Cooperative $r$-Passivity-Based Control

Development of a Multi-Agent Passivity-Based Control Scheme with Robustness towards Network Effects

O. de Groot
Cooperative $r$-Passivity-Based Control

Development of a Multi-Agent Passivity-Based Control Scheme
with Robustness towards Network Effects

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft University of Technology

O. de Groot

December 5, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology
In this work we consider the problem of cooperative formation control between heterogeneous agents when time-varying delays and/or packet loss are present. Specifically, we introduce a control law for nonlinear fully actuated mechanical agents that separates the cooperative coordinates from the local coordinates, which removes the necessity for scenario-dependent tuning. The cooperative outputs encode task-space coordinates and velocities which are transformed into wave-variables to overcome the destabilising effects of the communication network. The cooperative agent couplings incorporate task-space constraints such as collision- and singularity avoidance, while ensuring performance in arbitrary workspaces. The proposed approach improves robustness of existing methods against network effects, and allows to expand their application scope by the inclusion of constraints and nonlinear dynamics. In addition, our approach is scalable by design, and simplifies the tuning task considerably. We demonstrate the efficacy of the proposed approach experimentally.
# Table of Contents

## Acknowledgements

## 1 Introduction

1.1 Motivation and Vision .................................................. 1
1.2 Introduction to the Scope of This Work ............................... 2
   1.2.1 Control of Nonlinear Systems .................................. 2
   1.2.2 Homogeneous and Heterogeneous Multi-Agent Systems ...... 2
   1.2.3 Dynamical Effects of the Communication Network .......... 2
   1.2.4 Fully Actuated and Underactuated Systems .................... 3
   1.2.5 Collision Avoidance ............................................. 3
1.3 Introduction to Cooperative Passivity-Based Control ............... 3
   1.3.1 Cooperative Multi-Agent Control ............................... 3
   1.3.2 Passivity-Based Control ........................................ 4
1.4 Previous Works ........................................................... 4
   1.4.1 Network Effects ................................................ 4
   1.4.2 Collision Avoidance ............................................. 5
1.5 Novelty of This Work .................................................... 5
1.6 Notation .................................................................. 5
1.7 Layout of This Work ....................................................... 6

## 2 Modelling of Mechanical Systems

2.1 Modelling of Mechanical Agents ....................................... 9
   2.1.1 System Coordinates and Velocities ............................. 9
   2.1.2 Hamiltonian Mechanical Modelling ............................. 9
2.2 End-Effector Dynamics and Task-Space Control ..................... 10
2.3 Problem Definition .......................................................... 11
# Table of Contents

## 3 Cooperative Passivity-Based Control with Delays

3.1 Passivity .................................................. 13
  3.1.1 Definition of Passivity .................................. 13
  3.1.2 Passivity of Hamiltonian Systems ......................... 14
3.2 The Scattering Transformation ............................. 14
  3.2.1 Definition and Passivity ................................ 15
  3.2.2 Impedance Matching .................................... 16
3.3 The Cooperative Passivity-Based Control Scheme .......... 16
3.4 Full State r-Passivity .................................... 17
3.5 Summary ................................................. 18

## 4 Cooperative r-Passivity-Based Control

4.1 Task-Space r-Passivity and Potential-Based Cooperative Controls .... 19
  4.1.1 Potential-Based Cooperative Controls .................... 19
  4.1.2 Algebraic Loop Occurrence and Solutions for Linear Couplings .... 20
4.2 Cooperative Coordinate Synchronisation with Task-Space r-Passivity .... 21
4.3 Leader-Follower or Leaderless Formation Control ............ 23
  4.3.1 Formation Control ..................................... 23
  4.3.2 Leaders ............................................. 24
4.4 Summary ................................................. 24

## 5 Local Feedback Control for r-Passivity

5.1 Local Control for r-Passivity ............................... 25
  5.1.1 Scaled Dynamics ....................................... 27
  5.1.2 Remark on the Cooperative Dynamics ................... 28
5.2 Controlling the Excess Degrees of Freedom .................. 29
5.3 Extension to the Euler-Lagrange Framework .................. 30
5.4 Convergence of the Full Scheme ................................ 31
5.5 Summary .................................................. 31
5.6 Example: Fully Cooperative Control of Two Fully Actuated Point-Masses .... 32

## 6 Simulation Results

6.1 Modelling of a Fully Actuated N-Link Manipulator ............ 33
6.2 Simulation Results ........................................ 35
  6.2.1 Consensus Control .................................... 35
  6.2.2 Formation Control .................................... 36

O. de Groot ................................................. Master of Science Thesis
# Table of Contents

## II Implementation of Cooperative r-Passivity-Based Control

### 7 Navigation Functions and Discrete-Time Scattering

#### 7.1 Navigation Functions and Scattering

- 7.1.1 Introduction to Artificial Potential Functions and Navigation Functions
- 7.1.2 Real-Time Solution of Algebraic Loops for Navigation Functions with the Scattering Transformation
- 7.1.3 Impedance Matching

#### 7.2 Discrete-Time rPBC and Wave-Variable Reconstruction

- 7.2.1 Discretisation of rPBC
- 7.2.2 Wave-Variable Reconstruction

#### 7.3 Summary

## 8 Experimental Setup

### 8.1 Description

### 8.2 Software Architecture

- 8.2.1 Server-Side Implementation
- 8.2.2 Client-Side Implementation

### 8.3 Implementation of PBC for Differential Drive Robots

- 8.3.1 Feedback Linearisation and Internal Accelerations
- 8.3.2 Cooperative Model
- 8.3.3 Odometry and Vision Measurements

### 8.4 Implementation of PBC for a 7-DOF Robotic Manipulator

### 8.5 Additional Cooperative Features

- 8.5.1 Artificial Potential Functions
- 8.5.2 Cooperative Selectors

## III Experimental Results

### 9 Experimental Results

- 9.1 Emulated Network
- 9.2 Single-Agent Validation of Local Control
- 9.3 Two-Dimensional Heterogeneous Formation Control
- 9.4 Validation of Robustness to Network Effects
- 9.5 Comparison with IDA-PBC
# Conclusions and Recommendations

## 10 Conclusions and Discussion

10.1 Conclusions ........................................... 65
10.2 Discussion ........................................... 66

10.2.1 Heterogeneous and Nonlinear Control .............. 66
10.2.2 Fully Actuated and Underactuated Systems ........ 66
10.2.3 Robustness towards Network Effects .............. 66
10.2.4 Collision Avoidance ................................ 66
10.2.5 Concluding Remarks ................................ 67

## 11 Recommendations

11.1 Extension to Underactuated Systems .................. 69
11.2 Collision Avoidance ................................... 70

11.2.1 Static Point Computations ............................ 70
11.2.2 Passive Projection of Constraints on the Cooperative Input ......... 70

11.3 Cooperative Manipulation ................................ 71
11.4 High-level Control with Low-Level rPBC ............. 71
11.5 Model Predictive Control for r-Passivity ............... 72
11.6 Application to Other Fields of Research ............... 72

# Notes on Tuning rPBC

## B IDA-PBC Performance Improvements

B.1 Pseudo-Inverse Approach ................................ 75
B.2 Normalised Goal Functions ............................... 75

## C Selected C++ Implementations

C.1 Edge Interface ........................................ 78
C.2 Edge Implementation with NF and WVM ............... 79
C.3 CMM Interface ........................................ 81
C.4 Controller Interface ................................... 82
C.5 System Interface ...................................... 83

# Bibliography
List of Figures

3-1 The multi-agent scheme as in [1], referred to as Scheme 1. ............... 16

6-1 Schematic of a 3-link manipulator as modelled in this section. .......... 33

7-1 Numerical validation of the iterative method for solving the algebraic loop formed between controls and the ST. The control input calculated with the NF is drawn as dashed line. The steps of the iterative method and the static point are shown with green circles and a red cross, respectively. The computed value of $r_{js}$ is shown on the y-axis. The iteration converges to the static point where the x and y axes match. ......................... 42

8-1 The experimental setup .................................................. 45
8-2 Diagram of the server side functionalities and communication. .......... 46
8-3 FSM of the server side implementation. ................................ 47
8-4 Implementation diagram of cooperative rPBC for a single agent. ....... 47
8-5 Schematic view of a differential drive robot and the hand position. .... 48

9-1 Artificial delays generated via random walk processes. .................. 55
9-2 A demonstration of the interaction between the cooperative and local objectives using photos of a single-agent experiment. ................. 56
9-3 Trajectories of the agents in the ground plane. The initial and final positions of the agents are denoted with a square and circle respectively. The bounds in the x and y axes are drawn as black solid lines. ...................... 57
9-4 Cooperative coordinate evolution over time. .......................... 58
9-5 Cooperative velocities over time. ..................................... 58
9-6 Local velocities of the manipulator over time. ......................... 58
9-7 Robustness results of the presented method towards time delays and packet loss. .................................................. 59
9-8 Trajectories of IDA-PBC and r-PBC in a consensus scenario under two sets of delays. .......................... 60
9-9 A comparison of the coordinate evolution between cooperative IDA-PBC and rPBC for delays in the range 1000 ms – 1200 ms. .................... 61
11-1 An illustration of the projection process. The position of two agents is indicated as well as a singular point for agent $i$, denoted $z_{s,i}$. The input vector is projected such that a margin towards the singular point is maintained. . . . . . . . . . . 70
List of Tables

6-1 Simulation parameter values. ........................................ 35

9-1 Experimental parameter values. ................................. 57
Acknowledgements

This work is a continuation of the Thesis work from Laurens Valk on cooperative IDA-PBC ([2]). In extending these theories the research path lead to the discovery of a different control scheme which is presented in this Thesis. I would like to thank my supervisor Tamás Keviczky for his guidance and support throughout this project.

The practical setup was facilitated by the faculty of Aerospace engineering, for which I would like to express gratitude here. I would also like to thank Arjan Vermeulen, Gijs van de Hoorn and my DCSC lab mates: Daniel and Maurits.

Delft, University of Technology
December 5, 2019
Intelligent robotic systems are soon to become a staple in human life. Promising developments such as autonomous vehicles and the Internet Of Things are indicators of upcoming technological advancements that may reshape society. In industry, robotic systems are already ubiquitous, taking responsibility for a wide variety of tasks. A primary logistical example of such a task is the automatic storage and retrieval of stock items in automated warehouse, the operation of which depends solely on robots.

Further increase in robotic applications is warranted by the deployment of robotic swarms. These systems, exemplified by drone fleets or autonomous vehicle platoons, are made-up of of smaller systems that work cooperatively. The on-demand scaling capabilities, inherent to these systems, increases operational flexibility and efficiency, which opens up new applications in various domains. Following these developments over the past decades, we may expect the presence of robotic systems to increase in the years to come. This work anticipates on the related technical requirements of these up-and-coming applications.

1.1 Motivation and Vision

The above mentioned trends warrant an increase in interactions between robotic systems of various types in many different environments. State-of-the-art technologies are not yet capable of handling the variety of events and environments that a system may encounter in these applications. This work aims to contribute to the state-of-the-art by considering a wide variety of realistic issues in both a theoretical and practical framework. The purpose of this work, generally stated, is

“Developing a control approach that allows the controllers of arbitrary mechanical systems to be interconnected for the fulfilment of cooperative tasks in general environments.”

This objective fundamentally requires a reliable and robust control approach that may be implemented on any mechanical system. Hence the proposed application is effectively a low-level control layer or protocol, i.e., an approach that is implemented on a wide-variety of
Introduction

systems as a medium to facilitate higher level control applications. The main challenges in the development of such an approach are robustness and generality. In this work we aim to develop a scheme that theoretically matches these requirements by design. Since the robustness and performance of such a method may only be shown in practice we validate our approach experimentally.

1.2 Introduction to the Scope of This Work

The objective of this work comprises numerous non-trivial subproblems, each of which impairs the real-life application or application scope of a control methodology. The following section introduces the reader to these subproblems.

1.2.1 Control of Nonlinear Systems

In control engineering we often distinguish between two control categories: linear and nonlinear. All real-world systems are dynamically nonlinear. A linear control approach simplifies design by linearising a nonlinear system around one or more operating points. Using linear control techniques, such as PID or LQR, the system is then stabilised around an operating point. Although this approach works for many systems, there are numerous exceptions where the nonlinear dynamics are too dominant to be disregarded or there are too many realistic operating points for efficient linear control. A primary example of such a system is a robotic manipulator. All reachable end-effector positions of this system are realistic operating points and its location is a heavily nonlinear product of sines and cosines. In these cases we are dependent on nonlinear control techniques. Nonlinear dynamical systems, in the general sense, include any system that is described with nonlinear differential equations, which is an immensely large set of dynamics for which there exists no one-size-fits-all control law. This work therefore considers the subset of nonlinear systems that include robotic and mechanical systems. The dynamics of these systems are sufficiently smooth and can directly be related to a description of energy, the relevancy of which is detailed in later parts of this introductory section.

1.2.2 Homogeneous and Heterogeneous Multi-Agent Systems

One necessary technical distinction between multi-agent control schemes is that between homogeneous and heterogeneous multi-agent systems. Homogeneous multi-agent systems consist of a single system type, i.e., the dynamics of each system are strictly identical. Examples of these systems are swarms of drones or top-down platooning approaches. This work is not aimed at these specific applications, which therefore requires the control approach to work with heterogeneous (non-identical) systems.

1.2.3 Dynamical Effects of the Communication Network

Cooperation between systems is almost always facilitated by information exchange over a communication network. In the physical world this information exchange cannot occur instantaneously. In fact, information transmission over any practical network is subject to delay.
The effects of industrially standard networks, such as Ethernet, include delays that vary over time as well as loss of data, as a result of congestion or reliability issues. Multi-agent schemes that use information exchange but do not account for the associated network effects perform worse in practice than in theory. This degradation of performance can in some cases lead to instability, which makes these methods unfeasible in practical applications. In this work, we consider robustness towards the effects of networked information exchange as a design requirement.

1.2.4 Fully Actuated and Underactuated Systems

Control theory, fundamentally, comprises the computational process from the sensory inputs of a system to a set of actuators that influence the state and/or output of that system. For fully actuated systems, the placement of actuators is such that every state may directly be influenced by its control input. An example of such a system is a robotic manipulator where each joint is fit with an encoder and a motor to measure and control the joint-angle, respectively. In practice, not all systems are fully actuated. Underactuated systems do not permit control over each of their states at any point in time. A primary example is a quadrotor, which cannot instantaneously move parallel to the alignment of the rotor blades. Nevertheless, suitable control approaches may move the system to any point in space, by rotating the alignment of the system appropriately. The technical distinction between fully actuated and underactuated systems is further discussed in Chapter 2.

1.2.5 Collision Avoidance

In almost any environment and scenario, the workspace in which robots move is not empty. The environment may contain objects, other robots, walls, floors, ceilings, et cetera. These physical constraint need to be incorporated in the control approach to prevent collisions. Constrained control methods may incorporate either hard constraints, where constraint violation is prevented by design, or soft constraints, which allow violation of constraints but penalise their occurrence.

1.3 Introduction to Cooperative Passivity-Based Control

The control strategy presented in this work is part of a niche merger between two distinct fields of control theory. The following section briefly introduces these fields conceptually.

1.3.1 Cooperative Multi-Agent Control

The name multi-agent control indicates that instead of the traditional control of a single system, we collectively control a multitude of systems. Each of these individual systems is called an agent and as such we may refer to applications as single-agent or multi-agent. The second imperative keyword is “cooperative”, which indicates that the nature of the multi-agent systems is to work together. Since cooperative control implies multi-agent control, we often omit the multi-agent keyword. Computationally we distinguish central and distributed control.
methodologies. Central multi-agent control implies that multiple systems are controlled from a single controller, which has access to the information of all agents. Distributed control methods control each agent locally using communication or sensing to obtain information about other agents. This work applies distributed control due to its significant advantages for the scalability and flexibility of the approach.

In this work we focus on the subset of cooperative applications where the group of agent needs to achieve cooperative tracking, i.e., the coordinates of the agents need converge to a specified formation. The simplest case of these objectives is consensus control, where we aim for the systems to converge to the same point in the cooperative space without specifying this point beforehand. The extension of formation control allows agents to reach a specified configuration in the cooperative space at an unspecified point. In some cases, one or more agents are set to be leader such that their coordinates internally converge to a specified value. This allows control over the final agent locations when required. We formally include these objectives into the designed control approach.

1.3.2 Passivity-Based Control

Passivity-Based Control (PBC) comprises methods of control that exploit the energy-based dynamical description of agents to achieve control objectives. Compared to the traditional signals and systems viewpoint, PBC proposed a paradigm shift where instead of generating the correct input to obtain suitable output behaviour, we shape directly the energy behaviour of the system. The resulting shaped system then achieves suitable output behaviour by design. These approaches provide an intuitive measure of stability for nonlinear systems.

1.4 Previous Works

This work was initially motivated as continuation of the work in [2], in which a unified cooperative Interconnection-and-Damping Assignment Passivity-Based Control (IDA-PBC) scheme [3] was developed for fully actuated and underactuated systems. The unified scheme inherently assumes the absence of network effects. This work set out to relax this assumption. As we will see, this work does not extend IDA-PBC, but develops a different PBC control scheme based around an existing multi-agent scheme [1] and r-passivity [4]. Both of these results are summarised in Chapter 3.

Experimental results of PBC methods are rarely reported. Examples for single-agent IDA-PBC, are: pendulum on a cart [5], mobile inverted pendulum [6] and rotary pendulum system [7]. This motivates us to validate our results experimentally.

1.4.1 Network Effects

The key ingredient to overcome network effects in the adopted multi-agent scheme is the Scattering Transformation (ST), introduced for bilateral teleoperation applications in [8]. This technique transforms the networked input and output of a system into signals with a defined energy, such that the network becomes passive. Applying this tool leads to synchronisation of passive outputs (i.e., velocities) in [1]. It was shown in [4] that the coordinates may be encoded with the velocities in the outputs $r$ which, by local feedback, can be rendered passive.
1.5 Novelty of This Work

This r-passivity approach was used in [1] to show multi-agent coordinate convergence. The basic ST architecture, as presented in [8], achieves passivity in continuous-time in the presence of arbitrary constant communication delays. In [9] an extension is introduced that preserves passivity for time-varying delays in continuous-time. In [10] it is shown that varying time-delays and packet loss in discrete-time both lead to the absence of received data when the network is sampled. The algorithm developed in the same work reconstructs wave variables in a passive manner, thereby remaining passive under improved performance. The work in [11] introduces a passive reconstruction method, called Wave-Variable Modulation (WVM). WVM reconstructs wave-variables with maximal passive energy and is computed locally.

1.4.2 Collision Avoidance

Well known control methods with soft constraints are Artificial Potential Functions (APFs) [12]. These control methods follow the gradient of a virtual potential energy function. Agents may be repelled from constrained areas in the workspace via a local increase in the virtual potential function. In this work we consider a specific subset of APFs termed Navigation Functions (NFs) [13]. NFs are theoretically free of local minima, i.e., points where the repelling and attracting potential fields cancel to give zero control input. Additional improvements such as workspace normalisation [14], can make NFs both high performance and intuitive to tune. These criteria have inspired distributed applications, e.g. [15], [16].

1.5 Novelty of This Work

The base scheme for this work, introduced in [1], provides a framework for multi-agent PBC with output synchronisation. The same work additionally mentions the use of encoded outputs $r$ for coordinate synchronisation, presented in [4]. However, the latter paper achieves r-passivity only in the full state, i.e., the output is constructed from the full state and its derivatives. In realistic scenarios we do not require synchronisation in the states but rather in the end-effectors and hence this method does not apply. We show in this work that a modified output $r$ with task-space coordinates and task-space velocities can be applied in the same scheme. The existing local control law for r-passivity is not applicable due to the introduced mapping to the end-effector output. For this application we derive a control law that renders mechanical systems r-passive in task-space coordinates. We show that this results in a partial requirement on the dynamics of the controlled system. We then introduce additional control terms to influence the excess degrees-of-freedom. To show the efficacy of the approach, the scheme is validated experimentally. Our experiments include formation and consensus control with heterogeneous systems, performance validation of the control scheme in the presence of a range of network effects and comparison to cooperative IDA-PBC.

1.6 Notation

To distinguish between matrices, vectors and scalars we introduce typographical emphasis. Scalars are denoted by non-bold, lower-case variables. Vectors are denoted by bold, lower-case...
variables and matrices are denoted by bold and capitalised variables. Formally:

\[ q \in \mathbb{R}, \quad q \in \mathbb{R}^n, \quad Q \in \mathbb{R}^{n \times m}. \] (1-1)

If a scalar is part of a matrix we denote with the variable \( M_{ij} \) the element of matrix \( M \) at the \( i \)th row and the \( j \)th column.

The transpose of a matrix \( M \) is denoted by \( M^T \). We often use the Moore-Penrose Pseudoinverse, which is denoted as \( M^\dagger \). Lastly an annihilator of the matrix \( M \), if it exists, is denoted with \( M^\perp \).

We use two special matrices in this Thesis. The identity matrix \( I_n \in \mathbb{R}^{n \times n} \) denotes the matrix with \( I_{n,ij} = 1 \) if \( i = j \) and \( I_{n,ij} = 0 \) if \( i \neq j \). The zero matrix \( 0_n \in \mathbb{R}^{n \times n} \) has only zero elements. The vector \( 0 \in \mathbb{R}^n \) denotes the vector with only zero elements.

When we refer to “global stability” in a nonlinear system context, we refer to global stability of the cooperative equilibrium unless mentioned otherwise.

In this work we often leave out the agent identifier \( i \) when considering a local scenario to improve readability. Similarly we often leave out dependencies on time \( t \), unless it is contextually relevant, for example, when delays are present.

### 1.7 Layout of This Work

This main content of this work is split into two parts. In Part I we introduce the novel cooperative r-passivity-based control scheme. We first provide fundamentals of mechanical system modelling in Chapter 2. Essential results from literature on passivity theory, cooperative control and network effects are detailed in Chapter 3. In Chapter 4 we present the continuous-time r-passivity-based control scheme and we show its convergence. Moreover, we develop extensions that achieve formation control and leader-follower control. Chapter 5 derives a control law for mechanical systems that achieves r-passivity while additionally providing tools to control the local coordinates. In Chapter 6 we evaluate the continuous-time approach in simulation, concluding the first part.

In Part II we are concerned with control in discrete-time and implementation problems. Chapter 7 introduces a collision avoidance scheme. The discussion moves from a purely theoretical viewpoint to an implementation viewpoint in Chapter 8 where the design of the cooperative system in C++ and ROS is detailed.

The presented implementation is used in Chapter 9 to obtain experimental results that validate convergence of the proposed approach as well as its robustness in realistic network conditions. We conclude this work with a discussion and conclusion in Chapter 10 followed by recommendations for future work in Chapter 11.
Part I

Continuous-Time $r$-Passivity-Based Control
This section provides a brief overview of the dynamical modelling concepts used in the rest of this work. For a more detailed introduction to mechanical system modelling, the reader is referred to [17] and [2]. At the end of this chapter we formulate a mathematical problem definition with the introduced mechanical model.

2.1 Modelling of Mechanical Agents

2.1.1 System Coordinates and Velocities

The state of a nonlinear mechanical system can be described by a set of coordinates $\mathbf{q}(t) \in \mathbb{R}^n$. The derivative of these coordinates are referred to as the velocities $\dot{\mathbf{q}}(t)$. We are additionally interested in the momenta of mechanical system which we may relate to the velocities using

$$p = M(\mathbf{q}) \dot{\mathbf{q}}, \quad (2-1)$$

where $M(\mathbf{q}) > 0$, $M(\mathbf{q}) \in \mathbb{R}^{n \times n}$, denotes the mass matrix.

2.1.2 Hamiltonian Mechanical Modelling

In this work, the agent dynamics are described by Hamiltonian dynamics. The states of these systems are the coordinates and the momenta. A simple mechanical Hamiltonian model for agent $i$ is given by

$$\begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{p}}_i \end{bmatrix} = \begin{bmatrix} 0_{n_i} & I_{n_i} \\ -I_{n_i} & 0_{n_i} \end{bmatrix} \begin{bmatrix} \frac{\partial H_i}{\partial \mathbf{q}_i} \\ \frac{\partial H_i}{\partial \mathbf{p}_i} \end{bmatrix} + \begin{bmatrix} 0_{n_i \times m_i} \\ \mathbf{F}_i(\mathbf{q}_i) \end{bmatrix} \mathbf{\tau}_i,$$

$$H_i = \frac{1}{2} \mathbf{p}_i^T M_i^{-1} \mathbf{p}_i + V_i(\mathbf{q}_i), \quad (2-2)$$

where $\mathbf{q}_i, \mathbf{p}_i \in \mathbb{R}^{n_i}$ are the coordinates and momenta respectively, the Hamiltonian $H_i$ is the sum of kinetic and potential energy in the system. The inputs, denoted by $\mathbf{\tau}_i \in \mathbb{R}^{m_i}$, enter
the system via input matrix $F_i(q) \in \mathbb{R}^{n_i \times m_i}$. Systems for which the actuated dimensions $m_i$ are smaller than the coordinate dimension $n_i$ are underactuated. In contrast, if $m_i = n_i$ the system is fully actuated and as a result $F_i(q)$ is full rank. For these fully actuated systems we can set, without loss of generality, $F_i(q) = I_{n_i}$, since invertibility of $F_i$ allows cancellation via the input gain $\tau_i = F_i^{-1} \hat{\tau}_i$.

The term simple in the naming of the model refers to the specific Hamiltonian function. For non simple systems, the Hamiltonian $H_i$ can be a general, smooth function of $q, p$.

### 2.2 End-Effector Dynamics and Task-Space Control

Rather than synchronising the states $q_i$, this work aims to synchronise end-effectors of connected agents. The term end-effector refers to a functionally relevant point on an agent’s body. As an example, the end-effector of a robotic manipulator often refers to the end-point of the kinematic chain, i.e., the part of the arm that manipulates its environment. Since in this work, the cooperation of the multi-agent system is facilitated via these end-effector coordinates, we refer to them as the cooperative coordinates. The cooperative coordinates are denoted $z_i \in \mathbb{R}^l$ and, in general, are obtained from the system states via a nonlinear transformation $z_i = a_i(q_i)$. Since the cooperative coordinates are the variables to synchronise, their dimension $l$ is equal for all agents. The cooperative velocities $\dot{z}_i$, denote the derivatives of the cooperative coordinates. For these velocities we may write

$$\dot{z}_i = \frac{\partial^T z_i}{\partial q_i} \dot{q}_i = \Psi_i^T(q_i) \dot{q}_i.$$  \hspace{1cm} (2-3)

The matrix $\Psi_i^T \in \mathbb{R}^{l \times n_i}$ is the end-effector Jacobian. For systems where $n_i > l$ we additionally introduce the local coordinates $\theta_i \in \mathbb{R}^{s_i}$ with $s_i = n_i - l$. These coordinates denote the orthogonal complement of the cooperative coordinates and are by construction strictly unrelated to the end-effector position. Similar to the cooperative velocities, we derive an expression for the local velocities as

$$\dot{\theta}_i = \frac{\partial^T \theta_i}{\partial q_i} \dot{q}_i = \Psi_i^\perp(q_i) \dot{q}_i.$$  \hspace{1cm} (2-4)

Since the local and cooperative coordinates span the full dynamical space, we obtain $\Psi_i^\perp \Psi_i = 0$. Hence we may write

$$\begin{bmatrix} \dot{z}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \Psi_i^T \\ \Psi_i^\perp \end{bmatrix} \dot{q}_i.$$  \hspace{1cm} (2-5)

In the rest of this work we assume that the Jacobian $\Psi_i^T$ is of rank $l$, i.e., systems are free of singularities. For systems where singularities are present it is often possible to avoid them using obstacle avoidance techniques.
2.3 Problem Definition

In this work we consider $N$ heterogeneous mechanical agents, each agent described by (2-2). For the purpose of cooperative synchronisation we consider the end-effector coordinates $z_i \in \mathbb{R}^l$, with $l \leq m_i = n_i \forall i$, given by a nonlinear output mapping $z_i = a_i(q_i)$. These coordinates denote exactly the coordinates that should synchronise over the network. The $N$ systems communicate over a directed, bidirectional and strongly connected graph $G(\Sigma, \mathcal{E})$. The bidirectional edge pair $E_{ij}, E_{ji} \in \mathcal{E}$ is subject to possibly time-varying and distinct time-delays $T_{ij}(t), T_{ji}(t) \geq 0$. In this setting the systems are said to *synchronise* iff

$$\lim_{t \to \infty} ||z_i(t - T_{ij}(t)) - z_j(t)|| = 0, \quad \forall (i, j) \in \mathcal{E}. \quad (2-6)$$
Chapter 3

Cooperative Passivity-Based Control with Delays

This chapter serves to introduce the reader to passivity, the Scattering Transformation (ST) and derived control methods. These tools from literature provide the basis of the novel control approach introduced in the following chapters. The interested reader is encouraged to follow the provided references.

3.1 Passivity

Passivity is a system property related to energy-like behaviour, which is particularly useful for stabilisation of nonlinear systems. We briefly introduce its fundamentals.

3.1.1 Definition of Passivity

A passive system is a system for which the energy output is never greater than the energy supplied to it externally[1]. Energy injected into the system is either stored or dissipated. Strictly passive systems will always dissipate energy such that without external power supply the system will reach zero stored energy asymptotically. Formally we state passivity as follows.

**Definition 3.1[1]:** A system is said to be passive if there exists a $C^1$ storage function $V(x) > 0$, $V(0) = 0$ and a dissipation function $S \geq 0$ for which the passivity relation

$$V(x(t)) - V(x(0)) = \int_0^t y(s)^T \tau(s) ds - \int_0^t S(s) ds,$$

(3-1)

holds. By differentiating we may obtain the equivalent differential passivity relation

$$\dot{V} + S = \tau^T y.$$

(3-2)
The right-hand side of (3-2) quantifies the external energy input. The term $\dot{V}$ denotes the change in the stored energy while $S$ is the dissipated energy. Externally supplied energy is therefore stored or dissipated. Systems for which $S(x) > 0 \forall x \neq 0$ are said to be strictly passive. Systems for which $S(x) = 0$ are said to be lossless. Although originally motivated by the behaviour of physical energy, the definition of passivity applies to general inner-products and as such does not need to refer to physical energy.

3.1.2 Passivity of Hamiltonian Systems

Hamiltonian mechanical systems with the velocities as outputs are passive, a proof of which is provided here.

Theorem 3.1[2]: Simple fully actuated mechanical Hamiltonian systems are passive and lossless with respect to the input-output pair $(\tau, \dot{q})$ and the storage function $H$.

Proof. We calculate the derivative of the Hamiltonian.

$$\dot{H}(q, p) = \partial^T H(q, p) \dot{q} + \partial^T H(q, p) \dot{p}$$
$$\quad = \partial^T H(q, p) \frac{\partial H(q, p)}{\partial q} + \partial^T H(q, p) \left( -\frac{\partial H(q, p)}{\partial q} + \tau \right)$$
$$\quad = p^T M^{-1} \tau = \dot{q}^T \tau.$$

Which admits to the passivity description (3-2).

We remark that passivity is only obtained when the output of these systems are velocities. For this reason the velocities are often referred to as the passive outputs. Notice that the passivity property for these systems refers to physical energy.

3.2 The Scattering Transformation

A network without delays is trivially passive, since energy is neither stored nor dissipated. However, when delays are present in a network it is not possible to guarantee passivity. In fact the derivation of an expression for the energy stored in the network is non-trivial, since energy associated with data in the network is only determined when the data arrives and an input is generated. A well known result to overcome this problem is the Scattering Transformation (ST) ([8], [18]). The ST mimics the behaviour of a physical transmission line, which is passive. The networked control input and system output of an agent are transformed into wave-variables, which are communicated over the network instead of the system output. These wave-variables are packets of energy, in fact, in the case of physical energy the unit of these variables is $\sqrt{\text{Watt}}$.
### 3.2.1 Definition and Passivity

The ST, schematically drawn in Figure 3-1, is mathematically given by

\[
\begin{align*}
  s_{ij}^+ &= \frac{1}{\sqrt{2b}}(-\tau_{jisi} + br_{js}) ; \quad s_{ij}^- = \frac{1}{\sqrt{2b}}(-\tau_{jisi} - br_{js}), \\
  s_{ji}^+ &= \frac{1}{\sqrt{2b}}(\tau_{isi} + br_{is}) ; \quad s_{ji}^- = \frac{1}{\sqrt{2b}}(\tau_{isi} - br_{is}),
\end{align*}
\]

(3-3) (3-4)

where \( b > 0 \) is the virtual impedance of the network (see [8]). The wave-references \( r_{js}, r_{is} \) replace the neighbour outputs \( r_j, r_i \) respectively. The inputs \( \tau_{jisi}, \tau_{isi} \) denote the output generated by cooperative controller. We can conveniently rewrite these scalar ST equations as two-port multivariate equations in the wave reference and the output wave, which results in

\[
\begin{align*}
  \begin{bmatrix} r_{js} \\ s_{ij}^+ \end{bmatrix} &= \begin{bmatrix} -\sqrt{2}B^{-1} & -(B^{-1})^2 \\ -I_i & -\sqrt{2}B^{-1} \end{bmatrix} \begin{bmatrix} s_{ij}^- \\ \chi_{jisi} \end{bmatrix} = T^i s_{ij}^- \chi_{jisi}, \\
  \begin{bmatrix} r_{is} \\ s_{ji}^+ \end{bmatrix} &= \begin{bmatrix} \sqrt{2}B^{-1} & -(B^{-1})^2 \\ -I_i & \sqrt{2}B^{-1} \end{bmatrix} \begin{bmatrix} s_{ji}^- \\ \chi_{isi} \end{bmatrix} = T^i s_{ji}^- \chi_{isi},
\end{align*}
\]

(3-5) (3-6)

where \( B = B^T > 0 \) is the multidimensional wave impedance (see [19]). The propagation of these variables through the delayed network gives rise to the transport equations

\[
\begin{align*}
  s_{ij}^+(t - T_{ij}(t)) &= s_{ij}^+(t), & (3-7) \\
  s_{ji}^-(t - T_{ji}(t)) &= s_{ji}^-(t). & (3-8)
\end{align*}
\]

The stored energy in the network may now be expressed by a storage function, as is shown in the following theorem.

**Theorem 3.2**[1]: A bidirectional network edge with the ST (3-3), (3-4) applied on either side is lossless with respect to the storage function

\[
V_{ij}^{ij} = \frac{1}{2} \int_0^t (||s_{ij}^+||^2 - ||s_{j}^{+\tau}||^2 + ||s_{ji}^-||^2 - ||s_{j}^{-\tau}||^2)d\tau,
\]

(3-9)

in the presence of arbitrary constant delays.

**Proof.** Using the transport equations (3-7), (3-8) we may write

\[
\begin{align*}
  V_{ij}^{ij} &= \frac{1}{2} \int_0^t (||s_{ij}^+||^2 - ||s_{ij}^+(\tau - T_{ij})||^2 + ||s_{ji}^-||^2 - ||s_{ji}^-(\tau - T_{ji})||^2)d\tau \\
  &= \frac{1}{2} \int_{t-T_{ij}}^t ||s_{ij}^+||^2d\tau + \frac{1}{2} \int_{t-T_{ji}}^t ||s_{ji}^-||^2d\tau \geq 0.
\end{align*}
\]

Which shows positive definiteness of the storage function. The derivative of the storage function is given by

\[
\dot{V}_{ij}^{ij} = \frac{1}{2}(||s_{ij}^+||^2 - ||s_{ij}^+(t - T_{ij})||^2 + ||s_{ji}^-||^2 - ||s_{ji}^-(t - T_{ji})||^2).
\]

(3-10)

It can be verified that, using the ST equations (3-3), (3-4), this reduces to

\[
\dot{V}_{ij}^{ij} = -\tau_{jisi} r_{js} - \tau_{isi} r_{is},
\]

(3-11)

which satisfies the passivity property (3-2).
The work in [9] extends this result to *arbitrary time-varying delays* with the use of a delay dependent gain.

### 3.2.2 Impedance Matching

The tuning parameter $b$ may have significant impact on the performance of the ST. Similar to physical transmission lines, a mismatch of impedance between the network and the connected systems results in partial reflection of the transmitted power. These *wave-reflections* deteriorate performance [8]. Another formulation of this criteria, given in [19], is that there may exist no direct feedthrough from $s^-$ to $s^+$ or vice versa. Controllers connected to the network may vary their impedance based on the received networked input and hence impedance matching at all time instances is not trivially obtained. A solution to this problem is to apply a lowpass filter to on the wave-variables, since reflections are generally high frequency signals [18]. This reduces the wave-reflections at the expense of bandwidth, and therefore performance, of the network.

### 3.3 The Cooperative Passivity-Based Control Scheme

In the interest of multi-agent control we now present the PBC scheme introduced in [1], which applies the ST in a multi-agent setting to achieve synchronisation. The scheme is depicted in Figure 3-1 and is referred to as Scheme 1.

![Figure 3-1: The multi-agent scheme as in [1], referred to as Scheme 1.](image)

The scheme consists of three passive system components:

- The network with ST (lossless),
- The cooperative controls ($V_{jsi}$, $S_{jsi}$),
- The agents ($V_i$, $S_i$).

The cooperative controls and the ST are applied to each pair of connected agents as in Scheme 1. Each agent receives wave-variables from the network that are used to construct the wave reference $r_{jsi}$. Each difference $r_{jsi} - r_i$ functions as input to a cooperative control function. In the rest of this work we denote $r_{jsi} = r_{jsi} - r_i$. The cooperative control input for agent $i$ is obtained by summing its individual contributions as

$$\tau_i = \sum_{j \in N_i} \phi_{jsi}(r_{jsi}),$$

(3-12)
where $\mathcal{N}_i$ denotes the set of neighbours of agent $i$. It is shown in [1] that a Lyapunov candidate for the multi-agent system is

$$V = \sum_{i=1}^{N} V_i + \sum_{(i,j) \in \mathcal{E}} V_{ij}^{\text{channel}} + \sum_{i=1}^{N} \sum_{(i,j) \in \mathcal{E}} V_{jsi},$$

(3-13)

the derivative of which is

$$\dot{V} = -\sum_{i=1}^{N} S_i - \sum_{i=1}^{N} \sum_{(i,j) \in \mathcal{E}} S_{jsi} \leq 0.$$

(3-14)

Hence the system is globally stable and

$$S_{jsi} \to 0 \ \forall \ (i,j) \in \mathcal{E}, \ S_i \to 0 \ \forall \ i.$$

(3-15)

Synchronisation of the agent coordinates then depends on the properties of these storage functions and the selected outputs.

### 3.4 Full State r-Passivity

Scheme 1 achieves synchronisation of outputs. However synchronisation of the typical velocity outputs $\dot{q}_i$ does not imply synchronisation of the coordinates $q_i$. The latter is exactly the objective of most applications including the one discussed in this work. In simple cases where the delay is strictly constant and there is no packet loss, integration of either side of the network is sufficient to achieve synchronisation. However if either of these effects are present than there is no way to recover lost data, since the coordinates are unobservable from the output, which results in drift [4]. A common solution (seemingly developed separately in e.g. [20],[4] and [11]) is to encode the coordinates and the velocities into a single output, i.e.,

$$r_{q,i} = \dot{q}_i + \lambda q_i, \ \lambda > 0.$$  

(3-16)

The coordinates are observable in this output definition and the drift problems are ameliorated. It was shown in [4] that this output with linear inter-agent couplings and without the ST achieves synchronisation in the presence of constant delays. However application of the ST in combination with this output definition enlarges the application scope to networks where time-varying delays and packet-loss are present.

Notably, the outputs $r_{q,i}$ result in a passivity definition based on non-physical energy. These outputs are only applicable in Scheme 1 if the agents are passive with respect to this new energy, which inherently is not the case. We may, however, shape the system dynamics via local control, such that they admit to a passivity description with the outputs $r_{q,i}$. For Euler-Lagrange (EL) systems this control law is derived in [4].
3.5 Summary

Networks with delays are not guaranteed to be passive. We have introduced the ST that overcomes the issue and renders delayed networks passive. Scheme 1 contains the ST and achieves output synchronisation of multi-agent systems in the presence of arbitrary constant delays. To synchronise coordinates we can encode the coordinates with the velocities into a new output $r_{q,i}$, where the observability of the coordinates in the output increases the robustness of the scheme towards network effects. Local control is applied to achieve passivity in the outputs $r_{q,i}$. 
Chapter 4

Cooperative r-Passivity-Based Control

In this chapter we use the results from the previous chapters to derive new results. Mainly we adapt Scheme 1 to task-space synchronisation of multi-agent systems. In Section 4.1 we pose the modified outputs $r$ which encode cooperative coordinates and velocities rather than the full state. We additionally rephrase the control conditions from [1] to a potential-based cooperative control formulation. In Section 4.2, we show that Scheme 1 with the modified outputs $r$ achieve synchronisation of the cooperative coordinates in the presence of arbitrary constant delays. We then show in Section 4.3 that minor additions to the scheme allow for leader-follower or leaderless formation control.

4.1 Task-Space r-Passivity and Potential-Based Cooperative Controls

Rather than full state synchronisation as presented in the previous chapter we aim to synchronise only the cooperative subset of coordinates, $z_i$. To that end we modify the outputs $r_{q,i}$ in (3-16), to the new outputs $r_i$ given by

$$r_i = \dot{z}_i + \lambda z_i, \quad \lambda > 0.$$  \hfill (4-1)

4.1.1 Potential-Based Cooperative Controls

This work adapts a potential based approach to cooperative control, which requires rewriting of the conditions on passive cooperative controls posed in [1]. We define a continuous artificial potential function $\varphi(u) : \mathbb{R}^d \rightarrow \mathbb{R}$, with $\varphi(0) = 0$ and $\varphi(u) > 0, \forall \ u \neq 0$. Additionally we pose the following assumptions:
Assumption 4.1(i):  $\frac{\partial \varphi(u)}{\partial u}$ Lipschitz and continuous,
Assumption 4.1(ii):  $\frac{\partial \varphi(u)}{\partial u} = 0 \iff u = 0,$
Assumption 4.1(iii):  $\frac{\partial \varphi(-u)}{\partial u} = -\frac{\partial \varphi(u)}{\partial u},$
Assumption 4.1(iv):  $\frac{\partial^2 \varphi(u)}{\partial u^2} u \geq 0.$

Assuming these conditions are satisfied we may define the cooperative controls in (3-12) by

$$\phi_{jsi} = K_{ij} \frac{\partial \varphi(r_{jsi})}{\partial r_{jsi}},$$

with $K_{ij} = K_{ij}^T > 0,$ the network gain on edges $\mathcal{E}_{ij}, \mathcal{E}_{ji}$. A storage function for these controls is $V_{jsi} = 0,$ for which the differential passivity equation (3-2) yields

$$\frac{\partial T \varphi(r_{jsi})}{\partial r_{jsi}} K_{ij} r_{jsi} = S_{jsi} \geq 0,$$

which is strictly passive.

### 4.1.2 Algebraic Loop Occurrence and Solutions for Linear Couplings

The derived potential based cooperative controls are memoryless and as such directly generate a control input when the ST computes a reference from an arriving wave. This results in an algebraic loop between the ST and the cooperative controls. We note that this problem is rarely reported in literature as applications of the ST often relate to bilateral teleoperation where controllers tend to have memory. One report of the problem is given in [21] where the problem is solved by lowpass filtering the wave reference output. In the following we illustrate and resolve the algebraic loop for quadratic cooperative potential functions.

The cooperative controls and wave references at time step $k$ are given by

$$r_{js}[k] = T_{11}^{ij} s_{js}^{+}[k] + T_{12}^{ij} \chi_{jsi}[k],$$

$$\chi_{jsi}[k] = K_{ij} (r_{js}[k] - r_{i}[k]),$$

which cannot be computed due to the mutual dependency of $r_{js}[k]$ and $\chi_{jsi}[k]$. However, for the linear case, we note that substituting (4-4) in (4-3) yields an algebraic solution for the wave reference

$$r_{js}[k] = (I_i - T_{i2}^{ij} K_{ij})^{-1} (T_{11}^{ij} s_{js}^{+} - T_{12}^{ij} K_{ij} r_{i}).$$

To illustrate that this solution is allowed with regards to wave-reflections, we now derive the matched value of the impedance $b$. The following derivations are for scalar communications, but apply component wise to multivariate systems. The scalar equivalent of (4-5) is given by

$$r_{js} = \frac{T_{11}^{ij} s_{js}^{+} - T_{12}^{ij} K_{ij} r_{i}}{1 - T_{12}^{ij} K_{ij}}.$$
We now substitute this expression and the controls (4-4) in the expression for the outgoing wave $s_{ij}^+$ (3-4), which yields

$$s_{ij}^+ = T_{21}^{ij}s_{ij}^- + T_{22}^{ij}k_{ij}\left(\frac{T_{11}^{ij}s_{ij}^- - T_{12}^{ij}k_{ij}r_i}{1 - T_{12}^{ij}k_{ij}} - r_i\right).$$

(4-7)

For impedance matching we require no feedthrough from $s_{ij}^-$ to $s_{ij}^+$. The relevant dependency can be expressed as

$$s_{ij}^+ = \left(T_{21}^{ij} + T_{22}^{ij}k_{ij}\frac{T_{11}^{ij}}{1 - T_{12}^{ij}k_{ij}}\right)s_{ij}^- + (\cdot)r_i$$

(4-8)

$$s_{ij}^+ = \left(-1 + \sqrt{\frac{2}{b}k_{ij}\sqrt{\frac{2}{b}}}\right)s_{ij}^- + (\cdot)r_i.$$  

(4-9)

Hence the line impedance $b$ that eliminates feedthrough is given by

$$\frac{2k_{ij}}{b} = \frac{b + k_{ij}}{b},$$

which results in the well known solution

$$b = k_{ij}.$$  

(4-10)

Hence we always obtain impedance matching by setting the line impedance equal to the gain. We have therefore shown that the algebraic loop formed between the ST and the cooperative controls has an algebraic solution that allows for impedance matching. We note that this algebraic solution is only possible if the dependency of the controls on $r_{js}$ is linear, which does not hold for cooperative control functions in general. This problem will be addressed later in Section 7.1.2.

### 4.2 Cooperative Coordinate Synchronisation with Task-Space r-Passivity

We are now ready to show that the scheme with the modified outputs $r$ achieves synchronisation. We pose one additional assumption on the local control of the agents, namely:

**Assumption 4.2:** We require cooperative velocities to be damped locally, i.e.,

$$S_i \rightarrow 0 \implies \dot{z}_i \rightarrow 0.$$  

This assumption is not restrictive, which is further detailed in Chapter 5, where a control law is developed that satisfies this assumption.

**Theorem 4.1:** Scheme 1 with strictly r-passive agents (2-2) satisfying Assumption 4.2, outputs (4-1) and controls (3-12), (4-2) satisfying Assumptions 4.1(i) - 4.1(iv), achieve synchronisation in the sense of (2-6) in the presence of arbitrary constant time delays.
Proof. Since all components in the system are passive, the derivations in [1] ensure that

\[ S_i \to 0, \quad S_{jsi} \to 0. \]

This implies that cooperatively, we have

\[ S_{jsi} \to 0 \iff r_{jsi} \to 0, \quad \tau_{jsi} \to 0, \quad \forall (i,j) \in E, \tag{4-11} \]

while Assumption 4.2 implies \( r_i \to \lambda z_i \). Hence we may write

\[ \dot{z}_{js} = -\lambda (z_{js} - z_i), \quad \forall (i,j) \in E, \]

which is a linear system where the input \( \lambda z_i \) is constant. The coordinate transformation \( \hat{z}_{js} = z_{js} - z_i \) yields

\[ \dot{\hat{z}}_{js} = -\lambda \hat{z}_{js}, \quad \forall (i,j) \in E, \]

which is Hurwitz and hence converges exponentially to \( \hat{z}_{js} = 0 \). Reverting the transformation shows that

\[ \lim_{t \to \infty} z_{js} = \lim_{t \to \infty} z_i, \quad \lim_{t \to \infty} \dot{z}_{js} = 0, \quad \forall (i,j) \in E. \tag{4-12} \]

With local convergence established, we rewrite the scattering transformation to obtain cooperative convergence. The wave references are given by

\[ r_{js} = \frac{1}{c}( -\sqrt{2b s_{ij}} - \tau_{jsi} ), \]

\[ r_{is} = \frac{1}{c}( \sqrt{2b s_{ji}^+} - \tau_{isj} ), \quad \forall E_{ij} \in E. \]

Using (3-3), (3-4), (3-7) and (3-8) in this description results in

\[ \lim_{t \to \infty} r_{js} = \lim_{t \to \infty} \frac{1}{c}( -\tau_{isj}(t - T_{ji}) + b r_{is}(t - T_{ji}) - \tau_{jsi} ) \]

and a similar description for \( r_{is} \) for all edge pairs. Since \( \tau_{jsi} \to 0 \) and \( \tau_{isj} \to 0 \), we obtain

\[ \lim_{t \to \infty} r_{js}(t) = \lim_{t \to \infty} r_{is}(t - T_{ji}), \quad \forall (i,j) \in E. \tag{4-13} \]

Using Assumption 4.2 and (4-12) we finally obtain

\[ \lim_{t \to \infty} z_j(t - T_{ji}) - z_i = 0, \quad \forall (i,j) \in E. \tag{4-14} \]

Which together with a strongly connected network implies synchronisation of the agents. ☐

This shows that the scheme with our modified outputs synchronises. Remarkably, this synchronisation poses no requirements on the local behaviour of the system. Hence, systems where the storage function \( S_i \) does not contain \( \dot{\theta}_i \), allow trajectories where the cooperative coordinates \( z_i \) converge within the network, while the local coordinates \( \theta_i \) persistently move in the excess degrees-of-freedom of the cooperatively converged system. Since this behaviour is unpractical, we develop a control law in Chapter 5 that damps the local coordinates and additionally influences the behaviour of the local dynamics.
4.3 Leader-Follower or Leaderless Formation Control

Theorem 4.1 shows that we can achieve multi-agent consensus in the presence of time delays with modified outputs. We now enlarge the application scope with formation control, where the agents achieve a specified task-space configuration. In both consensus and formation control the relative point where the agents synchronise is unspecified. This point is determined by the agent dynamics and cooperative couplings. However for some applications we would like to influence the position of this point, either partially or entirely. For that purpose we assign leaders that, next to the cooperative process, track a specified reference. In this section we extend Scheme 1 with leader-follower and formation control.

4.3.1 Formation Control

To achieve formation control, we rephrase synchronisation condition (2-6) to include formation distances. The resulting condition is given by

$$\lim_{t \to \infty} ||z_j(t - T_{ji}(t)) - z_i(t) - z_{ji}^*|| = 0, \forall (i, j) \in \mathcal{E},$$

(4-15)

where $z_{ij}^* = -z_{ji}^*$ denotes the specified formation target between agent $i$ and $j$. We propose to modify the input of the cooperative controls to

$$u_{jsi} = r_{jsi} - \frac{\lambda}{2} z_{ij}^*, \forall (i, j) \in \mathcal{E}.$$  

(4-16)

**Theorem 4.2**: Scheme 1 with strictly r-passive agents (2-2) satisfying Assumption 4.2, outputs (4-1) and controls (3-12), (4-2) with modified inputs (4-16) and satisfying Assumptions 4.1(i) - 4.1(iv), achieve synchronisation in the sense of (4-15) in the presence of arbitrary constant time delays.

**Proof.** Equation (4-11) still holds with the modified control inputs, however $\tau_{jsi} \to 0$ implies that instead of coordinate convergence (4-12) we obtain

$$\lim_{t \to \infty} z_{js} - z_i = \frac{1}{2} z_{ij}^*, $$

(4-17)

$$\lim_{t \to \infty} z_{is} - z_j = -\frac{1}{2} z_{ij}^*, \forall E_{ij}.$$  

(4-18)

Since (4-13) holds, we may write

$$\lim_{t \to \infty} z_{js}(t) = \lim_{t \to \infty} z_{is}(t - T_{ji}) = \lim_{t \to \infty} z_j(t - T_{ji}) - \frac{1}{2} z_{ij}^*.$$  

(4-19)

Substituting this relation into (4-17) yields

$$\lim_{t \to \infty} z_j(t - T_{ji}) - z_i = z_{ji}^*, \forall (i, j) \in \mathcal{E}.$$  

(4-20)

\[
\square
\]

Instead of spacing the wave-references on either side of the network by half the formation distance $z_{ij}^*$, we could shift more of the spacing to either side. This possibility is noted but not further investigated in this work.
4.3.2 Leaders

We can assign leaders using a cooperative potential function that satisfies Assumptions 4.1(i) - 4.1(iv), given by

\[ \phi_i^* = K_i^* \frac{\partial \phi_i^*(z_i^* - z_i)}{\partial (z_i^* - z_i)}, \quad \forall i. \]  

(4-21)

The leader coordinates are selected with nonzero rows in matrix \( K_i^* \). The cooperative control input is modified to include leaders, which results in

\[ \tau_i = \phi_i^*(z_i^* - z_i) + \sum_{j \in N_i} \phi_{jsi}(r_{jsi}). \]  

(4-22)

**Theorem 4.3:** Scheme 3-1 with strictly r-passive agents (2-2) satisfying Assumption 4.2, outputs (4-1) and controls (4-22), (4-2), (4-21) and satisfying Assumptions 4.1(i) - 4.1(iv), achieve synchronisation in the sense of (2-6) in the presence of arbitrary constant time delays while additionally \( z_i \rightarrow z_i^* \quad \forall i. \)

**Proof.** Convergence of the dissipation function \( S_{jsi} \) to zero directly implies that \( \lim_{t \rightarrow \infty} z_i = z_i^* \), while synchronisation of the network is given by Theorem 4.1.

4.4 Summary

In this chapter a modified output definition \( r \) based on task-space coordinates was introduced. It was shown that agents coupled with these outputs as in Scheme 1, synchronise their end-effectors asymptotically. The scheme was further extended to leader-follower and formation control.
Chapter 5

Local Feedback Control for r-Passivity

In the previous chapter a set of couplings was developed that achieve synchronisation with the assumption of r-passive agents. In this chapter we show that we can achieve r-passivity for fully mechanical agents via a suitable feedback control law. We additionally show that the convergence speed can be influenced per system by assigning virtual masses to each agent. Next to cooperative strict r-passivity, this novel control law achieves objectives in the local coordinates such as joint-limit avoidance and local tracking, which are essential in practical applications. Both the local and the cooperative coordinates are damped to ensure that velocities of the agents converge to zero asymptotically.

Section 5.1 derives the base control law from the cooperative requirement of r-passivity, which is extended to include the shaping of the cooperative mass matrix. In Section 5.2, local damping and control is added to form the full control law. Section 5.3 briefly shows that the results apply to the Euler-Lagrange modelling framework as well. Section 5.4 ties the results of this chapter and Chapter 4 together to show synchronisation of the proposed cooperative PBC scheme. The results of this chapter are illustrated with an example in Section 5.6.

5.1 Local Control for r-Passivity

The following theorem derives a control law that achieves r-passivity for fully actuated agents. The key insight of this derivation is the requirement of r-passivity that directly yields the preferred cooperative dynamics of the system. The assumption of full actuation ensures that we can achieve those dynamics via control. For readability we drop the agent specific subscript \( i \) in the following derivations. Before introducing the theorem we show that we can rewrite derivative of the outputs \( \dot{r} \) in terms of the velocities \( \dot{q} \) and the derivatives of the momenta \( \dot{p} \).

We first introduce the following lemma that simplifies the control law.

**Lemma 5.1:** For any invertible and square matrix \( A \in \mathbb{R}^{n \times n} \), the identity

\[
A^{-1} \dot{A} = -A^{-1} \dot{A},
\]

holds.
Proof. For invertible matrices we have

\[ A^{-1}A = I_n. \]

Hence, using the product rule, the derivative is given by

\[ \dot{A}^{-1}A + A^{-1}\dot{A} = 0_n. \]

Rearranging yields the proposed identity.

The outputs \( \mathbf{r} \), repeated here for convenience, are given by

\[ \mathbf{r} = \dot{z} + \lambda z. \]

The derivative of these outputs are

\[
\dot{\mathbf{r}} = \dot{\mathbf{z}} + \lambda \mathbf{z} \\
= \Psi^T \dot{\mathbf{q}} + \Psi^T \dot{\mathbf{q}} + \lambda \Psi^T \dot{\mathbf{q}} \\
= \Psi^T (M^{-1} \mathbf{p} + M^{-1} \dot{\mathbf{p}}) + \lambda \Psi^T \dot{\mathbf{q}} + \dot{\Psi}^T \dot{\mathbf{q}} \\
= \Psi^T M^{-1} \dot{\mathbf{p}} + \Psi^T (\lambda I_n - M^{-1} \dot{M}) \dot{\mathbf{q}} + \dot{\Psi}^T \dot{\mathbf{q}}. \tag{5-2}
\]

Where we used the definition of the cooperative coordinates (2-3), the definition of momenta (2-1) as well as Lemma 5.1.

**Theorem 5.1:** For Hamiltonian agents (2-2), where \( n = m \), the control law

\[
\mathbf{r} = M(\mathbf{\Psi}^\dagger)^T (\mathbf{r}_c - \mathbf{K}_z \dot{\mathbf{q}}) + \frac{\partial H}{\partial \mathbf{q}}, \tag{5-3}
\]

\[
\mathbf{K}_z = \Psi^T ((\lambda + \gamma)I_n - M^{-1} \dot{M}) + \dot{\Psi}^T, \tag{5-4}
\]

with \( \gamma > 0 \) a tuning parameter, result in strict passivity with respect to the input-output pair \((\mathbf{r}_c, \mathbf{r})\) and the storage and dissipation function

\[
V = \frac{1}{2} \mathbf{r}^T \mathbf{r} + \frac{1}{2} \gamma \lambda \mathbf{z}^T \mathbf{z}, \quad S = \gamma \dot{\mathbf{z}}^T \dot{\mathbf{z}}.
\]

Proof. The following proof is constructive. The derivative of the storage function is

\[
\dot{V} = \dot{\mathbf{r}}^T \mathbf{r} + \gamma \lambda \dot{\mathbf{z}}^T \mathbf{z} \\
= \mathbf{r}^T \dot{\mathbf{r}} + \gamma \dot{\mathbf{z}}^T (\mathbf{r} - \dot{\mathbf{z}}) \\
= (\dot{\mathbf{r}} + \gamma \dot{\mathbf{z}})^T \mathbf{r} - \gamma \dot{\mathbf{z}}^T \dot{\mathbf{z}}.
\]

Substituting in differential passivity relation (3-2) yields

\[
\mathbf{r}_c^T \mathbf{r} = \dot{V} + S \\
\mathbf{r}_c^T \mathbf{r} = (\dot{\mathbf{r}} + \gamma \dot{\mathbf{z}})^T \mathbf{r} - \gamma \dot{\mathbf{z}}^T \dot{\mathbf{z}} + \gamma \dot{\mathbf{z}}^T \dot{\mathbf{z}} \\
\mathbf{r}_c^T \mathbf{r} = (\dot{\mathbf{r}} + \gamma \dot{\mathbf{z}})^T \mathbf{r}.
\]
5.1 Local Control for r-Passivity

Hence the $l$-dimensional dynamics that achieve r-passivity are given by

$$\tau_c = \dot{r} + \gamma \dot{z}.$$  

Using the expanded version of the output derivatives in (5-2), we may rewrite the dynamics as

$$\tau_c = \Psi^T (M^{-1} p + M^{-1} \dot{p}) + (\lambda + \gamma) \Psi^T \dot{q} + \dot{\Psi}^T \dot{q},$$

$$= \Psi^T M^{-1} \dot{p} + K_z \dot{q}, \quad (5-5)$$

Now define the cooperative feedback law $\tau_c = \hat{\tau}_c + K_z \dot{q}$ such that (5-5) reduces to

$$\hat{\tau}_c = \Psi^T M^{-1} \dot{p}. \quad (5-6)$$

These dynamical requirements pose restrictions on the cooperative, $l$-dimensional subset of the $n$-dimensional system dynamics. We transform from the cooperative dynamics to the full dynamics via the transformation matrix $M(\Psi^1)^T \in \mathbb{R}^{n \times l}$. To obtain passivity, we match these dynamics with the plant momenta equations, i.e.,

$$\dot{p} = M(\Psi^1)^T \dot{\tau}_c = -\frac{\partial H}{\partial q} + \tau. \quad (5-7)$$

The proposed control laws satisfy this equation.

We emphasise that the $n$-dimensional dynamics that achieve cooperative r-passivity in $l$ dimensions is non-unique for systems where $l < n$. We will show that the excess dynamical degrees of freedom can be controlled separately, but first we show that the cooperative transient response of the agents may be tuned via shaping of the cooperative mass matrix.

5.1.1 Scaled Dynamics

The storage function in Theorem 5.1 is given by $V = \frac{1}{2} r^T I_l r$, which is a measure of the stored energy. Discounting the encoded coordinates, this expression is the familiar description of mechanical kinetic energy (as presented for Hamiltonian systems in (2-2)). Hence to influence the transient behaviour of the system we propose to modify the storage function to incorporate a virtual cooperative mass matrix $M_z$. The following theorem shows that this is possible.

**Theorem 5.2**: Theorem 5.1 with the modified cooperative control law

$$\hat{\tau}_c = M_z \tau_c - \hat{K}_z \dot{q} + \frac{1}{2} M_z M_z^{-1} \dot{r}, \quad (5-8)$$

$$\hat{K}_z = \Psi^T (\lambda I_n - M^{-1} \dot{M}) + \dot{\Psi}^T + \gamma M_z \Psi^T. \quad (5-9)$$

results in r-passivity with respect to the storage function

$$V = \frac{1}{2} r^T M_z^{-1} r + \frac{1}{2} \lambda \gamma z^T z. \quad (5-10)$$

while asymptotic properties remain unchanged.
Proof. The storage function derivative is
\[ \dot{V} = r^T M_z^{-1} \dot{r} + \frac{1}{2} r^T M_z^{-1} r + \gamma z^T (r - \dot{z}). \] (5-11)

Similarly to Theorem 5.1 we derive an input description using the differential passivity equation, which results in
\[ \tau_c = M_z^{-1} \dot{r} + \frac{1}{2} M_z^{-1} r + \gamma \dot{z}. \]

Using the description of the output derivatives, we write
\[ M_z \tau_c = \Psi^T M^{-1} \dot{p} + \Psi^T (\lambda I_n - M^{-1} \dot{M}) \dot{q} + \Psi^T \dot{q} + \gamma M_z \dot{z} + \frac{1}{2} M_z M_z^{-1} r \]
\[ M_z \tau_c = \hat{\tau}_c + \hat{K}_z \dot{q} - \frac{1}{2} M_z M_z^{-1} r. \]

The proposed cooperative control law reduces the cooperative dynamics to those given in (5-6). The rest of the proof follows that of Theorem 5.1.

A simple, but useful application of Theorem 5.2 is to set the virtual cooperative mass matrix as
\[ M_z^{-1} = \begin{bmatrix} \eta_1 & 0 & \cdots & 0 \\ 0 & \eta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \eta_l \end{bmatrix}, \quad M_z = \begin{bmatrix} \frac{1}{\eta_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\eta_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\eta_l} \end{bmatrix}, \] (5-12)

Since \( \dot{M}_z = 0 \), the effect on the control law is a gain of \( \frac{1}{\eta_j} \) on element \( j \) of the cooperative input \( \tau_c \).

To illustrate a practical advantage of this approach we consider a scenario with two manipulators. The joints of one of the manipulators suffers from kinetic friction on the cooperative coordinates. Since friction is not modelled, shaping both systems with cooperative identity masses will result in more cooperative motion from the frictionless system, which is often impractical. A rigorous solution is to accurately identify and model the friction in the system and translate that to the cooperative dynamics. However, it is much simpler to set the cooperative mass matrix of the system with friction as \( M_z^{-1} = \eta I_l \), with \( 0 < \eta < 1 \). The result is a gain of \( \frac{1}{\eta} \) on the cooperative input. We can now tune that system until its transient response is satisfactory.

5.1.2 Remark on the Cooperative Dynamics

To recap on the derived results, we emphasise that our control approach directly shapes cooperative dynamics such that r-passivity is achieved. Even if connected systems are heterogeneous, the proposed method shapes the cooperative dynamics to be homogeneous, or suitably heterogeneous. The only limit to the achievable dynamics are the allowable control inputs of the system. Hence the presented approach scales easily and is flexible with respect to tuning requirements and heterogeneous systems. Once the behaviour of a system is satisfactory in the cooperative space then it may be interconnected with other systems following the same protocol, which fits precisely with the objective of this work.
5.2 Controlling the Excess Degrees of Freedom

For systems where \( n > l \), the method derived above is not the limit to the possibilities of control. In fact there are \( s = n - l \) degrees of freedom that are not influenced. In the following we use these degrees to achieve locally relevant objectives, e.g. joint-limit avoidance or local tracking, while asymptotically damping the velocities. We introduce a potential \( V_s(\theta) \) that strictly depends on the local coordinates and satisfies the following assumption.

**Assumption 5.1:** \( V_s(\theta) \) is convex, i.e.,

\[
\frac{\partial V_s(\theta)}{\partial \theta} = 0 \iff \theta = \theta^*, \quad (5-13)
\]

where \( \theta^* \) is the global minimum of the local potential function.

Our key insight is that we may isolate the local dynamics using a transformation that annihilates the cooperative dynamics. This transformation is performed by the matrix \( \Psi \perp M^{-1} \), which maps the \( n \)-dimensional desired dynamics (5-7) to the \( s \)-dimensional dynamics

\[
\dot{p} = \Psi \perp M^{-1} M(\Psi \perp)^T \dot{\hat{\tau}}_c - M(\Psi \perp)^\dagger \tau_l,
\]

\[
\dot{p} = 0.
\]

Noting that \( \Psi \perp M^{-1} \in \mathbb{R}^{s \times n} \), there exists a right-pseudo inverse which we can exploit to influence the strictly local dynamics. The resulting input matrix is given by \( M(\Psi \perp)^\dagger \). The following theorem shows that extensions to the control law achieve \( r \)-passivity while converging to the minimum of the local potential function.

**Theorem 5.3:** Under Assumption 5.1, a Hamiltonian mechanical agent (2-2), with \( n = m \) and the control input

\[
\tau = \frac{\partial H}{\partial q} + M(\Psi \perp)^T \dot{\hat{\tau}}_c - M(\Psi \perp)^\dagger \tau_l,
\]

\[
\dot{\hat{\tau}}_c = \tau_c - K_z \dot{q}^z,
\]

\[
\tau_l = K_v \dot{q}^z + \frac{\partial V_s}{\partial \theta},
\]

\[
K_z = \Psi^T((\lambda + \gamma)I_n - M^{-1}M) + \dot{\Psi}^T,
\]

\[
K_v = \Psi \perp (\kappa I_n - M^{-1}\dot{M}) + \dot{\Psi} \perp,
\]

with \( \kappa > 0 \), achieves \( r \)-passivity with respect to the input-output pair \( (\tau_c, r) \) and the storage- and dissipation functions

\[
V = \frac{1}{2} r^T r + \frac{1}{2} \gamma \dot{z}^T \dot{z}, \quad S = \gamma \dot{z}^T \dot{z},
\]

while in the absence of cooperative inputs, \( \dot{q} \to 0 \) and \( \theta \to \theta^* \).

**Proof.** By (2-2) and (5-14), the full dynamics of the controlled system are

\[
\dot{p} = M(\Psi \perp)^T \tau_c - M(\Psi \perp)^\dagger \tau_l.
\]
We now transform these dynamics to the strictly local dynamics using the transformation \( \Psi \perp M^{-1} \). The resulting dynamics are
\[
\Psi \perp M^{-1} \dot{p} = -K_v \dot{q} - \frac{\partial V_s}{\partial \theta}
\]
\[
\Psi \perp (\dot{\dot{q}} + M^{-1} \dot{M} \dot{q}) = -K_v \dot{q} - \frac{\partial V_s}{\partial \theta}
\]
Substituting the proposed damping matrix \( K_v \), it can be verified that the local dynamics reduce to
\[
\ddot{\theta} + \kappa \dot{\theta} + \frac{\partial V_s}{\partial \theta} = 0. \tag{5-21}
\]
Similarly for the cooperative dynamics with zero cooperative input, noting that by design the local terms cancel, we have
\[
\ddot{z} + (\lambda + \gamma) \dot{z} = 0.
\]
Consider the Lyapunov candidate
\[
V = \frac{1}{2} \dot{z}^T \dot{z} + \frac{1}{2} \dot{\theta}^T \dot{\theta} + V_s(\theta). \tag{5-22}
\]
The derivative of which is
\[
\dot{V} = \dot{z}^T \ddot{z} + \dot{\theta}^T \dot{\theta} + \frac{\partial V_s}{\partial \theta} \dot{\theta}
\]
\[
= \dot{z}^T \left( - (\lambda + \gamma) \dot{z} \right) + \dot{\theta}^T \left( - \kappa \dot{\theta} - \frac{\partial V_s}{\partial \theta} \right) + \frac{\partial V_s}{\partial \theta} \dot{\theta}
\]
\[
= - (\lambda + \gamma) \dot{z}^T \dot{z} - \kappa \dot{\theta}^T \dot{\theta} \leq 0.
\]
Hence the system is globally stable. The system converges to the set \( \mathcal{M} = \{ q, \dot{q} \mid \dot{V} \equiv 0 \} \), where we have \( \dot{q} \to 0 \) and therefore \( \dot{\theta} \to 0 \), \( \theta \to 0 \). By LaSalle’s invariance principle, the system converges to the largest invariant set in \( \mathcal{M} \), which is characterised by substituting asymptotic signal properties in (5-21), resulting in
\[
\frac{\partial V_s}{\partial \theta} = 0. \tag{5-23}
\]
Assumption 5.1 then implies that \( \theta \to \theta^* \).

\section*{5.3 Extension to the Euler-Lagrange Framework}

This work has, up until this point, only considered Hamiltonian dynamical models as described in Chapter 2. There is however a second, often used mechanical modelling framework called the Euler-Lagrange (EL) framework. For technical details we refer to \[17\]. We briefly show now that systems in the EL framework may be controlled with the proposed control law.

\textbf{Theorem 5.4:} Theorem 5.3 applies to systems formulated in the Euler-Lagrange framework by substituting
\[
\frac{\partial H}{\partial q} = -C^T \dot{q} + \frac{\partial V}{\partial q}. \tag{5-24}
\]
5.4 Convergence of the Full Scheme

Proof. From the definition of the Coriolis matrix (see [17, Chapter 6]), we have $\dot{M} = C + C^T$. Substituting in the EL-dynamic equations yields

$$M \ddot{q} + C \dot{q} + \frac{\partial V}{\partial q} = \tau$$

$$\dot{p} - \dot{M} \dot{q} + C \dot{q} + \frac{\partial V}{\partial q} = \tau$$

$$\dot{p} - C^T \dot{q} + \frac{\partial V}{\partial q} = \tau.$$

The proposed substitution yields the Hamiltonian momenta equations (2-2), after which the proof follows the proof of Theorem 5.3.

5.4 Convergence of the Full Scheme

Chapter 4 and Chapter 5 have introduced all the necessary results to achieve cooperative coordinate synchronisation and local convergence of all agents in Scheme 1. We formally construct this proof in the following theorem.

**Theorem 5.5:** Consider Scheme 1 with strictly r-passive agents (2-2) for which $n_i = m_i$, $\forall i$, which apply the local feedback law formed by (5-14) - (5-18). Let the outputs be given by (4-1) and the cooperative controls by (3-12), (4-2) which satisfy Assumptions 4.1(i) - 4.1(iv). Then the scheme achieves synchronisation in the sense of (2-6) in the presence of arbitrary constant time delays, while additionally $\dot{q}_i \to 0$ and $\theta_i \to \theta^*_i$, $\forall i$.

Proof. Theorem 5.3 is an extension of Theorem 5.1. Without any assumptions Theorem 5.1 achieves passivity and therefore provides, for each agent, a storage function $V_i$ and a dissipation function $S_i(\dot{z}_i)$ that satisfies Assumption 4.2. Hence, satisfying all its requirements, we can apply Theorem 4.1 to establish synchronisation in the sense of (2-6) emphasising that we additionally obtain $\tau_{c,i} = 0$ $\forall i$. Applying this last result to Theorem 5.3, we establish $\dot{q}_i \to 0$ and $\theta_i \to \theta^*_i$, $\forall i$, which concludes the proof.

This proof concludes the theoretical derivations of the cooperative r-passivity-based control scheme, that for the rest of this work will be referred to as rPBC.

5.5 Summary

This chapter derived a feedback control law that achieves strict r-passivity for fully actuated Hamiltonian mechanical agents. Additions to cooperative transient behaviour and local coordinate dynamics were introduced. The presented theoretical results were derived for continuous-time systems and for arbitrary but constant delays in the communication network. Before we proceed to analyse the discrete-time aspects of the control scheme we show continuous-time simulation results in Chapter 6. A simple example of the derived scheme is presented first.
5.6 Example: Fully Cooperative Control of Two Fully Actuated Point-Masses

We now treat the simplest application of rPBC. Consider two force actuated point-masses in the $x,y$ plane where we aim to achieve full state convergence, i.e.,

$$q_i = z_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \quad (5-25)$$

In the following let $i = 1, 2$. We can describe the dynamics of the systems with the matrices

$$M_i = I_2, \quad \Psi_i = I_2, \quad H_i = \frac{1}{2} q_i^T \dot{q}_i. \quad (5-26)$$

Noting the absence of coordinate dependencies, we find that $\dot{M}_i = \dot{\Psi}_i = 0$, which allows us to compute the example here on paper. Specifically we note that there are no local coordinates since $l = n_i$. Moreover, the gradient $\frac{\partial H_i}{\partial q_i} = 0$, as the Hamiltonian does not depend on $q_i$.

For simplicity we set $\lambda = \gamma = 1$. It then directly follows that the control law (5-14) - (5-18) reduces to

$$\tau_i = \tau_{c,i} - 2\dot{q}_i. \quad (5-27)$$

To couple agent states we introduce linear couplings between the agents, resulting from quadratic potential functions. We set $K_{ij} = K_{ji} = I_l$. From (4-10) we obtain impedance matching by setting component-wise $b = k_{ij} = 1$, or in the multivariate case $B = I_l^2 = I_l$.

It can be verified from (3-5) and (3-6) that the wave reference, cooperative input and the reflecting wave are given by

$$r_{js} = -\frac{1}{\sqrt{2}} s_{ij} - \frac{1}{2} r_s, \quad (5-28)$$

$$\tau_{jsi} = -\frac{1}{\sqrt{2}} s_{ij} + \frac{1}{2} \tau_s, \quad (5-29)$$

$$s_{ij}^+ = \frac{1}{\sqrt{2}} \tau_s, \quad (5-30)$$

Noting the linearity in these equations, we may efficiently compute the output and reflecting wave as

$$\begin{bmatrix} \tau_{jsi} \\ s_{ij}^+ \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} I_l & -\frac{1}{2} I_l \\ 0_l & \frac{1}{\sqrt{2}} I_l \end{bmatrix} \begin{bmatrix} s_{ij}^- \\ r_i \end{bmatrix}. \quad (5-31)$$

Since there are only two agents and no leaders $\tau_{jsi} = \tau_{c,i}$. To conclude this example we can fully specify the control law as

$$\tau_i = -\frac{1}{\sqrt{2}} s_{ij}^- - \frac{1}{2} q_i - \frac{5}{2} \dot{q}_i, \quad (5-32)$$

$$s_{ij}^+ = \frac{1}{\sqrt{2}} (\dot{q}_i + q_i). \quad (5-33)$$
Chapter 6

Simulation Results

In this section we validate the rPBC scheme proposed in Chapter 4 and Chapter 5 via simulation. We first outline an existing modelling technique for the simulated N-link manipulators as reference for the reader.

6.1 Modelling of a Fully Actuated N-Link Manipulator

Modelling of an N-link flexible manipulator was discussed in [2, Chapter 12], from which we can derive a fully actuated N-link manipulator model. For completeness we perform these calculations here, emphasising that the original work was done in [2]. Figure 6-1 shows the model setup of a 3-link manipulator. We remark that the centre of mass of each link is

![Figure 6-1: Schematic of a 3-link manipulator as modelled in this section.](image)
assumed to be in the middle of the link. The coordinates $q$ are the joint angles. We denote by $\gamma_k$ the absolute angle, i.e.,

$$\gamma_k = q_1 + \ldots + q_k. \tag{6-1}$$

For control we are required to compute the gravitational vector $\frac{\partial V(q)}{\partial q}$, the mass matrix $M(q)$ and the end-effector position $z(q)$. We first compute the centre of gravity positions of each link via

$$x_1 = x_0 + \frac{1}{2} l_1 \cos (\gamma_1),$$
$$y_1 = y_0 + \frac{1}{2} l_1 \sin (\gamma_1),$$
$$x_2 = x_1 + \frac{1}{2} l_2 \cos (\gamma_2),$$
$$y_2 = y_1 + \frac{1}{2} l_2 \sin (\gamma_2),$$

$$\vdots$$

$$x_j = x_{j-1} + \frac{1}{2} l_j \cos (\gamma_j),$$
$$y_j = y_{j-1} + \frac{1}{2} l_j \sin (\gamma_j).$$

From these positions we can compute the potential energy and its gradient using

$$V(q) = g \sum_{k} N y_k m_k, \quad \frac{\partial V(q)}{\partial q} = g \sum_{k} m_k \frac{\partial y_k}{\partial q_k}. \tag{6-2}$$

The end-effector position, located at the end of the kinematic chain is given, noting the recursive dependency on the previous link, by

$$z(q) = \left[ x_j + \frac{1}{2} l_j \cos (\gamma_j) \right. \left. y_j + \frac{1}{2} l_j \sin (\gamma_j) \right]. \tag{6-3}$$

The mass matrix can be derived from the kinetic energy description. First we collect all coordinates that have energy associated with them, i.e.,

$$w = \begin{bmatrix} x_1 & y_1 & \gamma_1 & \ldots & x_j & y_j & \gamma_j \end{bmatrix}. \tag{6-4}$$

This description, combined with the mass $m_k$ and inertia $I_k$ of each link, permits us to express the kinetic energy as

$$T(q, \dot{q}) = \frac{1}{2} \dot{w}^T \text{diag}(m_1, m_1, I_1, \ldots, m_j, m_j, I_j) \dot{w}. \tag{6-5}$$

Noting that $w = w(q)$, we may write

$$\dot{w} = \frac{\partial^T w}{\partial q} \dot{q}. \tag{6-6}$$
which allows us to rewrite (6-5) as

$$T(q, \dot{q}) = \frac{1}{2} \dot{q}^T \frac{\partial w}{\partial q} M_w(q) \frac{\partial^T w}{\partial q} \dot{q}. \quad (6-7)$$

Hence the mass matrix is given by

$$M(q) = \frac{\partial^T w}{\partial q} M_w(q) \frac{\partial^T w}{\partial q}. \quad (6-8)$$

Other control matrices such as $\Psi$ and $\dot{M}$ may now be computed from the derived matrices. The control law does not simplify as was the case with point-masses and is therefore directly given by (5-14)-(5-18).

### 6.2 Simulation Results

In the following, we place two fully actuated 3-link manipulators, in a 2D environment. We then apply the developed continuous-time control law with the parameters summarised in Table 6-1.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\gamma$</th>
<th>$\kappa$</th>
<th>$K_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>3.0</td>
<td>1.0</td>
<td>$3.0 I_n$</td>
</tr>
</tbody>
</table>

To test the results with respect to delays, we introduce the constant delays $T_{ij} = 200$ ms and $T_{ji} = 400$ ms. We need to remark that the proposed method is a pseudo-inverse method and therefore cannot move through kinematic singularities. To evade these points we use the projection proposed in [22]. The cooperative inputs are projected such that neither system can come close to their singularities. Theoretical implications of this method are discussed in Section 11.2.2.

#### 6.2.1 Consensus Control

The objective of the first simulation is to synchronise the end-effectors. Figure 6-2a and Figure 6-2b show the resulting trajectories and cooperative coordinate evolution respectively. The figures show that synchronisation of the end-effectors is obtained. Remarkably, the cooperative trajectories follow straight lines while the underlying system dynamics are nonlinear. Without proper handling of the delays we expect the trajectories to overshoot, since the delay causes the inputs to lag. As such we expect to see spiralling behaviour. In the presented figures we remark the absence of overshoot which indicates robustness towards delays.
6.2.2 Formation Control

We now present a second simulation with three 3-link manipulators to validate the proposed formation control. Instead of consensus we seek to form an equilateral triangle in 2D space with side lengths of 0.3 m. We fully connect the graph and introduce constant delays in the range between 200 ms and 400 ms on each edge. Figure 6-2a and Figure 6-2b show the resulting trajectories and cooperative coordinate evolution respectively. We observe that the specified formation is achieved with well behaved trajectories.
Part II

Implementation of Cooperative r-Passivity-Based Control
We consider in this chapter the discrete-time aspects of the proposed cooperative multi-agent scheme, which comprises the development of a collision avoidance scheme with a real-time solution and obtaining robustness towards time-varying delays and packet loss.

In practice most mechanical tasks are subjected to constraints. Primary examples are collision avoidance with other agents or obstacles in the workspace, bounds on the workspace size or connectivity range. A well known strategy to introduce these constraints as soft constraints is Artificial Potential Functions (APF) based control. An APF is a combination of attracting and repelling potential functions, the gradient of which is followed by the agents. We show in Section 7.1 that a particular type of APF, termed Navigation Functions (NFs) [13] are suitable for application in the proposed rPBC scheme. We then present two real-time solutions to overcome the algebraic loop between the ST and the cooperative controls for this particular case.

Adaptation of the rPBC scheme to discrete-time is considered in Section 7.2. In the same section we consider reconstruction of wave-variables to overcome the effects of time-varying delays and packet loss.

### 7.1 Navigation Functions and Scattering

#### 7.1.1 Introduction to Artificial Potential Functions and Navigation Functions

In [12], APF based control is proposed as a collision avoidance scheme. A classical APF consists of the sum of an attracting and multiple repelling potentials, i.e.,

\[
V(z) = \gamma(z) + \sum_{j=1}^{N_{\text{obstacles}}} \beta_j(z),
\]

(7-1)

Master of Science Thesis

O. de Groot
where $\gamma$ denotes the attracting potential and $\beta_j$ is the $j^{th}$ constraint function. Gradient descent of this potential results in the control input. We remark that the introduced cooperative control function (4-2) is an APF without repelling potentials. Adding these potentials is considered in the following.

A downside of the classical APF is that convexity of the potential function with repelling potentials is not guaranteed. Application of classical APFs to cooperative control is therefore non-robust as, especially in unknown environments, convergence cannot be guaranteed. Since robustness and scalability are essential requirements of this work, we are motivated to apply a more involved potential based control scheme.

A particular type of APF that can be made free of local minima is the Navigation Function (NF) introduced in [13]. A NF is an APF that is described by

$$
\varphi(\gamma, \beta) = \frac{\gamma}{(\gamma^\alpha + \beta)^{\frac{1}{\alpha}}}, \quad \nabla \varphi(\gamma, \beta) = \frac{\alpha \beta \nabla \gamma - \gamma \nabla \beta}{\alpha(\gamma^\alpha + \beta)^{\frac{1}{\alpha} + 1}}.
$$

(7-2)

in which $0 \leq \gamma \leq 1$ is the attracting goal potential where $\gamma = 0$ if the goal is achieved. The constraint function $\beta$ is a product of individual constraints, i.e.,

$$
\beta = \prod_{j=1}^{N_{\text{obstacles}}} \beta_j, \quad 0 \leq \beta_j \leq 1,
$$

(7-3)

where $\beta_j = 0$ indicates constraint violation. $\alpha \gg 1$ is the exponential parameter. It was shown in [14] that there exists a lower bound for $\alpha$ in any environment, for which the NF is free of local minima. We remark that next to this property, both the attractive and repelling potentials are normalised between 0 and 1. This gives NFs the added benefit of a bounded gradient.

The choice of $\gamma$ and $\beta$ depends on the environment. The design of these functions is outside the scope of this work, however we would like highlight the existing potential functions presented in [14]:

- A normalised goal function for distributed applications, to achieve high performance control in arbitrary environment sizes,
- A constraint function for workspace bounding,
- A constraint function for obstacle avoidance,
- A constraint function for inter-agent collision avoidance,
- A constraint function for connectivity maintenance.

As part of this work, the first three functions are implemented experimentally, for more details we refer to Section 8.5.1.

### 7.1.2 Real-Time Solution of Algebraic Loops for Navigation Functions with the Scattering Transformation

In Section 4.1.2 it was shown that the Scattering Transformation in conjunction with memoryless cooperative controls result in an algebraic loop. In that section we algebraically solved
for the static point in the case of linear inter-agent couplings. NF induced inter-agent couplings, however, are not linear. The proposed algebraic solution is therefore not applicable to NFs. In the following we present two real-time methods that find the static point of the algebraic loop.

**LUT-based Solution** In the first method we assume that both $\gamma$ and $\beta$ are strictly functions of the difference vector $r_{jisi}$, i.e., they do not contain constraints such as obstacle or singularity avoidance that depend only on $r_i$. In this case, substituting control law (4-2) in the expression for $r_{jsi}$, (4-3), and rearranging yields the expression

$$r_{jsi}^*[k] - T_{12}K_d \frac{\partial \varphi}{\partial r_{jsi}^*[k]} = -T_{11}s_{ij}[k] + r_i[k]. \tag{7-4}$$

Equation (7-4) takes the form $f(r_{jsi}^*) = u$ with $u$ given at runtime and $f(\cdot)$ known a priori. By constructing a Look-Up Table (LUT) of $f^{-1}(u)$ the static point can be retrieved at runtime from

$$r_{jsi}^*[k] = f^{-1}(u) + r_i[k]. \tag{7-5}$$

**Iterative Solution** To allow constraint functions $\beta$ with dependencies on both $r_{jisi}$ and $r_i$, consider the following alternative iterative method. We note that convergence of this method with constraints is not guaranteed. We present this method as a convenient real-time solution that works in most cases. The gradient of $\gamma$ may be written as

$$\nabla \gamma = c_\gamma(d)(r_i - r_{jsi}). \tag{7-6}$$

The gradient of each $\beta_j$ is given by one of the two expressions below, depending on its dependency.

$$\nabla \beta_j(r_{jisi}) = c_{\beta,j}(d)(r_i - r_{jsi}),$$

$$\nabla \beta_j(r_i) = c_{\beta,j}(r_i, r_0)(r_i - r_0).$$

We propose to freeze the weights $c_\gamma, c_{\beta,j}$ at the start of each iteration, such that the gradient becomes linear in $r_{jsi}$ and $r_i$, i.e.,

$$\nabla \varphi(r_i, r_{jsi}) = Ar_i + ar_{jsi}[k], \quad A \in \mathbb{R}^{l \times l}, \quad a \in \mathbb{R}. \tag{7-7}$$

We then iteratively solve for the algebraic solution, as presented in Section 4.1.2. The corresponding procedure is summarised by Algorithm 1. Superscripts are used to denote the iteration count $i$. Numerical evidence of the convergence of these iterations without an obstacle function is presented in Figure 7-1. Empirical evidence confirms convergence for these cases. With obstacle functions, there exist cases where the iteration diverges. We suspect that low values of the obstacle function result in a higher sensitivity on the weights that is not captured by this method. We provide insights on possible improvements in recommendation Section 11.2.
Algorithm 1 Iterative method to resolve the algebraic loop of the ST for the case of Navigation Functions.

Hot start: $r_{js}^0[k] = r_{js}^{k-1}[k - 1]$

while $||r_{js}^i[k] - r_{js}^{i-1}[k]|| > \epsilon$ do

Evaluate Potentials: $(r_{js}^{i-1}[k], r_{i}[k]) \rightarrow \beta^i, \gamma^i$.

Compute Weights: $(r_{i}^{i-1}[k], r_{i}[k]) \rightarrow c_\gamma, c_\beta, j \forall j$.

Derive Gradient: $\nabla \phi_i = A_i r_i[k] + a_i r_{js}^i[k]$

Approximate: $r_{js}^i[k] = \frac{1}{1-a^i}(T_{11}s_{ij}[k] + T_{12}K_aA_i r_i[k])$

end while

Figure 7-1: Numerical validation of the iterative method for solving the algebraic loop formed between controls and the ST. The control input calculated with the NF is drawn as dashed line. The steps of the iterative method and the static point are shown with green circles and a red cross, respectively. The computed value of $r_{js}$ is shown on the y-axis. The iteration converges to the static point where the x and y axes match.

7.1.3 Impedance Matching

The combination of the ST with NFs results in an issue in the impedance matching of the network. The gain on the wave reference $r_{js}$ now incorporates the additional gain $a^i$, which results from the nonlinear gain on the inter-agent distance that is introduced by the NF. It can be shown using derivations similar to those in Section 4.1.2, that the matched line impedance is given by

$$b = a^i k_{ij}.$$  \hfill (7-8)

Noting that the gain $a^i$ is not static, we find that there exists no static solution for impedance matching. Furthermore the problem is not solvable by adapting the gain $k_{ij}$ since we require at all times $k_{ij} = k_{ji}$. The presence of wave-reflections is therefore inevitable. This is true in general for any control method that introduces a nonlinear gain on the inter-agent distance, e.g. as a result of normalisation. The effects of wave-reflections may be reduced with wave fil-
Discretisation of rPBC

The transition from continuous-time to discrete-time requires consideration of discretisation effects. Noting the three passive components of the continuous-time scheme, we propose the following discretisation:

- For scope limitation of this work we emulate the continuous-time local controllers and assume fast enough sampling for passive behaviour.
- The cooperative controls are by design robust towards time delays and are therefore emulated without risking loss of passivity.
- The network is discretised by assigning a common network sampling frequency $f_s$ between agents. Arriving waves are buffered in-between sampling instances. At each sampling instance the agent retrieves the data from the buffer to generate an input and a reflecting wave. The latter is send back over the network.

Wave-Variable Reconstruction

In the following, the network effects of time-varying delay and packet loss are considered in discrete-time. It was shown in [10] that the effect of time-varying delays and packet loss are identical: an empty wave-variable buffer at the sampling instance. An approach that resolves both issues is wave-variable reconstruction. When no data arrives during a sampling period, we replace the empty buffer data with a local reconstruction that preserves passivity. Two trivial strategies for wave-variable reconstruction are [10]:

- **Zero-Output Strategy (ZOS)** - The reconstructed wave-variable is zero. The strategy is low performance, but preserves passivity.
- **Hold-Last-Sample strategy (HLS)** - The last received wave-variable is buffered and used in the absence of data. This strategy has high performance but has no passivity guarantee.

An improved ZOS strategy is presented in [10]. In this work we apply Wave-Variable Modulation (WVM) as introduced in [11]. This reconstruction is an extension of HLS that is both high performance and passive. In fact, we implement HLS if it preserves passivity, which is validated locally. If HLS does not preserve passivity, a reconstruction is computed with...
the maximal energy that preserves passivity and the sign of the buffered sample. For completeness of this Thesis, we provide here the WVM equations from [11] for one side of the network.

\[
\hat{s}^{-}_{ij}[k] = \begin{cases} 
  s^{-}_{ij}[k - 1], & \text{if } ||s^{+}_{ij}[k]||^2 \geq ||s^{-}_{ij}[k - 1]||^2 \\
  \text{sgn}(s^{-}_{ij}[k-1])s^{+}_{ij}[k], & \text{if } ||s^{+}_{ij}[k]||^2 < ||s^{-}_{ij}[k - 1]||^2 
\end{cases} 
\]

\[
s^{-}_{ij}[k] = s^{-}_{ij}[k - 1] 
\]

Application of this reconstruction scheme ensures robustness towards time-varying delays and packet loss.

### 7.3 Summary

In this chapter we added a real-time solution for constraint satisfaction in the cooperative control function. A discretisation method for the rPBC scheme was proposed. The discretisation of the network is such that time-varying delays and packet loss can be addressed by reconstruction of wave-variables for which we selected WVM.
To show the efficacy of the proposed approach, this work includes a real-time implementation of cooperative rPBC. The experimental setup consists of a 7 Degree-Of-Freedom (DOF) robotic manipulator and multiple differential drive robots. The scheme is implemented in ROS and C++ and is designed with scalability and robustness as central requirements. In this chapter we first present the general design of the software stack, we then detail parts of the implementation that are relevant to the developed control scheme.

8.1 Description

A photo and schematic of the experimental setup are depicted in Figures 8-1a and 8-1b respectively. The experimental setup consists of a 7 Degree-Of-Freedom (DOF) robotic manipulator\(^1\) and 2 Differential Drive robots\(^2\).

---

\(^1\)Franka Emika Panda, see https://www.franka.de/technology.
The differential drive robots move in the $x, y$ plane. Their positions relative to the manipulator are required by cooperative rPBC. This measurement problem is solved with a camera marker detection system\cite{23}. The camera detects the camera orientation via the markers visible in Figure 8-1a. The markers fitted on the differential drive robots are detected and transformed to the base frame of the detection system. These positions are then translated to the common frame of reference at the lowest point of the manipulator base, indicated in Figure 8-1b.

### 8.2 Software Architecture

The software architecture is designed with scalability and robustness as central design requirements. To that end the architecture includes a number of interfaces that may be extended to include different systems, controllers or network connections. The software stack uses C++ and ROS and is made publicly available at [https://github.com/oscardegroot/ROS_rPBC](https://github.com/oscardegroot/ROS_rPBC).

Although the software in the current experimental setup runs at a single PC, the software may directly be deployed in a distributed manner. Exploiting the capabilities of the ROS framework we develop two types of nodes: server-side and client-side. An overview of functions of both node types are given below.

#### 8.2.1 Server-Side Implementation

The server side implementation ensures synchronous starting and stopping of the system and handles the distribution of system parameters, such as the formation and the assigned leaders. A block diagram of the system is depicted in Figure 8-2.

![Diagram of the server side functionalities and communication.](image)

To ensure synchronisation between all parts of the systems, the program flow follows a Finite-State Machine (FSM), that is depicted schematically in Figure 8-3.

---

\cite{23}The design and implementation of this system was the thesis work of Arjan Vermeulen.

O. de Groot Master of Science Thesis
The design registers connected agents initially. Based on the settings and the registered agents, the server generates formation distances and leader references. These are then communicated to the agents which initiate communication edges with all their specified neighbours. When this process finishes for all agents, the server starts the controllers synchronously.

### 8.2.2 Client-Side Implementation

The control of each agent is implemented as a client-side node. Since the functionalities within agents overlap, the architecture defines four interfaces. The composition of these interfaces in the control scheme is shown in Figure 8-4.

From small to large, these interfaces are:

- **Edge** - A one-sided implementation of a bidirectional communication edge to another agent. Handles reconstruction of wave-variables, the ST and the cooperative controls. Generates the cooperative input on a single edge. See Appendix C.1 and Appendix C.2.

- **CMM** - The Communication Management Module. Contains the edges to other agents. Generates the cooperative input of an agent by sampling its edges. See Appendix C.3.

- **Controller** - Implements local feedback and generates a system input from the cooperative input. See Appendix C.4.

- **System** - Implements interaction with the physical plant and provides system model matrices. Receives sensor data and sends the control input to the plant. See Appendix C.5.
The implementation directly follows from the derived theory. Given the relevance of the communication network, we provide pseudo-code for the sampling process. Algorithm 2 shows the sampling procedure of a network edge. The computations of the output waves and the network input are performed via Algorithm 1 and the ST equations (3-3), (3-4).

**Algorithm 2** Network sampling algorithm that reconstructs incoming waves when data is absent.

**input:** $r_i$

**output:** $\tau_{jsi}$

1. if $s_{in}[k]$ is empty then
   1. Compute $s_{out}[k]$ from $r_i[k], s_{in}[k-1]$.
   2. if $||s_{out}[k]||^2 > ||s_{in}[k-1]||^2$ then
      1. $s_{sample}[k] = s_{in}[k-1]$.
   3. else
      1. $s_{sample}[k] = \text{sgn}(s_{in}[k-1])|s_{out}[k]|$.
   4. end if
5. else
   6. $s_{sample}[k] = s_{in}[k]$.
   7. $s_{in}[k-1] = s_{in}[k]$.
5. end if

Compute $s_{out}[k], \tau_{jsi}[k]$ from $r_i[k], s_{sample}[k]$.

Send $s_{out}[k]$ over the network.

**return** $\tau_{jsi}[k]$

### 8.3 Implementation of PBC for Differential Drive Robots

#### 8.3.1 Feedback Linearisation and Internal Accelerations

Implementing PBC for differential drive robots requires additional steps, which we present in the following. It is well known that the dynamical equations describing differential drive systems suffer from nonholonomic constraints due to no slip constraints on either wheel. We circumvent this problem by applying the output feedback linearisation around the hand position, presented in [24]. A schematic diagram of a differential drive robot and the hand position is depicted in Figure 8-5.

\[ h \]

**Figure 8-5:** Schematic view of a differential drive robot and the hand position.
To obtain cooperative point-mass dynamics around the hand position we implement the following procedure. Starting with the control force input applied to the hand-position \( \tau_h \), apply output feedback linearisation to retrieve the physical system force inputs.

\[
\tau = \left[ \frac{1}{m} \cos \theta \ - \frac{L}{J} \sin \theta \right]^{-1} \left( \tau_h - \left[ -\frac{v \omega \sin \theta - L \omega^2 \cos \theta}{v \omega \cos \theta - L \omega^2 \sin \theta} \right] \right),
\]

where \( m, J \) are the mass and inertia respectively and \( v, \omega \) denote the translational and rotational velocity. Next we convert these force inputs to velocity inputs by applying

\[
\begin{bmatrix}
v[k] \\
\omega[k]
\end{bmatrix} = \begin{bmatrix}
v[k-1] \\
\omega[k-1]
\end{bmatrix} + \begin{bmatrix}
\frac{1}{m} & 0 \\
0 & \frac{1}{J}
\end{bmatrix} \tau[k] \cdot T_s,
\]

where the control sampling period is denoted by \( T_s \). This generates velocity setpoints by generating accelerations internally and integrating them in discrete-time. Lastly we convert the translational and rotational velocity to wheel specific rotational velocity setpoints, by applying the transformation

\[
\begin{bmatrix}
\omega_r \\
\omega_l
\end{bmatrix} = \begin{bmatrix}
\frac{1}{l} & \frac{1}{l} \\
-\frac{1}{l} & -\frac{1}{l}
\end{bmatrix} \begin{bmatrix}
v \\
\omega
\end{bmatrix},
\]

with \( l \) the wheel circumference. The discussed steps translate the force inputs at the hand position \( \tau_h \) to wheel velocity inputs \( \omega_r, \omega_l \).

### 8.3.2 Cooperative Model

The feedback linearised, differential drive robots are 2-dimensional point-masses, as defined in Section 5.6, around their hand positions. Hence the maximal cooperative dimension equals the dimension of the system, i.e., \( l = n = 2 \). This additionally implies that there are no local coordinates. We note that the matrix derivatives \( \dot{M} \) and \( \dot{\Psi} \) are both zero, simplifying the computational steps.

### 8.3.3 Odometry and Vision Measurements

The camera detection system outputs agent locations at a variable rate that may be slower than the 100 Hz sampling frequency of the Elisa3 robots. To improve the dynamic behaviour we incorporate the internal odometry. When the detected location is received, the agent coordinates are set to the received location, while the current odometry data is saved. If an agent does not receive its detected location from the vision system, then it adds the odometry data, offset by the saved odometry reference, to the last detected location. This smoothens the position measurements significantly while the vision system prevents drift of the measurements.

### 8.4 Implementation of PBC for a 7-DOF Robotic Manipulator

The manipulator is fully actuated with \( n = 7 \). We define the position of the end-effector in 3D space as cooperative coordinates such that \( z = (x, y, z) \in \mathbb{R}^3 \). The matrices \( M(q), \Psi(q) \)
required for control are obtained from the robot interface over Ethernet and gravitational forces are internally compensated. The sampled data control loop is implemented at a rate of 1 kHz. Matrix derivatives $\dot{M}, \dot{\Psi}$ are obtained by numeric differentiation and low-pass filtering.

Next to the real-time implementation of the robotic manipulator class the developed codebase contains two simulation systems of the Panda which can be tested in a Gazebo environment\(^4\). The first simulation system evaluates model matrices, which were derived symbolically in Matlab. The second simulation system retrieves model matrices directly from the Ethernet link to the physical robot.

### 8.5 Additional Cooperative Features

#### 8.5.1 Artificial Potential Functions

Different practical scenarios may require a vast amount of different constraints. As generally, these should be incorporated in the cooperative potential function, we provide a set of base classes and functions that simplify possible extensions. The codebase contains the following base classes:

- **Obstacle** - Definition of a single obstacle $\beta_j$. Defines a value and gradient function, i.e., $\beta_j(r_i, r_{js}), \nabla \beta_j(r_i, r_{js})$. The latter returns the factorisation of the gradient (7-7).

- **Goal** - Definition of an attractive potential $\gamma$. Defines a gradient and value function, i.e., $\gamma(r_{jsi}), \nabla r_{jsi}\gamma(r_{jsi})$.

- **Navigation Function** - Implementation of a Navigation Function, which holds one Goal and any number of Obstacle entities. Computes $\varphi(r_i, r_{js}), \nabla \varphi(r_i, r_{js})$. The latter returns the factorisation of the gradient (7-7).

This build-up of the potential function allows the designer to dynamically add or remove obstacles. Additionally, any type of obstacles may be defined by extending the Obstacle class. The function specific values necessary for the computation of the NF, i.e., $\beta_j, \nabla \beta_j, \gamma, \nabla \gamma$, are computed by the Obstacle and Goal class respectively. As alternative to NFs, the codebase contains a general definition for APFs as well as a specific implementation for a quadratic APF without obstacles.

#### 8.5.2 Cooperative Selectors

The number of cooperative coordinates used by a system is bounded by the maximal physical cooperative dimensions. Depending on the practical scenario where agents are deployed, the number of cooperative coordinates may vary and part of the coordinates may be assigned as dedicated leader coordinates. The software architecture was adapted to facilitate this freedom using a Selector class. These selectors have two main functions. Firstly, when retrieving the cooperative state or the Jacobian, the selector needs to select only the relevant coordinates.

---

\(^4\)The simulation was created by Corrado Pezzato, see [https://github.com/cpezzato/panda_simulation](https://github.com/cpezzato/panda_simulation).
For that purpose we construct at initialisation a selector matrix \( S \in \mathbb{R}^{l \times l_{\text{max}}} \), which allows us to compute the state and Jacobian via

\[
    z = Sz_{\text{max}}, \quad \Psi^T = \Psi_{\text{max}}S^T. \tag{8-4}
\]

An inverse conversion is necessary in case part of the coordinates are leader coordinates only. In this case we add zeros on the relevant coordinates, which may conveniently be computed using the selector matrix, as

\[
    \tau_{\text{jsi}} = S_{\text{coop}}^T \tau_{\text{coop}} + S_{\text{leader}}^T \tau_{\text{leader}}. \tag{8-5}
\]

Consider an example in the case of cooperation in \( z^T = \begin{bmatrix} x & y \end{bmatrix} \), with \( z_{\text{max}}^T = \begin{bmatrix} x & y & z \end{bmatrix} \). In this case the selector matrix and resulting selection is given by

\[
    S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{8-6}
\]

This simple implementation allows the designer to freely select coordinates based on the preferred scenario.
Part III

Experimental Results
Chapter 9

Experimental Results

In Chapter 8 the implementation of rPBC in ROS, C++ was detailed. This chapter presents experimental results obtained with the presented setup. We firstly validate the local control law in a single-agent experiment, after which we show that the developed cooperative control scheme converges and achieves formation control in the presence of time-varying delays and packet loss. We then use artificial delays and packet loss to illustrate the robustness of the scheme over a range of network conditions. Lastly we implement cooperative IDA-PBC as presented in [2] and compare it to the rPBC scheme in a setting with network effects.

9.1 Emulated Network

To illustrate performance with respect to time-varying delays and packet loss, a network is emulated artificially. We generate time-varying delays with random walk processes between a minimum and maximum delay that we set per experiment. The slope of the random walk processes is limited to 5 ms per sampling period, which corresponds to 50% of the sampling period. We additionally drop a specified percentage of packets. We emphasise that the delays are unique per edge, so that any bidirectional edge has two distinct delays. Figure 9-1 shows typical time-varying delays generated by the emulated network.

Figure 9-1: Artificial delays generated via random walk processes.
9.2 Single-Agent Validation of Local Control

Before presenting cooperative results we first highlight the effect of the local control in a single-agent scenario. More precisely we have a two agent system with the physical manipulator and a virtual beacon. The latter system is defined by a static cooperative output. We then apply cooperative consensus control. The manipulator is assigned a local quadratic potential around the central angle of its joints such that it will avoid its joint-limits. The 4 images in Figure 9-2 illustrate the interaction between the cooperative and local coordinates.

![Start position of the manipulator](image1)
![The robot has moved in the cooperative space to the beacon, forcing its joints from their central position.](image2)
![The cooperative potential function has largely converged. The local potential is high enough to overcome friction and moves the joints towards their central position.](image3)
![The local potential function is minimised. All velocities have converged and the manipulator has come to a standstill.](image4)

**Figure 9-2:** A demonstration of the interaction between the cooperative and local objectives using photos of a single-agent experiment.
9.3 Two-Dimensional Heterogeneous Formation Control

We now present experimental results in a formation control scenario. For this experiment the agent set consists of the manipulator and 2 differential drive robots. The objective in the following experiment is to form a circle in the ground plane with a radius of 0.1 m between the three agents.

Without any constraints, the height of the manipulator may vary and possibly exceed the limits of the workspace or exceed its joint-limits. Hence we define the height of the manipulator end-effector as a leader coordinate with 0.4 m as reference and 1.0 as gain. The main parameter values are summarised in Table 9-1. We share insights on how these values are selected in Appendix A. Notice that the differential drive robots have no local degrees of freedom, such that $\kappa$ applies solely to the manipulator. The local potential function of the manipulator is the joint-limit avoidance potential applied in the previous section. The transient response is sped up by setting its mass to $\eta = 0.015$. Similarly for the differential drive robots we set $\eta = 0.05$. Hence the manipulator is virtually lighter in the cooperative space than the differential drive robots. This compensates for the friction observed in the manipulator joints.

For the following experiments we set the minimum- and maximum delay to 200 ms and 400 ms, respectively. Additionally, 5% of the packets are artificially dropped. Figures 9-3 and 9-4 show the trajectories and cooperative coordinates of the agents respectively.

The specified circular formation is drawn around the centre point of the robots and is achieved by the agents. Notice that the height of the leader also converges to the reference of 0.4 m. Although the effects of the time delays are visible in slight oscillations of the trajectories, their deteriorating effect on the performance is small and the system converges. Figures 9-5

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\gamma$</th>
<th>$\kappa$</th>
<th>$f_N$</th>
<th>$K_{ij}$</th>
<th>$R_w$</th>
<th>$\epsilon$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.05</td>
<td>0.05</td>
<td>100 Hz</td>
<td>1.5$I_n$</td>
<td>30.0</td>
<td>0.0004</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Figure 9-3: Trajectories of the agents in the ground plane. The initial and final positions of the agents are denoted with a square and circle respectively. The bounds in the $x$ and $y$ axes are drawn as black solid lines.
and 9-6 show the cooperative and local velocities respectively.

The movement induced by cooperative forces causes deviation in the local coordinates. The local dynamics then try to minimise the local potential function which generates motion. Due to the damping in both the cooperative and local dynamics all velocities converge.

O. de Groot

Master of Science Thesis
9.4 Validation of Robustness to Network Effects

The theoretical results presented in this work achieve robust cooperative control in the presence of arbitrary time-varying delays and packet loss. In this section we aim to validate these results in practice. To that end we perform multiple consensus experiments with the manipulator and a single differential drive robot. In the first set of experiments, packet loss is disabled and the delays are constant and identical on all edges of the network. We then increase this constant delay in 200 ms increments. In the second set of experiments the constant delays are 200 ms on one side and 250 ms on the other side of each bidirectional edge, while packet loss is increased in increments of 10%. To compare the performance of the experiments we define a “final settling-time” metric. This metric is defined as the last time where the Euclidean distance between agents exceeds $\epsilon_m$. We note that $\epsilon$ is not identical in the two set of experiments. In the first set of experiments we set $\epsilon = 0.01$ since larger values are directly reached in all cases, hiding the actual trend in performance. In the second set of experiments we set $\epsilon = 0.02$. Lower values are not always achieved due to the decrease in networked energy. Since the system does in fact converge we choose to plot the results with this increased value. We then plot the results for the two set of experiments in a single figure, depicted in Figure 9-7.

![Figure 9-7: Robustness results of the presented method towards time delays and packet loss.](image)

From this figure we may derive the following remarks. Firstly performance for delays up to 800 ms and packet loss up to 30% is not significantly degraded. This range includes most realistic networks. In worse conditions, performance starts to degrade. A difference between the effects of delay and packet loss is visible. For more than 30% packet loss, the performance degrades rapidly. Tests with percentages above the 80% resulted in steady-state offsets. We hypothesise that the energy received by the agents in these scenarios is insufficient to overcome the friction. For packet loss of 90% and higher, velocities do not converge and it seems that passivity is lost. This is likely a result of mismatches in the modelling or sensing of either system. We remark that these conditions are hardly realistic.

The final settling time for increased delays seems to follow a linear trend. Empirical evidence indicates that convergence is obtained for arbitrary constant delays beyond the plotted results. These robustness results confirm the presented theoretical results and motivate the applicability of the proposed approach in a wide range of network conditions.
9.5 Comparison with IDA-PBC

Our last set of results compares the proposed method to cooperative IDA-PBC. Cooperative IDA-PBC, introduced in [25], is a state-of-the-art cooperative passivity-based control method that works for both fully actuated and underactuated simple mechanical systems. In the research related to this work, two performance improvements for IDA-PBC were discovered, which are also applied in rPBC. For fair comparison, we implement these improvements. Appendix B briefly formulates and motivates the improvements in the notation of [25]. We compare the trajectories of both methods in the previous consensus scenario for two sets of time-varying delays. Figures 9-8a and 9-8b show the trajectories for delays of 200 – 400 ms and 1000 – 1200 ms, respectively.

![Figure 9-8](image)

(a) Time-varying delays in the range 200 ms – 400 ms.

(b) Time-varying delays in the range 1000 ms – 1200 ms.

**Figure 9-8**: Trajectories of IDA-PBC and r-PBC in a consensus scenario under two sets of delays.

Figure 9-8a shows two distinctly different trajectories, both of which converge without overshoot in the x, y plane. In Figure 9-8b the effect of the network becomes visible. Cooperative IDA-PBC shows spiralling as a result of the delays, in contrast with the proposed method which converges directly. To validate the temporal aspect of these comparisons we show the coordinate evolution over time in Figure 9-9. This figure shows marginal, but visible improvement in the settling time in addition to the improvement in trajectories.
Figure 9-9: A comparison of the coordinate evolution between cooperative IDA-PBC and rPBC for delays in the range 1000 ms – 1200 ms.
Part IV

Conclusions and Recommendations
In this chapter, we briefly summarise the results of this work. We then reflect on the obtained results with regards to the objectives set in the introduction to conclude on the contributions of this work and remaining limitations of the methodology. The latter is discussed further in Chapter 11, where recommendations for future research are provided.

10.1 Conclusions

In this work we developed a cooperative control scheme for nonlinear fully actuated heterogeneous mechanical agents using passivity theory. By encoding the task-space coordinates and velocities into the output and sending this output as wave-variable over the network, we obtain robust synchronisation in the presence of time delays. Application of wave-variable reconstruction in discrete-time ensured passivity in the presence of time-varying delays and packet loss.

We have shown that this output scheme poses only partial requirements on the dynamical shaping of individual agents. The presented local control law exploits this by introducing local control terms, which influence the local behaviour of the system. Accordingly, the designer is provided with an intuitive set of weights to tune for appropriate local and cooperative behaviour. The set of introduced cooperative controls based on Navigation Functions incorporates constraints such that obstacles, other agents or singularities may be avoided. Moreover, it was shown that minor extension to the scheme resulted in leader-follower or leaderless formation control. The proposed continuous-time scheme was validated in simulation, where convergence was obtained for both a consensus and formation control scenario.

A real-time experimental setup of the proposed scheme in C++ and ROS was designed as part of this work. Experimental results, obtained with this setup, showed convergence in the presence of significant time-varying delays and packet loss. Further experiments related to the robustness with regards to network effects as well as a comparison to a state-of-the-art method were provided. These experiments indicated superior performance of the proposed method over existing architectures.
10.2 Discussion

We now reflect on the objectives, set in the introduction in Chapter 1.

10.2.1 Heterogeneous and Nonlinear Control

The developed control scheme is designed for nonlinear, heterogeneous systems. Specifically, we have shown in Chapter 4 that we may shape heterogeneous systems for homogeneous cooperative dynamics. As such systems become identical only in the cooperative coordinates, which simplifies cooperative tuning significantly. We have additionally derived a result for heterogeneous cooperative shaping, which gives the designer full control over the cooperative dynamics.

10.2.2 Fully Actuated and Underactuated Systems

The dynamical matching procedure was developed strictly for fully actuated mechanical systems. Hence the obtained local control results cannot be applied to underactuated systems in its current form. Since the class of underactuated systems is practically relevant, we present in Section 11.1 an approach for extending rPBC towards underactuated systems. This methodology is based on the well-known IDA-PBC matching procedure for underactuated systems.

10.2.3 Robustness towards Network Effects

The developed passivity-based control approach is by design robust towards arbitrary constant delays. Passivity theory and the ST are the key ingredients that facilitated these results. We included existing wave-variable reconstruction in the developed approach to achieve discrete-time robustness towards time-varying delays and packet loss. The results in Chapter 9 validated robustness towards arbitrary constant delays in practice and showed robustness to packet loss up to 80% in the presented scenario. Additionally, the performance was shown to be consistent in the range of realistic network conditions. We therefore conclude that we solved the problem of network effects in cooperative control for almost all realistic applications.

10.2.4 Collision Avoidance

The problem of collision avoidance was approached from a potential-based control perspective. Specifically, we applied Navigation Functions in the presented context and solved the algebraic loop that these controls form with the ST. We note that with obstacles, the solutions to the algebraic loop are not always found, which in some cases leads to failure. In Section 11.2 we detail possible solutions to this problem. Additionally, a method for hard constraints is suggested based on projection of the cooperative input.
10.2 Discussion

10.2.5 Concluding Remarks

We have introduced a cooperative passivity-based control method in this work that obtains robust control of nonlinear heterogeneous mechanical systems in the presence of network effects. To the author’s knowledge, the properties and capabilities of the presented scheme exceed those of related existing works. The application scope of the method is large: It includes a vast amount of practically relevant systems and guarantees performance and stability in a wide variety of practical scenarios. Additionally, we have shown the efficacy of the approach in practice, as opposed to typical simulation based validations found in literature. The largest limitation to the scheme at present is the lack of support for underactuated systems, which is dedicated to future research.
To conclude this work, we present recommendations for future work. These recommendations include direct extension of the presented work as well as an overview of related topics.

11.1 Extension to Underactuated Systems

The results in this work strictly apply to fully actuated mechanical systems. However, there are a number of practically relevant systems that are not included in this set of systems. A notable example is the quadrotor UAV, which could have interesting applications for package delivery or other logistic purposes. For the purpose of extending rPBC to underactuated systems, we lay-out an approach based on IDA-PBC. As discussed in Chapter 2, the input matrix $F \in \mathbb{R}^{n \times m}$ for underactuated systems, is not square or invertible, as $m < n$. However similar to the IDA-PBC related work in [2] it may be possible to match dynamics, in our case to r-passive cooperative dynamics. In Theorem 5.3, equation (5-20) outlines the dynamics that are matched with the plant dynamics. Applying an IDA-PBC approach, we may write the following expression that matches the systems,

$$\dot{p} = M(\Psi)^T \tau_c - M(\Psi^\perp)^\dagger \tau_l - \frac{\partial H}{\partial q} + F \tau.$$ \hspace{1cm} (11-1)

Notably, $F$ has no left pseudo-inverse and as such this equation cannot directly be solved for an input. The key insight applied in IDA-PBC is that the transformation $\begin{bmatrix} F^T \\ F^\perp \end{bmatrix}$ is full rank, such that we obtain two sets of equations. The first $m$ equations give an expression for the control law, i.e.,

$$\tau = F^\dagger \left( M(\Psi)^T \tau_c - M(\Psi^\perp)^\dagger \tau_l + \frac{\partial H}{\partial q} \right).$$ \hspace{1cm} (11-2)

The last $n - m$ equations pose conditions which must hold for matching to hold and are therefore called the matching equations. These are given by

$$F^\perp \left( M(\Psi)^T \tau_c - M(\Psi^\perp)^\dagger \tau_l \right) = 0.$$ \hspace{1cm} (11-3)
Solving this last equation may require additional terms to overcome the plant dynamics \( \frac{\partial H}{\partial q} \) lying in the underactuated manifold. The reader is referred to [2] and the references therein for an in-depth discussion on this topic. A notable reference is [26], which presents an algebraic solution to this problem in IDA-PBC for underactuated systems of degree one.

11.2 Collision Avoidance

Although this work presents a number of results on collision avoidance, the problem of collision avoidance is not fully solved. The iterative computation of the static point is not guaranteed to converge and as such reduces the robustness of the presented methodology. The following two points provide possible improvements.

11.2.1 Static Point Computations

We remark that improvements exist within the presented potential-based control framework. The main issue of the current method is divergence when the obstacle function \( \beta \) is small. The presented solutions for finding the static point of the algebraic loop are possibly too simple. An alternative approach towards finding the static point is real-time optimisation, which is applicable if the problem is computationally feasible. The LUT strategy could be extended to included obstacles, if the increased dimensionality of the problem can be overcome. Lastly, the Navigation Function based cooperative control function could be replaced with another potential function that has more suitable properties with regards to the resulting algebraic loop, if such a function exists.

11.2.2 Passive Projection of Constraints on the Cooperative Input

An alternative method that incorporates hard constraints is a projection based method. This method, inspired by the singularity avoidance scheme in [22], projects the input vector such that the system cannot violate a constraint. An illustration of the method is provided in Figure 11-1.

![Figure 11-1: An illustration of the projection process. The position of two agents is indicated as well as a singular point for agent \( i \), denoted \( z_{s,i} \). The input vector is projected such that a margin towards the singular point is maintained.](image)
This figure presents a scenario with agents $i$ and $j$. Agent $i$ has a kinematic singularity at location $z_{s,i}$, denoted in the figure. Following [22] we may compute a marginal area around this point which the agent should avoid. When the cooperative input pulls the agent towards the singularity, we project the input such that it lies perpendicular to the boundary of the constrained area. As such the input pushes the system to an evasive trajectory. This strategy was applied successfully in simulation to avoid singularities, out of necessity. However, as this projection becomes part of the cooperative PBC scheme it is essential to establish if this projection is passive, or generally, which projections of the input vector are passive. If such a projection is in fact passive, then the concept of projection could extend beyond singularity avoidance to for example obstacle avoidance. If these projections are generally passive then we may integrate collision avoidance schemes, such as Optimal Reciprocal Collision Avoidance (ORCA), on a higher level of control, which map their outputs onto the lower-level rPBC controls. Such a strategy would merge the advantages of the existing state-of-the-art collision avoidance schemes with the robustness properties of cooperative rPBC.

11.3 Cooperative Manipulation

In many cooperative tasks, multiple agents attempt to manipulate an object in cooperation. Such tasks are valuable in many contexts, an example of which is the Aerial Towing Problem (ATP), where multiple UAVs lift a single object. Another example is cooperative manipulation using multiple manipulators which has applications in industrial automation or search-and-rescue operations. When multiple agents grasp an object cooperatively, their dynamics become constrained together. Different control schemes could be applied in this situation. The system dynamics could be joined together into a single system description, which is then controlled centrally. A computationally more interesting approach is to maintain a distributed control architecture, while introducing constraints into the individual systems. Each system is extended with the constraints

\[ A^T \frac{\partial H}{\partial q} = 0. \]  

(11-4)

It is shown in [27] that definition of the annihilator $S$, for which

\[ A^T S = 0, \]  

(11-5)

allows a coordinate transformation to the constrained system manifold. For fully actuated systems and for suitable constraints we may have $S^T F$ invertible. In these cases the proposed results in this work may apply to the constrained framework. The definition of the outputs $r$ could pose a problem, since velocities may be of less interest when constraints on the coordinates are present.

11.4 High-level Control with Low-Level rPBC

The method proposed in this work provides a robust and reliably performing control scheme for cooperative control. Any two mechanical systems that implement this scheme may be interconnected to achieve cooperative objectives. Integration of this scheme as low-level control
with higher level planning methods could therefore lead to novel applications that go beyond the formation of configurations. Additionally, distributed implementation of rPBC could give more insight in the capabilities of the scheme with regards to scalability and applicability.

11.5 Model Predictive Control for r-Passivity

The control strategy developed in Chapter 5 is one solution to render an agent cooperatively r-passive. An alternative solution that may be simpler or more widely applicable is Model Predictive Control (MPC). We may express cooperative r-passivity as a hard constraint. Given the presented solutions in this work, a feasible solution should always exist. The benefit of this approach is that the designer may incorporate additional hard constraints within the framework while transient performance may be tuned via the cost function. It is however not certain that the structure of this optimisation problem is computationally feasible. The general nonlinear dynamics of the problem likely result in a non-convex optimisation with non-convex constraints.

A middle-of-the-road alternative to this approach is to implement a partial preliminary feedback term for r-passivity as designed in this work, while controlling the local coordinates via an optimisation based strategy.

11.6 Application to Other Fields of Research

The robustness properties obtained by the presented scheme possibly suit applications in other research areas. An example may be that of lateral platooning. If it is possible to shape the individual vehicles as trailers with the first vehicle as leader than the robustness with respect to time delays and modelling errors and the inclusion of heterogeneous systems may make this technique applicable in such situations.
Appendix A

Notes on Tuning rPBC

The division between cooperative and local coordinates simplifies tuning of rPBC. We can tune the local dynamics for suitable single-agent behaviour. Afterwards the cooperative dynamics may be tuned based on high performance for a specific scenario or robust performance in general scenarios. We now share empirically obtained insights on the tuning process. This guide is provided as an aid for the interested reader.

**Local Tuning**  We first define the local dynamics by selecting a local potential function that includes all necessary constraints. Examples of such constraints are local tracking on a coordinate (e.g., the rotation of the end-effector) or the previously applied joint-limit avoidance function. The only other local parameter is the damping coefficient $\kappa$. We note that the dynamic effect of $\kappa$ is similar to that of viscous friction. High values may in practice cause drift due to velocity penalisation. We initialise $\kappa$ with a low value and then increase it until the system velocities are sufficiently limited while no drift is observed.

**Workspace Selection**  Based on the application of the agent we select a workspace radius. Noting that the output encodes both velocities and coordinates we ideally configure a bound on both and set the workspace radius to the corresponding output. For some applications, such as drone flight, the system may move continuously. In this case a more suitable selection of the workspace is the visibility range of the agent.

**Network Gains**  The effect of $\lambda$ on the performance is significant. In this work we set $\lambda = 1.0$, after some initial experiments with other values. We do not claim that this is the most suitable value in all cases. Given this value of $\lambda$, an initial value of $\gamma = 1.0$ and $K_{ij} = I$, the performance of the system can be tuned empirically. The transient response of all connected systems can directly be modified through $K_{ij}$, especially if kept consistent over all edges in the network. The value of $\epsilon$ in the normalised NF may be reduced until no drift is observed. The value of $\gamma$ is most effective in preventing movement of the consensus point. Sensor noise may cause a system to leave the equilibrium pulling with it the other systems. If
this happens frequently, than the cooperative system drifts. Higher values of $\gamma$ punish small movements, preventing the drift. Alternatively $\gamma$ may be used to penalise fast cooperative behaviour, although this may also be influenced via the virtual cooperative masses $\eta_i$. This last tuning parameter is especially effective in tuning behaviour with different system types. In the performed experiments the mass of the manipulator is lowered which greatly increases its contribution to cooperative consensus.
In the implementation of IDA-PBC and rPBC, two performance improvements for IDA-PBC were discovered. We briefly detail these in the following sections.

### B.1 Pseudo-Inverse Approach

The derivations in [2, Section 3.5.2] lead to the definition of the cooperative input matrix $\Psi$. The results in [2, Chapter 9] suit the applications of this thesis, where the cooperative input is regarded as an external signal. [2, (209)] poses the condition on which the matching conditions are not influenced by this external input, which is

$$F^\perp_F d = 0.$$  \hfill (B-1)

The presented solution is to set $F_d = M_d M^{-1} \Psi$. This is equivalent to a transposed Jacobian approach. Noting that

$$(\Psi^T)^\dagger = \Psi \left(\Psi^T \Psi\right)^{-1},$$  \hfill (B-2)

we can set $F_d = M_d M^{-1} (\Psi^T)^\dagger$, which satisfies (B-1) and corresponds to a pseudo-inverse method. It is well documented that performance of pseudo-inverse methods exceeds that of transposed Jacobian methods (see e.g. [28]). This performance increase is also observed empirically. We note that this does not rigorously apply to the composed matrix framework in [2], since the input matrix $\Psi$ is strictly the result of a partial derivative ([2, (56)]).

### B.2 Normalised Goal Functions

The potential-based control framework of [2] admits application of alternate potential functions. In practice we observe significant steady-state errors as a result of friction. As solution we apply a NF with the goal function as presented in [14]. Suitable tuning results in small steady-state errors and significant performance improvements.
Selected C++ Implementations

This appendix contains selected classes from the C++ implementation specifically discussed in this work. Irrelevant parts of the files are removed for clarity. The full software stack is available at https://github.com/oscardegroot/ROS_rPBC.
C.1 Edge Interface

```cpp
/**
 * @file: Edge.h
 * @brief Interface for the local implementation of a bidirectional communication edge.
 */

class Edge {
public:
  // Constructor and destructor
  Edge (Agent& agent, int j, Eigen::MatrixXd gain_set, int l_set, Eigen::VectorXd r_star_set, int rate_mp_set);
  ~Edge();

  // Receives and sends waves
  virtual void waveCallback (const panda::Waves::ConstPtr& msg);

  // Publishes a wave
  virtual void publishWave (const Eigen::VectorXd& r_i);

  // Initialises the communication channels with ROS
  virtual void initChannels();

  // Samples this edge with the agent output
  virtual Eigen::VectorXd sample (const Eigen::VectorXd& r_i);

  // Reconstructs incoming waves
  virtual void applyReconstruction (Eigen::VectorXd& wave_reference, const Eigen::VectorXd& r_i);

  // Computes the control input
  virtual Eigen::VectorXd calculateControls (const Eigen::VectorXd& s_in, const Eigen::VectorXd& r_i) = 0;

  // Computes the reflecting wave
  virtual Eigen::VectorXd calculateWaves (const Eigen::VectorXd& s_in, const Eigen::VectorXd& r_i) = 0;

  // Leader boolean, set to true for a leader edge
  virtual bool isLeader () const { return false;};

protected:

  // IDs, cooperative dimension, gains and formation variables
  int i_ID, j_ID, l;
  Eigen::MatrixXd gain;
  Eigen::VectorXd r_star;

  // Frequency mismatch counter
  int rate_mp;
  std::unique_ptr< helpers::Counter > publish_counter;

  // Timestamps for rejection
  unsigned int timestamp = 0;
  unsigned int received_timestamp = 0;

  // Data received flag
  bool data_received = false;

  // The Ros node handle
  ros::NodeHandle nh;

  // Define a publisher and subscriber for the waves
  ros::Publisher wave_pub;
  ros::Subscriber wave_sub;

  // WVM buffer, received buffer, actual used sample
  Eigen::VectorXd s_wvm_buffer, s_received, s_sample;

};
```
C.2 Edge Implementation with NF and WVM

/*
 * Implements communication on an edge on one side using ST and WVM.
 * Externally calling sampleEdge will sample the buffer and return a wave to the other side of the channel.
 */

#include "EdgeFlex.h"

EdgeFlex::EdgeFlex(Agent& agent, int j, Eigen::MatrixXd gain_set, int l_set, Eigen::VectorXd r_star_set, int rate_mp_set)
    : Edge(agent, j, gain_set, l_set, r_star_set, rate_mp_set)
{
    // Create a potential function
    potential = std::make_unique<NavigationFunction>(agent, l, r_star);

    // Counter that handles sampling rate converting
    retrieval_counter = std::make_unique<helpers::Counter>(rate_mp, false);

    // Set the scattering gains
    setScatteringGain(gain_set);

    // Initialize r_js at some non-zero point
    r_js_last = Eigen::VectorXd::Zero(l);
    r_js_last << 0.1, 0.1, 0.1;
}

/* Main public function that samples this edge */
Eigen::VectorXd EdgeFlex::sample(const Eigen::VectorXd& r_i)
{
    Eigen::VectorXd tau(l), r_js(l), s_out(l);
    bool hls_applied = false;
    bool retrieval_triggered = false;

    // If we can sample the network at this local sampling instance
    if(retrieval_counter->trigger())
    {
        retrieval_triggered = true;

        // If no data was received, reconstruct.
        if(! data_received)
        {
            s_out = fullSTLoop(tau, r_js, r_i, s_wvm_buffer);

            // If we can apply HLS
            if(s_out.transpose()*s_out > s_wvm_buffer.transpose()*s_wvm_buffer)
            {
                hls_applied = true;
                s_sample = s_wvm_buffer;
            }
            else
            {
                s_sample = elementSign(s_wvm_buffer).cwiseProduct(s_out.cwiseAbs());
            }
        }
    }
    // Save Data
    else
    {
        s_wvm_buffer = s_received;
        s_sample = s_received;
        data_received = false;
    }

    // Find the wave reference, input and outgoing wave
    s_out = fullSTLoop(tau, r_js, r_i, s_sample);

    // Save the correct values for the next sampling period
    r_js_last = r_js;

    // Lastly publish the waves
    publishWave(s_out);
    return tau;
}

/* Calculates all variables in the ST loop with input s_in. Returns s_out */
Eigen::VectorXd EdgeFlex::fullSTLoop(Eigen::VectorXd tau, Eigen::VectorXd r_js, const Eigen::VectorXd r_i, const Eigen::VectorXd s_in)
{
    // Solves the algebraic loop of the ST
    STIterations(r_js, r_i, s_in);

    // Calculate the new controls with r_js
    tau = calculateControls(r_i, r_js);

    // Determine the output waves
    return calculateWaves(tau, r_js);
}
/* Solves the algebraic loop of the ST */

```cpp
void EdgeFlex::STIterations(Eigen::VectorXd& r_js, const Eigen::VectorXd& r_i, const Eigen::VectorXd& s_in)
{
  // For calculating the improvement over the loop
  Eigen::VectorXd r_js_prev(l);
  r_js_prev = r_js_last;
  // Initialize with the final value of the last sampling period
  r_js = r_js_last;
  // Keep track of iteration count
  int k = 0;
  double diff = 10.0;

  /* Loop until the improvement is marginal */
  while (diff > 0.001 && k < 100)
  {
    // Calculate the gradient of the potential function
    PotentialFactors gradient = potential->gradient_factors(r_i, r_js);
    // Calculate r*[k] for all dimensions
    for (int i = 0; i < l; i++)
    {
      // Retrieve the wave impedance
      double b = impedance(i);
      // Compute the LHS and RHS of the approximation
      double RHS = agent_i * std::sqrt(2.0/b)*s_in(i) + 1.0/b * gain(i, i) * gradient.RHS_vector(i);
      double LHS = 1.0 - 1.0/b * gain(i, i) * gradient.LHS_multiplier;
      // Approximate
      r_js(i) = RHS / LHS;
    }
    // Calculate the improvement
    diff = helpers::normOf(r_js - r_js_prev);
    // Remember the last value of r_js
    r_js_prev = r_js;
    k ++;
  }

  // Detect divergent iterations
  if (k > 90)
  {
    logMsg("EdgeFlex", "Warning: iteration count k > 90, the iteration may not converge!", WARNING);
    r_js = r_i;
    PotentialFactors gradient = potential->gradient_factors(r_i, r_js);
  }
}
```

// Compute the control input
```cpp
Eigen::VectorXd EdgeFlex::calculateControls(const Eigen::VectorXd& r_i, const Eigen::VectorXd& r_js)
{
  // Multiply the gradient with the gain
  return -gain * potential->gradient(r_i, r_js);
}
```

// Retrieve the output wave from the scattering transformation
```cpp
Eigen::VectorXd EdgeFlex::calculateWaves(const Eigen::VectorXd& tau, const Eigen::VectorXd& r_js)
{
  Eigen::VectorXd result = Eigen::VectorXd::Zero(l);
  for (int i = 0; i < l; i++)
  {
    double b = impedance(i);
    result(i, 0) = agent_i / std::sqrt(2*b) * (tau(i, 0) - b*r_js(i, 0));
  }
  return result;
}
```

// Set the impedance
```cpp
void EdgeFlex::setScatteringGain(const Eigen::MatrixXd& gain)
{
  // Define the gains on this edge and the impedance
  Eigen::VectorXd impedance = Eigen::VectorXd::Zero(l);
  for (int i = 0; i < l; i++)
  {
    impedance(i) = std::sqrt(gain(i, i));
  }
  // Take into account the different ST depending on which agent this is
  agent_i = (i_ID < j_ID) ? -1 : 1;
}
```
C.3 CMM Interface

```cpp
/**
 * @file: CMM.h
 * @brief Manages cooperative communication by managing connections to other agents
 */

// FSM Status
enum Status {
    STARTED = 0,
    WAITING_FOR_OTHER, // Temporary status
    INIT_CMM,
    RUNNING
};

class CMM{
public:
    CMM(std::string agent_type);
    ~CMM();

    // FSM status of the CMM
    Status status;

    // The contained agent
    std::unique_ptr<Agent> agent;

    // Sample edges based on the agent output
    Eigen::VectorXd sample(const Eigen::VectorXd& r_i);

    // Wait for the server message to retrieve the network
    void performHandshake();

    // Initialise the edges of this agent
    bool initEdges(std_srvs::Empty::Request& req, std_srvs::Empty::Response& res);
    bool retrieveConnections();
    void setupLeader();
    void setupEdges();

    // Returns if agent has initialised its network
    bool hasInitialised() const;

    // Getters for cooperative dimensions
    int coopDim() const { return coop_selector->dim(); };
    int leaderDim() const { return leader_selector->dim(); };
    int allDim() const { return coop_selector->dim() + leader_selector->dim(); };

private:
    // Cooperative dimensions
    int l, N;
    // Network gain
    Eigen::MatrixXd gain;
    // Sampling rate ratio
    int rate_mp;
    // Initialisation flag
    bool initialised = false;
    // ROS servers
    ros::NodeHandle nh;
    ros::ServiceClient connect_client, leader_client;
    // Edges of this agent
    std::vector<std::unique_ptr<Edge>> edges;
    // Selectors for cooperative and leader inputs
    std::unique_ptr<Selector> coop_selector, leader_selector;
};
```
C.4 Controller Interface

```cpp
/**
 * @file Controller.h
 *
 * @brief An interface for controllers using IDAPBC or rPBC control.
 */

class Controller {
public:
    Controller(Agent& agent);
    virtual ~Controller();

    /** @brief Get the agent output, possibly modified */
    virtual Eigen::VectorXd getOutput(System& system) = 0;

    /** @brief Compute the input */
    virtual Eigen::VectorXd computeControl(System& system, const Eigen::VectorXd& tau_c) = 0;

    /** @brief Local control potential */
    virtual Eigen::VectorXd dVsdq(System& system) = 0;

    /** @brief Damping matrix */
    virtual Eigen::MatrixXd Kv(System& system);

protected:
    int l;
    ros::NodeHandle nh;

    // Gains
    Eigen::VectorXd Vs_gains, theta_star, limit_avoidance_gains, limits_avg, limits_min, limits_max;
    bool local_enabled, limit_avoidance_enabled, gravity_enabled;
};
```

O. de Groot Master of Science Thesis
C.5 System Interface

```cpp
/** @file: System.h
 * Defines an interface for nonlinear control of systems
 * Defines matrices modelled or retrieved for control computation and includes functions to read sensors and
 * actuate
 */

/** Saves relevant state information
 * @variable q Coordinates / State variables of the system
 * @variable dq Derivatives of the above (mechanically: velocities)
 * @variable z End-effector coordinates */
struct State{
    Eigen::VectorXd q;
    Eigen::VectorXd dq;
    Eigen::VectorXd z;
};

class System{
public:
    System (int n_set , int m_set , int lmax_set , const std::string& name);
    virtual ~System ();
    int n = 0;
    int m = 0;
    int s = 0;
    int lmax = 0;
    State state ;

    // Communication Management Module, handles cooperative communication
    std::unique_ptr<CMM> cmm ;

    /** @brief Systems may not always want to cooperate in all the DOF of their end-effectors.
     * The selector matrices multiply with the z and psi matrices such that only the selected DOF remain */
    Eigen::MatrixXd selectPsi (const Eigen::MatrixXd& psi );
    Eigen::VectorXd selectZ (const Eigen::VectorXd& z);

    // Matrices are saved in this class on the heap and updated only once per sampling instance
    // functions return references
    virtual Eigen::MatrixXd& M() = 0;
    virtual Eigen::VectorXd& dVdq () = 0;
    virtual Eigen::MatrixXd& Psi () = 0;
    virtual Eigen::MatrixXd& dM () = 0;
    virtual Eigen::MatrixXd& dMinv () = 0;
    virtual Eigen::MatrixXd& dPsi () = 0;
    virtual Eigen::VectorXd& C() = 0;

    void setState (Eigen::VectorXd new_q , Eigen::VectorXd new_qd , Eigen::VectorXd new_z );
    void resetUpdatedFlags ();
    virtual void checkSafety () ;

    bool sendInput (const Eigen::VectorXd & tau ) = 0;
};
```

Master of Science Thesis
O. de Groot
void readSensors();

/**
 * @brief Enable this system, starting the controller
 */
bool enableSystem(std_srvs::Empty::Request &req, std_srvs::Empty::Response &res);

/**
** Move back to protect and move functions from control to system */
Eigen::MatrixXd approximateDerivative(const Eigen::MatrixXd &current, const Eigen::MatrixXd &previous)

protected:

// Selector for Psi and z
std::unique_ptr<Selector> selector;

// System matrices
Eigen::MatrixXd m_m, psi, dm, dpsi, dminv;
Eigen::VectorXd dvdq, c_m;
Eigen::MatrixXd psi_previous, m_previous;

// Enable server and flag
ros::ServiceServer enable_server;
bool is_enabled = false;

// Update parameters
bool m_updated, dm_updated, psi_updated, dpsi_updated, dvdq_updated, dminv_updated;
Bibliography


