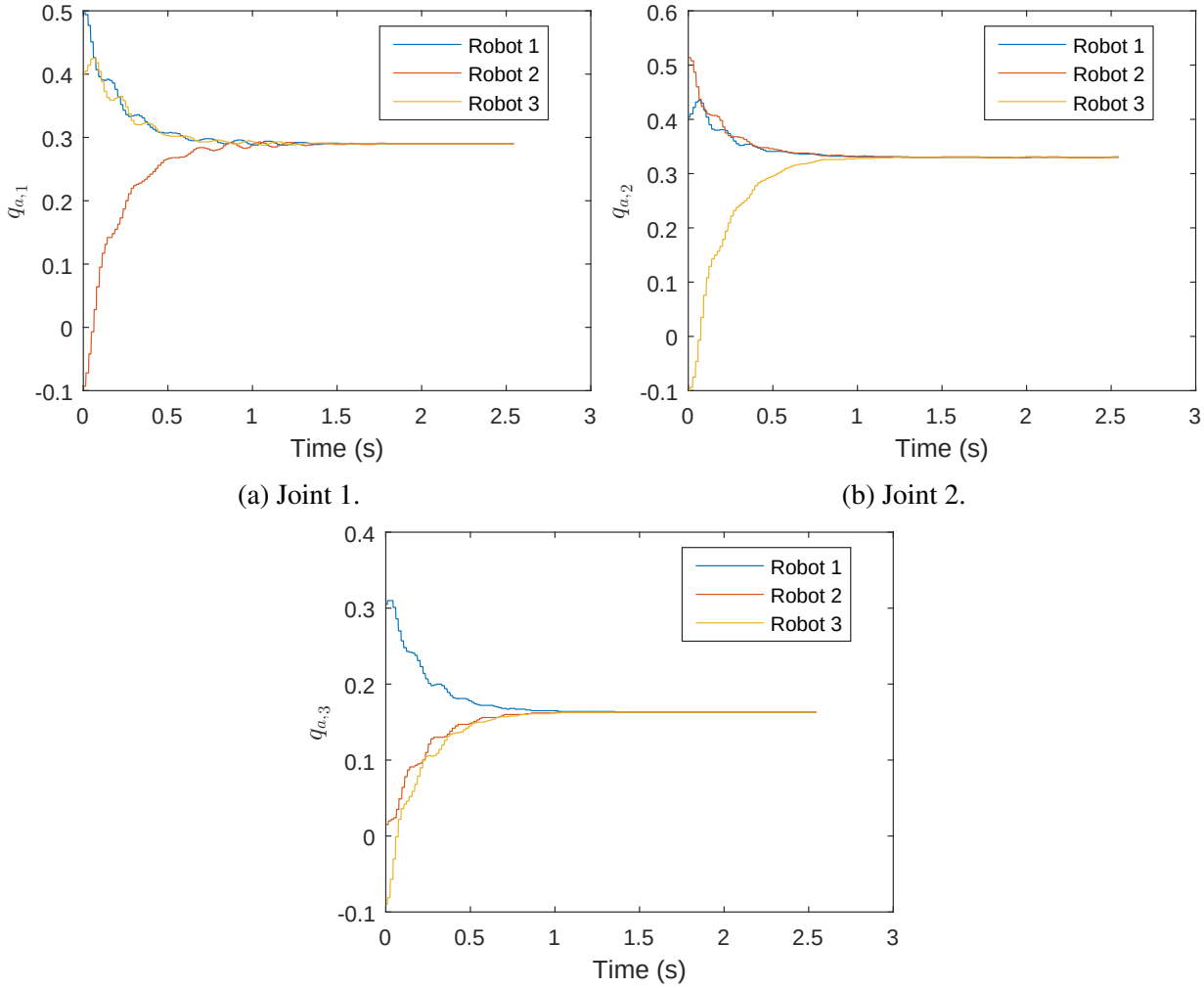
4.2 Consensus

This section reports on results from several experiments using the controller defined in Section 1.2.8 that reproduce the numerical results in Figs. 13-16 in (Nguyen and Dankowicz, 2016). For the same reasons as in Section 4.1, only three degrees of freedom of the manipulators will be used, avoiding problems that could distract the reader from the network behavior we are interested in. Here the controller parameters are $T = 1$ ms, $k = 5$, $A_m = 5\mathbb{I}_3$ and there is no added network delay. Unless otherwise noted, the edge weights are 50, corresponding to the value of λ in the single manipulator setup (Section 3.2), which plays a similar role in the calculation of r. The experiments use several network topologies, with or without a leader. In all the experiments in this chapter, data was collected once for each topology, and the experiment was manually stopped once the robots reached consensus or were close to the desired trajectory of the leader.

4.2.1 Pure consensus without leader

The expected results in this section are similar to those in Fig. 13 in (Nguyen and Dankowicz, 2016). In this section there is no leader and no desired trajectory, so $c_i = 0$ for all i. Two topologies from Fig. 4.1 are used, with the initial configuration in Section 4.1.1. One anticipates that gravity will have an asymmetric effect on the final consensus angles of joints 2 and 3. Also, for joint 1, if two robots are in a symmetric flow of information, such as robots ① and ② in Fig. 4.1 (c), the consensus is expected to take place close to the average of their initial positions, as the two robots should behave approximately in a symmetric way. Figure 4.11 shows the time series for each joint angle for the case of a circular topology. Consensus is quickly achieved, in a similar manner to the numerical results in the cited paper. The results are very similar when a different topology is used; as an example Fig. 4.12 shows the time series for the ① $\rightleftarrows$ ② $\rightarrow$ ③ network topology.

Next, we report on the experimental results using the same initial conditions and the two network topologies in Fig. 4.13. In this case, a minus sign indicates an edge weight of -50. Figure 14 in (Nguyen and Dankowicz, 2016) predicts that a bipartite consensus be achieved. Indeed, as



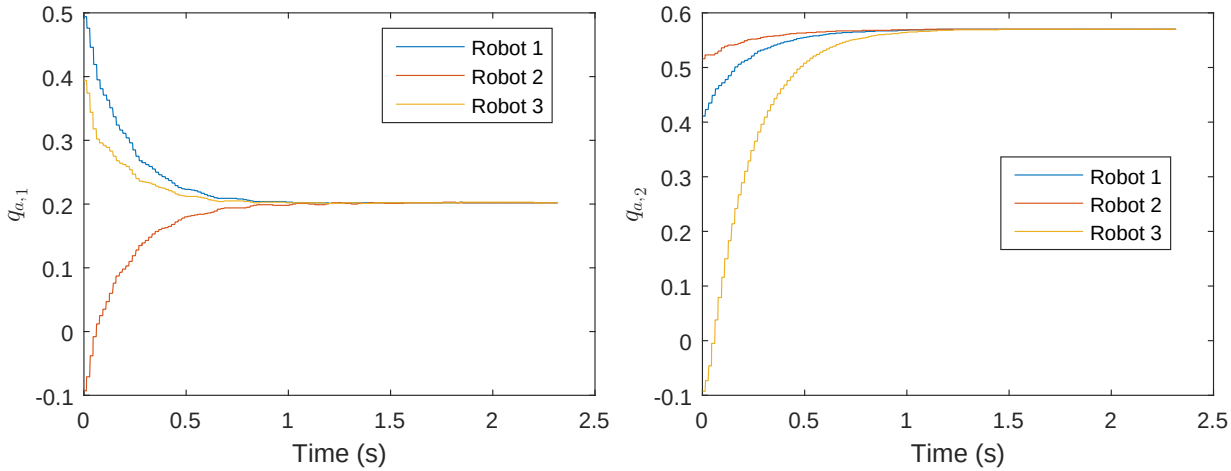
(a) Joint 1.

(b) Joint 2.

(c) Joint 3.

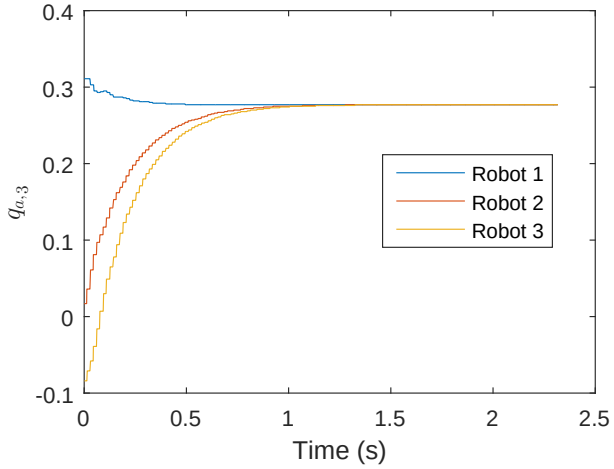
Figure 4.11: Time series for a circular network topology, in leaderless consensus.

seen in Fig. 4.14 a bipartite consensus appears in the earlier part of the time series, but a slow drift towards $q_{a,i} = 0$ is then observed. As shown in Fig. 4.15, this effect disappears when using the ① $\rightleftarrows$ ② $\rightarrow$ ③ network topology (Fig 4.13 (b)), suggesting that the source of the drift is the asymmetry of the network. In the ① $\rightleftarrows$ ② $\rightarrow$ ③ topology, ① and ② create a symmetric network with one robot on each side of the bipartite consensus. Robot ③ does not send information back to ① or ②, so it does not perturb their consensus.



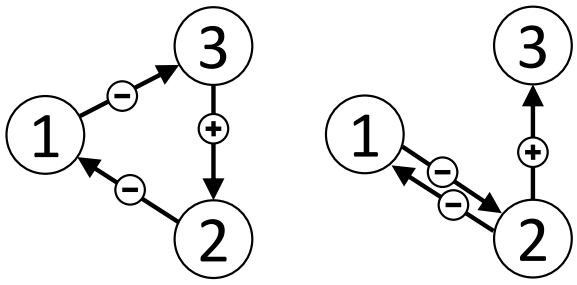
(a) Joint 1.

(b) Joint 2.



(c) Joint 3.

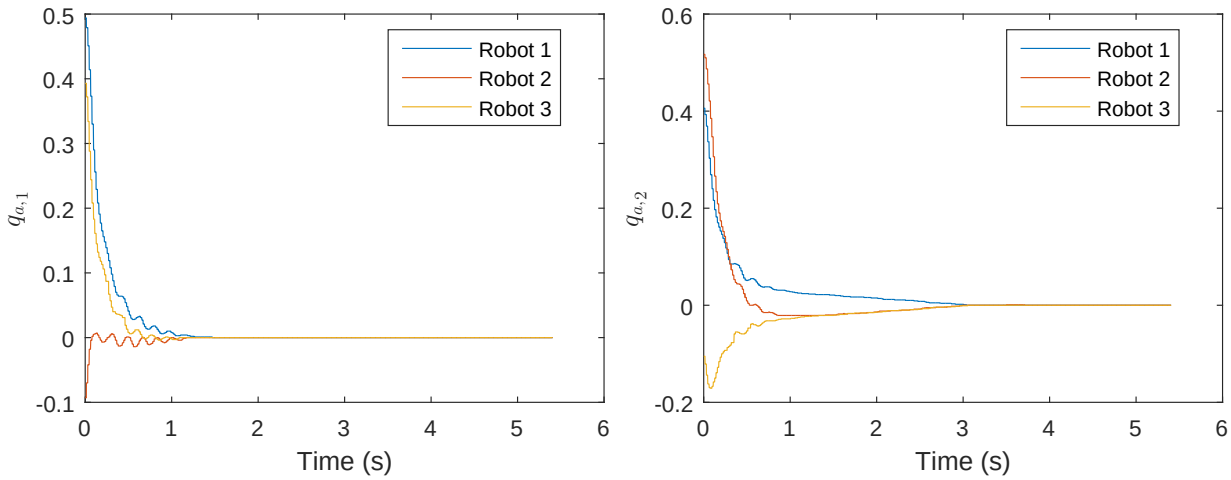
Figure 4.12: Time series for the ① ⇄ ② → ③ network topology, in the case of leaderless consensus.



(a) Circular.

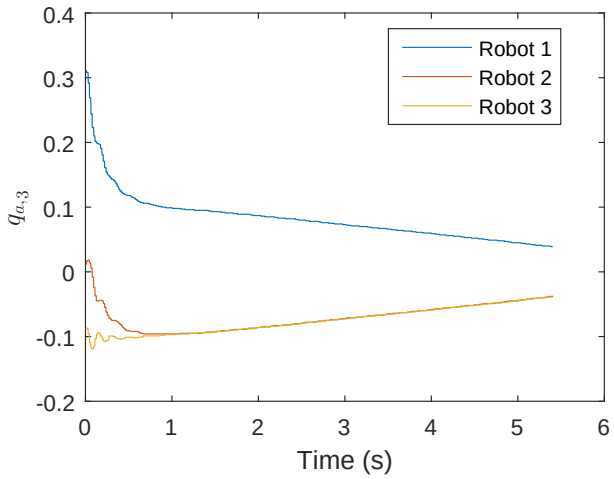
(b) ① ⇄ ② → ③.

Figure 4.13: Different network topologies used in bipartite leaderless consensus experiments. A ⊕ marks a positive weight, while ⊖ represents a negative weight for a connection.



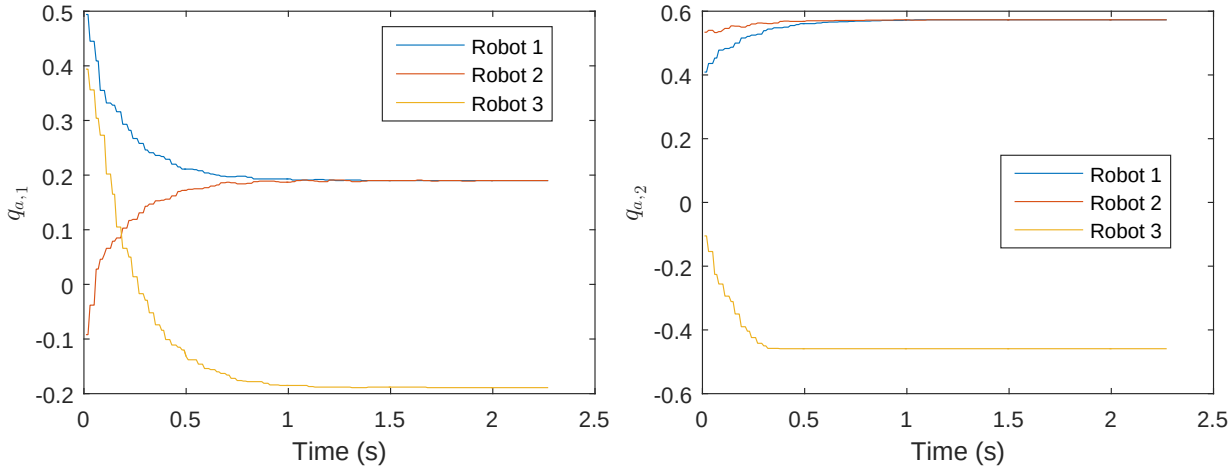
(a) Joint 1.

(b) Joint 2.



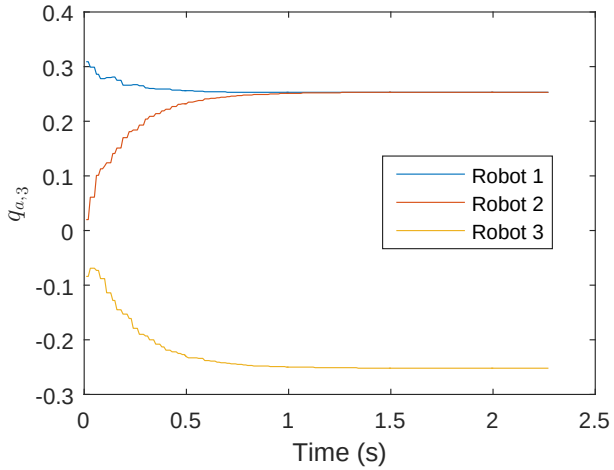
(c) Joint 3.

Figure 4.14: Time series for the circular network topology, in the case of bipartite leaderless consensus.



(a) Joint 1.

(b) Joint 2.



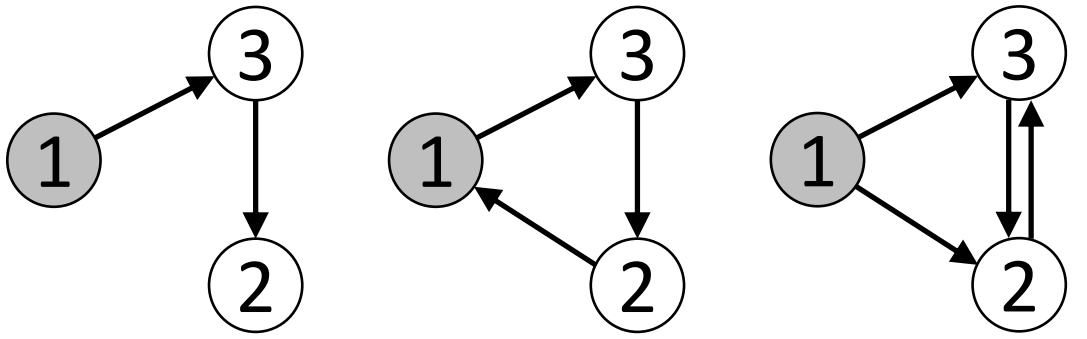
(c) Joint 3.

Figure 4.15: Time series for the ① ⇄ ② → ③ network topology, in the case of bipartite leaderless consensus.

4.2.2 Consensus with a leader and a desired trajectory

This final section reports on experiment performed on the network topologies in Fig. 4.16, where the shaded robot follows a desired trajectory q_d with $c_1 = 50$, and the remaining robots seek consensus, so that $c_i = 0$ for $i \neq 1$.

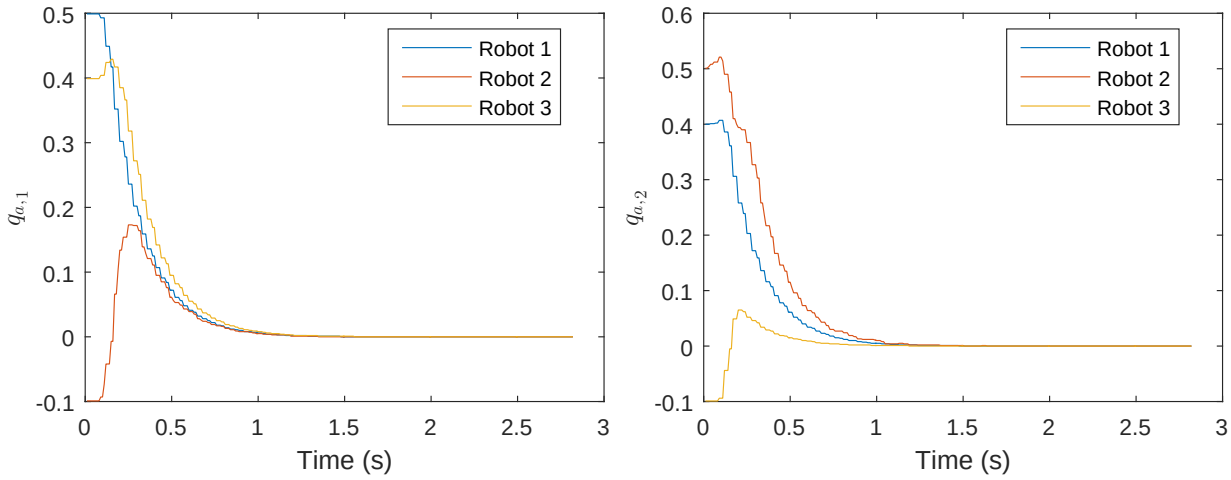
In the first experiment (modeled on Fig. 15 in (Nguyen and Dankowicz, 2016)), $q_d = 0$. Figure 4.17 shows the results for the leader-follower network topology in Fig. 4.16 (a). Very similar results are achieved for the circular network topology (Fig. 4.16 (b)), as seen in Fig. 4.18.



(a) Leader-follower. (b) Circular. (c) ① → ② ⇄ ③ ← ①.

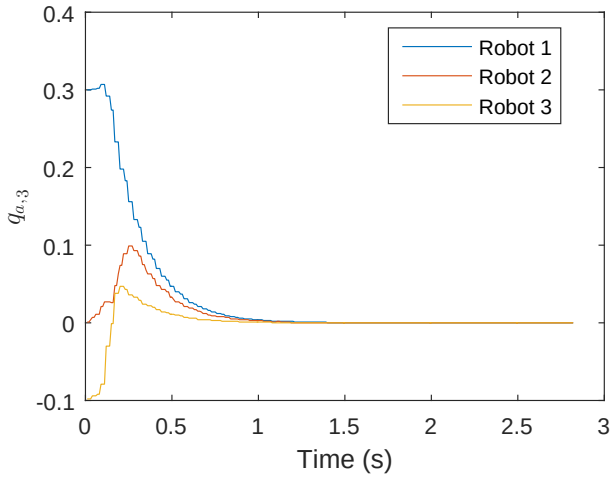
Figure 4.16: Different network topologies used in consensus with a leader. The shaded robot is the leader.

In the second experiment, modelled on Fig. 16 in (Nguyen and Dankowicz, 2016), q_d is given by the sinusoidal desired trajectory in Eq. (4.2). Figure 4.19 shows the results for a leader-follower network topology in Fig. 4.16 (a). Very similar results are obtained with the ① → ② ⇄ ③ ← ① network topology (Fig. 4.16 (c)) in Figure 4.20. These results match qualitatively the numerical predictions in the cited paper. Since the ① → ② ⇄ ③ ← ① topology is symmetric for robots ② and ③, they achieve almost the same consensus angles, and their corresponding time histories overlap. This is very similar to what happens to robots 1 and 2 in Fig. 16 of (Nguyen and Dankowicz, 2016).



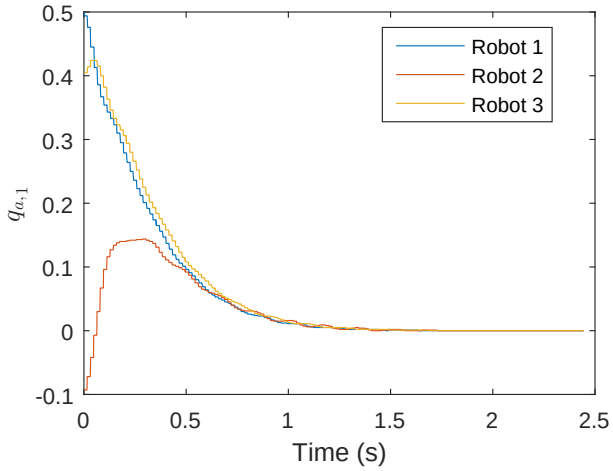
(a) Joint 1.

(b) Joint 2.

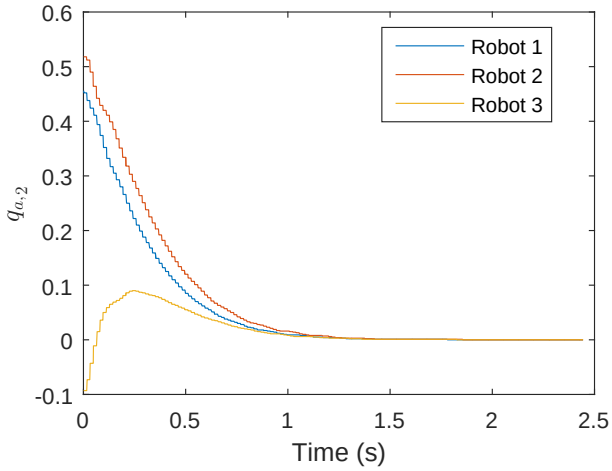


(c) Joint 3.

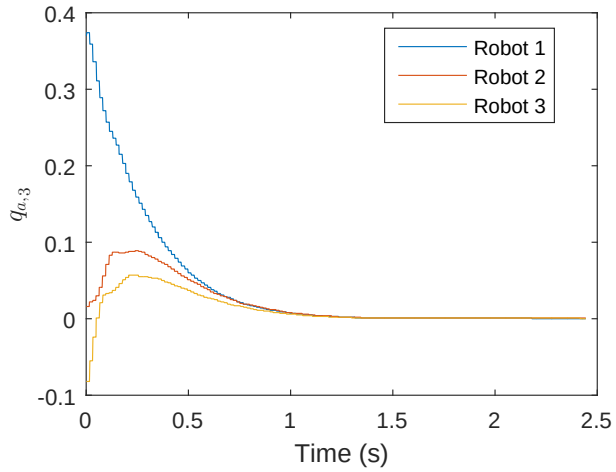
Figure 4.17: Time series of the results for consensus on a leader-follower network topology ① → ② → ③ in which the leader is ①, with $q_d = 0$.



(a) Joint 1.

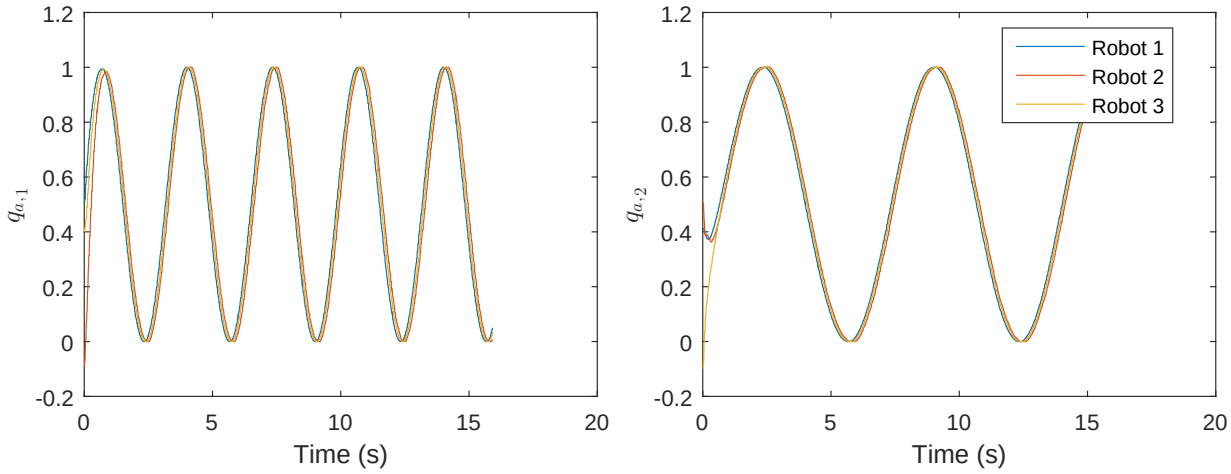


(b) Joint 2.



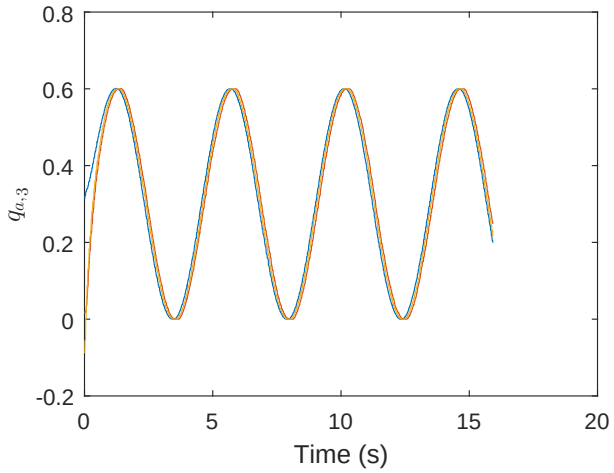
(c) Joint 3.

Figure 4.18: Time series of the results for consensus on a circular network topology on which the leader is ①, with $q_d = 0$.



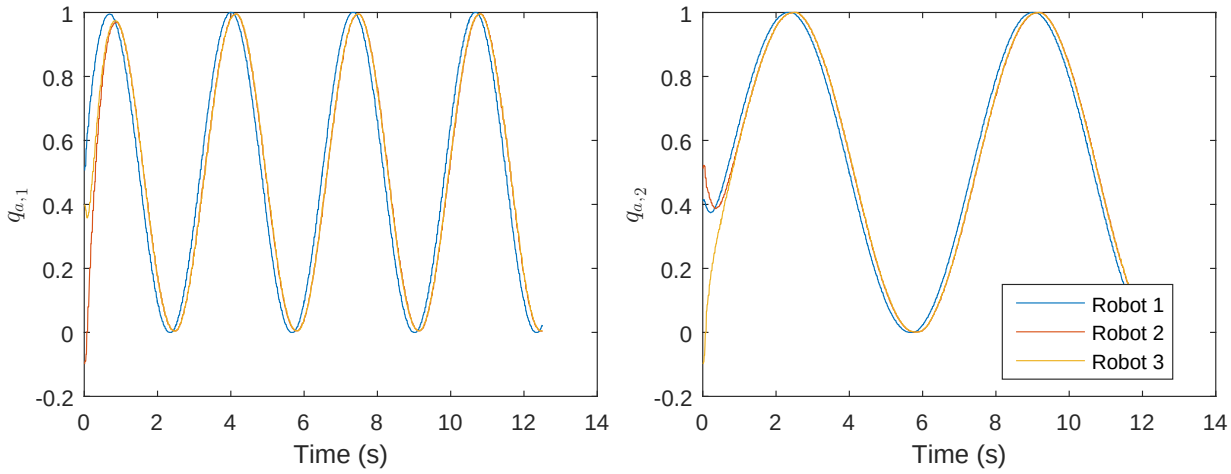
(a) Joint 1.

(b) Joint 2.



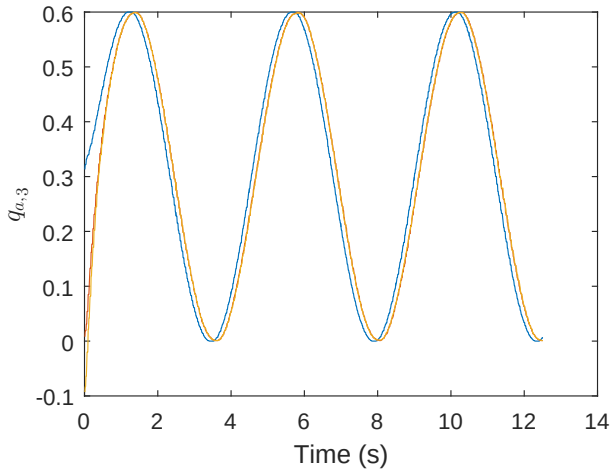
(c) Joint 3.

Figure 4.19: Time series for the ① → ② → ③ topology in the case of consensus with robot ① as leader, with a sinusoidal q_d.



(a) Joint 1.

(b) Joint 2.



(c) Joint 3.

Figure 4.20: Time series for the ① → ② ⇄ ③ ← ① topology with robot ① as leader, with a sinusoidal q_d.

Chapter 5

Conclusions

An extensive experimental program has been carried out to investigate realizations of several $\mathcal{L}_1$ control strategies for single and networked robotic manipulators. Provided that care is taken in the choice of realization, the $\mathcal{L}_1$ control formulation is found to be robust with fast adaptation, without a detailed model of the manipulator. The lesson learned from the experiments in Chap. 3 is that it is highly advisable to use the Cao and Hovakimyan discretization in (Cao and Hovakimyan, 2009) (and Chap. 3.3 of (Hovakimyan and Cao, 2010)) in all experimental setups, as hardware behaves better using this formulation. Nevertheless, for some range of values of the filter bandwidth, both discretizations show trends consistent with theoretical predictions from analysis of the continuous-time system.

A possible improvement to the experimental setup used in this thesis would be the use of a direct measurement of the velocity, as the differentiation of a noisy signal in a small sampling time leads to large relative error. Furthermore, there is opportunity to investigate changes the control design to achieve a higher degree of stabilization the end effector in task space. Future experiments could also consider an agricultural inspection task with a larger manipulator running through a large crop field, or a seed refilling vehicle with a feed mechanism integrated in a manipulator (Li et al., 2016). Inspection tasks often require aggressive stabilization of the end effector. This requires IMUs with higher sampling rate and would greatly benefit from a kinematic design that decouples inertial forces from the end effector orientation, such as gimbals.

The reported experimental study of synchronization and consensus in networks of robotic manipulators validates the control strategies in (Nguyen and Dankowicz, 2016). The experimental results, obtained for different topologies of three robots, agree qualitatively with the predictions

from simulations in (Nguyen and Dankowicz, 2016) for four robots. Even though there was no external motion imposed on the platforms, these experiments give valuable information about the performance of the controller in a homogeneous network and the insensitivity to network time delay. The results from Chap. 3 show that robots are able to respond to disturbances from the terrain and the results in Chap. 4 show that the network does not make the system unstable. This gives confidence that the combination will result in a stable controller for networked manipulators on platforms excited by external disturbances.

Future research on larger networks with more manipulators will probably encounter rich interaction between the robots, depending on the different network topologies. Heterogeneous networks also provide an interesting future research topic, including manipulators with different mass properties, or even completely different robotic mechanisms. The theory assumes that the disturbances on the platform are bounded, and that all the robots are mounted in the same platform, which is so massive that its dynamics dominate, instead of being influenced by the manipulators. Different platform couplings could be researched, including examples where the influence of the dynamics of the manipulator on the platform is large. In addition, it would be worth researching other higher level strategies for consensus and synchronization, such as planning a path leading to the desired consensus while using separate $\mathcal{L}_1$ controllers to follow the individual paths.

A different branch of possible future research includes more complex cooperation tasks for networked robots, in applications as diverse as space exploration (Stroupe et al., 2005), in-flight seed refilling (Li et al., 2016), or underwater docking (Majid et al., 2014). Examples of such tasks include coordinated handling of an object or interaction with the environment, as well as transferring objects between robots.

References

Antonelli, G. and Leonessa, A. (2008). Underwater robots: motion and force control of vehicle-manipulator systems. *IEEE Control Systems Magazine*, pages 138–139.

Ball, D., Upcroft, B., Wyeth, G., Corke, P., English, A., Ross, P., Patten, T., Fitch, R., Sukkarieh, S., and Bate, A. (2016). Vision-based obstacle detection and navigation for an agricultural robot. *Journal of Field Robotics*.

Bennehar, M., Chemori, A., and Pierrot, F. (2015). $\mathcal{L}_1$ adaptive control of parallel kinematic manipulators: Design and real-time experiments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1587–1592.

Berghuis, H. and Nijmeijer, H. (1993). A passivity approach to controller-observer design for robots. *IEEE Transactions on Robotics and Automation*, 9(6):740–754.

Bogacki, P. and Shampine, L. F. (1989). A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*, 2(4):321–325.

Cao, C. and Hovakimyan, N. (2009). L1 adaptive output-feedback controller for non-strictly-positive-real reference systems: missile longitudinal autopilot design. *AAIA Journal of Guidance, Control, and Dynamics*, 32(3):717–726.

Cheein, F. and Carelli, R. (2013). Agricultural robotics: Unmanned robotic service units in agricultural tasks. *IEEE Industrial Electronics Magazine*, 7(3):48–58.

Chen, G. (2015). Cooperative controller design for synchronization of networked uncertain euler–lagrange systems. *International Journal of Robust and Nonlinear Control*, 25(11):1721–1738.

Chung, S.-J. and Slotine, J.-J. E. (2007). Cooperative robot control and synchronization of lagrangian systems. In *Proceedings of 46th IEEE Conference on Decision and Control*, pages 2504–2509.

Chung, S.-J. and Slotine, J.-J. E. (2009). Cooperative robot control and concurrent synchronization of lagrangian systems. *IEEE Transactions on Robotics*, 25(3):686–700.

Craig, J. J., Hsu, P., and Sastry, S. S. (1987). Adaptive control of mechanical manipulators. *The International Journal of Robotics Research*, 6(2):16–28.

DesForges, D. (1979). The application of model-referenced adaptive control to robotic manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 101:193.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Edan, Y., Rogozin, D., Flash, T., and Miles, G. (2000). Robotic melon harvesting. *IEEE Transactions on Robotics and Automation*, 16(6):831–835.

Foglia, M. M. and Reina, G. (2006). Agricultural robot for radicchio harvesting. *Journal of Field Robotics*, 23(6-7):363–377.

Frazzoli, E., Dahleh, M. A., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129.

Hovakimyan, N. (retrieved 2017). Naira Hovakimyan research website. *http://naira-hovakimyan.mechse.illinois.edu/l1-adaptive-control-tutorials/*.

Hovakimyan, N. and Cao, C. (2010). $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H. D., Aghazarian, H., Ganino, A. J., Garrett, M., Joshi, S. S., and Schenker, P. S. (2003). CAMPOUT: A control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(5):550–559.

Johnson, D. A., Naffin, D. J., Puhalla, J. S., Sanchez, J., and Wellington, C. K. (2009). Development and implementation of a team of robotic tractors for autonomous peat moss harvesting. *Journal of Field Robotics*, 26(6-7):549–571.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. *TR 98-11, Computer Science Dept., Iowa State University*.

Lee, S. Y., Lee, Y. S., Park, B. S., Lee, S. H., and Han, C. S. (2007). MFR (multipurpose field robot) for installing construction materials. *Autonomous Robots*, 22(3):265–280.

Li, Y., Nguyen, K., and Dankowicz, H. (2016). A robust adaptive controller for a seed refilling system on a moving platform. In *Proceedings of 5th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture*, volume 49(16), pages 341–346. Elsevier.

Mahony, R., Hamel, T., and Pflimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218.

Majid, M., Yahya, M., Siang, S., and Arshad, M. (2014). Cooperative positioning of multiple AUVs for underwater docking: A framework. In *Proceeding of Colloquium on Robotics, Unmanned Systems and Cybernetics*, page 1.

Minshall, G., Saito, Y., Mogul, J. C., and Verghese, B. (2000). Application performance pitfalls and TCP's nagle algorithm. *ACM Sigmetrics Performance Evaluation Review*, 27(4):36–44.

Moin, P. (2010). *Fundamentals of Engineering Numerical Analysis*, chapter 4. Cambridge University Press.

Morales, J., Martinez, J. L., Mandow, A., Seron, J., and Garcia-Cerezo, A. J. (2013). Static tip-over stability analysis for a robotic vehicle with a single-axle trailer on slopes based on altered supporting polygons. *IEEE/ASME Transactions on Mechatronics*, 18(2):697–705.

Nagle, J. (1984). Congestion control in IP/TCP internetworks. *Internet Engineering Task Force, RFC 896*.

Nguyen, K. and Dankowicz, H. (2014). Synchronization and consensus of a robot network on an underactuated dynamic platform. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 117–122.

Nguyen, K. and Dankowicz, H. (2015). Adaptive control of underactuated robots with unmodeled dynamics. *Robotics and Autonomous Systems*, 64:84–99.

Nguyen, K. and Dankowicz, H. (2016). Cooperative control of networked robots on a dynamic platform in the presence of communication delays. *International Journal of Robust and Nonlinear Control*.

Nguyen, K., Li, Y., and Dankowicz, H. (2015). Delay robustness of an $\mathcal{L}_1$ adaptive controller for a class of systems with unknown matched nonlinearities. Unpublished, under review.

Nguyen, K., Rodriguez Reina, A., and Dankowicz, H. (2016). End-effector control for manipulators operating on dynamic platforms. Unpublished.

Nicosia, S. and Tomei, P. (1984). Model reference adaptive control algorithms for industrial robots. *Automatica*, 20(5):635–644.

Niemeyer, G. and Slotine, J.-J. (1991). Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1):152–162.

Ortega, R. and Spong, M. W. (1989). Adaptive motion control of rigid robots: A tutorial. *Automatica*, 25(6):877–888.

Ortega, R., Van Der Schaft, A., Maschke, B., and Escobar, G. (2002). Interconnection and damping assignment passivity-based control of port-controlled hamiltonian systems. *Automatica*, 38(4):585–596.

Reid, J. F., Zhang, Q., Noguchi, N., and Dickson, M. (2000). Agricultural automatic guidance research in north america. *Computers and Electronics in Agriculture*, 25(12):155 – 167.

Slaughter, D., Giles, D., and Downey, D. (2008). Autonomous robotic weed control systems: A review. *Computers and Electronics in Agriculture*, 61(1):63 – 78.

Slotine, J.-J. E. and Li, W. (1987). On the adaptive control of robot manipulators. *The International Journal of Robotics Research*, 6(3):49–59.

Stroupe, A., Huntsberger, T., Okon, A., Aghazarian, H., and Robinson, M. (2005). Behavior-based multi-robot collaboration for autonomous construction tasks. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Tomizuka, M. (1986). An adaptive control scheme for mechanical manipulators-compensation of nonlinearity and decoupling control. *Journal of Dynamic Systems, Measurement, and Control*, 108:127–136.

van Henten, E., Schenk, E., van Willigenburg, L., Meuleman, J., and Barreiro, P. (2010). Collision-free inverse kinematics of the redundant seven-link manipulator used in a cucumber picking robot. *Biosystems Engineering*, 106(2):112 – 124.

van Henten, E., Slot, D. V., Hol, C., and Willigenburg, L. V. (2009). Optimal manipulator design for a cucumber harvesting robot. *Computers and Electronics in Agriculture*, 65(2):247 – 257.

Wang, H. (2013a). Flocking of networked uncertain Euler–Lagrange systems on directed graphs. *Automatica*, 49(9):2774–2779.

Wang, H. (2013b). Passivity based synchronization for networked robotic systems with uncertain kinematics and dynamics. *Automatica*, 49(3):755–761.

Wang, H. (2013c). Task-space synchronization of networked robotic systems with uncertain kinematics and dynamics. *IEEE Transactions on Automatic Control*, 58(12):3169–3174.

Wang, H. (2014). Consensus of networked mechanical systems with communication delays: A unified framework. *IEEE Transactions on Automatic Control*, 59(6):1571–1576.

Wilson, J. (2000). Guidance of agricultural vehicles a historical perspective. *Computers and Electronics in Agriculture*, 25(12):3 – 9.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151.

Zhao, Y., Duan, Z., and Wen, G. (2015). Distributed finite-time tracking of multiple Euler–Lagrange systems without velocity measurements. *International Journal of Robust and Nonlinear Control*, 25(11):1688–1703.

Zhou, Y., Cheng, N., Lu, N., and Shen, X. S. (2015). Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture. *IEEE Vehicular Technology Magazine*, 10(4):36–44.